centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

A LOS ASISTENTES A LOS CURSOS DEL CENTRO DE EDUCACION CONTINUA

Las autoridades de la Facultad de Ingeniería, por conducto del Jefe del Centro de Educación Continua, otorgan una constancia de asistencia a quienes cumplan con los requisitos establecidos para cada curso. Las personas que deseen que aparezca su título profesional precediendo a su nombre en la constancia, deberán entregar copia del mismo o de su cédula a más tardar el SEGUNDO DIA de clases, en las oficinas del Centro con la señorita encargada de inscripciones.

El control de asistencia se llevará a cabo a través de la persona encargada de entregar las notas del curso. Las inasistencias serán computadas por las autoridades del Centro, con el fin de entregarle constancia solamente a los alumnos que tengan un mínimo del 80% de asistencia.

Se recomienda a los asistentes participar activamente con sus ideas y experiencias, pues los cursos que ofrece el Centro están planeados para que los profesores expongan una tesis, pero sobre todo, para que coordinen las opiniones de todos los interesados constituyendo verdaderos seminarios.

Es muy importante que todos los asistentes llenen y entregen su hoja de inscripción al inicio del curso. Las personas comisionadas por alguna institución deberán pasar a inscribirse en las oficinas del Centro en la misma forma que los demás asistentes, entregando el oficio respectivo.

Con objeto de mejorar los servicios que el Centro de Educación Continua ofrece, al final del curso se hará una evaluación a tráves de un cuestionario diseñado para emitir juicios anónimos por parte de los asistentes.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
FACULTAD DE INGENIERIA
DIVISION DE ESTUDIOS SUPERIORES
CENTRO DE EDUCACION CONTINUA
DIRECTORIO GENERAL

REGISTRO DE ASISTENTES Y PROFESORES.

NOMBRE DEL CURSO: ______________________ FOLIO □□□□□ 1 5 CLAVE ASOC □□ 6 7

□□□□□□ 8 13 □□□□□□□□□□□□□□□□□□□□□□□□□□□□ 14 NOMBRE(S) APELLIDO PATERNO APELLIDO MATERNO 41

REG. FED. CAUS. □□□□□□□□□□ 42 51 CED. PROF. □□□□□□□ 52 58

TEL. PARTICULAR □□□□□□□ 59 65 TEL. OFICINA □□□□□□□ 66 72 EXTENSION □□□□ 73 76

MARQUE CON UNA CRUZ

ASISTENTE □ PROFESOR □ □ 77 1 80

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 8 DOMICILIO PARTICULAR (CALLE. NUMERO Y No. INTERIOR) 41

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 42 COLONIA 71 Z.P. □□ 72 73

______________________ ESTADO □□ 74 75

______________________ TITULO PROFESIONAL □□ 76 77 ______________________ ESPECIALIDAD □□ 78 79 2 80

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 8 DOMICILIO DE OFICINA (CALLE, NUMERO Y No. INTERIOR) 41

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 42 COLONIA 71 Z.P. □□ 7273

______________________ ESTADO □□ 74 75 3 80

ˉOCIACIONES A LAS QUE PERTENECE.

______________________ □□ ______________________ □□

______________________ □□ ______________________ □□

```
C     PROGRAMA PARA OBTENER LA RUTA CRITICA DE UN CONJUNTO DE ACTIVIDA- 19RUC001
C     DES.                                                              19RUC002
C     EL SIGNIFICADO DE LAS VARIABLES USADAS ES                         19RUC003
C     N=CANTIDAD DE ACTIVIDADES, G=MATRIZ DE SUBORDINACION DE ACTIVIDA- 19RUC004
C     DES, GB=MATRIZ PARA OBTENER LA RELACION AL RECORRER LA RED EN SEN-19RUC005
C     TIPO INVERSO, D=DURACION DE LAS ACTIVIDADES, CS=CONTADOR, EST=FE- 19RUC006
C     CHA MAS TEMPRANA DE INICIO, EFT=FECHA MAS TEMPRANA DE TERMINO,    19RUC007
C     LST=ULTIMA FECHA DE INICIO, LFT=FECHA DE TERMINO MAS TARDIA, NCON=19RUC008
C     CONTADOR, FF=TIEMPO LIBRE FLOTANTE, TF=TIEMPO FLOTANTE TOTAL, RC= 19RUC009
C     VARIABLE EN LA QUE SE ARCHIVAN LAS ACTIVIDADES QUE PERTENECEN A LA19RUC010
C     RUTA CRITICA                                                      19RUC011
      INTEGER G( 20, 20),GB( 20, 20),D( 20),EST( 20),EFT( 20),LST( 20),L19RUC012
     1FT( 20),FF( 20),TF( 20),CS( 20),L( 20),RC( 20),IDASH( 20)         19RUC013
      CALL IOCS1(2,5)
      CALL IOCS1(5,6)
      DO 1 I=1,100                                                      19RUC014
    1 IDASH(I)=1H-                                                      19RUC015
C     LECTURA DEL NUMERO DE ACTIVIDADES,SUBORDINACION ENTRE ELLAS Y DURA19RUC016
C     CION DE LAS MISMAS                                                19RUC017
    2 READ(5,100) N                                                     19RUC018
      IF(N) 3,3,4                                                       19RUC019
    3 CALL EXIT                                                         19RUC020
    4 NM1=N-1                                                           19RUC021
      DO 5 I=1,N                                                        19RUC022
    5 READ(5,150) (G(I,J),J=1,N)                                        19RUC023
      READ(5,160) (D(I),I=1,N)                                          19RUC024
C     IMPRESION DE LA MATRIZ DE SUBORDINACION                           19RUC025
      WRITE(6,200)                                                      19RUC026
      DO 6 I=1,N                                                        19RUC027
      WRITE(6,240) I                                                    19RUC028
    6 WRITE(6,250) (G(I,J),J=1,N)                                       19RUC029
      DO 9 I=1,N                                                        19RUC030
      CS(I)=0                                                           19RUC031
      DO 8 J=1,N                                                        19RUC032
      GB(J,I)=0                                                         19RUC033
      IF(G(I,J)) 7,8,7                                                  19RUC034
    7 GB(J,I)=1                                                         19RUC035
    8 CONTINUE                                                          19RUC036
    9 CONTINUE                                                          19RUC037
C     OPTENCION DE EST Y EFT                                            19RUC038
      EST(1)=0                                                          19RUC039
      EFT(1)=0                                                          19RUC040
      LST(1)=0                                                          19RUC041
      LFT(1)=0                                                          19RUC042
      CS(1)=1                                                           19RUC043
   10 NCON=0                                                            19RUC044
      DO 14 J=2,N                                                       19RUC045
      IF(CS(J).EQ.1) GO TO 14                                           19RUC046
      NUM=0                                                             19RUC047
      DO 12 I=1,N                                                       19RUC048
      IF(I.EQ.J) GO TO 12                                               19RUC049
      IF(G(J,I).EQ.0) GO TO 12                                          19RUC050
      IF(CS(I).FQ.1) GO TO 11                                           19RUC051
      NCON=NCON + 1                                                     19RUC052
      GO TO 14                                                          19RUC053
   11 NUM=NUM + 1                                                       19RUC054
      L(NUM)=I                                                          19RUC055
   12 CONTINUE                                                          19RUC056
      MAX=EFT(L(1))                                                     19RUC057
      DO 13 I=1,NUM                                                     19RUC058
      IF(MAX.GE.EFT(L(I))) GO TO 13                                     19RUC059
      MAX=EFT(L(I))                                                     19RUC060
   13 CONTINUE                                                          19RUC061
      EST(J)=MAX                                                        19RUC062
      EFT(J)=MAX + D(J)                                                 19RUC063
      CS(J)=1                                                           19RUC064
   14 CONTINUE                                                          19RUC065
      IF(NCON.NE.0) GO TO 10                                            19RUC066
C     OBTENCION DE LST Y LFT                                            19RUC067
      LFT(N)=EFT(N)                                                     19RUC068
      LST(N)=EST(N)                                                     19RUC069
      FF(N)=0                                                           19RUC070
      TF(N)=0                                                           19RUC071
      RC(N)="R.C."                                                      19RUC072
      DO 15 I=2,NM1                                                     19RUC073
   15 CS(I)=0                                                           19RUC074
   16 NCON=0                                                            19RUC075
      DO 20 I=2,NM1                                                     19RUC076
      II=N+1-I                                                          19RUC077
      IF(CS(II).EQ.1) GO TO 20                                          19RUC078
      NUM=0                                                             19RUC079
      DO 18 J=1.N                                                       19RUC080
      IF(II.EQ.J) GO TO 18                                              19RUC081
      IF(GB(II,J).EQ.0) GO TO 18                                        19RUC082
      IF(CS(J).EQ.1) GO TO 17                                           19RUC083
      NCON=NCON + 1                                                     19RUC084
      GO TO 20                                                          19RUC085
   17 NUM=NUM + 1                                                       19RUC086
      L(NUM)=J                                                          19RUC087
   18 CONTINUE                                                          19RUC088
      MIN=LST(L(1))                                                     19RUC089
      DO 19 J=1,NUM                                                     19RUC090
      IF(MIN.LE.LST(L(J))) GO TO 19                                     19RUC091
      MIN=LST(L(J))                                                     19RUC092
   19 CONTINUE                                                          19RUC093
      LFT(II)=MIN                                                       19RUC094
      LST(II)=MIN-D(II)                                                 19RUC095
      CS(II)=1                                                          19RUC096
   20 CONTINUE                                                          19RUC097
      IF(NCON.NE.0) GO TO 16                                            19RUC098
C     OBTENCION DE FF , TF Y ACTIVIDADES DE LA RUTA CRITICA             19RUC099
      DO 25 J=1,NM1                                                     19RUC100
      NUM=0                                                             19RUC101
      DO 21 I=1.N                                                       19RUC102
      IF(I.EQ.J) GO TO 21                                               19RUC103
      IF(GB(J,I).EQ.0) GO TO 21                                         19RUC104
      NUM=NUM+1                                                         19RUC105
      L(NUM)=I                                                          19RUC106
   21 CONTINUE                                                          19RUC107
      MIN=EST(L(1))                                                     19RUC108
      DO 22 I=1,NUM                                                     19RUC109
      IF(MIN.LE.EST(L(I))) GO TO 22                                     19RUC110
      MIN=EST(L(I))                                                     19RUC111
   22 CONTINUE                                                          19RUC112
      FF(J)=MIN-EFT(J)                                                  19RUC113
      TF(J)=LST(J)-EST(J)                                               19RUC114
      IF(EST(J).EQ.LST(J)) GO TO 24                                     19RUC115
   23 RC(J)=" "                                                         19RUC116
      GO TO 25                                                          19RUC117
   24 IF(EFT(J).NE.LFT(J)) GO TO 23                                     19RUC118
      RC(J)="R.C."                                                      19RUC119
      CONTINUE                                                          19RUC120
C     IMPRESION DE RESULTADOS                                           19RUC121
      WRITE(6,300)                                                      19RUC122
      WRITE(6,340) IDASH                                                19RUC123
      DO 26 I=1,N                                                       19RUC124
   26 WRITE(6,350) I,D(I),EST(I),EFT(I),LST(I),LFT(I),FF(I),TF(I),RC(I) 19RUC125
      GO TO 2                                                           19RUC126
C     FORMATOS DE LECTURA E IMPRESION                                   19RUC127
  100 FORMAT(I3)                                                        19RUC128
```

```
 150 FORMAT(36I2)                                                       19RUC129
 160 FORMAT(14I5)                                                       19RUC130
 200 FORMAT(1H1,4(/),40X,'MATRIZ DE SUBORDINACION DE ACTIVIDADES',//)   19RUC131
 240 FORMAT(/,2X,I3)                                                    19RUC132
 250 FORMAT(6X,29(I1,3X))                                               19RUC133
 300 FORMAT(5(/),46X,'LOS RESULTADOS OBTENIDOS SON',///,17X,'ACTIVIDAD'19RUC134
     ',2X,'DURACION',4X,'EST',7X,'EFT',7X,'LST',7X,'LFT',7X,'TFL',7X,'TF19RUC135
     T',/)                                                              19RUC136
 340 FORMAT(11X,100A1,/)                                                19RUC137
 350 FORMAT(/,20X,I3,5X,7(I5,5X),A4)                                    19RUC138
     END                                                                19RUC139
/ XEQ RUTA
12

1
  1
1
      1
    1   1
    1   1
          1 1
              1   1
        1
                1 1
                    1
   0    3    2    4    6    2    1    2    4    2    2    0

*
```

```
C         PROGRAMA GRAN M
C      ----------------- SIMPLEX ---------------
C
C      THIS PROGRAM USES THE BIG M METHOD IN THE SIMPLEX ALGORITHM TO
C      SOLVE A LINEAR PROGRAMMING PROBLEM, AS AS PUT FORTH IN
C      THE BOOK BY HILLIER AND LIEBERMAN, ENTITLED
C      THIS PROGRAM WAS WRITTEN BY ANDREW J. CANFIELD IN APRIL, 1973.
C
C      -------- COMMON AREA --------
C      A(10,15) -THE TABLEAU
C      B(10) - THE RIGHT HAND SIDE
C      FM(15) - FACTORS OF BIG M IN THE OBJECTIVE FUNCTION
C      C(15) - UNIT TERMS IN OBJECTIVE FUNCTION
C      AI(10,15),BI(10,15),FMI(10,15),CI(10,15) - SAVES INPUT DATA
C      IBV(10) - INDICATES BASIC VARIABLE FOR EACH ROW
C      LABP(35),LABR(10,3),LABC(15,3) - LABELS
C      FMZ - FACTORS OF BIG M IN CURRENT VALUE OF Z
C      CZ - UNIT TERMS IN CURRENT VALUE OF Z
C      M,N,LABLS,IBTCH - CONTROL VARIABLES
C
C      MAIN PROGRAM
C
       COMMON AI(10,15),BI(10),FMI(15),CI(15),A(10,15),B(10),FM(15),
      1C(15),IBV(10),LABP(35),LABR(10,3),LABC(15,3),FMZ,CZ,M,N,
      2LABLS,IBTCH
       CALL IOCS1(5,3)
C
C      INITIALIZE
  100  IFAIL = 0
       CALL INIT(IFAIL)
       IF (IFAIL)  105,105,400
   05  ISTEP = 0
C
C      TEST IF CURRENT BASIS OPTIMAL
  110   DO 115    J = 1, N
       IF  (FM(J)) 117,112,115
  112  IF  (C(J))    117,115,115
  115   CONTINUE
       IF (FMZ .NE. 0.0) GO TO 500
       GO TO 200
C
C      LOCATE ENTERING BASIC VARIABLE
  117   FME = 0.0
       CE  = 0.0
       DO 128     J = 1, N
       IF  (FM(J)-FME) 125,122,128
  122  IF  (C(J)-CE)      125,128,128
  125   FME = FM(J)
       CE = C(J)
       JENT = J
  128   CONTINUE
C
C      LOCATE LEAVING BASIC VARIABLE
       ILEV = 0
       DO 138    I = 1, M
       IF  (B(I))    500,130,130
   30  IF  (A(I,JENT)) 138,138,132
  132   IF (ILEV)  135,135,133
  133   IF ((B(I)/A(I,JENT))-RATIO) 135,135,138
  135   ILEV = I
       RATIO = B(I)/A(I,JENT)
  138  CONTINUE
       IF (ILEV) 300,300,140
C
```

```
C     PIVOT                                                             SIM10440
  140 CALL PIVOT (JENT,ILEV,ISTEP)                                      SIM10450
  142 WRITE(3,11)                                                       SIM10460
   11 FORMAT(36H0-----------------------------------)                   SIM10470
      CALL TABPR(A,B,C,FM)                                              SIM10480
      GO TO 110                                                         SIM10490
C                                                                       SIM10500
C     OPTIMAL SOLUTION FOUND                                            SIM10510
  200 WRITE (3,12)                                                      SIM10520
   12 FORMAT('0******** LA BASE ACTUAL ES OPTIMA ********')             SIM10530
      GO TO 400                                                         SIM10540
C                                                                       SIM10550
C     SOLUTION IS UNBOUNDED                                             SIM10560
  300  IF (LABLS)  310,320,310                                          SIM10570
  310 WRITE(3,14)(LABC(JENT,K),K=1,3)                                   SIM10580
   14 FORMAT('0//////// SOLUCION NO ACOTADA,  ',3A2,                    SIM10590
     1' PUEDE SER INCREMENTADO SIN LIMITE ////////')                    SIM10600
      GO TO 400                                                         SIM10610
  320 WRITE(3,15) JENT                                                  SIM10620
   15 FORMAT('0//////// SOLUCION NO ACOTADA,  ',I6,
     1' PUEDE SER INCREMENTADO SIN LIMITE')                             SIM10640
  400 WRITE(3,16)(LABP(K),K=1,35)
   16 FORMAT(1X,35A2)                                                   SIM10660
      IF (IBTCH) 100,410,100                                            SIM10670
C                                                                       SIM10680
C     FOLLOWING CARD MAY HAVE TO READ "CALL EXIT" ON SOME SYSTEMS       SIM10690
  410 CALL EXIT
  500 WRITE(3,17)
   17 FORMAT('0******** SOLUCION NO FACTIBLE ********')
      GO TO 400                                                         SIM10730
      END                                                               SIM10740
/' `UP
*.  RE       WS  UA  GRANM
// FOR
*ONE WORD INTEGERS
*LIST ALL
      SUBROUTINE INIT(IFAIL)                                            SIM20010
C                                                                       SIM20020
C     INIT READS THE DATA AND CALLS FOR AN INITIAL SOLUTION             SIM20030
C                                                                       SIM20040
      COMMON AI(10,15),BI(10),FMI(15),CI(15),A(10,15),B(10),FM(15),
     1C(15),IBV(10),LABP(35),LABR(10,3),LABC(15,3),FMZ,CZ,M,N,
     2LABLS,IBTCH                                                       SIM20070
C                                                                       SIM20080
   10 FORMAT(35A2)                                                      SIM20090
   11 FORMAT(7I10)                                                      SIM20100
   12 FORMAT(/(3A2,4X))                                                 SIM20110
   13  FORMAT(7F10.0)                                                   SIM20120
C                                                                       SIM20130
C     READ IN DATA                                                      SIM20140
      CALL IOCS1(5,3)
      READ(2,10)(LABP(K),K=1,35)
      READ(2,11) M,N,LABLS,IBTCH
      IF (LABLS) 110,120,110                                            SIM20170
  110 READ(2,12)((LABR(I,K),K=1,3),I=1,M)
      READ(2,12)((LABC(J,K),K=1,3),J=1,N)
  120 READ(2,13)(FMI(J),J=1,N)
      READ(2,13)(CI(J),J=1,N)
      DO 127 I = 1, M                                                   SIM20220
  127 READ(2,13)(AI(I,J),J=1,N)
      READ(2,13)(BI(I),I=1,M)
      READ(2,11)(IBV(I),I=1,M)
      FMZ = 0.0                                                         SIM20260
      CZ =0.0                                                           SIM20270
      WRITE(3,31)(LABP(K),K=1,35)
   31 FORMAT(/,'0-------- DATOS INICIALES -------- ',35A2)
```

```
      CALL TABPR(AI,BI,CI,FMI)
C
C     INITIALIZE TABLEAU
      CALL TABNU(IFAIL)
      IF (IFAIL) 139,139,400
  139 WRITE (3,32)
   32 FORMAT('0-------- TABLA INICIAL --------')
      CALL TABPR(A,B,C,FM)
      RETURN
C
C     ERROR - INITIAL BASIS UNINVERTABLE
  400 WRITE(3,33)
   33 FORMAT('0EL JUEGO INICIAL DE VARIABLES BASICAS NO SE PUEDE USAR')
      RETURN
      END
// DUP
*STORE      WS  UA  INIT
// FOR
*ONE WORD INTEGERS
*LIST ALL
      SUBROUTINE PIVOT (JENT,ILEV,ISTEP)
C
C     PIVOT PERFORMS THE ITERATION, EITHER BY THE NORMAL SIMPLEX
C     PIVOT, OR BY RECALCULATING THE TABLEAU FROM INITAL DATA.
C     THE DIFFERENCE IS TRANSPARENT TO THE USER AND IS NEEDED ONLY TO
C     PRESERVE ACCURACY.
C
      COMMON AI(10,15),BI(10),FMI(15),CI(15),A(10,15),B(10),FM(15),
     1C(15),IBV(10),LABP(35),LABR(10,3),LABC(15,3),FMZ,CZ,M,N,
     2LABLS,IBTCH
      CALL IOCS1(5,3)
      IF (LABLS) 103,101,103
    1 WRITE(3,11) JENT,IBV(ILEV)
   11 FORMAT(/,I5,' ENTRA A LA BASE,    ',I4,' SALE DE LA BASE')
      GO TO 105
  103 L = IBV(ILEV)
      WRITE(3,12)(LABC(JENT,K),K=1,3), (LABC(L,K),K=1,3)
   12 FORMAT (/,1X,3A2,' ENTRA A LA BASE,    ',3A2,' SALE DE LA BASE')
C
C
C     DECIDE WHETHER TO DO A NORMAL SIMPLEX PIVOT
C     OR TO RECOMPUTE THE TABLEAU FROM THE ORIGINAL DATA
C     AND THE INVERSE OF THE CURRENT BASIS
C     WE RECOMPUTE EVERY FIVE ITERATIONS
  105 ISTEP = ISTEP + 1
      IF (ISTEP-5) 130,110,110
C
C     RECOMPUTE TABLEAU
  110 ISTEP = 0
      IBV(ILEV) = JENT
      CALL TABNU(ICANT)
      IF (ICANT) 190,190,115
C     IF BASIS IS OF THAT RARE TYPE WHICH CANNOT BE INVERTED BY TABNU,
C     WE INSTEAD DO A NORMAL SIMPLEX PIVOT.
  115 ISTEP = 5
C
C     NORMAL SIMPLEX PIVOT
C
C     NORMALIZE PIVOTAL EQUATION
  130  TERM = A(ILEV,JENT)
      DO 135 J = 1, N
  135 A(ILEV,J) = A(ILEV,J) / TERM
      B(ILEV) = B(ILEV) / TERM
C
C     ELIMINATE ENTERING VARIABLE FROM ALL OTHER EQUATIONS
      DO 148 I = 1, M
      IF (I-ILEV) 142,148,142                                           SIM30460
  142 RATIO = A(I,JENT)                                                 SIM30470
      DO 145 J = 1, N                                                   SIM30480
      TERM = A(I,J) - A(ILEV,J) * RATIO                                 SIM30490
  145  A(I,J) = CLEAN(TERM)                                             SIM30500
      TERM = B(I) - B(ILEV) * RATIO                                     SIM30510
      B(I) = CLEAN(TERM)                                                SIM30520
  148 CONTINUE                                                          SIM30530
      IBV(ILEV) = JENT                                                  SIM30540
C                                                                       SIM30550
C     ELIMINATE ENTERING VARIABLE FROM OBJECTIVE FUNCTION               SIM30560
      CALL RMOVE(ILEV)                                                  SIM30570
  190 RETURN                                                            SIM30580
      END                                                               SIM30590
// DUP
*STORE      WS  UA  PIVOT
// FOR
*ONE WORD INTEGERS
*LIST ALL
      SUBROUTINE TABNU(ICANT)                                           SIM40010
C                                                                       SIM40020
C     TABNU USES CROUT"S METHOD TO IMPLICITLY FIND THI BASIS INVERSE,   SIM40030
C     THEN USES THIS TO REGENERATE THE SIMPLEX TABLEAU FROM THE         SIM40040
C     INITIAL DATA.                                                     SIM40050
C                                                                       SIM40060
      COMMON AI(10,15),BI(10),FMI(15),CI(15),A(10,15),B(10),FM(15),
     1C(15),IBV(10),LABP(35),LABR(10,3),LABC(15,3),FMZ,CZ,M,N,
     2LABLS,IBTCH                                                       SIM40090
      DIMENSION BASIS(10,10), IPS(10), SCALE(10)
C                                                                       SIM40110
C     THE ARRAY "BASIS" IS USED AS A WORK AREA                          SIM40120
C                                                                       SIM40130
C     FIND SCALE FACTOR TO PRESERVE ACCURACY                            SIM40140
      DO 120 I = 1, M                                                   SIM40150
      IPS(I) = I                                                        SIM40160
      ROMAX = 0.0                                                       SIM40170
      DO 115 JB = 1,M                                                   SIM40180
      K= IBV(JB)                                                        SIM40190
      TERM = AI(I,K)                                                    SIM40200
      BASIS(I,JB) = TERM                                                SIM40210
      IF (TERM) 111,115,113                                             SIM40220
  111 TERM = -TERM                                                      SIM40230
  113  IF (ROMAX-TERM) 114,115,115                                      SIM40240
  114 ROMAX = TERM                                                      SIM40250
  115 CONTINUE                                                          SIM40260
      IF (ROMAX) 700,700,118                                            SIM40270
  118 SCALE(I) = 1.0/ROMAX                                              SIM40280
  120 CONTINUE                                                          SIM40290
C                                                                       SIM40300
C                                                                       SIM40310
C     IMPLICITLY INVERT BASIS                                           SIM40320
      DO 190 JB = 1, M                                                  SIM40330
      IF (JB-2) 150,150,131                                             SIM40340
  131 JBM1 = JB - 1                                                     SIM40350
      DO 140 I = 2, JBM1                                                SIM40360
      LESS = I - 1                                                      SIM40370
      IP = IPS(I)                                                       SIM40380
      TERM = BASIS(IP,JB)                                               SIM40390
      DO 136 K = 1, LESS                                                SIM40400
      KP = IPS(K)                                                       SIM40410
  136 TERM = TERM - BASIS(IP,K) * BASIS(KP,JB)                          SIM40420
      TERM = CLEAN(TERM)                                                SIM40430
  140 BASIS(IP,JB) = TERM                                               SIM40440
  150 RATPV = 0.0                                                       SIM40450
      DO 170 I = JB, M                                                  SIM40460
      IP = IPS(I)                                                       SIM40470
      TERM = BASIS(IP,JB)                                               SIM40480
      IF (JB-1) 162,162,157                                             SIM40490
  157 LESS = JB - 1                                                     SIM40510
      DO 160 K = 1, LESS                                                SIM40520
      KP = IPS(K)                                                       SIM40530
  160  TERM = TERM - BASIS(IP,K) * BASIS(KP,JB)                         SIM40540
      TERM = CLEAN(TERM)                                                SIM40550
      BASIS(IP,JB) = TERM                                               SIM40560
  162  IF (TERM) 165,170,166                                            SIM40570
  165 TERM = -TERM                                                      SIM40580
  166 TERM = TERM * SCALE(IP)                                           SIM40590
      IF (RATPV-TERM) 168,170,170                                       SIM40600
  168 RATPV = TERM                                                      SIM40610
      IPV = I                                                           SIM40620
  170 CONTINUE                                                          SIM40630
      IF (RATPV) 700,700,174                                            SIM40640
  174  IF (M-JB) 190,190,176                                            SIM40650
  176 IPVOT = IPS(IPV)                                                  SIM40660
      IF (JB-IPV) 178,178,181                                           SIM40670
  178  IPS(IPV) = IPS(JB)                                               SIM40680
      IPS(JB) = IPVOT                                                   SIM40690
  181  JBP1 = JB + 1                                                    SIM40700
      DO 188 I = JBP1, M                                                SIM40710
      IP = IPS(I)                                                       SIM40720
  188 BASIS(IP,JB) = BASIS(IP,JB) / BASIS(IPVOT,JB)                     SIM40730
  190 CONTINUE                                                          SIM40740
      ICANT = 0                                                         SIM40750
C                                                                       SIM40760
C     SOVE FOR THE PROPER RIGHT HAND SIDE VECTOR B                      SIM40770
      IP = IPS(1)                                                       SIM40780
      B(1) = BI(IP)                                                     SIM40790
      DO 224 I = 2, M                                                   SIM40800
      IP = IPS(I)                                                       SIM40810
      SUM = 0.0                                                         SIM40820
      KLIM = I -1                                                       SIM40830
      DO 222 K = 1, KLIM                                                SIM40840
      KP = IPS(K)                                                       SIM40850
  222 SUM = SUM + BASIS(IP,K) * B(K)                                    SIM40860
  224 B(I) = BI(IP) - SUM                                               SIM40870
      MP = IPS(M)                                                       SIM40880
      TERM  = B(M) / BASIS(MP,M)                                        SIM40890
      B(M) = CLEAN(TERM)                                                SIM40900
      LOLIM = M - 1                                                     SIM40910
      DO 256 INEG = 1, LOLIM                                            SIM40920
      I = M - INEG                                                      SIM40930
      IP = IPS(I)                                                       SIM40940
      SUM = 0.0                                                         SIM40950
      KLOW = I + 1                                                      SIM40960
      DO 254 K = KLOW,M                                                 SIM40970
  254 SUM = SUM + B(K) * BASIS(IP,K)                                    SIM40 80
      TERM= (B(I) -SUM) / BASIS(IP,I)                                   SIM40990
  256 B(I) = CLEAN(TERM)                                                SIM41000
C                                                                       SIM41010
C     SOLVE FOR THE PROPER MATRIX A                                     SIM41020
      DO 280 J= 1, N                                                    SIM41030
      IP = IPS(1)                                                       SIM41040
      A(1,J) = AI(IP,J)                                                 SIM41050
      DO  262 I = 2, M                                                  SIM41060
      SUM = 0.0                                                         SIM41070
      IP = IPS(I)                                                       SIM41080
      KLIM = I -1                                                       SIM41090
      DO 260 K = 1, KLIM                                                SIM41100
      KP = IPS(K)                                                       SIM41110
  260 SUM= SUM + BASIS(IP,K) * A(K,J)                                   SIM41120
  262 A(I,J) = AI(IP,J) - SUM                                           SIM41130
      MP = IPS(M)                                                       SIM41140
      TERM = A(M,J) / BASIS(MP,M)                                       SIM41150
      A(M,J) = CLEAN(TERM)                                              SIM41160
      LOLIM = M - 1                                                     SIM41170
      DO 276 INEG = 1, LOLIM                                            SIM41180
      I = M - INEG                                                      SIM41190
      SUM = 0.0                                                         SIM41200
      IP = IPS(I)                                                       SIM41210
      KLOW = I + 1                                                      SIM41220
      DO 274 K = KLOW, M                                                SIM41230
  274 SUM = SUM + A(K,J) * BASIS(IP,K)                                  SIM41240
      TERM = (A(I,J) -SUM) / BASIS(IP,I)                                SIM41250
  276 A(I,J) = CLEAN(TERM)                                              SIM41260
  280 CONTINUE                                                          SIM41270
      DO 310 J= 1, N                                                    SIM41280
      FM(J) = FMI(J)                                                    SIM41290
  310 C(J) = CI(J)                                                      SIM41300
C                                                                       SIM41310
C     REGENERATE THE OBJECTIVE FUNCTION FROM INITIAL DATA               SIM41320
      FMZ = 0.0                                                         SIM41330
      CZ = 0.0                                                          SIM41340
      DO 320 I = 1, M                                                   SIM41350
  320 CALL RMOVE(I)                                                     SIM41360
      RETURN                                                            SIM41370
C                                                                       SIM41380
C     BASIS NOT ADEQUATELY INVERTABLE                                   SIM41390
C     SCALING MAY NOT HAVE BEEN SUCCESSFUL                              SIM41400
C     ZERO ROW IN BASIS                                                 SIM41410
  700  ICANT = 1                                                        SIM41420
      RETURN                                                            SIM41430
      END                                                               SIM41440
// DUP
*STORE      WS  UA  TABNU
// FOR
*ONE WORD INTEGERS
*LIST ALL
      SUBROUTINE RMOVE(I)                                               SIM50010
C                                                                       SIM50010
C     RMOVE REMOVES THE I-TH BASIS VARIABLE FROM THE OBJECTIVE          SIM50030
C     FUNCTION.                                                         SIM50040
C                                                                       SIM50050
      COMMON AI(10,15),BI(10),FMI(15),CI(15),A(10,15),B(10),FM(15),
     1C(15),IBV(10),LABP(35),LABR(10,3),LABC(15,3),FMZ,CZ,M,N,
     2LARGS,IRTCH                                                       SIM50080
C                                                                       SIM50090
      L = IBV(I)                                                        SIM50100
      TERMM = FM(L) / A(I,L)                                            SIM50110
      TERMC = C(L) / A(I,L)                                             SIM50120
      DO 110 J = 1, N                                                   SIM50130
      TERM = FM(J) - A(I,J) * TERMM                                     SIM50140
      FM(J) = CLEAN(TERM)                                               SIM50150
      TERM = C(J) - A(I,J) * TERMC                                      SIM50160
  110 C(J) = CLEAN(TERM)                                                SIM50170
      TERM = FMZ -B (I) * TERMM                                         SIM50180
      FMZ = CLEAN(TERM)                                                 SIM50190
      TERM = CZ - B(I) * TERMC                                          SIM50200
      CZ = CLEAN(TERM)                                                  SIM50210
      RETURN                                                            SIM50220
      END                                                               SIM50230
// DUP
*STORE      WS  UA  RMOVE
// FOR
*ONE WORD INTEGERS
*LIST ALL
      FUNCTION CLEAN(REAL)                                              SIM60010
C                                                                       SIM60020
C     CLEAN RETURNS A VALUE WHICH IS ZERO IF THE ARGUMENT IS WITHIN     SIM60030
```

```
C     PLUS OR MINUS 0.001 OF ZERO, AND WHICH IS THE VALUE OF THE        SIM60040
C     ARGUMENT OTHERWISE.                                               SIM60050
C                                                                       SIM60060
      IF (REAL-0.001) 111,118,118                                       SIM60070
  111 IF (REAL+0.001) 118,118,115                                       SIM60080
  115 CLEAN = 0.0                                                       SIM60090
      RETURN                                                            SIM60100
  1    CLEAN = REAL                                                     SIM60110
      RETURN                                                            SIM60120
      END                                                               SIM60130
// DUP
*STORE      WS  UA  CLEAN
// FOR
*ONE WORD INTEGERS
*LIST ALL
      SUBROUTINE TABPR(AW,BW,CW,FMW)                                    SIM70010
C                                                                       SIM70020
C     TABPR PRINTS OUT THE SIMPLEX TABLEAU                              SIM70030
C     TABPR USES 115 PRINT POSITIONS                                    SIM70040
C                                                                       SIM70050
      COMMON AI(10,15),BI(10),FMI(15),CI(15),A(10,15),B(10),FM(15),
     1C(15),IBV(10),LABP(35),LABR(10,3),LABC(15,3),FMZ,CZ,M,N,
     2LABLS,IBTCH                                                       SIM70080
      DIMENSION AW(10,15), BW(10), CW(15), FMW(15)
      CALL IOCS1(5,3)
C                                                                       SIM70100
C     DETERMINE LENGTH OF FIRST LINES                                   SIM70110
      LIM = 7                                                           SIM70120
      IF (N-LIM)   103,105,105                                          SIM70130
  103  LIM = N                                                          SIM70140
  105  IF (LABLS)   200,300,200                                         SIM70150
C                                                                       SIM70160
C     LABELED OUTPUT                                                    SIM70170
C                                                                       SIM70180
C     FIRST LINES SECTION                                               SIM70190
  200 WRITE(3,12)((LABC(J,K),K=1,3),J=1,LIM)
   12 FORMAT(/,8HORENGLON,5X,14H L.D.    BASE ,4X,7(6X,3A2))
      WRITE(3,13) FMZ, (FMW(J),J=1,LIM)
   13 FORMAT(7H0 FM    ,F12.3,12X,7F12.3)
      WRITE(3,14) CZ, (CW(J),J=1,LIM)
   14 FORMAT(1X,6H -C    ,F12.3,2X,6HZ      ,4X,7F12.3)
      WRITE(3,60)
   60 FORMAT(/)
      DO 214   I = 1, M                                                 SIM70260
      L = IBV(I)                                                        SIM70270
  214 WRITE(3,15)(LABR(I,K),K=1,3), BW(I), (LABC(L,K),K=1,3),
     1(AW(I,J),J=1,LIM)                                                 SIM70290
   15 FORMAT(1X,3A2,F12.3,2X,3A2,4X,7F12.3)                             SIM70300
C                                                                       SIM70310
C     REMAINING SECTIONS LOOP
C                                                                       SIM70330
      LOW = 8                                                           SIM70340
C                                                                       SIM70350
C     DETERMINE IF FINISHED                                             SIM70360
  231 IF (N-LOW)   400,233,233                                          SIM70370
C                                                                       SIM70380
C     DETERMINE LINES LENGTH                                            SIM70390
  2   LIM = LOW + 8                                                     SIM70400
      IF (N-LIM)   235,236,236                                          SIM70410
  235  LIM = N                                                          SIM70420
C                                                                       SIM70430
C     PRINT SECTION                                                     SIM70440
  236 WRITE(3,17)((LABC(J,K),K=1,3),J=LOW,LIM)
   17 FORMAT(//,1X,6HRENG  ,9(6X,3A2))
      WRITE(3,18)(FMW(J),J=LOW,LIM)
   18 FORMAT(7H0 FM    ,9F12.3)
```

```
      WRITE(3,19)(CW(J),J=LOW,LIM)
   19 FORMAT(7H0 -C    ,9F12.3)
      WRITE(3,60)
        DO 245    I = 1, M
  245 WRITE(3,20)(LABR(I,K),K=1,3), (AW(I,J),J=LOW,LIM)
   20 FORMAT(1X,3A2,9F12.3)                                             SIM70530
      LOW = LOW + 9                                                     SIM70540
      GO TO 231                                                         SIM70550
C                                                                       SIM70560
C     UNLABELED OUTPUT                                                  SIM70570
C                                                                       SIM70580
C     FIRST SECTION                                                     SIM70590
  300 WRITE(3,32)(J,J=1,LIM)
   32 FORMAT(/,8H0RENGLON,5X,15H L.D.    BASE    ,7(9X,I3))
      WRITE(3,13)FMZ, (FMW(J),J=1,LIM)
      WRITE(3,14)CZ, (CW(J),J=1,LIM)
      WRITE(3,60)
      DO 314    I = 1, M                                                SIM70640
      L = IBV(I)                                                        SIM70650
  314 WRITE(3,14)I, BW(I), L, (AW(I,J),J=1,LIM)
   35 FORMAT(1X,I3,3X,F12.3,I5,7X,7F12.3)                               SIM70670
C                                                                       SIM70680
C     REMAINING SECTIONS LOOP                                           SIM70690
C                                                                       SIM70700
      LOW = 8                                                           SIM70710
C                                                                       SIM70720
C     DETERMINE IF FINISHED                                             SIM70730
  331 IF (N-LOW)   400,333,333                                          SIM70740
C                                                                       SIM70750
C     DETERMINF LINES LENGTH                                            SIM70760
  333 LIM = LOW + 8                                                     SIM70770
      IF (N-LIM)   335,336,336                                          SIM70780
   .5 LIM = N                                                           SIM70790
C                                                                       SIM70800
C     PRINT SECTION                                                     SIM70810
  336 WRITE(3,37)(J,J=LOW,LIM)
   37 FORMAT(//,1X,3HREN,9(9X,I3))
      WRITE(3,18)(FMW(J),J=LOW,LIM)
      WRITE(3,19)(CW(J),J=LOW,LIM)
      WRITE(3,60)
      DO 345    I = 1, M                                                SIM70860
  345 WRITE(3,40)I, (AW(I,J),J=LOW,LIM)
   40 FORMAT(1X,I3,3X,9F12.3)
      LOW = LOW + 9                                                     SIM70890
      GO TO 331                                                         SIM70900
C                                                                       SIM70910
  400 RETURN                                                            SIM70920
      END                                                               SIM70930
```

```
// JOB
// FOR
*ONE WORD INTEGERS
*IOCS(CARD,1403 PRINTER)
*N      INVEI
*S
C
C      PROGRAMA MOPASIN
C      MODELOS PARA SISTEMAS DE INVENTARIOS
C
C      TARJETAS DE DATOS
C
C      LA PRIMERA TARJETA ES UNA IDENTIFICASION,
C      CON UN MAXIMO DE CUARENTA CARACTERES
C
       DIMENSION BETA(10)
       CALL IOCS1(5,3)
     1 READ (2,26) BETA
       WRITE (3,27) BETA
C
C      SEGUNDA TARJETA (CONTROL CARD), DONDE:
C      N= TOTAL DE SEMANAS POR ANALIZAR
C      IR=A= PORCENTAJE APLICADO AMENUDEO
C      IW=B= PORCENTAJE APLICADO AL VALOR DE MAYOREO
C      LW= FACTOR DE MANDO DE TIEMPO APLICADO A MAYOREO
C      LF= FACTOR DE MANDO DE TIEMPO APLICADO A LA PRODUCCION
C
       READ(2,28) N,IR,IW,LW,LF
       IF (N .EQ. 0) GO TO 40
       IF (IR .GT. 0) GO TO 3
       A=1.0
       GO TO 4
     3 A=IR/100.0
     4 IF (IW .GT. 0) GO TO 6
     5 B=1.0
       GO TO 7
     6 B=IW/100.0
C
C      RI= NIVEL DE INVENTARIO DE LA SEMANA ANTERIOR PARA MENUDEO
C      RO= ORDENES MENUDEO = WS = PRODUCCION DE FAB. SEMANA ANT.
C      WI = NIVEL DE INVENT. DE LA SEMANA ANT. PARA MAYOREO
C      WO2 = ORDEN DE MAYOREO PARA LF NO = 0
C      WO1 = ORDEN DE MAYOREO PARA LF = 0
C      FR = PRECIO DE FABRICA
C
     7 RI=100.0
       RO=100.0
       WS=100.0
       WI=200.0
       WO2=100.0
       WO1=100.0
       FR=100.0
C
C      IMPRIME ENCABEZADOS PARA LA SALIDA SEMANAL
C
       WRITE (3,29)
       WRITE (3,30)
C
C      EMPIEZA EL LOOP PARA LAS COMPUTACIONES DE LA SEMANA
C
       DO 24 I=1,N
C
C      LEE Y VERIFICA LAS TARJETAS DE DATOS QUE CONTIENEN LAS VENTAS
C      SEMANALES
```

```
// JOB
// FOR
*ONE WORD INTEGERS
*IOCS(CARD,1403 PRINTER)
*NAME INVEI
*SAVE
C
C      PROGRAMA MOPASIN
C      MODELOS PARA SISTEMAS DE INVENTARIOS
C
C      TARJETAS DE DATOS
C
C      LA PRIMERA TARJETA ES UNA IDENTIFICASION,
C      CON UN MAXIMO DE CUARENTA CARACTERES
C
       DIMENSION BETA(10)
       CALL IOCS1(5,3)
     1 READ (2,26) BETA
       WRITE (3,27) BETA
C
C      SEGUNDA TARJETA (CONTROL CARD), DONDE:
C      N= TOTAL DE SEMANAS POR ANALIZAR
C      IR=A= PORCENTAJE APLICADO AMENUDEO
C      IW=B= PORCENTAJE APLICADO AL VALOR DE MAYOREO
C      LW= FACTOR DE MANDO DE TIEMPO APLICADO A MAYOREO
C      LF= FACTOR DE MANDO DE TIEMPO APLICADO A LA PRODUCCION
C
       READ(2,28) N,IR,IW,LW,LF
       IF (N .EQ. 0) GO TO 40
       IF (IR .GT. 0) GO TO 3
     2 A=1.0
       GO TO 4
     3 A=IR/100.0
     4 IF (IW .GT. 0) GO TO 6
     5 B=1.0
       GO.TO 7
     6 B=IW/100.0
C
C      RI= NIVEL DE INVENTARIO DE LA SEMANA ANTERIOR PARA MENUDEO
C      RO= ORDENES MENUDEO = WS = PRODUCCION DE FAB. SEMANA ANT.
C      WI = NIVEL DE INVENT. DE LA SEMANA ANT. PARA MAYOREO
C      WO2 = ORDEN DE MAYOREO PARA LF NO = 0
C      WO1 = ORDEN DE MAYOREO PARA LF = 0
C      FR = PRECIO DE FABRICA
C
     7 RI=100.0
       RO=100.0
       WS=100.0
       WI=200.0
       WO2=100.0
       WO1=100.0
       FR=100.0
C
C      IMPRIME ENCABEZADOS PARA LA SALIDA SEMANAL

       WRITE (3,29)
       WRITE (3,30)
C
C      EMPIEZA EL LOOP PARA LAS COMPUTACIONES DE LA SEMANA
C
       DO 24 I=1,N
C
C      LEE Y VERIFICA LAS TARJETAS DE DATOS QUE CONTIENEN LAS VENTAS
C      SEMANALES
```

```
C
      WRITE (3,34) N,IR,IW.LW.LF
   25 CONTINUE
      GO TO 1
   26 FORMAT (10A4)
   27 FORMAT (1H1,22HPROGRAMA MOPASIN PARA ,10A4)
   28 FORMAT (I2,8X,I2.8X,I2,8X,I1,9X,I1)
   29 FORMAT (49H0SEMANA-----------MENUDEO-----------    **:********,
     127HMAYOREO***********  FABRICA)
   30 FORMAT (1H ,46HNO     VENTAS  RECIBO  INVT  ORDEN   EMBARCO  ,
     130HRECIBO  INVTO  ORDEN  RELACION)
   31 FORMAT (I2,8X,F3.0)
   32 FORMAT (24H ALGUNOS DATOS ESTAN MAL)
   33 FORMAT (1H ,I2,5X,F6.1,3F7.1,2F9.1,2F7.1,F8.1)
   34 FORMAT (1H0,I3,17H SEMANAS CORRIDAS//
     129H VALOR EN % , QUE INTERVIENE ,
     238HEN LA FORMULA DE ORDENES DE MENUDEO = ,I3//
     329H VALOR EN % , QUE INTERVIENE ,
     438HEN LA FORMULA DE ORDENES DE MAYOREO = ,I3//
     549H FACTOR DE MANDO DE TIEMPO APLICADO AL MAYOREO = ,I3//
     654H FACTOR DE MANDO DE TIEMPO APLICADO A LA PRODUCCION = ,I3)
   40 CONTINUE
      CALL EXIT
      END
      PROGRAM NET
C     ROY D HARRIS  MAY 1972
C     THIS PROGRAM WILL SOLVE NETWORK PROBLEMS
C     FOR SHORTEST PATH(S)
C     FOR MINIMUM COST (AT SPECIFIED FLOWRATE)
C     FOR MAXIMUM FLOW
C     THIS VERSION FOR CDC 3100
      COMMON I(200),J(200),JWIN(100),KADJ(200),KARC(200),KASG(200),KCUM(
     1200),KFLOW(200),KIST(100),LABEL(200),MAX(100),ITERM,JFLAG,KALL,KKE
     2ND,KFL,RFST,KEND,LOOP,NA,NIN,NOUT,ILPHA(10)
C     I(200) FROM NODE NUMBER
C     J(200) 10 NODE NUMBER
C     KARC(200)  ARC DISTANCE OR COST
C     MAX(100) ARC CAPACITY
C     NA NUMBER OF ARCS IN NETWORK
      COMMON /DATA/ IEND
      DATA (IEND = 4HSTOP)
      ISTOP=IEND
      NIN = 5
      NOUT = 6
C
C     RETURN 10 HERE  ON NEW DATA SET
1     DO 2 N=1,100
2     LABEL(N)=9999
C
C     READ AND  RITE USERS NAME CARD
      READ (NIN,33) ILPHA
      WRITE (NOUT,34) ILPHA
      IF (ILPHA(1).EQ.ISTOP) GO TO 32
C
C     READ AND WRITE NETWORK INPUT DATA
      WRITE (NOUT,35)
      WRITE (NOUT,36)
      NA=0
3     READ (NIN,37) A1,A2,A3,A4
C     99 IN FIRST FIELD STOPS NETWORK INPUT
      IF (A1.GT.98.) GO TO 4
      NA=NA+1
      IF (NA.GT.99) GO TO 30
      I(NA)=A1
      J(NA)=A2
      KARC(NA)=A3
      MAX(NA)=A4
      II=A1+1
      LABEL(II)=II
      JJ=A2+1
      LABEL(JJ)=JJ
      WRITE (NOUT,38) I(NA),J(NA),KARC(NA),MAX(NA)
      GO TO 3
4     WRITE (NOUT,39) NA
C
C     THIS SECTION CHECKS OUT THE INPUT NETWORK
C     THE NETWORK IS SORTED AND TESTED FOR BAD DATA
      DO 7 II=1,NA
      NA1=NA-II
      DO 7 KK=1,NA1
C     CHECK DATA AS SORT BEING MADE
      IF (I(KK).EQ.J(KK)) GO TO 30
      IF (I(KK).LT.0) GO TO 30
      IF (J(KK).LT.0) GO TO 30
      IF (KARC(KK).LT.0) GO TO 30
      IF (MAX(KK).LT.0) GO TO 30
C     SORT ON FROM NODE AND TO NODE
      KKK=KK+1
      IF (I(KK)-I(KKK)) 7,5,6
5     IF (J(KK)-J(KKK)) 7,7,6
6     IS=I(KK)
      JS=J(KK)
      KA=KARC(KK)
      KM=MAX(KK)
      I(KK)=I(KKK)
      J(KK)=J(KKK)
      KARC(KK)=KARC(KKK)
      MAX(KK)=MAX(KKK)
      I(KKK)=IS
      J(KKK)=JS
      KARC(KKK)=KA
      MAX(KKK)=KM
7     CONTINUE
      WRITE (NOUT,40)
      WRITE (NOUT,36)
      DO 8 II=1,NA
8     WRITE (NOUT,38) I(II),J(II),KARC(II),MAX(II)
C     FIND AND PRINT NODE NUMBERS IN NETWORK
      IJ=0
      DO 9 II=1,99
      IF (LABEL(II).EQ.9999) GO TO 9
      IJ=IJ+1
      LABEL(IJ)=LABEL(II)-1
9     CONTINUE
      WRITE (NOUT,41) IJ
      WRITE (NOUT,42) (LABEL(II),II=1,IJ)
C
C     READ AND PRINT ANALYSIS CONTROL CARD
10    READ (NIN,43) IFLAG,X1,X2,X3,JFLAG
      KFST=X1
      KEND=X2
      KFL=X3
      IND=KFL
C     9 IN FIRST FIELD--RETURN FOR NEXT DATA SET
      IF (IFLAG.GE.4) GO TO 1
      WRITE (NOUT,44) ILPHA
      WRITE (NOUT,45)
      WRITE (NOUT,46)
      WRITE (NOUT,47) IFLAG,KFST,KEND,KFL
C
C     CHECK ANALYSIS CONTROL CARD FOR BAD DATA
      IF (KFST.LT.0) GO TO 31
      IF (KEND.LE.0) GO TO 31
      IF (KFST.EQ.KEND) GO TO 31
      IF (IFLAG.EQ.0) GO TO 31
C
C     CHECK IF KFST AND KEND EXIST IN NETWORK
      INFES=9
      JFES=9
      DO 12 II=1,NA
      IF (I(II).NE.KFST) GO TO 11
      INFES=0
11    IF (J(II).NE.KEND) GO TO 12
      JFES=0
12    CONTINUE
      IF (INFES.EQ.9) GO TO 31
      IF (JFES.EQ.9) GO TO 31
C
C     BRANCH TO TYPE OF ANALYSIS DESIRED
      GO TO (13,18,25), IFLAG
C
C     SHORT ANALYSIS CONTROL SECTION
13    WRITE (NOUT,48)
      DO 14 II=1,NA
```

```
14    LABEL(II)=0
      LOOP=0
      ITERM=0
      CALL SHORT
      IF (ITERM .NE. 40) GO TO 15
      WRITE (NOUT,49)
      WRITE (NOUT,50)
      WRITE (NOUT,51)
      DO 17 II=1,NA
      IF (LABEL(II).NE.5) GO TO 16
      WRITE (NOUT,52) I(II),J(II),KARC(II),KCUM(II)
      GO TO 1/
16    WRITE (NOUT,53) I(II),J(II),KARC(II)
17    CONTINUE
C     RETURN FOR NEXT ANALYSIS CONTROL CARD
      GO TO 10
C
C     MINIMUM COST FLOW CONTROL SECTION
18    LOOP=0
      ITERM=0
      DO 19 II=1,NA
19    LABEL(II)=0
      CALL SHORT
      IF (ITERM.EQ.40) GO TO 29
      IF (JFLAG.EQ.0) GO TO 20
      WRITE (NOUT,69)
20    CALL FLOW
      WRITE (NOUT,54)
      IF (ITERM.EQ.30) GO TO 21
      WRITE (NOUT,55) IND
      IND=IND-KFL
      WRITE (NOUT,56) IND
      WRITE (NOUT,57)
C     SUM TOTAL FLOW AND TOTAL COST
      NA=NA/2
      KOST=0
     *DO 22 K=1,NA
     *KCUM(K)=KARC(K)*KFLOW(K)
22    KOST=KOST+KCUM(K)
      DO 24 K=1,NA
      IF (KFLOW(K).EQ.0) GO TO 23
      WRITE (NOUT,58) I(K),J(K),KARC(K),MAX(K),KFLOW(K),KCUM(K)
      GO TO 24
23    WRITE (NOUT,58) I(K),J(K),KARC(K),MAX(K)
24    CONTINUE
      WRITE (NOUT,59) IND
      WRITE (NOUT,60) KOST
C     RETURN FOR NEXT ANALYSIS CONTROL CARD
      GO TO 10
C
C     MAXIMUM FLOW CONTROL SECTION
25    KFL=999999
      IND=999999
      LOOP=0
      ITERM=0
     *DO 26 II=1,NA
      LABEL(II)=0
      CALL SHORT
      IF (ITERM.EQ.40) GO TO 29
      IF (JFLAG.EQ.0) GO TO 27
      WRITE (NOUT,69)
27    CALL FLOW
      WRITE (NOUT,61)
      WRITE (NOUT,62)
      NA=NA/2
      DO 28 K=1,NA
```

```
28    WRITE (NOUT,63) I(K),J(K),KARC(K),MAX(K),KFLOW(K)
      IND=IND-RFL
      WRITE (NOUT,59) IND
C     RETURN FOR NEXT ANALYSIS CONTROL CARD
      GO TO 10
C
C     THIS SECTION PRINTS ERROR MESSAGES
      WRITE (NOUT,64)
      GO TO 32
30    WRITE (NOUT,65)
      WRITE (NOUT,66) I(KK),J(KK)
      GO TO 32
31    WRITE (NOUT,68)
32    WRITE (NOUT,67)
C
C
C
33    FORMAT (10A4)
34    FORMAT (17H1PROGRAM NET FOR ,10A4)
35    FORMAT (29H0****INPUT NETWORK AS READ***)
36    FORMAT (29H FROM TO  ARC DATA  MAX FLOW)
37    FORMAT (2(F2.0,3X),2(F5.0,5X))
38    FORMAT (1X,2(I2,3X),2(I5,5X))
39    FORMAT (1H0,I2.26H DATA CARDS (ARCS) READ IN)
40    FORMAT (29H0----SORTED INPUT NETWORK----)
41    FORMAT (11H0THERE ARE  ,I2,18H NODES IN NETWORK )
42    FORMAT (1X,5I5)
43    FORMAT (I1.4X,2(F2.0,3X).F5.0,I1)
44    FORMAT (17H1NET RESULTS FOR  ,10A4)
45    FORMAT (32H0***ANALYSIS CONTROL CARD IS***)
46    FORMAT (32H *****CODE FROM TO   FLOW*****)
47    FORMAT (6H ***(,I1,4X,I2.3X,I2,3X,I5,6H)*****)
      FORMAT (32H0*****SHORTEST PATH RESULTS*****)
49    FORMAT (32H0***NO ROUTE TO TERMINAL NODE***)
50    FORMAT (32H **PARTIAL SOLUTION SHOWN BELOW*)
51    FORMAT (32H FROM TO  ARC DATA  CUMMULATIVE)
52    FORMAT (1X,I2,3X,I2,3X.I5,3X,I7,9H  *SHORT*)
53    FORMAT (1X.I2.3X,I2,3X,I5)
54    FORMAT (51H0************MINIMUM COST FLOW RESULTS************)
55    FORMAT (26H **********FLOW DEMAND OF  ,I5.20H NOT FESIBLE********)
56    FORMAT (26H *********FESIBLE FLOW OF  ,I5,20H SHOWN BELOW********)
57    FORMAT (51H FROM TO   ARC DATA  MAX FLOW  ***FLOW***---COST---)
58    FORMAT (1X,2(I2,3X).2(I5,5X),2X,I5,3X,I7)
59    FORMAT (23H **********TOTAL FLOW  ,I7,11H **********)
60    FORMAT (23H **********TOTAL COST  ,I7.11H **********)
61    FORMAT (41H0**********MAXIMUM FLOW RESULTS**********)
62    FORMAT (41H FROM TO   ARC DATA  MAX FLOW  ***FLOW***)
63    FORMAT (1X,2(I2,3X),2(I5,5X),I7)
64    FORMAT (38H0FLOW NOT POSSIBLE FROM SOURCE TO SINK)
65    FORMAT (35H0SOMETHING WRONG WITH INPUT NETWORK)
66    FORMAT (16H CHECK ARC FROM ,I2,3H TO,I3)
67    FORMAT (23H PROGRAM NET TERMINATED)
68    FORMAT (35H0ANALYSIS CONTROL CARD IS INFESIBLE)
69    FORMAT (26H0****FLOW ASSIGNMENTS*****)
      END
      SUBROUTINE SHORT
C     THIS SUBROUTINE FINDS THE SHORTEST PATH BETWEEN
      NODES SPECIFIED ON CONTROL CARD
C     USING A DYMANIC PROGRAMMING APPROACH
      COMMON I(200), J(200), JWIN(100), KADJ(200), KARC(200), KASG(200),
     1 KCUM(200), KFLOW(200), KIST(100). LABEL(200), MAX(100), ITERM, JF
     2LAG, KALL, KKEND, KFL, KFST, KEND, LOOP, NA, NIN, NOUT
C
C     LABEL(II) CONTAINS A CODE FOR STATUS OF ARC
C     0 = UNEVALUATED
C     1 = UNDER CONSIDERATION
```

```
C     5 = ON SHORTEST PATH(S)
C     9 = ELIMINATED
C     JWIN(II) CONTAINS THE ELEMENT NUMBERS OF MULTIPLE
C     POTENTIAL AND ACTUAL SHORTEST PATHS
C     KIST(II) CONTAINS DISTANCE FROM TERMINAL TO THAT NODE
C     KCUM(II) CONTAINS CUMMULATIVE DISTANCE FROM SOURCE
C
C     ELIMINATE ARCS LEADING TO SOURCE AND FROM TERMINAL
      DO 2 II = 1,NA
      IF (J(II) .NE. KFST) GO TO 1
      LABEL(II) = 9
 1    IF (I(II) .NE. KEND) GO TO 2
      LABEL(II) = 9
 2    CONTINUE
C     INITIALIZE KIST AND KCUM
      DO 3 II = 1, NA
 3    KCUM(II) = 0
      DO 4 II = 1, 100
 4    KIST(II) = 999999
      KKK = KEND+1
      KIST(KKK) = 0
C
C     BACKWARD PASS TO FIND DISTANCE TO EACH NODE
C     REMAINING DISTANCE FOR EACH NODE STORED IN KIST
      DO 6 II = 1, 100
      JALL = 0
      DO 5 KK = 1, NA
      JJ = NA-KK+1
      IF (LABEL(JJ) .EQ. 9) GO TO 5
      KI = I(JJ)+1
      KJ = J(JJ)+1
      IF (KIST(KI) .LE. (KARC(JJ)+KIST(KJ))) GO TO 5
      KIST(KI) = KIST(KJ)+KARC(JJ)
      JALL = 9
 5    CONTINUE
      IF (JALL .EQ. 0) GO TO 7
C     CONTINUE COMPUTING CUMMULATIVE DISTANCES UNTIL NO MORE SHIFTS
 6    CONTINUE
C     NEGATIVE CYCLE IF THIS POINT REACHED
      INFES = 9
      KALL = 0
      GO TO 15
C
C     FORWARD PASS TO FIND ARCS ON SHORTEST PATH
C     INITIALIZE SEARCH AT KFST
 7    KALL = 0
      JW = 1
      JWIN(JW) = KFST
C     FIND NODES TO BE CONSIDERED NEXT
C     POTENTIAL WINNERS LABEL SET AT 1
 8    INFES = 9
      DO 10 II = 1, JW
      IJK = JWIN(II)
      DO 9 JJ = 1, NA
      IF (I(JJ) .NE. IJK) GO TO 9
      IF (LABEL(JJ) .GT. 1) GO TO 9
      LABEL(JJ) = 1
      INFES = 0
 9    CONTINUE
 10   CONTINUE
C     CHECK IF ANY MORE NODES TO BE CONSIDERED
C     IF NO MORE NODES --SHUT DOWN
      IF (INFES .EQ. 9) GO TO 15
C
C     FIND WINNERS AMONG LABELED NODES
C     ACTUAL WINNERS LABEL SET AT 5
```

```
      JW = 0
      DO 14 II = 1,NA
      IF (LABEL(II) .NE. 1) GO TO 14
      KI = I(II)+1
      KJ = J(II)+1
      KADD = KIST(KI)-KIST(KJ)
      IF (KARO(II) .GT. KADD) GO TO 13
      LABEL(II) = 5
      KKK = KFST+1
      KCUM(II) = KIST(KKK)-KIST(KJ)
      JW = JW+1
      JWIN(JW) = J(II)
      IF (J(II) .EQ. KEND) GO TO 12
C     ELIMINATE ARCS LEADING TO THIS WINNER
      DO 11 JJ = 1, NA
      IF (J(JJ) .NE. J(II)) GO TO 11
      IF (LABEL(JJ) .EQ. 5) GO TO 11
      LABEL(JJ) = 9
 11   CONTINUE
      GO TO 14
C     SET FLAG IF TERMINAL NODE REACHED
 12   KALL = 9
      GO TO 14
 13   KCUM(II) = 999999
      LABEL(II) = 9
 14   CONTINUE
C     GO TO FIND NODES NEXT CONSIDERED
      GO TO 8
C
C     SHORT TERMINATED
C     SET FLAGS FOR ENDING CONDITIONS AND RETURN
 15   IF (KALL .EQ. 0) GO TO 16
      NORMAL ENDING,TERMINAL NODE REACHED
      LOOP = LOOP+1
      GO TO 18
 16   IF (LOOP .GT. 0) GO TO 17
C     NO PATH FOUND FIRST TIME THROUGH
      ITERM = 40
      GO TO 18
C     NO FUTHER PATHS POSSIBLE, N-TH ITERATION
 17   ITERM = 5
 18   RETURN
      END
      SUBROUTINE FLOW
C     THIS SUBROUTINE ASSIGNS FLOW TO A NETWORK
C     BY USE OF THE SHORTEST PATH
C     AND CAPACITATING REVERSE FLOW IN ARCS WITH
C     POSITIVE FLOW ASSIGNMENT
C     MAXIMUM FLOW IS FOUND BY ASSIGNING FLOW
C     UNTIL THE ENTIRE NETWORK IS SATURATED
      COMMON I(200), J(200), JWIN(100), KADJ(200), KARC(200), KASG(200),
     1 KCUM(200), KFLOW(200), KIST(100), LABEL(200), MAX(100), ITERM, JF
     2LAG, KALL, KKEND, KFL, KFST, KEND, LOOP, NA, NIN, NOUT
C
C     KADJ(200)  TEMPORARY FLOW ASSIGNMENT TO ARC
C     KASG(200)  REMAINING ARC CAPACITY
C     KFLOW(200)  CUMMULATIVE FLOW ASSIGNMENT TO ARC

C     SET UP MIRROR IMAGE OF NETWORK
      DO 1 N = 1, NA
      NAX = N+NA
      I(NAX) = J(N)
      J(NAX) = I(N)
      KARC(NAX) = 99999
      LABEL(NAX) = 9
      KADJ(N) = MAX(N)
 1      KADJ(NAX) = 0
        NA1 = NA+1
C       INCREASE SIZE OF NETWORK TO INCLUDE REVERSE (MIRROR) ARCS
 NAX =NAX = NA
       *NA = NA+NA
C       INITALIZE THE STORAGE VECTORS
        DO 2 N = 1, NA
        KFLOW(N) = 0
 2      KASG(N) = 0
        IF (LOOP .EQ. 1) GO TO 4
C
C       MAIN LOOP BEGINS HERE
C       CALL SHORT TO FIND SHORTEST (CHEAPEST) PATH
 3      CALL SHORT
C       CHECK IF FESIBLE PATH FOUND BY SHORT
        IF (ITERM .EQ. 50) GO TO 24
C
C       FIND NEXT NODE IN FESIBLE PATH
C       STARTING AT SINK AND WORKING BACK
C       MAKE TEMPORARY ASSIGNMENT OF FLOW TO PATH
 4      DO 5 IX = 1, NA
        IF (J(IX) .NE. KEND) GO TO 5
        IF (LABEL(IX) .EQ. 5) GO TO 6
 5      CONTINUE
 6      NEXT = I(IX)
        KASG(IX) = KADJ(IX)
 7      IF (NEXT .EQ. KFST) GO TO 10
        IPOS = NA+1
        DO 8 KX = 1, NA
        IP = IPOS-KX
        IF (J(IP) .NE. NEXT) GO TO 8
        IF (LABEL(IP) .EQ. 5) GO TO 9
 8      CONTINUE
 9      NEXT = I(IP)
        KASG(IP) = KADJ(IP)
        GO TO 7
        CONTINUE
C
C       FIND SMALLEST ARC CAPACITY
C       ADJUST FLOW TO SMALLEST ARC CAPACITY
        NMAX = 99999
        DO 11 KU = 1, NA
        IF (KASG(KU) .EQ. 0) GO TO 11
        IF (KASG(KU) .GT. NMAX) GO TO 11
        NMAX = KASG(KU)
 11     CONTINUE
        IF (NMAX .LT. KFL) GO TO 12
        NMAX = KFL
 12     DO 13 KU = 1, NA
        IF (KASG(KU) .EQ. 0) GO TO 13
        KASG(KU) = NMAX
 13     CONTINUE
C
C       REDUCE DEMAND BY FLOW ASSIGNMENT
        KFL = KFL-NMAX
C       CHECK IF FLOW DEMANDED IS SATISFIED
        IF (KFL .GT. 0) GO TO 14
        ITERM = 30
 14     CONTINUE
C
C       UPDATE CUMMULATIVE FLOW ASSIGNMENTS IN KFLOW
        DO 15 NZ = 1, NAX
        NZX = NZ+NAX
 15     KFLOW(NZ) = KFLOW(NZ)+KASG(NZ)-KASG(NZX)
C       UPDATE REMAINING CAPACITY
        DO 16 NP = 1, NAX
```

```
      NPX = NP+NAX
      KADJ(NPX) = 0
   16 KADJ(NP) = MAX(NP)-KFLOW(NP)
C     LABEL = 9 FOR SATURATED ARCAS
      DO 18 IZ = 1, NAX
      IF (KADJ(IZ) .NE. 0) GO TO 17
      LABEL(IZ) = 9
      GO TO 18
   17 LABEL(IZ) = 0
   18 CONTINUE
C
C     CAPACITATE REVERSE FLOW UP TO POSITIVE FLOW ASSIGNMENT
C     BUT AT NEGATIVE OF COST
      DO 20 IZ = 1, NAX
      IZX = IZ+NAX
      IF (KFLOW(IZ) .EQ. 0) GO TO 19
      KADJ(IZX) = KFLOW(IZ)
      LABEL(IZX) = 0
      KARC(IZX) = -KARC(IZ)
      GO TO 20
   19 KADJ(IZX) = 0
      LABEL(IZX) = 9
      KARC(IZX) = 99999
   20 CONTINUE
C
C     PRINT OUT INTERMEDIATE RESULTS IF REQUESTED BY JFLAG
      IF (JFLAG .EQ. 0) GO TO 22
      WRITE (NOUT,25) LOOP, NMAX
      DO 21 II = 1, NA
      IF (KASG(II) .EQ. 0) GO TO 21
      WRITE (NOUT.26) I(II), J(II)
   21 CONTINUE

C     IF FLOW DEMAND IS SATISFIED, QUIT
   22 IF (ITERM .EQ. 30) GO TO 24
C     FLOW NOT SATISFIED, RETURN FOR NEXT ITERATION
      DO 23 KT = 1, NA
   23 KASG(KT) = 0
      GO TO 3
   24 RETURN
C
   25 FORMAT (10H ITERATION,I3,8H FLOW =,I5)
   26 FORMAT (1X,I2,3X,I2)
      RETURN
      END
```

```
// JOB
// FOR
*ONE WORD INTEGERS
*IOCS(CARD,1403 PRINTER)
*NAME MOCDE
*SAVE
C*********************************************************************
C*                                                                   *
C*                                                                   *
C*                         PROGRAMA  MOCOE.                          *
C*                                                                   *
C*        MODELO CUANTITATIVO DE ORDEN ECONOMICO.                    *
C*        PROGRAMA PREPARADO POR LA SECCION DE COMPUTACION           *
C*        DE LA FACULTAD DE INGENIERIA DE LA ** U.N.A.M **           *
C*        OCTUBRE DE  1974                                           *
C*                                                                   *
C*                                                                   *
C*                                                                   *
C*            VARIABLES QUE INTERVIENEN EN EL PROGRAMA               *
C*                                                                   *
C*    R= REQUERIMIENTO DEL USO ANUAL, NIVEL DE DEMANDA               *
C*    CP= COSTO DE UNA ORDEN DE COMPRA.                              *
C*    FH= COSTO DADO (SOSTENIDO) COMO PORCENTAJE DEL                 *
C*    PRECIO UNITARIO                                                *
C*    P(1)= PRECIO POR UNIDAD ANTES DEL DESCUENTO, P(1)=P1           *
C*    P(2)= PRECIO POR UNIDAD EN EL MOMENTO QUE OCURRE EL PRIMER     *
C*    DESCUENTO.                                                     *
C*    P(3)= PRECIO POR UNIDAD DESPUES DEL PRIMER DESCUENTO,          *
C*    P(3)= P2                                                       *
C*    P(4)= PRECIO POR UNIDAD EN EL MOMENTO QUE OCURRE EL            *
C*    SEGUNDO DESCUENTO.                                             *
C*    P(5)= PRECIO POR UNIDAD DESPUES DEL SEGUNDO DESCUENTO,         *
C*    P(5)=  P3                                                      *
C*    P(6)= :RESTRICCION: PRECIO POR UNIDAD A LA CAPACIDAD           *
C*    DE LA BODEGA.                                                  *
C*    ECOQ= LOTE ECONOMICO A SER ORDENADO.                           *
C*    B(1)= PUNTO DONDE OCURRE EL PRIMER DESCUENTO, B(1)=B1          *
C*    B(2)= PUNTO DONDE OCURRE EL SEGUNDO DESCUENTO, B(2)=B2         *
C*    CS= COSTO EN CASO DE ESCASEZ, (DEFICIT).                       *
C*    W= ESPACIO MAXIMO DISPONIBLE EN LA BODEGA.                     *
C*    TCST= COSTO TOTAL DEL LOTE ECONOMICO.                          *
C*    Q(3)= LOTE ECONOMICO PARA P(3)                                 *
C*    Q(1)= LOTE ECONOMICO PARA P(1)                                 *
C*    Q(5)= LOTE ECONOMICO PARA P(5)                                 *
C*    Q(2)= B(1)                                                     *
C*    Q(4)= B(2)                                                     *
C*    Q(6)= W PARA EL CASO EN QUE CS=0                               *
C*    Q(6)= LOTE OPTIMO PARA ENV=(6) = W CUANDO CS NO ES 0           *
C*    TC(I)= COSTO TOTAL.                                            *
C*    TC(I)= COSTO TOTAL PARA Q(I) Y ENV(I) PARA I=1,6               *
C*    ENV(I)= NIVEL ECONOMICO DE INVENTARIO PARA CUANDO EL LOTE      *
C*    ECONOMICO = Q(I)  I=1,5                                        *
C*    ENV(6)= W                                                      *
C*    ENVT= INVENTARIO OPTIMO CUANDO CS NO ES CERO.                  *
C*                                                                   *
C*                                                                   *
C*********************************************************************
C
C
C
C     LAS TARJETAS DE DATOS SON DOS :
C     I) IDENTIFICACION, FORMATO  10A4
C     II) DATOS : R,CP,FH,P(1),CS,B(1),P(3),B(2),P(5),W, FORMATO 11F5.0
C
```

```
C     AL TERMINAR DE PROCESAR UN JUEGO DE DATOS
C     EL PROGRAMA REGRESA A LEER OTRO JUEGO.
C
      DIMENSION P(6),Q(6),ENV(6),TC(6),B(3),ALPHA(10)
      CALL IOCS1(5,3)
C
C     LEE Y ESCRIBE LA PRIMERA TARJETA DE DATOS
C
    1 READ(2,71) ALPHA
C
C     LEE Y ESCRIBE LA SEGUNDA TARJETA DE DATOS
C
      READ(2,72) R,CP,FH,P(1),CS,B(1),P(3),B(2),P(5),W
      WRITE(3,73) ALPHA
      WRITE(3,76) R,CP,FH,P(1),CS,B(1),P(3),B(2),P(5),W
C
C     CHEQUEO DE DATOS, SI  R=0, TERMINA EL PROGRAMA.
C
      IF (R) 19,100,2
    2 IF (CP) 19,19,3
    3 IF (FH) 19,19,4
    4 IF (P(1)) 19,19,5
    5 IF (B(1)) 19,14,6
    6 IF (P(3)) 19,19,7
    7 IF (B(2)) 19,16,8
    8 IF (B(2)-B(1)) 19,9,9
C
C     DOS PUNTOS DE CAMBIO DE PRECIO.
C
    9 NSEG=3
   10 IF (P(5)) 19,19,11
   11 IF (W) 19,12,13
   12 W=1.E25
   13 B(3)=W
      IF (CS) 19,20,20
C
C     NO HAY PUNTO DE CAMBIO DE PRECIO.
C
   14 NSEG=1
      B(1)=1.E25
      IF (P(3)) 19,15,19
   15 P(3)=P(1)
      IF (B(2)) 19,17,19
C
C     UN PUNTO EN EL CAMBIO DE PRECIO.
C
   16 NSEG=2
   17 B(2)=1.E25
      IF (P(5)) 19,18,19
   18 P(5)=P(3)
      GO TO 10
C
C     ERROR EN LOS DATOS
C
   19 WRITE(3,77)
      GO TO 1
   20 P(4)=P(5)
      P(2)=P(3)
      IF (P(5)-P(3)) 22,22,21
   21 P(4)=P(3)
   22 IF (P(3)-P(1)) 24,24,23
   23 P(2)=P(1)
   24 P(6)=P(2)
      IF (W-B(1)) 28,30,25
   25 IF (W-B(2)) 27,29,26
   26 P(6)=P(5)
```

```
      GO TO 30
   27 P(6)=P(3)
      GO TO 30
   28 P(6)=P(1)
      GO TO 30
   29 P(6)=P(4)
   30 Q(2)=B(1)
      Q(4)=B(2)
      Q(6)=B(3)
      IF (CS) 31,31,61
C
C     NO SE PERMITEN DEFICITS.
C
   31 DO 32 I=1,3
      J=2*I-1
C*    ON= NUMERO DE ORDENES POR A@O DEL LOTE ECONOMICO.                    *
C
C     CALCULA Q(I) I=1,3,5
C
   32 Q(J)=SQRT(2.*CP*R/(FH*P(J)))
      DO 35 I=1,6
      IF (Q(I)-1.E25) 33,34,33
C
C     CALCULA TC(I)  I=1,6
C
   33 TC(I)=(CP*R/Q(I)+P(I)*R+P(I)*FH*Q(I)/2.)
      GO TO 35
   34 TC(I)=1.E25
   35 CONTINUE
      DO 36 I=1,6
   36 ENV(I)=Q(I)
C
C     PRUBA DE FACTIBILIDAD CON RESPECTO A LOS PUNTOS DE
C     CAMBIO DE PRECIO.
C
   37 IF (Q(1)-Q(2)) 39,39,38
   38 TC(1)=1.E25
   39 IF (Q(2)-Q(3)) 40,40,41
   40 IF (Q(3)-Q(4)) 42,42,41
   41 TC(3)=1.E25
   42 IF (Q(4)-Q(5)) 44,44,43
   43 TC(5)=1.E25
C
C     ENCUENTRE EL LOTE OPTIMO SIN DEFICIT.
C
   44 KFLB=1
      TCST=TC(1)
      ECOQ=Q(1)
      ENVT=ENV(1)
      DO 46 I=1,5
      IF (TCST-TC(I)) 46,46,45
   45 TCST=TC(I)
      KFLB=I
      ECOQ=Q(1)
      ENVT=ENV(I)
   46 CONTINUE
      ON=R/ECOQ
C
C     IMPRIMA RESULTADOS ANTES DE LAS LIMITACIONES POR DEFICIT.
C
      WRITE(3,78)
      IF (W-1.E25) 47,48,47
   47 WRITE(3,79)
   48 WRITE(3,80) ECOQ
      IF (CS) 49,50,49
   49 WRITE (3,90) ENVT
   50 WRITE (3,81) P(KFLB)
      WRITE (3,82) TCST
      WRITE (3,83) ON
      IF (W-1.E25) 51,1,51
C
C     PRUEBA DE FACTIBILIDAD CON RESPECTO A LAS LIMITACIONES DEL
C     DEFICIT.
C
   51 DO 53 I=1,5
      IF (ENV(I)-W) 53,53,52
   52 TC(I)=1.E25
   53 CONTINUE
      IF (TC(KFLB)-1.E25) 54,55,54
   54 WRITE (3,84)
      GO TO 1
C
C     ENCUENTRE EL LOTE OPTIMO, CON LAS LIMITACIONES DEL DEFICIT.
C
   55 KFLB=1
      TCST=TC(1)
      ECOQ=Q(1)
      ENVT=ENV(1)
      DO 57 I=1,6
      IF (TCST-TC(I)) 57,57,56
   56 TCST=TC(I)
      ECOQ=Q(I)
      KFLB=I
      ENVT=ENV(I)
   57 CONTINUE
      ON=R/ECOQ
C
C     IMPRIMA LOS RESULTADOS DESPUES DE LAS LIMITACIONES DEL
C     DEFICIT
C
      WRITE (3,85)
      WRITE (3,86)
      WRITE (3,87)
      WRITE (3,78)
      WRITE (3,88)
      WRITE (3,80) ECOQ
      IF (CS) 58,59,58
   58 WRITE(3,90) ENVT
   59 CONTINUE
      WRITE(3,81) P(KFLB)
      WRITE(3,82) TCST
      WRITE(3,83) ON
      IF (KFLB-6) 1,60,1
   60 WRITE(3,89)
      GO TO 1
C
C     DEFICITS PERMITIDOS.
C
   61 ENV(6)=Q(6)
      DO 62 I=1,5,2
      ENV(I)=SQRT(2.*CP*R*CS/(FH*P(I)*(FH*P(I)+CS)))
   62 Q(I)=SQRT((2.*CP*R*(FH*P(I)+CS))/(FH*P(I)*CS))
      DO 65 I=1,2
      J=2*I
      IF (Q(J)-1.E25) 64,63,64
   63 ENV(J)=1.E25
      GO TO 65
   64 FNV(J)=(Q(J)*CS)/(P(J)*FH+CS)
   65 CONTINUE
      IF (ENV(6)-1.E25) 66,67,66
   66 Q(6)=SQRT((2.*CP*R+ENV(6)**2*(CS+P(6)*FH))/CS)
   67 DO 70 I=1,6
```

```
      IF (Q(I)-1.E25) 68,69,68
   68 TC(I)=(CP*R/Q(I)+FH*P(I)*ENV(I)**2/(2.*Q(I))+R*P(I)+(CS*(Q(I)-E
     1NV(I))**2)/(2.*Q(I)))
      GO TO 70
   69 TC(I)=1.E25
   70 CONTINUE
      GO TO 37

C
C     TABLA DE FORMATOS.
C
C
   71 FORMAT (10A4)
   72 FORMAT (11F5.0)
   73 FORMAT (42H1              ---------- PROGRAMA MOCOE PARA ,10A4,
     112H ---------- ///
     248H ********** LOS DATOS DE ENTRADA SON ********** //
     362H         R           CP           FH           P1           CS           B1           P2,
     427H         B2          P3          W )
   76 FORMAT (3X,F6.0,1X,9F9.2)
   77 FORMAT (49H0ERROR EN LOS DATOS DE ENTRADA, VERIFICA Y PRUEBA,
     110H OTRA VEZ )
   78 FORMAT (54H0*************** LOS RESULTADOS DEL ANALISIS SON *****,
     111H********** )
   79 FORMAT (48H0ANTES DE APLICAR EL LIMITE DE ALMACENAMIENTO DE,
     111H LA BODEGA )
   80 FORMAT (54H LA CANTIDAD OPTIMA ORDENADA ES DE -------------------,
     117H-----------------,F10.2)
   81 FORMAT (54H AL PRECIO UNITARIO DE ------------------------------,
     117H-----------------,F10.2)
   82 FORMAT (54H OBTENIENDO UN COSTO TOTAL DE INVENTARIO DE ----------,
     117H-----------------,F10.2)
   83 FORMAT (54H DONDE EL NUMERO DE CICLOS DE ORDENES POR A@O ES DE --,
     117H-----------------,F10.2)
   84 FORMAT (48H0EL LIMITE DE LA BODEGA NO TUBO EFECTO EN MOCOE )
   85 FORMAT (51H0LA CANTIDAD ORDENADA ESTA LIMITADA POR EL ESPACIO ,
     120HFISICO DE LA BODEGA )
   86 FORMAT (48H Y ESTA RESTRICCION NO ES OPTIMA. SE ELIMINA LA )
   87 FORMAT (54H RESTRICCION Y SE CORRE EL PROGRAMA DE NUEVO, OBSERVE ,
     110HEL EFECTO.)
   88 FORMAT (53H DESPUES DE HABER APLICADO LA LIMITACION A LA BODEGA )
   89 FORMAT (52H ESTA ORDEN ESTA SUJETA A LA CAPACIDAD MAXIMA DE LA ,
     17HBODEGA )
   90 FORMAT (31H CON UN INVENTARIO OPTIMO DE : ,50X,F10.2)
C
C
  100 CONTINUE
      CALL EXIT
      END
```

```
1     PROGRAM DYNAM
C     SOLVES DYNAMIC PROGRAMMING PROBLEMS
C     SIZE UP 10 9 STAGES AND 30 STATES
C     WILLIAM G. LESSO,DEC 1971    REVISED RDH 4/73
C     THIS VERSION FOR CDC 3100
      DIMENSION ITLE(10), G(9,30), F(2,30), X(30), FX(9,30), XOPT(9,30),
     1 XOUT(9), IH(12)
      COMMON /DATA/IDAT(12)
      DATA (IDAT=1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,3HMAX,3HMIN,4HSTOP)
      INTEGER X,XOPT,XOUT
      DO 1 I=1,12
1     IH(I)=IDAT(I)
      ITOT=0
      NIN=5
      NOUT=6
C
C     READ AND WRITE USER NAME CARD
2     READ (NIN,21) ITLE
      WRITE (NOUT,22) ITLE
      IF (ITLE(1).EQ.IH(12)) GO TO 20
C
C     READ AND WRITE DATA CONTROL CARD
      READ (NIN,23) MCODE,MSTAGE,STATE,MCOPY
      MSTATE=STATE
      WRITE (NOUT,24) MCODE,MSTAGE,MSTATE,MCOPY
      IF (MSTAGE.LT.2) GO TO 17
      IF (MSTATE.GT.30) GO TO 19
C
C     READ AND WRITE STATE RETURNS
      WRITE (NOUT,25) MSTATE
      DO 3 I=1,MSTAGE
      READ (NIN,26) (G(I,J),J=1,MSTATE)
      II=MSTAGE-I+1
      WRITE (NOUT,28) IH(II)
3     WRITE (NOUT,27) (G(I,J),J=1,MSTATE)
      IF (MCODE.EQ.IH(11)) GO TO 5
      IF (MCODE.NE.IH(10)) GO TO 13
      DO 4 I=1,MSTAGE
      DO 4 J=1,MSTATE
      G(I,J)=-G(I,J)
      G(I,J)=-G(I,J)
4     CONTINUE
5     CONTINUE
      DO 6 J=1,MSTATE
      X(J)=0.
      F(1,J)=0.
6     CONTINUE
      WRITE (NOUT,29) ITLE
C
C     MAIN DO LOOP THROUGH EACH STAGE
      DO 14 ISG=1,MSTAGE
      DO 8 I=1,MSTATE
      F(2,I)=99999.
      X(I)=0.0
      II=I
      DO 7 J=1,II
      K=II-J+1
      SUM=G(ISG,J)+F(1,K)
      IF (SUM.GE.F(2,II) GO TO 7
      F(2,II)=SUM
      X(II)=J-1
7     CONTINUE
8     CONTINUE
      IF (MCOPY.GT.0) GO TO 9
      II=MSTAGE-ISG+1
C     WRITE OUT CURRENT RESULTS
      WRITE (NOUT,30) IH(II)
      WRITE (NOUT,31)
9     DO 10 I=1,MSTATE
10    F(1,I)=F(2,I)
      IF (MCODE.EQ.IH(II)) GO TO 12
      DO 11 I=1,MSTATE
      F(2,I)=-F(2,I)
11    CONTINUE
12    CONTINUE
      DO 13 I=1,MSTATE
      XOPT(ISG,I)=X(I)
      FX(ISG,I)=F(2,I)
      IF (MCOPY.GT.0) GO TO 13
      IOUT=I-1
      WRITE (NOUT,32) IOUT,X(I),F(2,I)
13    CONTINUE
14    CONTINUE
C
C     WRITE OUT OPTIMUN RESULTS
      WRITE (NOUT,33) ITLE
      WRITE (NOUT,34)
      WRITE (NOUT,35) (IH(I),I=1,MSTAGE)
      DO 16 I=1,MSTATE
      IOUT=I-1
      ITOT=I
      DO 15 ISG=1,MSTAGE
      M=MSTAGE-ISG+1
      XOUT(ISG)=XOPT(M,ITOT)
15    ITOT=ITOT-XOUT(ISG)
      WRITE (NOUT,36) IOUT,FX(MSTAGE,I),(XOUT(J),J=1,MSTAGE)
16    CONTINUE
C     RETURN FOR NEXT DATA SET
      GO TO 2
C
C     THIS SECTION PRINTS ERROR MESSAGES
17    WRITE (NOUT,37)
      GO TO 20
18    WRITE (NOUT,39)
      GO TO 20
19    WRITE (NOUT,40)
20    WRITE (NOUT,38)
C
C
21    FORMAT (10A4)
22    FORMAT (19H1PROGRAM DYNAM FOR ,10A4)
23    FORMAT (A3,2X,I1,4X,F2.0,3X,I1)
24    FORMAT (1H0,A3,8H-MIZE    ,I1,10H STAGES    ,I2,10H STATES    ,I1)
25    FORMAT (52H0STATE(I)-----------STATE RETURNS AS READ-----------,6H
     1STATE(I2,1H)
26    FORMAT (6F5.0)
27    FORMAT (6F10.4)
28    FORMAT (7H STAGE ,A1)
29    FORMAT (18H1DYNAM RESULTS FOR ,10A4)
30    FORMAT (27H0RETURN FUNCTION FOR STAGE ,A1)
31    FORMAT (25H STATE DECISION  RETURNS)
32    FORMAT (3X,I2,4X,I2,1X,F13.4)
33    FORMAT (22H0OPTIMUN DECISION FOR ,10A4)
34    FORMAT (1H0,8X,30H DESIRED      DECISION AT STAGE)
35    FORMAT (16H STATE     RETURN,9(5X,A1))
36    FORMAT (3X,I3,F10.2,9I6)
37    FORMAT (34H0SORRY, AT LEAST 2 STAGES REQUIRED)
38    FORMAT (18H0DYNAM RUN STOPPED)
39    FORMAT (33H0CAN-T TELL IF MAX OR MIN PROBLEM)
+0    FORMAT (28H0SORRY, 30 STATES IS MAXIMUN)
      END
      PEOGRAM QUEUES
C     MARCH 1972 M J MAGGARD
C     THIS VERSION FOR CDC 3100
C     DICTIONARY OF VARIABLES
C     ATTIME         THE HOURS OF SYSTEM IDLE TIME - TOTAL
C     ANUM,SNUM     NUMBER OF READS IN ARRIVALS AND SERVICE
C     AR(500)       AN ARRAY OF READ IN ARRIVAL TIMES
C     ARR RI,ARR TM THE ARRIVAL RATE AND TIME
C     CH            AN ARRAY OF ARRIVAL AND SERVICE ON FIRST 20 CUSTOMERS
C     CIDLE         THE COST OF EYETEM IDLE TIME - TOTAL
C     CUMQUE(100)   AN ARRAY WHICH STORES IDLE CUSTOMER HOURS
C     COSTS         THE COST PER TIME UNIT OF IDLE SERVICE
C     COSTA         THE COST PER TIME UNIT OF IDLE CUSTOMERS
C     CUSERV,KCUS   THE NUMBER OF CUSTOMERS BEING SERVED
C     CWAIT         THE COST OF CUSTOMERS HOURS IN QUEUE - TOTAL
C     DEP RT,DEP TM THE SERVICE RATE AND MEAN DURATION
C     HRSNQ         THE HOURS OF CUSTOMER TIME IN QUEUE - TOTAL
C     I,J           THE CHANNEL NUMBER BEING PROCESSED
C     IZ            NUMBER OF ARRIVALS WHICH HAVE OCCURED
C     KA,KS         OPTION CODES FOR ARRIVALS AND SERVICE
C     N,MAXS        BEGINNING,MAXIMUM NUMBER CHANNELS
C     PCUTIL        THE PERCENT UTILIZATION OF THE SERVICE FACILITY
C     QUE           THE NUMBER OF CUSTOMERS IN QUEUE AT ANY POINT
C     SR(500)       AN ARRAY OF READ IN SERVICT TIMES
C     TCOOP         THE TOTAL COST OF THE SYSTEM
C     TIME,TTIME    CLOCK TIME,MAX SIMULATION TIME
C     TNARV         THE LATEST ARRIVAL TIME
C     TNDPR         THE DEPARTURE TIME OF THE LATEST DEPARTURE
C                  MAY BE SET ARTIFICALLY FOR PROGRAM EFFICIENCY
C     XMNTN,XMNTX   MEAN WAIT TIME,MEAN NUMBER IN QUE
C     AVARRA        AVERAGE ARRIVALS PER TIME UNIT
C     AVSERV        AVERAGE SERVICE TIME
C     AMNIS         MEAN NO. IN THE SYSTEM
C/    AMTIS         MEAN TIME IN THE SYSTEM
C     VCOST         TOTAL VARIABLE COST OF OPERATIONS
C     VCOSTS        VARIABLE COST PER UNIT
C     FCOST         TOTAL FIXED COST OF OPERATIONS
C     FCOSTS        FIXED COST PER UNIT
C     QSVCT         AN ARRAY OF QUEUED SERVICE TIMES
C     KRULE         SCHEDULE RULE CODE (1=RANDOM,2=FCFS,3=SOT)
      COMMON ILPHA(10),ANUM,ARRRT,ARRTM,CH(20,10),CUMUTL,CUSERV,DEPRT,
     1      DEPTM,I,IUSERV,IZ,KA,KCUS,KS,N,NFLAG,SNUM,STATUS(9),T,
     2      TIME,TTIME,TNARV,TNDPR(9),AVARRA,AVSERV,AMNIS,AMTIS,
     3      VCOST,VCOSTS,FCOST,FCOSTS,KRULE,IQUE,CUMQUE(101),
     4      QSVCT(101),AR(500),SR(500)
      COMMON/DATA/ IEND
      DATA (IEND=4HSTOP)
      ISTOP=IEND
      NIN=5
      NOUT=6
C     READ AND PRINT USER NAME CARD
 11   CONTINUE
      READ (NIN,311) ILPHA
      WRITE (NOUT,321) ILPHA
         IF (ILPHA(1).EQ.ISTOP) GO TO 301
C     READ AND PRINT ARRIVAL DATA CARD AND SCHEDULE RULE CODE
      READ (NIN,331) KA,ARRRT,COSTA,KRULE,ANUM
      WRITE (NOUT,341) KA,ARRRT,COSTA,KRULE
         IF (KA.LT.1) GO TO 291
         IF (KA.GT.4) GO TO 291
         IF (KRULE.LT.1) GO TO 291
         IF (KRULE.GT.3) GO TO 291
         GO TO (21,31,41), KRULE
 21   CONTINUE
```

```
      WRITE (NOUT,351)
         GO TO 51
 31   CONTINUE
      WRITE (NOUT,361)
         GO TO 51
 41   CONTINUE
      WRITE (NOUT,371)
C     READ ARRIVAL STATISTICS
 51   CONTINUE
         IF (ANUM.LE.0.0) GO TO 71
      NUM=ANUM
      ARRRT=1.234
      KA=4
      WRITE (NOUT,391) NUM
         IF (ANUM.GT.500.0) GO TO 291
      READ (NIN,381) (AR(I),I=1,NUM)
      WRITE (NOUT,401) (AR(I),I=I,NUM)
      DO 61 I=2,NUM
      J=I-1
         IF (AR(J).GT.AR(I)) GO TO 291
 61   CONTINUE
C     SET ARRIVAL TIME AT INVERSE OF ARRIVAL RATE
 71   CONTINUE
         IF (ARRRT.LE.0.0) GO TO 291
      ARRTM=1.0/ARRRT
C     READ AND PRINT SERVICE DATA CARD
      READ (NIN,411) KS,DEPTM,FCOSTS,VCOSTS,SNUM
      WRITE (NOUT.421) KS,DEPTM
      WRITE (NOUT,431) FCOSTS,VCOSTS
         IF (KS.EQ.0) GO TO 291
         IF (KS.GT.4) GO TO 291
         IF (SNUM.LE.0.0) GO TO 91
      NUM=SNUM
      DEPTM=1.234
      KS=4
      WRITE (NOUT.441) NUM
         IF (SNUM.GT.500.0) GO TO 291
      READ (IN,381) (SR(I),I=1.NUM)
      WRITE (NOUT,401) (SR(I),I=1,NUM)
      DO 81 I=1,NUM
         IF (SR(I).LT.0.0) GO TO 291
 81   CONTINUE
C     SET SERVICE RATE AT INVERSE OF SERVICE TIME
 91   CONTINUE
         IF (DEPTM.LE.0.0) GO TO 291
      DEPRT=1.0/DEPTM
C     READ SIMULATION CONTROL CARD AND PRINT
      READ (NIN,451) N,MAXS,TTIME
      WRITE (NOUT,461) N,MAXS,TTIME
C     CHECK SIMULATION RUN LIMITS
         IF (N.EQ.0) GO TO 291
         IF (MAXS.LT.N) GO TO 291
         IF (TTIME.LE.0.0) GO TO 291
C     END OF INPUT DATA CHECK
      BCOUP=999999.9
C     SIMULATION OF A GIVEN NUMBER OF CHANNELS (N) BEGINS HERE
C     INITIALIZE SYSTEM FOR NEXT SIMULATION RUN
      CONTINUE
      TIME=0.0
      TNARV=0.0
      IQUE=1
      CUMUTL=0.0
      CUSERV=0.0
      IZ=0
      KCUS=0
      NFLAG=0
      IUSERV=0
      SET=RAND(1234567.0)
      DO 111 M=1,100
      QSVCT(M)=0.0
      CUMQUE(M)=0.0
 111  CONTINUE
C       DETERMINE SERVICE TIME FOR THE 1ST ARRIVAL
          GO TO (121,131,141,151), KS
C      POISSON SERVICE RATE
 121  CONTINUE
      R=RAND(0,9)
      T=ABS(DEPRT*ALOG(R))
          GO TO 161
C       NEGATIVE EXPONENTIAL SERVICE TIME
 131  CONTINUE
      R=RAND(0,9)
      T=ABS(DEPTM*ALOG(R))
          GO TO 161
C       CONSTANT SERVICE TIME
 141  CONTINUE
      T=DEPTM
          GO TO 161
C     READ IN SERVICE TIME
 151  CONTINUE
      T=SR(1)
 161  CONTINUE
      QSVCT(1)=T
      DO 171 L=1,N
      TNDPR(L)=999999.9
      STATUS(L)=0.0
 171  CONTINUE
      DO 181 I=1,20
      DO 181 J=1,10
      CH(I,J)=0.0
 181  CONTINUE
C     PRINT HEADING FOR RESULTS
      WRITE (NOUT,471) N
      WRITE (NOUT,481)
       WRITE (NOUT,491)
C     SET FIRST ARRIVAL OCCURANCE AT TIME ZERO
      TNARV=0.0
      CH(1,1)=0.0
      IZ=IZ+1
C     MAIN SIMULATION BRANCH POINT
C      CHECK EACH CHANNEL IN TURN FOR POSSIBLE DEPARTURE
C     IF ALL CHANNELS ARE IDLE (TNDPR = 999999.9) THEN GO TO ARRIVE
C     IF ALL CHANNELS ARE BUSY (TNARV IS .GE. TNDPR) THEN GO TO ARRIVE
C     IF A DEPART IS NEXT (TNDPR IS .GE. TNARV) THEN GO TO DEPART
C     SET AND IVALUE KEEP MULTIPLE DEPARTURES IN CORRECT TIME SEQUENCE
 191  CONTINUE
      SET=888888.
      DO 201 I=1.N
         IF (TNDPR(I).GT.TNARV) GO TO 201
         IF (TNDPR(I).GT.SET) GO TO 201
      SET=TNDPR(I)
      IVALUE=I
 201  CONTINUE
      I=IVALUE
         IF (SET.LT.888888.) GO TO 211
      CALL ARRIVE
         GO TO 191
 211  CONTINUE
      CALL DEPART
C     ON RETURN FROM DEPART CHECK SIMULATION TIME LIMIT
         IF (TTIME.GT.TIME) GO TO 191
C     END OF SIMULATION RUN---PRINT FIRST TWENTY TRIALS
```

```
      N1=N+1
         IF (CUSERV.GE.20.0) GO TO 221
      NXX=CUSERV
         GO TO 231
 221  CONTINUE
      NXX=20.
      CONTINUE
      DO 241 I=1,NXX
      WRITE (NOUT,501) (CH(I,J),J=1,N1)
 241  CONTINUE
C     COMPUTE HOURS IN QUEUE FOR SUMMARY PRIN OUTOUT
      HRSNQ=0.0
      MAXQUE=0
      DO 261 M=2,100
         IF (CUMQUE(M).EQ.0.0) GO TO 251
      MAXQUE=M-1
 251  CONTINUE
      XM=M-1
      HRSNQ=HRSNQ+(XM*CUMQUE(M))
 261  CONTINUE
         IF (MAXQUE.LT.99) GO TO 271
      WRITE (NOUT,511)
 271  CONTINUE
         IF (NFLAG.NE.76) GO TO 281
      WRITE (NOUT,521)
 281  CONTINUE
      XN=N
      IZ=IZ-1
      XIZ=IZ
C     PRINT SUMMARY STATISTICS FOR THIS NUMBER (N) CHANNELS
      WRITE (NOUT,531) IZ,CUSERV,TIME
      WRITE (NOUT,541) MAXQUE
      XMNTX=HRSNQ/TIME
      WRITE (NOUT,551) XMNTX
      XMNTM=HRSNQ/XIZ
      WRITE (NOUT,561) XMNTM
      PCUTIL=((CUMUTL/TIME)*100.)/XN
      WRITE (NOUT,571) PCUTIL
      CWAIT=HRSNQ*COSTA
      ATTIME=(TIME*XN)-CUMUTL
C     COMPUTE AVERAGE ARRIVALS PER TIME UNIT
      TZ=IZ
      AVARRA=TZ/TIME
      WRITE (NOUT,581) AVARRA
C     COMPUTE AVERAGE SERVICE TIME
      AVSERV=CUMUTL/CUSERV
      WRITE (NOUT,591) AVSERV
C     COMPUTE MEAN NO. IN THE SYSTEM
      AMNIS=XMNTX+(AVARRA/(1.0/AVSERV))
      WRITE (NOUT,601) AMNIS
C     COMPUTE MEAN TIME IN THE SYSTEM
      AMTIS=XMNTM+AVSERV
      WRITE (NOUT,611) AMTIS
      WRITE (NOUT,621)
      WRITE (NOUT,631) HRSNQ,COSTA,CWAIT
C     COMPUTE TOTAL VARIABLE COST
      VCOST = CUSERV*VCOSTS
C     COMPUTE TOTAL FIXED COST
      FCOST=XN*FCOSTS
C     PRINT SUMMARY COST INFORMATION
      WRITE (NOUT,641) CUSERV,VCOSTS,VCOST
      WRITE (NOUT,651) FCOSTS,N,FCOST
      TCOOP=FCOST+VCOST+CWAIT
      WRITE (NOUT,661) TCOOP
C     STOP RUN IF TCOOP INCREASED FROM LAST RUN
         IF (TCOOP.GE.BCOOP) GO TO 11
```

```
C     STOP RUN IF MAXIMUM NUMBER OF SERVERS REACHED                     A 262
        + IF (N.GE.MAXS) GO TO 11                                       A 263
C     UPDATE NUMBER OF CHANNELS AND CURRENT TOTAL COST                  A 264
      N=N+1                                                             A 265
      BCOOP=TCOOP                                                       A 266
C     RETURN FOR NEXT RUN WITH MORE CHANNELS (N)                        A 267
         GO TO 101                                                      A 268
C     PRINT OUT DATA ERROR MESSAGE                                      A 269
 291  CONTINUE                                                          A 270
      WRITE (NOUT,671)                                                  A 271
      WRITE (NOUT,681)                                                  A 272
 301  CONTINUE                                                          A 273
      WRITE (NOUT,691)                                                  A 274
C                                                                       A 275
C                                                                       A 276
 311  FORMAT  (10A4)                                                    A 277
 321  FORMAT  (20H1PROGRAM QUEUES FOR ,10A4)                            A 278
 331  FORMAT  (I1,9X,F5.0,5X,F5.0,5X,I1,9X,F3.0)                        A 279
 341  FORMAT  (14H0ARRIVAL TYPE ,I1,8H RATE = ,F8.2,8H  COST =,F8.2,16H A 280
     1 SCHEDULE RULE ,I1)                                               A 281
 351  FORMAT  (54X,8H(RANDOM))                                          A 282
 361  FORMAT  (55X,6H(FCFS))                                            A 283
 371  FORMAT  (55X,5H(SOT))                                             A 284
 381  FORMAT  (12F5.0)                                                  A 285
 391  FORMAT  (1H ,I4,28H ARRIVALS READ IN AS FOLLOWS)                  A 286
 401  FORMAT  (1H ,12F5.0)                                              A 287
 411  FORMAT  (I1,9X,F5.0,5X,F10.0,F10.0,F3.0)                          A 288
 421  FORMAT  (14H0SERVICE TYPE ,I1,8H TIME = ,F8.2)                    A 289
 431  FORMAT  (8X,13H FIXED COST $,F8.2,8X,16H VARIABLE COST $,F8.2)    A 290
 441  FORMAT  (1H ,I4,27H SERVICE READ IN AS FOLLOWS)                   A 291
 451  FORMAT  (I1,9X,I1,9X,F5.0)                                        A 292
 [illegible]  FORMAT  (20H0NO. CHANNELS START ,I1,5H MAX ,I1,8X,10H MAX TIME ,F6 A 293
     1.0)                                                               A 294
 471  FORMAT  (31H0FIRST TWENTY OCCURRENCES FOR   I1,18H SERVICE CHANNE A 295
     1LS)                                                               A 296
 481  FORMAT  (8H ARRIVAL,4X,44H------DEPARTURE TIME AT CHANNEL NUMBER-- A 297
     1----)                                                             A 298
 491  FORMAT  (8H TIME---,4X,44HONE TWO THREE FOUR FIVE SIX SEVEN EIGHT A 299
     1NINE)                                                             A 300
 501  FORMAT  (1H ,F6.1,3X,9F5.1)                                       A 301
 511  FORMAT  (50H0***WARNING****QUE EXCEEDED PROGRAM LIMIT OF 99***)   A 302
 521  FORMAT  (48H0***WARNING****OUT OF DATA BEFORE TIME LIMIT****)     A 303
 531  FORMAT  (6H0AFTER,I6,8H ARRIVED,F6.0,7H SERVED,F6.0,11H TIME UNITS A 304
     1)                                                                 A 305
 541  FORMAT  (22H QUEUE-MAXIMUM LENGTH ,8X,I7)                         A 306
 551  FORMAT  (22H      -MEAN LENGTH    ,8X,F9.1)                       A 307
 561  FORMAT  (22H      -MEAN WAIT TIME ,8X,F9.1)                       A 308
 571  FORMAT  (22H SERVICE UTILIZATION  ,8X,F9.1,8H PERCENT)            A 309
 581  FORMAT  (32H AVERAGE ARRIVALS PER TIME UNIT ,F10.4)               A 310
 591  FORMAT  (32H AVERAGE SERVICE TIME           ,F10.4)               A 311
 601  FORMAT  (32H MEAN NO. IN THE SYSTEM         ,F10.4)               A 312
 611  FORMAT  (32H MEAN TIME IN THE SYSTEM        ,F10.4)               A 313
 621  FORMAT  (31H0COST INFORMATION OF OPERATIONS)                      A 314
 631  FORMAT  (20H COSTS-WAIT IN QUEUE,  F9.1,11H UNITS AT $,F6.2,4H = $ A 315
     1,F9.2)                                                            A 316
 641  FORMAT  (22H SERVICE COST VARIABLE,F7.1,11H UNITS AT $,F6.2,4H = $ A 317
     1,F9.2)                                                            A 318
 651  FORMAT  (22H SERVICE COST FIXED   ,F7.2,6H WITH ,I1,14H CHANNELS  A 319
     1= $,F9.2)                                                         A 320
 661  FORMAT  (20H0TOTAL COST OF OPERATIONS                       $,F9. A 321
     12)                                                                A 322
 671  FORMAT  (35H0****ERROR IN QUESIM DATA CARDS****)                  A 323
 681  FORMAT  (35H ****CORRECT DATA AND TRY AGAIN****)                  A 324
 691  FORMAT  (22H0QUEUES RUN TERMINATED)                               A 325
      END                                                               A 326-
      SUBROUTINE ARRIVE                                                 B   1
      COMMON ILPHA(10),ANUM,ARRRT,ARRTM,CH(20,10),CUMUTL,CUSERV,DEPRT,
     1       DEPTM,I,IUSERV,IZ,KA,KCUS,KS,N,NFLAG,SNUM,STATUS(9),T,
     2       TIME,TTIME,TNARV,TNDPR(9),AVARRA,AVSERV,AMNIS,AMTIS,
     3       VCOST,VCOSTS,FCOST,FCOSTS,KRULE,IQUE,CUMQUE(101),
     4       QSVCT(101),AR(500),SR(500)
C     THIS SUBROUTINE CALLED WHEN AN ARRIVAL IS THE NEXT OCCURANCE
C     IT UPDATES THE TIME SPENT IN QUE
C     IT UPDATES THE CLOCK TO THE TIME OF THE NEW ARRIVEL (PREVIOUSLY
C        SELECTED)
C     IT CHECKS EACH CHANNEL TO SEE IF THE NEW ARRIVEL CAN BEGIN SERVICE
C     IF A CHANNEL IS AVAILABLE IT DOES THE FIRST PART OF THE
C        DEPART PROCESSING OTHERWISE IT ADDS ONE TO THE QUE
C     LASTLY, IT SELECTS THE TIME FOR THE NEXT ARRIVAL TO OCCUR
         IF (IQUE.LT.100) GO TO 11
C     CHECK LENGTH OF QUE, IF OVER 99 HOLD AT 99
      IQUE=100
C     UPDATE HOURS SPENT IN QUEUE
   11 CONTINUE
      CUMQUE(IQUE)=CUMQUE(IQUE)+TNARV-TIME
C     UPDATE CLOCK TIME TO NEXT ARRIVAL
      TIME=TNARV
C     CHECK EACH CHANNEL, IF STATUS = 0 IT IS AVAILABLE
      DO 61 J=1,N
         IF (STATUS(J).GT.0.0) GO TO 51
C     DO FIRST PART OF THE DEPART PROCESSING
      STATUS(J)=1.0
C     SET TIME OF NEXT DEPARTURE
      TNDPR(J)=TIME+QSVCT(1)
C     STORE FIRST TWENTY DEPARTURE TIMES IN CH
      KCUS=KCUS+1
         IF (KCUS.GT.20) GO TO 21
      II=J+1
      CH(KCUS,II)=TNDPR(J)
   21 CONTINUE
C     CHECK IF OUT OF SIMULATION TIME
         IF (TTIME.LT.TNDPR(J)) GO TO 31
C     ACCUMLATE PROCESSING TIME
      CUMUTL=CUMUTL+QSVCT(1)
      CUSERV=CUSERV+1.0
         GO TO 81
C     END OF SIMULATION UPDATING
C     LAST DEPARTURE FORCED OUT AT TTIME
   31 CONTINUE
      TK=QSVCT(1)-(TNDPR(J)-TTIME)
         IF (TK.GT.0.0) GO TO 41
      TK=0.
   41 CONTINUE
      TNDPR(J)=TIME
      CUMUTL=CUMUTL+TK
         GO TO 81
   51 CONTINUE
         IF (J.GE.N) GO TO 71
   61 CONTINUE
C     ALL CHANNELS ARE BUSY, ADD ONE TO THE QUE
   71 CONTINUE
      IQUE=IQUE+1
C     SELECT ARRIVAL TIME (STORE FIRST TWENTY IN CH)
   81 CONTINUE
         GO TO (91,101,111,121), KA
C     POISSON ARRIVAL TIME DISTRIBUTION
   91 CONTINUE
```

```
 101  CONTINUE
      R=RAND(0,0)
      TNARV=ABS(ARRRT*ALOG(R))
      TNARV=TNARV+TIME
         GO TO 141
C     CONSTANT ARRIVAL TIME
 111  CONTINUE
      TNARV=TIME+ARRIM
         GO TO 141
C     READ IN ARRIVAL TIMES
 121  CONTINUE
      NUM=ANUM
      IZQQ=IZ+1
         IF (IZQQ.LE.NUM) GO TO 131
      AR(IZQQ)=777777.
      NFLAG=76
 131  CONTINUE
      TNARV=AR(IZQQ)
 141  CONTINUE
      IZ=IZ+1
         IF (IZ.GT.20) GO TO 151
      CH(IZ,1)=TNARV
C       BEGIN LOGIC TO STORE ARRIVAL AND SERVICE TIMES IN QUEUE ARRAYS
C       FOR EACH WAITING CUSTOMER/PRODUCT
C       DETERMINE SERVICE TIME FOR THE NEW ARRIVAL
 151  CONTINUE
         GO TO (161,171,181,191), KS
C     POISSON SERVICE RATE
 161  CONTINUE
      R=RAND(0,9)
      T=ABS(DEPRT*ALOG(R))
         GO TO 211
C     NEGATIVE EXPONENTIAL SERVICE TIME
 171  CONTINUE
      R=RAND(0,9)
      T=ABS(DEPTM*ALOG(R))
         GO TO 211
C     CONSTANT SERVICE TIME
 181  CONTINUE
      T=DEPTM
         GO TO 211
C     READ-IN SERVICE TIME
 191  CONTINUE
      NUM=SNUM
         IF (IZ.LE.NUM) GO TO 201
      NFLAG=76
      T=666666.
         GO TO 211
 201  CONTINUE
      T=SR(IZ)
 211  CONTINUE
      QSVCT(IQUE)=T
C       IF ONLY ONE IN QUE - NO SCHEDULING NECESSARY
         IF (IQUE.LE.2) GO TO 241
      IQQ=IQUE
C       USE SCHEDULE RULE TO REORDER THE QUEUE FOR PROCESSING
C       KRULE = 1 FOR RANDOM
C       KRULE = 2 FOR FCFS
C       KRULE = 3 FOR SOT
         GO TO (231,241,221), KRULE
C     C         SOT SCHEDULE RULE
 221  CONTINUE
         IF (IQQ.LE.2) GO TO 241
      QSVCT(IQQ)=QTS
      IQQ=IQQ-1
         GO TO 221
C      RANDOM SCHEDULE RULE
  231 CONTINUE
      QI=IQUE
      IQQ=(RAND(0,99)*QI)
      IQQ=IQQ+1
         IF (IQQ.LE.1) GO TO 241
      QTS=QSVCT(IQUE)
      QSVCT(IQUE)=QSVCT(IQQ)
      QSVCT(IQQ)=QTS
C     QUEUE IS SCHEDULED - RETURN
  241 CONTINUE
      RETURN
      END
      SUBROUTINE DEPART
      COMMON ILPHA(10),ANUM,ARRRT,ARRIN,CH(20,10),CUMUTL,CUSERV,DEPRT,
     1      DEPTM,I,IUSERV,IZ,KA,KCUS,KS,N,NFLAG,SVUM,STATUS(9),T,
     2      TIME,TTIME,TNARV,TNDPR(9),AVARRA,AVSERV,AMNIS,AMTIS,
     3      VCOST,VCOSTS,FCOST,FCOSTS,KRULE,IQUE,CUMQUE(101),
     4      QSVCT(101),AR(500),SR(500)
C     THIS SUBROUTINE PROCSSSES THE DEPARTURE OF EVERY CUSTOMER
C     IT UPDATES THE HOURS SPENT IN THE QUE
C     IT UPDATES THE CLOCK TO THE NEXT DEPARTURE TIME (PREVIOUSLY
C        SELECTED)
C     IT CHECKS THE LENGTH OF THE QUE
C     IF NO ONE IN THE QUE IT SETS THE CHANNEL AT AN IDEE STATUS (THIS
C        DEPARTURE WAS PREVIOUSLY PARTIALLY PROCESSED EITHER AT
C        ARRIVE OR BY A PRIOR PASS THROUGH DEPART)
C     IF A QUE EXIST THEN TAKE ONE FROM QUE, ITS DEPARTURE TIME,
C        SET THE CHANNEL AT A BUSY STATUS AND RETURN
C     CHECK LENGTH OF QUE, IF OVER 99 HOLD AT 99
         IF (IQUE.LT.100) GO TO 11
      IQUE=100
C     UPDATE THE HOURS SPENT IN QUEUE
   11 CONTINUE
      CUMQUE(IQUE)=CUMQUE(IQUE)+TNDPR(I)-TIME
C     UPDATE THE CLOCK TO NEXT DEPARTURE TIME
      TIME=TNDPR(I)
         IF (IQUE.GT.1) GO TO 21
C     THIS SECTION COMPLETES THE PROCESSING OF A CUSTOMER
C     WHEN NO ONE IS WAITING IN THE QUE
      STATUS(I)=0.0
      TNDPR(I)=999999.9
      RETURN
C     THIS SECTION DOES THE DEPART PROCESSING
C     WHEN THE CHANNEL HAS BEEN BUSY
C     SET NEXT DEPARTURE TIME
   21 CONTINUE
      TNDPR(I)=TIME+QSVCT(1)
C     STORE FIRST TWENTY DEPARTURE TIMES IN CH
      KCUS=KCUS+1
         IF (KCUS.GT.20) GO TO 31
      II=I+1
      CH(KCUS,II)=TNDPR(I)
   31 CONTINUE
C     CHECK IF OUT OF SIMULATION TIME
         IF (TTIME.LT.TNDPR(I)) GO TO 51
C     RESET STATUS BACK TO BUSY AND RETURN
      CUMUTL=CUMUTL+QSVCT(1)
      CUSERV=CUSERV+1.0
      STATUS(I)=1.0
C     SHIFT SERVICE QUEUE UP ONE POSITION
      DO 41 II=1,IQUE
      III=II+1
      QSVCT(IT)=QSVCT(III)
 41   CONTINUE
      QSVCT(IQUE)=0.0
C     SUBRTACT ONE FROM QUE
      IQUE=IQUE-1
      RETURN
C     ADJ   T AND CUMUTL AT TERMINATION OF SIMULATION
C     LAST CUSTOMER FORCED TO DEPART AT TTIME
 51   CONTINUE
      TK=QSVCT(1)-(TNDPR(1)-TTIME)
         IF (IK.GT.0.0) GO TO 61
      IK=0.
 61   CONTINUE
      TNDPR(I)=TTIME
      CUMUTL=CUMUTL+IK
      STATUS(I)=1.0
      RETURN
      END
      FUNCTION RAND (K,KK)
C     MACHINE DEPENDENT RANDOM NUMBER GENERATOR (0 TO 1)
C     THIS VERSION FOR CDC 3100
C     K SET AT POSITIVE ODD INTEGER TO INITIALIZE
C     K SET AT ZERO TO CONTINUE STRING OF RANDOM NUMBERS
C     SEE NAYLOR,COMPUTER SIMULATION TECHNIQUES,WILEY +SONS,1966
         IF (K) 21,21,11
 11   CONTINUE
      N=K
      NN=K
      NNN=K
 21   CONTINUE
         IF (KK) 31,31,61
      CONTINUE
      N=N*2051
         IF (N) 41,51,51
 41   CONTINUE
      N=N+8388607+1
 51   CONTINUE
      XN=N
      RAND=XN/8388607.
      RETURN
C     POSITIVE KK RUNS SECOND STRING OF RANDOM NUMBERS
 61   CONTINUE
         IF (KK-50) M1,71,101
 71   CONTINUE
      NN=NN*2051
         IF (NN) 81,91,91
 81   CONTINUE
      NN=NN+8388607+1
 91   CONTINUE
      XNN=NN
      RAND=XNN/8388607.
      RETURN
C     KK OVER 5  RUNS THIRD STRING OF RANDOM NUMBERS
 101  CONTINUE
      NNN=NNN*2051
         IF (NNN) 111,121,121
      CONTINUE
      NNN=NNN+8388607+1
 121  CONTINUE
      XNNN=NNN
      RAND=XNNN/8388607.
      RETURN
      END

      CONTINUE
```

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

APLICACIONES DE LA COMPUTADORA A LA SIMULACION Y OPTIMIZACION

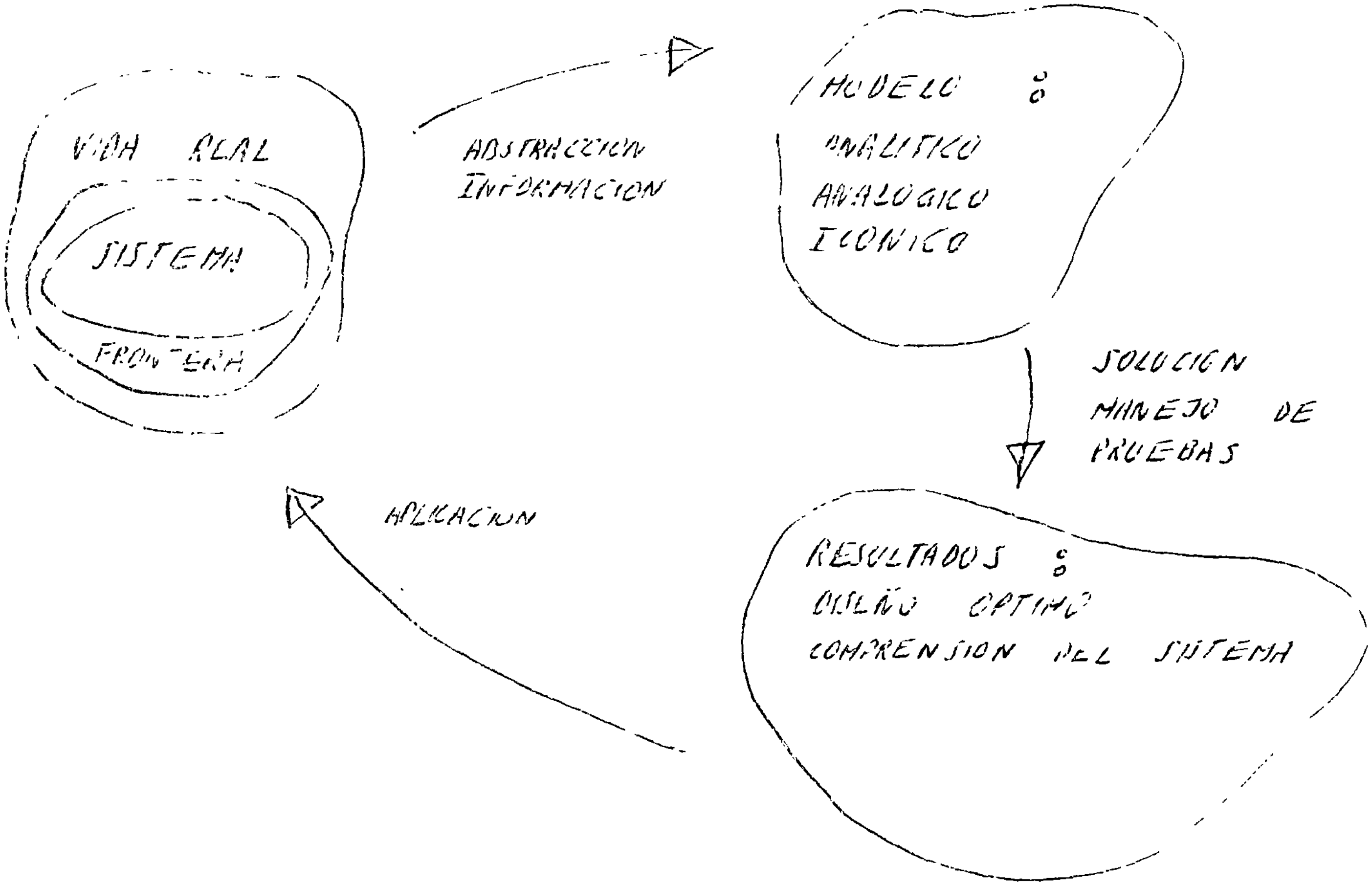
TEMA: MODELADO Y SIMULACION.

M. en C. MAURICIO MIER MUTH.

marzo-abril,1978.

I) CONCEPTO DE SIMULACION

SIMULACION ES EL PROCESO DE CONDUCIR EXPERIMENTOS CON UN MODELO DEL SISTEMA QUE ESTA SIENDO ESTUDIADO O DISEÑADO

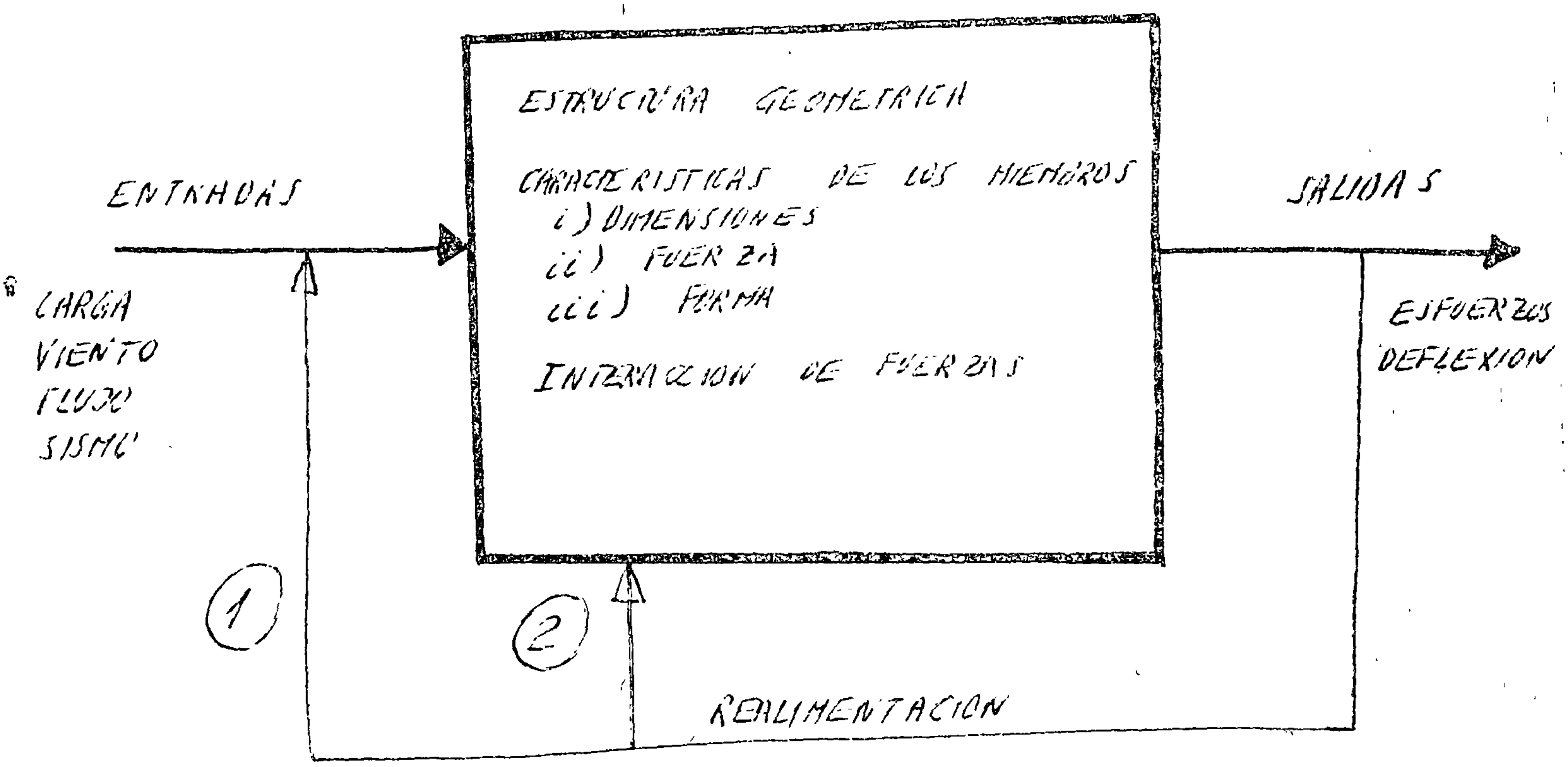


LAS ENTRADAS DEFINEN UN CONJUNTO DE EVENTOS Y CONDICIONES A LAS CUALES PUEDE ESTAR SUJETO EL SISTEMA EN LA VIDA REAL

LAS SALIDAS PREDICEN LAS RESPUESTAS DEL SISTEMA.

EJEMPLO :

MODELO DE UN PUENTE



(1) PARA DEFINIR EL NUEVO CONJUNTO DE VALORES PARA LAS VARIABLES DE LAS ENTRADAS (ANALISIS POR SIMULACION)

(2) PARA CAMBIAR LAS CARACTERISTICAS DE LAS COMPONENTES DEL SISTEMA (SINTESIS POR SIMULACION)

II) MODELOS DE SIMULACION

UN BUEN MODELO DEBERA REPRESENTAR LAS CARACTERISTICAS DEL SISTEMA DE TAL FORMA QUE EL SISTEMA CONSIDERADO PUEDA SER RESUELTO.

LOS MODELOS PARA SIMULACION PUEDEN SER :

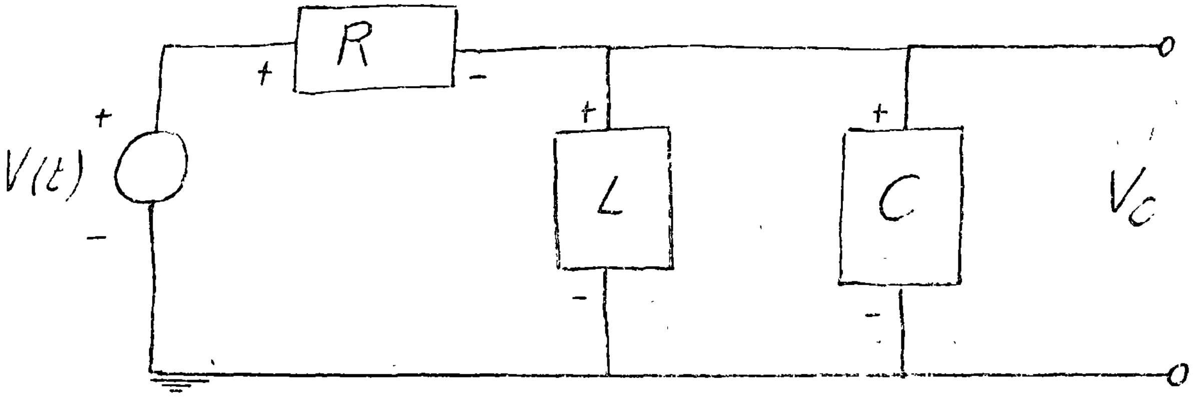
a) ANALITICOS : a') PROBABILISTICOS
DETERMINISTICOS a'') DINAMICOS
ESTATICOS

b) ANALOGICOS

c) ICONICOS

a) ANALITICOS : SON UTILIZADOS EN PROBLEMAS EN LOS CUALES LAS CARACTERISTICAS DE LAS COMPONENTES Y LA ESTRUCTURA DEL SISTEMA PUEDEN SER DESCRITAS MATEMATICAMENTE. PUEDE ESTAR COMPUESTO POR ECUACIONES, DATOS NUMERICOS, CONDICIONES DE FRONTERA, ETC... ES NECESARIO ENTENDER LAS PROPIEDADES FUNDAMENTALES DE LOS COMPONENTES DEL SISTEMA Y SU INTERACCION CON EL ADVENIMIENTO DE LAS COMPUTADORAS ELECTRONICAS DE GRAN VELOCIDAD Y MEMORIA LOS MODELOS ANALITICOS REPRESENTAN UNA GRAN AYUDA EN TODAS LAS RAMAS DE LA INGENIERIA.

UNA HERRAMIENTA MUY UTIL SON LOS MODELOS EN VARIABLES DE ESTADO. ESTADO DE UN SISTEMA ES LA MINIMA CANTIDAD DE INFORMACION QUE RESUME EL EFECTO DE ANTERIORES ENTRADAS.



$V_R = R I_R$ — ①

$V_L = L \frac{dI_L}{dt}$ — ②

$I_C = C \frac{d}{dt} V_C$ — ③

LEY DE VOLTAJES DE KIRCHHOF

$V(t) = V_R + V_L$ — ④

$V_L = V_C$ — ⑤

LEY DE CORRIENTES DE KIRCHHOF

$I_R = I_L + I_C$ — ⑥

MODELO EN VARIABLES DE ESTADO

(LINEAL E INVARIANTE CON EL TIEMPO)

$\underline{X} \triangleq \begin{bmatrix} V_C \\ I_L \end{bmatrix}$ ESTADO DEL SISTEMA

LLEVANDO (3) A (6)

$$I_R = I_L + C\frac{d}{dt}V_C \quad \text{---} \quad (7)$$

LLEVANDO (5), (1) Y (7) A (4)

$$V(t) = RI_R + V_C =$$

$$V(t) = R\left[I_L + C\frac{dV_C}{dt}\right] + V_C$$

$$\Longrightarrow \frac{dV_C}{dt} = \frac{1}{RC}\left[-V_C - RI_L + V(t)\right] \quad \text{———} \quad (I)$$

LLEVANDO (2) A (5)

$$L\frac{dI_L}{dt} = V_C$$

$$\Longrightarrow \frac{dI_L}{dt} = \frac{1}{L}\left[V_C\right] \quad \text{———} \quad (II)$$

LO CUAL IMPLICA QUE :

$$\dot{\underline{X}} = \begin{bmatrix} -\frac{1}{RC} & -\frac{1}{C} \\ \frac{1}{L} & 0 \end{bmatrix} \underline{X} + \begin{bmatrix} \frac{1}{RC} \\ 0 \end{bmatrix} V(t)$$

$$V_C = \begin{bmatrix} 1 & 0 \end{bmatrix} \underline{X} + \begin{bmatrix} 0 \end{bmatrix} V(t)$$

(MODELO MATEMATICO)
DETERMINISTICO

LA SOLUCION DE UN MODELO MATEMATICO DE VARIABLES DE ESTADO ESTA DADO POR LA FORMULA DE VARIACION DE PARÁMETROS

$$\overset{\circ}{X} = AX + BU$$

$$Y = CX + DU$$

SISTEMA LINEAL E INVARIANTE EN EL TIEMPO

$$\Longrightarrow \quad Y(t) = C e^{A(t-t_0)} X(t_0) + \int_{t_0}^{t} \left[C e^{A(t-\sigma)} B + D\delta(t-\sigma) \right] U(\sigma)\, d\sigma$$

$$\Longrightarrow \quad Y(t) = Y_H(t) + Y_P(t)$$

DONDE $\delta(t-\sigma)$ = ES UN IMPULSO DE PESO 1 EN $t = \sigma$

e^{At} = MATRIZ DE TRANSICION

LA EXPRESION $Ce^{A(t-\sigma)}B + D\delta(t-\sigma)$ SE CONOCE CON EL NOMBRE DE PATRON DE PESO DEL SISTEMA. SU TRANSFORMADA DE LAPLACE SE CONOCE CON EL NOMBRE DE FUNCION DE TRANSFERENCIA $H(S)$ DEL SISTEMA.

$$Y(S) = H(S) \cdot U(S) \qquad \text{SI } X(t_0) = 0$$

b) ANALOGICOS : SON MODELOS EN LOS QUE EL SISTEMA REAL ES MODELADO ATRAVES DE UN MEDIO FISICO COMPLETAMENTE DIFERENTE .

OTRA FORMA DE RESOLVER EL CONJUNTO DE RELACIONES MATEMATICAS DE UN MODELO ANALITICO ES POR SIMULACION . EN LA COMPUTADORA ANALOGICA ENCONTRAREMOS SOLUCIONES DE ECUACIONES DIFERENCIALES CONSTRUYENDO CIRCUITOS ELECTRONICOS QUE SE CARACTERICEN POR LAS MISMAS ECUACIONES. (SIMULAREMOS DISTINTOS PROCESOS FISICOS CON CIRCUITOS ELECTRONICOS ANALOGOS) .

MODELOS ANALÓGICOS

COMPUTADORA ANALÓGICA

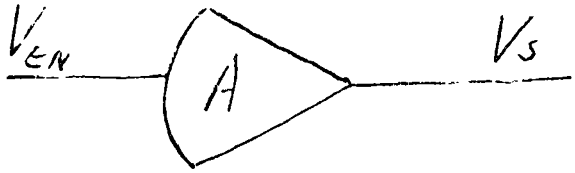
a) SUMADORES

b) INVERSORES

c) INTEGRADORES

d) POTENCIOMETROS

AMPLIFICADORES OPERACIONALES



$$Z_{EEN} \Rightarrow \infty$$

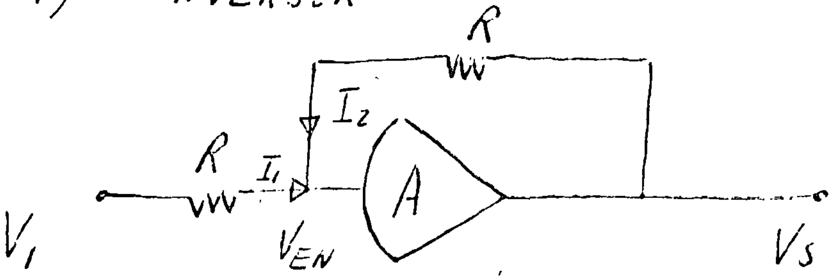
$$A \Rightarrow \infty$$

$$V_S = -A V_{EN}$$

$$V_{S_{MAX}} = \pm 10 \text{ VOLT}$$

$$\Rightarrow V_{EN} \simeq 0$$

a) INVERSOR



$$\frac{V_1 - V_{EN}}{R} = I_1$$

$$\frac{V_S - V_{EN}}{R} = I_2$$

SI $\quad Z_{EEN} = \infty$

$$I_1 = -I_2$$

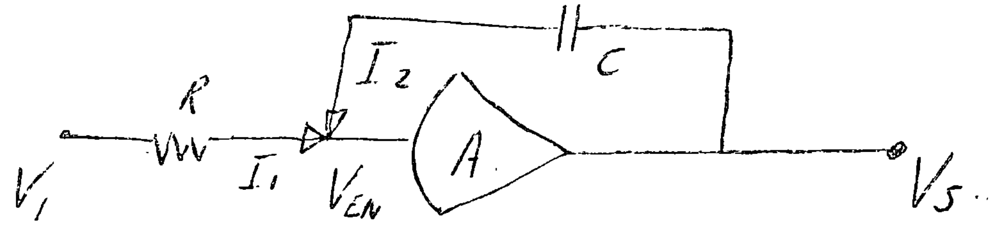
$$\Rightarrow \quad \frac{V_1 - V_{EN}}{R} = \frac{V_{EN} - V_S}{R}$$

$$\frac{V_1}{R} = -\frac{V_S}{R}$$

$$V_S = -V_1$$

(INVERSOR)

b) INTEGRADOR



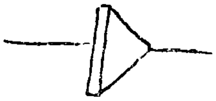
$$\frac{V_1 - V_{EN}}{R} = I_1$$

$$C\frac{d(V_S - V_{EN})}{dt} = I_2$$

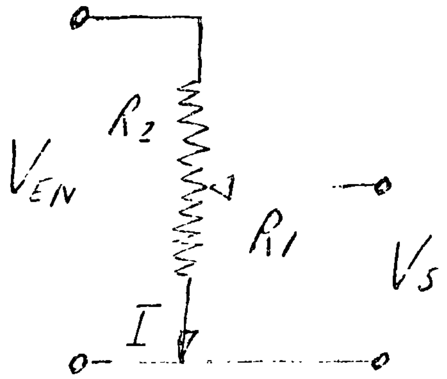
$$\Rightarrow \quad \frac{V_1 - V_{EN}}{R} = -C\frac{d}{dt}(V_S - V_{EN})$$

$$-\int_0^t \frac{V_1}{RC}\,dt = V_S$$

(INTEGRADOR).



d) Potenciometros



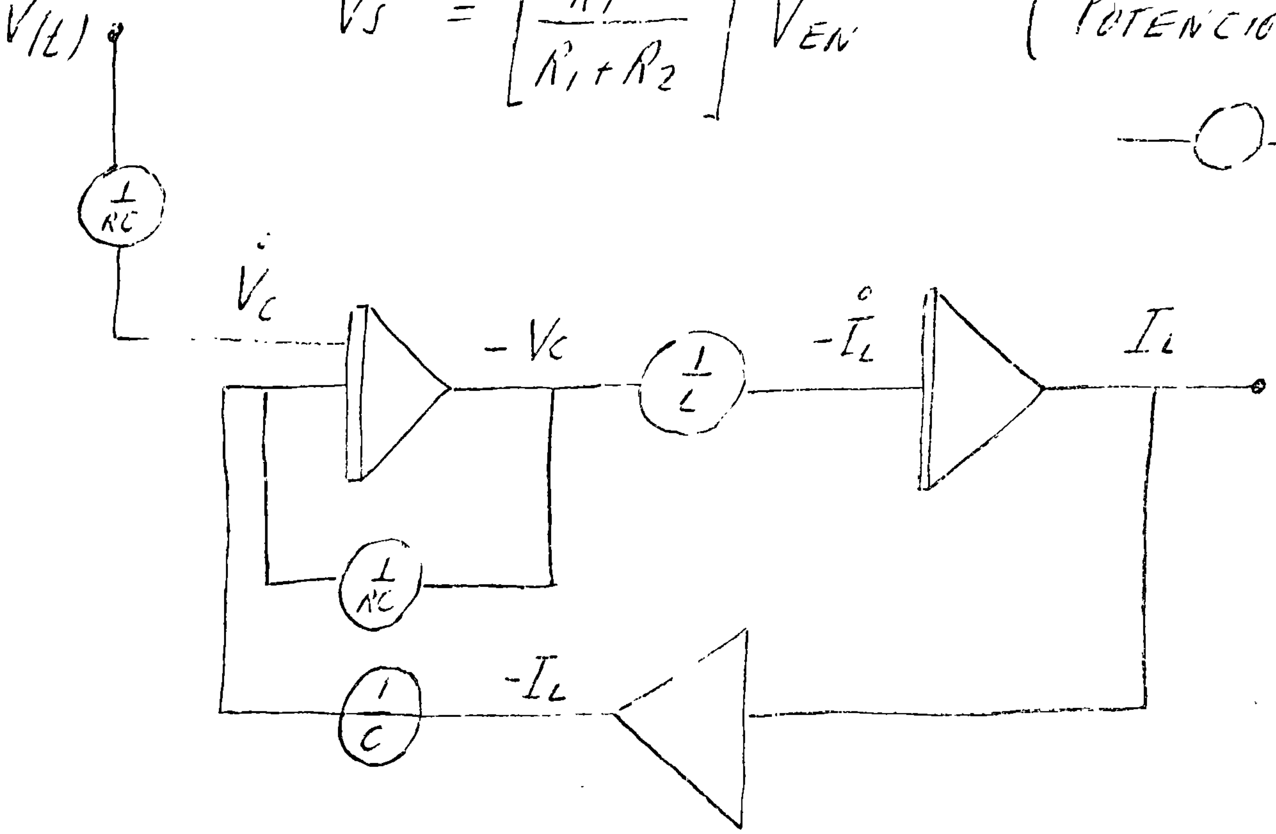
$$V_{EN} = (R_1 + R_2)\, I$$

$$V_S = R_1 I$$

$$\Longrightarrow \quad \frac{V_S}{V_{EN}} = \frac{R_1}{R_1 + R_2}$$

$$V_S = \left[\frac{R_1}{R_1 + R_2}\right] V_{EN}$$

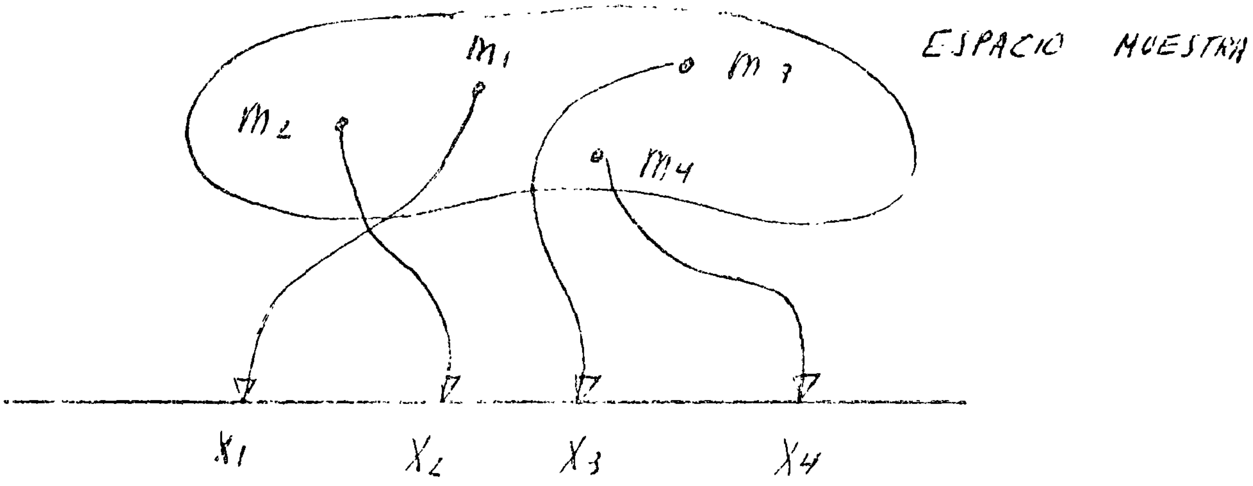
(Potenciometro)



MODELOS ICÓNICOS : SON RÉPLICAS FÍSICAS DEL SISTEMA REAL AUNA ESCALA REDUCIDA.
(EN DISEÑO DE AERONAVES SE UTILIZAN TÚNELES DE VIENTO PARA SIMULAR NAVES EN VUELO.)

III) METODO DE MONTECARLO :

ESTE NOMBRE SE HA DADO EN FORMA GENERICA A LAS TECNICAS DE SIMULACION QUE UTILIZAN VARIABLES ALEATORIAS. UNA VARIABLE ALEATORIA ES UNA FUNCION $X(m_j)$ QUE TRANSFORMA LOS PUNTOS MUESTRA m_j DEL ESPACIO MUESTRA DE UN EXPERIMENTO EN UN NUMERO.



DE ESTA MANERA PODEMOS HACER REFERENCIA A UN VALOR DE UNA VARIABLE EN VEZ DE MENCIONAR LA SALIDA DEL EXPERIMENTO.

1) FUNCION DE DENSIDAD DE PROBABILIDAD

X_i SON LOS VALORES DE LA VARIABLE ALEATORIA X

A CADA X_i LE CORRESPONDE UN EVENTO m_i
A CADA m_i LE CORRESPONDE UNA PROBABILIDAD P_i

$\Longrightarrow$ A CADA X_i LE CORRESPONDE UNA P_i

$$P_i = f(X_i)$$

DONDE $F(X)$ FUNCION DE DENSIDAD DE P.

DADO QUE $X_1, X_2, \dots X_n$ SON TODOS LOS POSIBLES VALORES DE LA VARIABLE ALEATORIA X

$$\sum_{i=1}^{n} F(X_i) = \sum_{i=1}^{n} P_i = 1$$

2) FUNCION DE DISTRIBUCION DE PROBABILIDAD

$$P(X_i \leq a) \triangleq F(a) \qquad \text{(FUNCION CRECIENTE)}$$

$$P(X_i \leq a) = \sum_{X_i=0} f(X_i)$$

$$P(b \leq X_i \leq a) = F(a) - F(b)$$

Si $a = \infty$ $\qquad P(X_i \leq \infty) = 1$

Si $a = -\infty$ $\qquad P(X_i \leq -\infty) = 0$

MOMENTOS

i) ALREDEDOR DEL ORIGEN $E\left[X^{K}\right] = \sum_{j=1}^{n} X_j \cdot f(x_j) = m_K$

ii) ALREDEDOR DE LA MEDIA $E\left[\left(X - E(X)\right)^{K}\right] = \sum_{j=1}^{n} \left(X_j - E(X)\right)^{K} F(X_j) = \mu$

MEDIA Ó PROMEDIO

$$E(X) = \sum_{j=1}^{n} X_j \, f(x_j) = m_1$$

VARIANCIA

$$E\left[\left(X - E(X)\right)^{2}\right] = \sum_{j=1}^{n} \left(X_j - E(X)\right)^{2} F(x_j) = \sigma^2 = \mu_2$$

DESVIACION ESTANDAR

$$\sigma = \sqrt{\text{VARIANCIA}}$$

a) EXPONENCIAL ———— $f(x) = \frac{1}{a} e^{-\frac{x}{a}}$

$E(X) = 1/a$

$\sigma^2_x = 1/a^2$

b) NORMAL ———— $f(x) = \frac{1}{\sqrt{2\pi b}} e^{-\frac{(x-a)^2}{2b}}$

$E(X) = a$

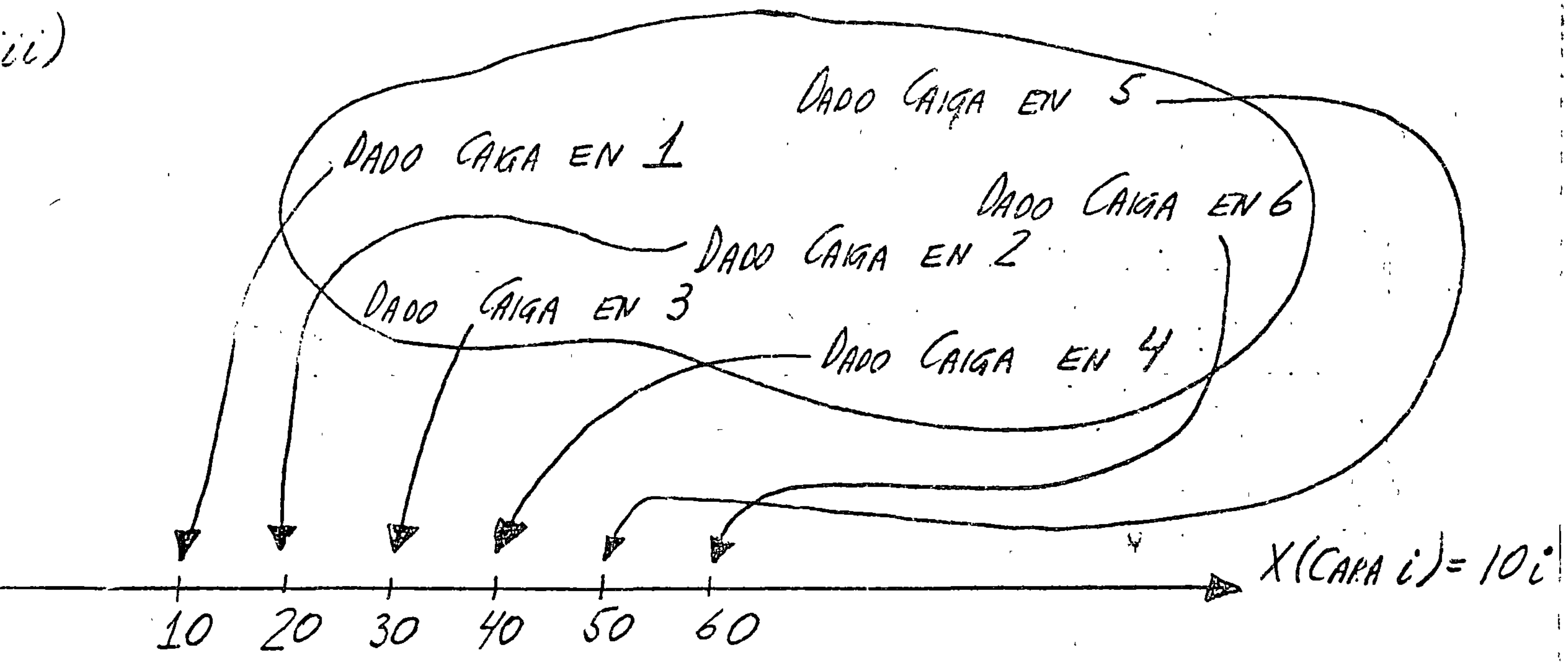
$\sigma^2_x = b$

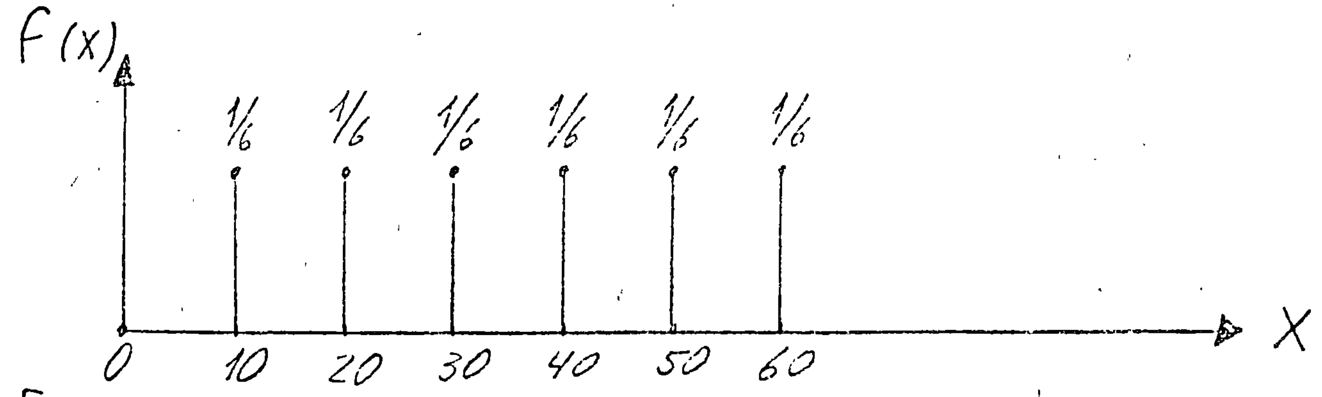
EJEMPLO : RODAR UN DADO

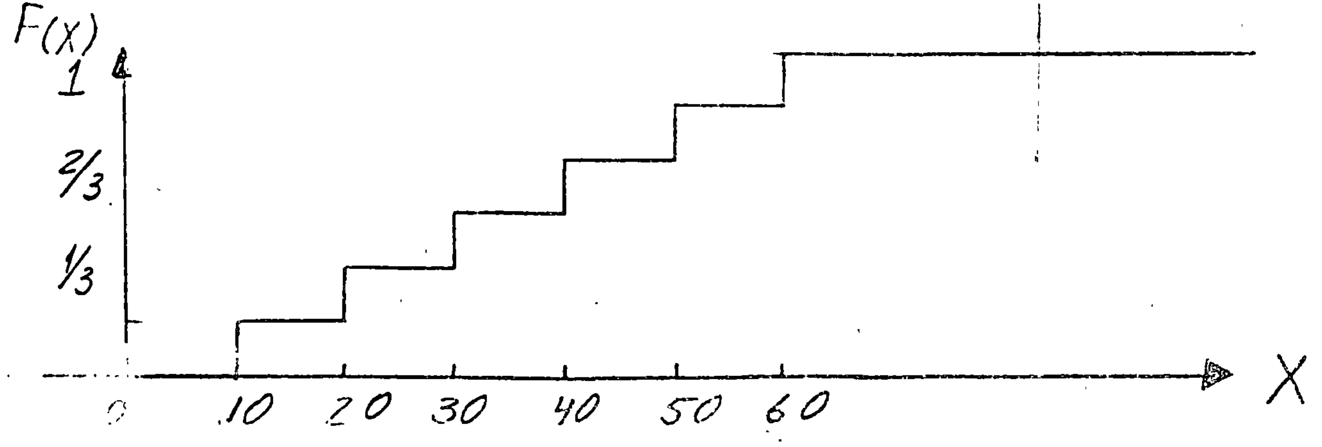
i) NUMERO DE POSIBLES SALIDAS DEL EXPERIMENTO =

ii) PARA DADOS NO CARGADOS, LA PROBABILIDAD DE CUALQUIERA DE ELLAS = 1/6 $P(\text{CARA } i) = 1/6$

iii)







$$E(X) = (1+2+3+4+5+6)\frac{1}{6} = \frac{21}{6} = \frac{7}{2}$$

$$E[(X-7/2)^2] = [(1-7/2)^2 + (2-7/2)^2 + (3-7/2)^2$$

$$+ (4-7/2)^2 + (5-7/2)^2 + (6-7/2)^2]\,1/6$$

$$= [(-5/2)^2 + (-3/2)^2 + (-1/2)^2$$

$$+ (1/2)^2 + (3/2)^2 + (5/2)^2]\,1/6$$

$$= \left(\frac{25}{4} + \frac{9}{4} + \frac{1}{4} + \frac{1}{4} + \frac{9}{4} + \frac{25}{4}\right) 1/6$$

$$= \left(\frac{70}{4}\right)\frac{1}{6} = \frac{35}{12}$$

$$\Longrightarrow \sigma^2 = 35/12$$

$$\sigma = \sqrt{35/12}$$

$$F(x^2 - 2x\,7/2 + 49/4) = E(X^2) - 7E(x) + 49/4$$

$$\sigma^2 = (1+4+9+16+25+36)\frac{1}{6}$$

$$- 7 \cdot 7/2 + 49/4$$

$$\sigma^2 = \frac{91}{6} - \frac{49}{4} = \frac{35}{12}$$

IV) SIMULACION DE UNA PLANTA DE CONCRETO

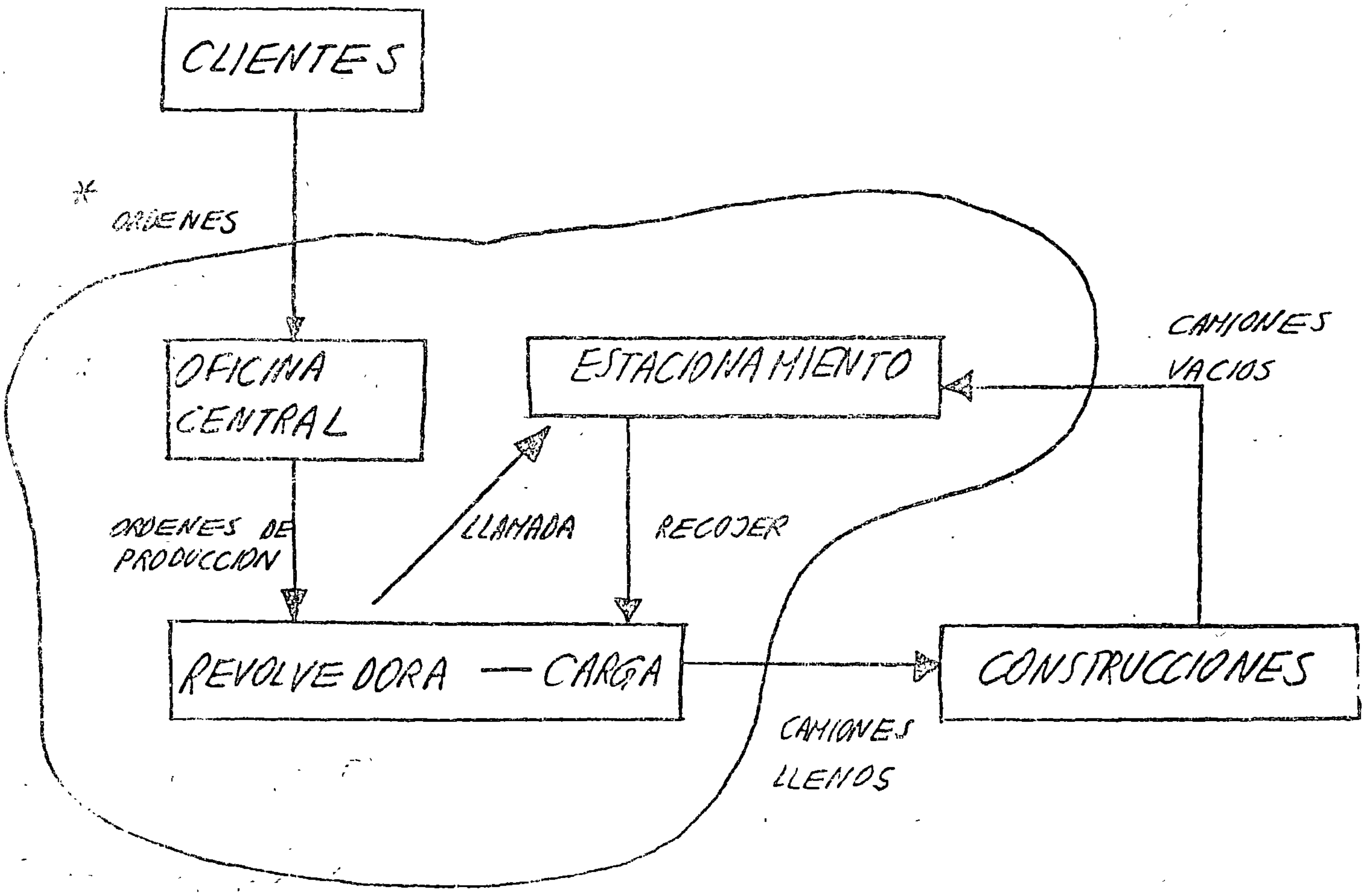
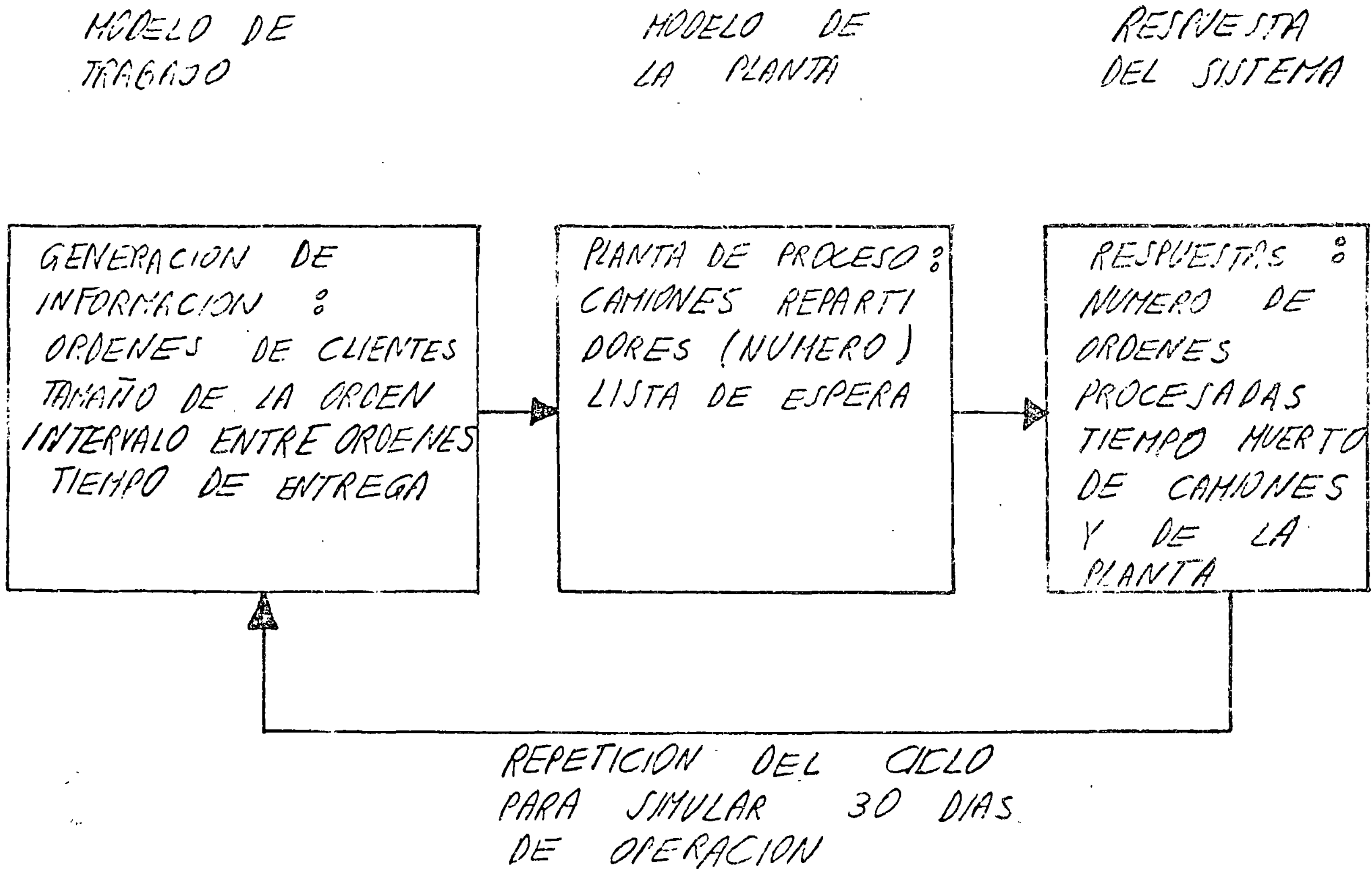


DIAGRAMA DEL PROCESO DE PRODUCCION

* HORA DE INICIACION DE LABORES 8 AM
<u></u>NO PROCESAMIENTO DE ORDENES DESPUES DE 3.30 PM



SIMULACION DE LA OPERACION DE UNA PLANTA DE CONCRETO

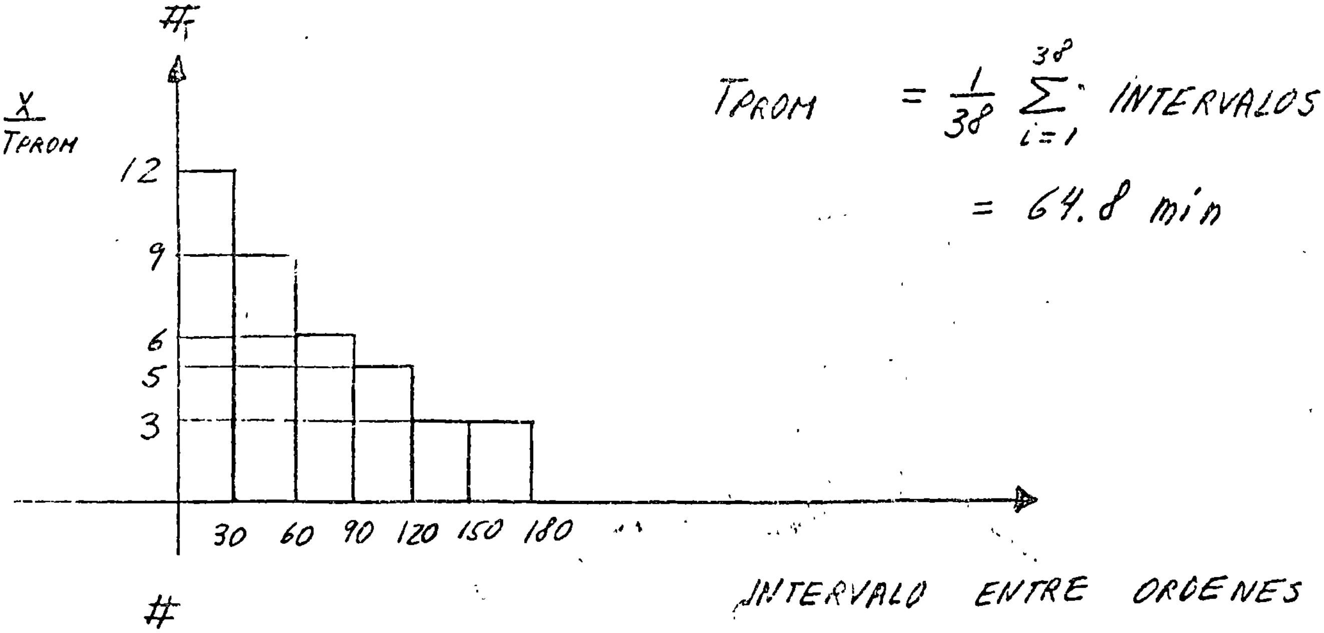
NOTA

LA SALIDA DEL PROCESO DE SIMULACION PROVEE UNA MEDIDA DE LA RESPUESTA DEL SISTEMA EN FUNCION DEL NUMERO DE CAMIONES REPARTIDORES DISPONIBLE.

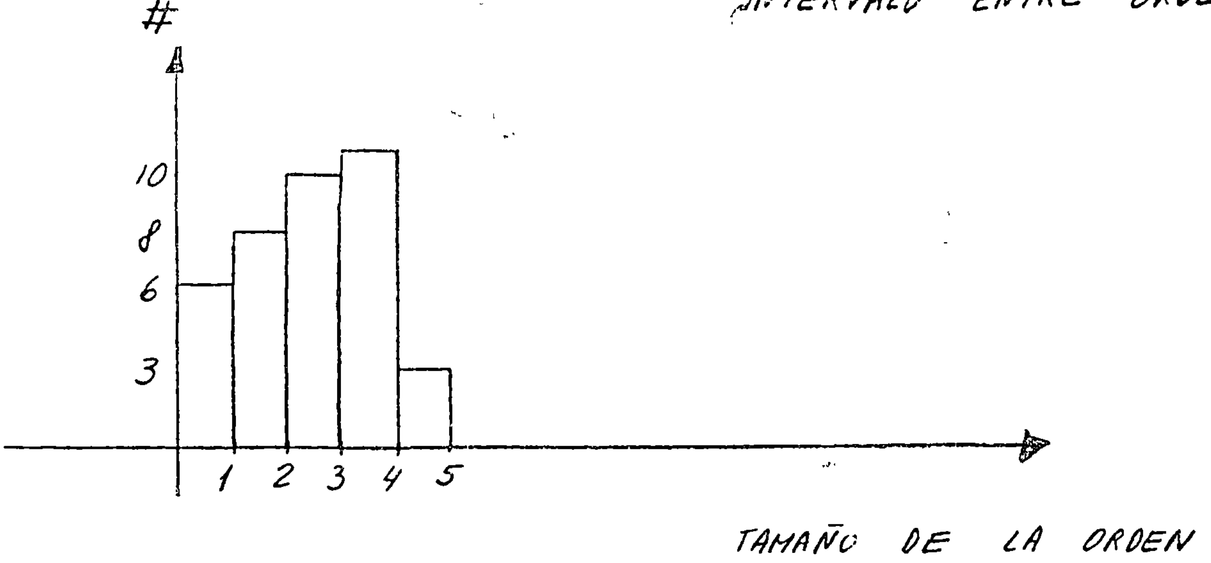
REGISTROS DEL SISTEMA REAL

DIA	ORDEN NUMERO	HORA	INTERVALO ENTRE ORDENES	NUMERO DE CAMIONES	TIEMPO DE ENTREGA CAMION					PROMEDIO	DESVIACION DE LA MEDIA
					1	2	3	4	5		
LUN	1	8.15	15	2	64	69				67	-3,2
	2	8.32	17	1	75					75	0
	3	9.03	31	3	73	77	70			73	0,4,-3
	4	9.47	44	3	123	116	123			121	2,-5,2
	5	10.00	13	2	105	104				105	0,-1
	6	11.16	76	4	27	34	34	30		31	-4,3,3,-1
	7	1.03	107	1	48					48	0
	8	2.06	63	2	55	59				57	-2,2
	9	2.18	12	1	83					83	0
	10	2.49	31	3	50	46	54			49	1,-3,5
	11	4.06	77	2	61	65				66	-5,-1
MAR	12	9.40	100	3	73	78	77			76	-3,2,1
	13	10.12	32	4	66	71	70	69		69	-3,2,1,0
	14	10.45	33	4	41	43	45	48		44	-3,-1,1,4
	15	11.55	70	3	60	65	65			63	-3,2,2
	16	1.24	89	2	91	103				97	-6,6
	17	2.22	58	1	7					7	0
	18	4.27	125	3	99	105	94			99	0,6,-5
MIER	19	8.25	25	4	47	41	40	45		43	4,-2,-3,2
	20	10.35	130	4	68	66	62	66		66	2,0,-4,0
	21	12.03	88	3	88	88	85			87	1,1,-2
	22	3.02	179	5	53	59	52	53	53	54	-1,5,-2,-1,-1
	23	3.50	48	2	59	38				39	0,-1
	24	4.15	25	5	30	39	32	37	32	34	-4,5,-2,3,-2
JUE	25	8.05	5	4	74	71	70	72		72	2,-1,-2,0
	26	10.33	148	3	63	62	63			63	0,-1,0
	27	1.04	151	2	94	88				91	3,-3
	28	1.28	14	4	81	87	88	82		85	-4,2,3,-3
	29	1.40	12	3	38	40	41			40	-2,0,1
	30	2.23	43	1	21					21	0
	31	4.10	107	4	71	69	69	70		70	1,-1,-1,0
VIER	32	8.30	30	5	72	76	75	73	69	73	-1,3,2,0,-4
	33	9.28	38	4	107	94	100	94		99	8,-5,1,-5
	34	12.12	164	4	60	69	66	65		65	-5,4,1,0
	35	12.33	21	3	55	57	54			55	0,2,-1
	36	2.05	92	1	82					82	0
	37	2.15	10	4	87	91	85	82		86	1,5,-1,-4
	38	4.13	118	2	55	59				57	2,-2

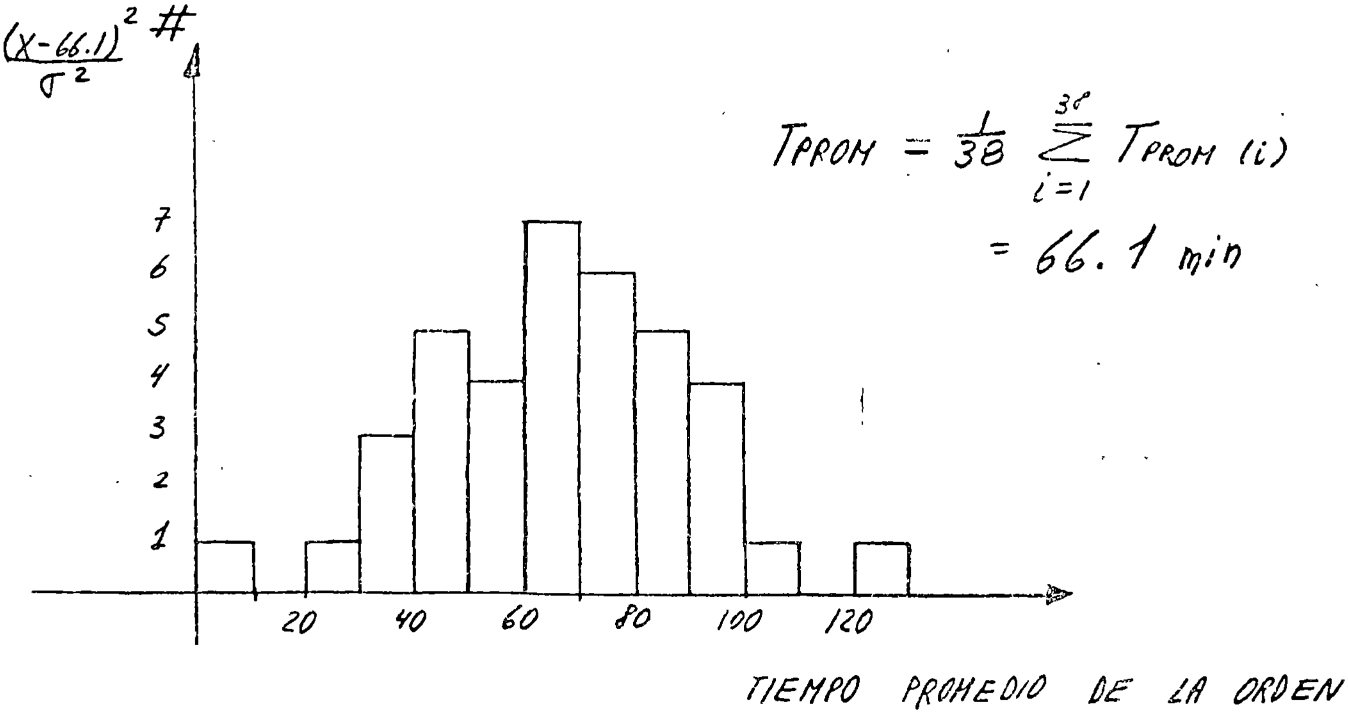
$$f(x) = \frac{1}{T_{PROM}} e^{-\frac{x}{T_{PROM}}}$$



$$T_{PROM} = \frac{1}{38} \sum_{i=1}^{38} \text{INTERVALOS}$$
$$= 64.8 \text{ min}$$



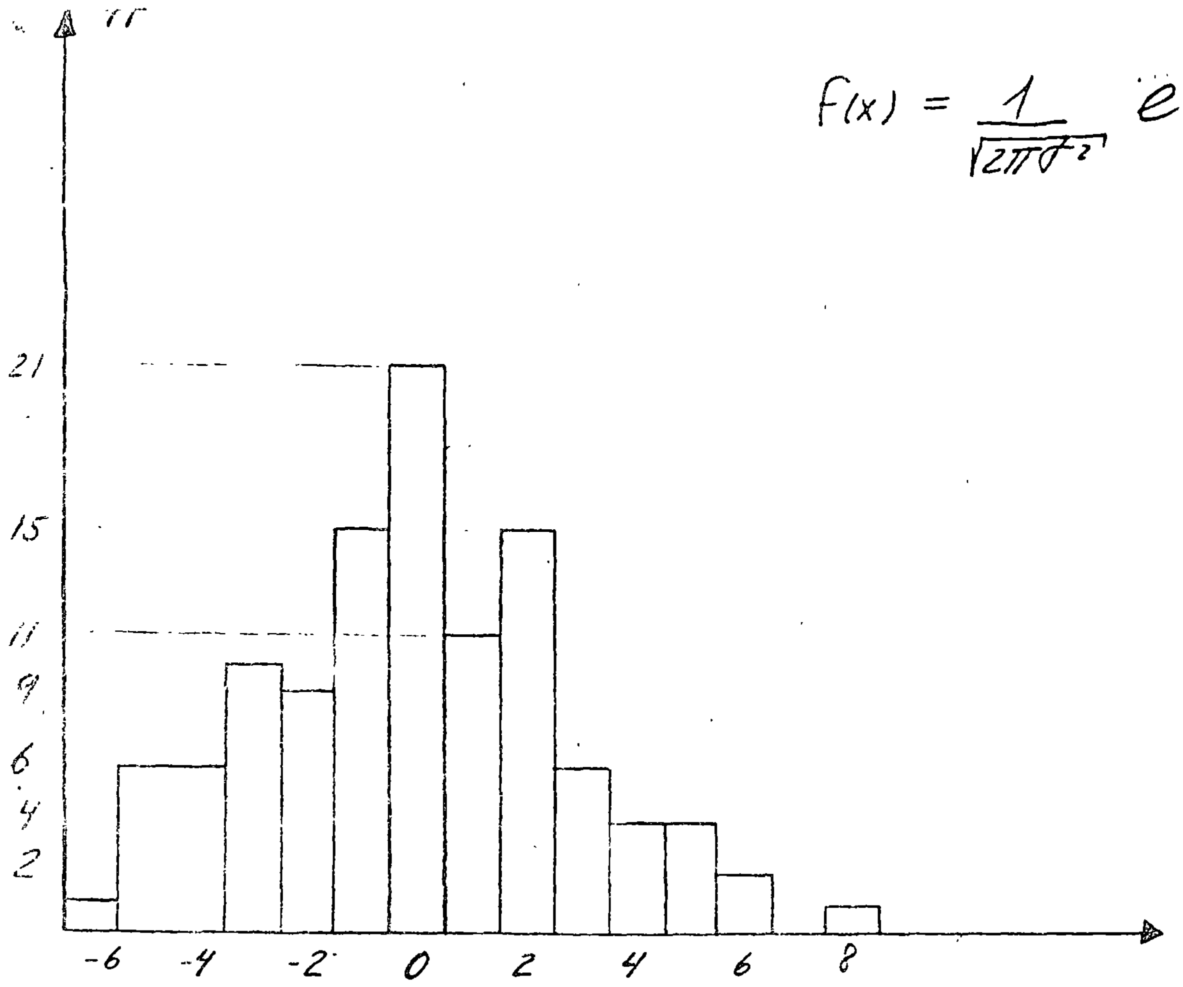
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-66.1)^2}{\sigma^2}}$$



$$T_{PROM} = \frac{1}{38} \sum_{i=1}^{38} T_{PROM}(i)$$
$$= 66.1 \text{ min}$$

NUMERO TOTAL DE ORDENES = 38

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{x^2}{\sigma^2}}$$

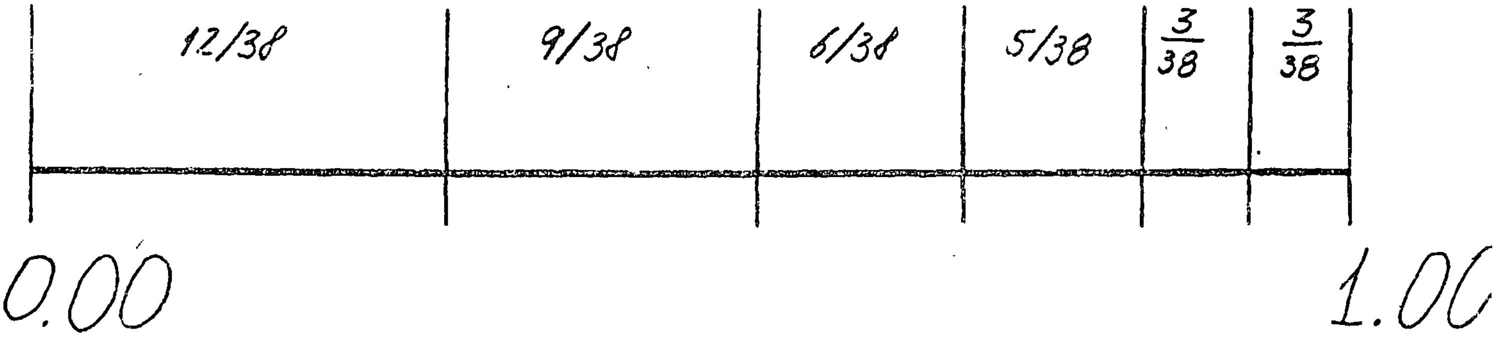


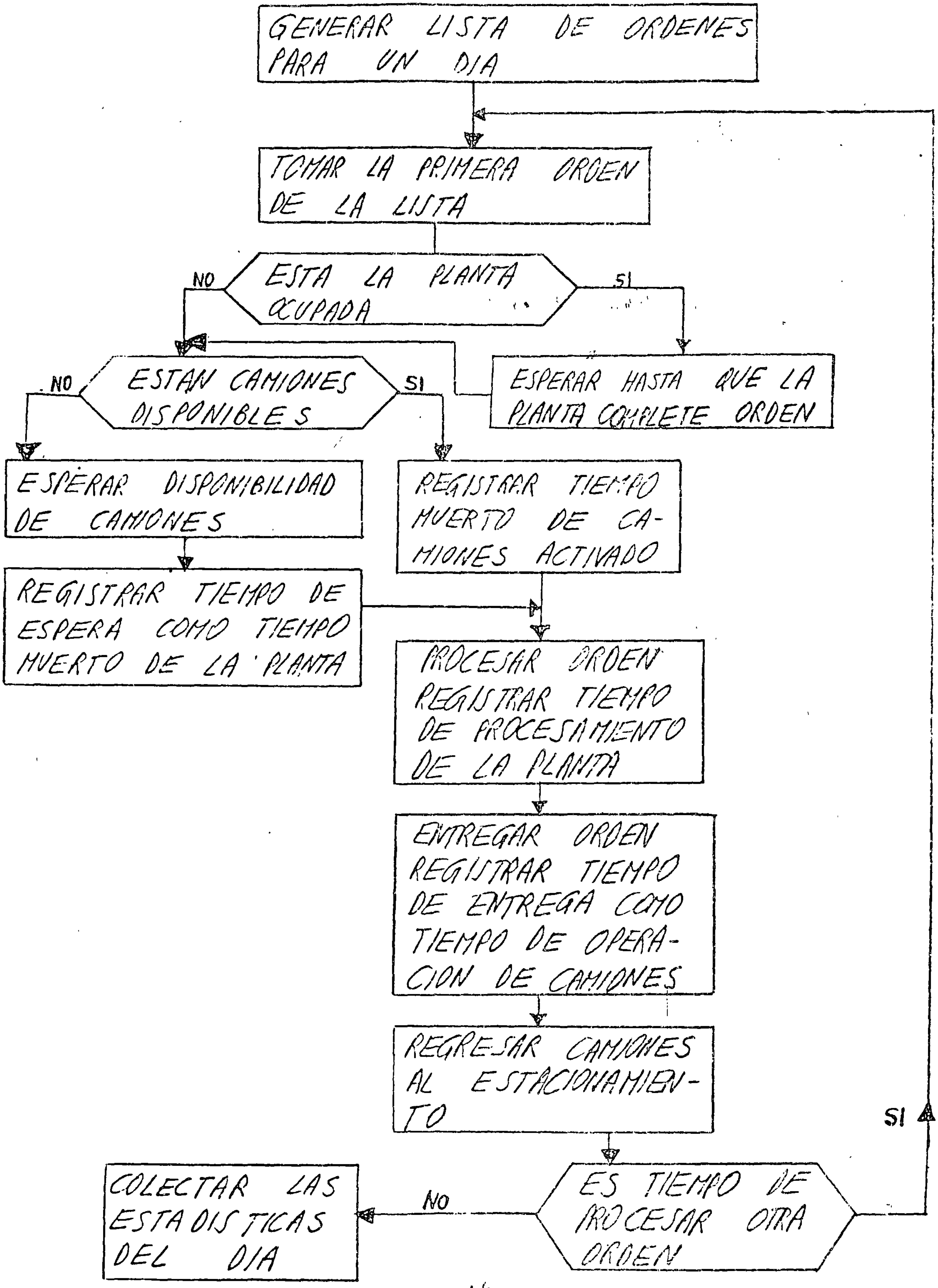
DESVIACION EN EL TIEMPO DE ENTREGA

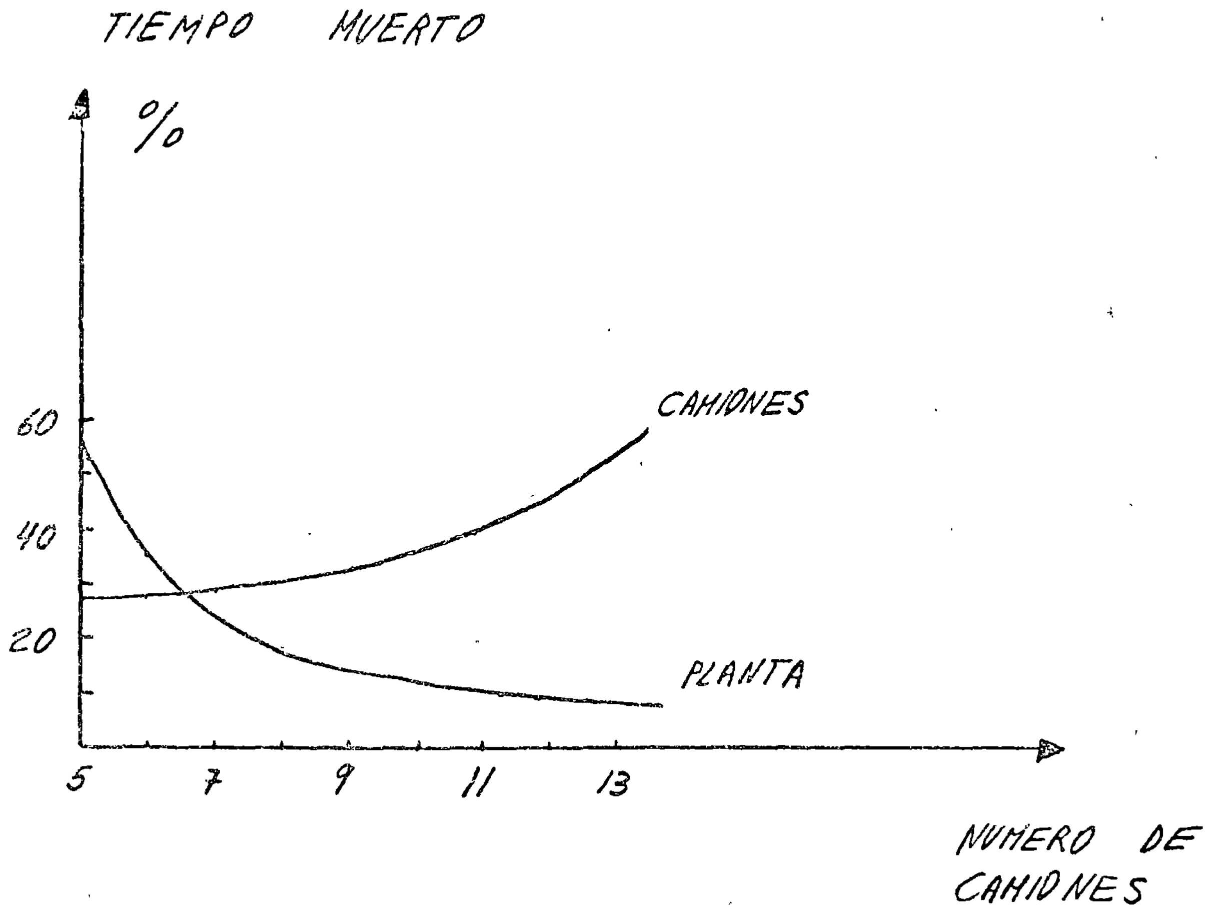
GENERAR LISTA DE ORDENES PARA UN DIA.

A) SE PUEDEN UTILIZAR NUMEROS ALEATORIOS
(TABLAS O GENERACION CON COMPUTADORA)

EJEMPLO : INTERVALO ENTRE ORDENES




GENERAR LISTA DE ORDENES PARA UN DIA
TOMAR LA PRIMERA ORDEN DE LA LISTA
ESTA LA PLANTA OCUPADA
NO
SI
ESPERAR HASTA QUE LA PLANTA COMPLETE ORDEN
ESTAN CAMIONES DISPONIBLES
NO
SI
ESPERAR DISPONIBILIDAD DE CAMIONES
REGISTRAR TIEMPO MUERTO DE CAMIONES ACTIVADO
REGISTRAR TIEMPO DE ESPERA COMO TIEMPO MUERTO DE LA PLANTA
PROCESAR ORDEN REGISTRAR TIEMPO DE PROCESAMIENTO DE LA PLANTA
ENTREGAR ORDEN REGISTRAR TIEMPO DE ENTREGA COMO TIEMPO DE OPERACION DE CAMIONES
REGRESAR CAMIONES AL ESTACIONAMIENTO
ES TIEMPO DE PROCESAR OTRA ORDEN
SI
NO
COLECTAR LAS ESTADISTICAS DEL DIA



RESULTADOS DE LA SIMULACION

V) SIMULACION DE UNA LINEA DE ESPERA

1) TIEMPO DE LLEGADA ENTRE AUTOMOVILES :

$$f(X) = \frac{1}{T_{PROM}} e^{-\frac{X}{T_{PROM}}} \qquad \text{(EXPONENCIAL)}$$

SE SABE QUE $T_{PROM} = 3 \text{ min}$

$$\Longrightarrow \quad f(X) = 0.33\, e^{-0.33X}$$

$$P(\alpha < X < \beta) = \int_{\alpha}^{\beta} f(x)\,dX$$

2) TIEMPO DE SERVICIO EN CADA BOMBA

$$f(X) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(X-\mu)^2}{\sigma^2}} \qquad \text{(NORMAL)}$$

SE SABE QUE EL TIEMPO PROMEDIO DE SERVICIO $\mu = 5 \text{ min}$ Y SU DESVIACION ESTÁNDAR ES $\sigma = 2 \text{ min}$.

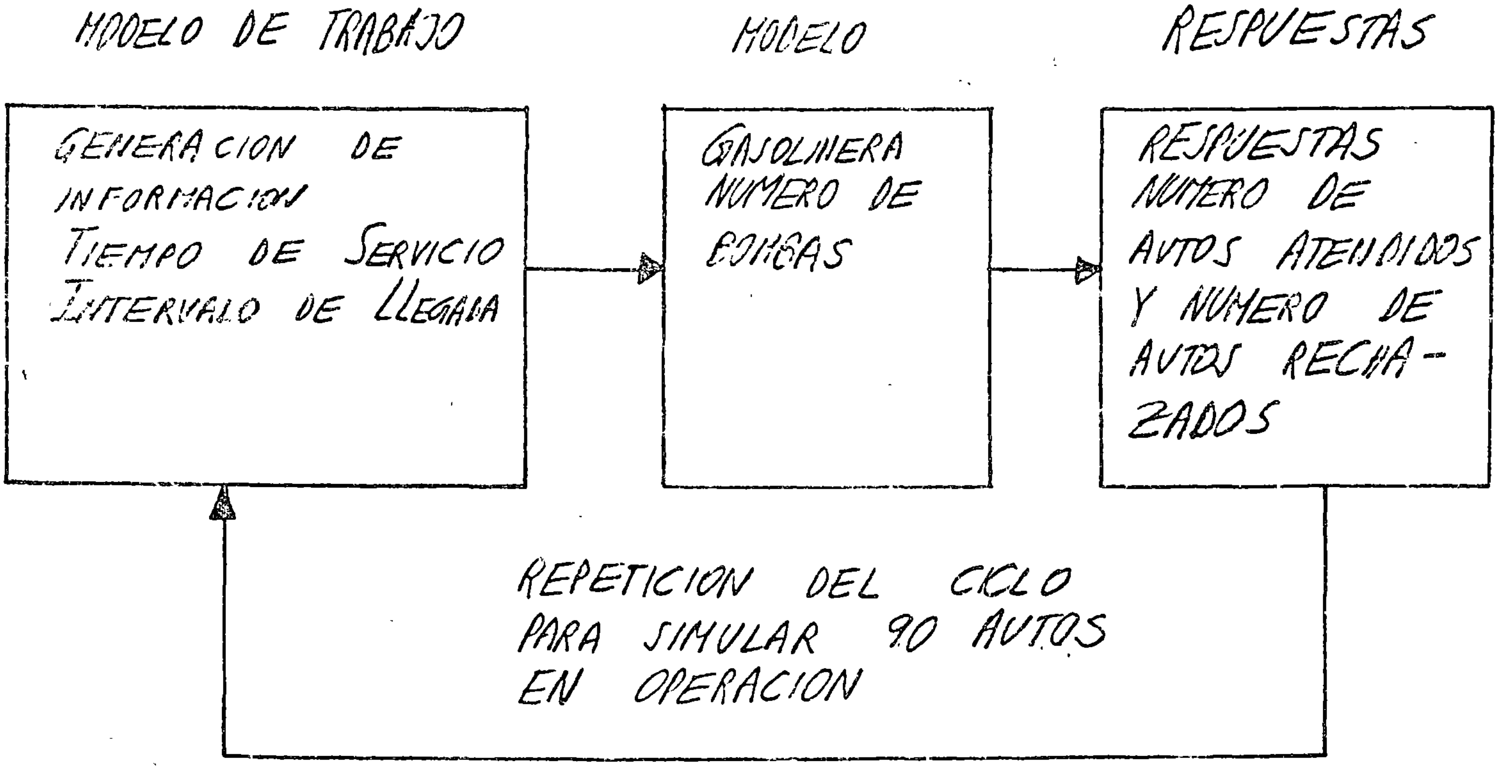
$$P(\alpha < X < \beta) = \int_{\alpha}^{\beta} f(X)\,dX$$

PREMISAS :

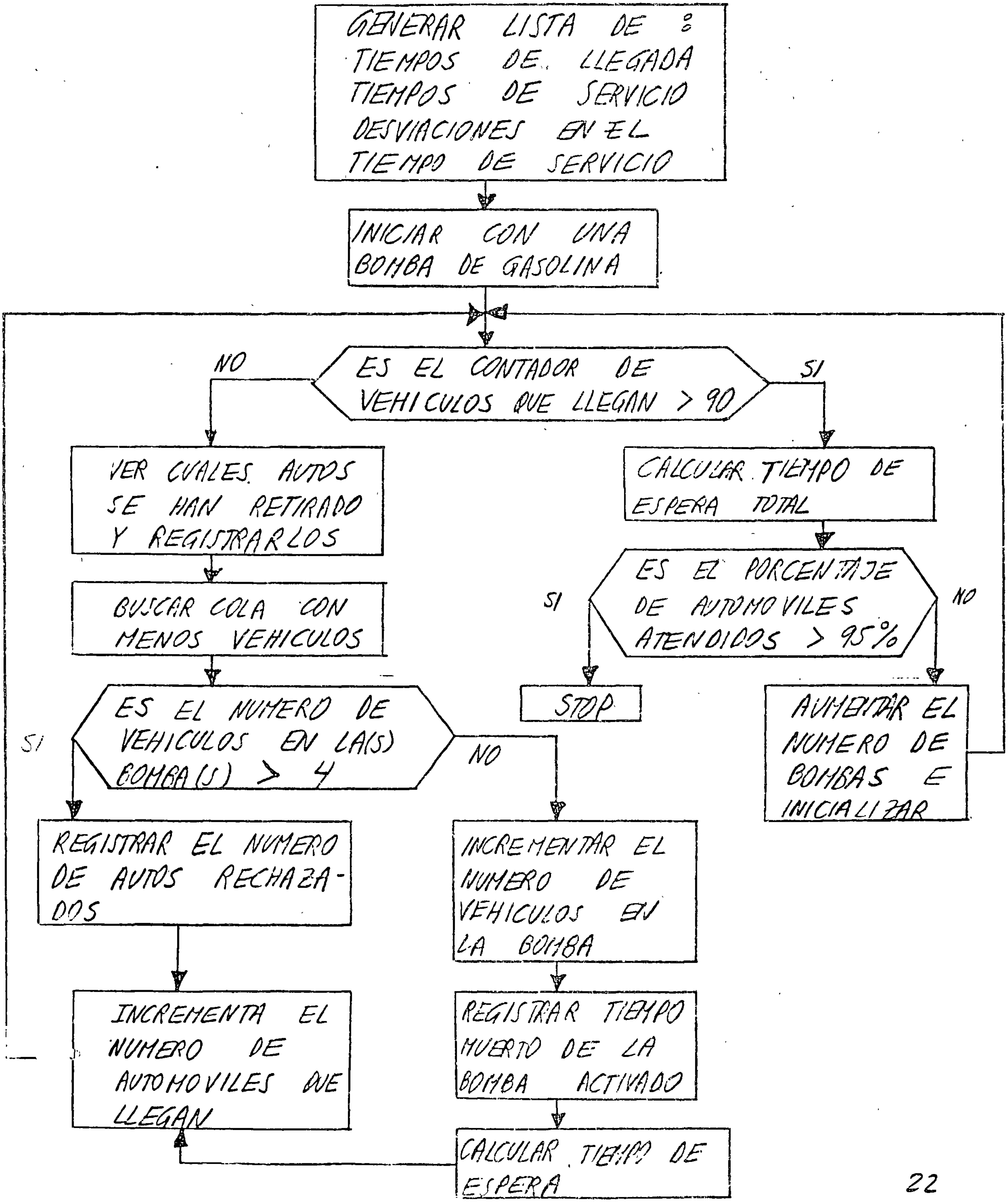
a) EL AUTOMOVILISTA SABE CUANTA GENTE HAY ESPERANDO
b) LOS AUTOMOVILISTAS SE FORMAN EN LAS COLAS MAS CORTAS
c) NO CAMBIAN DE COLA

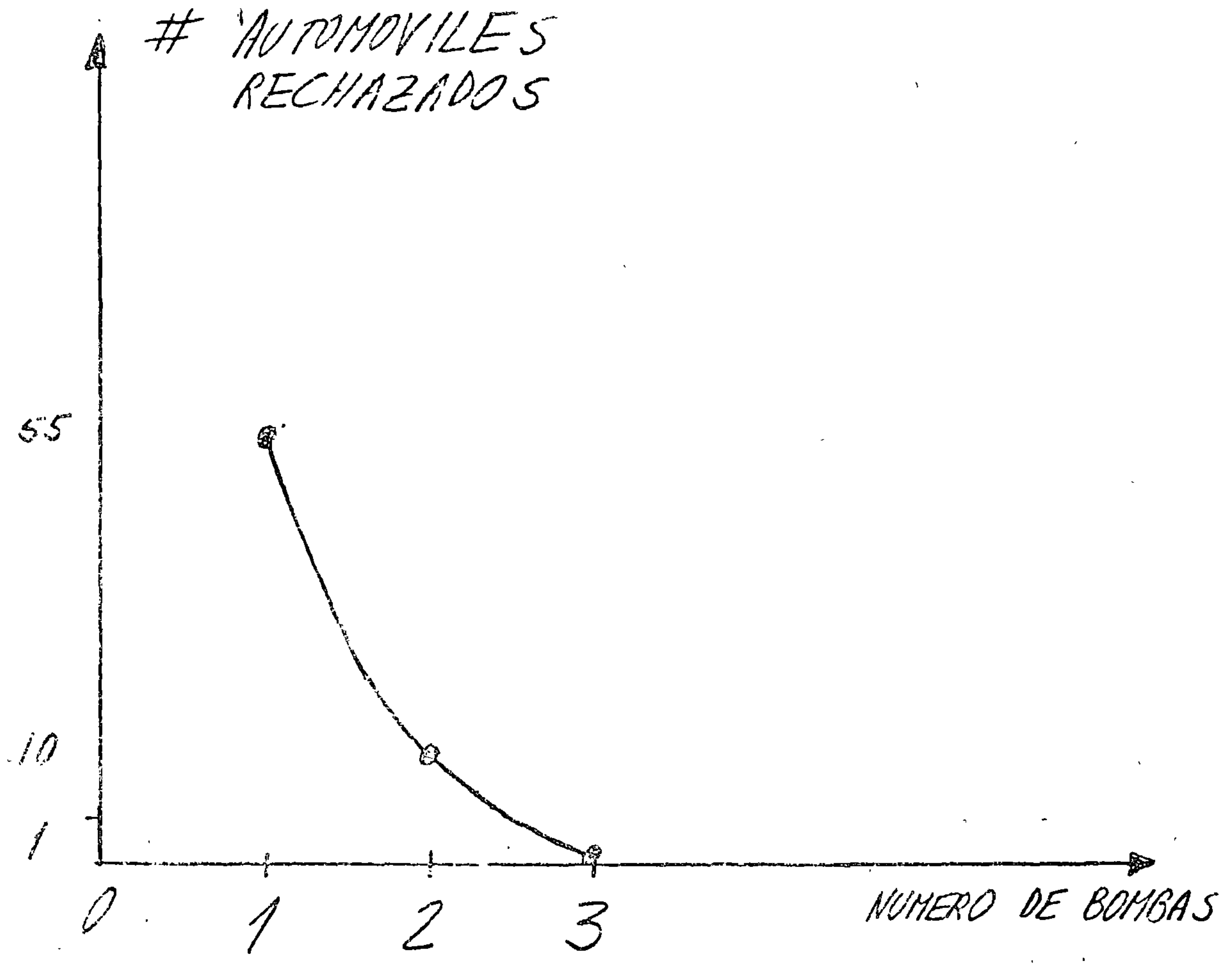
d) SI TODAS LAS BOMBAS TIENEN MÁS DE 4 CLIENTES EL NUEVO AUTOMOVILISTA " SE SIGUE DE FRENTE
e) SE DESEA ATENDER 95% Ó MAS CLIENTES

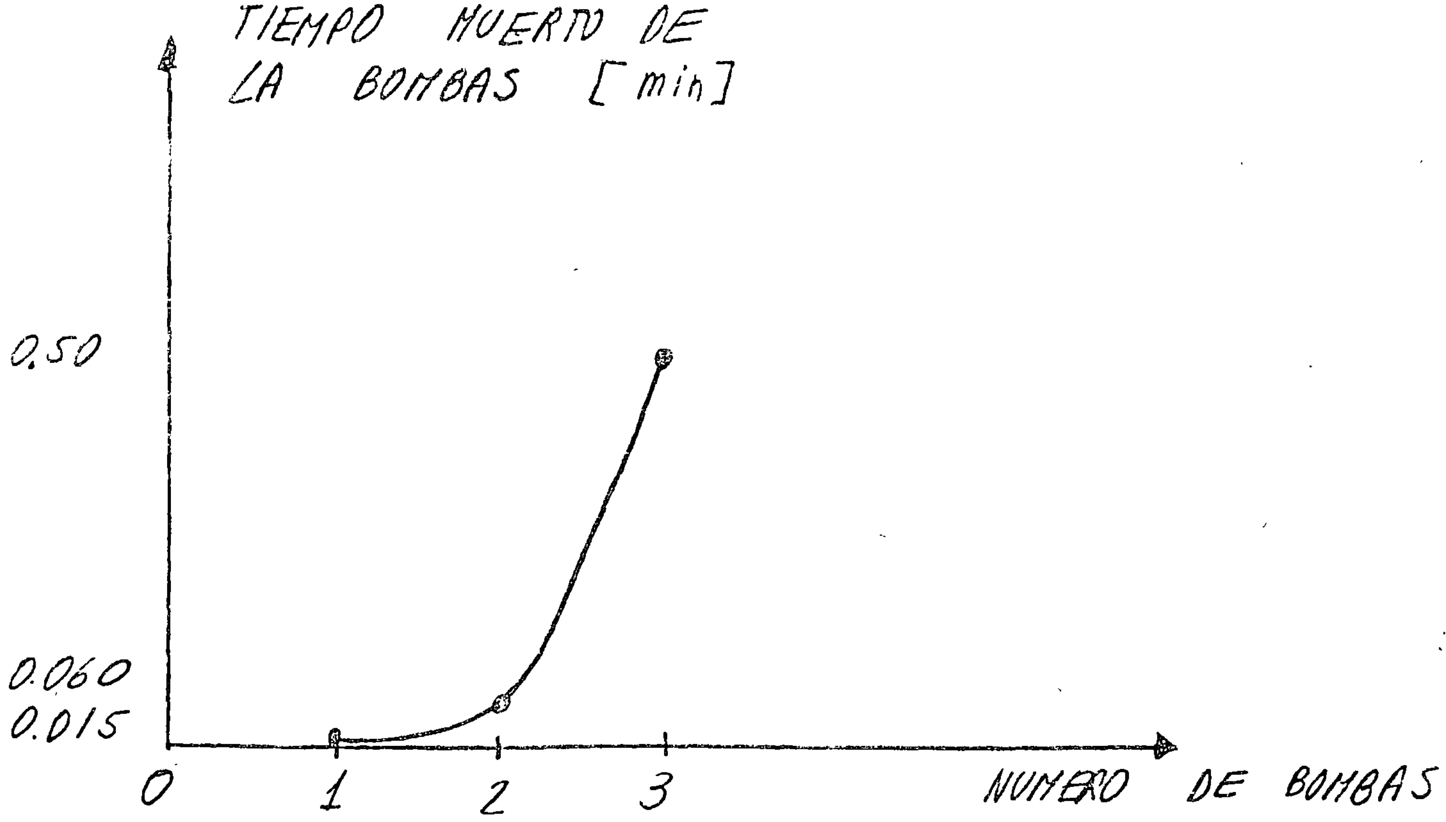
EL PROBLEMA CONSISTE EN DETERMINAR EL NUMERO DE BOMBAS N QUE SATISFAGAN e).


MODELO DE TRABAJO
MODELO
RESPUESTAS
GENERACION DE INFORMACION
TIEMPO DE SERVICIO
INTERVALO DE LLEGADA
GASOLINERA
NUMERO DE BOMBAS
RESPUESTAS
NUMERO DE AUTOS ATENDIDOS Y NUMERO DE AUTOS RECHAZADOS
REPETICION DEL CICLO PARA SIMULAR 90 AUTOS EN OPERACION

SIMULACION PARA 90 AUTOMÓVILES


GENERAR LISTA DE : TIEMPOS DE LLEGADA TIEMPOS DE SERVICIO DESVIACIONES EN EL TIEMPO DE SERVICIO
INICIAR CON UNA BOMBA DE GASOLINA
ES EL CONTADOR DE VEHICULOS QUE LLEGAN > 90
NO
SI
VER CUALES AUTOS SE HAN RETIRADO Y REGISTRARLOS
BUSCAR COLA CON MENOS VEHICULOS
ES EL NUMERO DE VEHICULOS EN LA(S) BOMBA(S) > 4
SI
NO
REGISTRAR EL NUMERO DE AUTOS RECHAZADOS
INCREMENTA EL NUMERO DE AUTOMOVILES QUE LLEGAN
INCREMENTAR EL NUMERO DE VEHICULOS EN LA BOMBA
REGISTRAR TIEMPO MUERTO DE LA BOMBA ACTIVADO
CALCULAR TIEMPO DE ESPERA
CALCULAR TIEMPO DE ESPERA TOTAL
ES EL PORCENTAJE DE AUTOMOVILES ATENDIDOS > 95%
SI
NO
STOP
AUMENTAR EL NUMERO DE BOMBAS E INICIALIZAR


# AUTOMOVILES
RECHAZADOS
55
10
1
0
1
2
3
NUMERO DE BOMBAS


TIEMPO MUERTO DE
LA BOMBAS [min]
0,50
0.060
0.015
0
1
2
3
NUMERO DE BOMBAS

VI TRANSPORTE PLUVIAL

DEFINICION : UNA FUNCION DE PRODUCCION ES UNA EXPRESION GENERAL DE TODAS LAS SALIDAS QUE PUEDEN SER OBTENIDAS DE TODAS AQUELLAS COMBINACIONES TECNICAMENTE EFICIENTES DE LAS ENTRADAS.

NOTA — LAS CARACTERISTICAS PUEDEN SER DEFINIDAS ATRAVES DE UN CONOCIMIENTO DETALLADO DEL PROCESO FISICO O ATRAVES DE ANALISIS ESTADISTICO DE LA INFORMACION

PARA EL CASO DEL TRANSPORTE PLUVIAL LA SALIDA ESTA DEFINIDA POR EL PRODUCTO :

$$Z = S \cdot C$$

EN DONDE S = VELOCIDAD DE LA BARCAZA (MILLAS / HORA
C = CAPACIDAD DE LA BARCAZA (TONELADAS)

LA VELOCIDAD S DE LA EMBARCACION PUEDE SER DETERMINADA DEL SIGUIENTE CONJUNTO DE ECUACIONES :

$$S = S^* + (-1)^{\delta} S_W$$

EN DONDE : S^* = VELOCIDAD DE LA BARCAZA CON RESPECTO AL AGUA
S_W = VELOCIDAD DEL AGUA

$\delta = 0$ SI SE VIAJA RIO ABAJO.
$\delta = 1$ SI SE VIAJA RIO ARRIBA

$$S^{*} = -1.14\ HP + \frac{\left[1.3039\ HP^{2} - 31.8\ HP - 0.38\ HP\cdot D - 4\beta\cdot(-1)^{\delta+1} F\right]^{1/2}}{2\beta}$$

donde :

$$F = 0.00086\ S_W^{2}\ D^{-4/3}\left[(52+0.44H)HLB + 24300 + 350\ HP - 0.021\ HP^{2}\right]$$

$$\beta = 0.0729\ e^{1.46/(D-H)}\ H^{0.6+(50/W-B)}\ L_c^{0.38}\ B^{1.19} + 172$$

$$\left.\begin{array}{lcl} HP & = & \text{POTENCIA AL FRENO} \\ D & = & \text{PROFUNDIDAD DEL CANAL} \\ W & = & \text{ANCHO DEL CANAL} \end{array}\right\} \text{POTENCIA DE LA BARCAZA}$$

$$\left.\begin{array}{lcl} H & = & \text{PROFUNDIDAD DE LA BARCAZA} \\ L & = & \text{LONGITUD DE LA BARCAZA} \\ B & = & \text{ANCHO DE LA BARCAZA} \end{array}\right\} \text{AREA DE LAS CUBIERTAS}$$

$$\Longrightarrow \quad Z = h\,(\text{AREA DE CUBIERTAS, POTENCIA})$$

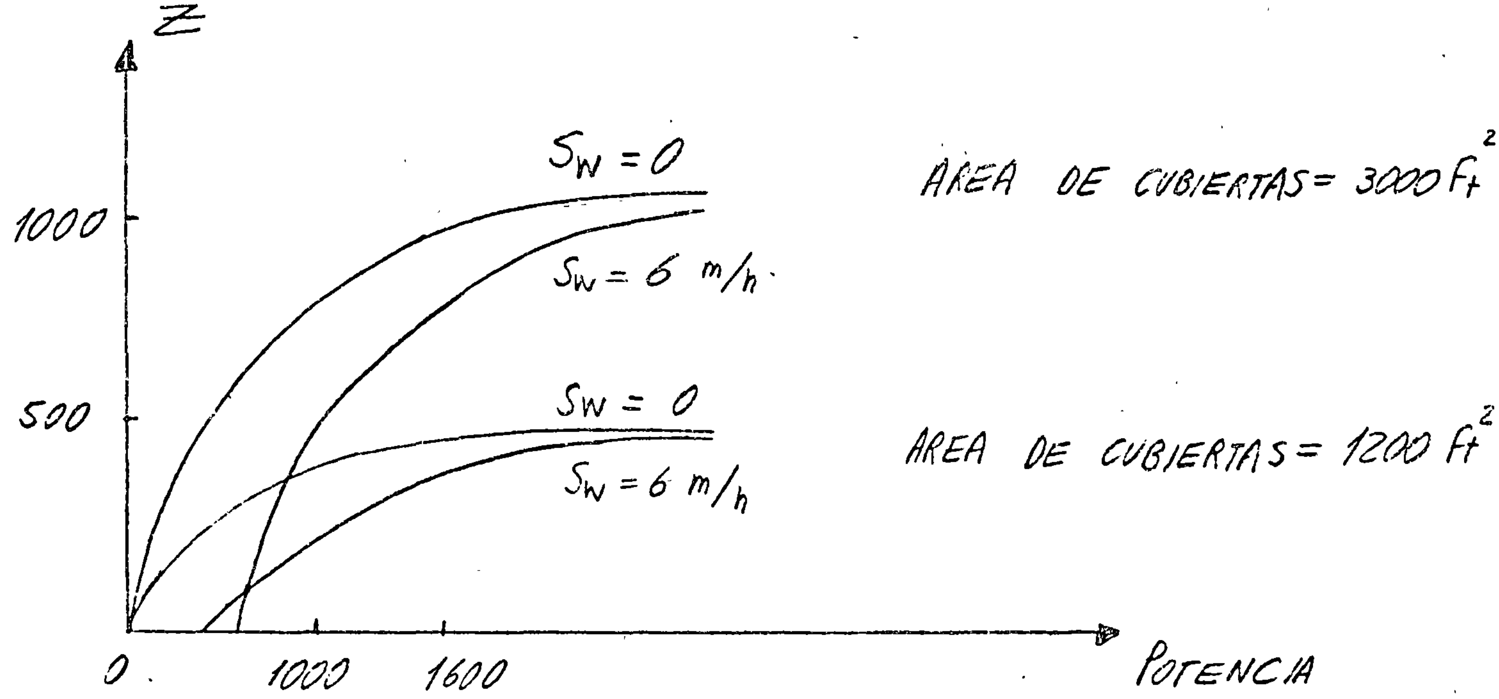
$$\text{ICION DE} \atop \text{DUCCIÓN} \quad = h\left(\begin{array}{l}\text{COSTOS DE CONSTRUCCION} \\ \text{COSTOS DE OPERACION}\end{array}\right)$$

$$Z$$

(REFERENCIA NUMERO 3)

EXPLORACION DE LA FUNCION DE PRODUCCION

I)



W = 100' L = 80'
D = 20' B = 25'
S_W VARIA H = 3'

$$\text{PRODUCTOS MARGINALES}_i = \frac{\partial Z}{\partial \text{ENTRADA } i} \quad (\text{DECRECIENTES})$$

NOTAS : i) A MEDIDA QUE LA VELOCIDAD ANGULAR AUMENTA EL FLUJO DE AGUA QUE LLEGA A LAS ASPAS DE LA BARCAZA SE REDUCE; POR TANTO NO ES POSIBLE SACAR MAXIMA VENTAJA DE POTENCIA EXTRA.

ii) EL PRIMER EFECTO DE LA EXISTENCIA DE UNA CORRIENTE ES UNA DISMINUCION EN EL PRODUCTO Z QUE PUEDA SER OBTENIDO CON

CUALQUIER COMBINACION DE ENTRADAS.

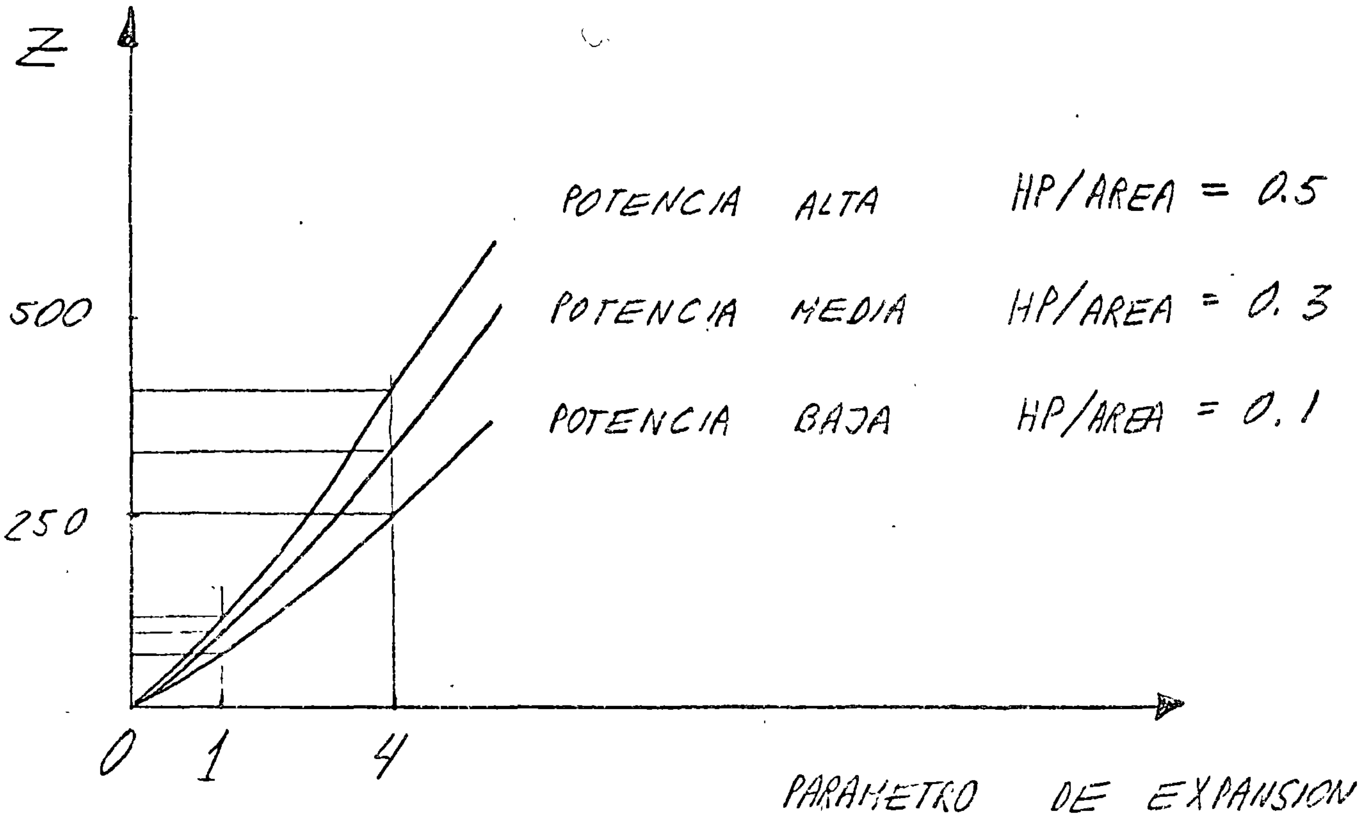
iii) LA EXISTENCIA DE UNA CORRIENTE ORIGINA QUE ALGUNOS PRODUCTOS MARGINALES DE ALGUNAS ENTRADAS SEAN NEGATIVOS. (PUEDE ENTONCES RESULTAR TECNICAMENTE DEFICIENTE INCREMENTAR ALGUNA ENTRADA

II) FACTORES DE ESCALA

SE REFIEREN UNICAMENTE A LA RELACION ENTRE CAMBIOS ENTRE LOS VALORES DE LA SALIDA Y CAMBIOS EN LOS VALORES DE TODAS LAS ENTRADAS SIMULTANEAMENTE Y EN LAS MISMAS PROPORCIONES.
LA EXISTENCIA DE FACTORES DE ESCALA CRECIENTES PARA UNA FUNCION DE PRODUCCION ES IMPORTANTE, YA QUE DE ESTA FORMA PODEMOS SOBREDISEÑAR INICIALMENTE UN ELEMENTO PARA AHORRAR, CUANDO UN INCREMENTO EN LA DEMANDA ES ESPERADO EN EL FUTURO.

EJEMPLO :

SI DUPLICAR O TRIPLICAR TODAS LAS ENTRADAS IMPLICA DUPLICAR O TRIPLICAR LA SALIDA, SE DICE QUE LA FUNCION DE PRODUCCION POSEE FACTORES DE ESCALA CONSTANTES.
SI DUPLICAR TODAS LAS ENTRADAS AUMENTA A MAS DEL DOBLE LA SALIDA, LA FUNCION DE PRODUCCION POSEE FACTORES DE ESCALA CRECIENTES; SI SON MENORES QUE EL DOBLE DE LA SALIDA, LA FUNCION DE PRODUCCION POSEERA FACTORES DE ESCALA DECRECIENTES.

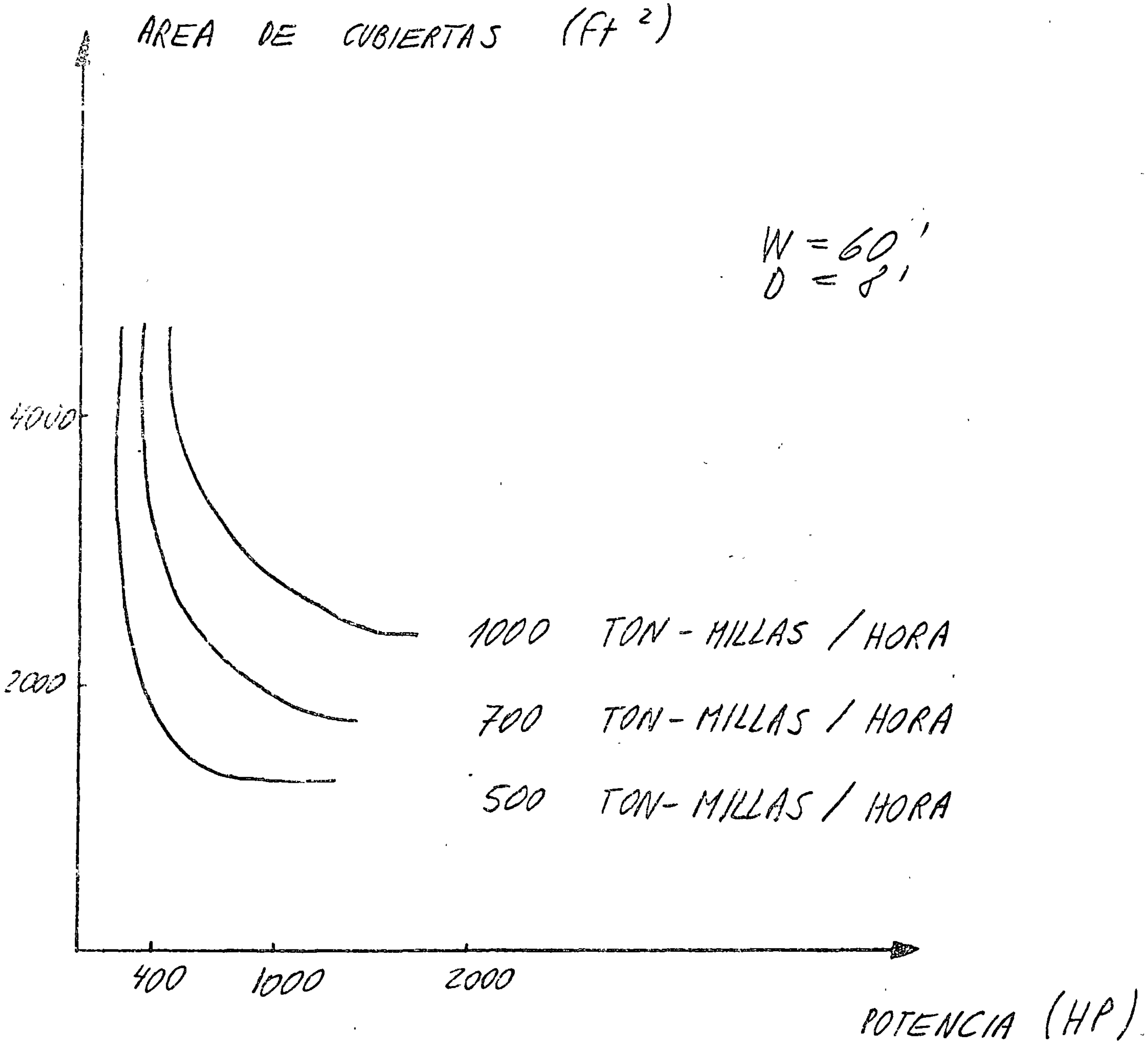


$W = 60'$
$D = 8'$
$S_W = 0$

⟹ LA FUNCION DE PRODUCCION PRESENTA FACTORES DE ESCALA CONSTANTES PARA POTENCIAS BAJAS Y FACTORES DE ESCALA CRECIENTES PARA POTENCIAS MEDIAS Y ALTAS.
(ESTE FENOMENO JUSTIFICA EL CRECIMIENTO DE LOS BUQUES TANQUE INTER OCEANICOS)

III) ISOCUANTAS

LA FUNCION DE PRODUCCION Z PUEDE SER REPRESENTADA POR EL LUGAR GEOMETRICO DE LOS PUNTOS CON IGUAL SALIDA (ISOCUANTAS)



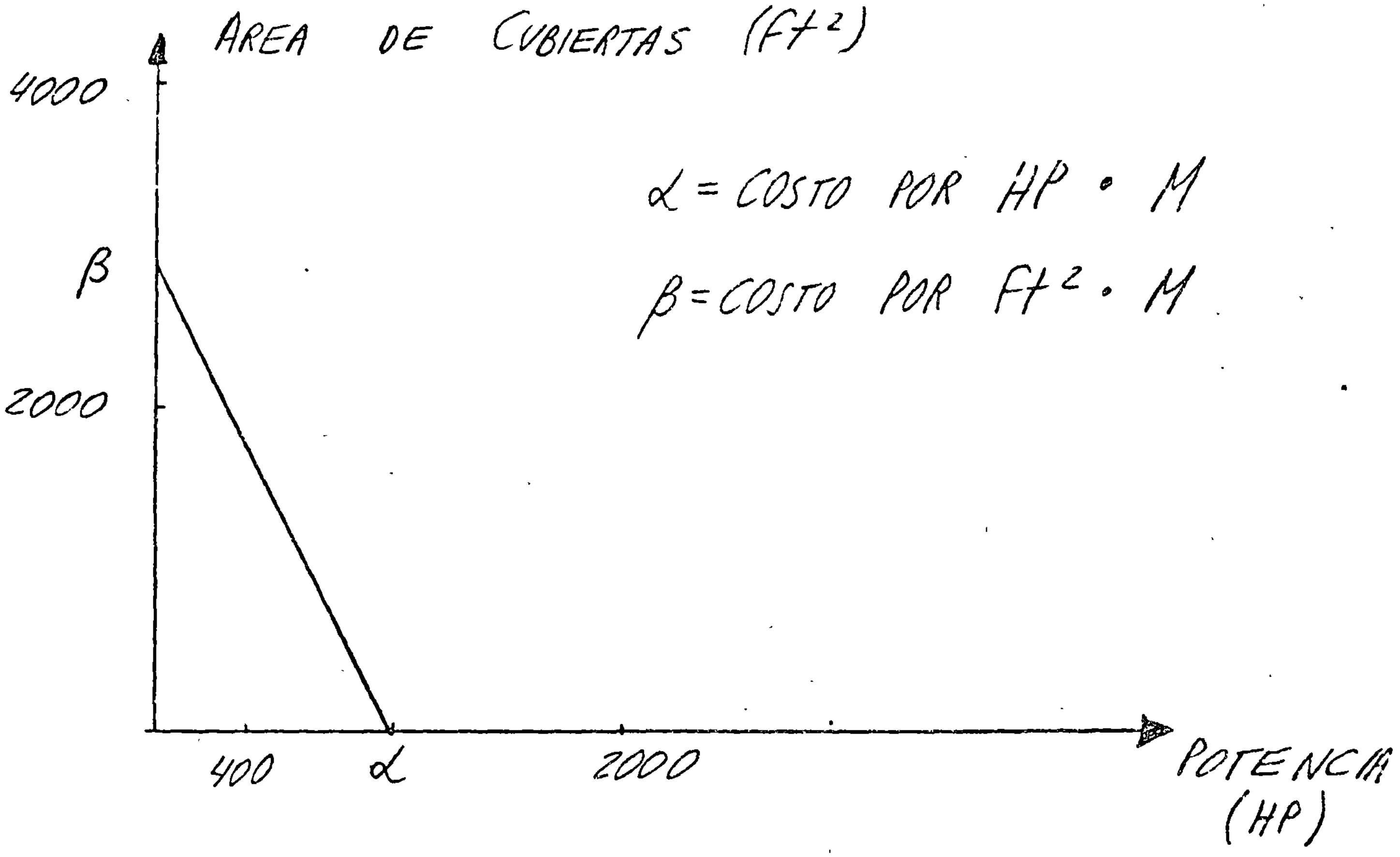
LA PENDIENTE DE LA ISOCUANTA ES IGUAL A LA RELACION MARGINAL DE SUSTITUCION DE LAS ENTRADAS DEFINIDA COMO :

$$MRS = -\frac{\text{PRODUCTO MARGINAL } i}{\text{PRODUCTO MARGINAL } j}$$

MRS ES LA MEDIDA DE LAS CONDICIONES EN QUE SE ESTA DISPUESTO A PERMUTAR UN POCO DE UNA DE LAS VARIABLES DE ENTRADA POR UN POCO MÁS DE LA OTRA. (EL SIGNO NEGATIVO PROVIENE DE ESTA PERMUTACION)

IV) RECTA DE BALANCE

PARTIENDO DE LOS COSTOS POR HP, DE LOS COSTOS POR Ft^2 DE CUBIERTAS Y DE LA CANTIDAD DE DINERO DISPONIBLE M ES POSIBLE OCUPAR CUALQUIERA DE LAS POSICIONES SOBRE LA RECTA DE BALANCE



α = COSTO POR HP · M

β = COSTO POR Ft^2 · M

EN ALGUNO DE LOS PUNTOS DE LA RECTA DE BALANCE SE ALCANZA LA ISOCUANTA MAS ELEVADA. ESE PUNTO OPTIMO ES EL DE TANGENCIA DE LA RECTA DE BALANCE CON LA ISOCUANTA DE MAYOR SALIDA Z.
EN ESE PUNTO, LA RELACION MARGINAL DE SUSTITUCION ES IGUAL A LA RELACION DE PRECIOS.

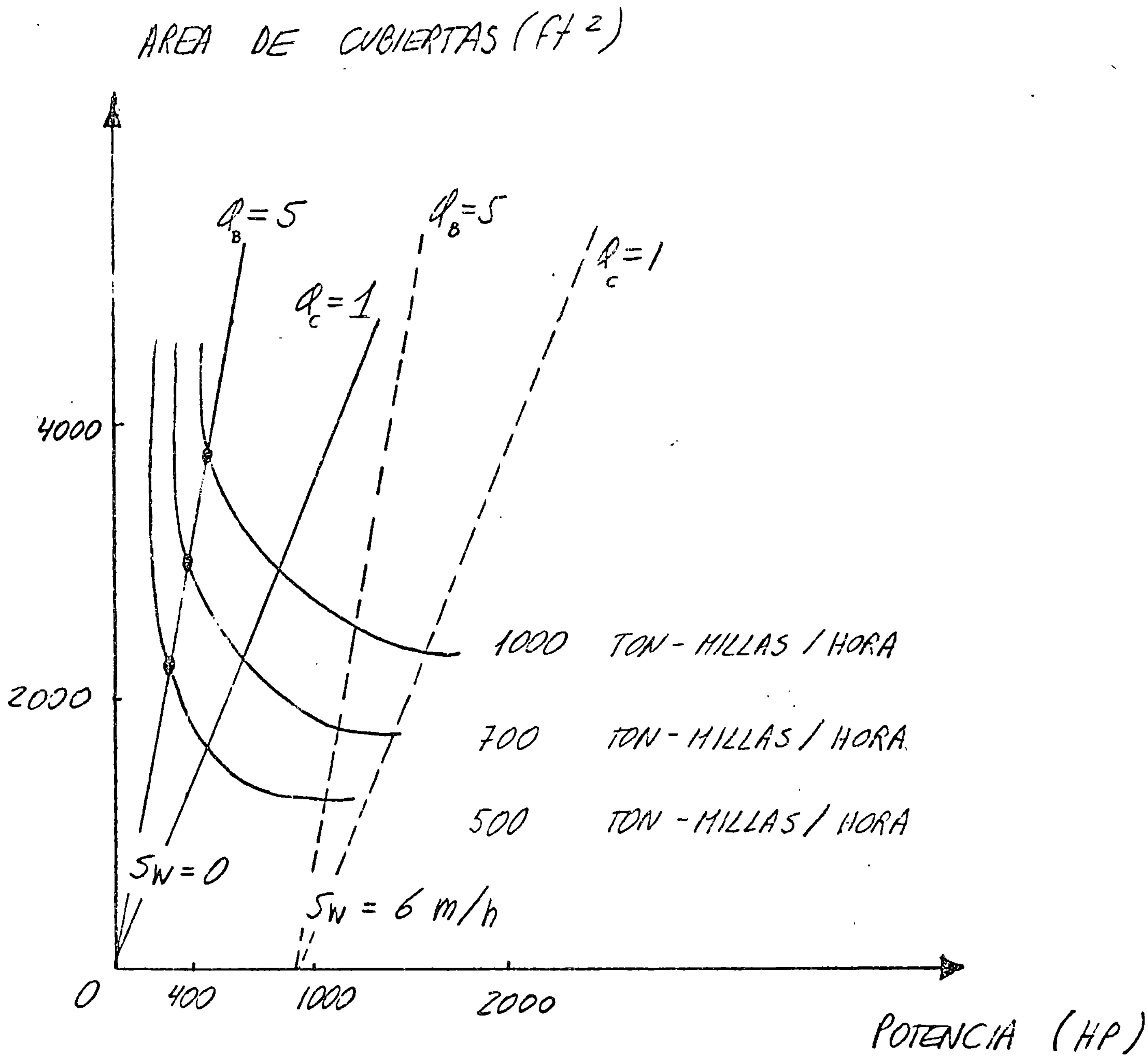
V) TRAYECTORIAS DE EXPANSION.

SON LOS LUGARES GEOMETRICOS DE DISEÑO OPTIMO PARA EL CONJUNTO ESPECIFICO DE RELACIONES ECONOMICAS ENTRE LAS VARIABLES DE ENTRADA. CADA PUNTO DEL LUGAR GEOMETRICO CUMPLE CON LA CONDICION:

$$Q = \frac{\text{COSTO POR HP}}{\text{COSTO POR } ft^2 \text{ DE CUBIERTAS}}$$

⟹ DADO QUE, EN EL OPTIMO, LA RELACION MARGINAL DE SUSTITUCION MRS ES IGUAL A LA RELACION DE PRECIOS

⟹ $MRS = -Q$



$W = 60'$
$D = 8'$

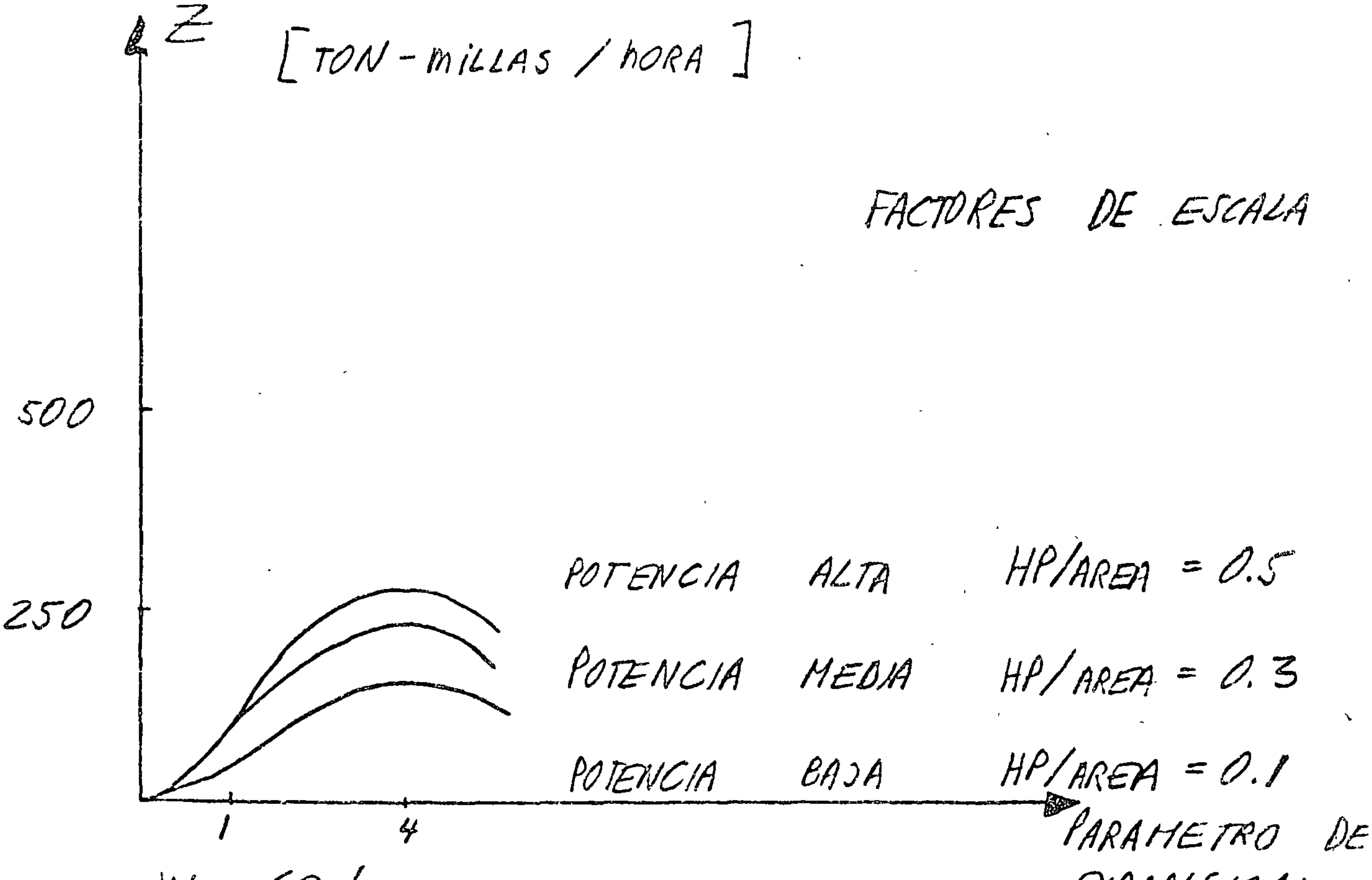
CUANDO $S_W = 0$ LAS TRAYECTORIAS DE EXPANSION Q PASAN POR EL ORIGEN. Z ES ENTONCES LINEAL Y HOMOGÉNEA. (LA RELACION ENTRE LAS ENTRADAS EN EL DISEÑO OPTIMO NO CAMBIA PARA DIFERENTES ESCALAS DE DISEÑO)

CUANDO $S_W \neq 0$, Z SE CONVIERTE EN LINEAL, NO HOMOGENEA. Q SON LINEALES PERO NO PASAN POR EL ORIGEN. (LA RELACION ENTRE LAS ENTRADAS EN EL DISEÑO OPTIMO VARIA PARA DIFERENTES ESCALAS DE DISEÑO, AUN CUANDO LOS PRECIOS RELATIVOS DE LAS ENTRADAS SEAN FIJOS Y LINEALES EN CANTIDAD)

VI RESTRICCIONES DEL CANAL

EL CANAL PUEDE IMPONER LIMITACIONES EN EL DISEÑO DE LAS BARCAZAS.

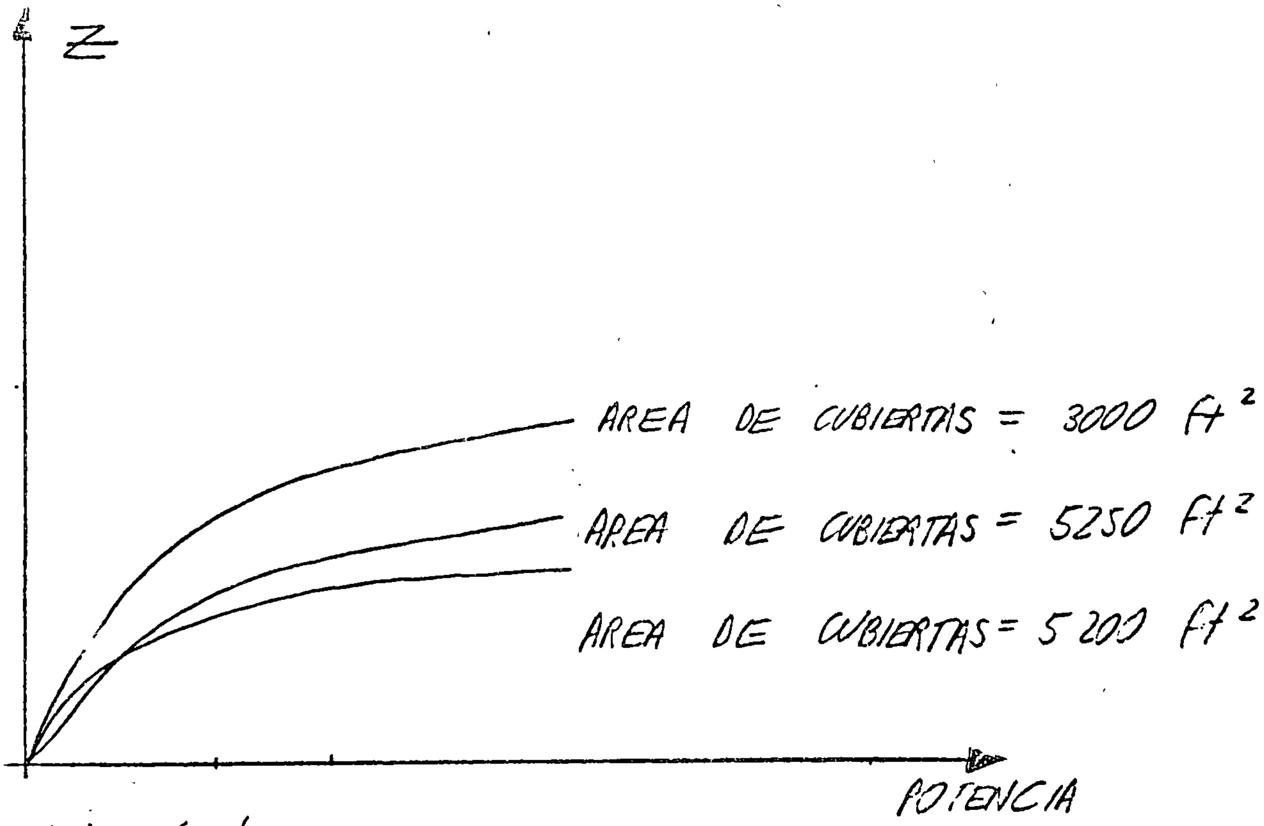
i) PROFUNDIDAD
ii) ANCHURA
iii) LONGITUD



W = 60'
D = 8'
$S_W = 0$

LONGITUD ≤ 125'

LA EXISTENCIA DE RESTRICCIONES EN EL DISEÑO NOS LLEVA A IDENTIFICAR DISEÑOS DOMINANTES . ESTO ES UNA CONSECUENCIA DEL DECRECIMIENTO DE LOS FACTORES DE ESCALA QUE PUEDEN EXISTIR . (ESTOS DISEÑOS NO EXISTIAN ANTERIORMENTE ; UN DISEÑO MAYOR PODRIA OFRECER SIEMPRE UNA SALIDA MAYOR)



$W = 60'$
$D = 8'$
$S_W = 0$

BIBLIOGRAFIA

1) SYSTEMS PLANNING AND DESIGN
R. DE NEUFVILLE AND D. MARKS
PRENTICE HALL 1974

2) EL ENFOQUE DE SISTEMAS
GEREZ - GRIJALVA
LIMUSA 1976

3) HOWE, C.W 1965
METHODS FOR EQUIPMENT SELECTION AND BENEFIT EVALUATION IN INLAND WATERWAY TRANSPORTATION
WATER RESOURCES RESEARCH
VOL. I P.P. 25 - 39

4) DESIGN AND PLANNING OF ENGEENIRING SYSTEMS
D. MEREDITH, K. WONG, R WOODHEAD, R. WORTMAN
PRENTICE HALL

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

APLICACIONES DE LA COMPUTADORA A LA SIMULACION Y OPTIMIZACION

TEMA: OPTIMIZACION

PROF. M. en C. VERONICA CZITROM.

marzo - abril, 1978.

OPTIMIZACIÓN

FORMULACIÓN MATEMÁTICA GENERAL:

ENCONTRAR EL VALOR DE LAS VARIABLES

$(x_1, x_2, \dots, x_n)$

QUE MAXIMICEN O MINIMICEN (OPTIMICEN) A LA FUNCIÓN (**OBJETIVO**)

$$M = M(x_1, x_2, \dots, x_n)$$

SUJETA A LAS **RESTRICCIONES**

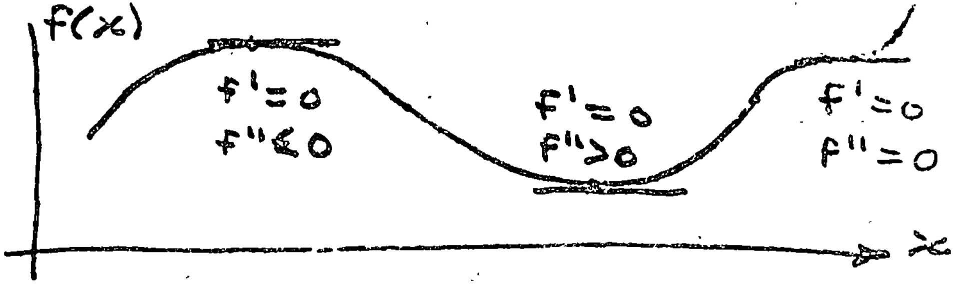
$$C_i\,(x_1, x_2, \dots, x_n) = 0 \qquad i = 1, \dots, p$$
$$C_i\,(x_1, x_2, \dots, x_n) \leq 0 \qquad i = p+1, \dots, r$$
$$C_i\,(x_1, x_2, \dots, x_n) \geq 0 \qquad i = r+1, \dots, m$$

m ECS., n VARIABLES (INCÓGNITAS)

2 ESTRATEGIAS DE OPTIMIZACIÓN: $\begin{cases} \text{GRADIENTE} \\ \text{ENUMERACIÓN} \end{cases}$

RESTRICCIONES → REGIÓN DE SOLUCIONES FACTIBLES

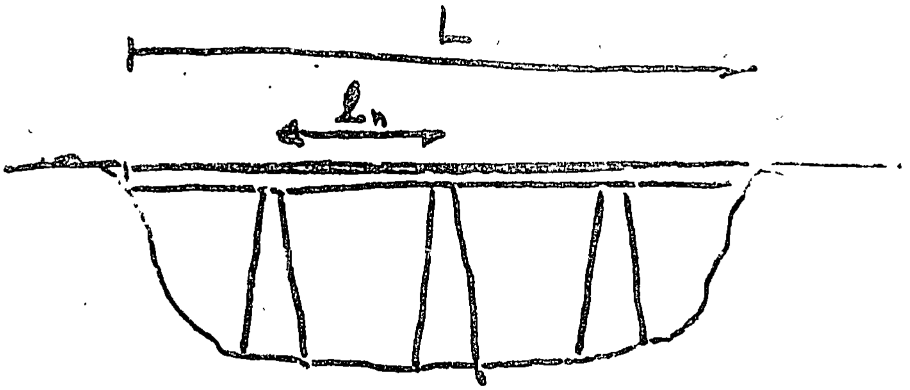
OPTIMIZACIÓN POR DIFERENCIACIÓN



$f(\underset{\uparrow}{x})$

UNA SOLA VAR. INDEPENDIENTE

EJEMPLO.



l_n = ? PARA MIN. COSTO

COSTO POR METRO PUENTE ∼ DIST. ENTRE PILARES

COSTO PILARES ≃ CONST.

n = # CLAROS = L/L_n

A = COSTO (FIJO) PILARES TERMINALES

P = " PILASTRA INTERMEDIA

ω = COSTO POR METRO DE PUENTE

$$\omega = k L_n$$

$$\text{MIN: COSTO} = A + (n-1)P + L \underbrace{k L_n}_{\omega}$$

$$= A + \left(\frac{L}{L_n} - 1\right)P + L k L_n$$

$$\frac{dC}{dL_n} = -\frac{LP}{L_n^2} + kL = 0$$

$$\therefore \boxed{L_n = \sqrt{\frac{P}{k}}}$$

$$\frac{d^2C}{dL_n^2} = \frac{2LP}{L_n^3} > 0 \Rightarrow$$ COSTO MÍNIMO

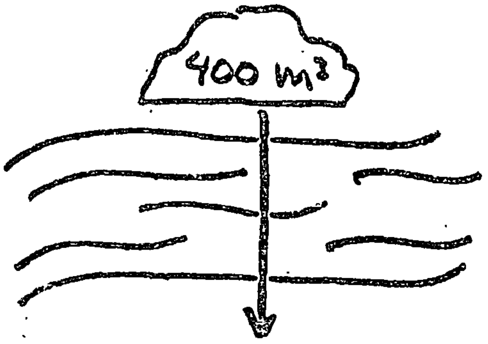
VARIAS VARIABLES INDEPENDIENTES:

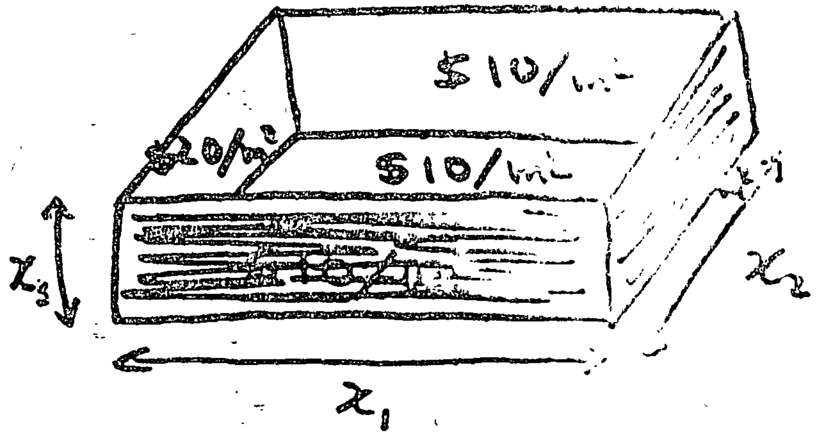
$f(x_1, \dots, x_n)$

$$\text{SI: } \begin{cases} \dfrac{\partial f}{\partial x_1} = 0 \\ \vdots \\ \dfrac{\partial f}{\partial x_n} = 0 \end{cases}$$

ENTONCES LA FUNCIÓN F ES MAX O MIN

EJEMPLO





CADA VIAJE: $.10

$x_1, x_2, x_3 = ?$, x_4 = NUM VIAJES = ?

MIN: COSTO TRANSPORTE

$$\text{COSTO} = 10(x_1 x_2 + 2 x_1 x_3) + 20(2 x_2 x_3) + .10 x_4$$
$$= 10 x_1 x_2 + 20 x_1 x_3 + 40 x_2 x_3 + .10 x_4$$

RESTRICCIÓN: $\underbrace{x_1 x_2 x_3 x_4}_{\text{\# VIAJES x VOLUMEN CAJA}} = \underbrace{400}_{\text{MATERIAL TRANSPORTADO}}$

DESPEJANDO $x_4 = \dfrac{400}{x_1 x_2 x_3}$ Y SUST.

MIN: COSTO $= 10 x_1 x_2 + 20 x_1 x_3 + 40 x_2 x_3 + \dfrac{40}{x_1 x_2 x_3}$

$$\left\{\begin{aligned}
\frac{\partial C}{\partial x_1} &= 10x_2 + 20x_3 - \frac{40}{x_1^2 x_2 x_3} = 0 \\
\frac{\partial C}{\partial x_2} &= 10x_1 + 40x_3 - \frac{40}{x_1 x_2^2 x_3} = 0 \\
\frac{\partial C}{\partial x_3} &= 20x_1 + 40x_2 - \frac{40}{x_1 x_2 x_3^2} = 0
\end{aligned}\right.$$

SOLUCIÓN $x_1 = 2$ $\quad x_4 = \dfrac{400}{x_1 x_2 x_3} = 400$

$x_2 = 1$

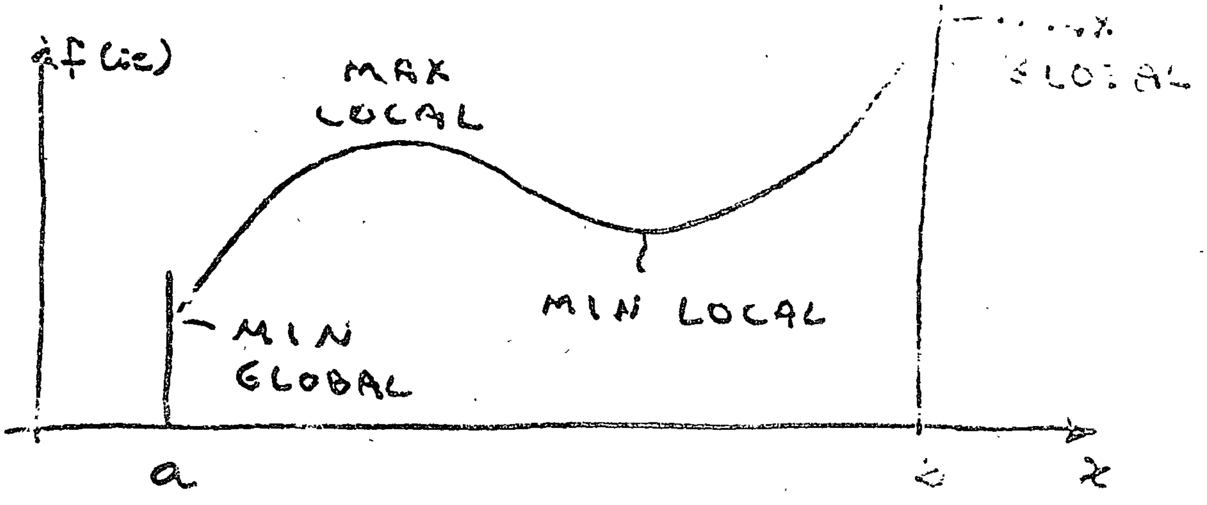
$x_3 = 1/2$

$C = \$100$

OPTIMIZACIÓN DE FUNCIONES DE UNA SOLA VARIABLE, SIN RESTRICCIONES.

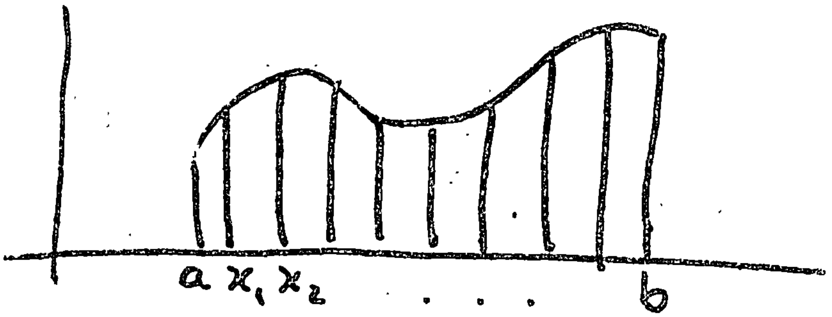
MET. SIMULTAN: BÚSQUEDA EXHAUSTIVA
BÚSQUEDA ALEATORIA

MET. SECUENC: TRISECCIÓN, FIBONACCI } UN SOLO MAX O MIN EN INTERVALO (UNIMODAL



DIFERENCIACIÓN D

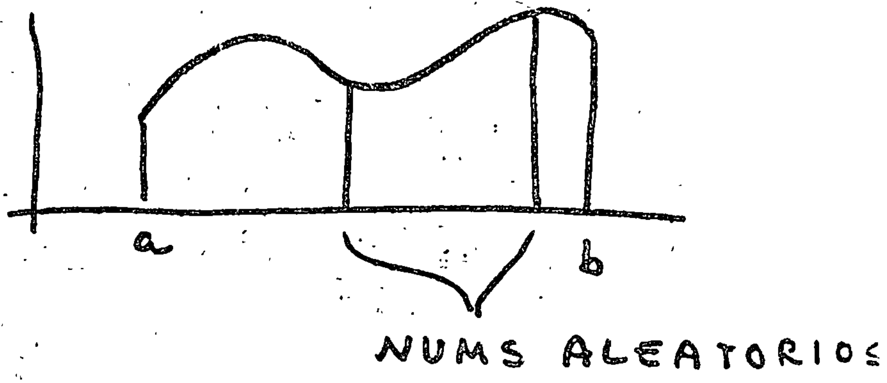
BÚSQUEDA EXHAUSTIVA



MUCHAS EVALUACIONES

AUMENTA NUM. INTERVALOS, AUMENTA PRECISIÓN

BÚSQUEDA ALEATORIA

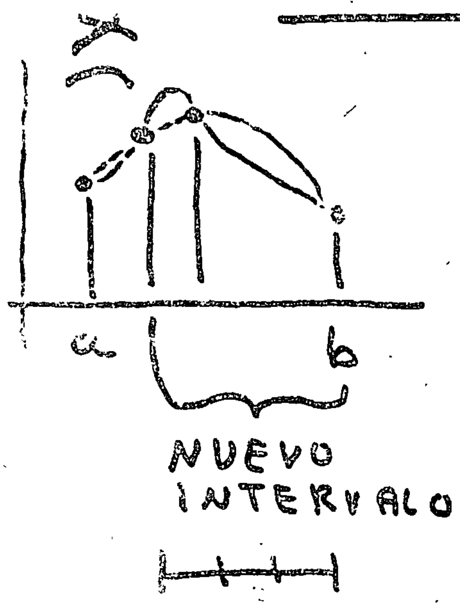


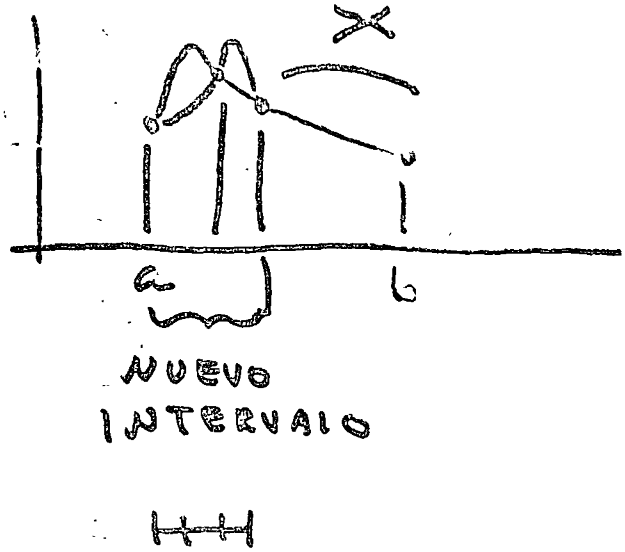
EJEMPLO: $f(x) = -.4x^2 + 4x$ EN $(0,10)$

DIFERENCIANDO: $f(\underset{x}{5}) = 10$ EXACTO

# DE NUMS. ALEATORIOS	x	$f(x)$
25	4.9051	9.9964
100	4.9051	9.9964
250	4.9091	9.9966
500	5.0088	9.9999

TRISECCIÓN

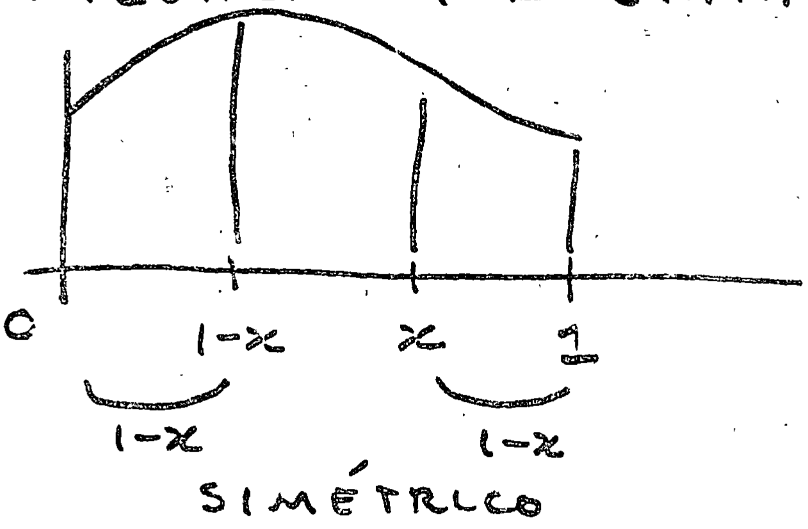


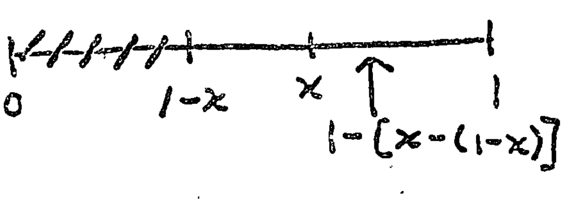


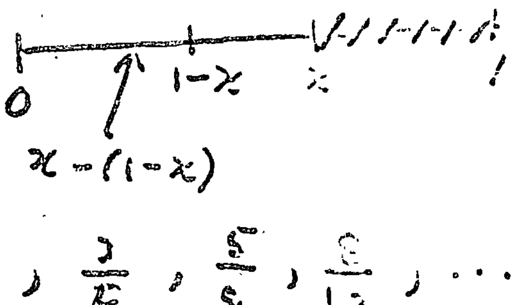
HASTA LLEGAR A INTERVALO SUF. CHICO
EN CADA ETAPA, SE EVALUA F EN 2 PUNTOS

FIBONACCI

1 EVALUACIÓN POR ETAPA





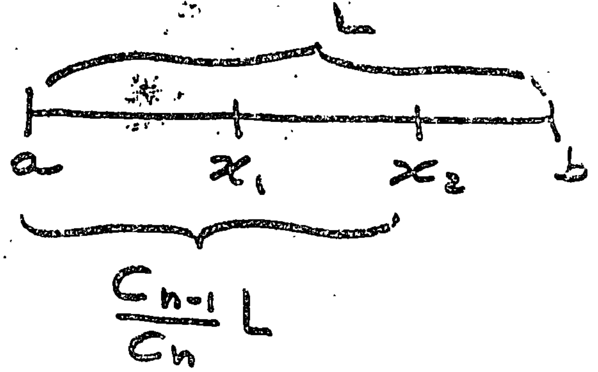


$$= \frac{\text{INT. DESP. ITER.}}{\text{INT. ANTES ITER}} = \frac{1}{2}, \frac{2}{3}, \frac{3}{5}, \frac{5}{8}, \frac{8}{13}, \ldots \to .618$$

(COCIENTE DORADO)

$r = \frac{2}{3}$ TRISECCIÓN (SE ELIMINA $1/3$ DEL INTERVALO)

FIBONACCI: 40% EVALUACIONES TRISECCIÓN



OPTIMIZACIÓN CON RESTRICCIONES:

MULTIPLICADORES DE LAGRANGE

OPTIMIZAR: FUNCIÓN OBJETIVO $M = M(x_1, x_2, \dots, x_n)$

SUJETA A RESTRICCIONES TIPO IGUALDAD

$$C_i(x_1, x_2, \dots, x_n) = 0 \qquad i = 1, 2, \dots, m$$

n PUNTO OPTIMO $P^*(\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n)$: $dM = \sum_{i=1}^{n} \frac{\partial M}{\partial x_i} dx_i \Big|_{P^*} = 0$ ①

n CUALQUIER PUNTO: $dC_j = \sum_{i=1}^{n} \frac{\partial C_j}{\partial x_i} dx_i = 0$
(..TAMBIEN EN P*) $\qquad j = 1, 2, \dots, m$

MULTIPLICANDO POR λ_j, Y RESTANDO ESAS M ECUACIONES DE ① SE TIENE:

$$dM - \sum_{j=1}^{m} dC_j = \sum_{i=1}^{n} \left(\frac{\partial M}{\partial x_i} - \sum_{j=1}^{m} \lambda_j \frac{\partial C_j}{\partial x_i} \right) dx_i \Big|_{P^*} = 0$$

COMO dx_i SON ARBITRARIAS,

$$\frac{\partial m}{\partial x_i} - \sum_{j=1}^{m} \lambda_j \frac{\partial C_j}{\partial x_i} = 0 \qquad i = 1, 2, \dots, n$$

$$\left\{\begin{array}{ll} \text{LAS } n \text{ ECS} & \dfrac{\partial M}{\partial x_i} - \displaystyle\sum_{j=1}^{m} \lambda_j \dfrac{\partial C_j}{\partial x_i} = 0 \qquad i=1,\dots,n \\ \text{Y LAS } m & \\ \text{RESTRICCIONES:} & C_j(x_1,\dots,x_n)=0 \qquad j=1,\dots,m \end{array}\right.$$

FORMAN UN SISTEMA DE $m+n$ ECUACIONES EN LAS INCOGNITAS $\hat{x}_1,\dots,\hat{x}_n$ (COORDENADAS DE P^*) Y $\lambda_1,\dots,\lambda_m$.

λ_i : MULTIPLICADORES DE <u>LAGRANGE</u>.

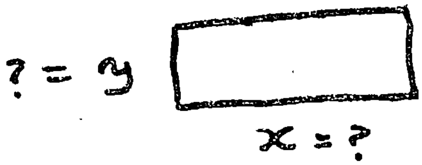
SI $L(x_1,x_2,\dots,x_n,\lambda_1,\dots,\lambda_m) = M - \sum_{j=1}^{m} \lambda_j C_j$

LAS $m+n$ ECS EN $m+n$ INCOGNITAS SON EQUIVALENTES A:

$$\left.\begin{array}{l} \left.\dfrac{\partial L}{\partial x_1}\right|_{P^*} = \left.\dfrac{\partial M}{\partial x_1}\right|_{P^*} - \displaystyle\sum_{j=1}^{m} \lambda_j \left.\dfrac{\partial C_j}{\partial x_1}\right|_{P^*} = 0 \\ \vdots \\ \left.\dfrac{\partial L}{\partial x_n}\right|_{P^*} = \left.\dfrac{\partial M}{\partial x_n}\right|_{P^*} - \displaystyle\sum_{j=1}^{m} \lambda_j \dfrac{\partial C_j}{\partial x_n} = 0 \end{array}\right\} n \text{ ECS.}$$

$$\left.\begin{array}{l} \dfrac{\partial L}{\partial \lambda_1} = -C_1(x_1,\dots,x_n) = 0 \\ \vdots \\ \dfrac{\partial L}{\partial \lambda_m} = -C_m(x_1,\dots,x_n) = 0 \end{array}\right\} m \text{ RESTRICCIONES}$$

EJEMPLO



AREA MÁXIMA

P = PERIMETRO CONOCIDO

FUNCION OBJETIVO : MAX AREA = $M = xy$

RESTRICCIONES: $P = 2x + 2y$

$C_1(x, y) = P - 2x - 2y = 0$

SOLUCIÓN POR DERIVADAS:

$$M = xy = x\left(\frac{P-2x}{2}\right)$$

$$\frac{\partial M}{\partial x} = \frac{P}{2} - 2x = 0, \qquad x = \frac{P}{4} \qquad y = \frac{P}{4}$$

SOLUCION POR MULT. DE LAGRANGE:

$$L = M - \lambda C = xy - \lambda(P - 2x - 2y)$$

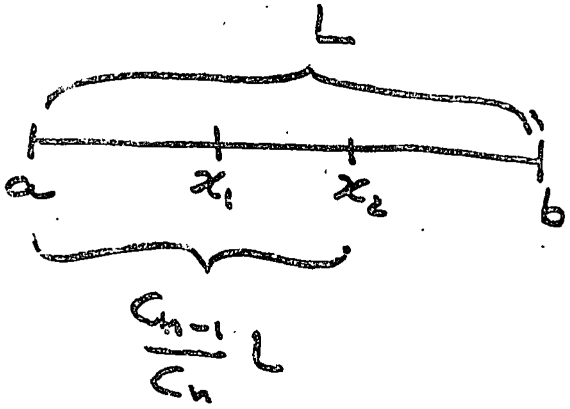
$$\begin{cases} \dfrac{\partial L}{\partial x} = y + 2\lambda = 0 \\ \dfrac{\partial L}{\partial y} = x + 2\lambda = 0 \\ \dfrac{\partial L}{\partial \lambda} = -P + 2x + 2y = 0 \end{cases}$$

SOLUCIÓN: $x = \frac{P}{4}$ $\quad y = \frac{P}{4}$ $\quad \lambda = -\frac{P}{8}$

METODO CLÁSICO: SE SUSTITUYEN RESTRICCIONES EN FUNCIÓN OBJETIVO PARA DISMINUIR # INCOGNITAS.
PUEDE SER ALGÉBRAICAMENTE COMPLICADO

METODO DE LAGRANGE: NO SE REQUIERE SUSTITUCIÓN.
PUEDE LLEVAR A SIST. DE ECS. COMPLICADO (POR EJ., NO LINEAL)

FIBONACCI: 40% EVALUACIONES TRISECCIÓN

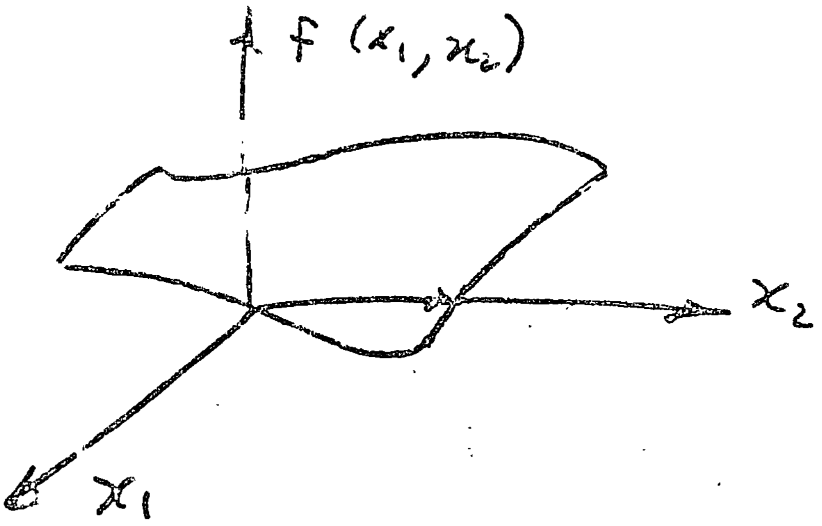


MÉTODOS DE BÚSQUEDA MULTIDIMENSIONAL

MET. SIMULTÁNEOS: BÚSQUEDA EXHAUSTIVA
BÚSQUEDA ALEATORIA

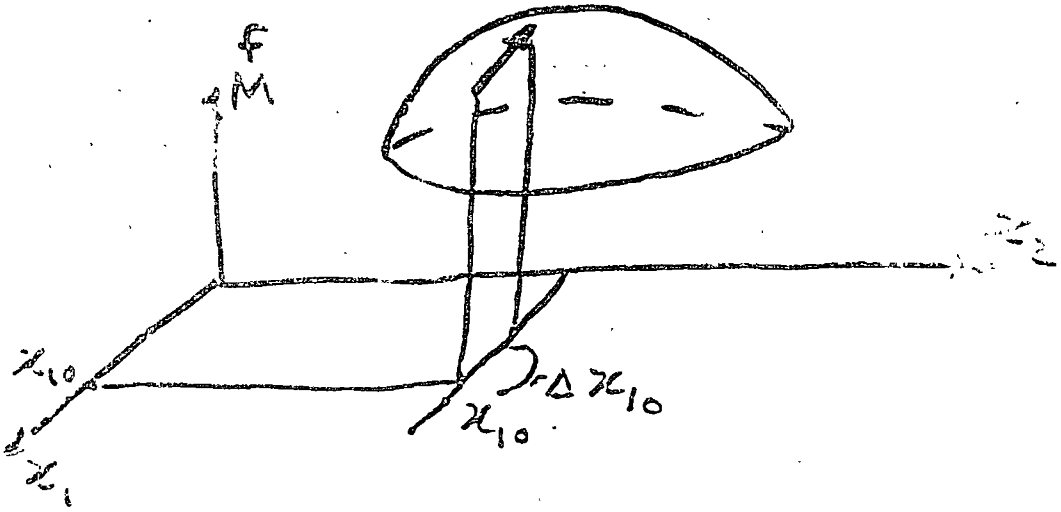
MET. SECUENCIALES: BUSQUEDA DE REJILLA
UNIVARIADA
DE GRADIENTE } FUN. UNIMODALES

$F(x_1, x_2, \ldots, x_n)$



TÉCNICAS DE GRADIENTE

$f(x_1, x_2, \ldots, x_n)$

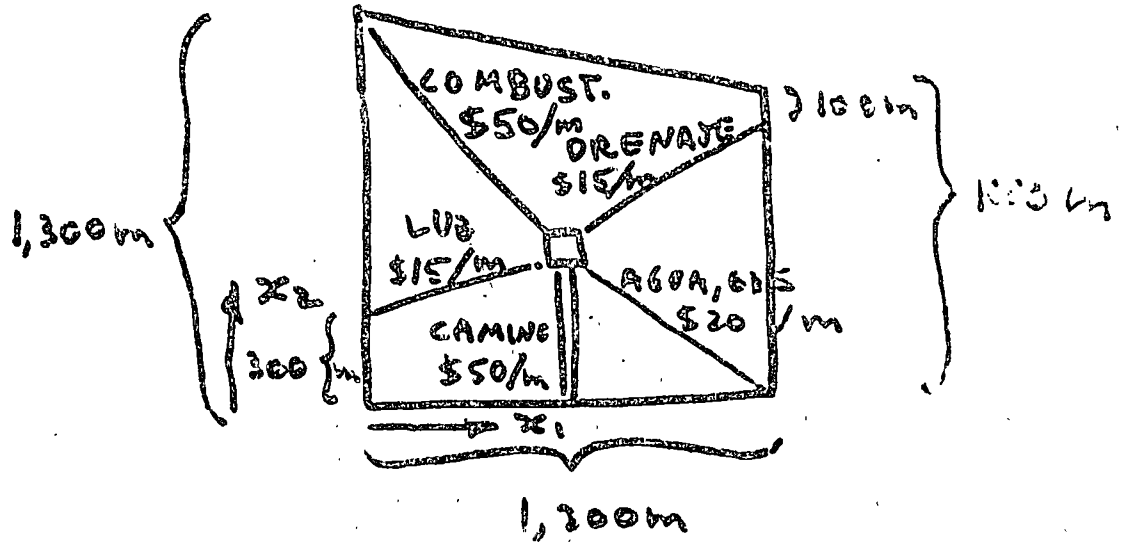


$$\Delta M = \left.\frac{\partial M}{\partial x_1}\right|_{\underline{x}_0} \Delta x_1 + \left.\frac{\partial M}{\partial x_2}\right|_{\underline{x}_0} \Delta x_2$$

$$\simeq \frac{M(x_{10} + \Delta x_{10}, x_{20}) - M(x_{10}, x_{20})}{\Delta x_1}$$

9

EJEMPLO: LOCALIZAR ÓPTIMAMENTE LA PLANTA PARA MINIMIZAR COSTOS



$$\text{MIN: COSTO} = 50x_2 + 15\{x_1^2 + (x_2-300)^2\}^{1/2}$$

$$+50\{x_1^2 + (1300-x_2)^2\}^{1/2} +$$

$$+15\{(1300-x_1)^2 + (900-x_2)^2\}^{1/2}$$

$$+20\{(1300-x_1)^2 + x_2^2\}^{1/2}$$

RESTRICCIONES: NO SALE DEL PREDIO

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

APLICACIONES DE LA COMPUTADORA A LA SIMULACION Y OPTIMIZACION

TEMA: PROGRAMACION.

M.en C. VERONICA CZITROM.

marzo - abril, 1978.

PROGRAMACIÓN LINEAL

ARQUITÉCTURA, INGENIERÍA, CONSTRUCCIÓN, PLANEACIÓN URBANA Y REGIONAL

MINIMIZAR COSTO
MAXIMIZAR GANANCIA

$\Big\}$ FUNCIONES OBJETIVO LINEALES

RESTRICCIONES LINEALES

EJEMPLO

MÁQUINA	# HORAS REQUERIDAS		# TOTAL HORAS DISPONIBLES
	PROD. 1	PROD. 2	
1	2	1	70
2	1	1	40
3	1	3	90
GANANCIA/PIEZA	40	60	

MAX. GANANCIA

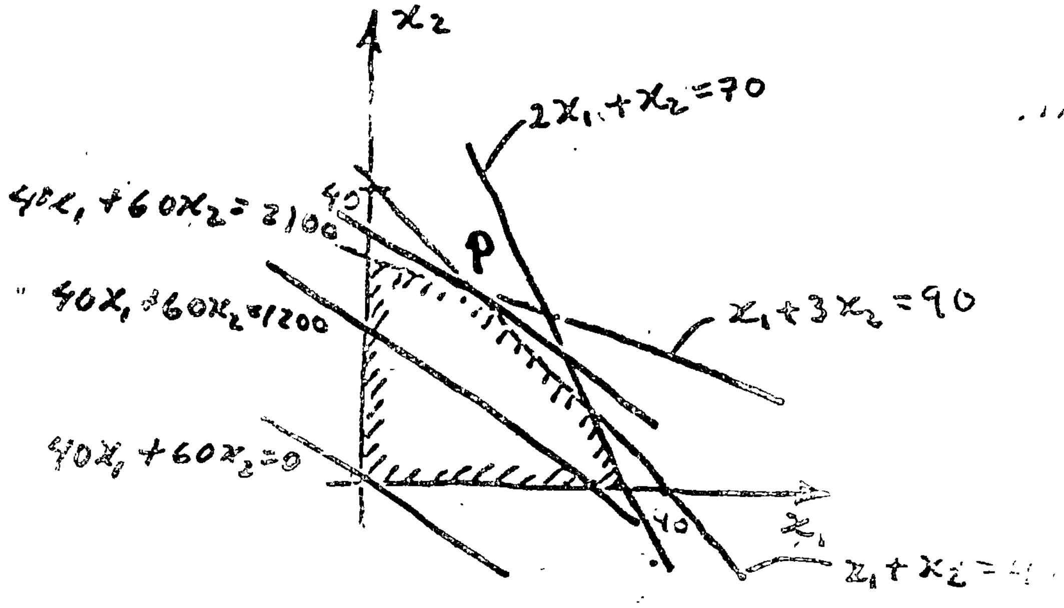
x_1 = # DE PRODUCTOS 1
x_2 = # " " 2

RESTRICCIONES
$$\begin{cases} \text{MAQ. } 1 : & 2x_1 + x_2 \leq 70 \\ \text{'' } 2 : & x_1 + x_2 \leq 40 \\ \text{'' } 3 : & x_1 + 3x_2 \leq 90 \end{cases}$$

$$\left.\begin{array}{l} x_1 \geq 0 \\ x_2 \geq 0 \end{array}\right\} \text{CONDICIONES DE NO NEGATIVIDAD}$$

MAX: GANANCIA $= 40x_1 + 60x_2$

SOLUCIÓN GRÁFICA



: REGIÓN DE SOLUCIONES FACTIBLES

SOL. EN FRONTERA PARA REGION CONVEXA

P: $x_1=15,\ x_2=25$

SOLUCIÓN ANALÍTICA (METODO SIMPLEX)

LADOS POLÍGONO CONV

$$\begin{cases} 2x_1+x_2 \le 70 \\ x_1+x_2 \le 40 \\ x_1+3x_2 \le 90 \end{cases} \longrightarrow \begin{aligned} 2x_1+x_2+x_3 &= 70 \\ x_1+x_2+x_4 &= 40 \\ x_1+3x_2+x_5 &= 90 \end{aligned}$$

$x_i \ge 0 \quad i=1,2,3,4,5$

SIST. 3 ECS 5 INCOGNITAS

MAX. $M=40x_1+60x_2$

1ª SOLUCION FACTIBLE: $x_1=x_2=0$

$$x_1=x_2=0 \Rightarrow \left.\begin{aligned} x_3&=70 \\ x_4&=40 \\ x_5&=90 \end{aligned}\right\} \text{VARIABLES DE LA BASE}$$

$M = 40\times0+60\times0 = 0$

$$M=40x_1+60x_2 \begin{cases} x_1: 0\to1 \Rightarrow M: 0\to40 \\ x_2: 0\to1 \Rightarrow M: 0\to60 \Leftarrow \end{cases}$$

MAXIMO INCREMEN

CON $x_1 = 0$

$$\left.\begin{array}{l} 2x_1 + x_2 + x_3 = 70 \\ x_1 + x_2 + x_4 = 40 \\ x_1 + 3x_2 + x_5 = 90 \end{array}\right\} \longrightarrow \begin{array}{l} x_3 = 70 - x_2 \\ x_4 = 40 - x_2 \\ x_5 = 90 - 3x_2 \end{array}$$

$$x_2 \nearrow \Rightarrow \left\{\begin{array}{l} x_3 \searrow \\ x_4 \searrow \\ x_5 \searrow \end{array}\right.$$

$x_3 = 0 \Rightarrow x_2 = 70$

$x_4 = 0 \Rightarrow x_2 = 40$

$x_5 = 0 \Rightarrow x_2 = \frac{90}{3} = 30 \longleftarrow$ EL MENOR

2ª SOLUCION FACTIBLE

$$\begin{array}{ll} x_1 = 0 & x_2 = 30 \\ x_5 = 0 & x_3 = 40 \\ & x_4 = 10 \end{array} \Big\rangle \text{ VARS. BASE}$$

$$M = 40x_1 + 60x_2 = 0 + 60 \times 30 = 1800 = M$$

$$M = 40x_1 + 60x_2 = 40x_1 + 60\left(30 - \frac{x_5}{3} - \frac{[illegible]}{[illegible]}\right)$$

$M = 1800 + 20x_1 - 20x_5$ < $x_1 \nearrow 1 \Rightarrow$ [illegible] $\longleftarrow$ / $x_5 \nearrow \Rightarrow$ [illegible]

SIST. ECN:

$$\left\{\begin{array}{l} \frac{5}{3}x_1 + x_2 - \frac{1}{3}x_5 = 40 \\ \frac{2}{3}x_1 + x_4 - \frac{1}{3}x_5 = 10 \\ \frac{1}{3}x_1 + x_2 + \frac{1}{3}x_5 = 30 \end{array}\right.$$

USANDO 3ª ECN

VARS. BASE 2ª ITERACIÓN

c.c $x_5 = 0$

$$\left\{\begin{array}{l} \frac{5}{3}x_1 + x_3 - \frac{1}{3}x_5 = 40 \\ \frac{2}{3}x_1 + x_4 - \frac{1}{3}x_5 = 10 \\ \frac{1}{3}x_1 + x_2 + \frac{1}{3}x_5 = 30 \end{array}\right. \quad \left\{\begin{array}{ll} x_3 = 40 - \frac{5}{3}x_1 = 0 & x_4 = 24 \\ x_4 = 10 - \frac{2}{3}x_1 = 0 & x_1 = 15 \longleftarrow \\ x_2 = 30 - \frac{1}{3}x_1 = 0 & x_1 = 90 \end{array}\right.$$

3ª SOLUCIÓN FACTIBLE

$x_4 = 0 \qquad x_1 = 15$

$x_5 = 0 \qquad x_2 = 25$

$\qquad\qquad x_3 = 15$

$M = 1800 + 20x_1 - 20x_5 = 1800 + 20(15 - \frac{3}{2}x_4 + \frac{1}{2}x_5) - 20x_5$

$M = 2100 - 30x_4 - 10x_5$

$M = 2100$

$$M = 2100 - 30x_4 - 10x_5 \begin{cases} x_4 \nearrow 1, & M \searrow 30 \\ x_5 \nearrow 1, & M \searrow 10 \end{cases}$$

∴ YA SE TIENE SOL. FACTIBLE ÓPTIMA.

INTERPRETACIÓN

$x_1 = 15$: 15 PRODUCTOS # 1

$x_2 = 25$: 25 " " 2

MAQ. 1: $2x_1 + x_2 + x_3 = 70$

MAQ. 2: $x_1 + x_2 + x_4 = 40$

MAQ. 3: $x_1 + 3x_2 + x_5 = 90$

$x_3 = 15$ ⇒ HOLGURA DE 15 HORAS MAQ. 1

$\left.\begin{matrix} x_4 = 0 \\ x_5 = 0 \end{matrix}\right\}$ ⇒ NO HAY HOLGURA MAQS. 2 Y 3

MAX. GANANCIA = M = 2100

NOTACIÓN MATRICIAL

$2x_1 + x_2 \le 70$

$x_1 + x_2 \le 40$

$x_1 + 3x_2 \le 90$

$x_1 \ge 0, x_2 \ge 0$

$M = 40x_1 + 60x_2$

$A\underline{x} \le \underline{b}$

$\underline{x} \ge \underline{0}$

MAX: $M = \underline{c}^T \underline{x}$

$A = \begin{pmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 3 \end{pmatrix}, \underline{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \underline{b} = \begin{pmatrix} 70 \\ 40 \\ 90 \end{pmatrix}$

$\underline{c}^T = (40 \quad 60)$

METODO SIMPLEX EN TÉRMINOS DE MATRICES

PROGRAMA PARA COMPUTADORA

1ª ITERACIÓN

BASE

TABLEAUX
(TABLA)

$$
\begin{array}{c|cccccc|l}
 & x_1 & x_2 & x_3 & x_4 & x_5 & b & \\
 & 2 & 1 & 1 & 0 & 0 & 70 & 70 \\
 & 1 & 1 & 0 & 1 & 0 & 40 & 40 \\
 & 1 & \boxed{3} & 0 & 0 & 1 & 90 & 90/3 = 30 \longleftarrow \text{MAS CHICO} \\
\hline
M & -40 & -60 & 0 & 0 & 0 & 0 &
\end{array}
$$

$\uparrow$ MASNEG

PIVOTE

$M - 40x_1 - 60x_2 = 0$

2ª ITERACION

$$
\begin{array}{c|cccccc|l}
 & x_1 & x_2 & x_3 & x_4 & x_5 & b & \\
 & 5/3 & 0 & 1 & 0 & -1/3 & 40 & 40/5/3 = 24 \\
 & \boxed{2/3} & 0 & 0 & 1 & -1/3 & 10 & 10/2/3 = 15 \longleftarrow \\
 & 1/3 & 1 & 0 & 0 & 1/3 & 30 & 30/1/3 = 90 \\
\hline
M & -20 & 0 & 0 & 0 & 20 & 1800 &
\end{array}
$$

$\uparrow$ MAS NEG.

$M - 20x_1 + 20x_5 = 1800$

3ª ITERACIÓN

$$
\begin{array}{cccccc}
x_1 & x_2 & x_3 & x_4 & x_5 & b \\
0 & 0 & 1 & -2.5 & 0.5 & 15 \\
1 & 0 & 0 & 1.5 & -0.5 & 15 \\
0 & 1 & 0 & -0.5 & 0.5 & 25 \\
0 & 0 & 0 & 30 & 10 & 2100
\end{array}
\Rightarrow
\begin{array}{l}
x_3 = 15 \\
x_1 = 15 \\
x_2 = 25
\end{array}
\quad x_4 = x_5 = 0
$$

M

$M + 30x_4 + 10x_5 = 2100$

FORMA STANDARD PROGRAMACION LINEAL

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \le b_1 \\ \vdots \\ a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n \ge b_j \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m \end{array}\right\} \text{m ECUACIONES LINEALES Ó DESIGUALDADES CON n VARIABLES}$$

COEFICIENTES ESTRUCTURALES

RESTRICCIONES

$x_i \ge 0$ CONDICIONES DE NO-NEGATIVIDAD

$$\text{OPT. } z = \underbrace{c_1x_1 + c_2x_2 + \dots + c_nx_n}_{\text{COEFICIENTES DE COSTO}}$$

EJEMPLO

$$\text{MIN: } w = 2x_1 + 4x_2 \qquad \left\{\begin{array}{l} 3x_1 + 2x_2 \le 5 \\ x_1 - 4x_2 \ge 3 \\ 5x_1 + x_2 = 7 \end{array}\right.$$

DESIGUALDADES → IGUALDADES

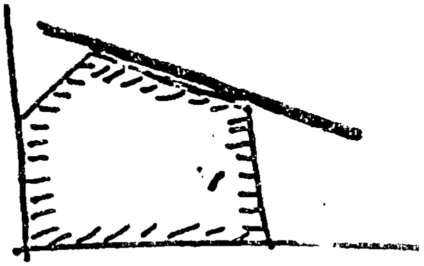
$$\begin{array}{lr|l} & & \text{VARIABLES} \\ 3x_1 + 2x_2 + x_3 & = 5 & + \text{HOLGURA} \\ x_1 - 4x_2 \quad - x_4 + a_1 & = 3 & - \text{HOLGURA} + \text{ARTIFICIAL} \\ 5x_1 + x_2 \quad + a_2 & = 7 & + \text{ARTIFICIAL} \end{array}$$

$$\text{MIN: } w = 2x_1 + 4x_2 \underset{-}{+} 1000a_1 \underset{-}{+} 1000a_2$$

PARA MAXIMIZACIÓN

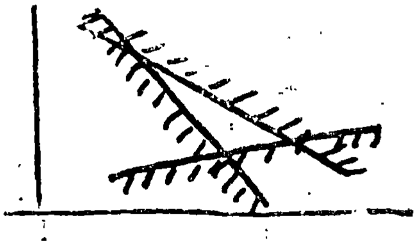
CASOS ESPECIALES

1) SOLUCIONES ÓPTIMAS NO ÚNICAS



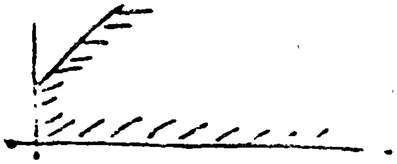
MÉTODO SIMPLEX: $\underbrace{M_i = M_{i+1}}_{2\ \text{ITERACIONES}} = M$ MÁXIMA

2) RESTRICCIONES CONTRADICTORIAS



SIMPLEX: SOL. FACTIBLE CONTIENE VAR. ARTIF.

3) SOLUCIÓN NO ACOTADA



SIMPLEX: M TAN GRANDE COMO SE DESEE

PROBLEMA DUAL

PROBLEMA PRIMO	PROBLEMA DUAL
MAX: $M = \underline{c}^T \underline{x}$	MIN: $V = \underline{b}^T \underline{y}$
$A\underline{x} \leq \underline{b}$	$A^T \underline{y} \geq \underline{c}$
$\underline{x} \geq \underline{0}$	$\underline{y} \geq 0$

AL MENOS 21 UNIDADES VITAMINA A
" 12 " " B

ALIMENTO	VITAMINA/UNIDAD ALIMENTO		COSTO/UNIDAD ALIMENTO
	A	B	
(NARANJA)	1	0	20
2 (MANZANA)	0	1	20
3 (LECHUGA)	1	2	31
4 (CHÍCHARO)	1	1	11
5 (ZANAHORIA)	2	1	12

MIN. COSTO

DIETISTA

x_i = CANTIDAD DE ALIMENTO i

VIT. A: $x_1 + x_3 + x_4 + 2x_5 \geq 21$
VIT. B: $x_2 + 2x_3 + x_4 + x_5 \geq 12$ RESTRICCIONES

$x_i \geq 0$ CONDS. NO NEGATIVIDAD

MIN: COSTO $= 20x_1 + 20x_2 + 31x_3 + 11x_4 + 12x_5$
FUNCIÓN OBJETIVO

DUAL: CIA. FARMACEUTICA "LA CAMPANA"

λ_1 = PRECIO DE CADA PÍLDORA DE VIT. A
λ_2 = " " " " " B

$\lambda_i \geq 0$ CONDS. NO NEGATIVIDAD

MAX. GANANCIA $= 21\lambda_1 + 12\lambda_2$ FUNCIÓN OBJETIVO

PRECIOS COMPETITIVOS:
$$\begin{aligned} 1\lambda_1 &\leq 20 \\ \lambda_2 &\leq 20 \\ \lambda_1 + 2\lambda_2 &\leq 31 \\ \lambda_1 + \lambda_2 &\leq 11 \\ 2\lambda_1 + \lambda_2 &\leq 12 \end{aligned}$$
RESTRICCIONES

DIETISTA

$$\begin{pmatrix}1&0&1&1&2\\0&1&2&1&1\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\\x_4\\x_5\end{pmatrix}\geq\begin{pmatrix}21\\12\end{pmatrix}$$

$$\begin{pmatrix}x_1\\x_2\\x_3\\x_4\\x_5\end{pmatrix}\geq\begin{pmatrix}0\\0\\0\\0\\0\end{pmatrix}$$

MIN: COSTO $= (20 \quad 20 \quad 31 \quad 11 \quad 12)\begin{pmatrix}x_1\\x_2\\x_3\\x_4\\x_5\end{pmatrix}$

"LA CAMPANA"

$$\begin{pmatrix}1&0\\0&1\\1&2\\1&1\\2&1\end{pmatrix}\begin{pmatrix}\lambda_1\\\lambda_2\end{pmatrix}\leq\begin{pmatrix}20\\20\\31\\11\\12\end{pmatrix}$$

$$\begin{pmatrix}\lambda_1\\\lambda_2\end{pmatrix}\geq\begin{pmatrix}0\\0\end{pmatrix}$$

MAX: GANANCIA $= (21 \quad 12)\begin{pmatrix}\lambda_1\\\lambda_2\end{pmatrix}$

EJEMPLO

CONSTRUCCION CASAS DE 2, 3 Y 4 RECÁMARAS
MAX. GANANCIA.
RESTRICCIONES:

1.- PROYECTO ≤ \$9,000,000

2.- # UNIDADES ≥ 350

3.- MAX PORCENTAJE: CASAS 2 REC: 20%
" 3 " : 60%
" 4 " : 40%

4.- COSTOS CONSTRUC.: " 2 " : \$20,000
" 3 " : \$25,000
" 4 " : \$30,000

5.- GANANCIA NETA: " 2 " : \$2,000
" 3 " : \$3,000
" 4 " : \$4,000

5.- GANANCIA: $Z = 2x_1 + 3x_2 + 4x_3$ (EN MILES DE \$)

-4.- $20{,}000x_1 + 25{,}000x_2 + 30{,}000x_3 \leq 9{,}000{,}000$

$20x_1 + 25x_2 + 30x_3 \leq 9{,}000$

5.- $x_1 + x_2 + x_3 \geq 350$

- $x_1 \leq 0.2(x_1 + x_2 + x_3)$

$x_2 \leq 0.6(x_1 + x_2 + x_3)$

$x_3 \leq 0.4(x_1 + x_2 + x_3)$

$20x_1 + 25x_2 + 30x_3 \leq 9{,}000 \rightarrow 20x_1 + 25x_2 + 30x_3 + x_4 = 9{,}000$

$x_1 + x_2 + x_3 \geq 350 \rightarrow x_1 + x_2 + x_3 - x_5 = 350$

$x_1 \leq .2(x_1 + x_2 + x_3) = .2(350 + x_5) \rightarrow x_1 - .2x_5 + x_6 = 70$

$x_2 \leq .6(x_1 + x_2 + x_3) \rightarrow x_2 - .6x_5 + x_7 = 210$

$x_3 \leq .4(x_1 + x_2 + x_3) \rightarrow x_3 - .4x_5 + x_8 = 140$

ECUACIONES EN 8 INCOGNITAS $\Rightarrow$ 56 POSIBLES SOLS.

SIN SIMPLEX: RESOLVER SIST. 5 ECS, 5 INCOGNITAS 56 VECES, PARA DETERMINAR SOL. OPTIMA).

ARA TENER 1a SOLUCION FACTIBLE [illegible]

RESTRICCION: $x_1 + x_2 + x_3 - x_5 + x_9 = 350$ [illegible] ARTIFICIAL

FUN. OBJETIVO: $Z = 2x_1 + 3x_2 + 4x_3 - 1000x_9$ [illegible] COSTO DE

$$
\begin{array}{lrcl}
\text{MAX:} & Z = 2x_1 + 3x_2 + 4x_3 - 1000x_9 & & \\
\text{ST.} & 20x_1 + 25x_2 + 30x_3 + x_4 & & = 9{,}000 \\
 & x_1 + x_2 + x_3 \quad - x_5 & + x_9 & = 350 \\
 & x_1 \quad - .2x_5 + x_6 & & = 70 \\
 & x_2 \quad - .6x_5 + x_7 & & = 210 \\
 & x_3 \quad - .4x_5 + x_8 & & = 140
\end{array}
$$

MINIMIZACIÓN

MIN → MAX

EJEMPLO: EQUIVALENTE

MIN: $y = 3x_1 + 4x_2 + 7x_3$	MAX: $z = -y = -3x_1 - 4x_2 - 7x_3$
$x_1 - 3x_2 + x_3 \geq 7$	$x_1 - 3x_2 + x_3 \geq 7$
$2x_1 + x_2 - x_3 \geq 9$	$2x_1 + x_2 - x_3 \geq 9$
$x_i \geq 0$	$x_i \geq 0$

APLICACIONES

PROBLEMA DE COMBINACIÓN

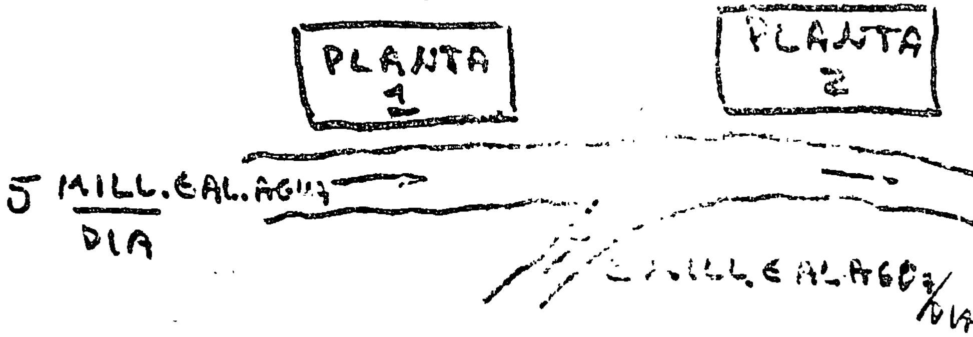
COMPONENTES: SECOMBINAN PARA DAR 1 Ó MAS PRODOCTOS

TIENEN CIERTOS COSTOS Y CARACTERISTICA

? = CANTIDAD DE CADA COMPONENTE SUJETAS A CIERTAS LLMITACIONES

MIN: COSTO TOTAL

EJEMPLO



GENERACIÓN DE UNIDADES DE INTAMINANTE AL DÍA	20	14
COSTO REMOVER 1 UNIDAD CONTAM.	$1000	$800

20% → CONT. SE ELIMINA

CUANDO MÁS: 2 UNID. CONT/MILLON DE GALONES

? = OPERACIÓN MAS BARATA DE TRATAMIENTO CONTAMIN.

$x_{1,2}$ ≡ # DE UNIDADES DE CONTAMINANTE ELIMINADAS POR LAS PLANTAS 1,2

MIN: COSTO = $\$1000 x_1 + \$800 x_2$

PLANTA 1:

$$\overset{\text{SE GENERAN}}{\underbrace{(20 - x_1)}_{\text{UNIDADES DE CONTAMINANTE}}} \overset{\text{SE ELIMINAN}}{\le} \left(\frac{5 \text{ MILL. GAL. AGUA}}{\text{DIA}}\right)\left(\frac{2 \text{ UNID. CONT.}}{\text{MILL. GAL.}}\right)$$

PLANTA 2:

$$\underbrace{.8(20-x_1)}_{\text{UNID. CONT. LLEGAN DE PLANTA 1}} + \underbrace{(14-x_2)}_{\text{UN. CONT. SEGENERAN}} \le (5+2)\frac{\text{MILL.G.}}{\text{DIA}}\left(\frac{2 \text{ U.C.}}{\text{M.G.}}\right)$$

$x_1 \le 20$ $\qquad$ $x_i \ge 0$

$x_2 \le 14$

MIN. COSTO = $1000 x_1 + 800 x_2$

RESTRICCIONES: $x_1 \ge 10$

$.8x_1 + x_2 \ge 16$

$x_1 \le 20$

$x_2 \le 14$

CONDS. NO NEGAT. $x_1 \ge 0$

$x_2 \ge 0$

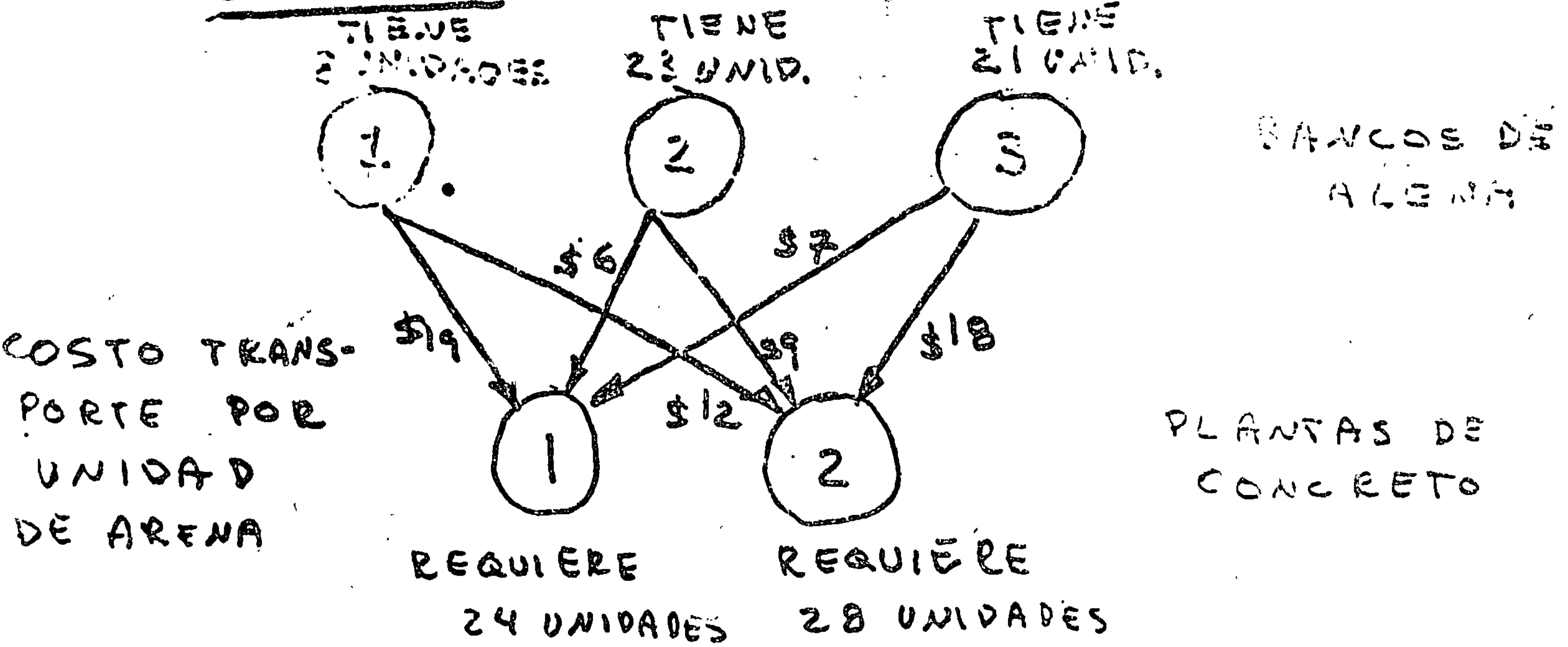
PROBLEMAS DE TRANSPORTE

UN PRODUCTO - SE TRANSPORTA DE m CENTROS DE PRODUCCIÓN EN CANTIDADES $a_1, \ldots, a_m$

- SE RECIBE EN n CENTROS EN CANT. $b_1, \ldots, b_n$
- SE CONOCEN COSTOS TRANSP. CENTRO i DESTINO j
- ? = CANT. QUE SE TRANSP CENTRO i A DESTINO j

[illegible]

EJEMPLO



?= CUÁNTA ARENA TRANSPORTAR DE CADA BANCO A CADA PLANTA

MIN: COSTOS DE TRANSPORTE.

---

x_{ij} = NUM. UNIDADES DE ARENA DE BANCO i A PLANTA j.

BANCO DE ARENA 1: $x_{11} + x_{12} \leq 8$
" " " 2: $x_{21} + x_{22} \leq 23$
" " " 3: $x_{31} + x_{32} \leq 21$

PLANTA DE CONCRETO 1: $x_{11} + x_{21} + x_{31} = 24$
" " " 2: $x_{12} + x_{22} + x_{32} = 28$

$$x_{ij} \geq 0$$

MIN: COSTOS DE TRANSPORTE =

$$19x_{11} + 12x_{12} + 6x_{21} + 9x_{22} + 7x_{31} + 18x_{32}$$

PROBLEMAS DE TRANSPORTE

TIPO PROG. LINEAL; SE SIMPLIFICAN POR LA ESTRUCTURA PARTICULAR DEL PROBLEMA)

m FABRICAS QUE PRODUCEN CANTIDADES $a_1, \ldots, a_m$ DE UN PRODUCTO

n CENTROS DE CONSUMO QUE CONSUMEN $b_1, \ldots, b_n$

c_{ij} : COSTO DE TRANSPORTAR PRODUCTO DE FABRICA i A CENTRO j.

$z = x_{ij}$ = NUM. DE PRODS MANDADOS DE FABRICA i A CENTRO j.

LO QUE SE FABRICA SE CONSUME

EJEMPLO

		CENTROS CONSUMO 1	2	INVENTARIOS
	1	15	20	15
FABRICAS	2	18	22	25
	3	25	19	20
DEMANDA		35	25	+ 60

COSTOS TRANSPORTE

OFERTA = DEMANDA

MIN COSTO TRANSPORTE

PLAN DE TRANSPORTE = ?

x_{ij} = NUM. UNIDADES DE FABRICA i A CENTRO j

$$\text{MIN}: z = 15x_{11} + 18x_{21} + 25x_{31} + 20x_{12} + 22x_{22} + 19x_{32}$$

$$\left.\begin{array}{l} x_{11} + x_{12} = 15 \\ x_{21} + x_{22} = 25 \\ x_{31} + x_{32} = 20 \end{array}\right\} \text{PRODUCCIÓN (INVENTARIOS)}$$

$$\left.\begin{array}{l} x_{11} + x_{21} + x_{31} = 35 \\ x_{12} + x_{22} + x_{32} = 25 \end{array}\right\} \text{DEMANDA CENTRO CONSUMO}$$

$$x_{ij} \geq 0 \quad \forall i, \forall j$$

COMO OFERTA = DEMANDA, SÓLO 4 DE LAS 5 ECUACIONES SON LINEALMENTE INDEPENDIENTES; SE PUEDE ELIMINAR 1 ECUACIÓN.

POR METODO SIMPLEX, HABRÍA QUE INTRODUCIR 4 VARS. ARTIFICIALES PARA FORMAR UNA SOL INICIAL FACTIBLE.

SE PUEDE APLICAR UNA SIMPLIFICACIÓN DEL MÉTODO SIMPLEX PORQUE:

1.- LOS COEFICIENTES DE TODAS LAS VARS. = 1

2.- CUALQUIER VARIABLE x_{ij} APARECE UNA SOLA VEZ EN LAS PRIMERAS 3 ECUACIONES
UNA " " " " ÚLTIMAS 2

$$\text{MATRIZ DE COSTOS} = \begin{pmatrix} 15 & 20 \\ 18 & 22 \\ 25 & 19 \end{pmatrix}$$

$$\text{MATRIZ DE DISTRIBUCIÓN} = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{pmatrix} = ?$$

QUITANDO ULTIMA ECUACIÓN:

$$\left.\begin{aligned} x_{11} + x_{12} &= 15 \\ x_{21} + x_{22} &= 25 \\ x_{31} + x_{32} &= 20 \\ x_{11} + x_{21} + x_{31} &= 35 \end{aligned}\right\}$$ CONJUNTO DE 4 ECUACIONES LINEALMENTE INDEPENDIENTES CON 6 INCÓGNITAS

UNA SOL. FACTIBLE: DOS VARIABLES IGUALES A CERO, SIN CREAR INCONSISTENCIA

POR EJEMPLO: $x_{12} = 0$, $x_{31} = 0$

$$\begin{cases} x_{11} = 15 \\ x_{21} + x_{22} = 25 \\ x_{32} = 20 \\ x_{11} + x_{21} = 35 \end{cases} \qquad \begin{cases} x_{11} = 15 \\ x_{11} + x_{21} = 35 \\ x_{21} + x_{22} = 25 \end{cases}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{21} \\ x_{22} \\ x_{32} \end{pmatrix} = \begin{pmatrix} 15 \\ 35 \\ 25 \\ 20 \end{pmatrix}$$

MATRIZ TRIANGULAR => SIST. SE RESUELVE FÁCILMENTE

$x_{11} = 15$ $\quad x_{12} = 0$

$x_{21} = 20$ $\quad x_{22} = 5$

$x_{31} = 0$ $\quad x_{32} = 20$

SOLUCIÓN FACTIBLE, NO NECESARIAMENTE ÓPTIMA.

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

APLICACIONES DE LA COMPUTALORA A LA SIMULACION Y OPTIMIZACION

PROGRAMACION ENTERA

M. EN C. L.P. GRIJALVA LOPEZ

MARZO-ABRIL, 1978.

Solve the game and show that the solutions to the linear-programming problem and its dual correspond to the optimal probabilities of the game players.

5.20 Given below is an input-output matrix.

(a) Set the problem as a linear program

(b) Solve it (use a computer).

		OUTPUTS (PRODUCTS)								
		COM. 1[a]	COM. 2	COM. 3	COM. 4	COM. 5	COM. 6	COM. 7	COM. 8	TOTAL SUPPLY
INPUTS	FACTOR 1	1	1	0	0	0	2	2	0	150
	FACTOR 2	0	0	0	0	1	0	1	1	75
	FACTOR 3	4	2	1	2	0	2	2	3	325
	FACTOR 4	0	0	0	0	1	1	2	0	100
	FACTOR 5	0	0	0	0	0	0	0	2	50
	FACTOR 6	2	3	5	3	1	1	2	0	425
	FACTOR 7	2	1	3	3	0	3	1	0	275
	FACTOR 8	0	1	0	0	0	0	1	3	125
	COM. 1	0	0.2	0	0.1	0	0.1	0	0.1	
	COM. 2	0	0	0.1	0.2	0	0.2	0	0.2	
	COM. 3	0.1	0	0	0	0.2	0	0	0	
	COM. 4	0	0.1	0.1	0	0.1	0.1	0	0.1	
	COM. 5	0.2	0	0	0	0	0	0.3	0	
	COM. 6	0	0.2	0.1	0	0	0	0.1	0	
	COM 7	0	0	0.1	0.2	0.2	0	0	0	
	COM. 8	0.2	0	0.1	0	0	0.1	0.1	0	
	PROFIT PER UNIT	80	95	110	115	120	125	140	200	

[a] Com. = Commodity

5.21 Explain why, in a transportation problem, a solution with $m+n-1$ occupied cells is a basic feasible solution.

INTEGER PROGRAMMING 6

6.1 INTRODUCTION

Linear-programming models assume divisibility. In other words, any non-negative continuous values can be assigned to the solution variables. However, in many practical cases, this assumption is unrealistic. For example, an optimal solution calling for scheduling 2.3 machines does not have an operational meaning. We must schedule two or three machines—not 2.3. Similarly, in shipbuilding one cannot proceed to build 7.5 ships. The *assignment problem* represents another example in which divisibility is not appropriate. (However, the assignment algorithm always yields an integer solution.) All of these examples suggest the need for imposing an additional constraint, namely that some, or all, of the solution variables must be restricted to integer values.[1] The resulting model, which is called *integer (linear) programming*, consists of (1) a linear objective function, (2) a set of linear constraints, (3) a set of non-negativity constraints, and (4) integer-value constraints for one or more variables.

When all the *variables* of the optimal program are required to be *integers*, we have an *all-integer* problem. If only some of the variables must be integers we have a *mixed-integer* problem.

[1] An integer number is a whole number as distinguished from a fraction.

Adding the integer requirement creates more constraints. This means that the optimal integer solution will always be equal to or less favorable than the optimal non-integer solution. In other words, the decision maker usually pays a price for imposing the indivisibility requirements.

Integer programming is extremely important not only because it allows us to solve practical problems with indivisibility requirements, but also because it can be used as an auxiliary tool in the solution of several complicated problems that cannot otherwise be solved. For example, many nonlinear, nonconvex, combinatorial, and discrete problems can be reduced to integer linear-programming form.

As in the case of nonlinear programming we encounter difficulties in the solution of medium- and large-size problems (for example, more than 100 constraints and 100 variables to be integerized). Some of the difficulties arise mainly in the process of verifying the optimal solution (optimality test).[2]

Various methods are available for solving the integer-programming problem.[3] In this chapter we shall discuss the following:

1. Rounding off a noninteger solution.
2. Complete enumeration.
3. Graphical approach.
4. Gomory's all-integer method.
5. Land and Doig's method.
6. Branch-and-bound approach.
7. Heuristic programming.

In addition, we shall discuss, very briefly, discrete programming, the dual-integer problem, and the nonlinear integer problem. We shall also illustrate several important applications of integer programming.

6.2 ROUNDING THE NONINTEGER SOLUTION

A practical approach to an integer-programming problem, in some cases, is to solve it as a regular linear-programming problem and then round off the optimal results. The major advantage of such an approach is economy of the time and cost that would have been required for formulating and solving the special integer-programming model, since the integer requirements usually result in additional iterations. The major disadvantage of the rounding approach is that we may arrive at a solution that may be different from the optimal integer solution, and possibly infeasible.

In order to illustrate this point we summarize in Table 6.1 the "rounded" and the actual integer solutions to two different problems In the first case we want to maximize an objective function of $3x_1 + 2x_2$, subject to the constraints $10x_1 + 5x_2 \leq 100$ and $20x_1 + 30x_2 \leq 300$, and in the second we want to maximize an objective function of $10{,}000x_3 + 20{,}000x_4$, subject to the constraints $x_3 \leq 4.5$, $x_4 \leq 3.5$, and $x_3 + x_4 \leq 7$. In each case we present for comparison the noninteger result, a rounded result, and the optimal integer solution..

Table 6.1 Comparison of integer and noninteger solutions

	OBJECTIVE FUNCTION	NONINTEGER		ONE WAY OF ROUNDING		OPTIMAL INTEGER	
		x_j	$F(x)$	x_j	$F(x)$	x_j	$F(x)$
CASE I	$3x_1 + 2x_2$	$x_1 = 7.5$ $x_2 = 5$	$32.5	$x_1 = 7$ $x_2 = 5$	$31	$x_1 = 8$ $x_2 = 4$	$32
CASE II	$10{,}000x_3 +$ $20{,}000x_4$	$x_3 = 3.5$ $x_4 = 3.5$	$105,000	$x_3 = 3$ $x_4 = 3$	$90,000	$x_3 = 4$ $x_4 = 3$	$100,000

It is quite evident that in the first case the rounding did not lead us to the true-integer optimal solution, but it did lead us to a rounded solution that is only $1 away from the optimal integer solution. In the second case, however, the difference between the optimal integer solution and the rounded solution is substantial ($10,000).

Another variant of the rounding method is the *trial-and-error* approach, in which one must enumerate selected integer solutions in the neighborhood of the noninteger solution. In Table 6.1 for example, one can check (in case I) the value of the objective function given by the pairs (7,5) and (8,4), and then select the pair with the highest value of the objective function. One word of caution: Rounding might result in violation of one or more constraints. Therefore, *when rounding, one must check all constraints that include the rounded variables* against possible violation.

An interesting approach to the rounding method is a systematic "rounding algorithm." (See Gomory's work [23] involving the use of dynamic programming.)

6.3 COMPLETE ENUMERATION

Theoretically, any all-integer program can be solved by complete enumeration. It is possible to assign all possible integer values to all variables and check all possible feasible solution combinations (if the feasible region is bounded) to determine that combination which yields the l[illegible]st value (in maximization) of the objective function within the limits of the constraints.

[2] For example, an extreme point can be locally optimal among neighboring "all-integer" points and still not be globally optimal.

[3] For a survey s [illegible] linski [6, 7].

In cases where the number of variables and the possible combinations is small, this method might be efficient. However, in most practical problems we find an astronomical number of combinations, and the method is therefore impractical.

There is one special case in which complete enumeration might be used with advantage. For example, several business and economic decision problems can be so formulated that the value 1 designates a "yes" choice, and the value 0 designates a "no" choice; thus, the variables are restricted to the values of either 1 or 0. For such cases an implicit enumeration search has been developed by Balas [3]. This method has also been used in a problem of allocating funds to independent research and development projects (see Peterson [43]), and has been found to be very efficient with as many as 50 variables. Enumeration efforts, in general, may be reduced with the *branch-and-bound* approach (see Section 6.7).

6.4 GRAPHICAL METHOD

6.4.1 GENERAL

All $2 \times n$ integer-programming problems can be solved by the graphical method. Problems of the dimension $3 \times n$ can also be solved graphically, but the solution is not as easy to obtain. The major advantages of the graphical method are its simplicity and its applicability for solving both the all-integer and the mixed-integer problems.

The graphical approach to all-integer and mixed-integer problems is similar to the graphical approach for solving regular linear-programming problems. The difference lies in the nature of the feasible solution spaces for the two problems. Whereas in the regular case we construct the convex set of feasible solutions bounded by the linear constraints, in integer programming we obtain a collection of lattice, all-integer points.[4]

6.4.2 AN ILLUSTRATIVE EXAMPLE

a. The Problem

The ABC Company is a large manufacturer of household appliances. Recently its board of directors approved a $12.5 million budget for constructing additional plants and/or warehouses. The construction of each warehouse will cost $1 million, and the management does not want more than eight warehouses. The construction of each plant will cost $2 million, and the management does not plan to construct more than five plants. It is estimated that each warehouse will contribute $31,000/month to the company's profit and each plant will contribute $60,000/month. The problem is to determine the optimal number of plants and warehouses.

[4] Points where all coordinates are given by integer numbers.

Since we cannot build fractions of plants or warehouses, our problem is clearly an integer-programming one.

The problem can be mathematically stated as follows:[5]

$$\max z = 31x_1 + 60x_2$$

s/t

$$\begin{aligned} x_1 + 2x_2 &\leq 12.5 \\ x_1 \quad\quad &\leq 8 \\ x_2 &\leq 5 \end{aligned}$$

and x_1 and x_2 must take non-negative integer values (x_1 = number of warehouses and x_2 = number of plants).

b. Graphical Noninteger Solution

Using the method suggested in Chapter 3, we have solved the problem graphically, as shown in Figure 6.1. Point C ($2\frac{1}{4}$ plants and 8 warehouses)

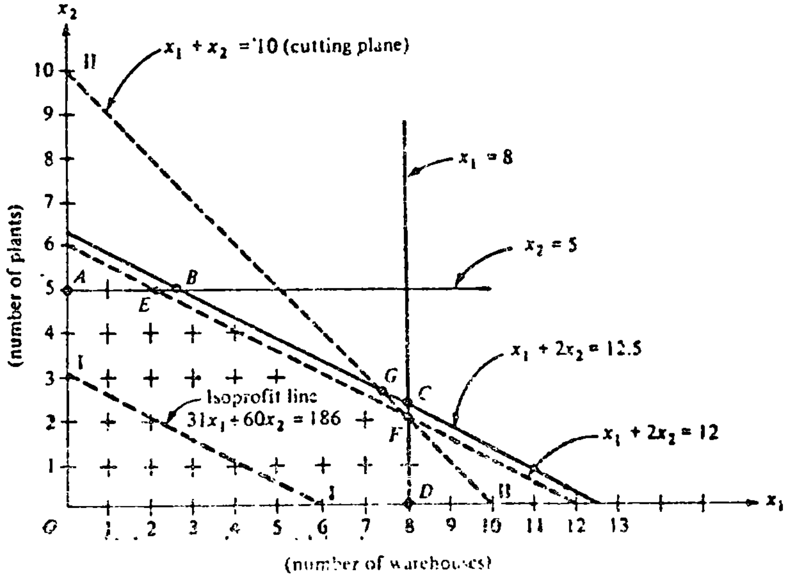


FIGURE 6.1

[5] Note that we have scaled the objective function as well as the first constraint.

represents the optimal program. Such a program will result in a profit of $31,000 × 8 + $60,000 × 2¼ = $383,000/month.

c. Graphical Integer Solution

Our first task, as in the noninteger case, is to determine the feasible solution set. In this simple example, this can easily be accomplished by marking all possible integer solution combinations (lattice points) with a "+" sign (Figure 6.1). If we construct a convex set with a minimum area covering all the integer-solution set, we get the area (*OAEFD*), which is a convex set smaller than *OABCD*. To solve this problem we draw an arbitrary profit line I-I, derived from $31x_1+60x_2=186$, and then we draw profit lines parallel to the line I-I until the optimal solution, at point *F* (8, 2), is obtained. The expected profit in this case is 8 × $31,000 + 2 × $60,000 = $368,000.

d. A New Constraint

The difference between the noninteger feasible area (*OABCD*) and the integer feasible area (*OAEFD*) is the replacement of the constraint $x_1+2x_2 \leq 12.5$ by a new constraint (line *EF*) $x_1+2x_2 \leq 12$. The major problem in integer programming is to find a constraint of this type that eliminates noninteger corners, such as *B* and *C*, from consideration.

6.4.3 COST OF INDIVISIBILITY

The integer solution indicated a monthly profit of $368,000 as compared to $383,000 for the noninteger solution. The value of the objective function is thus reduced by $15,000/month. This difference is called the *cost of indivisibility*. In our example the cost of indivisibility is actually smaller than $15,000/month. This is because the integer solution, under the assumption of the problem, leaves $500,000 idle funds (we use only $12 million out of the $12.5 million available). This money can be invested in, say, the bond market, and the yield can be deducted from the $15,000 figure just calculated. If we assume that the bond investment yields 4.8 per cent a year, and we invest $500,000 in bonds we will net $2000 each month. Thus the actual cost of indivisibility in this case is $15,000 − $2000 = $13,000 per month. In other words, the indivisibility requirement has forced us to *divert some resources*[6] from the most profitable projects to less profitable projects. Thus the actual cost of indivisibility is the difference between utilizing the resources in the most profitable outlet provided by the problem at hand and utilizing them in the most profitable alternative.

[6] In some cases the characteristics of the unutilized resources are such that they cannot be diverted to alternative uses.

6.5 GOMORY'S METHOD—ALL-INTEGER CASE

6.5.1 CONGRUENCY

Before discussing Gomory's method [21] it is helpful to present the mathematical notion of congruency used in this method.

Definition

Two numbers are said to be congruent if, and only if, their difference is a positive or a negative integer. The sign ≡ denotes congruency.

Examples:

(a) $4.5 \equiv 1.5$; (because $4.5 - 1.5 = 3$, an integer)
(b) $-2.75 \equiv 3.25$; $(-2.75 - 3.25 = -6)$
(c) $3 \equiv 5$; $(3 - 5 = -2)$
(d) $2.625 \equiv 0.625$; $(2.625 - 0.625 = 2)$
(e) $-0.7 \equiv 0.3$; $(-0.7 - 0.3 = -1)$

The fractional part f_x of a real number x is defined to be the *smallest nonnegative number congruent* with x. In other words, f_x is the *smallest fractional part* that one can subtract from a noninteger number in order to convert it into an integer number.

Examples:

(a) Given $x = 5.3$, then $f_x = 0.3$; $(5.3 - 0.3 = 5)$
(b) Given $x = -2.25$, then $f_x = 0.75$; $(-2.25 - 0.75 = -3)$
(c) Given $x = -6$, then $f_x = 0$
(d) Given $x = 0.33$, then $f_x = 0.33$

Properties of Congruent Numbers

If we have two numbers x and y such that $x \equiv y$, then

(1) $f_x \equiv f_y$
(2) $f_{x+y} \equiv f_x + f_y$
(3) $x + c \equiv y + c$, for all c
(4) $-x \equiv -y$

Also, if k is an integer, then

$$kx \equiv f_{kx} \equiv kf_x$$

6.5.2 GOMORY'S BASIC IDEA

Gomory's major idea was to construct a convex area covering all lattice points. He accomplished this by constructing "cutting planes" with the aid of additional constraints imposed on the problem. These "cutting planes," which are introduced *one at a time*, reduce the original feasible area to the desired integer configuration. Gomory's constraints have the following properties.

1. They usually cut a convex area out of the previous feasible area.
2. The cutting plane goes through *at least* one lattice point (not necessarily a feasible one).
3. Each cut approaches the *smallest area* that is required to cover all feasible lattice points.

The method insures us an optimal solution in a finite number of iterations.[7]

The Gomory method is described below:

First we solve the problem without paying any attention to the integer constraints. Then we examine the optimal solution. If each variable is an integer, the problem is solved. Otherwise, we construct a Gomorian constraint and impose it on the original problem. This constraint is our "cutting plane." The addition of the Gomorian constraint turns the optimal and feasible (noninteger) solution into an optimal but infeasible (noninteger) solution. Hence, the next step is to employ the dual-simplex method to arrive at a new optimal and feasible solution. If this is an all-integer solution, the problem is solved. Otherwise, we keep on adding Gomorian constraints, one at a time, until an optimal integer solution is obtained. These steps are summarized in Figure 6.2.

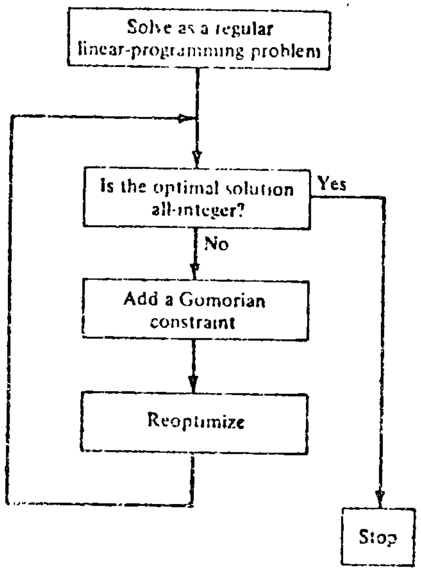

Solve as a regular linear-programming problem
Is the optimal solution all-integer?
Yes
No
Add a Gomorian constraint
Reoptimize
Stop

FIGURE 6.2

[7] This may not be true when computation is executed by a computer, because of "rounding" errors.

6.5.3 FORMULATION

The following is a condensed presentation of the mathematical formulation of Gomory's method.

If we examine the optimal solution to a linear-programming problem we can isolate a given row, and write this row in the form of an equation such that

$$x_i = q_i - \sum_{j \neq i} \hat{a}_{ij} x_j \tag{6.1}$$

where

x_i are *basis* variables
q_i is the value of variable i in the solution
$\hat{a}_{ij}$ are the substitution ratios in the optimal tableau
x_j are the nonbasis variables

If all q_i are integers, we have an all-integer solution and the problem is solved. If not all q_i are integers, we add a Gomorian constraint. The addition of the new constraint follows these steps:

1. We divide all noninteger q_i values into an integer and a fractional part; that is, $q_i = k_i + f_i$ (where k_i is an integer and f_i is a non-negative fractional part of q_i).
2. Divide the substitution ratios in a similar way:

$$\hat{a}_{ij} = k_{ij} + f_{ij}$$

Then, we substitute these new values into (6.1) and get

$$x_i = k_i + f_i - \sum_{j=m+1}^{m+n} (k_{ij} + f_{ij}) x_j$$

$$= \underset{\text{integer}}{(k_i - \Sigma k_{ij} x_j)} + (f_i - \Sigma f_{ij} x_j) \tag{6.2}$$

In order for x_i to have an integer value the expression $(f_i - \sum f_{ij} x_j)$ in Equation (6.2) must be either zero or a negative integer;[8] (that is,

$$f_i - \sum f_{ij} x_j \leq 0 \qquad \text{or } f_i \leq f_{ij} x_j \tag{6.3}$$

$$f_i - \sum f_{ij} x_j + s_i = 0 \tag{6.4}$$

where $s_i \geq 0$ is a new slack known as a Gomorian slack; it has a zero price coefficient).

Equation (6.4) is the equation of the cutting plane.

[8] First of all, $x_j \geq 0$ because of the non-negativity requirement; and x_i by definition are integers. Second, if x_i's are to be integers, $(f_i - \Sigma f_{ij} x_j)$ must be an integer since $(k_i - \Sigma k_{ij} x_j)$ is an integer. However, if $(f_i - \Sigma f_{ij} x_j)$ is to be an integer, it cannot be a positive integer because $0 \leq f_i < 1$, and the only way to have x_i an integer is to have $\Sigma f_{ij} x_j$ large enough to make $(f_i - \Sigma f_{ij} x_j)$ either zero or a negative integer.

6.5.4 AN ILLUSTRATIVE EXAMPLE

In order to illustrate Gomory's method we shall solve the same investment problem that was solved graphically in Section 6.4 2. We can formally write the linear integer-programming problem as

$$\max z = 31x_1 + 60x_2$$

s/t

$$\begin{aligned} x_1 + 2x_2 &\le 12.5 \\ x_1 &\le 8 \\ x_2 &\le 5 \end{aligned}$$

and

$$x_1, x_2 \text{ integer} \ge 0$$

Solution, Step I: Solve the Problem by the Regular Simplex Method

The optimal solution (disregarding the integer requirements) is shown in Table 6.2. The optimal solution calls for $x_1 = 8$, $x_2 = 2.25$, and $s_3 = 2.75$. The value of the objective function is $383,000. The value of s_3 indicates the unused capacity of the third constraint.

The solution is not "all-integer," and therefore we proceed to the next step.

Table 6.2 Fourth and optimal tableau (noninteger)

PROGRAM	PROFIT	QUANTITY	x_1	x_2	s_1	s_2	s_3
x_1	31	8	1	0	0	1	0
s_3	0	2.75	0	0	− 0.5	0.5	1
x_2	60	2.25	0	1	0.5	−0.5	0
NET EVALUATION		3.83	0	0	+30	+1	0

Solution, Step II: Select the "Key Row"

In this step we examine all the noninteger entries under the quantity column, divide them into integer and fractional parts, and designate the row with the largest fractional part as the key row. In our case,

	Integer Part	Fractional Part
From second row: $q_3 = 2.75 = 2 + 0.75$	2	0.75
From third row: $q_2 = 2.25 = 2 + 0.25$	2	0.25

Gomory suggested the following rule of thumb[9] for selecting the key row:

Select the row that has the largest fractional part of any *real* variable[10] in "quantity" column. In this case, if we follow Gomory's rule, we select third row as the key row (x_2 being the only real noninteger variable).

Solution, Step III: Write the Key Row in an Equation Form

We proceed now to write the key row of step II in the form of Equation (6.

This equation can easily be derived from the third row of the optin tableau (Table 6.2). The relevant information is shown in Table 6.3, t results of which can be stated as

Table 6.3

	PROGRAM	PROFIT	QUANTITY	x_1	x_2	s_1	s_2	s_3
THIRD ROW	x_2	60	2.25	0	1	0.5	−0.5	0
(6.1) FORM	$x_2 =$		2.25	$-(0x_1$	$0x_2$	$+0.5s_1$	$-0.5s_2$	$+0s$

$$\boxed{x_2 = 2.25 - 0.5s_1 + 0.5s_2}$$

Solution, Step IV: Build the Gomorian Constraint

We build the Gomorian constraint according to inequality (6.3). Recall th in order to obtain an integer solution the following condition must hold:

$$f_i - \sum f_{ij}x_j \le 0 \quad \text{or} \quad f_i \le \sum f_{ij}x_j \qquad (6.3$$

An examination of Table 6.3 indicates that

$f_i = f_2 = 1/4$, because $x_2 = 2 + 1/4$ and the fractional part of 1/4 is 1/4.
$f_{23} = 1/2$, because the fractional part of 1/2 is 1/2. ($\bar{a}_{23} = +0.5$.)
$f_{24} = 1/2$, because the fractional part of − 1/2 is 1/2. ($\bar{a}_{24} = -0.5$.)

Note: The two nonbasic variables s_1 and s_2 in Table 6.3 are given th subscripts 3 and 4, respectively, while using 6.3. We will make similar adjustments throughout Chapter 6.

Hence, according to (6.3),

$$\frac{1}{4} \le \frac{1}{2}s_1 + \frac{1}{2}s_2$$

[9] This rul humb does not guarantee the most efficient computation. However, it is simple and it .. better than making a random choice.

[10] A real variable in this case is any of the original variables Slac artificial variables are auxiliary variables and by this definition are not real. However, it is sometimes also possible to integerize such auxiliary variables.

If we add a new slack variable[11] s_4, we get the Gomory cutting plane as an equation:[12]

$$\frac{1}{4}+s_4=\frac{1}{2}s_1+\frac{1}{2}s_2 \quad \text{or} \quad s_4=-\frac{1}{4}-\left(-\frac{1}{2}s_1-\frac{1}{2}s_2\right) \qquad (6.5)$$

Note: We can express this new constraint in terms of the original variables x_1 and x_2. For this purpose we reproduce the original constraints

$$x_1+2x_2+s_1=12.5 \qquad (1)$$

$$x_1 \qquad +s_2=\ 8 \qquad (2)$$

and, from Equation (6.5),

$$s_1+s_2-2s_4=\ 0.5 \qquad (3)$$

Substituting relations (1) and (2) into (3), we get

$$12.5-x_1-2x_2+8-x_1-2s_4=0.5$$

or

$$x_1+x_2+s_4=10$$

or

$$x_1+x_2\leq 10 \qquad (6.6)$$

This is the cutting plane in its *explicit form*. It expresses the Gomory constraint in terms of the real variables only. The reader is referred now to Figure 6.1, from which we can see that

1. The cutting plane $x_1+x_2\leq 10$, (line II-II), goes through point *F*.
2. It bars point *C* (noninteger) from the feasible area.
3. It allows *all* integer lattice points (marked with +) to remain in the feasible area.
4. It reduces the original feasible area.

Solution, Step V: Revise the Noninteger Program

We can take one of two approaches to revise the noninteger program. One approach is to take the *explicit* form of Gomory's constraint, Equation (6.6), add it to the *original* constraints, and re-solve the problem with the enlarged set of constraints. The other and more efficient approach (to be demonstrated here) is to add the *implicit form* of the constraint, given in Equation (6.5), to the optimal solution (Table 6.2) as the last row and change it to Table 6.4 and then reoptimize the modified problem.

We start by adding the new constraint to the optimal solution. However, in doing so, we introduce a negative sign into the quantity column (see Table 6.4, line of s_4) This means that the solution becomes nonfeasible; this is because the solution is equivalent to point *C* (Fig. 6.1), which is now outside the feasible area. Our new solution (Table 6.4) is optimal but not feasible. We can now solve the problem by the dual-simplex method. The dual-simplex computations will not be given here. The reader can verify that the optimal integer solution to this problem is:

$$x_1=8, \qquad x_2=2$$

Table 6.4 Gomory's constraint added to optimal solution (noninteger)

PROGRAM	PROFIT	QUANTITY	x_1	x_2	s_1	s_2	s_3	s_4	
x_1	31	8	1	0	0	1	0	0	
s_3	0	2.75	0	0	−0.5	0.5	1	0	
x_2	60	2.25	0	1	0.5	−0.5	0	0	
s_4	0	−0.25	0	0	−0.5	−0.5	0	1	Added Gomory constraint
NET EVALUATION z_j-c_j			0	0	+30	+1	0	0	

Let us make another observation before we leave this example. If we take the first approach (using the explicit form) we develop an interesting situation that is typical of the Gomorian method. We get a *noninteger* optimal solution (Table 6.5) that *cannot be integerized* because all substitution ratios are integers (that is, their fractional part=0), so an additional Gomorian constraint cannot be added. This solution is seen in Figure 6.1 (point *G*). For this reason, Gomory's "all-integer" algorithm assumes that all variables, including the slack variables must be integer. This can be accomplished if all coefficients and constants in the original problem are made integers. In this example we should have multiplied the first constraint by two before we started our computation.

Table 6.5 Optimal solution (noninteger)

PROGRAM	PROFIT	QUANTITY	x_1	x_2	s_1	s_2	s_3	s_4
x_1	31	7.5	1	0	−1	0	0	2
s_2	0	0.5	0	0	1	1	0	−2
x_2	60	2.5	0	1	1	0	0	−1
s_3	0	2.5	0	0	−1	0	1	1
NET EVALUATION			0	0	29	0	0	2

[11] The new slack variables (required by Gomory constraints) will be identified in the tableaux by separating them by double vertical lines.

[12] This is the *implicit form* of the Gomory constraint. It is arranged in the form: $s_i=-f_i-(-\Sigma f_{ij}x_j)$ so that it can be inserted directly into the optimal tableau.

6.5.5. SOME SELECTED EXAMPLES FOR BUILDING GOMORY'S CONSTRAINTS

Example 1:

An optimal solution is shown in Table 6.6. Both x_2 and x_3 are nonintegers. Following Gomory's rule of thumb, we select x_3 to be integerized first (because 2/3 > 1/4).

Table 6.6

PROGRAM	COST	QUANTITY	x_1	x_2	x_3	s_1	s_2	s_3	s_4 (NEW)	
x_2	22	$10\frac{1}{4}$	0	1	0	1	0	0	0	Optimal tableau
x_3	10	$6\frac{2}{3}$	−1/4	0	1	−2/3	1/3	0	0	
s_3	0	4	1/2	0	0	3/4	−1/2	1	0	
s_4	0	−2/3	−3/4	0	0	−1/3	−1/3	0	1	Added Gomory constraint

In order to derive equation (6.4) for this problem, we note from Table 6.6:

$$f_3=\frac{2}{3},\qquad f_{31}=\frac{3}{4},\qquad f_{34}=\frac{1}{3},\qquad f_{35}=\frac{1}{3}$$

Putting these values in Equation (6.3),

$$\frac{2}{3}\leq\frac{3}{4}x_1+\frac{1}{3}s_1+\frac{1}{3}s_2$$

$$\frac{2}{3}+s_4=\frac{3}{4}x_1+\frac{1}{3}s_1+\frac{1}{3}s_2$$

or

$$s_4=-\frac{2}{3}-\left(-\frac{3}{4}x_1-\frac{1}{3}s_1-\frac{1}{3}s_2\right)$$

The Gomorian constraint equation just derived is now added as the last row in Table 6.6.

Example 2:

We write in Table 6.7 only that row of the optimal solution which is to be integerized. We note from the table that

$$f_5=\frac{5}{8}\quad f_{52}=\frac{2}{5}\quad f_{53}=\frac{1}{4}\quad f_{56}=\frac{1}{2}\quad f_{57}=\frac{3}{4}\quad f_{58}=\frac{2}{3}\quad f_{5,10}=0$$

Putting these values in (6.4) form and rearranging terms,

$$s_6=-\frac{5}{8}-\left(-\frac{2}{5}x_2-\frac{1}{4}x_3-\frac{1}{2}s_1-\frac{3}{4}s_2-\frac{2}{3}s_3\right)$$

This Gomorian constraint[13] is now added as row s_6 in Table 6.7.

Table 6.7

	q	x_1	x_2	x_3	x_4	x_5	s_1	s_2	s_3	s_4	s_5	s_6 (NEW)	
x_5	$6\frac{5}{8}$	0	−3/5	1/4	0	1	−1/2	7/4	−7/3	0	1	0	Row to be integerized
s_6	−5/8	0	−2/5	−1/4	0	0	−1/2	−3/4	−2/3	0	0	1	Gomory constraint

Gomory's method has sometimes been known to run to several hundred (or even thousand) iterations without converging to the optimal solution on even relatively small problems. Another major disadvantage of this method is that it does not yield any integer feasible solution until it terminates with the optimal solution. For these reasons Gomory's method in the form presented has more of a theoretical interest than a computational value, although it can be efficiently coded into a computer routine. For more efficient computer codes see Section 6.13.

6.6 MIXED-INTEGER PROGRAMMING

6.6.1 INTRODUCTION

When some, but not all, of the variables are required to be integers, the problem is labeled as a mixed-integer problem. The value of the objective function in the optimal solution of a mixed-integer maximization problem is always larger than or equal to the optimal functional value for the same problem under the all-integer constraint and always smaller than or equal to the optimal functional value for the same problem without integer constraints. The opposite is true for a minimization problem. The reason is that each integer requirement has a non-negative (zero or positive) indivisibility (opportunity) cost.

[13] The same constraint can be derived by first utilizing Equation (6.3), in which case $f(x)$ will designate the fractional part of x. For Table 6.7, Equation (6.3) will be written as

$$f\left(\frac{5}{8}\right)-\left[f\left(-\frac{3}{5}\right)x_2+f\left(\frac{1}{4}\right)x_3+f\left(-\frac{1}{2}\right)s_1+f\left(\frac{7}{4}\right)s_2+f\left(-\frac{7}{3}\right)s_3+f(\ldots)s_5\right]\leq 0$$

Several computational methods were developed to treat the mixed-integer problem.[14] Problems with two or three variables (or constraints) can be solved graphically similarly to the all-integer problems (see Section 6.4). Gomory [22] has extended his method for the all-integer case to cover the solution for the mixed-integer case. In Section 6.6.2 we shall briefly review the method developed by Land and Doig [33].

6.6.2 THE BASIC IDEA OF LAND AND DOIG'S METHOD

Land and Doig's method is based on a systematic search for an optimum solution. It can be applied both to the mixed-integer and the all-integer case. As in the Gomorian approach, the starting point in Land and Doig's method is an optimal regular simplex solution. If this solution violates one or more of the integer requirements, one of two approaches can be used. In one approach we consider one variable at a time and use parametric programming to determine the range of feasible integer values for variables to be integerized. Within this range, all integer lattice points are checked with regard to their impact on the objective function. Once the optimal integer lattice point within the range of one variable has been found, we proceed to a second variable, and so on until the last variable required to be integer is integerized.

The alternative computational method is based on the solution of several simple linear-programming subproblems that are created when additional Integer constraints are added, one at a time, to the initial program. The second approach is simpler than the first, although it does involve more computations. Examples of both approaches are given by Land and Doig [33] and by Balinski [6]. The original form of the method seems to be very inefficient in large problems handled by computers, although it is fairly efficient in manual calculations of small problems. The modified method is a base for several computer codes (see Section 6.13).

6.7 BRANCH-AND-BOUND METHOD

The branch-and-bound approach developed by Little *et al.* [36] is an iterative technique for an intelligent partial enumerative search (see Lawler and Wood [34] and Mitten [39]). It can be used to solve both all-integer and mixed-integer problems. The idea is to solve the problem first without paying attention to the integer requirement and then if the solution violates the integer requirement, to employ the branching.

The approach is to split the problem into two parts by aiming the search at two integer values for the variable that is noninteger. The integer values to be searched are those integers that are immediately next to the noninteger value. Assume, for example, that a variable x_2 in the solution equals 2.25. Then, we split the problem into two parts by introducing two new constraints, $x_2 \geq 3$ and $x_2 \leq 2$, one in each branch. The following example illustrates this branch-and-bound approach.

We reproduce the problem that has already been solved graphically:

$$\max z = 31x_1 + 60x_2$$

s/t

$$\begin{aligned} x_1 + 2x_2 &\leq 12.5 \\ x_1 \quad &\leq 8 \\ x_2 &\leq 5 \end{aligned}$$

The optimal noninteger solution is

$$x_1 = 8$$
$$x_2 = 2.25$$

We now create two new problems:

Problem A	Problem B
$\max z = 31x_1 + 60x_2$	$\max z = 31x_1 + 60x_2$
s/t	s/t
$x_1 + 2x_2 \leq 12.5$	$x_1 + 2x_2 \leq 12.5$
$x_1 \leq 8$	$x_1 \leq 8$
$x_2 \leq 5$	$x_2 \leq 5$ (redundant)
$x_2 \geq 3$	$x_2 \leq 2$

Since in our optimal solution $x_2 = 2.25$ is infeasible (because it is noninteger), the integer feasible solution must be *either* in the region $x_2 \geq 3$ *or* in the region $x_2 \leq 2$. We solve the two new problems, the optimal solution being:

For problem A: $x_1 = 6.5 \quad x_2 = 3 \quad z = 381.5$
For problem B: $x_1 = 8 \quad x_2 = 2 \quad z = 368$

We stop the search in problem B since it has an all-integer solution. Problem A is searched further since the value of its objective function is larger than $z = 368$. It is possible that the optimal integer solution to A could yield a $z > 368$.

We branch the solution $x_1 = 6.5$ and $x_2 = 3$ by splitting it into two subproblems, one with $x_1 \leq 6$ and the other with $x_1 \geq 7$. Both problems result in a z less than 368. Hence the optimal solution to our problem is:

$$x_1 = 8 \quad x_2 = 2 \quad z = 368$$

[14] A computational difficulty stems from the fact that a mixed-integer problem *is not* a special case of an all-integer problem and it requires a special algorithm.

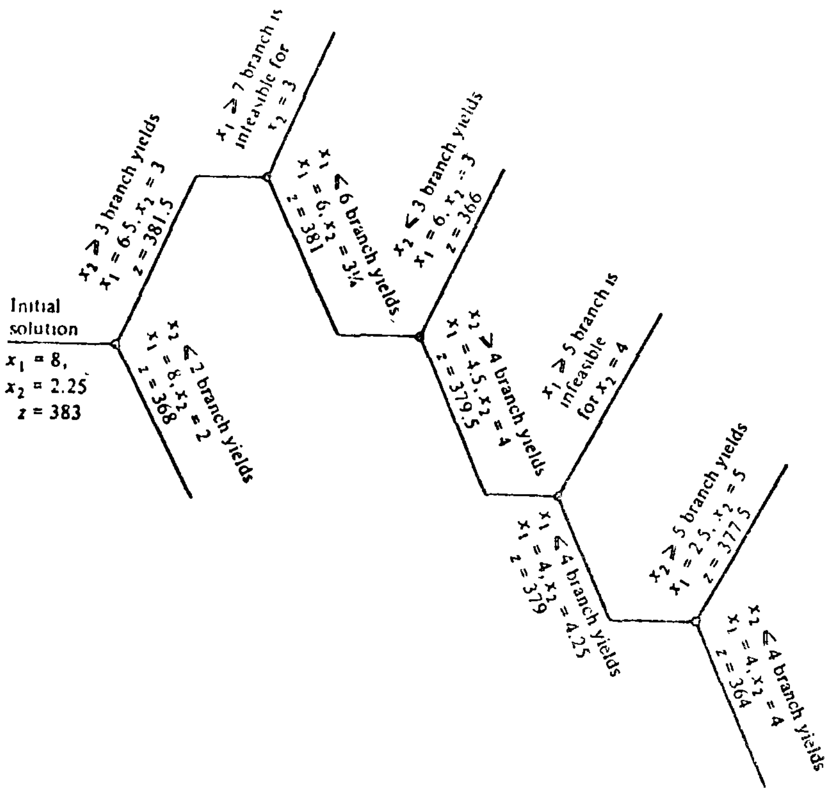

Initial solution
$x_1 = 8$,
$x_2 = 2.25$
$z = 383$
$x_2 \geqslant 3$ branch yields
$x_1 = 6.5, x_2 = 3$
$z = 381.5$
$x_2 \leqslant 2$ branch yields
$x_1 = 8, x_2 = 2$
$z = 368$
$x_1 \geqslant 7$ branch is infeasible for $x_2 = 3$
$x_1 \leqslant 6$ branch yields,
$x_1 = 6, x_2 = 3\frac{1}{4}$
$z = 381$
$x_2 \leqslant 3$ branch yields
$x_1 = 6, x_2 = 3$
$z = 366$
$x_2 \geqslant 4$ branch yields
$x_1 = 4.5, x_2 = 4$
$z = 379.5$
$x_1 \geqslant 5$ branch is infeasible for $x_2 = 4$
$x_1 \leqslant 4$ branch yields
$x_1 = 4, x_2 = 4.25$
$z = 379$
$x_2 \geqslant 5$ branch yields
$x_1 = 2.5, x_2 = 5$
$z = 377.5$
$x_2 \leqslant 4$ branch yields
$x_1 = 4, x_2 = 4$
$z = 364$

FIGURE 6.3

In the search process, branching is stopped when (1) we have no more branches that can be further partitioned, or (2) a solution results in a z value less than an *upper bound*. In our solution, the first upper bound was established with a z value of 368.[15]

When the search is terminated, we declare as optimal the solution with the highest (in maximization case) value of the objective function. The entire branching is shown in Figure 6.3.

The branch-and-bound approach can be efficiently coded into a computer routine; it works well in problems where only a few variables are required to be integers. However in problems requiring a large number of variables to be integers, and in cases where the noninteger solution is far from the optimal, the number of iterations may be too large for a practical application. The algorithm is being used, combined with other methods (such as Land and Doig's) as a basis for improved computer codes (see Section 6.13).

[15] If we find an integer solution whose z is more than 368, and the problem can be further partitioned, then that new solution becomes the modified *upper bound*. Any branch whose value of the objective function is *less than* the upper bound should not be considered any furthe[illegible]e search process.

6.8 HEURISTIC PROGRAMMING

The determination of optimal solutions to some complex integer-programming and combinatorial-type problems could involve a prohibitive amount of time and cost. In such situations it is possible to arrive at "good enough" solutions by using a set of heuristics (that is, rules of thumb) that produce an economy of search, Heuristic programs are employed to yield an acceptably high value (as opposed to the optimal) of the objective function, and are usually executed by a computer. Heuristic programming has worked well in a number of practical applications.[16] The interested reader is referred to Wiest [53].

6.9 DUALITY, SHADOW PRICES, AND ECONOMIC INTERPRETATIONS

6.9.1 GENERAL

As in the case of a regular linear-programming problem, each integer-programming problem has a corresponding dual problem. Generally speaking, the dual integer-programming problem has an integer solution with values that can be interpreted as implicit (or shadow) prices. In this section we shall derive the dual to an all-integer problem, and present the economic interpretation of and the relationship between the dual solution and the cost of indivisibility.

6.9.2 AN ILLUSTRATIVE EXAMPLE

Given an all-integer problem:

$$\max z = 6x_1 - 2x_2 + 10x_3 + x_4$$

s/t

$$\begin{aligned} x_2 + 2x_3 \quad &\leq 5 \\ 3x_1 - x_2 + x_3 + x_4 &\leq 10 \\ x_1 + \quad x_3 + x_4 &\leq 8 \end{aligned}$$

The optimal (noninteger) solution is given in Table 6.8. Since the solution is not integer, we shall try to integerize variable x_3 by adding a Gomorian constraint (in an implicit form):

[16] Some of the integer problems that have been solved by heuristic programming are the traveling-salesman problem (see Chapter 5) assembly-line balancing, job scheduling, facilities location, and delivery problems. With sufficient ingenuity one can devise an integer-programming representation of almost any combinatorial optimizatio[illegible]blem (see, for example, the machine-sequencing problem in Section 6 10.4).

$$\frac{1}{2}x_2+\frac{1}{2}s_1-s_4=\frac{1}{2}$$

or

$$s_4=-\frac{1}{2}-\left(-\frac{1}{2}x_2-\frac{1}{2}s_1\right)$$

If we add this as an additional row to Table 6.8 we get an infeasible solution and, with the aid of the dual-simplex method, we obtain, in one iteration, an optimal all-integer solution (see Table 6.9).

Table 6.8

PROGRAM	PROFIT	QUANTITY	x_1	x_2	x_3	x_4	s_1	s_2	s_3
x_3	10	5/2	0	1/2	1	0	1/2	0	0
x_1	6	5/2	1	−1/2	0	1/3	−1/6	1/3	0
s_3	0	3	0	0	0	2/3	−1/3	−1/3	1
NET EVALUATION		40	0	4	0	1	4	2	0

Table 6.9 Optimal all-integer solution

PROGRAM	PROFIT	Q	x_1	x_2	x_3	x_4	s_1	s_2	s_3	s_4
x_3	10	2	0	0	1	0	0	0	0	1
x_1	6	3	1	0	0	1/3	1/2	1/3	0	−1
s_3	0	3	0	0	0	2/3	−1/3	−1/3	1	0
x_2	−2	1	0	1	0	0	1	0	0	−2
NET EVALUATION		36	0	0	0	1	0	2	0	8

Now, writing the Gomorian constraint in its explicit form, we get $x_3 \leq 2$. The explicit form is obtained by substituting $s_1=5-x_2-2x_3$ in the implicit form of the Gomory constraint We add this constraint to the original maximization problem and write the *dual* to the modified problem:

$$\min z=5u_1+10u_2+8u_3+2u_4$$

s/t

$$\begin{aligned}
3u_2+u_3 &\geq 6\\
u_1-u_2 &\geq -2\\
2u_1+u_2+u_3+u_4 &\geq 10\\
u_2+u_3 &\geq 1
\end{aligned}$$

The optimal solution to the dual problem is shown in Table 6.10.

Table 6.10 Optimal solution of the dual

PROGRAM	COST	Q	u_1	u_2	u_3	u_4	s_1	s_2	s_3	s_4
s_4	0	1	0	0	−2/3	0	−1/3	0	0	1
u_1	5	0	1	0	1/3	0	−1/3	−1	0	0
u_4	2	8	0	0	0	1	1	2	−1	0
u_2	10	2	0	1	1/3	0	−1/3	0	0	0
z_j-c_j		36	0	0	−3	0	−3	−1	−2	0

6.9.3 ECONOMIC INTERPRETATION

All the characteristics of ordinary dual prices, including price condition (Section 3 4.7), apply here as well. An additional property is that dual price are integer in the optimal program. It is interesting to compare the results o the all-integer and noninteger optimal programs (Table 6.11).

Table 6.11

	PRIMAL SOLUTION	DUAL SOLUTION	TOTAL PROFIT
NONINTEGER	$x_1=x_3=2.5$	$u_1=4,\ u_2=2$	40
INTEGER	$x_1=3, x_2=1, x_3=2$	$u_2=2, u_4=8$	36

In the noninteger solution we had an optimal program that utilized, in full, both the first and the second constraints. For this reason we have nonzero dual prices. ($u_1>0$, $u_2>0$) that correspond to these two constraints. In the optimal integer solution we move to a new solution point, and according to the price conditions, u_1 and u_3 *must be zero* because we do not utilize fully the first and the third constraints (see Table 6.9). In other words, we have *free goods* whose shadow prices are zero.

Our solution in Table 6.11 calls for $u_4=8$. This shadow price corresponds to the Gomorian constraint $x_3+s_4=2$. This constraint is fully utilized in the optimal solution, and therefore u_4 is greater than zero. This value can be interpreted as a measure of the *cost of indivisibility* (opportunity cost). If we compare the noninteger and the integer solutions, we will note that the integer solution reduces x_3 by 1/2 unit (from $2\frac{1}{2}$ to 2). This reduction resulted in a loss of 4 in our objective function (from 40 to 36), which is measured by $\Delta x_3 u_4=1/2\times 8=4$.

For a detailed discussion of the dual prices and their relationship to the marginal yields of scarce indivisible resources, and their efficient allocation, see Gomory and Baumol [24].

6.10 INTEGER PROGRAMMING AS AN AUXILIARY TOOL

6.10.1 GENERAL

A host of combinatorial problems that at the present time are not amenable to analytical solution procedures can be converted into integer-programming problems. In such cases integer programming, and especially mixed-integer programming, can be used either to solve the problems completely or to derive one or more approximate solutions. Thus, the *potential application* of integer programming as an auxiliary tool is indeed impressive. In this section we shall present some interesting uses of integer programming as an auxiliary tool.[17]

6.10.2 BOOLEAN VARIABLES

Boolean variables, by definition, take a binary form; in other words, they can assume a value either of 0 or of 1. These variables may be denoted by d_i. They are used in several forms of problems, as we shall see next.

6.10.3 MUTUALLY EXCLUSIVE CONSTRAINTS (AND SETS OF CONSTRAINTS)

Mutually exclusive constraints (or sets of constraints) can frequently be found in practical cases. For example, consider the optimization problem of an electric power generation and distribution system. The system is to be designed under the assumption that only one of two alternative modes of power generation (nuclear or fossil fuel) can be used. Associated with each mode is a set of one or more constraints that must be honored if, and only if, that particular mode of power generation is being used. This is obviously an either-or type of problem. In cases such as just mentioned, the variables are restrained by either one constraint (or a set) or by another, but not by both. Constraints of this type are called *mutually exclusive constraints*, *dichotomous constraints*, or *"either-or" constraints.*[18] Integer programming offers an elegant way to solve problems involving mutually exclusive constraints.[19]

[17] The basic ideas for these applications were developed by Dantzig [15].

[18] We have already mentioned that the more the number of active constraints, the lower the value of the objective function of a programming problem. The value of the objective function in an either-or type problem, therefore, will be equal to or greater than the case when all constraints must hold simultaneously (in a maximization problem), unless the constraints create no feasible area for solution.

[19] It is possible to solve the either-or problem by solving it once with one constraint (or set of constraints) and once with the second constraint, then comparing the results and selecting the better solution. In many cases this approach may be more efficient than the elegant integer-programming solution.

We shall now illustrate an "either-or" type problem by considering the example discussed in Chapter 3. This problem involved a planting decision faced by a farmer. We solved the problem graphically and obtained an optimal program of 2400 eggplants and 800 tomato plants, with a profit of $10,000. Our farmer was constrained by upper limits on both labor and land. Let us reproduce the problem:

$$\max z = 3x_1 + 3.5x_2$$

s/t

$$\begin{aligned} x_1 + \ 2x_2 &\leq 4000 \\ 4x_1 + \ 3x_2 &\leq 12{,}000 \end{aligned}$$

Let us now assume that either the labor or the land constraint must be faced by the farmer—but not both simultaneously. In other words, our new problem can be formulated as:

$$\max z = 3x_1 + 3.5x_2$$

s/t

$$\begin{aligned} &\textit{either} \quad & x_1 + \ 2x_2 &\leq 4000 \\ &\textit{or} \quad & 4x_1 + \ 3x_2 &\leq 12{,}000 \end{aligned}$$

A glance at Figure 6.4 indicates that instead of the constrained area *OAPD* in the case of the two constraints holding simultaneously, we have either the constrained area *OBD* or the constrained area *OAC*. They give rise to a nonconvex region *OCPB*, and therefore the simplex method cannot

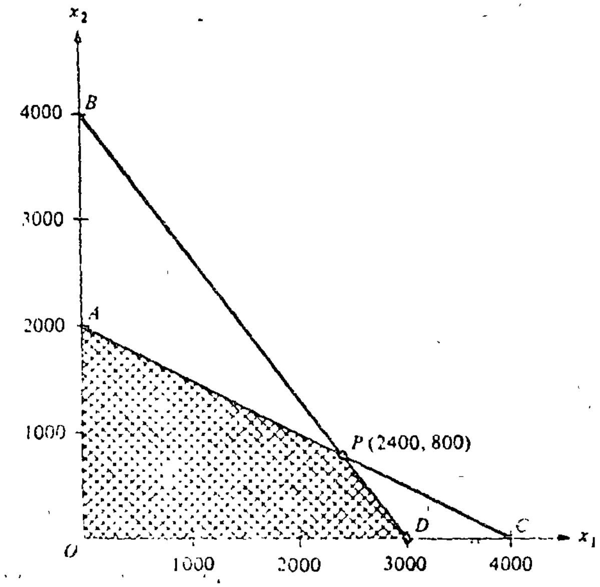


FIGURE 6.4

be applied directly in solving this problem. The solution is derived below (a) by comparing all possible combinations of the subproblems (the enumeration approach) and (b) by the use of integer programming.

Solution, Method A: Enumeration Approach

The problem can be separated into two subproblems:

$$\begin{aligned}
&(1) \quad \max z = 3x_1 + 3.5x_2 \\
&\qquad \text{s/t} \\
&\qquad\qquad x_1 + 2x_2 \le 4000 \\
&(2) \quad \max z = 3x_1 + 3.5x_2 \\
&\qquad \text{s/t} \\
&\qquad\qquad 4x_1 + 3x_2 \le 12{,}000
\end{aligned}$$

The solution to the first problem is $x_1 = 4000$, with a profit of \$12,000: the solution to the second problem is $x_2 = 4000$, with a profit of \$14,000. It is obvious that the solution for the either-or problem is $x_2 = 4000$, $x_1 = 0$, and $z = \$14{,}000$. The enumeration approach is efficient in solving small problems involving a small number of alternative (either-or) constraints.

Solution, Method B: Integer-Programming Approach

In this approach, we modify the constraints of the problem before the application of integer programming. In order to illustrate this modification, we will again utilize our farmer's problem.

Let us consider the addition of a very large number M to the right-hand side of the two constraints.[20]

The first constraint becomes $x_1 + 2x_2 \le 4000 + M$ and the second constraint becomes $4x_1 + 3x_2 \le 12{,}000 + M$. The number M should be sufficiently large to allow x_1 and x_2 to take their highest feasible values ($x_1 = 4000$ and $x_2 = 4000$ in our case). For example, if we set $M = 100{,}000$ we get, in the first constraint,

$$4000 + 8000 < 104{,}000 \qquad \text{(true)}$$

and similarly, in the second constraint,

$$16{,}000 + 12{,}000 < 112{,}000 \qquad \text{(true)}$$

The reason for making M sufficiently large is to avoid eliminating any feasible solutions from consideration. The result of adding M to any constraint is that, even in the extreme case, we will not fully utilize that constraint.

Let us further modify the constraints by utilizing the Boolean variable d (see Table 6.12). An examination of the modified constraints in Table 6.12 shows that if $d = 0$, our first constraint returns to its original form, and the second modified constraint becomes

$$4x_1 + 3x_2 \le 12{,}000 + M$$

In this form the second constraint is not binding and, hence, can be eliminated. Thus, only the first constraint will hold.

Table 6.12

ORIGINAL CONSTRAINTS	MODIFIED CONSTRAINTS
Either $x_1 + 2x_2 \le 4000$	$x_1 + 2x_2 \le 4000 + dM = 4000 + 100{,}000d$
or $4x_1 + 3x_2 \le 12{,}000$	$4x_1 + 3x_2 \le 12{,}000 + (1-d)M = 12{,}000 +$ $(1-d)100{,}000$

On the other hand, if $d = 1$, the second constraint holds and the first constraint is not binding and can be eliminated.

We have thus shown the equivalence of the original and the modified constraints.

The original either-or problem can now be presented as[21]

$$\max z = 3x_1 + 3.5x_2 + 0d$$

s/t

$$\begin{aligned}
&(1) \quad x_1 + 2x_2 - 100{,}000d \le 4000 \\
&(2) \quad 4x_1 + 3x_2 - (1-d)100{,}000 \le 12{,}000 \quad \text{or} \\
&\qquad 4x_1 + 3x_2 + 100{,}000d \le 112{,}000 \qquad 0 \le d \le 1 \text{ and integer}
\end{aligned}$$

This is a mixed-integer problem with three variables and three constraints. The initial solution is given in Table 6.13.

The optimal solution is

$$x_1 = 0 \quad x_2 = 4000 \quad d = 1 \quad z = \$14{,}000$$

The either-or problem, as expected, gave a higher value than the regular problem (\$14,000 vs. \$10,000)

Table 6.13

PROGRAM	COST	Q	x_1	x_2	d	s_1	s_2	s_3
s_1	0	4000	1	2	−100,000	1	0	0
s_2	0	112,000	4	3	100,000	0	1	0
s_3	0	1	0	0	1	0	0	1
NET EVALUATION $z_j - c_j$			−3	−3.5	0	0	0	0

[20] It is possible to select a specific M_i for each constraint i, where M_i is the upper bound on constraint i. Here, for simplicity, we use M as the upper bound for the entire set of constraints.

[21] [illegible]

We can use a similar approach for handling any pair of either-or constraints. For an interesting application in capital budgeting problems see Weingartner [52].

We shall now consider three more classes of mutually exclusive problems.

N Mutually Exclusive Constraints

Suppose that we have *N mutually exclusive constraints:*

$$g_i(x_1, x_2, \dots, x_n) \le b_i \qquad i = 1, 2, \dots, N$$

and either $g_1, g_2, g_3, \dots,$ or g_N is binding. The equivalent integer-programming formulation is

(1) $g_i(x_1, x_2, \dots, x_n) \le b_i + d_i M$

(2) $\sum_{i=1}^{N} d_i = N - 1$

(3) $0 \le d_i \le 1$, and d_i is an integer, for $i = 1, 2, \dots, N$

It is obvious from (2) and (3) that all except one of the *d_i's* must equal 1: that is, only one of the constraints is effective.

This case can be extended to cover a case involving mutually exclusive sets of constraints.

Let us illustrate a simple example of three mutually exclusive constraints: either

(1) $\qquad 2x_1 + 3x_2 - x_3 \le 4$

or

(2) $\qquad x_1 - 2x_2 \le 6$

or

(3) $\qquad x_2 \le 1$

The equivalent integer-programming set is:

$$\begin{aligned} 2x_1 + 3x_2 - x_3 &\le 4 + d_1 M \\ x_1 - 2x_2 \quad &\le 6 + d_2 M \\ x_2 \quad &\le 1 + d_3 M \end{aligned}$$

$$d_1 + d_2 + d_3 = 3 - 1 = 2$$

$0 \le d_1, d_2, d_3 \le 1$, and d_i is an integer.

N Constraints, of Which *k* Must Hold

Given below is the generalized equivalent integer program for the case involving *N* constraints, of which *k* must hold:

$$g_i(x_1, x_2, \dots, x_n) \le b_i + d_i M$$

$$\sum_{i=1}^{N} d_i = N - k$$

and $0 \le d_i \le 1$, and d_i is an integer, for $i = 1, 2, \dots, N$.

Let us illustrate this case by stating that any two of the three constrain considered in the problem under the preceding class must hold. In oth words, either (1) and (2), or (1) and (3), or (2) and (3) must hold (here, $N=$ and $k = 2$).

The equivalent integer-programming formulation is:

$$\begin{aligned} 2x_1 + 3x_2 - x_3 &\le 4 + d_1 M \\ x_1 - 2x_2 \quad &\le 6 + d_2 M \\ x_2 \quad &\le 1 + d_3 M \end{aligned}$$

$$d_1 + d_2 + d_3 = 3 - 2 = 1$$

$0 \le d_i \le 1$, and is an integer.

Note that since $d_1 + d_2 + d_3 = 1$ and each d_i may take either the value of or 1, our solution will show that one $d_i = 1$ and the other two d_i's $= 0$.

Some Additional Combinations

First, let us assume that we have two constraints and it is required that a *least* one of them will hold. Then we add to each constraint the usual Md_i an in addition we add $d_1 + d_2 \le 1$.

Example:

$$\begin{aligned} x_1 + 2x_2 &\le 4 + Md_1 \\ 2x_1 + 3x_2 &\le 17 + Md_2 \end{aligned}$$

$0 \le d_i \le 1$, and integer.

(a) The first constraint will be active only if $d_1 = 0$ and $d_2 = 1$.
(b) The second constraint will be active only if $d_1 = 1$ and $d_2 = 0$.
(c) Both constraints will be active only if $d_1 = 0$ and $d_2 = 0$.

The last three conditions can be expressed by adding the additional constraint

$$d_1 + d_2 \le 1$$

(*Note:* In practice we will rarely find such a situation. However, we chose it to demonstrate the power of integer programming.)

Second, let us assume that a given variable x_1 can take only one of a given set of integral values. If x_1 can be either 2 or 3, we can use Boolean variables d_1 and d_2 and add these constraints:

$$x_1 = 2d_1 + 3d_2$$
$$0 \le d_i \le 1, \text{ and integer}$$
$$d_1 + d_2 = 1$$

A simpler way to state the same requirements is to add a constraint: $0 \le 3 - x_1 \le 1$ and x_1 is integer. Another simple way:

$$\begin{matrix} 2 \le x_1 \le 3 & \text{or} & x_1 = 3 - d_1 \\ x_1 \text{ integer} & & d_1 \text{ Boolean} \end{matrix}$$

6.10.4 MULTISTAGE MACHINE-SEQUENCING PROBLEM

Job shop manufacturing operations usually involve a number of batch-type jobs that must be processed through several machines in a certain sequence. An example is a machine shop where parts are sheared, then drilled or punched, then bent, and finally welded together. Usually, one machine can process only one job at a time. The problem is to find a sequence (or schedule) to feed the jobs into various available machines, and minimize the overall processing cost (or time).

The major difficulty in solving this type of problem is the enormous number of possible solutions. The theoretical number of all possible sequences or combinations is $(n!)^m$ (where m is the number of machines and n the number of jobs). In the simple case of five jobs and five machines, for example, we have about 25 billion possible combinations! Traditionally this type of problem has been solved by trial and error or with scheduling charts such as the *Gantt chart*. In some very simple problems this approach might, by chance, hit the optimal solution or closely approximate it. But in most cases the solution will most probably be far from the optimal solution.

Integer programming offers an analytical method for arriving at the optimal solution to such problems. It should be noted that although integer programming guarantees an optimal solution, the calculations (given the present state of the art) are rather lengthy. Also, a sensitivity analysis requires a completely new set of calculations of the optimal solution for each assumed change in the given data. Hence, the integer programming approach can be costly and impractical.

Two-Job Two-Machine Case[22]

We now proceed to illustrate the multistage machine-sequencing problem by considering a simple example involving two jobs and two machines.

A small machine shop has one shear and one punch press. Two jobs are to be processed through the shop. It takes 4 hours to process job 1, and 7 hours to process job 2 on the shear. Job 1 requires 12 hours of punching, whereas job 2 requires 10 hours. Because of large set-up costs we want to run each job until it is completed. Our problem is to determine the optimal schedule—that is, the order in which to process the jobs in the shop so that the overall processing time (calendar time) is minimized. Since we have only two jobs and two machines, we have only $(2!)^2 = 4$ possible combinations. By enumeration we can easily find that the optimal solution is to process job 1 on the shear, then job 2 on the shear while at the same time performing the punching operation on job 1 (Figure 6.5). The total cycle is 26 hours.

[22] The two machine sequencing problems have been solved analytically by Johnson [29] in a more efficient way than integer programming. However, Johnson's method is good only for two-machine scheduling, whereas integer programming can be applied to any number of machines.

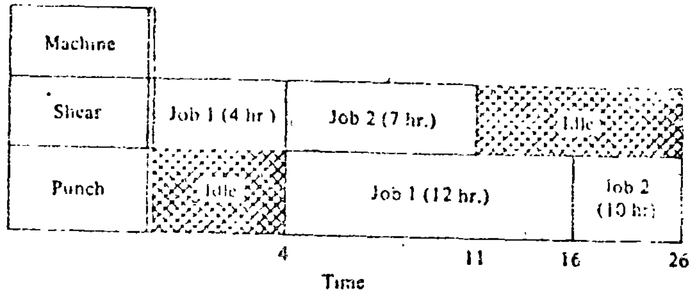


FIGURE 6.5

Integer-Programming Formulation

Let

x_{11} = starting time (calendar time) of job 1 on the shear
x_{21} = starting time (calendar time) of job 2 on the shear
x_{12} = starting time (calendar time) of job 1 on the punch press
x_{22} = starting time (calendar time) of job 2 on the punch press
k_{11} = process time of job 1 on the shear = 4 hours
k_{12} = process time of job 1 on the punch press = 12 hours
k_{21} = process time of job 2 on the shear = 7 hours
k_{22} = process time of job 2 on the punch press = 10 hours

and let x_t be the total elapsed (cycle) time. The problem thus is to minimize x_t subject to the following constraints:

(a) No job can enter a station before it has left the previous operation (for example, punching cannot be started before shearing has been completed). This constraint can be expressed as:

(1) For job 1: $x_{12} \geq x_{11} + k_{11}$, or $x_{12} \geq x_{11} + 4$
(2) For job 2: $x_{22} \geq x_{21} + k_{21}$, or $x_{22} \geq x_{21} + 7$

(b) The total time must be greater than or equal to the starting time of that job entering last into the last processing station (punch press in our case) plus the processing time of this job at that station. This constraint can be expressed as:

(3) For job 1: $x_t \geq x_{12} + k_{12}$, or $x_t \geq x_{12} + 12$
(4) For job 2: $x_t \geq x_{22} + k_{22}$

(c) No job will enter a machine before the other job has been completed; that is, a machine must be empty to accept a job. This constraint can be expressed as an either-or type. For the shearing operation we have:

(5) either $x_{21} - x_{11} \geq k_{11}$ (if job 1 precedes 2)
(6) or $x_{11} - x_{21} \geq k_{21}$ (if job 2 precedes 1)

Integer programming in such a case requires, that we modify the original either-or set into the following equivalent set: (by utilizing d_i)

(5A) $x_{21} - x_{11} + d_1 M \geq k_{11}$
(6A) $x_{11} - x_{21} + (1 - d_1)M \geq k_{21}$

(where M is a large positive number and it acts as an upper bound) Similarly, for the punching constraints, we get:

(7) $x_{22} - x_{12} + d_2 M \geq k_{12}$ (for the case when job 1 precedes 2)
(8) $x_{12} - x_{22} + (1 - d_2)M \geq k_{22}$ (for the case when job 2 precedes 1)

Constraints (7) and (8) are not required in our case because it is obvious that the sequence achieved on the shear will be maintained for the punch press. However, when we have several jobs, a large number of either-or type constraints must be built for all possible pairs of jobs. The reader will quickly realize that the problem can thus become quite complicated. In a problem of six jobs and three machines, for example, we have about 100 constraints.

The practical value of this analytical method in the present state of integer-programming computations, therefore, is questionable. The interested reader is referred to Muth and Thompson [41].

6.10.5 ALLOCATION OF RESOURCES (SELECTION OF PROJECTS)

The allocation of limited resources to various projects is a familiar problem faced by many organizations.

A certain class of such problems can be formulated as integer-programming problems.[23] We shall illustrate by the following example.

Example: Five different sites for locating new manufacturing plants are available to the ABC Company. Expected construction time is three years and the company can spend no more than $30 million the first year, $34 million the second year, and $36 million the third year. It is estimated that the expected returns (present value) of the plants in the various alternative sites are: $115 million if a plant is built on site 1, $80 million on site 2, $132 million on site 3, $102 million on site 4, and $65 million on site 5. Table 6.14 gives estimated cost projections for each of the available sites.

The company cannot build on all five sites because of the unavailability of required funds. The problem, therefore, is to maximize expected return by choosing the proper number of sites in view of the data given in Table 6.14.

In formulating the equivalent integer-programming problem, we shall utilize the Boolean variables d_i. If $d_i = 1$, we proceed to build a plant on site i; a value of $d_i = 0$ implies that we should not build on site i.

The objective function is:

$$\max z = 115d_1 + 80d_2 + 132d_3 + 102d_4 + 65d_5$$

subject to the following budget constraints:

[23] For a complete discussion of the use of integer programming in budget allocation, see Weingartner [52].

$$\begin{aligned} 7d_1 + 6d_2 + 8d_3 + 7d_4 + 5d_5 &\leq 30 \\ 9d_1 + 7d_2 + 10d_3 + 8d_4 + 8d_5 &\leq 34 \\ 11d_1 + 8d_2 + 12d_3 + 9d_4 + 7d_5 &\leq 36 \end{aligned}$$

and also to the requirements that

$$0 \leq d_i \leq 1$$

and d_i is an integer.

We shall not actually solve the problem. The interested reader is referred to a special, more efficient enumeration algorithm developed by Balas [3] for the 0-1 type of problem. For other applications in the allocation area see Beged Dov [13] and Moodie and Mandeville [40].

Table 6.14

	MILLIONS OF DOLLARS			
	PROJECTED COSTS			PRESENT VALUE
SITE	1ST YEAR	2ND YEAR	3RD YEAR	OF EXPECTED RETURN
1	7	9	11	115
2	6	7	8	80
3	8	10	12	132
4	7	8	9	102
5	5	8	7	65
MAXIMUM AVAILABLE BUDGET	30	34	36	

6.10.6 INTEGER PROGRAMMING USED IN INCREASING RETURNS TO SCALE

One of the major problems in nonlinear programming is the case of an objective function with *increasing returns to scale.*

An interesting approach was suggested by Markowitz and Manne [37] whereby the nonlinear increasing-returns-to-scale problem can be approximated by an integer-programming problem. By using their approach, the optimal solution of the integer-programming problem can approximate the optimal solution for the nonlinear increasing-returns problem. An example of this approach is the *fixed-charge problem*, which we describe next.

In a product-mix problem we assume a linear objective function. Since profit per unit is the difference between the unit selling price and the unit manufacturing cost, the linear assumption requires that both, or the difference between the two, remain constant. This may not be the case in several practical situations where set-up costs and other charges are fixed for a given range of activity. In such cases, the profit function is the sum of a fixed charge and a variable charge (function of the number of units produced). Thus, the cost per unit will be relatively smaller at higher levels, and relatively larger at lower levels of production (Figure 6.6) Assuming that the price per unit remains constant, the profit per unit will be increased at higher levels of production. That is, we will have increasing returns to scale. The objective function thus is nonlinear, whereas the constraints are linear. Integer programming provides an interesting method of converting such a problem into a linear integer-programming problem. Next, we formulate such a problem as a mixed-integer problem.

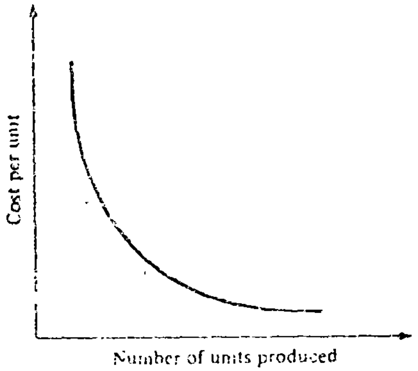


FIGURE 6.6

Example (discrete case): The ABC machine shop is required to supply parts for 750 units. Work can be performed on any of four available machines. The set-up costs for each machine, the technology of the machine, and the cost involved in producing parts for one unit on each machine are summarized in Table 6.15.

The problem is to find the quantity x_j to be produced on each machine to minimize total cost.

We introduce auxiliary Boolean variables d_j with the definition that:

"if $d_j = 0$; $x_j = 0$" implies "do not use machine j"

and

"if $d_j = 1$; $x_j > 0$" implies "use machine j"

Table 6.15

MACHINE	SET-UP COSTS	PROCESSING COST (PER UNIT)	MAXIMUM CAPACITY
A	$860	$2	400
B	270	4	300
C	490	3	600
D	520	3	1000

The problem can now be formulated as a mixed-integer problem:

$$\min z = (860d_1 + 270d_2 + 490d_3 + 520d_4) + (2x_1 + 4x_2 + 3x_3 + 3x_4)$$

(set-up cost) (processing cost)

s/t

$$x_1 + x_2 + x_3 + x_4 = 750$$

$$x_1 \le 400d_1, \qquad x_2 \le 300d_2, \qquad x_3 \le 600d_3, \qquad x_4 \le 1000d_4$$

$$x_j \text{ and } d_j \ge 0 \quad \text{and} \quad d_j \le 1, \text{ and integer}$$

6.10.7 OTHER USES OF INTEGER PROGRAMMING

Integer programming has also been used (see Balinsky [6]) to solve the following well-known problems:

1. The warehouse location problem
2. The delivery problem
3. Combinatorial-type problems
   (a) The marriage problem
   (b) The assignment problem
   (c) The transportation problem
   (d) The network flow problem
   (e) The caterer problem

6.11 DISCRETE PROGRAMMING

In certain programming problems it may be required that some or all of the variables are allowed to take on only certain discrete values, not necessarily integers. For example, x_1 may be limited to 1/4, 1/2, 3/4, . . . ; x_2 may be limited to 0.1, 0.2, ... Such requirements are common in real life; for example, milk is distributed in quarts and many canned foods weigh 1/8, 1/4, or 1/2 of a pound. In all such cases we can transform the discrete problem into an integer program by simple scaling

6.11.1 TRANSFORMATION TO INTEGER PROGRAMMING

Example 1:

$$\max z = 2x_1 + 2x_2$$

s/t

$$x_1 + 3x_2 \le 20$$

and x_1 can take values of 1/16, 2/16, . . . , and x_2 is an integer.

This is a practical example where the constraint and the variable x_2 are given in pounds.

We introduce a new variable $x_3 = 16x_1$. The transformed problem is an integer program:

$$\max z = \frac{x_3}{8} + 2x_2$$

s/t

$$x_3 + 48x_2 \leq 320$$

and x_2, x_3 are integers.

After finding an optimal solution for x_2 and x_3, we scale back by utilizing the relationship $x_3 = 16x_1$.

Example 2:

$$\min z = 2x_1 + 3x_2$$

s/t

$$x_1 + x_2 \leq 10$$

x_1 can take decimal values 0.1, 0.2, 0.3, ... only, and x_2 can take discrete values 1/2, 1, 3/2, ...

Transformation to integer programming requires two new variables y_1 and y_2 defined below. Let

$$y_1 = 10x_1, \text{ or } x_1 = 0.1y_1$$
$$y_2 = 2x_2, \text{ or } x_2 = 0.5y_2$$

Our transformed problem will be:

$$\min z = 2(0.1y_1) + 3(0.5y_2) = 0.2y_1 + 1.5y_2$$

s/t

$$0.1y_1 + 0.5y_2 \leq 10, \text{ or } y_1 + 5y_2 \leq 100$$

and y_1, y_2 are integers.

6.12 INTEGER NONLINEAR PROGRAMMING

Our discussion thus far has been limited to the case of *integer linear programming*, which involved a linear objective function subject to linear constraints.

In nonlinear programming the objective function, the constraints, or both, are nonlinear. In such cases, the additional integer requirement on some or all variables will transform the problem into integer nonlinear programming. The methods of rounding noninteger solution (Section 6.2), complete enumeration (Section 6.3) and the graphical method (Section 6.4) can be used in solving such problems, provided they are of small size. Analytical methods such as Gomory's can be employed only in limited cases, and both dynamic programming and the branch-and-bound technique can be used in relatively small and simple problems. For larger nonlinear problems, no effective solution methods are currently available. One attempt is to reduce the integer nonlinear programming into integer linear programming, and successful transformation in certain cases (such as separable functions) is reported by Woller [54].

6.13 COMPUTATIONAL ASPECTS

6.13.1 INTRODUCTION

We mentioned earlier (see Section 6.5.2) that Gomory's cutting-plane method yields an optimal solution to integer programming in a finite number of steps. This "finite number" has been found to be excessively large,[24] or even prohibitive, in many experiments and attempted applications. Hence, even with large-scale high-speed electronic digital computers, we face computational difficulties in the actual application of integer programming. Nevertheless, computational experience has demonstrated that some computer codes can solve efficiently certain practical problems with up to 100 variables and 50 constraints. This size is modest compared with the present-day capability for solving linear-programming problems with several thousand variables and constraints. But the state of the art is in rapid flux and it is reasonable to expect that models containing several hundred integer-valued variables will be solvable in the near future.

The purpose of this section is to present a condensed survey of the major computer codes available and to discuss some computational experience.

6.13.2 CODES

The SHARE catalogue is a good source for computer codes. Several surveys list the codes including format varieties and computational experience (for example, Haldi and Isaacson [25], Balinski and Spielberg (in Aronofsky [2]), and Trauth and Woolsey [49]). Some of the most acceptable codes are listed below:

(a) *IPM* 1 is an all-integer programming code (developed by IBM) and available through SHARE [50]. This code seems to have had the least computational success, according to Balinski [6], p. 304.

(b) *IPM* 2 is an all-integer programming code available through SHARE [51]. IPM 2 is an extension of IPM 1, including several additional subroutines, and has generally proved to be superior to IPM 1 as well as to most other integer-programming codes. The code can handle problems with $n \leq 100$ and $n + m \leq 200$ and was found to be very effective with small problems.

(c) *IPM* 3 though primarily based on Gomory's fractional algorithm, also, makes use of all-integer constraints. The code is available through SHARE

[24] This large number stems essentially from the arbitrary manner in which cutting planes can be introduced during the solution process and the fact that, once introduced, they remain as additional constraints.

[35]. It can handle problems with $n \leq 100$ and $n+m \leq 200$ and was found to be efficient in large problems.

(d) *LIP* is a fractional programming algorithm. One of its major advantages is its ability to print continuously the intermediate solutions which are valuable to the user. The code was developed by Haldi and Isaacson [26] and is available through SHARE. The code, which has two versions (LIP 1 and LIP 2), is especially efficient when used on large problems.

(e) *IPSC* is an all-integer programming code, which is user-oriented, having an unlimited flexibility for the user in modifying the code to suit individual problems. The code can be used even by an operator who is unfamiliar with integer programming. For details, see Woolsey and Trauth [55].

(f) *BBMIP* is a branch-and-bound mixed-integer programming code, based on the Land and Doig algorithm. The code is machine-independent and may be run on any Fortran IV compiler. The output is easily interpreted. For details see Shareshian [47].

(g) *ILPH* is a heuristic tree-search technique for integer linear programming [14]. The code is machine-independent and may be run on any Fortran IV compiler. The output is easily interpreted.

(h) The *CEIR* LP90/94 code is a general code with a provision for integer programming. It is able to accept problems of greater dimension than any other code since it is not all in the core. Its theoretical basis is similar to BBMIP. For details see Beale [12].

(i) The *ILP 2* code was developed by Summers as a Control Data Corporation code. It is based upon Gomory's all-integer method.

(j) *IPLP 6* is an IBM experimental code. Experience with IPLP 6 led to a convergence (optimal solution) in remarkably fewer steps than IPM 2.

(k) *Ophelie* is a mixed-integer code developed for CDC 6600. It has several versions.

(l) IBM's newest code is MPS/X MIP option.

Interesting results are reported by Roy *et al.* [46] in the use of Ophelie II; for example, a problem with 3884 continuous variables, 24 integer variables, 1244 constraints, and 20,233 nonzero matrix coefficients took about 6 minutes to solve (core time on CDC 6600).

6.13.3 SOME COMPUTATIONAL EXPERIENCE

The availability of several codes makes it difficult for the user to select the proper one. As Mears and Dawkins [38] point out, the proper selection of the best algorithm for integer programming is an art. The computational experience gained over the years will be extremely helpful to users. Some of the major conclusions found by Balinski [6] and by Mears and Dawkins [38] are as follows:

1. Problems frequently exceed the storage capacity of the codes.
2. Different arrangements of rows (constraints) yield different numbers of iterations.
3. "Results are very mixed; each algorithm, LIP 1, IPM 2 and IPM 3, is better for some problems than the others" (Balinski [6], p. 307).
4. A limitation of all codes (except the BBMIP) is that unless the optimal solution is attained, the user does not gain any useful intermediate data.
5. Cumulative roundoff errors were encountered.
6. The ILPH code was found inferior to the cutting-plane constraint codes (IPM 1, 2, and 3, and LIP 1).
7. The BBMIP is faster and more reliable than the ILPH but slower than the cutting-plane constraint codes.
8. IPM 2 requires fewer iterations than other cutting-plane constraint codes.
9. The number of iterations required by IPM 2 and LIP 1 can be described by linear models (see Mears and Dawkins [38]).

Based on the authors' experience with small- and medium-size problems (up to 50 constraints), the BBMIP was found to be a relatively efficient programming code.

Some of the problems encountered in integer programs (based on Gomory's cutting-plane method) are: (1) Because of the nature of the methods, the *optimal* answer may not be achieved in a reasonable number of iterations. Unfortunately, since there is no way to determine in advance how many iterations are needed to obtain the solution, this system can be very inefficient. (2) The *optimal* integer solution obtained by the cutting-plane methods is determined by testing the solution for integer values—that is, those variables that are required to be integers. Computers with a floating-point feature, however, may have an inaccuracy (roundoff error) problem, resulting in a need for special tests to identify the integer optimal solution.

Finally, computational experience revealed that problems that are extremely difficult or impossible to solve using a given code may be easily solved with another. (For several interesting examples see Trauth and Woolsey [55]).

6.14 CONCLUDING REMARKS

This text is devoted to the applied aspects of mathematical-programming methods. We have thus far restricted our presentation to linear- and integer-programming models. Our next task is to explain and illustrate, at as elementary a level as possible, some of the nonlinear models. The next chapter is devoted to this task.

BIBLIOGRAPHY

1. ABADIE, J., ed., *Integer and Nonlinear Programming*. Amsterdam: North-Holland, 1970.
2. ARONOFSKY, J. S., ed., *Progress in Operations Research—The Relationship Between Operations Research and the Computer*. New York: Wiley, 1969.
3. BALAS, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Operations Research*, vol. 13, no. 4, July 1965.
4. BALAS, E., "Duality in Discrete Programming," in Kuhn [31].
5. BALINSKI, M. L. "On Finding Solutions to Linear Programs," *Proceedings of the IBM Scientific Symposium on Combinatorial Problems*, March 16, 1964.

6. Balinski, M. L., "Integer Programming: Methods, Uses, Computation," *Management Science*, vol. 12, no. 3, Nov. 1965, pp. 253–313.

7. Balinski, M. L., "Integer Programming; Methods, Uses, Computation," in Kuhn [31].

8. Balinski, M. L., "On Recent Developments in Integer Programming," in Kuhn [31].

9. Balinski, M. L., and R. E. Quandt, "On an Integer Program for a Delivery Problem," *Operations Research*, vol. 12, 1964, pp. 300–304.

10. Baumol, W. J., and P. Wolfe, "Warehouse Location Problem," *Operations Research*, vol. 6, 1958, pp. 252–263.

11. Beale, E. M. L., P. A. B. Hughes, and R. E. Small, "Experiences in Using a Decomposition Program," *The Computer Journal*, vol. 8, no. 1, April 1965.

12. Beale, E. M. L., *Mathematical Programming in Practice*. London: Sir Isaac Pitman & Sons Ltd., 1968.

13. Beged Dov, A. G., "Optional Assignment of R & D Projects in a Large Company Using an Integer Programming Model," *IEEE Transactions on Engineering Management*, vol. EM-12, no. 4, December 1965.

14. Cooper, L., and C. Drebes, *Investigation in Integer Linear Programming by Direct Search Methods*. Report No. COO-1493-10, Department of Applied Mathematics and Computer Science, Washington University, St. Louis, Mo.

15. Dantzig, G. B., "On the Significance of Solving Linear Programming Problems with Some Integer Variables," *Econometrica*, vol. 28, 1960, pp. 30–44.

16. Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson, "Solution of a Large Scale Traveling Salesman Problem," *Journal of the Operations Research Society of America*, vol. 2, 1954, pp. 393–410.

17. Dantzig, G. B., D. R. Fulkerson, and S. M. Johnson, "On a Linear Programming, Combinatorial Approach to the Traveling Salesman Problem," *Operations Research*, vol. 7, 1959, pp. 58–66.

18. Ford, L. R., Jr., and D. R. Fulkerson, *Flows in Networks*. Princeton, N.J.: Princeton University Press, 1963.

19. Gilmore, P. C., and R. E. Gomory, *Sequencing a One State Variable Machine: A Solvable Case of the Traveling Salesman Problem*. Report RC-1103, IBM Watson Research Center, Yorktown Heights, N.Y., January 24, 1964.

20. Gilmore, P. C., and R. E. Gomory, "A Solvable Case of the Traveling Salesman Problem," *Proceedings of the National Academy of Sciences*, vol. 51, 1964, pp. 178–181.

21. Gomory, R. E., "All-Integer Programming Algorithm," in Muth and Thompson [41], pp. 193–206. First issued as Research Report RC-189, IBM Research Center, January 29, 1960.

22. Gomory, R. E., *An Algorithm for the Mixed Integer Problem*. Rand Report RM-2597, The Rand Corporation, Santa Monica, Calif., July 7, 1960.

23. Gomory, R. E., "On the Relation Between Integer and Noninteger Solutions to Linear Programs," *Proceedings of the National Academy of Sciences*, 1965, pp. 260–265.

24. Gomory, R. E., and William J. Baumol, "Integer Programming and Pricing," *Econometrica*, vol. 28, 1960, pp. 521–550.

25. Haldi, J., and L. M. Isaacson, "A Computer Code for Integer Solutions to Linear Problems," *Operations Research*, vol. 13, November–December 1965, pp. 946–959.

26. Haldi, J., and L. M. Isaacson, *User's Manual: Linear Integer Programming I (LIP I)*. SHARE General Program Library. New York: IBM, 1965.

27. Hu, T. C., *Integer Programming and Network Flows*, Reading, Mass.: Addison-Wesley, 1969.

28. IBM, *Mathematical Programming System/360*. Nos. H20-0476-0 and H20-0136-2, IBM Technical Publication Department, White Plains, N.Y., 1967.

29. Johnson, S. M., "Optimal Two and Three Stage Production Schedules and Set-up Times Included," *Naval Research Logistics Quarterly*, vol. 1, March 1954. Also Chapter 2 in Muth and Thompson [41].

30. Kuhn, H. W., "On Certain Convex Polyhedra," *Bulletin of the American Mathematical Society*, vol. 61, 1955, pp. 557–558.

31. Kuhn, H. W., ed., *Proceedings of the Princeton Symposium on Mathematical Programming*, Princeton, N.J.: Princeton University Press, 1970.

32. Kunzi, H. P., H. G. Tzschuch, and C. A. Zehnder, *Numerical Methods of Mathematical Optimization*. New York: Academic Press, 1968.

33. Land, A. H., and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, vol. 28, 1960, pp. 497–520.

34. Lawler, E. L., and D. W. Wood, "Branch-and-Bound Methods, A Survey," *Operations Research*, vol. 14, 1966, pp. 699–719.

35. Levitan, R. E., *IPM 3*. SHARE Distribution, No. 1190, September 1961.

36. Little, J. D. C., *et al.*, "An Algorithm for the Traveling Salesman Problem," *Operations Research*, vol. 11, 1963.

37. Markowitz, H. M., and A. S. Manne, "On the Solution of Discrete Programming Problems," *Econometrica*, vol. 25, 1957, pp. 84–110.

38. Mears, W. J., and G. S. Dawkins, *Comparison of Integer Programming Algorithms*. Paper presented at the 1968 Joint National Meeting of ORSA and TIMS, San Francisco, Calif., May 1–3, 1968.

39. Mitten, L. G., "Branch and Bound Methods: General Formulation and Properties," *Operations Research*, January–February 1970.

40. Moodie, C. L., and D. E. Mandeville, "Project Resource Balancing by Assembly Line Balancing Techniques," *The Journal of Industrial Engineering*, vol. 17, July 1966.

41. Muth, J. F., and G. L. Thompson, eds., *Industrial Scheduling*. Englewood Cliffs, N.J.: Prentice-Hall, 1963.

42. Orchard-Hays, W., "Structure of Mathematical Programming Systems," in *ACM Proceedings of 23rd National Conference*. Princeton, N.J.: Brandon/Systems Press, 1966, pp. 439–458.

43. Peterson, C. C., "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R & D Projects," *Management Science*, vol. 13, no. 9, May 1967, pp. 736–745.

44. Peterson, C. C., "Integer Linear Programming," *The Journal of Industrial Engineering*, vol. 18, no. 8, August 1967, 0pp. 456 464.

45. Rappaport, A., "Integer Programming and Managerial Analysis," *The Accounting Review*, April 1969, pp. 297–299.

46. Roy, B. *et al.*, "From S.E.P. Procedure to the Mixed Ophelie Program," in Abadie [1].

47. Shareshian, R., *User's Manual: Branch and Bound Mixed Integer Programming-BBMIP*. File No. 360D-15.2.005, IBM Corporation, 1967.

48. TOMLIN, J. A., "Branch and Bound Methods for Integer and Non-convex Programming," in Abadie [1].

49 TRAUTH, C. A., JR., AND R. E. D. WOOLSEY, "Integer Linear Programming: A Study in Computational Efficiency," *Management Science*, vol. 15, no. 9, May 1969, pp 481 493.

50. WADE, C. S., AND R. E. GOMORY, *IPM 1*. SHARE, No. 1192, September 1961.

51. WADE, C. S., AND R. E. GOMORY, *IPM 2*. SHARE, No. 1191, September 1961.

52. WEINGARTNER, H. M., *Mathematical Programming and the Analysis of Capital Budgeting Problems*. Chicago: Markham Publishing Company, 1967.

53. WIEST, J. D., "Heuristic Programs for Decision Making," *Harvard Business Review*, September–October 1966, pp. 129–143.

54. WOILER, S., *Implicit Enumeration Algorithms for Discrete Optimization Problems*. Report 4, Department of Industrial Engineering, Stanford University, Stanford, Calif., May 1967

55. WOOLSEY, R. E. D., AND C. A. TRAUTH, JR., *IPSC, A Machine Independent Integer Linear Program*. SC-RR-66-433, Sandia Corporation, Albuquerque, N.Mex., July 1966.

PROBLEMS

6.1 A U.S. student organization is chartering flights to Europe each summer. In 1970, 2000 students registered for the flights. The WW Company, which provides the airplanes, has three available types: type 1 can carry up to 90 students, with a crew of 5 and a cost of $8000 (15 such flights available); type 2 can carry up to 150 students, with a crew of 9 and cost of $11,000 (10 such flights available); and type 3 can carry up to 360 students, with a crew of 18 and a cost of $25,000 (only one such flight available). The company can spare 120 crewmen for the entire mission. Find the best schedule for the WW company.

6.2 Explain why the addition of Gomory's constraint cuts the feasible area of solutions.

6.3 Products A, B, and C, are to be made on three machines. Net profit per unit of A, B, and C respectively, is $21, $26, and $22. Each product is processed by three different machines. Processing time per unit of production and data on machine availability are shown in the table below:

	PRODUCT			MACHINE AVAILABILITY (MINUTES PER TWO-WEEK SCHEDULING PERIOD)
MACHINE	A	B	C	
1	273	221	374	9282
2	273	442	187	9282
3	91	182	159	4732

Assuming no set-up requirements, find the best product mix (that is, what products should be produced and in what quantities) if the production schedule must meet all-integer constraints; that is, no fractions of products can be produced. Use Gomory's method; however, if you do not find an optimal integer solution three iterations after you add Gomory's constraint, stop.

(a) Find the all-integer solution using Gomory's method.
(b) Set the dual to the all-integer problem and solve it.
(c) Find the opportunity cost of indivisibility with the aid of the dual's optimal variables.

6.4 Given:

$$\max z = 2.5x_1 + 2.25x_2 + 0.5x_3$$

s/t

$$7.5x_1 + 7.9\ x_2 \qquad \leq 75$$

and either

$$2.5x_1 + 3.2\ x_2 + 7.5x_3 \leq 38.5$$

or

$$1.4x_1 \qquad + 5.8x_3 \leq 18$$

(a) Formulate the problem as a mixed-integer programming problem.
(b) Write the initial simplex tableau to part (a).
(c) Solve the problem.

6.5 Given:

$$\max z = 3x_1 + 2x_2$$

s/t

$$\frac{20}{3}x_1 + 10x_2 \leq 100$$

$$10x_1 + 5x_2 \leq 100$$

(a) Find an all-integer solution graphically.
(b) Find an all integer solution by Gomory's method, manually.
(c) Assuming that either the first or the second constraint holds, set the problem as an all-integer programming problem. Solve graphically.

6.6 Given:

$$\max z = 14x_1 + 20x_2 + 10x_3$$

s/t

$$6x_1 + 10x_2 + 3x_3 \leq 100$$

and either

$$8x_1 + 10x_2 + 6x_3 \leq 120$$

or

$$4x_1 + 8x_2 + 9x_3 \leq 150$$

(a) Formulate as an all-integer problem. Write the first simplex tableau.
(b) Solve the problem.

6.7 Given:

$$\max z = 20x_1 + 5x_2$$

s/t

$$13\frac{4}{7}x_1 + 10x_2 \leq 100$$
$$10x_1 \leq 35$$
$$10x_1 + 15x_2 \leq 120$$

and x_1, x_2 are integers.

(a) Find the all-integer solution using Gomory's method.
(b) Set the dual to the all-integer problem and solve it.
(c) Find the opportunity cost of indivisibility with the aid of the dual's optimal variables.

6.8 The Elster Machine Corporation has a department specializing in job-shop orders. One day the foreman received an order for three jobs, whose processing times on one of several available machines with equivalent capabilities, are

JOB	TIME (HOURS)
A	4
B	6
C	7

Each job is processed through one machine only. Once a job is started on a machine it must be completed. The department can spare only one employee for the order. The employee can handle no more than two machines simultaneously. The foreman's objective is to minimize the total elapsed time required for one production run. Find the best scheduling.

(a) Solve by enumeration.
(b) Set up as a mixed-integer program, but do not solve.

Assume:

1. No setups are involved.
2. Processing times are constant.
3. The machines are working without interruptions.
4. Only one job can be processed on a machine at one time.

6.9 The ABC Company is producing three types of canned beef. Type 1 is packed in ½-pound cans, type 2 is packed in 1-pound cans, and type 3 is packed in 3-pound cans. Net profit from selling each pound of canned beef is 14 cents for type 1, 20 cents for type 2, and 10 cents for type 3. Production is subject to the following constraints:

$$6x_1 + 10x_2 + 3x_3 \leq 100 \text{ pounds}$$
$$8x_1 + 10x_2 + 6x_3 \leq 120 \text{ pounds}$$
$$4x_1 + 8x_2 + 9x_3 \leq 150 \text{ pounds}$$

where x_i is the number of pounds of product i. The objective of the company is profit maximization.

(a) Find the best product mix if the number of cans produced must be integer. (*Hint:* A transformation is advisable.) *Note:* If you do not have a computer program for integer programming, stop after three iterations. Try to enumerate for optimal solution.
(b) Determine the best production plan if the number of cans produced must be integer, and if at least 10 cans and no more than 40 cans of beef type 1 should be in the program. *Note:* If you do not have a computer program, try to enumerate.

6.10 The research department of ABC is selecting projects for the next two years. Seven proposed projects are to be evaluated. The yearly cost of each product in man-hours required and the data on available man-hours are given in the table below. Also, the expected profits (discounted to time zero) are given. The research department wants to maximize its profits. Find the projects they should select.

(a) Set up as a mixed-integer programming problem.
(b) Solve (use common sense if you have difficulties in getting results with Gomorian constraints).

	MAN-HOURS REQUIRED		DISCOUNTED EXPECTED PROFITS IN THOUSANDS OF DOLLARS
PROJECT	1ST YEAR	2ND YEAR	
A	1000	4000	120
B	1200	2000	100
C	1800	1600	80
D	2000	2400	140
E	1200	1800	100
F	2600	2000	160
G	2200	2200	140
AVAILABLE MAN-HOURS/YEAR	10,000	12,000	[illegible]

6.11 Four different processes are available for producing a certain paint. The processing cost of each gallon in any of the four available processes, with the maximum capacity of each process, and its set-up costs are given in the table below. Assume that a daily demand of 35,000 gallons must be supplied. Find the best processing schedule (minimize total costs). Base your solution on an elapsed time of one day.

PROCESS	SET-UP COST, DOLLARS	PROCESSING COST, CENTS PER GALLON	MAXIMUM CAPACITY, GALLONS
A	500	6	20000
B	600	5	15000
C	1000	4	40000
D	600	3	25000

(a) Formulate the problem as an integer-programming problem.
(b) Which processes should be used, and to what extent, in order to minimize total cost. Solve this part with the aid of a computer.
(c) Find the best and the second-best production schedule by a common-sense approach.

6.12 Explain why integer programming can be viewed as nonlinear programming.

6.13 Given:

$$\min z = 20x_1 + 22x_2 + 24x_3 + 24x_4$$

s/t

$$\begin{aligned} 5x_1 + 6x_2 + 3x_3 + 4x_4 &\geq 12 \\ 3x_1 + 3x_2 + 5x_3 + 4x_4 &\geq 11 \\ 2x_1 + 2x_2 + 5x_3 + 6x_4 &\geq 10 \end{aligned}$$

$x_i \geq 0$, and x_i can take either the value of 0 or the value of 1. Find the optimal solution by enumeration, and explain why enumeration is the best technique in this case.

6.14 In solving an all-integer problem by Gomory's method it may happen that after we add the Gomory's cutting plane, the resulting program (not necessarily all-integer yet) will have multiple solution. Illustrate graphically this sort of situation, explain its source, and suggest a way to avoid it.

6.15 Write the following sets of constraints as constraints for the mixed-integer set.

(a) Either

$$3x_1 + x_2 - 2x_3 \leq 10$$

or

$$2x_1 - 3x_2 \qquad \geq 6$$

or

$$2x_2 - x_3 \leq 8$$

(b) *At least* two of the following constraints hold:

$$\begin{aligned} 2x_1 + x_2 &\leq 10 \\ x_1 - x_2 &\leq 2 \\ x_2 &\leq 1 \end{aligned}$$

(c) Given four sets of constraints, *any* two sets must hold.

$$\begin{aligned} (1)\quad 2x_1 + x_2 &\leq 6 \\ x_1 \qquad &\leq 1 \\ (2)\quad 2x_1 - x_2 &\leq 5 \\ x_2 &\leq 1 \\ (3)\quad 3x_1 + 2x_2 &\leq 12 \\ x_1 - x_2 &\leq 1 \\ (4)\quad x_1 + x_2 &\leq 8 \\ x_1 \qquad &\leq 2 \end{aligned}$$

6.16 Given:

$$\min z = 2x_1 + 3x_2 + x_3$$

s/t

$$\begin{aligned} x_1 + 2x_2 - x_3 &\geq 20 \\ x_1 + 3x_2 \qquad &= 18 \\ 3x_1 \qquad - x_3 &\leq 16 \end{aligned}$$

Formulate the problem as a mixed-integer problem when
(a) At least any two constraints must hold.
(b) Either the first constraint, or the second one, or any combination involving two out of the three constraints must hold.
(c) A new constraint is added: x_3 must be either 2, 3, or 4.

6.17 Given:

$$\min z = 3x_1 - x_2 + 2x_3$$

s/t

$$\begin{aligned} (1)\quad x_1 + x_2 + x_3 &\geq 16 \\ (2)\quad 2x_1 + x_2 - x_3 &\geq 18 \\ (3)\quad x_1 + 3x_2 + 5x_3 &\geq 24 \end{aligned}$$

(4) $x_1 - x_2 +$ [illegible]

Use integer programming to express the following:
(a) At least three of the constraints must hold.
(b) No more than any two constraints must hold.
(c) No more than any single constraint must hold.

6.18 Given a fixed-charge problem:

$$\min z = f(x_1) + g(x_2)$$

where

$$f(x_1) \begin{cases} 5+3x_1 & \text{if } x_1>0 \\ 0 & \text{if } x_1=0 \end{cases}$$

and

$$g(x_2) \begin{cases} 20+x_2 & \text{if } x_2>0 \\ 0 & \text{if } x_2=0 \end{cases}$$

subject to a set of constraints.

(a) Formulate as a mixed-integer problem.

(b) Give the general formulation of this type of a fixed-charge problem.

6.19 The pipeline design problem (Problem 3.18) allows any combination of sizes in one span. Suppose that we require that there will be only *one size* in each span.

(a) Formulate the problem as an integer-programming problem with the aid of Boolean variables.

(b) Solve the problem.

6.20 The ABC Corporation wants to maximize the present value of the dividends it will pay. The discount rate it must use is 10 percent. The firm has $600 now, it will receive $200 in the next time period, and $100 in the third time period from investments now outstanding. At this time the firm knows that there are nine investment alternatives to look at for the next three time periods. The firm's policy is to ignore the possibility of future advantageous investments until they are definite. The relevant data are given in the accompanying table.

INVESTMENT ALTERNATIVES	INVEST IN PERIOD			RETURN IN PERIOD				
	1	2	3	2	3	4	5	6
1	200	—	—	100	90	80	—	—
2	300	—	—	—	150	300	—	—
3	400	—	—	300	200	—	—	—
4	—	100	—	—	120	—	—	—
5	—	200	—	—	—	100	200	—
6	—	400	—	—	180	180	180	—
7	—	—	200	—	—	100	150	—
8	—	—	200	—	—	150	80	—
9	—	—	300	—	—	—	—	500

The firm always has the option of earning 5 percent on money not paid out in dividends, but invested in bonds.

(a) Find the best investment schedule. Formulate only (write first tableau.)

(b) Find the investment alternatives that the company should select; that is, solve part (a) by use of a computer.

(c) How much should be paid in dividends each investment period? (Include all six periods.)

(d) What is the present value of the dividend stream?

Hint: This is a problem in mixed-integer programming. Let $x_i = 1$ if investment is made, $x_i = 0$ if no investment is made, and $i = 1, \ldots, 9$. Let $x_j \geq 0$, $j = 10$ to 20; dividends or 5 percent money can be at any positive value. (x_{10} to x_{15} are dividend payments; x_{16} to x_{20} are the 5 percent alternatives.)

6.21 A contractor must decide which of 12 contract offers he should accept. The only constraints that are binding are on the capital needed to finance the different contracts. These 12 contracts are the only offers that will be made in the next five years. The contractor can hold his money in 5 percent liquid securities, if he finds it desirable to keep the money available for future contracts. The contractor wishes to maximize the present value of the money he draws from the firm. He will eventually draw all the money from the firm. He discounts future drawings by a factor of 10 percent for each period. He feels that a one-dollar drawing made nine periods from now is worth 0.424 dollar now. The contract offers are given in Tables 1 and 2. All values are given in millions of dollars. The contractor now has 60 million dollars, which he can use for drawings, 5 per cent securities, or two contract offers for the first period.

(a) Find the best policy. Formulate only.

(b) Solve (use a computer).

Table 1 Expenditures

CONTRACT	PERIOD				
	1	2	3	4	5
1	20	—	—	—	—
2	30	—	—	—	—
3	—	10	—	—	—
4	—	30	—	—	—
5	—	20	—	—	—
6	—	—	30	—	—
7	—	—	20	—	—
8	—	—	—	10	—
9	—	—	—	20	—
10	—	—	—	—	30
11	—	—	—	—	20
12	—	—	—	—	30

Table 2 Returns

	PERIOD									
CONTRACT	1	2	3	4	5	6	7	8	9	10
1	—	10	15	—	—	—	—	—	—	—
2	—	10	10	10	10	10	—	—	—	—
3	—	—	7	7	—	—	—	—	—	—
4	—	—	9	15	21	—	—	—	—	—
5	—	—	5	7	6	6	6	—	—	—
6	—	—	—	11	27	—	—	—	—	—
7	—	—	—	8	10	11	—	—	—	—
8	—	—	—	—	7	6	—	—	—	—
9	—	—	—	—	8	8	8	3	—	—
10	—	—	—	—	—	12	12	11	11	—
11	—	—	—	—	—	9	9	9	6	—
12	—	—	—	—	—	10	10	10	9	4

Discussion:

This problem is designed to show how money streams can be so interrelated that the objective is not necessarily to pick the contracts with the highest rate of return. The time at which the capital is returned to the company is very important. The rate of return that makes the present value of each contract zero is given in the accompanying tabulation. It can be seen that none of the contracts has a rate less than 10 percent.

CONTRACT	RATE OF RETURN, PERCENT
1	15
2	20
3	25
4	20
5	15
6	15
7	20
8	20
9	15
10	20
11	25
12	15

These rates are for the year when the contract starts; therefore what the capital was used for before the contract was started is important. If the capital had to be held in 4 percent liquid securities until the contract period started, it may have a true rate of return that is less than the required 10 percent. On the other hand, if the capital had previously been in other contracts, this rate may still be in error. Because the acceptance of one contract may stop the acceptance of others, its rate needs to be adjusted.

Put the problem in the form of a mixed-integer programming problem. The objective function shows the present value of the drawings made in each of the 10 periods. This is done by making the coefficients of the drawing in the objective function the appropriate 10 percent discount factor. All other variables have zero coefficients in the objective function. The first 12 variables are for the contracts and they must have a value of zero or one. The next 10 variables are for the drawings and the last four variables are for 5 percent liquid securities. There is a constraint for each period, which reflects the sources and uses of capital in a period. These constraints are all strictly equal to constants.

6.22 Given below is a minimization problem presented in the form of the first tableau (original constraints were in the ≥ form).

COSTS	15	17	19	20	20	21	21	21	21	22	22	22	22	22	23
QUANTITY	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
16	3	3	7	4	2	4	3	5	4	2	3	6	3	4	2
21	4	2	5	5	6	4	6	1	5	9	5	3	5	5	7
26	4	4	9	7	5	7	6	7	6	5	6	8	5	7	5
26	4	3	6	6	7	5	7	4	6	9	6	5	6	6	8
14	0	4	1	3	5	4	2	8	3	0	4	4	4	3	2
9	2	1	3	2	1	2	2	2	2	3	2	2	2	2	2
5	1	0	3	2	2	1	1	0	1	0	0	1	0	1	1
15	3	1	1	1	1	2	3	4	3	9	6	2	4	4	5
14	4	2	3	2	2	3	3	5	3	4	4	4	3	4	3
17	1	4	0	1	3	4	2	9	4	5	7	4	6	4	4

It is also required that all variables take either the value of zero or the value of 1 (that is, $x_j = 0$ or 1). Find the optimal solution, using any method desired (for example, Balas [3]).

6.23 Consider the following transportation problem.

	I	II	III	
FROM A	0.2 7	0.3 5	0.4 4	100
FROM B	0.3 6	0.2 10	0.5 1	200
	60	70	80	

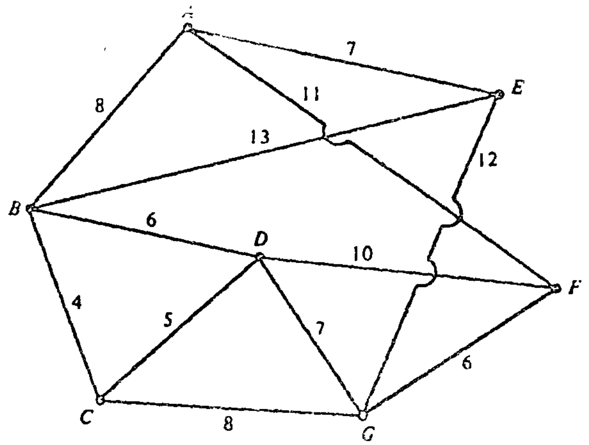
The upper (left) numbers in the cells are transportation costs per unit shipped, and the lower (right) numbers are fixed charges. Find the best shipment plan.
(a) Set up as an integer-programming problem.
(b) Solve.

6.24 Solve the illustrative example in Section 6.9.2 by the branch-and-bound approach.
(a) If x_3 must be integer.
(b) For the all-integer solution.

6.25 Solve the assignment problem presented in Table 5.36 with the aid of the branch-and-bound technique.

6.26 A salesman has n cities to visit. He knows the cost of traveling from city i to city j (c_{ij}). He must visit all cities and can be in each city only once. The problem is to find the least expensive route. The salesman starts from his home city and returns to it. Propose a general solution to the problem using integer programming.

6.27 Use the branch-and-bound approach (manually) to solve the traveling-salesman problem given in problem 8.5. The objective is to minimize the total distance. The tour starts and ends in New York. Each city must be visited *once and only once*. Show the tree and the branches.



NONLINEAR PROGRAMMING 7

7.1 INTRODUCTION

In many real-life situations the objective function and/or the constraints are nonlinear. The branch of mathematical programming that deals with these types of problems is labeled as *nonlinear programming* (NLP).

Unlike the simplex method, which is a general model for solving linear-programming problems, there is no *general* method for solving *all* nonlinear-programming problems. Instead, various computational techniques have been developed to solve different categories of nonlinear problems. In general, nonlinear programming requires a high level of mathematical background. In this text we shall limit our discussion to those nonlinear-programming methods that are relatively simple.

7.2 CLASSIFICATION AND SOME ASPECTS OF NONLINEAR-PROGRAMMING MODELS

In Figure 7.1 we show a classification of deterministic nonlinear-programming problems (based on Dantzig [11], p. 8). Of the various categories shown, most of the formal work has been performed in the area of convex programming (convexity and concavity are discussed in Appendix [illegible]

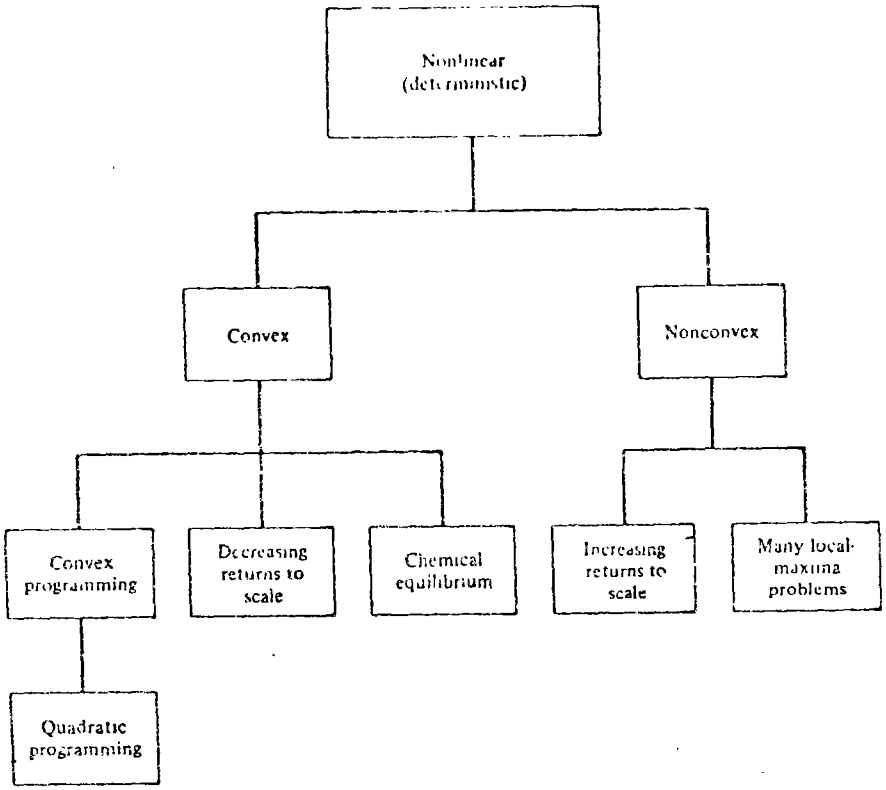


FIGURE 7.1

7.2.1 DEFINITIONS

a. Nonlinear Relations

Functional relationships that contain such terms as $2x^3$, log $(1/x)$, and $2e^x$, as well as noncontinuous functions, are termed *nonlinear functions*. In general, any functional relation that does not meet the linearity conditions is considered nonlinear. In a two-dimensional space, such relations (for the case of continuous functions) are represented by curves rather than by straight lines (see Figure 7.2).

b. Nonlinear Programming

A few examples of nonlinear-programming problems are as follows:

(1) $\min z = 3x^2 - 2y$

s/t

$$\begin{aligned} 3x + 4y &\geq 12 \\ x - y &\geq 3 \\ x,y &\geq 0 \end{aligned}$$

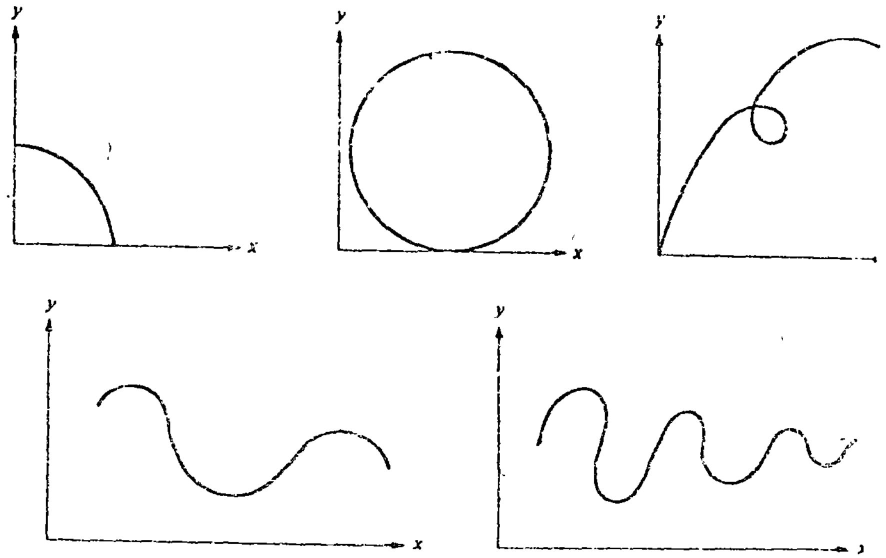


FIGURE 7.2

(2) $\min z = 2xy - \dfrac{2}{x}$

s/t

$$\begin{aligned} 3x^2 + 2y &\leq 100 \\ x + y^3 &\leq 80 \\ x,y &\geq 0 \end{aligned}$$

(3) $\max z = 5x + 7y$

s/t

$$\begin{aligned} x^2 + 2y^3 &\leq 65 \\ 2xy + y &\geq 50 \\ x,y &\geq 0 \end{aligned}$$

A wide range of practical problems with constrained maximization (or minimization) and involving decreasing returns to scale fall under the category of nonlinear-programming models. Let us illustrate:

The linear-programming example discussed in Chapter 3 had a linear objective function $3x_1 + 3.5x_2$ based on the assumption that no matter what quantities we sell, our profit from producing one unit of eggplant (x_1) will be \$3, and from one tomato plant (x_2) will be \$3.5. Let us now assume that we face an objective function in which *decreasing*, rather than constant, returns to scale exist.[1] Let us further assume that in our example the profit

[1] This can occur because, as volume is increased, distribution costs may go up disproportionately. Another common reason is that as volume is increased, per-unit price declines. This is shown in Figure 7.3, where a shift in the supply curve from S_1 to S_2 results in a decrease in price (from p_1 to p_2). Also, decreasing returns to scale can result if production cost per unit remains constant, or decreases at a rate slower than the corresponding decrease in price per unit.

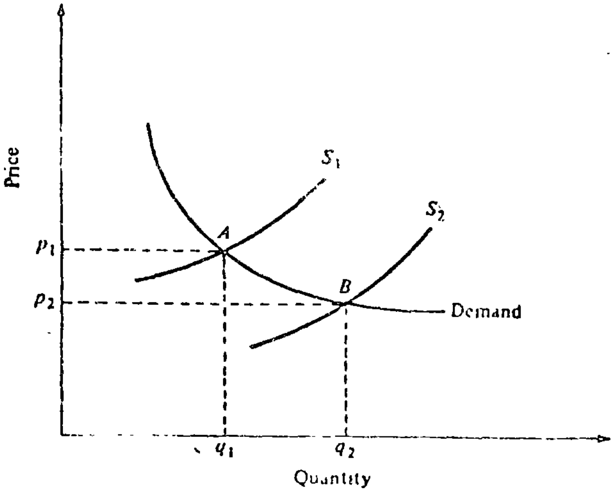


FIGURE 7.3

from selling eggplants (z_1) is decreasing linearly according to the relation

$$z_1 = 3 - \frac{x_1}{1000}$$

and the profit from selling tomatoes (z_2) is decreasing linearly according to the relation:

$$z_2 = 3.5 - \frac{x_2}{400}$$

Then the total-profit function that we would like to maximize becomes

$$z_1x_1 + z_2x_2 = \left(3 - \frac{x_1}{1000}\right)x_1 + \left(3.5 - \frac{x_2}{400}\right)x_2$$

$$= 3x_1 - \frac{{x_1}^2}{1000} + 3.5x_2 - \frac{{x_2}^2}{400}$$

This total-profit function is clearly nonlinear.

Another example of nonlinearity can be illustrated by considering the question of "resource" utilization. We assumed in linear-programming models that the coefficients of the constraints (input-output coefficients) remain constant (constant technology). This is not always true. With increasing utilization of a certain process and/or machine, we may find that the processing time per unit may increase because of excessive heat or decrease because of savings in set-up times per unit. In such cases the constraints behave in a nonlinear manner. For example, the constraint $x_1 + 2{x_2}^2 \leq 4000$, is clearly a nonlinear constraint.

A graphical example of partially nonlinear boundary of a feasible area

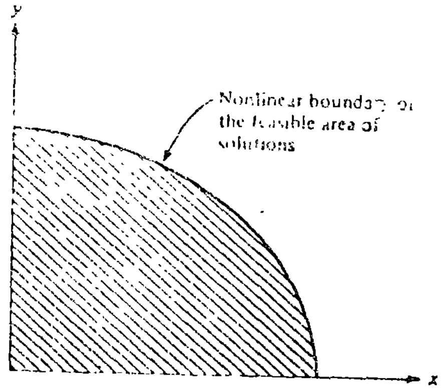


FIGURE 7.4

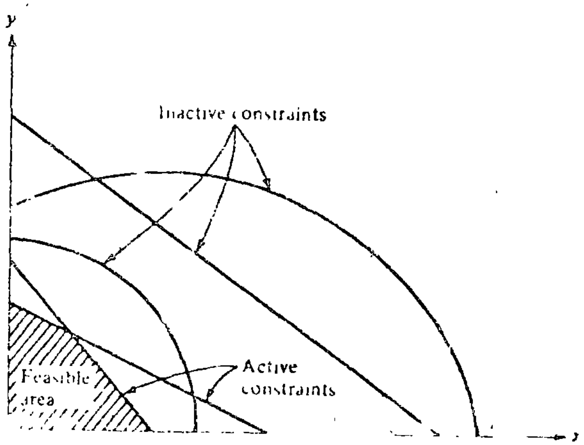
c. Convex Programming

Convex programming involves problems of minimizing a convex objective function (or maximizing a concave objective function) over convex regions. (For a discussion of convexity and concavity, see Appendix C.)

d. Types of Constraints

To facilitate understanding of the material in this chapter, note the following four types of constraints:

1. Constraints that form convex regions (see Figure 7.7a).
2. Constraints that form nonconvex regions (see Figure 7.7b).
3. Constraints that are *active* or *defining* constraints (see Figure 7.5).
4. Constraints that are *inactive* or *redundant* constraints (see Figure 7.5).



e. Returns to Scale

Diminishing returns to scale imply *decreasing marginal profit* or *increasing marginal cost.*

7.2.2 GENERAL STRUCTURE OF THE NONLINEAR-PROGRAMMING PROBLEM

A reasonably general nonlinear-programming can be expressed as follows: Find

$$x_1, x_2, \ldots, x_n$$

so as to maximize

$$f(x_1, x_2, \ldots, x_n)$$

s/t

$$\begin{aligned} g_1(x_1, \ldots, x_n) &\le 0 \\ g_2(x_1, \ldots, x_n) &\le 0 \\ &\cdots\cdots\cdots \\ &\cdots\cdots\cdots \\ &\cdots\cdots\cdots \\ g_m(x_1, \ldots, x_n) &\le 0 \end{aligned} \tag{7.1}$$

and

$$x_1, x_2, \ldots, x_n \ge 0$$

Any or all of the f and/or $g_1, \ldots, g_m$ may be nonlinear. ~~As in the linear-programming model,~~ we assume

1. Certainty (that is, deterministic models).
2. ~~Non-negativity of all variables (except that it is possible to handle a free variable by expressing it as the difference of two nonnegative variables).~~
3. Maximization or minimization as the only goal.
4. Divisibility.

7.2.3 DIFFICULTIES IN SOLVING NONLINEAR-PROGRAMMING PROBLEMS

a. General

Methods of solving linear-programming problems are based on the property that the optimal solution can be found at extreme points of the solution space. This property enables us to limit our search to corner points of the region of feasible solution and thus obtain an optimal solution in a finite number of iterations. Unfortunately, no such universality regarding the nature of optimal solutions can be established in nonlinear-programming models. In nonlinear programming the optimal solution can be any point along the boundaries of the feasible region, or it can exist within the feasible region. Figure 7.6 demonstrates these two cases (the point P is the optimal solution in each case).

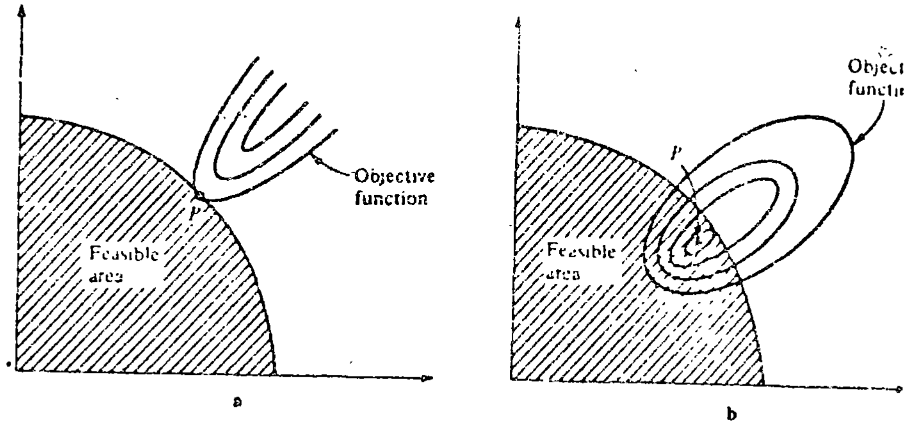


FIGURE 7.6

In general, we face ~~two major~~ difficulties in solving nonlinear-programming problems. First, because of the nonlinearity of the objective function and/or the constraints, it can be difficult in certain cases to distinguish between the local and the global solutions.[2] This, for example, would be the case when, in a maximization problem, the objective function is convex. Also, the same difficulty will arise when, in a maximization problem having a linear objective function, the region of feasible solutions is nonconvex (see Figure 7.7b). Second, it is sometimes difficult to test optimality in nonlinear-programming problems. The reason is that it is necessary to identify and evaluate all extreme points unless the functions involved are either strictly convex or strictly concave. This task becomes increasingly more difficult when we deal with higher-order (higher-degree) polynomials. Furthermore, a

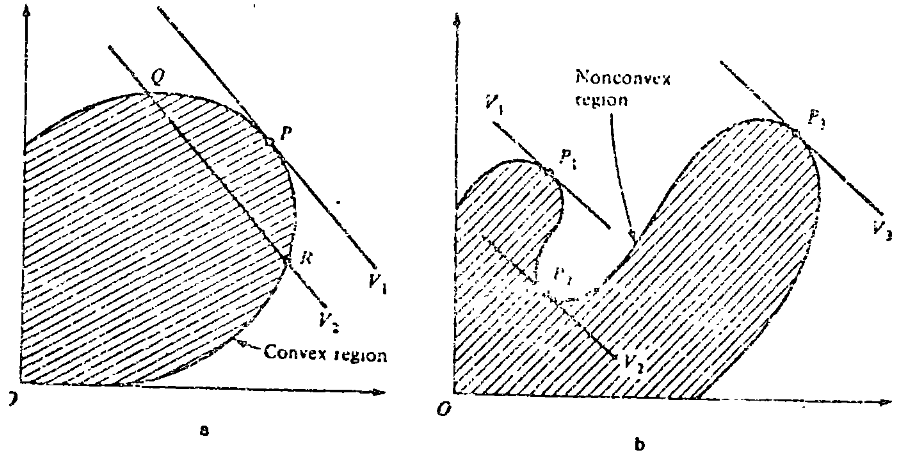


FIGURE 7.7

[2] See Appendix B.

characteristic of objective functions containing high-order variables is that small incremental changes in the variables can quite often result in large changes in the value of the function. Thus an attempt to "home in" on an extreme point can be exceedingly difficult.

b. Nonlinear Constraints with a Linear Objective Function

This category of nonlinear-programming problems gives rise to two types of situations: (1) a convex region of feasible solutions and a linear objective function (Figure 7.7a); and (2) a nonconvex region of feasible solution and a linear objective function (Figure 7.7b).

In the first case it is relatively easy to identify the optimal solution and then test it for optimality. For example, the optimal solution (maximization) in Figure 7.7a is given by the point P, which is a tangency point to the boundary. Any movement away from this point will reduce the value of the objective function. For example, if we move from point P to point Q or point R, the value of the objective function is decreased from V_1 to V_2.[3]

The situation in the case of a nonconvex region, however, is not so simple. Here, the point P_1, though a tangency point, is obviously a local maximum. In this case a movement along the boundary will bring us first to P_2 (where $V_2 < V_1$), but then we come to P_3 (where $V_3 > V_1$). This type of situation points up the difficulty in identifying true global extrema in nonlinear-programming problems.

In addition to the problem of finding extreme points, the problem of identifying local versus global extrema can also be difficult. The utilization of calculus to identify extrema (see Appendix B) becomes quite difficult if the number of variables contained in the functions becomes large and/or the functions contain variables of a high order. As the powers of the variables increase, there is a corresponding increase in the number of stationary points and thus the search for the set of possible points can become very difficult. Difficulties also arise in functions of the logarithmic and/or exponential form.

7.2.4 SOME PROPERTIES OF OPTIMAL SOLUTIONS

a. Economic Interpretation

A concave objective function, if it is a profit function, represents a situation involving diminishing returns, and if it were to represent a cost function it would be one involving increasing returns (decreasing marginal costs as output expands). Similarly, a convex objective function represents an increasing return profit function or a diminishing return cost function.

These properties lead to the following conclusions: In a nonlinear-

[illegible]

[3] Note that in Fi[illegible].7a V_2 is closer to the origin than V_1 and $V_2 < V_1$. In some cases, movement toward th[illegible]igin could very well yield a better solution.

programming problem with a convex feasible area one can apply the classical method (Appendix B) of locating extrema if, and only if, (1) the objective function is pseudoconcave or strictly concave (see Appendix C)—that is, diminishing returns—in the case of maximization, or (2) pseudoconvex or strictly convex (see Appendix C)—that is, increasing returns—in the case of minimization.

b. The Number of Variables in an Optimal Solution to a Nonlinear-Programming Problem

As the reader will recall, an important property of an optimal linear-programming solution is that the number of basic variables is always "equal to or less than" the number of nonredundant structural constraints. The "less than" case occurs when the optimal solution is degenerate.

In the optimal solution to a nonlinear-programming problem, the number of solution variables can be less than, equal to, or greater than the number of structural constraints. This property can also be explained by the fact that in nonlinear problems, optimal solutions can exist at points other than the "corner" points. For example, in Figure 7.6a we have two solution variables and only one constraint is fully utilized; that is, there is only one active constraint and two variables in the optimal solution. In Figure 7.6b there are two solution variables, and none of the constraints is restraining.

Thus, for nonlinear programming, unlike linear programming, we cannot make any general statements regarding the relationship between the number of solution variables and the number of structural cconstraints.

7.2.5 METHODS OF SOLUTION

As the reader will recall, there is no general efficient algorithm for the solution of nonlinear problems. However, for problems with certain identifiable structures efficient algorithms have been developed. In addition, it is often possible to transform the given nonlinear problem into one in which these structures become apparent. In Appendix D we present a method of solving nonlinear constrained optimization problems. A relatively simple method to handle such problems is presented in Chapter 8 (see Section 8.2.2). For more sophisticated methods the reader is referred to Bracken and McCormick [8], Fiacco and McCormick [18], Graves and Wolfe [21], Lasdon [35], Mangasarian [39], Wilde [56], and Zangwill [60].

7.3 QUADRATIC PROGRAMMING[4]

7.3.1 INTRODUCTION

Quadratic programming (QP) is the simplest nonlinear programming form. It deals with the problem of minimizing (maximizing) a qua[illegible]tic objective

[4] For an excellent treatment of quadratic programming, see Boot [71].

~~function subject to linear constraints~~. Many practical problems can be formulated as quadratic programs, and fairly efficient methods have been developed for solving such problems.

The necessary mathematical concepts for quadratic programming are given in Section 7.3.2. We shall give a brief introduction to these four specific types of quadratic programming methods: (1) Frank and Wolfe's, Dantzig's, and similar methods; (2) Beale's method; (3) Theil and Van de Panne's method, and (4) Lemke's method. Of these, only one method (Frank and Wolfe's) will be illustrated by a numerical example.

7.3.2 SOME MATHEMATICAL CONCEPTS FOR QUADRATIC PROGRAMMING

~~a. A Quadratic~~ Function

~~A quadratic function in the n scalar variables $x_1, x_2, \ldots, x_n$ is a polynomial function that contains terms of an order no higher than the second~~ (for example, for a single variable x, the highest order is x^2). The general form of such a function is

$$f(x_1, x_2, \ldots, x_n) = c_{11}x_1{}^2 + c_{22}x_2{}^2 + \cdots + c_{nn}x_n{}^2 + (c_{12}+c_{21})x_1x_2 + \cdots + (c_{n-1,n}+c_{n,n-1})x_{n-1}x_n + c_1x_1 + c_2x_2 + \cdots + c_nx_n + c_0 \qquad (7.2)$$

~~If we set $c_{ij} = c_{ji}$~~, we obtain a ~~symmetric function~~[5] and the $c_{ij}+c_{ji}$ term becomes $2c_{ij}$.

A ~~quadratic function in two variables can be~~ written as

$$~~f(x_1, x_2) = c_{11}x_1{}^2 + c_{22}x_2{}^2 + (c_{12}+c_{21})x_1x_2 + c_1x_1 +~~ c_2x_2 + c_0$$

An equivalent matrix form of the quadratic function is most useful and will be discussed later in detail.

Quadratic functions are well known in geometry. For example, the ~~circle~~, ~~the parabola, the hyperbola, and the ellipse (Figure 7.8) are all loci of special cases~~ (with some coefficients equal to zero in each special case) of a quadratic equation given in Equation (7.3).

$$~~Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0~~ \qquad (7.3)$$

where $A, B, \ldots$ are parameters equivalent to the c_{ij} of Equation (7.2).

~~Quadratic functions with three variables have the general form~~

$$f(x, y, z) = Ax^2 + By^2 + Cz^2 + Dyz + Exz + Fxy + Gx + Hy + Iz + K \qquad (7.4)$$

A~~gain, the well-known geometric forms of elliptic cylinder, ellipsoid, elliptic paraboloid, and hyperboloid are all defined by special cases of this general~~ equation. They ~~can be convex, concave, or neither (Figure 7.9). Two examples~~ are

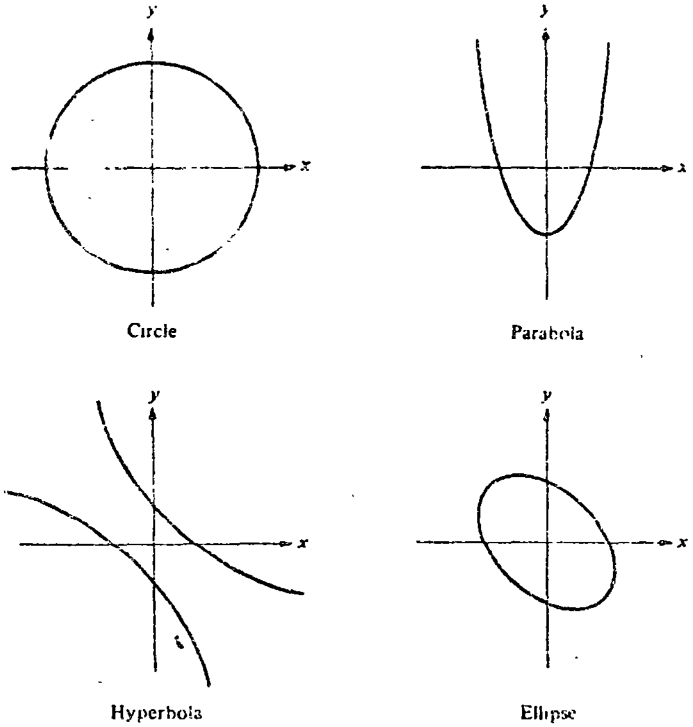


FIGURE 7.8

$$f(x_1, x_2) = 3x_1{}^2 - x_2{}^2 + x_1x_2 - 6x_1 + 6$$

$$f(x_1, x_2, x_3) = x_3{}^2 - 2x_1x_2 + x_1x_3 - 5x_2 + x_3$$

b. ~~A Quadratic~~ Form

A qua~~dratic form is a function that contains only second-order terms~~. Thus it is a special case of quadratic function; ~~its general form can be expressed~~ as

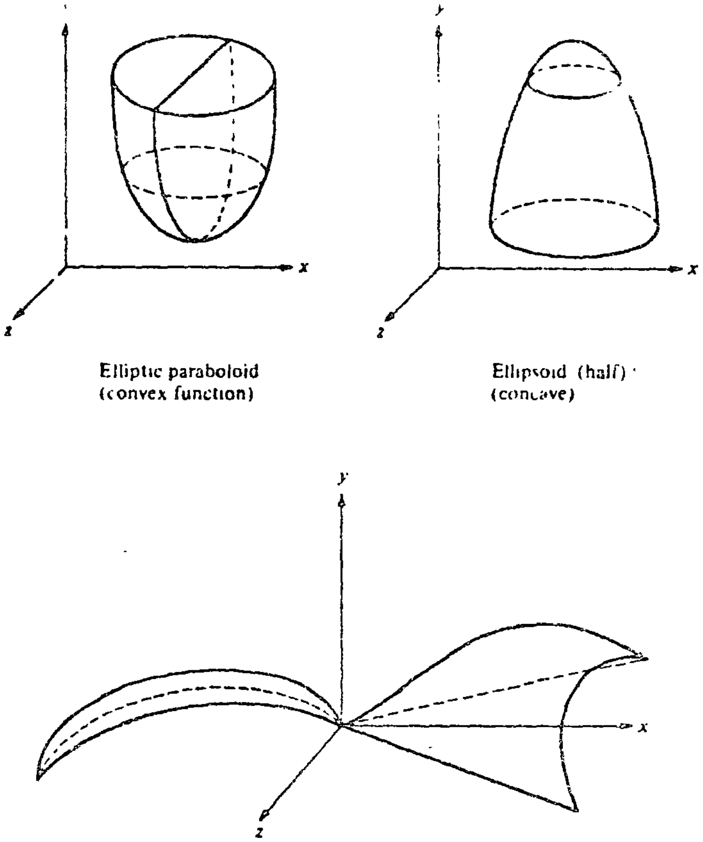
$$\begin{aligned} f(x_1, x_2, \ldots, x_n) = {} & (c_{11}x_1 + c_{12}x_2 + \cdots + c_{1n}x_n)x_1 \\ & + (c_{21}x_1 + c_{22}x_2 + \cdots + c_{2n}x_n)x_2 \\ & \cdot \\ & \cdot \\ & \cdot \\ & + (c_{n1}x_1 + c_{n2}x_2 + \cdots + c_{nn}x_n)x_n \end{aligned} \qquad (7.5)$$

T~~wo examples of quadratic forms are~~

$$f(x_1, x_2) = 2x_1{}^2 + 5x_2{}^2 - x_1x_2$$

$$f(x_1, x_2, x_3) = x_1{}^2 + 2x_1x_2 - x_1x_3 + x_3{}^2$$

[5] In a nonsymmetric function $c_{ij} \neq c_{ji}$. It can, however, be easily made symmetric by defining new coefficients c'_{ij} as: $c'_{ij} = \frac{1}{2}(c_{ij} + c_{ji})$.



Elliptic paraboloid
(convex function)

Ellipsoid (half)
(concave)

Hyperbolic paraboloid
(neither convex nor concave;
has a saddle point)

FIGURE 7.9

~~A quadratic form can be linearly transformed~~ to a sum (or ~~difference~~) of squares of ~~independent homogeneous linear expressions~~. For example,

$$\sout{5x_1^2+16x_1x_2+3x_2^2=(3x_1+2x_2)^2-(2x_1-x_2)^2}$$

This ~~property of a quadratic form leads to the following four types~~ of ~~quadratic functions: *positive definite, positive semidefinite, negative definite,*~~ and *~~negative~~ semidefinite*. We shall return to these functions.

~~c. Matrix Presentation of a Quadratic Form and a Quadratic Function~~

~~Every quadratic form can be expressed as a product of a symmetric~~ matrix (made ~~up of its coef~~ficients) and ~~a given vector representation of the variables~~

x_j. Consider, for example, the general two-variable symmetric quad form:

$$f(x_1, x_2)=c_{11}x_1^2+c_{22}x_2^2+2c_{12}x_1x_2$$

Let us define

$$X=\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

and a 2 × 2 symmetric matrix C_1 as

$$C_1=\begin{bmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{bmatrix}$$

Then the ~~quadratic form $f(x_1, x_2)$ can be expressed~~ as

$$f(x_1, x_2)=[x_1 \;\; x_2] \begin{bmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\sout{= X^T C_1 X} \qquad (7.$$

Similarly, any polynomial of the second degree (order) with n terms ca be expressed in a matrix form as follows:

$$f(x)=C^TX+\tfrac{1}{2}X^TC_1X+c_0 \qquad (7.$$

where

$$X=\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}; \qquad C=\begin{bmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{bmatrix}$$

X^T and C^T are the transposes of X and C, and ~~C_1 is a symmetric $n \times n$ matrix~~ (~~that is, $c_{ij}=$~~c_{ji}), which can be expressed as

$$C_1=\begin{bmatrix} c_{11} & \dots & c_{1n} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ c_{n1} & \dots & c_{nn} \end{bmatrix}$$

Examples:

~~(1) $f(x)=4x_1+2x_2+3x_1x_2-x_1^2-2x_2^2$~~

$$=[x_1 \;\; x_2] \begin{bmatrix} -1 & 1.5 \\ 1.5 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+[4 \;\; 2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

~~The reader should note that $2c_{12}=3$, and therefore $c_{12}=1.5$.~~

(2) $f(x) = 2x_1^2 + 3x_2^2 - x_3^2 + 4x_1x_2 - 5x_1x_3 + 8x_2x_3 + 7x_1 - 9x_2 + x_3 + 6$

$$= [x_1 \quad x_2 \quad x_3] \begin{bmatrix} 2 & 2 & -2.5 \\ 2 & 3 & 4 \\ -2.5 & 4 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + [7 \quad -9 \quad 1] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + 6$$

where

$$\begin{array}{ll} 2c_{12} = 4 & c_{12} = 2 \\ 2c_{13} = -5 & c_{13} = -2.5 \\ 2c_{23} = 8 & c_{23} = 4 \\ c_{11} = 2 & c_1 = 7 \\ c_{22} = 3 & c_2 = -9 \\ c_{33} = -1 & c_3 = 1 \end{array}$$

d. Types of Quadratic Functions

Positive Definite Function

If we have an n-variable quadratic form and we can reduce the function into squares, all squared terms preceded by a positive sign, then the function will always be non-negative. If it is positive for all $x_j \neq 0$, the function is *positive definite*. If it is equal to zero with some x_j not equal to zero, the function is *positive semidefinite*. (See Figure 7.10a for a function of one variable.)

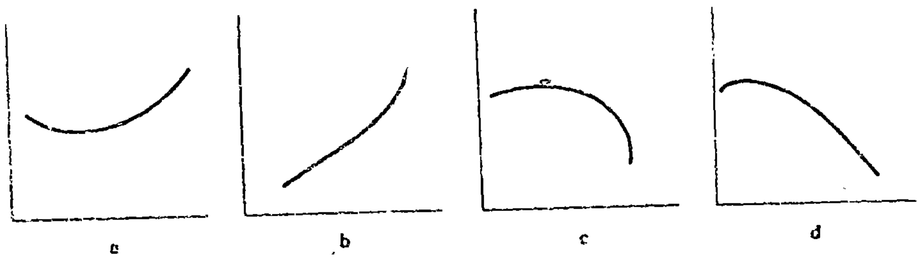


FIGURE 7.10

Stated mathematically, a function is positive definite if

$$X^T C_1 X > 0 \qquad \text{for all } X \neq 0 \tag{7.8}$$

where C_1 is a square symmetric matrix. A positive definite function is always *strictly convex*.

Example: Consider the function

$$f(x_1, x_2) = 2x_1^2 + 5x_2^2 + 6x_1x_2$$

Since this function can be expressed as

$$f(x_1, x_2) = (x_1 + x_2)^2 + (x_1 + 2x_2)^2$$

it is positive definite according to the definition. Note that the only values of x_1, x_2 which make $x_1 + x_2$ and $x_1 + 2x_2$ simultaneously vanish are $x_1 = 0, x_2 = 0$.

Positive Semidefinite Function (Figure 7.10b)

If we can reduce an n-variable quadratic form into squares, all squared terms preceded by a positive sign, and the function can be equal to zero for some x_j not equal to zero, it is a *positive semidefinite function*. The value of such a function is either positive or zero.

Stated formally, the function is positive semidefinite if

$$X^T C_1 X \geq 0, \qquad \text{for all } X \tag{7.9}$$

Example:

$$f(x_1, x_2, x_3) = 2x_1^2 + 5x_2^2 + x_3^2 + 6x_1x_2 + 2x_1x_3 + 2x_2x_3$$
$$= (x_1 + 2x_2)^2 + (x_1 + x_2 + x_3)^2$$

This function, with three variables, has been reduced to a sum of squares, all with positive coefficients. Further, for $x_1 = 2$, $x_2 = -1$, $x_3 = -1$, the function is zero and hence, by definition, it is positive semidefinite, and is convex.

Negative Definite Function (Figure 7.10c)

A function $f(x)$ is negative definite if $-f(x)$ is positive definite. A negative definite function is always strictly concave.

Example:

$$f(x_1, x_2) = -2x_1^2 - 10x_2^2 - 4x_1x_2$$
$$= -(x_1 - x_2)^2 - (x_1 + 3x_2)^2$$

This function, according to our definition, is negative definite, and is strictly concave everywhere.

Negative Semidefinite Function (Figure 7.10d)

A function $f(x)$ is negative semidefinite if $-f(x)$ is positive semidefinite.

Example:

$$f(x_1, x_2, x_3) = -2x_1^2 - 5x_2^2 - x_3^2 + 2x_1x_2 + 2x_1x_3 - 4x_2x_3$$
$$= -(x_1 + x_2)^2 - (x_1 - 2x_2 - x_3)^2$$

This function, according to our definition, is negative semidefinite and is concave. Note that $f(x) = 0$ when $x_1 = 1$, $x_2 = -1$, $x_3 = 3$.

e. Tests of Convexity

1. An easy test of convexity for quadratic functions is to check the values of the determinants of the matrix C_1 of the quadratic form part of the function[6]

[6] The regular test of convexity (discussed in Appendix C), which is more complicated, can be used for cases not covered by this method. Appendix C also discusses the test of convexity around a stationary print.

Let the first determinant be $D_1 = c_{11}$

Let the second determinant be $D_2 = \begin{vmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{vmatrix}$

Let the third determinant be $D_3 = \begin{vmatrix} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & c_{23} \\ c_{13} & c_{23} & c_{33} \end{vmatrix}$

and so on. If the signs of the determinants are all positive, that is, if

$$D_1 > 0 \quad D_2 > 0 \quad \cdots \quad D_n > 0 \tag{7.10}$$

then the function is strictly convex. If, starting with a negative sign, the signs of the determinants alternate, that is,

$$D_1 < 0 \quad D_2 > 0 \quad D_3 < 0 \quad \cdots \quad D_n(-1)^n > 0 \tag{7.11}$$

Then the function is strictly concave.

Example 1:

$$f(x) = 3x_1 - 3.5x_2 - \frac{x_1^2}{1000} - \frac{x_2^2}{400}$$

$$= [x_1 \; x_2] \begin{bmatrix} \frac{-1}{1000} & 0 \\ 0 & \frac{-1}{400} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [3 \; -3.5] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$D_1 = c_{11} = \frac{-1}{1000}; \text{ hence } D_1 < 0.$$

$$D_2 = \begin{vmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{vmatrix} = \begin{vmatrix} \frac{-1}{1000} & 0 \\ 0 & \frac{-1}{400} \end{vmatrix} = \frac{1}{400{,}000}; \text{ hence } D_2 > 0.$$

The signs alternate (starting from a negative sign), and therefore the function is strictly concave and has a global maximum.

Example 2:

$$f(x) = 2x_1^2 + x_2^2$$

$$= [x_1 \; x_2] \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$D_1 = c_{11} = 2; \text{ hence } D_1 > 0.$$

$$D_2 = \begin{vmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{vmatrix} = \begin{vmatrix} 2 & 0 \\ 0 & 1 \end{vmatrix} = 2; \text{ hence } D_2 > 0.$$

All signs are positive; hence the function is strictly convex and has a global minimum.

2. Another test of convexity exists for all two-variable quadratic functions (second-degree polynomial) of the form

$$f(x, y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F \tag{7.12}$$

If $4AC - B^2 > 0$, and $A > 0$ (or $C > 0$), the function is strictly convex.
If $4AC - B^2 > 0$, and $A < 0$ (or $C < 0$), the function is strictly concave.
If $4AC - B^2 < 0$, the function is neither strictly convex nor strictly concave.

Example 1:

$$f(x) = 3x^2 + 4y^2 - 6xy - x + 2y$$

Here

$$A = 3 \quad B = -6 \quad C = 4 \quad D = -1 \quad E = 2$$

$$4AC - B^2 = 4 \times 3 \times 4 - 36 = 48 - 36 > 0 \quad \text{and} \quad A > 0$$

Therefore, the function is strictly convex. Notice that convexity (concavity) is independent of the first-degree parts. Notice also that the sign of the coefficient B of the product x_1x_2 is not relevant since the term is squared in the test.

Example 2:

$$f(x) = 2x_1 - 3x_2 - 5x_1^2 + 3x_2^2 + 4x_1x_2$$

Here

$$4AC - B^2 = -4 \times 5 \times 3 - 16 = -76 < 0$$

The function is neither strictly convex nor strictly concave.

f. Mathematical Presentation of Quadratic Programming

A quadratic-programming problem can be stated in various forms. In Equation (7.7) we presented the objective function in a matrix form. We now give two different presentations of the objective function as well as the constraints.

1.
$$\max f(x) = C^T X + \tfrac{1}{2} X^T C_1 X \tag{7.13}$$

s/t

$$AX \leq B$$

$$X \geq 0$$

where B is an $m \times 1$ column vector of scalars b_i ($i = 1, 2, \ldots, m$) and A is an $m \times n$ matrix of the coefficients a_{ij}.

2. An alternate but equivalent mathematical statement of the general concave[7] quadratic-programming problem is:

$$\max f(x) = \sum_{j=1}^{n} c_j x_j + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_i x_j \qquad (7.14)$$

s/t

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i = 1, 2, \ldots, m$$

and

$$x_j \geq 0 \qquad j = 1, 2, \ldots, n$$

7.3.3 QUADRATIC-PROGRAMMING METHODS

Of all the methods of solving quadratic-programming problems, four types of methods were found to be of special interest[8] (and efficiency). They are (a) Frank and Wolfe's, Dantzig's, and similar methods; (b) Beale's method; (c) Theil and Van de Panne's method; and (d) Lemke's method.

Since it is beyond the scope of this text to discuss each type in depth, we shall present only some major elements of these four types and illustrate one method (Frank and Wolfe's) with a concrete example. For further details, the interested reader is referred to Boot [7], and to Kunzi *et al* [33].

a. Frank and Wolfe's, Dantzig's, and Similar Methods

These methods are based on Kuhn and Tucker's generalizations of the concept of Lagrange multipliers (Appendix D). They modify the quadratic-programming problem so that the simplex method can be used. An example of Frank and Wolfe's method (which falls in this category) will be given in Section 7.3.4. For several other variations, see Wolfe [58], Dantzig [10], and Barankin and Dorfman [4].

b. Beale's Method[9]

This method, which also makes use of the simplex algorithm, converts the inequality constraints to equations, and starts by finding an initial feasible solution for the constraint equations.[10] The next step is an attempt to improve the value of the objective function. With the aid of partial derivatives[11] and auxiliary variables, it is possible to check whether the introduction of a nonbasic variable will improve the value of the objective function. It is also possible to show that a final solution can be obtained within a finite number of iterations.

Houthakker [28] developed a special method, called the *capacity method*, using parametric programming. This method complements Beale's method. It can handle cases in which all coefficients of the constraints are non-negative.

c. Theil and Van de Panne's Method [52]

In this method, also based on the Kuhn-Tucker conditions, a systematic search is made for those solutions that lie on the boundaries of structural and non-negativity constraints. The unconstrained problem is solved first; then the constraints are added, one at a time, until a solution is found that satisfies all of the constraints. By devising ingenious rules, the authors were able to limit the search to a small number of all possible solution combinations.

d. Lemke's Method [37]

In Lemke's method, which is related to his dual-simplex idea (Chapter 4), we solve the problem by solving the dual to the quadratic programming problem. The algorithm suggested by Lemke is similar to the one proposed by Beale.

e. Comparison of the Algorithms for Quadratic Programs[12]

Wolfe's method has the advantage that it can be easily generalized to include positive semidefinite objective functions. The other three methods are restricted to positive definite quadratic forms.

Both Wolfe's and Beale's methods appear to require approximately the same number of computations. Wolfe's method, in general, requires more iterations than Beale's, but each iteration involves fewer computations.

The methods developed by Theil and Van de Panne and by Lemke will be preferable to the others if the solution lies on relatively few of the constraining hyperplanes. In fact, if the solution lies in the interior (an unconstrained

[7] The objective function is concave if

$$\Sigma \ \Sigma c_{ij} x_i x_j \geq 0$$

for all values of x. At present there is no suitable method for solving the general quadratic program with an arbitrary symmetric C_1 matrix. Consequently, we will restrict ourselves to a concave objective function for which the quadratic form is negative semidefinite (in maximization). This requirement will always be assumed for the remainder of this chapter, whether or not it is so stated.

[8] The discussion in this section is based mainly on Dorn's survey [14], pp. 181–196. Several computational examples are given in that article. For other methods see Abadie [1], Zangwill [60], and Kuhn [31].

[9] For an example of Beale's method see Vajda [53], pp. 223–231; see also Beale [5].

[10] This step in quadratic programming is no different from finding the first feasible solution in linear programming, for the first feasible solution is usually determined by the constraint equations alone, without considering the nature of the objective function.

[11] The reader will recall that in linear programming the improvement in the objective function just after the *initial* solution was sought by checking the c_j (coefficients of the objective function). Note that these coefficients are, in fact, the partial derivatives of the objective function. In quadratic programming, we also make use of these partial derivatives

[12] Source: Dorn [14].

minimum) both the Theil and Van de Panne and the Lemke processes are identical and converge in one step.

Lemke's method does not require finding an initial primal feasible solution, but it does require the inverse of the C_1 matrix. Beale and Wolfe, on the other hand, do require an initial primal feasible solution but do not need C_1^{-1}. The computations involved in finding a primal feasible solution are approximately equivalent to finding C_1^{-1}, and on this basis there can be no choice between the three methods.

Any further comparisons do not seem possible at this time. The advantages of each of the methods seem to balance each other and leave little or no room for choice. The relative merits of the four methods will become apparent through extensive computational experience.

Finally, it should be noted that Wolfe has modified Kelley's cutting-plane method [30] for more general nonlinear programs so that it too becomes finite for quadratic objective functions.

7.3.4 FRANK AND WOLFE'S, DANTZIG'S, AND SIMILAR METHODS

We have chosen to discuss in detail this family of methods mainly because they utilize the simplex algorithm.

The methods in this family include Frank and Wolfe [19], Wolfe [58], Dantzig [10], and Barankin and Dorfman [4]. A complete guide to this family may be found in Boot [7], Chapter 9.

This class and related methods, based on the Kuhn-Tucker conditions, are capable of minimizing convex or strictly convex functions, or maximizing concave or strictly concave functions.

To provide a numerical illustration, let us apply *Frank and Wolfe's method* to the following product-mix problem:[13]

$$\max z = 3x_1 + 3.5x_2 - \frac{x_1^2}{1000} - \frac{x_2^2}{400}$$

s/t

$$\begin{aligned} x_1 + 2x_2 &\leq 4000 \\ 4x_1 + 3x_2 &\leq 12{,}000 \\ x_1, \quad x_2 &\geq 0 \end{aligned}$$

Solution, Step 1: Determine the Nature of the Objective Function

Since the method can handle only semidefinite or definite quadratic functions, it is necessary to find out whether our function meets these requirements. Using the test proposed in (7.3.2e) we get

$$4AC - B^2 = 4\left(\frac{1}{1000} \times \frac{1}{400}\right) - 0 = \frac{1}{100{,}000}; \text{ that is } > 0$$

and

$$A = \frac{-1}{1000}; \text{ that is } < 0$$

The function is strictly concave (negative definite) and thus suitable fo maximization by Frank and Wolfe's method.

Solution, Step 2: Write the Problem in Matrix Form

The objective function may be stated as

$$\max f(x) = CX + X^T C_1 X$$

In our case,

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}; \qquad X^T = [x_1 \; x_2]$$

Thus

$$C = [3 \;\; 3.5] \quad \text{and} \quad C^T = \begin{bmatrix} 3 \\ 3.5 \end{bmatrix}$$

$$C_1 = \begin{bmatrix} \frac{-1}{1000} & 0 \\ 0 & \frac{-1}{400} \end{bmatrix}$$

The objective function can be rewritten as

$$f(x) = [3 \quad 3.5] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [x_1 \quad x_2] \begin{bmatrix} \frac{-1}{1000} & 0 \\ 0 & \frac{-1}{400} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

The constraints are written as

$$AX \leq B$$

In our case,

$$A = \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix} \quad \text{and} \quad A^T = \begin{bmatrix} 1 & 4 \\ 2 & 3 \end{bmatrix}$$

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{and} \quad X \geq 0$$

$$B = \begin{bmatrix} 4000 \\ 12{,}000 \end{bmatrix}$$

[13] This problen [illegible] also be solved by classical calculus (Appendix B) and by the Lagrange-multipliers method (Appendix D).

Solution, Step 3: Transform the Original Problem into a Linear-Programming Form[14]

This is done by introducing non-negative slack variables, one for each inequality constraint and then by using auxiliary variables. The transformed problem has $2(m+n)$ variables and $(m+n)$ constraints, and the following general form:

$$\max z=-\tfrac{1}{2}\overset{*}{Z}Z \tag{7.15}$$

s/t

$$TZ=D \quad \text{and} \quad Z\geq 0$$

where

$$T=\begin{bmatrix} A & I_m & O_m & O_n \\ -2C & O_{nm} & A^T & -I_n \end{bmatrix}$$

$$D=\begin{bmatrix} B \\ C_n^T \end{bmatrix} \quad Z=\begin{bmatrix} X \\ S_x \\ U^T \\ S_u^T \end{bmatrix} \quad \text{and} \quad \overset{*}{Z}=[S_u \;\; U \;\; S_x^T \;\; X^T]$$

In these equations,

S_x = non-negative slack vector for original problem ($m\times 1$ column vector)
U = the dual variables ($m\times 1$ column vector)
U^T = dual's transpose ($1\times m$ row vector)
S_u^T = slack vector for dual (transposed) ($n\times 1$ column vector)
I_m = $m\times m$ identity matrix
I_n = $n\times n$ identity matrix
O_m = $m\times m$ zero matrix
O_n = $n\times n$ zero matrix
O_{nm} = $n\times m$ zero matrix

If we write our constraints $TZ=D$ according to the transformation, we get

$$\begin{bmatrix} A & I_m & O_m & O_n \\ -2C & O_{nm} & A^T & -I_n \end{bmatrix} \begin{bmatrix} X \\ S_x \\ U^T \\ S_u^T \end{bmatrix}=D$$

Inserting the data of our problem, we get

[14] The original problem is in a linear-programming form except that the objective function is quadratic. This transformation is based on the Kuhn-Tucker conditions (Appendix D). $\overset{*}{Z}$ in (7.15) is not the transpose of Z. It is instead a row vector whose elements are obtained, in each step, from the solution at hand.

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 1/500 & 0 \\ 0 & 1/200 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 4 \\ 2 & 3 \end{bmatrix} & \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ u_1 \\ u_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 4000 \\ 12{,}000 \\ 3 \\ 3.5 \end{bmatrix}$$

Solution, Step 4: Write the Constraints in Their Explicit Forms

$$\begin{aligned} x_1+2x_2+s_1 &= 4000 \\ 4x_1+3x_2+s_2 &= 12{,}000 \\ \frac{x_1}{500} \qquad + u_1+4u_2-s_3 &= 3 \\ \frac{x_2}{200} \quad +2u_1+3u_2-s_4 &= 3.5 \end{aligned}$$

These data were derived from the $TZ=D$ relation after the pertinent data for our problem were inserted.

Solution, Step 5: Find a Feasible Solution

By examination, the following feasible solution is found: $s_1=4000$, $s_2=12{,}000$, $u_1=3$, and $s_4=2.5$, and all other variables are equal to zero. These values now become the components of $\overset{*}{Z}$.

$$\overset{*}{Z}=[s_3\; s_4\; u_1\; u_2\; s_1\; s_2\; x_1\; x_2]=[0\;\; 2.5\;\; 3\;\; 0\;\; 4000\;\; 12{,}000\;\; 0\;\; 0]$$

Solution, Step 6: Transform into a Linear Objective Function

A new objective function is written

$$\max z=-\frac{1}{2}\overset{*}{Z}Z \tag{7.16}$$

In our case,

$$-\frac{1}{2}\overset{*}{Z}Z=-\frac{1}{2}[0\;\; 2.5\;\; 3\;\; 0\;\; 4000\;\; 12{,}000\;\; 0\;\; 0] \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ u_1 \\ u_2 \\ s_3 \\ s_4 \end{bmatrix}$$

$$=-\frac{5}{4}x_2-\frac{3}{2}s_1-2000u_1-6000u_2$$

This objective function is to be maximized subject to the constraints in step 4. This is now a regular linear-programming problem.

Solution, Step 7: Use the Simplex Method to Obtain Progressively Better Solutions

Now we continue to find solutions $\overset{*}{Z}_1$, $\overset{*}{Z}_2$, ..., until a $\overset{*}{Z}_i$ is found that satisfies the following condition:

$$\overset{*}{Z}_i Z_i = 0 \tag{7.17}$$

which indicates an optimal solution to the quadratic-programming problem. Note: It may be necessary in certain cases to follow a second phase in this algorithm. For details, see Frank and Wolfe [19].

It should be noted that, according to this method, successive $\overset{*}{Z}_i$ might yield different functions as calculated by (7.15). In our example, we start the algorithm with the objective function (obtained in step 5) and the constraints (from step 4). The reader can verify that, after three iterations, test of optimality ($\overset{*}{Z}_i Z=0$) is satisfied. The optimal solution, in terms of real variables, turns out to be $x_1=1500$, $x_2=700$.

7.4 SEPARABLE PROGRAMMING

7.4.1 SEPARABLE (CONVEX) PROGRAMMING

Separable (convex) programming is a technique for obtaining an approximate solution to some nonlinear problems through the device of "separating" the objective function terms into several single-variable functions. The basic idea of the technique, as developed by Charnes and Lemke [9], is to transform the nonlinear problem into an approximately equivalent linear program. The solution to the approximate linear program provides an approximate[15] solution to the original problem.

The separable programming problem may be expressed as:

$$\min f(x) = \sum_{j=1}^{n} f_j(x_j) \tag{7.18}$$

s/t

$$g_i(x) = \sum_{j=1}^{n} g_{ij}(x_j) \geq 0 \qquad i=1, \ldots, m. \tag{7.19}$$

The objective function is a *separable function*, as are the constraints. A separable function can be expressed as a linear combination of several single-variable functions; that is, it is a function that can be written as a sum of [illegible] functions, each of which includes a single variable.

In general, a separable function $f(x)$ is

$$f(x) = \sum_{j=1}^{n} f_j(x_j) = f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n) \tag{7.20}$$

where $0 \leq x_j \leq h_j$, and h_j is an upper bound.

Example:

$$f(x_1, x_2) = 2x_1^3 - x_2^2 + 3x_1 + 5x_2$$

This function can be written as equal to

$$f_1(x_1) + f_2(x_2) = (2x_1^3 + 3x_1) - (x_2^2 - 5x_2)$$

Essentially, the variables of a separable function are coupled together only in an additive fashion.

The first step is to write the objective function as a sum of several specific functions $f_j(x_j)$. Each $f_j(x_j)$ is then approximated by a linear function $f_i(x_i)$. (See Figure 7.11.) The problem can then be solved, after proper transformation, as a linear-programming problem. Further, since the curves $f_j(x_j)$ are all convex (in the minimization case) or concave (in the maximization case), the problem can be solved for a unique solution.

Several functions that do not seem to be separable can be separated after proper transformation (such as $x_1 x_2$, $e^{x_1+x_2}$, and $x_1^{x_2}$).

Separable objective functions often exist in practice. For example, if we

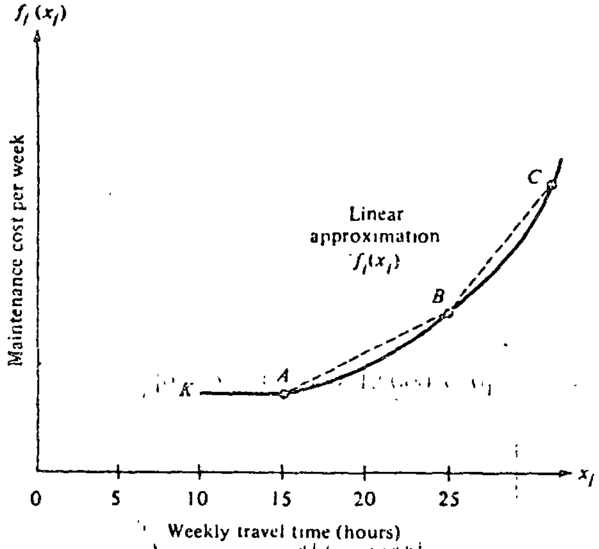


FIGURE 7.11

[15] The error of the approximation can be estimated; see Abadie [1], p. 180.

check the total maintenance cost of an automobile, we find that as long as the car is used for regular trips to work and for shopping, the costs are almost constant (see Figure 7.11, point K to point A). But for those weeks when we take extra out-of-town trips, the maintenance cost rises rapidly (from point A to B). Also if we use the car more than 25 hours per week, the maintenance cost will increase even faster (from B to C).

We shall discuss here two basic types of convex separable programming:

✓ 1. Nonlinear objective function with linear constraints (Section 7.4.2).
✓ 2. Piecewise linear convex objective function with linear constraints (Section 7.4.3)

Since a third type—namely, nonlinear separable constraints that form a convex region—will not be discussed here the interested reader is referred to Hartley [24]; Hadley [22], Chapter 4; and Miller [43].

Note: Theoretically, any separable problem can be solved with the aid of mixed-integer programming. The computation feasibility of the procedure is highly questionable, however, since there are few efficient computer programs for solving very large mixed-integer programming problems. Thus we are limited in practice to separable convex programming, with these three basic requirements:

1. The feasible area is convex.
2. The objective function is separable.
3. The objective function is convex in the case of minimization (concave in the case of maximization).

7.4.2 NONLINEAR OBJECTIVE FUNCTION WITH LINEAR CONSTRAINTS—AN ILLUSTRATIVE EXAMPLE

$$\min z = x_1^2 + x_2^2 - 6x_1 + 4x_2$$

s/t

$$\begin{aligned} x_1 + 2x_2 &\geq 10 \\ x_1 + x_2 &\leq 12 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Solve by separable programming.

Solution, Step I: Write the Objective Function as a Sum of Separable Terms, Each of Which Involves a Single Variable

$$z = x_1^2 + x_2^2 - 6x_1 + 4x_2 = f_1(x_1) + f_2(x_2) = (x_1^2 - 6x_1) + (x_2^2 + 4x_2)$$

Solution, Step II: Test for Convexity for $f_j(x_j)$ in the Area $x_j \geq 0$

(a) For $f_1(x_1) = x_1^2 - 6x_1$,

$$f' = \frac{\partial f}{\partial x_1} = 2x_1 - 6; \qquad f'' = \frac{\partial^2 f}{\partial x_1^2} = 2$$

The first and second derivatives are continuous, and $f'' > 0$. Thus, the function $f_1(x_1)$ is convex everywhere.

(b) For $f_2(x_2) = x_2^2 + 4x_2$,

$$\frac{\partial f}{\partial x_2} = 2x_2 + 4; \qquad \frac{\partial^2 f}{\partial x_2^2} = 2$$

Again, the first and second derivatives are continuous, and $f'' > 0$. Hence the function $f_2(x_2)$ is convex everywhere. *Note:* In maximization cases all $f_j(x_j)$ must be concave.

Solution, Step III: Break Each $f_j(x_j)$ into Separate Parts, Each Part to Be Approximated by a Linear Segment

The decision where to break the functions is an important one; the smaller the segments, the better the approximation, but the longer the required computation. The segments do not have to be equal; the closer some parts of the function are to being linear, the larger the segments can be in these parts. In our case we decided to break the function at integer points.

Solution, Step IV: Compute the Coordinates of the Break Points

For $f_1(x_1)$,

x_1	0	1	2	3	4	5	6	7	8	9	10	11	12
$f_1(x_1)$	0	−5	−8	−9	−8	−5	0	7	16	27	40	55	72

For $f_2(x_2)$,

x_2	0	1	2	3	4	5	6	7	8	9	10	11	12
$f_2(x_2)$	0	5	12	21	32	45	60	77	96	117	140	165	192

In building the approximation we have to make an additional decision: What is the upper limit on each variable? In our case, since we have a constraint, $x_1 + x_2 \leq 12$, we will compute the value of $f_j(x_j)$ to the point $x_j = 12$.

In addition, a glance at the curves (Figure 7.12) indicates that there is no sense in computing values right up to the upper limit, since the variable x_1 starts to contribute increasing marginal costs from the point $x_1 = 3$. Similarly, the variable x_2 starts to contribute increasing marginal costs from the point $x_2 = 0$. Thus, in view of the constraint $x_1 + x_2 \leq 12$, we should consider the variable x_1 up to $x_1 = 3$ and not consider variable x_2 at all. But, since we have an additional constraint, $x_1 + 2x_2 \geq 10$, we have to check x_1 up to point $x_1 = 10$ and x_2 up to point $x_2 = 5$.

In practical cases, such an analysis may save considerable computational work.

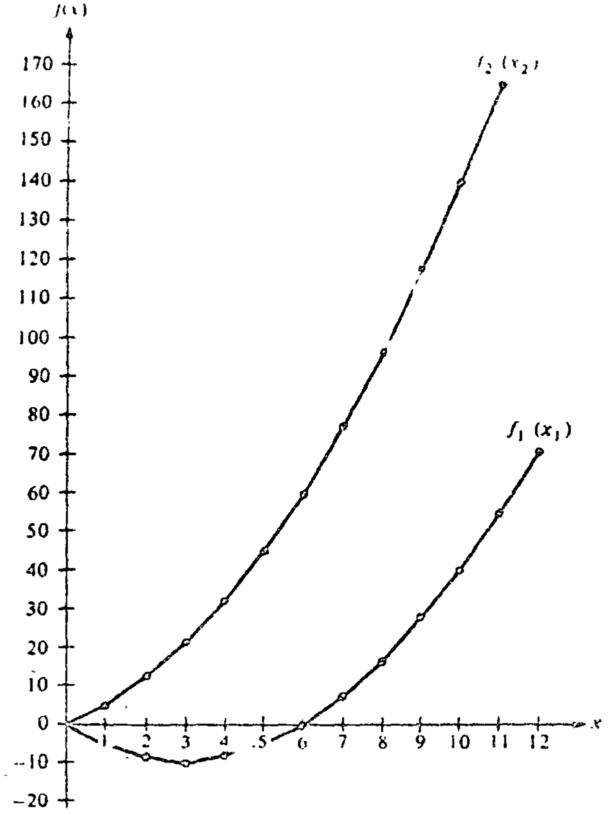


FIGURE 7.12

In this case, for purpose of demonstration we shall compute all points to the upper limit.

Solution, Step V: Decompose x_j into Auxiliary Variables x_{jm}

This step is necessary in order to write the separated or piecewise linear functions as an ordinary linear function. The subscript m corresponds to the total number of pieces into which the functions have been separated. In our case,

$$m = 12$$

The auxiliary variables, x_{jm}, are defined[16] in such a manner that

$$x_j = x_{j1} + x_{j2} + \cdots + x_{jm} \tag{7.21}$$

[16] A consequence of this definition for each x_j is that the value for $f_j(x_j)$ can be approximated by multiplying the auxiliary variables $x_{j1}, x_{j2}, \ldots, x_{jm}$ by their respective slopes. A eview of the basic definition of the term "slope" will support this argument. This argument s the rationale for step VI.

In our case, from Figure 7.12,

$$x_1 = x_{11} + x_{12} + \cdots + x_{1,12}$$
$$x_2 = x_{21} + x_{22} + \cdots + x_{2,12}$$

Note again, we have decomposed here up to the upper limit. Also note that the auxiliary variables are linear *segments*.

Solution, Step VI: Write the Equivalent Linear Objective Functions That Approximate $f_j(x_j)$

In order to write the equivalent functions, we need to compute the coefficients of the auxiliary variables x_{jm}. These coefficients are the *slopes* of the segments into which each separated or piecewise linear function has been divided. We distinguish two cases:

(a) In the *positive-slope case* (Figure 7.13) the slope is given by d_1/d_2.

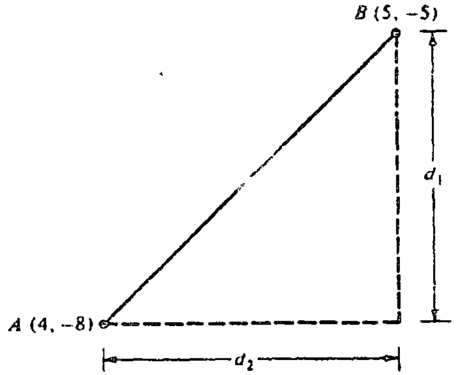


FIGURE 7.13

The distances d_1 and d_2 can easily be computed by subtracting the coordinates of A from the coordinates of B. For example, the slope associated with the auxiliary variable x_{15} is computed as follows: Since we have point A (4, −8) and point B (5, −5), we obtain

$$d_1 = -5 - (-8) = 3 \quad d_2 = 5 - 4 = 1$$

Therefore,

$$\text{Slope} = \frac{d_1}{d_2} = \frac{3}{1} = 3$$

(b) In the *negative-slope case* (Figure 7.14) the slope is given also by d_1/d_2 and we subtract the coordinates of C from those of D. For example, for the segment x_{12}, we have points $C(1, -5)$ and $D(2, -8)$, from which we obtain

$$d_1 = -8 - (-5) = -3 \quad d_2 = 2 - 1 = 1$$

Therefore,

$$\text{Slope}=\frac{d_1}{d_2}=\frac{-3}{1}=-3$$

Note that the coordinates of all points, such as $A, B, C, D, \ldots$, are obtained from solution step IV, in which x_1, $f_1(x_1)$, x_2, and $f_2(x_2)$ values are computed at the break points.

Similarly, we can compute all slopes and write the two separate linear objective functions:

$$f_1(x_1)_{\text{linear}}=-5x_{11}-3x_{12}-x_{13}+x_{14}+3x_{15}+5x_{16}+7x_{17}$$
$$+9x_{18}+11x_{19}+13x_{1,10}+15x_{1,11}+17x_{1,12}$$

and

$$f_2(x_2)_{\text{linear}}=5x_{21}+7x_{22}+9x_{23}+11x_{24}+13x_{25}+15x_{26}$$
$$+17x_{27}+19x_{28}+21x_{29}+23x_{2,10}+25x_{2,11}+27x_{2,12}$$

Solution, Step VII: Write the Equivalent Constraints

Utilizing step V, we write the constraints in terms of auxiliary variables.

Constraint 1:

$$(x_{11}+x_{12}+\cdots+x_{1,12})+2(x_{21}+x_{22}+\cdots+x_{2,12})\geq 10$$

Constraint 2:

$$(x_{11}+x_{12}+\cdots+x_{1,12})+(x_{21}+x_{22}+\cdots+x_{2,12})\leq 12$$

The other constraints are

$$0\leq x_{11}\leq 1$$
$$0\leq x_{12}\leq 1$$
$$\vdots$$
$$0\leq x_{1,12}\leq 1$$
$$0\leq x_{21}\leq 1$$
$$\vdots$$
$$0\leq x_{2,12}\leq 1$$

The upper limit (in this case, 1) on the auxiliary variables is the result of the breaking-point decision (integral in this case). As mentioned earlier, the segments do not have to be either equal or integral.

Solution, Step VIII: Solve the Problem as a Regular Linear Program

Now the problem has been reduced to a regular linear-programming problem. In Table 7.1 we enter the initial solution.

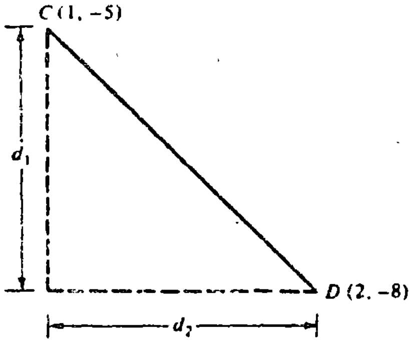


FIGURE 7.14

This formulation is especially appropriate to the upper-bound technique (see Section 4.5.3 and Dantzig [11]).

The optimal solution is:

$$\hat{x}_{11}=1\quad \hat{x}_{12}=1\quad \hat{x}_{13}=1\quad \hat{x}_{14}=1$$
$$\hat{x}_{15}=1\quad \hat{x}_{21}=1\quad \hat{x}_{22}=1\quad \hat{x}_{23}=0.5$$

Solution, Step IX: Transform to Original Variables

$$\hat{x}_1=\sum_{j=1}^{m}\hat{x}_{1j}=\hat{x}_{11}+\hat{x}_{12}+\hat{x}_{13}+\hat{x}_{14}+\hat{x}_{15}=5$$

$$\hat{x}_2=\sum_{j=1}^{m}\hat{x}_{2j}=\hat{x}_{21}+\hat{x}_{22}+\hat{x}_{23}=2.5$$

The approximate solution (5, 2.5) is very close to the actual solution (5.2, 2.4), which can be found by dynamic programming or by other methods.

Table 7.1 Initial solution

PROGRAM	COST	QUANTITY	x_{11}	x_{12}	$\cdots$	$x_{1,12}$	x_{21}	$\cdots$	$x_{2,12}$	s_1	s_2	$\cdots$	s_{26}	a_1
a_1	M	10	1	1		1	2		2	−1	0		0	1
s_2	0	12	1	1		1	1		1	0	1		0	0
s_3	0	1	1	0		0	0		0	0	0		0	0
$\vdots$	$\vdots$	$\vdots$												
s_{26}	0	1	0	0		0	0		1	0	0		1	0
z_j-c_j			$M+5$	$M+3$		$M-17$	$2M-5$		$2M-27$	$-M$	0		0	0

7.4.3 PIECEWISE LINEAR SEPARABLE CONVEX OBJECTIVE FUNCTION—AN ILLUSTRATIVE EXAMPLE

Consider the example

$$\max z = 3x_1 + 2x_2$$

s/t

$$x_1 + 2x_2 \leq 10$$

The per-unit contribution of x_1 drops from 3 to 1.5 after $x_1 > 5$.

This is a typical business problem of decreasing returns to scale.

Solution, Step I: Write the Objective Function as a Sum of Single-Variable Functions

$$z = 3x_1 + 2x_2 = f_1(x_1) + f_2(x_2) = (3x_1) + (2x_2)$$

This is a special case of separable convex programming where the separated objective functions are already piecewise linear (Figure 7.15).

Solution, Step II: Test for Convexity

The separated functions are linear, and hence separable convex programming can be used.

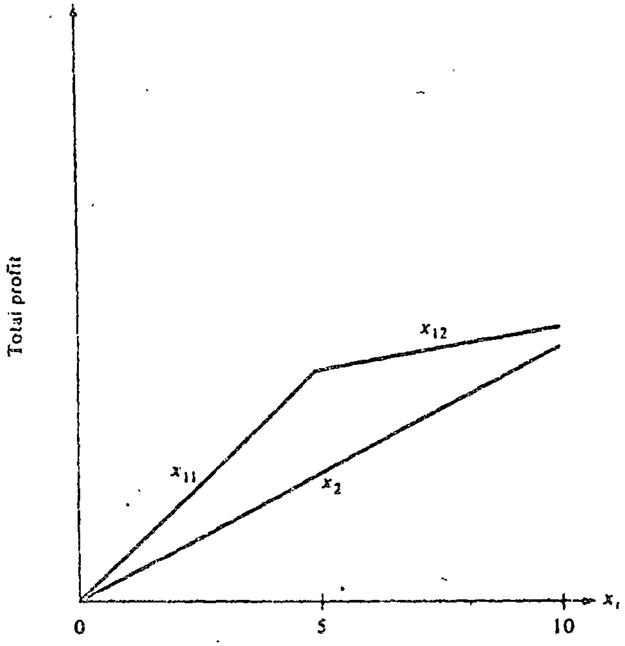


FIGURE 7.15

Solution, Step III: Write the Equivalent Problem

Break each $f_j(x_j)$ into separate parts: (a) We break $f_1(x_1)$ at point $x_1 =$ (b) We need not break $f_2(x_2)$.

Solution, Step IV: Compute the Coordinates of the Break Points

x_1	0	5	10
$f_1(x_1)$	0	15	22.5

x_2	0	10
$f_2(x_2)$	0	20

Solution, Step V: Decompose x_j into Auxiliary Variables x_{jm}

In this case it is not necessary to decompose x_2. The variable x_1 is decomposed as follows:

$$x_1 = x_{11} + x_{12}$$

Solution, Step VI: Write the Equivalent Linear Objective Functions That Approximate $f_j(x_j)$

$$f_1(x_1)_{\text{linear}} = 3x_{11} + 1.5x_{12}$$
$$f_2(x_2)_{\text{linear}} = 2x_2$$

Solution, Step VII: Write the Equivalent Constraints

$$\begin{aligned} x_{11} + x_{12} + 2x_2 &\leq 10 \\ x_{11} \qquad\qquad &\leq 5 \\ x_{12} \qquad &\leq 5 \\ x_{11},\ x_{12},\ x_2 &\geq 0 \end{aligned}$$

Solution, Step VIII: Solve the Problem as a Regular Linear Program

Our equivalent linear-programming problem is

$$\max z = 3x_{11} + 1.5x_{12} + 2x_2$$

s/t

$$\begin{aligned} x_{11} + \quad x_{12} + 2x_2 &\leq 10 \\ x_{11} \qquad\qquad\quad &\leq 5 \\ x_{12} \qquad &\leq 5 \\ x_{11},\quad x_{12},\quad x_2 &\geq 0 \end{aligned}$$

The optimal solution to the above linear-programming problem is

$$\hat{x}_{11} = 5 \quad \hat{x}_{12} = 5 \quad \hat{x}_2 = 0$$

Solution, Step IX: Transformation to Original Variables

$$\dot{x}_1 = \dot{x}_{11} + \dot{x}_{12} = 5 + 5 = 10$$

$$\dot{x}_2 \qquad\qquad = 0$$

Hence the optimal solution is (10, 0), and the value of the objective function, z, is $3(5)+1.5(5)=22.5$.

Note: The solution in this special case is the optimal solution and not an approximate solution, as was the case in the previous example.

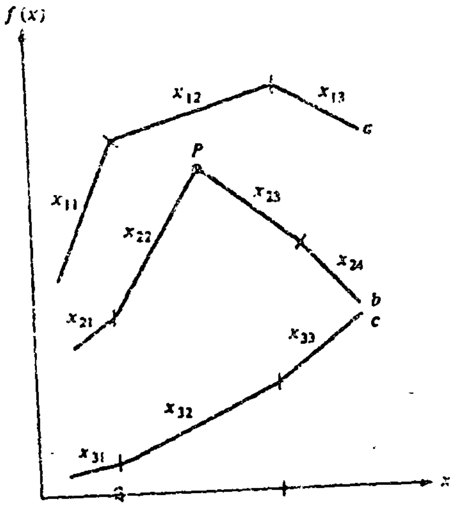
7.4.4 EVALUATION

appears to be an efficient technique for sol- Separable convex programming problems (for example, those ving certain classes of nonlinear-progamming constraints). problems that are linear except for a small number of nonlinear the use We listed in Section 7.4.1 a set of three basic requirements that limit functions of this technique. Of these, the requirement that the separable special be concave (in maximization) and convex (in minimization) is of interest. Let us examine Figure 7.16.

The nonlinear function represented by a is concave with decreasing slope for segments x_{12} and x_{13}.

If function a is a profit function that is to be maximized, we have no problem in arriving at the solution because we would include first the more profitable program, with the steeper slope (x_{11}), and then include x_{12} if additional capacity is available.

However, if function a is a cost function, part x_{12} should be included in



'FIGURE 7.16

the solution *before x_{11} is included.* This, however is not feasible in many real cases. For example, x_{11} may represent regular time, and x_{12} overtime. It is clear that one cannot operate exclusively on overtime.

Similarly, it is meaningless to talk about maximization of the convex function c, because our solution will indicate the preference of x_{32} over x_{31} since the contribution of x_{32} is larger than x_{31}. We can, however, minimize the function c since x_{31} is included first into the solution; and reality is in accord with the program output.

In the case of function b, we have a convex function up to point P, and then the function becomes concave. In such a case neither maximizing nor minimizing the entire function makes any sense. Instead we focus on maximizing or minimizing segments.

The three cases depicted in Figure 7.16 have been discussed here to show why the requirement of convexity (minimization) and concavity (maximization) fit the realities of practical problems. Of special interest is Miller's extension of this approach to nonconvex functions [43].

Finally, another difficulty is that separable programming yields a linear program of rather larger proportions.

7.4.5 SEPARABLE CONSTRAINTS

We have thus far presented methods that deal with separable convex objective functions, linear as well as nonlinear. Although the procedure is quite complicated, it is possible to make *any* nonlinear objective function linear by introducing additional nonlinear constraints.[17] This means that *any nonlinear problem can be transformed to a problem* with linear objective function subject to nonlinear constraints. The implication is that if we can find an efficient method of solving the transformed function, we can solve any nonlinear problem. Unfortunately, this is not easy to do. However, we do have a development in this area by Miller [43], known as *separable programming (of the constraints).* As in the case of the separable objective function, it is assumed here that all the nonlinear constraints can be separated into sums and/or differences of nonlinear functions of single variables. The method guarantees only local optima.

7.4.6 PRODUCT TERMS

A major requirement in separable programming is that the functions involved be separable. In many real-life problems this condition is satisfied. But in some problems there is an interdependence among variables in such a way that product terms (such as $3x_1\ x_2$, $4x_1\ x_2\ x_3$, and $-2x_1\ x_2^2$) are found in the objective function and/or in the constraints. Ordinarily, separable

[17] For a proof see Wolfe [58].

programming cannot be applied to such situations. However, in some cases it is possible to transform the product term into a separable form and then apply separable programming.

Of special interest is a quadratic form. A quadratic form can be easily expressed as a sum or difference of squares.

Example: A quadratic form x_1x_2 can be written as

$$x_1x_2 = \frac{1}{4}(x_1+x_2)^2 - \frac{1}{4}(x_1-x_2)^2$$

By the simple transformation $y_1 = x_1 + x_2$, $y_2 = x_1 - x_2$, we get

$$x_1x_2 = \frac{1}{4}y_1^2 - \frac{1}{4}y_2^2$$

which is a separable function.

Once the transformed function has been solved, it is easy to calculate the optimal value of the original variables. For example, assume that we get

$$\mathring{y}_1 = 6 \quad \mathring{y}_2 = 2$$

Then we have

$$x_1 + x_2 = 6$$

$$x_1 - x_2 = 2$$

This is a system of two linear equations, which yields

$$\mathring{x}_1 = 4 \quad \mathring{x}_2 = 2$$

Thus, any two-variable product term can be made to behave as separable.

Another possible transformation for x_1x_2 is with the aid of logarithmic transformation; that is, let

$$y = x_1x_2$$

then

$$\log y = \log x_1 + \log x_2$$

Hence the problem of maximizing x_1x_2 is equivalent to

$$\max y$$

s/t

$$\log y = \log x_1 + \log x_2$$

If we assume that both x_1 and x_2 are positive variables, then the problem is separable. The last transformation shows that an expression can be made separable by introducing additional variables and additional constraints. With some ingenuity it is possible to put into separable forms many nonlinear expressions.

7.4.7 THE PROPERTY OF ADJACENT WEIGHTS

Some separable-programming problems have a certain property termed t "property of adjacent weights." Such problems must have all linear co straints and a concave (in maximization) objective function. In such a cas the simplex method can be adapted. For details see Wagner [54] and Zangw [60].

7.5 OTHER METHODS FOR SOLVING NONLINEAR-PROGRAMMING PROBLEMS

7.5.1 INTRODUCTION

The calculus optimization approach (Appendix B) and the Lagrangiar method (Appendix D), though of important theoretical value, are very inefficient as computational devices for most of the programming problems More efficient computational techniques, such as separable programming, are limited to a small segment of the problems encountered in nonlinear programming. Thus it is logical for researchers to divert their attention toward developing a general and efficient nonlinear program. Attempts to find a general, efficient, nonlinear-programming method that is equivalent to the simplex method in linear programming have thus far been unsuccessful. However, several interesting methods have been developed, each with its own merits and limitations.

In the following section we will survey some of the more classical methods. Large numbers of the special methods are being published in the professional literature (for example, in *Operations Research*). For further details see [2], [8], [18], [31], [33], [35], [39], and [60].

7.5.2 THE GRADIENT METHODS[18]

Gradient methods are procedures that guide us in obtaining a better approximation of the optimal solution. A search for the optimal solution starts from some initial feasible solution such as point *A* in Figure 7.17 and proceeds to new points *B*, *C*, and so on.

The direction of movement is such that we improve the value of the objective function (that is, we search "uphill" on a profit function or "downhill" on a cost function). We search only those points that are within the feasible area. The "target" of the search is the stationary point (maximum, minimum, or saddle point).

[18] For a detailed discussion and examples see Hadley [22], Chapter 9

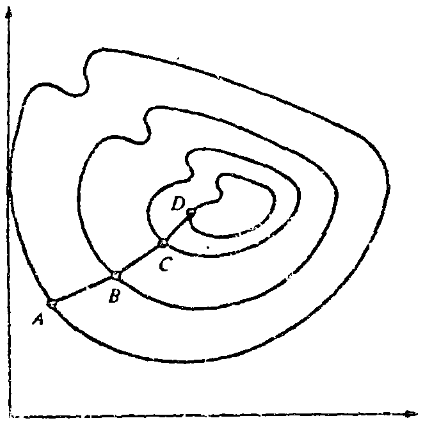


FIGURE 7.17

Basically, the method makes use of the vector of partial derivatives of the objective function:

$$f_x = \begin{bmatrix} f_{x_1} \\ f_{x_2} \\ \cdot \\ \cdot \\ \cdot \\ f_{x_n} \end{bmatrix}$$

The vector f_x is known as the *gradient* of f.

From the viewpoint of improving the objective function in a most efficient manner, when we move from a given point A to a point B, we move in the direction of the vector f_x. This is done by evaluating the partial derivatives at point A and choosing the direction of greatest advantage.

The method is similar to mountain climbing. Assuming that we have only one peak (concave function), we may search for this peak by adopting a policy of starting from a point A and going to a point B, which appears to be the highest point in the neighborhood. At point B we again look for the next highest point, say C. We continue to search till we arrive at the summit and find that no more improvement in the objective function can be made.

When we try to solve a constrained problem by this method, we change the constrained problem to a nonconstrained one, use the Lagrangian function, and then try to search for a *saddle point* on the Lagrangian function.

This method presents several practical difficulties in connection with convergence, stopping rules, parameter selection, and computerization. Of the several versions that exist, a prominent one is Rosen's gradient projection method [47, 48].

As is true for most nonlinear-programming methods, the gradient method will work successfully only in the presence of diminishing returns. The method can also be used to solve linear-programming problems. When the gradient method is applied to linear-programming problems, we can arrive at an approximate solution faster than we can arrive at the optimal solution by the simplex method.

The gradient method can be used in *unconstrained* as well as in *constrained* optimization problems. The method is more efficient for those problems that have *linear constraints*.

Gradient methods also have been labeled as *methods of feasible directions* About half of the methods listed in Table 7.2 belong to this family. Of special interest is Zoutendijk's algorithm [63]. This method has been extensively tested (see Section 7.7), especially for cases of linear constraints. For additional discussion of these methods see Hadley [22].

7.5.3 CUTTING-PLANE METHODS

Cutting-plane methods (see, for example, Kelley [30]) attempt to convert the given nonlinear problem to one of minimizing (or maximizing) a linear objective function while approximating the boundary of the feasible area by a convex polyhedron. This is done by solving a sequence of linear-programming problems whose solutions, in the limit, *approach* the solution of the original problem. The method relies almost exclusively on the fact that the tangent plane to any point on the boundary of a convex region lies entirely outside the region. For this reason it is not well suited to nonconvex problems. Even for convex problems the computational experience has not been extensive. The cutting-plane method is restricted to objective functions that are twice differentiable.

7.5.4 SEQUENTIAL UNCONSTRAINED MINIMIZATION TECHNIQUE (SUMT)

A more recent method, developed by Fiacco and McCormick [18], is known as SUMT. The basic idea of this technique is to transform the original problem into a sequence of unconstrained optimization problems. This is desirable because a number of methods of unconstrained optimization are available (for example, classical calculus and the steepest-ascent method) and many newer ones are being developed. SUMT is one of the most promising tools for solving nonlinear-programming problems. Research efforts are being currently conducted to improve its efficiency.

7.5.5 BRANCH-AND-BOUND APPROACH

For all-integer and mixed-integer convex nonlinear problems, the branch-and-bound approach (see Section 6.7) may be used.

The idea is to solve the problem first without paying attention to the integer requirement, thus finding the *bound* (upper for maximization, lower for minimization) on the objective function If the solution is noninteger (for example, $x_1 = 5.7$) instead of integer, the problem is *branched* into two problems: (a) the original problem with an additional constraint $x_1 \le 5$, and (b) the original problem with an additional constraint $x_1 \ge 6$. We then solve the two new subproblems. If one of them satisfies the integer requirements and its objective function value is better than the other one, we have the optimal solution; otherwise we continue to *branch* each subproblem further, until we arrive at an integer solution. Each time we have to use some method to proceed from the noninteger solution to an integer solution. The difficulty is that we add more and more constraints as the branching goes on. The process is similar to the one presented in Section 6.7.

7.5.6 GEOMETRIC PROGRAMMING

The geometric approach is based on the mathematical theory of inequalities and on the use of an associated dual problem. For a nonlinear problem with a special structure the solution may be obtained simply by solving a set of linear equations. Overall, the technique uses mathematics above the level of this text.

The interested reader is referred to the work of Duffin *et al.* [17].

7.5.7 OTHER METHODS

Many other methods have been developed both for convex and nonconvex programming. Most of them are limited to special applications, which are usually rather complicated. The interested reader is referred to such professional journals as *Operations Research*, *Management Science*, *Econometrica*, *SIAM Journal*, *Bulletin of the American Mathematical Society*, and *Naval Research Logistics Quarterly*, in which a large number of articles appear on the subject. Some of the methods of special interest are:

1. *Charnes and Lemke's* [9] *extended technique.* This is an extension of the technique of separable functions (Section 7.4) to the nonconvex case. However, the extended method does not insure global extrema.
2. *Use of integer programming.* This special approach is discussed in Chapter 6. The basic idea is that many nonlinear, and even many nonconvex problems can be approximated by, or reduced to, an integer linear-programming form (see Gomory [20]).
3. *Use of dynamic programming.* Dynamic programming can be applied to convex as well as to nonconvex nonlinear-programming problems, with varying degrees of success. In some instances the computation is very lengthy. It has been applied quite successfully to transportation problems with two origins having nonlinear costs. Alternative methods for such problems are not, in general, efficient. See Hadley [22] and Nemhauser [44].
4. *Heuristic search.* There is an increasing trend towards the use of heuristic search methods for solving complex, especially nonconvex, nonlinear programs. Utilizing such a search is similar to the enumeration approach. However, whereas in the enumeration approach we check all possible solutions, in the heuristic approach we check only a finite, relatively small, number of solutions. This procedure, of course, does not guarantee optimality. See Wagner [54].

7.5.8 STOCHASTIC PROGRAMMING

Stochastic programming deals with situations in which some or all the parameters of the problem are described by random variables.

A nonlinear stochastic-programming problem can be stated as

$$\min F(x) + G(y) \tag{7.23}$$

s/t

$$\mathbf{x}_i + \mathbf{y}_i \ge \mathbf{b}_i \qquad i = 1, 2, \ldots, m$$

where $\mathbf{b} = b_1, \ldots, b_m$ is a stochastic m-dimensional variable and $\mathbf{x}$, $\mathbf{y}$ are *production* and *purchase* vectors at two stages, with corresponding cost vectors $F(x)$ and $G(y)$.

An example of an actual situation is that of a two-stage manufacturing-inventory problem. The random vector $\mathbf{b}$ represents a demand for m manufactured products that can only be specified in advance by its probability distribution. The vector $\mathbf{x}$ determines the amount of each product that will be made during some period before the actual demand is known. This first stage production of products is given by the vector $\mathbf{x}$ at a cost of $F(x)$. Once the actual demand $\mathbf{b}$ is known, we are forced to choose the optimum value of $\mathbf{y}$ that will compensate, at minimum cost, for shortages. This is the second stage, and results in a production or outside purchase vector $\mathbf{y}$ at a cost $G(y)$. The problem is to select the first-stage production vector so as to minimize the expected value of the total cost $F(x) + G(y)$.

A linear two-stage problem has been discussed by Dantzig and Madansky [12] and Madansky [38]. Mangasarian and Rosen [40] have taken the results of Madansky (who obtained upper and lower bounds on the optimum solution to this two-stage problem for the completely linear case) and extended these results, under appropriate convexity, concavity, and continuity conditions, to the two-stage nonlinear case. Bui Trong Lien, in Abadie [1], attempted to point out certain possibilities of generalizing the inequalities found in the references cited above.

7.6 SOME APPLICATIONS OF QUADRATIC PROGRAMMING

7.6.1 GENERAL

Until recently there was very little evidence for practical applications of nonlinear programming. Most known applications were in the area of quadratic programming. However, with improved computational efficiency there is *increasing evidence* for the use of nonlinear programming in many areas of marketing, production, finance, and services. (See Williams [57].)

In this section we present some examples of quadratic programming that can be considered, for the most part, as classical.

7.6.2 OPTIMUM ALLOCATION OF RESOURCES

Allocation of resources under perfect competition is a classical linear-programming problem. In linear programming, the optimal solution is such that the total net revenue equals the total marginal net revenue. If prices are not constant but instead are a function of volume, the problem has a nonlinear, usually quadratic, objective function.

7.6.3 PORTFOLIO SELECTION

A well-known quadratic programming model, dealing with the problem of selecting an investment portfolio that will yield a given expected total return with minimum variance, was developed by Markowitz [41]. The problem, often referred to as the portfolio selection model, assumes that the investor wishes to maximize his anticipated returns, and considers variance of return as undesirable. Minimizing variance of course minimizes the risk involved. The above objectives are reasonable because the portfolio with maximum expected return is not necessarily the one with minimum variance. There is a rate at which the investor can gain expected return by accepting greater variance or reduce variance by trading off expected return.

Assuming that the only constraint to be satisfied is that of investing all available funds, then the foregoing problem becomes an optimization problem and the solution can be obtained by using quadratic-programming techniques. The problem is as follows:

An investor has a fixed amount of dollars to invest in *n* available potential activities. However, these *n* activities yield varying returns (dividends, interest), and also the rates of appreciation fluctuate differently for the *n* activities. The variance (when the actual rate is different than the expected rate) can be considered as the risk involved for each individual activity. A quadratic programming problem is generated if this variance, in equation form, is incorporated in the objective function.

The problem in this case can be solved with the Kuhn-Tucker conditions, using Lagrange multipliers.

7.6.4 INCOME VARIATION AND SELECTION OF ENTERPRISES [50]

A considerable amount of time, money, and effort has been devoted toward developing methods in aiding farmers to maximize their expected or average

profits. However, the variance of expected farm income (see Figure 7.18) has been seriously neglected, although it is recognized as being an important component in a farmer's decision-making process. The method used in this problem is similar to that used in the portfolio selection discussed previously, except that in this case the objective is to minimize total income variance when decisions are made concerning farm planning. In this case, total income variance V is given as follows:

$$V = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \sigma_{ij} \qquad (7.24)$$

where σ_{ij} = covariance between ith and jth enterprise
σ_i^2 = variance of ith enterprise
σ_j^2 = variance of jth enterprise

$$\sigma_{ij} = \sigma_i^2 = \sigma_j^2, \text{ for } i = j.$$

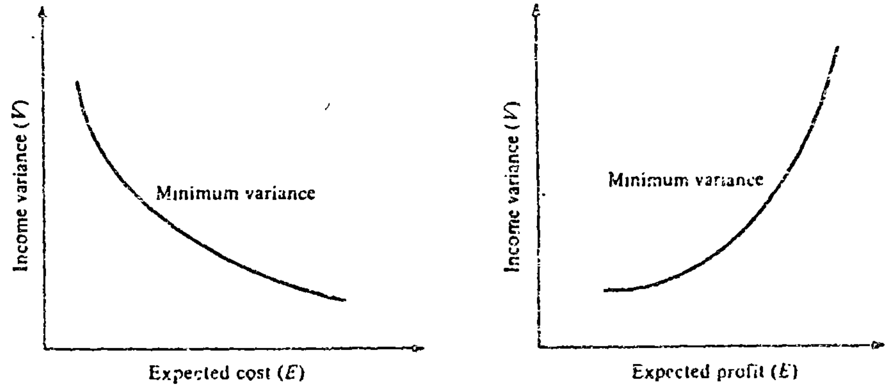


FIGURE 7.18

The minimum variance is a convex function of the expected cost or profit. Let us now see how income variance can affect a farmer in choosing his enterprise mix. Referring to Figure 7.19, assume that the shaded area indicates a set of all enterprises that yield desirable expected incomes and their corresponding variances. Each point represents the expected income and variance from a specific enterprise mix.

Point B represents the mix giving the maximum expected income, but likewise, the largest variance. Of course it is possible that maximum variance may occur with a mix that does not yield the maximum income, but we will not consider such a situation here. Points on the curve OAB yield the "efficient enterprise mixes," since for any specific income level the mix with the minimum variance will be found on this curve. For example, point A has a lower variance than point C, yet has an equivalent expected income.

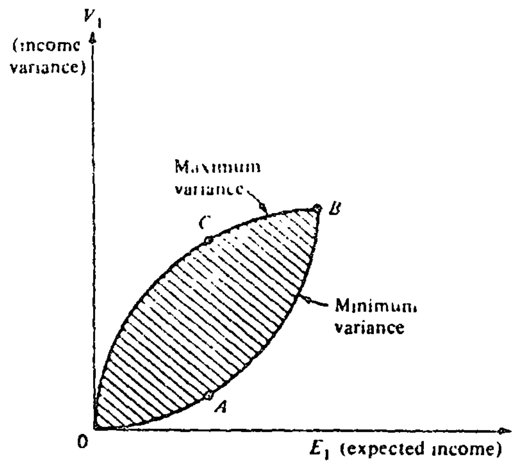


FIGURE 7.19

Now if the farmer has a good equity position, he may choose a mix represented by point B (the most risky, but yielding the highest expected return), whereas the conservative farmer or one with a low equity position might choose a mix on curve OAB closer to the origin. Using a linear-programming analysis, maximizing returns would indicate the optimal mix at point B.

What we have arrived at is a total variance function in a quadratic form that can be minimized, subject to linear constraints, including a resource restriction (land) and an income restriction. Let us illustrate.

Assume that there are two enterprises available: x_1 and x_2. Each is being considered for a farm. The average net income per acre for x_1 and x_2 are g_1 and g_2, respectively. Let us also assume that we know the variances $\sigma_1{}^2$ and $\sigma_2{}^2$ and the covariance σ_{12} of the net incomes g_1 and g_2. The problem then becomes

$$\min V=\sigma_1{}^2x_1{}^2+\sigma_2{}^2x_2{}^2+2\sigma_{12}x_1x_2 \qquad (7.25)$$

s/t

$$\begin{aligned} g_1x_1+g_2x_2&\geq b_1 \qquad \text{(income)}\\ x_1+\ \ x_2&\leq b_2 \qquad \text{(land)} \end{aligned}$$

where b_1 = desired income level and b_2 = amount of land available.

Through the use of quadratic programming the quantities of x_1 and x_2 that will minimize the income variance can be found.

7.6.5 SURPLUS MILK IN THE NETHERLANDS

A surplus of milk poses a problem in some countries. Theoretically, the farmer should be willing to increase his milk production to that point at which his marginal cost equals his marginal revenue. However, there are two major practical difficulties that contradict the theoretical viewpoint First, the marginal cost that the farmer will calculate for the short run is probably incorrect because he usually excludes his own labor from the costs Second, even if the farmer calculates his costs correctly, he cannot survive in the long run because of such factors as dependability of supply to customers, and so on In any event, the circumstances are such that in some countries the milk industry is subsidized and regulated by the government.

This, for example, is the case in the Netherlands. The farmers deliver all their milk to a government agency at a guaranteed price. The agency resells the milk and milk products, such as butter and cheese. Since the agency sets the price for both the producer and the consumer, it acts as a monopolist. Our problem in such a case is to determine how a given amount of milk (production in any year is predictable) should be allocated to milk, butter, and cheese, and what price should be charged for these products in order to minimize the subsidy the government must pay to the farmers. The problem has been formulated and solved as a quadratic program. For details, including sensitivity analysis, see Boot [7].

7.6.6 ELECTRICAL NETWORKS

An elegant analogy between the theory of electrical networks and the notion of duality in nonlinear programming is presented by Dennis [13]. Primal-dual presentation of linear and quadratic programming is discussed, in electrical-network terms.

7.6.7 STRUCTURAL MECHANICS

Quadratic programming can be used in structural mechanics. Dorn [16] has presented the case of elastic-plastic analysis of trusses as a primal-dual quadratic programming problem.

7.6.8 PRODUCTION SCHEDULING

Holt, Modigliani, and Muth developed a model that minimizes the cost of producing a product over a number of time periods. Details are given in [26], [27], and [51].

7.7 COMPUTATIONAL EXPERIENCE

7.7.1 GENERAL

Numerous algorithms have been programmed for computer use It is very difficult to measure and compare the effectiveness of these algorithms. The most detailed study, involving 34 different computer codes, has been made by Coville (in Kuhn [32]). For other general surveys and codes, see Aronofsky [3] and Kunzi *et al.* [34].

The major conclusion of Coville's comparative study is that the efficiency and performance of a nonlinear-programming code can be greatly affected by the method of implementing it on a computer. Another important conclusion is that many of the methods were quite efficient with regard to one or more specific problems and less efficient as to other problems.

7.7.2 CODES

Of the many available codes we chose to list in Table 7.2 the major codes studied by Coville. Many of the codes are available through SHARE. Information about the codes can be obtained from the "participants" listed in Table 7.2.

Table 7.2 Nonlinear programming codes[a]

	PARTICIPANT	AFFILIATION	DERIVATIVES	WEIGHTED AVERAGE SCORE[b]
1. SEARCH METHODS				
OPTIM	Boas	Mobil	none	0.37
Sequential search	Cooper	Washington Univ.	none	−0.67
COMPLEX	Davies	ICI Ltd.	none	−2.84
Rosenbrock	Davies	ICI Ltd.	none	−1.74
Klingman & Himmelblau	Grace	P and G	analytic	1.08
Mult. gradient summation technique	Himmelblau	Univ. of Texas	analytic	−4.54
CANDIDE	Himmelblau	Univ. of Texas	none	−5.00
Simplex search	Miller	Shell Dev.	none	0.38
PROBE	Sullivan	IBM	none	0.56
2. SMALL-STEP GRADIENT METHODS				
POP/360	Colville	IBM	numeric	0.94
Richochet gradient	Greenstadt	IBM NYSC	analytic	0.70
POP II/7094	Grigsby	Phillips	numeric	−0.73
Carbide optimization package	Hutton	Union Carbide	numeric	—
Generalized gradient search	Kephart	Union Carbide	numeric	−0.32
Method of approx. programming	Miller	Shell Dev.	numeric	−2.13
Deflected ascent	Miller	Shell Dev.	numeric	—
3. LARGE-STEP GRADIENT METHODS				
Generalized reduced gradient	Abadie	EdF	analytic	1.05
GRG II	Abadie	EdF	analytic	2.64
Method of feasible directions	Anthony	IBM Resch. [illegible]	analytic	0.28
Davidon with CRST	Davies	ICI Ltd.	analytic	0.70
Convex programming	Gauthier	IBM, France	analytic	−0.31
Conjugate gradient	Goldfarb	Courant Institute	analytic	−0.11
Reduced gradient	Huard	EdF	analytic	−1.21
Gradient projection corrigé	Kalfon	EdF	analytic	1.56
Gradient projection	Miller	Shell Dev.	analytic	0.53
Variable metric projection	Murtagh	Imperial College	analytic	0.86
Revised reduced gradient	Ribiere	IBM, France	analytic	1.49
Modified feasible directions	Tzschach	IBM, Germany	analytic	0.95
4. SECOND-DERIVATIVE METHODS				
Courant	Ballot	CCSA	analytic	1.02
Gauss-Newton-Carroll	Bard	IBM-CSC	analytic	0.67
SUMT	McCormick	RAC	analytic	0.85
SOLVER	Wilson	Stanford Univ.	analytic	0.87
5. MISCELLANEOUS ITEMS				
Separable programming	Harvey	Std. Oil of Cal.	none	−2.46
Method of centers	Huard	EdF	analytic	−0.36

[a] Source: Kuhn [32], pp. 497 and 498.

[b] The positive numbers indicate more efficient codes.

7.8 CONCLUDING REMARKS

It should be noted that there is similarity in all the models discussed in Chapters 3 and 7. In linear, as well as nonlinear, models we have been essentially dealing with only single-stage problems.

The reader must by now realize the power of these single-stage models to solve several practical problems. However, the question of multistage processes remains unresolved. Dynamic programming, the subject of our next chapter, deals with multistage decision problems. It also can be used to solve nonlinear-programming problems.

BIBLIOGRAPHY

1. ABADIE, J., ed., *Nonlinear Programming*. New York: Wiley, 1967.
2. ABADIE, J., ed., *Integer and Nonlinear Programming*. Amsterdam: North-Holland, 1970.
3. ARONOFSKY, J. S., ed., *Progress in Operations Research—The Relationship between Operations Research and the Computer*. New York: Wiley, 1968.

4. Barankin, E. W., and R. Dorfman, "On Quadratic Programming," *University of California Publications in Statistics*, vol. 2, 1958, pp. 258–318.
5. Beale, E. M. L., "On Quadratic Programming," *Naval Research Logistics Quarterly*, vol. 6, September 1959, pp. 227–244.
6. Beale, E. M. L., *Mathematical Programming in Practice*. London: Sir Isaac Pitman & Sons Ltd., 1968.
7. Boot, J. C. G., *Quadratic Programming*. Skokie, Ill.: Rand McNally, 1964.
8. Bracken, J., and G. P. McCormick, *Selected Applications of Nonlinear Programming*. New York, Wiley, 1968.
9. Charnes, A., and C. E. Lemke, "Minimization of Non-linear Separable Convex Functions," *Naval Research Logistics Quarterly*, vol. 1, December 1954, pp. 301–312.
10. Dantzig, G. B., *Quadratic Programming, A Variant of the Wolfe-Markowitz Algorithm*. Research Report 2, Operations Research Center, University of California, Berkeley, 1961.
11. Dantzig, G. B., *Linear Programming and Extensions*. Princeton, N.J.: Princeton University Press, 1963.
12. Dantzig, G. B., and A. Madansky, "On the Solution of Two-Stage Linear Programs Under Uncertainty," *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley: University of California Press, 1961, vol. 1, pp. 165–176.
13. Dennis, J. B., *Mathematical Programming and Electrical Networks*. Cambridge, Mass.: M.I.T. Press, and New York: Wiley, 1959.
14. Dorn, W. S., "Non-Linear Programming—A Survey," *Management Science*, vol. 9, 1963, pp. 171–208.
15. Dorn, W. S., "Duality Theorem for Convex Programs," *IBM Journal of Research and Development*, vol. 4, pp. 407–413.
16. Dorn, W. S., *Variational Principles for an Elastic-Plastic Material*. IBM Research Report (to appear).
17. Duffin, R. J., E. L. Peterson, and C. Zener, *Geometric Programming: Theory and Application*. New York; Wiley, 1968.
18. Fiacco, A. V., and G. P. McCormick, *Nonlinear Programming: Sequential and Unconstrained Minimization Techniques*. New York: Wiley, 1968.
19. Frank, M., and P. Wolfe, "An Algorithm for Quadratic Programming," *Naval Research Logistics Quarterly*, vol. 3, 1956, pp. 95–110.
20. Gomory, R. E., "Large and Non-Convex Problems in Linear Programming," *Proceedings of the 15th Symposium on Applied Mathematics of the AMS*, 1963, p. 125.
21. Graves, R. L., and P. Wolfe, *Recent Advances in Mathematical Programming*. New York: McGraw-Hill, 1963.
22. Hadley, G., *Nonlinear and Dynamic Programming*. Reading, Mass.: Addison-Wesley, 1964.
23. Hancock, H., *Theory of Maxima and Minima*. New York: Dover, 1960.
24. Hartley, H. O., "Non-Linear Programming by the Simplex Method," *Econometrica*, vol. 29, 1961, pp. 223–237.
25. Hillier, F. S., and G. J. Lieberman, *Introduction to Operations Research*. San Francisco: Holden-Day, 1967.
26. Holt, C. C., F. Modigliani, and J. F. Muth, "Derivation of Linear Decision Rule for Production and Employment," *Management Science*, vol. 2, January 1956, p. 159.
27. Holt, C. C., F. Modigliani, and H. A. Simon, "A Linear Decision Rule for Production and Employment Scheduling," *Management Science*, vol. 1, October 1955.
28. Houthakker, H. S., "The Capacity Method of Quadratic Programming," *Econometrics*, vol. 28, 1960, pp. 62–67.
29. IBM, *Mathematical Programming System/360*. Nos. H20-0476-0 and H20-0136-2, IBM Technical Publication Department, White Plains, N.Y. 1967.
30. Kelley, J. E., Jr., "The Cutting-Plane Method for Solving Convex Programs," *SIAM Journal*, vol. 8, December 1960, pp. 703–712.
31. Kuhn, H. W., ed., *Proceedings of the Princeton Symposium on Mathematical Programming*. Princeton, N.J.: Princeton University Press, 1970.
32. Kuhn, H. W., and A. W. Tucker, "Nonlinear Programming," *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley: University of California Press, 1951, pp. 481–492.
33. Kunzi, H. P., *et al.*, *Nonlinear Programming*. Waltham, Mass.: Blaisdell, 1966.
34. Kunzi, H. P., H. G. Tzschuch, and C. A. Zehnder, *Numerical Methods of Mathematical Optimization*. New York: Academic Press, 1968.
35. Lasdon, L. S., *Optimization Theory for Large Systems*. New York: Macmillan, 1970.
36. Lawler, E. L., and D. W. Wood, "Branch-and-Bound Methods, A Survey," *Operations Research*, vol. 14, 1966, p. 699.
37. Lemke, C. E., "A Method of Solution for Quadratic Programs," *Management Science*, vol. 8, no. 4, July 1962, pp. 442–453.
38. Madansky, A., "Methods of Solution of Linear Programs Under Uncertainty," *Operations Research*, vol. 10, 1962, pp. 463–471.
39. Mangasarian, O. L., *Nonlinear Programming*. New York: McGraw-Hill, 1969.
40. Mangasarian, O. L., and J. B. Rosen, "Inequalities for Stochastic Nonlinear Programming Problems," *Operations Research*, vol. 12, 1964, pp. 143–154.
41. Markowitz, H. M., *Portfolio Selection* (Cowles Commission Monograph No. 16). New York: Wiley, and London: Chapman & Hall, 1959.
42. McMillan, C., Jr., *Mathematical Programming*. New York: Wiley, 1970.
43. Miller, C. E., "The Simplex Method for Local Separable Programming," in Graves and Wolfe [21], pp. 89–100.
44. Nemhauser, G. L., *Introduction to Dynamic Programming*. New York: Wiley, 1966.
45. Orchard-Hays, W., "Structure of Mathematical Programming Systems," *ACM Proceedings of 23d National Conference*. Princeton, N.J.: Brandon/Systems Press, 1966, pp. 439–458.
46. Quirk, J. P., "The Capital Structure of Firms and the Risk of Failure," *International Economic Review*, vol. 2, no. 2, May 1961.
47. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part I—Linear Constraints," *SIAM Journal*, March 1960, pp. 181–217.
48. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part II—Nonlinear Constraints," *SIAM Journal*, December 1961, pp. 514–532.

49. Smith. V. L., *Investment and Production.* Cambridge, Mass.: Harvard University Press, 1961.

50. Stovall, J. G., "Income Variation and Selection of Enterprises," *Journal of Farm Economics*, vol. 48. no. 5, December 1966.

51. Teichroew, D., *An Introduction to Management Science: Deterministic Models.* New York: Wiley, 1964.

52. Theil, H., and C. Van de Panne, "Quadratic Programming as an Extension of Classical Quadratic Maximization," *Management Science*, vol. 7, no. 1 October 1960, pp. 1–20.

53. Vajda, S., *Mathematical Programming.* Reading, Mass.: Addison-Wesley, 1961.

54. Wagner, H. M., *Principles of Operations Research.* Englewood Cliffs, N.J.: Prentice-Hall, 1969.

55. Wagner, P , "A Nonlinear Programming Extension of the Simplex Method," *Management Science*, vol. 7, no. 1, October 1960, pp. 43–56.

56. Wilde, D., and C. Beightler, *Foundation of Optimization.* Englewood Cliffs, N.J.: Prentice-Hall, 1967.

57. Williams, N., *Linear and Nonlinear Programming in Industry.* London: Sir Isaac Pitman & Sons Ltd., 1968.

58. Wolfe, P., "The Simplex Method for Quadratic Programming," *Econometrica*, vol. 27, 1959, pp. 382–398.

59. Wolfe, P., in Graves and Wolfe [21], pp. 67–86.

60. Zangwill, W., *Nonlinear Programming—A Unified Approach*, Englewood Cliffs, N.J.: Prentice-Hall, 1969.

61. Ziemba, W. T., "Computational Algorithms for Convex Stochastic Programs with Simple Resources," *Operations Research*, vol. 18, no. 3, May–June, 1970.

62. Ziemba, W. T., "Transforming Stochastic Dynamic Programming Problems," *Management Science*, March 1971.

63. Zoutendijk, G., *Methods of Feasible Directions.* New York: Elsevier, and Princeton, N.J.: Van Nostrand, 1960.

PROBLEMS

7.1 What are the major hazards in approximating nonlinear programming by linear programming?

7.2 The International Chemical Corporation uses two raw materials A and B in making one of its leading products. In each batch it is required that at least 4 tons of raw materials be included and not less than 2.75 tons of raw material B.

The cost associated with the materials varies with the quantity in the following way: For material A the cost *per ton* is $x_1 - 1$ (where x_1 is the number of tons bought); for material B the cost *per ton* is $x_2 - 4$ (where x_2 is the number of tons bought).

The company's objective is to minimize the cost of the raw materials in each batch.

(a) Formulate as a mathematical programming problem.
(b) Solve (optional).
(c) Comment on the results (optional).

7.3 The ABC Company manufactures two products A and B. There are four departments whose capacities, per month, are given in the table below.

DEPARTMENT	UPPER CAPACITY IF ONLY A IS PRODUCED	UPPER CAPACITY IF ONLY B IS PRODUCED
I	250	400
II	250	—
III	—	360
IV	200	100

The products can be made by any department as long as capacity is available (at the same cost). Departments I and IV can produce either product, or both (in a linear proportion). Product A can be sold at quantities up to 600, yielding $p_1 = \$5000$ per unit sold. Product B, however, is facing rough competition and in order to sell larger quantities, considerable advertising is required. The net yield for unit of product B is given by

$$p_2 = 10{,}000 - 20x_2$$

where x_2 is the number of units of product B to be produced.

Find the product-mix and the production schedule that will maximize profit.

(a) Formulate as a mathematical programming problem in two different ways.
(b) Solve by the graphical method. (Show the feasible area and the objective function.)

Note: When solving this problem do not use common sense and shortcuts that could be employed in this specific case.

7.4 The Lehigh Computing Corporation is making two types of small specialized computers, A and B. The company cannot produce more than two computers a week. Of their three available teams, two are required for the production of type A and one for the production of type B every week.

The company profit (in thousands of dollars) for each unit of type A sold is $6 - 3x_1$ (where x_1 is the amount sold of type A) and for each unit of type B sold is $2 - x_2$ (where x_2 is the number sold of type B).

Find the most profitable production plan for the company.

(a) Formulate as a programming problem.
(b) Solve (optional).

7.5 Show that in a *quadratic form*,

$$\sum\sum c_{ij}x_ix_j = X^TCX$$

the matrix C will always be symmetric.

7.6 Write the following quadratic forms as a sum (difference) of squares of independent homogeneous linear expressions.

(a) $3x_1^2 - 3x_2^2 - 8x_1x_2$ (c) $2x_1^2 + 5x_2^2 - x_1x_2$
(b) $2x_1^2 + 2x_2^2$ (d) $x_1^2 + 2x_1x_2 - x_1x_3 + x_2^2$

7.7 Show that the product term $4x_1x_2x_3$ is separable.

7.8 Given the following functions:

$$6x_1^2 + 3x_2^2 + x_3^2 - x_1x_2 + 2x_1x_3 - 3x_2x_3$$
$$2x_1^2 + x_2^2 + 6x_1x_2$$
$$2.5x_1 + 2.25x_2 + 5x_3 - \frac{x_1^2}{300} - \frac{x_2^2}{500} - \frac{x_3^2}{1000}$$

(a) Write the functions in a matrix form.
(b) Check the convexity (concavity).
(c) Find and identify stationary points.

7.9 Given:

$$\max z = 2x_1 + 3x_2 - x_1^2 - x_2^2$$

s/t

$$x_1 + x_2 \leq 2$$
$$2x_1 + x_2 \leq 3$$

(a) Present as a separable program. Separate x_1 at 0.5, 0.8, 1, and 1.2. Separate x_2 at 0.5, 1, 1.3, 1.5, and 1.7.
(b) Solve. Find a second optimal approximate solution and comment on the results.

7.10 Solve Problem 7.9 by Frank and Wolfe's method.

7.11 Given:

$$\min z = 2x_1 + x_2$$

s/t

$$x_1 + x_2 \geq 5$$
$$x_1 + 2x_2 \leq 8$$

and when $x_2 \geq 2$ the contribution of x_2 is 3.
Solve this problem by separable programming.

7.12 Given:

$$\max z = 3x_1 + 3.5x_2 - 0.001x_1^2 - 0.0025x_2^2$$

s/t

$$x_1 + 2x_2 \leq 4000$$
$$4x_1 + 3x_2 \leq 12{,}000$$

Solve by separable programming and show graphically the separated functions. Separate x_1 at 0, 1000, 1500, 2000, and 3000. Separate x_2 at 0, 500, 700, 1000, and 2000

7.13 Solve Problem 7.4 by separable programming.

7.14 Prove that a positive definite function is always strictly convex.

7.15 Test for convexity (using determinants):

(a) $3.5x_2 - 3x_1 - \frac{x_1^2}{1000} - \frac{x_2^2}{400}$

(b) $5x_1^2 + 20x_1 - 3x_2^2 - 24x_2$

7.16 Show that a quadratic programming problem is separable if, and only if, the matrix C_1 is a diagonal matrix.

7.17 Explain why all linear-programming problems are essentially separable-programming problems.

7.18 Is the quadratic function with the general $x_ix_jx_k$ term separable?

7.19 Show that the function $e^{x_1+x_2}$ is separable.

7.20 Generalize the quadratic assignment problem and suggest a possible general method of solution.

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

APLICACIONES DE LAS COMPUTADORAS A LA SIMULACION Y OPTIMIZACION

APLICACIONES DE UN PAQUETE DE PROGRAMACION LINEAL

M. EN C. MARCIAL PORTILLA ROBERTSON

ABRIL DE 1978.

TARJETAS PARA USAR EL PROGRAMA GRANM

```
               Columna 1
               |
Tarjeta 1      // JØB T
Tarjeta 2      // XEQ GRANM    1
Tarjeta 3      *LØCALINIT, PIVØT, TABNU, RMØVE, CLEAN, TABPR

               -
               -
               -   Tarjetas de datos (Ver página 3)
               -
               -

Tarjeta        /*
 final
```

NOTAS:

- Este programa está listo para usarse en la computadora IBM 1130 de CECAFI.
- La tarjeta 1 es la tarjeta anaranjada obtenida del CECAFI.
- El número 1 que aparece en la 2a. tarjeta se perfora en la columna 17.
- El programa en la IBM, tiene una capacidad de 10 restricciones y 15 variables incluyendo de holgura y artificiales.
- Este programa también se encuentra disponible en la Burroughs del CIMASS, bajo el nombre de II/SIMPLEX. Las instrucciones para correrlo en el CIMASS aparecen en la siguiente hoja. Este admite una capacidad mayor sobre el número de restricciones y variables como se indica en la segunda hoja.
- Este programa utiliza el método de la gran M.

TARJETAS PARA USAR EL PROGRAMA II/SIMPLEX

	Columna 1		
Tarjeta 1	# USER	clave /	________
Tarjeta 2	# RUN	(JR82)	II/SIMPLEX
Tarjeta 3	# DATA	FILE 5	
	-		
	-		
	- Tarjetas de datos (Ver página 3)		
	-		
	-		
Tarjeta final	# END		

NOTAS:

- Este programa está listo para usarse en la computadora B 6700 de CIMAS/CSC.

- La tarjeta 1 es la tarjeta roja obtenida del CIMASS.

- El símbolo "#" significa un carácter inválido. Este se obtiene presionando las teclas MULTIPUNCH Y NUMERIC simultáneamente y perforando los números 1, 2, 3, 4.

- Este programa tiene una capacidad de 30 restricciones y 40 variables incluyendo de holgura y artificiales.

TARJETAS DE DATOS PARA EL PROGRAMA GRANM O II/SIMPLEX

La siguiente información deberá porporcionarse en lo que se indica como tarjetas de datos en las hojas anteriores.

TARJETA DE IDENTIFICACION DEL PROBLEMA.

En esta tarjeta puede usar desde la columna 1 a la 70 para poder dar cualquier identificación que desee dar a su problema.

TARJETA DE DIMENSION Y ETIQUETACION DEL PROBLEMA Y CONTROL PARA CORRER MAS DE UN PROBLEMA.

El usuario debe dar cuatro números enteros con formato (4I10) en la siguiente forma :

Columnas 1 - 10: Número de renglones del problema.

Columnas 11-20: Número de columnas del problema.

Columna 30 : Escriba el número 1 si desea poner etiquetas a los renglones y a las columnas.
Escriba el número 0 en caso contrario.

Columna 40 : Escriba un 1 si desea correr un problema adicional.
Escriba un 0 en caso contrario.

NOTAS:

El número de renglones no incluye la función objetivo.

Si escribe un 1 en la columna 30, el usuario, después de la tarjeta deberá dar el grupo de tarjetas para etiquetas de renglones y el grupo de tarjetas para etiquetas de columnas. Si en lugar de un 1 escribe cero deberá omitir este grupo de tarjetas y pasar a las tarjetas de coeficientes de las variables artificiales en la función objetivo.

Si escribe un 1 en la tarjeta 40 vea las notas generales.

TARJETAS PARA ETIQUETAS DE RENGLONES.

Las etiquetas para identificar a los renglones de las restricciones, pueden tener como máximo 6 caracteres de cualquier tipo.

En una tarjeta puede escribir hasta 7 etiquetas. Estas etiquetas deben ir en las columnas 1-6, 11-16, 21-26, 31-36, 41-46, 51-56, 61-66.

TARJETAS PARA ETIQUETAS DE COLUMNAS (VARIABLES)

Las tarjetas para identificar a las columnas o sea a las variables involucradas en el problema (incluyendo de holgura y artificiales) deberán escribirse de acuerdo a las reglas anteriores para etiquetar renglones.

TARJETAS DE COEFICIENTES DE LAS VARIABLES ARTIFICIALES EN LA FUNCION OBJETIVO.

A cada variable artificial asígnele un 1 y a las variables no artificiales asígnele un 0. Estos números escríbalos en las columnas 10, 20, 30, 40, 50, 60, 70, de acuerdo al orden en que etiquetó a sus variables (columnas)
IMPORTANTE. Esta tarjeta es requerida aún si el problema no tiene variables artificiales.

TARJETAS DE COEFICIENTES DE LAS VARIABLES NO ARTIFICIALES EN LA FUNCION OBJETIVO.

Escriba los coeficientes de la función objetivo con el formato (7 F 10.0). Estos coeficientes debe escribirlos de acuerdo al orden en que etiquetó sus variables (columnas). Los coeficientes de las variables de holgura y artificiales deberá ser cero.

IMPORTANTE : Los coeficientes de la función objetivo deben corresponder al problema de minimizar. Por lo tanto, si su problema es de maximizar multiplique por -1 y considere los coeficientes que resultan como los datos de entrada en este programa.

TARJETAS DE LOS COEFICIENTES DE LA MATRIZ DE RESTRICCIONES.

Cada renglón de restricciones va en una o varias tarjetas, escribiendo los elementos sucesivamente en una tarjeta con un formato (7 F 10.0). Cada vez que proporcione un nuevo renglón debe empezarlo en otra tarjeta.

TARJETAS DE LOS LADOS DERECHOS DE LAS RESTRICCIONES.

Los coeficientes del lado derecho de restricciones se proporcionan sucesivamente en una tarjeta o en caso de ser insuficiente use otra tarjeta. El formato es (7 F 10.0)

TARJETAS PARA INDICAR EL CONJUNTO INICIAL DE VARIABLES BASICAS.

En una tarjeta programe sucesivamente los números de los columnas que van a ser usados como columnas (variables) básicas iniciales. Use formato (7 I 10).

NOTAS GENERALES:

1. El orden de las tarjetas debe ser como el indicado.

2. Si en la TARJETA DE DIMENSION Y ETIQUETACION escribió un 1 en la columna 40 entonces su nuevo problema debe ir después de la TARJETA PARA INDICAR EL CONJUNTO INICIAL DE VARIABLES ARTIFICIALES. Es importante que en el nuevo problema empiece con la TARJETA DE IDENTIFICACION DEL PROBLEMA.

EJEMPLO 1. Considere el problema lineal

$$\max z = x_4 - x_5$$

s.a.

$$\begin{array}{l}
2x_2 - x_3 - x_4 + x_5 \geq 0 \\
-2x_1 + 2x_3 - x_4 + x_5 \geq 0 \\
x_1 - 2x_2 - x_4 + x_5 \geq 0 \\
x_1 + x_2 + x_3 = 1 \\
x_i \geq 0
\end{array}$$

Deberemos multiplicar la función objetivo por - 1 para que el problema sea de minimización y también agregar variables de holgura a las primeras tres restricciones para que -- lleguen a ser igualdades. Con estas observaciones el programa lineal estará en forma estandard, lo cual es una condición para aplicar el programa GRAN M. Si definimos $z' = -z$, - nuestro problema en forma estandard es

$$\min z' = - x_4 + - x_5$$

$$\begin{array}{l}
-2x_2 + x_3 + x_4 - x_5 + s_1 = 0 \\
2x_1 - 2x_3 + x_4 - x_5 + s_2 = 0 \\
-x_1 + 2x_2 + x_4 - x_5 + s_3 = 0 \\
x_1 + x_2 + x_3 = 1 \\
x_i \geq 0; \quad i = 1, 2, \ldots, 5 \\
s_i \geq 0; \quad i = 1, 2, 3
\end{array}$$

Obsérvese que aunque el programa lineal ya está en forma estandard, todavía no está listo para empezar el algoritmo de la Gran M porque en la última restricción no existe una variable que aparezca en esta restricción pero no se encuentre en las otras restricciones. (ie., no se tiene una solución básica factible inmediata). Por lo tanto, deberemos agregar una variable artificial que llamaremos t_1, a la cuarta restricción para así completar nuestra solución básica factible en la cual se inicia el algoritmo. Sin embargo, al introducir esta variable - artificial en la restricción deberemos agregarla en la función objetivo multiplicada por una - cantidad positiva M muy grande. Así nuestro problema resulta ser:

$$\min z' = - x_4 + x_5 + Mt_1$$

$$\begin{array}{l}
-2x_2 + x_3 + x_4 - x_5 + s_1 = 0 \\
2x_1 - 2x_3 + x_4 - x_5 + s_2 = 0 \\
-x_1 + 2x_2 + x_4 - x_5 + s_3 = 0 \\
x_1 + x_2 + x_3 + t_1 = 1 \\
x_i \geq 0; \quad i = 1, 2, \ldots, 5 \\
s_i \geq 0; \quad i = 1, 2, 3 \\
t_1 \geq 0
\end{array}$$

Es conveniente representar el programa lineal en un tablero (o tableau), para poder entender más fácilmente la información que deberemos proporcionar al programa de computadora GRAN M ó II/SIMPLEX. Esta representación aparece abajo

	x_1	x_2	x_3	x_4	x_5	s_1	s_2	s_3	t_1	
Función Obj. (F.O.)	0	0	0	-1	1	0	0	0	M	z'
Renglón 1 (R. 1)	0	-2	1	1	-1	1	0	0	0	0
Renglón 2 (R. 2)	2	0	-2	1	-1	0	1	0	0	0
Renglón 3 (R. 3)	-1	2	0	1	-1	0	0	1	0	0
Renglón 4 (R. 4)	1	1	1	0	0	0	0	0	1	1
						*	*	*	*	

Var. Holgura — Var. Art.

Solución inicial básica factible

Este tablero contiene toda la información necesaria y la notación apropiada para correr el programa GRAN M ó el II/SIMPLEX. A continuación se presenta su codificación para el GRAN M. Para correr el II/SIMPLEX la codificación es idéntica excepto por las tarjetas de control como se mencionó en la explicación de estos programas.

UNAM — HOJA DE CODIFICACION Y/O DATOS FORTRAN — FACULTAD DE INGENIERIA

NUMERO DE INSTRUCCION — COMENTARIOS — PROGRAMA — HOJA — DE — IDENTIFICACION Y SECUENCIA

PROGRAMADOR — FECHA

```
// JØB T
// XEQ GRANM      1
*LØCALINIT,PIVØT,TABNU,RMØVE,CLEAN,TABPR
EJEMPLØ CØN SØLUCIØN ACØTADA DE INVESTIGACIØN DE ØPERACIØNES !
         4         9         1         0
R1        R2        R3        R4
X1        X2        X3        X4        X5        S1        S2
S3        T1
         0         0         0         0         0         0         0
         0         1
0.0       0.0       0.0       -1.0      1.0       0.0       0.0
0.0       0.0
0.0       -2.0      1.0       1.0       -1.0      1.0       0.0
0.0       0.0
2.0       0.0       -2.0      1.0       -1.0      0.0       1.0
0.0       0.0
-1.0      2.0       0.0       1.0       -1.0      0.0       0.0
1.0       0.0
1.0       1.0       1.0       0.0       0.0       0.0       0.0
0.0       1.0
0.0       0.0       0.0       1.0
         6         7         8         9
/*
```

EJEMPLO 2

$$\max z = x_1 + x_2$$

s.a.

$$\begin{aligned} x_1 + x_2 &\geq 1 \\ x_1 + x_2 &\leq 1 \\ -x_1 + x_2 &\leq 1 \\ x_i &\geq 0 \end{aligned}$$

Expresando la función objetivo en términos de minimización e introduciendo variables de holgura, artificiales; el problema es equivalente a :

$$\begin{aligned} &\min (-z) = - x_1 - x_2 + Mt_1 \\ &x_1 + x_2 - s_1 \qquad\qquad + t_1 = 1 \\ &x_1 - x_2 \quad + s_2 \qquad\qquad = 1 \\ &-x_1 + x_2 \qquad + s_3 \qquad = 1 \end{aligned}$$

En forma de tableau:

	x_1	x_2	s_1	t_1	s_2	s_3	
Func. Obj. (F.O.)	-1	-1	0	M	0	0	-z
Renglón 1 (R.1)	1	1	-1	1	0	0	1
Renglón 2 (R. 2)	1	-1	0	0	1	0	1
Renglón 3 (R.3)	-1	+1	0	0	0	1	1
				*	*	*	

Solución básica factible inicial.

UNAM | HOJA DE CODIFICACION Y/O DATOS FORTRAN | FACULTAD DE INGENIERIA

PROGRAMA — HOJA — DE — IDENT

PROGRAMADOR — FECHA — SEC

```
// JØB T
// XEQ GRANM      1
*LØCALINIT,PIVOT,TABNU,RMOVE,CLEAN,TABPR
PRØBLEMA NØ ACØTADØ DE INVESTIGACIØN DE ØPERACIONES !
         3         6         1         0
R1        R2        R3
X1        X2        S1        T1        S2        S3
         0         0         0         1         0         0
-1.0      -1.0      0.0       0.0       0.0       0.0
1.0       1.0       -1.0      1.0       0.0       0.0
1.0       -1.0      0.0       0.0       1.0       0.0
-1.0      1.0       0.0       0.0       0.0       1.0
1.0       1.0       1.0
         4         5         6
/*
```

EJEMPLO 3 Resolver el dual del siguiente par de problemas primal - dual.

Primal

$$\min\ z = 2x_1 - 3x_2$$
$$2x_1 - x_2 - x_3 \geq 3$$
$$x_1 - x_2 + x_3 \geq 2$$
$$x_i \geq 0$$

Dual

$$\max\ w = 3\ \lambda_1 + 2\ \lambda_2$$
$$2\lambda_1 + \lambda_2 \leq 2$$
$$-\lambda_1 - \lambda_2 \leq -3 \rightarrow \lambda_1 + \lambda_2 \geq 3$$
$$-\lambda_1 - \lambda_2 \leq 0$$
$$\lambda_i \geq 0$$

Este dual es equivalente a

$$\min\ (-w) = -3\ \lambda_1 - 2\ \lambda_2 + Mt_1$$
$$2\lambda_1 + \lambda_2 + s_1 = 2$$
$$\lambda_1 + \lambda_2 - s_2 + t_1 = 3$$
$$-\lambda_1 + \lambda_2 + s_3 = 0$$

En forma de tableau; el dual está dado por

	λ_1	λ_2	s_2	s_1	t_1	s_3	
F.O.	-3	-2	0	0	M	0	-w
R1	2	1	0	1	0	0	2
R2	1	1	-1	0	1	0	3
R3	-1	1	0	0	0	1	0
				*	*	*	

UNAM | HOJA DE CODIFICACION Y/O DATOS FORTRAN | FACULTAD DE INGENIERIA

NUMERO DE PERFORACION ... PROGRAMA ... HOJA ... DE ... IDENTIFICACION

PROGRAMADOR ... FECHA ... SECUENCIA

```
// JØB T
// XEQ GRANM        1
*LØCALINIT,PIVOT,TABNU,RMØVE,CLEAN,TABPR
PRØBLEMA SIN SØLUCIØN DE INVESTIGACION DE ØPERACIONES !
         3         6         1         0
R1        R2        R3
L1        L2        S2        S1        T1        S3
         0         0         0         0         1         0
-3.0      -2.0      0.0       0.0       0.0       0.0
2.0       1.0       0.0       1.0       0.0       0.0
1.0       1.0       -1.0      0.0       1.0       0.0
-1.0      1.0       0.0       0.0       0.0       1.0
2.0       3.0       0.0
         /         5         6
/*
```

EJEMPLO 4

$$\max \ z = x_1 - x_2 + x_3 - 3x_4 + x_5 - x_6 - 3x_7$$

s.a.

$$\begin{array}{lr}
3x_3 + x_5 + x_6 & = 6 \\
x_2 + 2x_3 - x_4 & = 10 \\
-x_1 + x_6 & = 0 \\
x_3 + x_6 + x_7 & = 6
\end{array}$$

$$x_i \geq 0$$

$$\min \ (-z) = -x_1 + x_2 - x_3 + 3x_4 - x_5 + x_6 + 3x_7$$

s.a.

$$\begin{array}{lr}
3x_3 + x_5 + x_6 & = 6 \\
x_2 + 2x_3 - x_4 & = 10 \\
x_1 - x_6 & = 0 \\
x_3 + x_6 + x_7 & = 6
\end{array}$$

En forma de Tableau:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
F.O.	-1	1	-1	3	-1	1	3	-z
R1	0	0	3	0	1	1	0	6
R2	0	1	2	-1	0	0	0	10
R3	1	0	0	0	0	-1	0	0
R4	0	0	1	0	0	1	1	6
	*	*			*		*	

UNAM HOJA DE CODIFICACION Y/O DATOS FORTRAN

FACULTAD DE INGENIERIA

NUMERO DE PROPOSICION PROGRAMA HOJA DE IDENTIFICACION

CONTINUACION PROGRAMADOR FECHA SECUENCIA

```
// JØB T
// XEQ GRANM    1
*LØCALINIT,PIVØT,TABNU,RMØVE,CLEAN,TABPR
EJEMPLØ 4
         4         7         1         0
R.1       R.2       R.3       R.4
X1        X2        X3        X4        X5        X6        X7
         0         0         0         0         0         0         0
-1.0      1.0       -1.0      3.0       -1.0      1.0       3.0
0.0       0.0       3.0       0.0       1.0       1.0       0.0
0.0       1.0       2.0       -1.0      0.0       0.0       0.0
1.0       0.0       0.0       0.0       0.0       -1.0      0.0
0.0       0.0       1.0       0.0       0.0       1.0       1.0
6.0       10.0      0.0       6.0
         5         2         1         7
/*
```

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

APLICACIONES DE LA COMPUTADORA A LA SIMULACION Y OPTIMIZACION

PROGRAMACION DINAMICA

DR. VICTOR GEREZ GREISER

MARZO, 1978.

7. Programación dinámica

7.1. Introducción

7.1.1. Teoría Básica.

*En el capítulo 1 se señaló que los métodos de optimización pueden clasificarse en métodos de gradiente y métodos de búsqueda. *En los capítulos 3 y 4 se estudió el método de gradiente. En este capítulo final se estudia el método de optimización conocido con el nombre de programación dinámica un método de optimización de búsqueda. Este último método, todavía más que el de programación lineal requiere del uso de la computadora digital. *Como se trata de una técnica enumerativa, los tiempos de cómputo para este método son en general grandes, así como los requerimientos de memoria. Debido a ello el empleo de esta técnica es un cuanto limitado, a pesar de su extensivo número de aplicaciones potenciales.

*Métodos de optimización de gradiente y búsqueda

*La programación dinámica (p.d.) es un método de búsqueda

*Requiere de mucha memoria y largos tiempos de computación

*En los métodos de optimización estudiados en los capítulos anteriores, lineal, entera y no lineal todo el problema se resuelve en una sola etapa.	*En p.l., programación entera o no lineal se toma una sola decisión múltiple
*En p.d. (programación dinámica) el problema se resuelve en forma secuencial, descomponiendo un problema de toma de decisión múltiple, en una serie de etapas, donde en cada una de ellas, es necesario tomar solamente un número reducido de decisiones o de preferencia solamente una sola.	*En p.d. en cada etapa se toma una sola decisión
*La programación dinámica es una técnica de optimización enumerativa aplicable a problemas con restricciones y funciones objetivo que pueden ser no lineales y regiones factibles no convexas.	*Puede aplicarse a problemas no lineales
*Se aplica en forma natural a problemas que pueden descomponerse en etapas a lo largo del tiempo, pero también puede emplearse en problemas no secuenciales o con estructura en serie.	*El problema debe poder expresarse en forma secuencial

*La programación dinámica se basa en el principio de optimalidad expuesto por R.D. Bellman: (ref. 2)

*El principio de optimalidad de Bellman implica, que en cualquier etapa del proceso de toma de decisión, la política óptima para las etapas subsecuentes solo depende del estado del sistema en dicha etapa y no de la forma en que el sistema llegó a esta etapa.

*Para ilustrar el concepto de optimalidad de Bellman previamente enunciado, considérese el siguiente ejemplo

*Principio de optimalidad de Bellman

"Una serie de decisiones óptimas (políticas óptimas) tiene la propiedad, de que cualquiera que sea el estado inicial y la decisión inicial, las decisiones restantes deben ser óptimas con respecto al estado que resulte de la primer decisión"

7.1.2 Ejemplo.

*La decisión óptima de una etapa en adelanto depende de las subsecuentes y del estado del sistema.

*Ilustración del concepto de optimalidad de Bellmar

Ejemplo 7.1.1

Este problema muestra además el carácter ennumerativo de la técnica de programación dinámica y la forma en que el principio de optimalidad de Bellman permite reducirse número de posibles alternativas por explorar.

*La fig. 7.1.1 muestra una serie de posibles trayectorias entre un punto D y algún punto del litorial. Estos puntos son los puntos A_1, A_2, A_3 y A_4. Los números asociados a segmentos de recta dirigidos muestran la longitud de los diversos segmentos de las posibles trayectorias del punto D al litoral

*Trayectoria más corta de D hasta A_1, A_2, A_3 ó A_4.

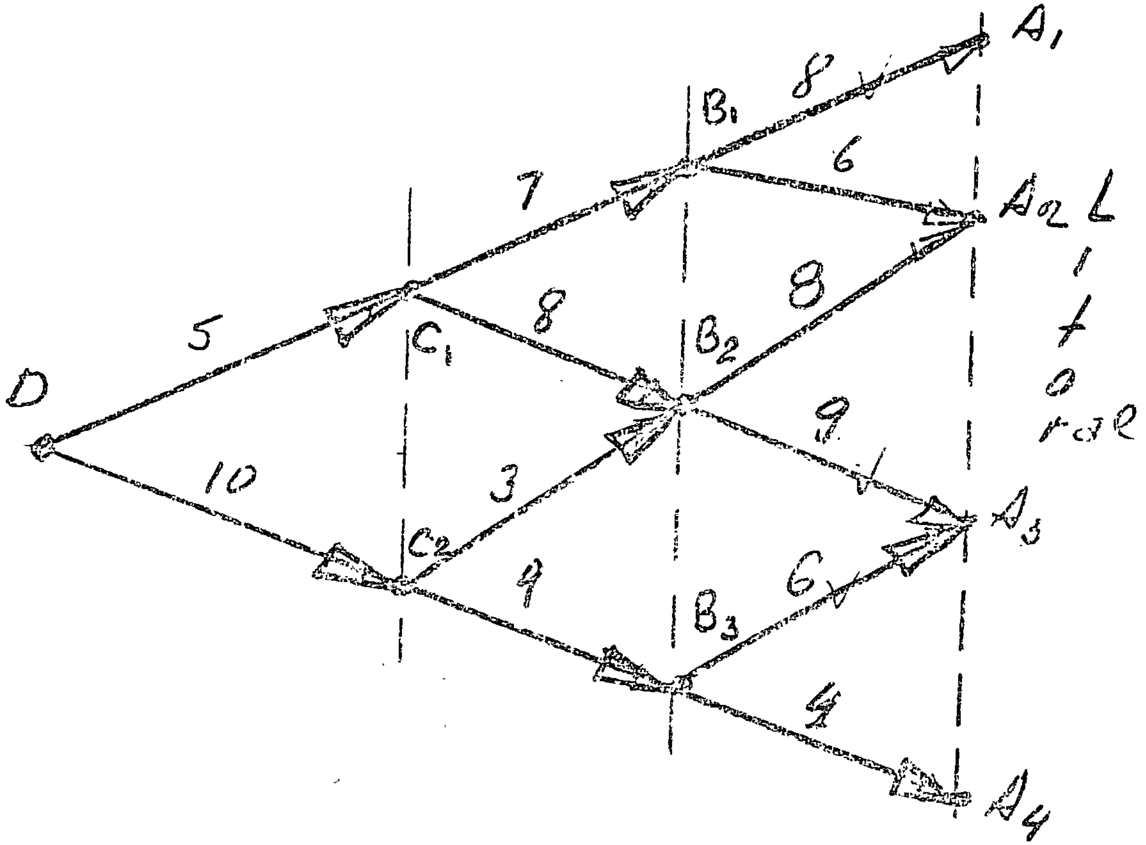


Fig. 7.1.1 Red de caminos de D al litoral

Determine la trayectoria más corta del punto D al litoral empleando la idea de optimalidad.

Solución.

Las posibles trayectorias del punto D al litoral aparecen en la fig. 7.1.2 y son en total 8 con las longitudes indicadas.

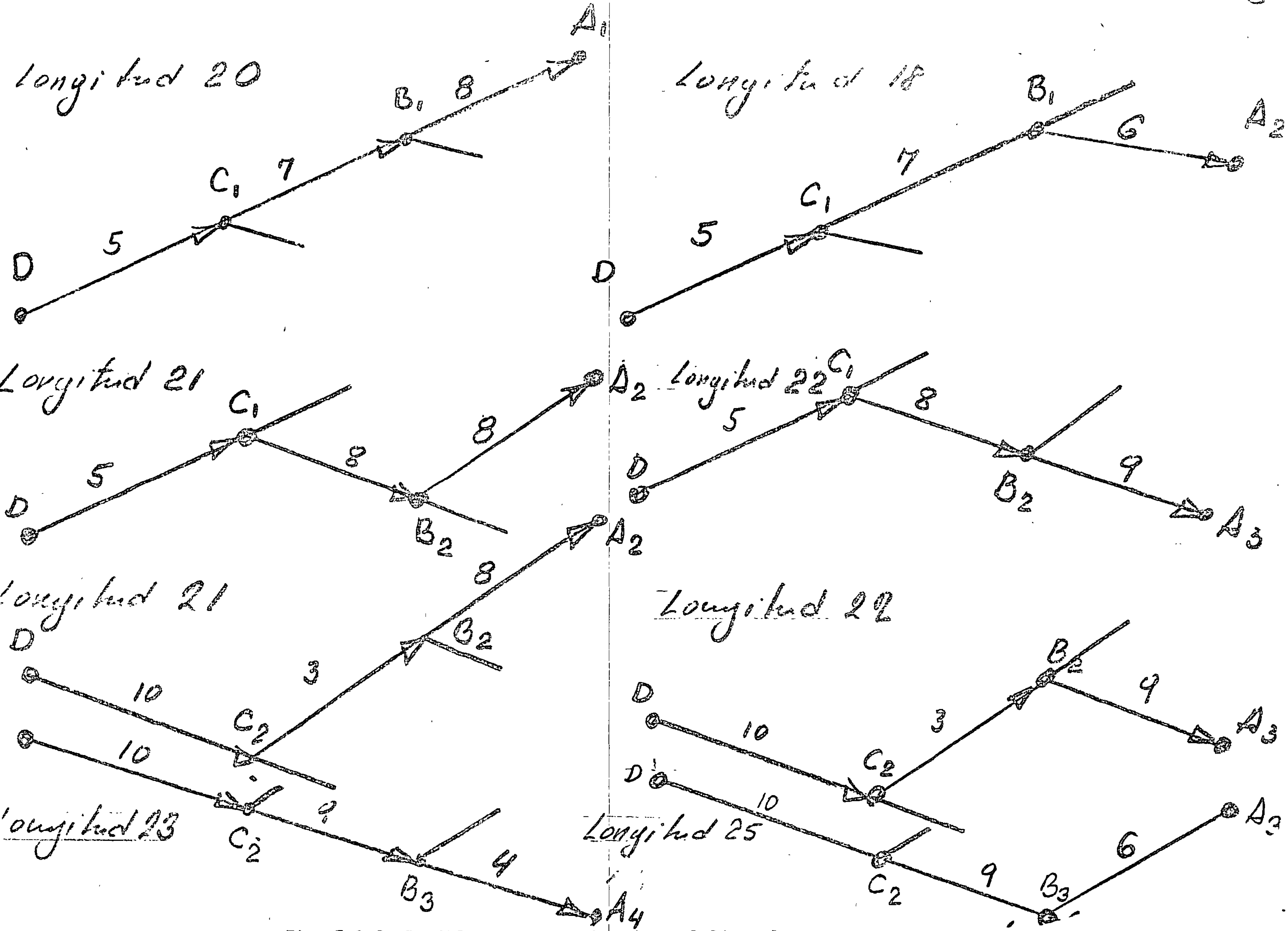


Fig. 7.1.2 Posibles trayectorias de   al litoral.

*Esta figura muestra de inmediato que la trayectoria más corta es la que pasa por los puntos intermedios C_1 B_1 y llega al punto A_2 y tiene una longitud de 18

*Para llegar a este resultado fue necesario explorar 8 alternativas si se hubiese querido explorar las posibles alternativas con ayuda de una computadora, *deberían de haberse conservado en la memoria de la máquina las localidades intermedias, el punto al que llega cada ruta y su longitud, es decir un total de:

y la selección final tendría que haberse realizado buscando un mínimo entre 8 datos. *Una vez localizado este mínimo hubiese sido necesario recuperar de la memoria de la máquina la designación de las localidades intermedias y del destino para poder especificar la trayectoria óptima.

*Trayectoria más corta D C_1 B_1 A_2

Longitud 18

*Se exploraron 8 alternativas

*Datos que deben conservarse en memoria:

$$\left.\begin{array}{l}\text{2 localidades}\\ \text{1 destino}\\ \text{1 longitud}\end{array}\right\} \text{X trayectoria}$$

$(2 + 1 + 1) \times 8 = 32$ datos

*Para especificar la trayectoria óptima es necesario conocer localidades por las que pasa y su destino.

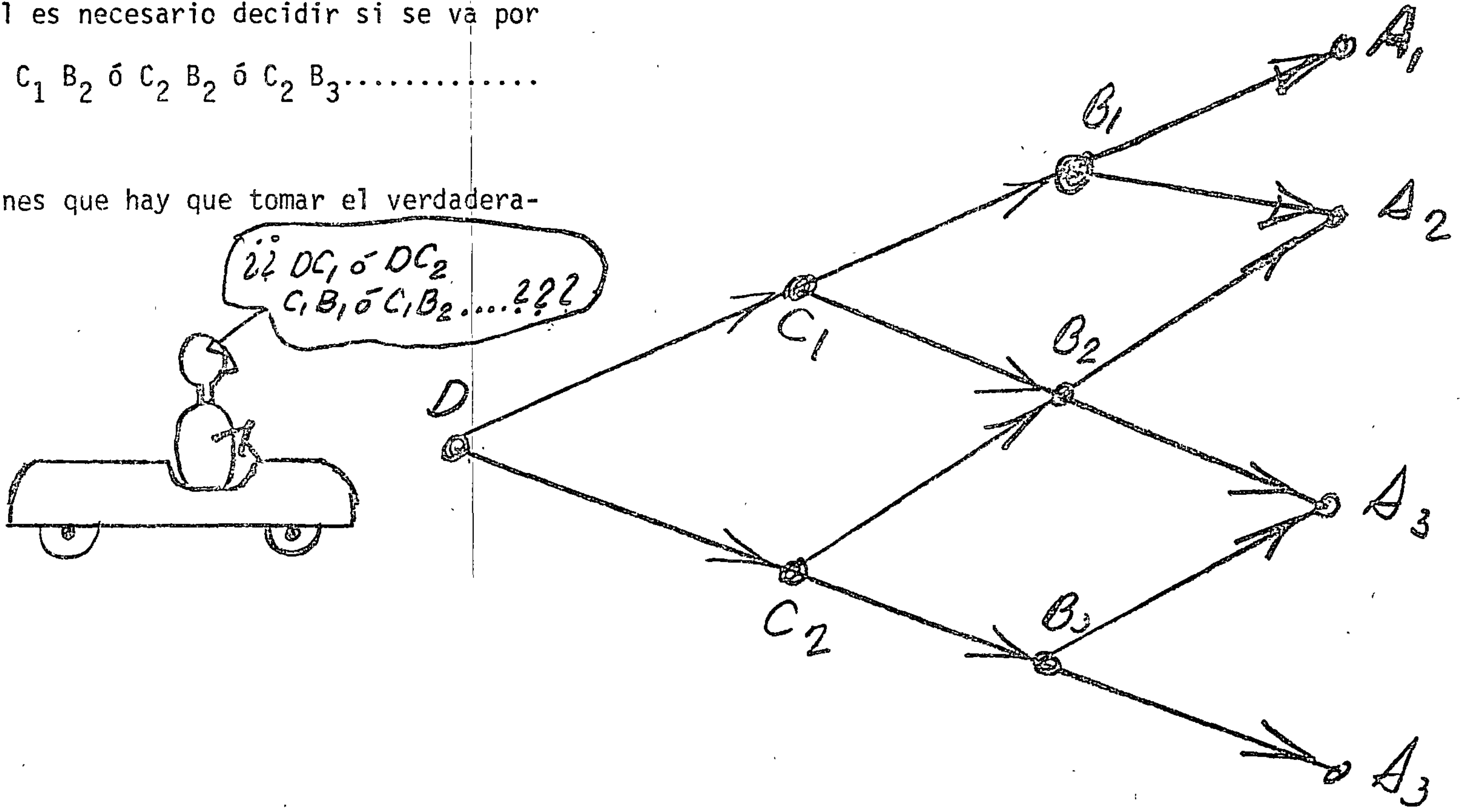
A continuación se muestra como el princípo de optimalidad reduce el número de trayectorias entre las que es necesario buscar el mínimo *Además ilustre como se convierte un problema de decisión múltiple en un problema de uan secuencia de decisiones tomadas una a la vez.

Si al iniciar el recorrido en D es necesario decidir por donde es ir al litoral es necesario decidir si se va por DC_1 ó DC_2 por C_1 B_1 ó C_1 B_2 ó C_2 B_2 ó C_2 B_3.............

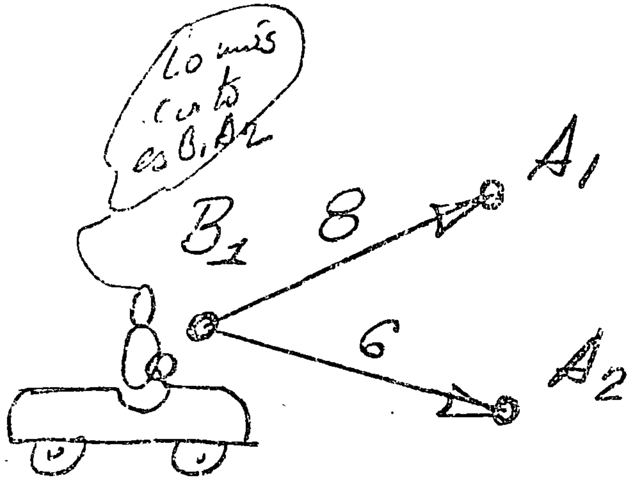
*El número de decisiones que hay que tomar el verdaderamente grande

*Múltiples decisiones $\xrightarrow[\text{dinámica}]{\text{programación}}$

Secuencia de decisiones tomadas una a la vez



*Supóngase por otra parte que se ha llegado a B_1 y hay que decidir cuál es la ruta más corta al litoral. La decisión es simple, evidentemente que por B_1 A_2 que tiene una longitud de 6.



*Si se designa con $F_1(B_1)$ al mínimo de la distancia de la población B_1 al litoral, el comentario anterior permite establecer:

* $F_1(B_i)$
optimo de la primer etapa

$F_1(B_1) = 6.$

y para las poblaciones B_2 y B_3

$F_1(B_2) = 8$

$F_1(B_3) = 4$

* Nótese que en este problema en cada ciudad solo hay dos posibles alternativas :

* Es decir, empleando la literal d para designar descisiones o posibles alternativas:

Desde luego que en otros

* Posibles alternativas en cada población:
ir hacia arriba ó norte
o ir hacia abajo o sur.

* d_i = variable de descición de la i'ésima etapa de solución.

d_i = N (norte) ó S (sur)

problemas las alternativas no estan restringidas a dos.

* La fig. 7.1.3 resume los resultados anteriores. En la figura se han anotado los valores dela trayectoria más corta desdecada ciudad, de donde pudiese iniciarse la última etapa del viaje, primera que se analiza, hasta el destino, además del valor de la descición optima.

trayectorias que no entraran en futuras busquedas

* Se anota el valor de la trayectoria más corta de cada población B_i a la costa y la descición optima d_1^* correspondiente

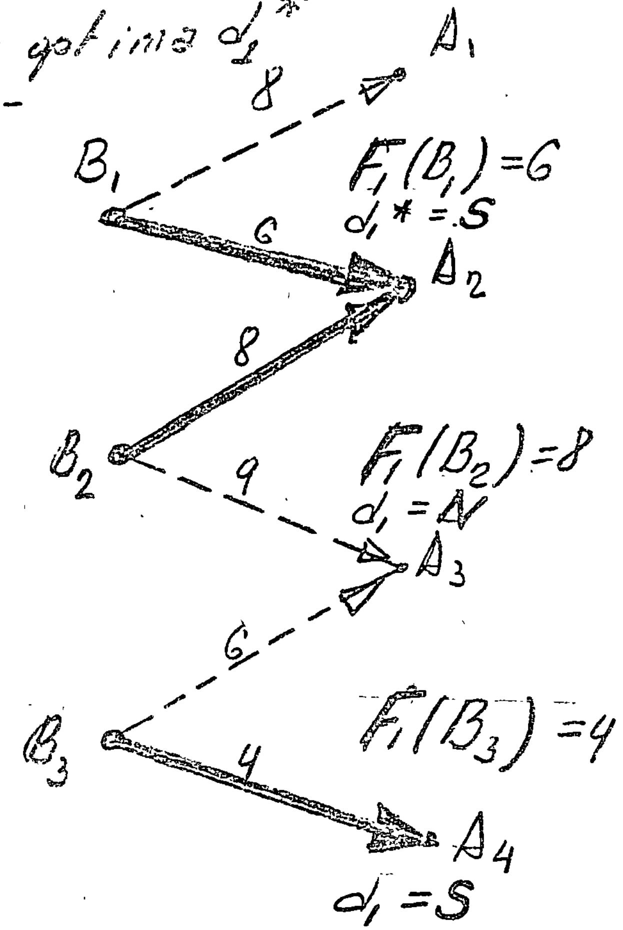


Fig. 7.1.3 Trayectorias más cortas de las poblaciones B_i al litoral.

* Con estos resultados a terminado la primer etapa.

* Fin de la primer etapa.

* Para introducir el modelo formal de programación dinámica, es

* Simbolos y funciones empleadas en p.d.

útil introducir algunos símbolos, variables y relaciones o funciones.

* Cada etapa de solución del problema (en este ejemplo, del viaje) se representa con un bloque.

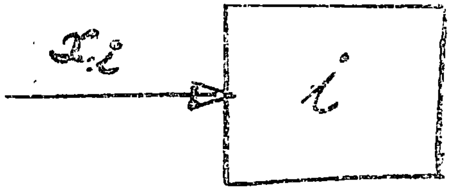
* Cada etapa se inicia con un estado inicial (en el ejemplo del viaje, una población). Este se representa con la letra:
y en el bloque con un segmento de recta que entras

* Además cada etapa termina con otro estado, llamado final (En el ejemplo del viaje, las poblaciones en que termina cada etapa). El estado

* Representación de cada etapa:

* Estado inicial de la i'sima etapa.

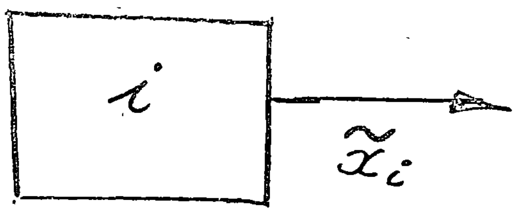
x_i



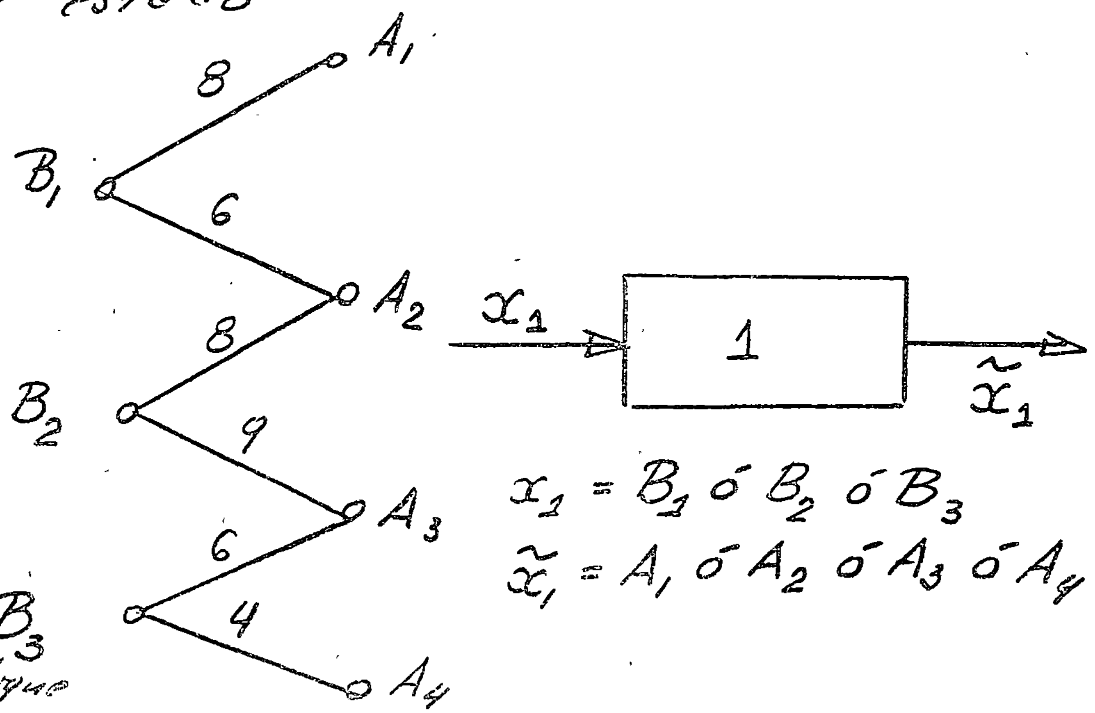
* Estado final de la i'sima etapa

final se representa con el símbolo: $\tilde{x}_i$

y en el bloque con un segmento de recta que sale.



La fig. 7.1.4 muestra la ultima etapa de la red de carreteras, primera que se analiza, el bloque correspondiente, y los posibles valores del estado inicial x_1 y del estado final $\tilde{x}_1$



$x_1 = B_1$ ó B_2 ó B_3

$\tilde{x}_1 = A_1$ ó A_2 ó A_3 ó A_4

Fig 7.1.4 Ultima etapa del viaje y primera que se analiza y representación con un bloque

Además del estado inicial y del estado final es necesario introducir *el <u>beneficio</u> o <u>costo</u> asociado a cada etapa (En el ejemplo, este costo es la longitud del camino entre la población inicial y final de la etapa). Este beneficio o costo se representa con : r_i y depende del estado inicial y la descición que se toma, es decir :

* Beneficio o costo de la i'sima etapa del analisis

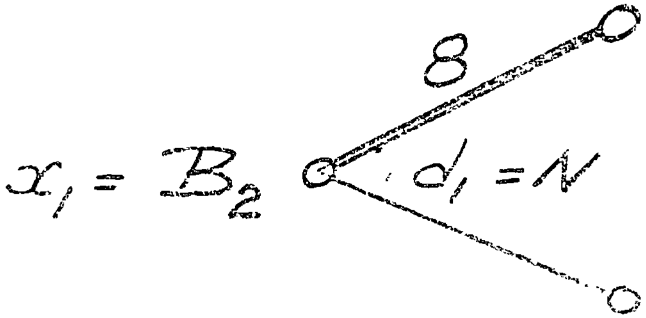
$$r_i = R_i(x_i, d_i)$$

Así por ejemplo si :

$$x_1 = B_2$$

$$d_1 = N \longrightarrow$$

$$r_1(B_2, N) = 8$$



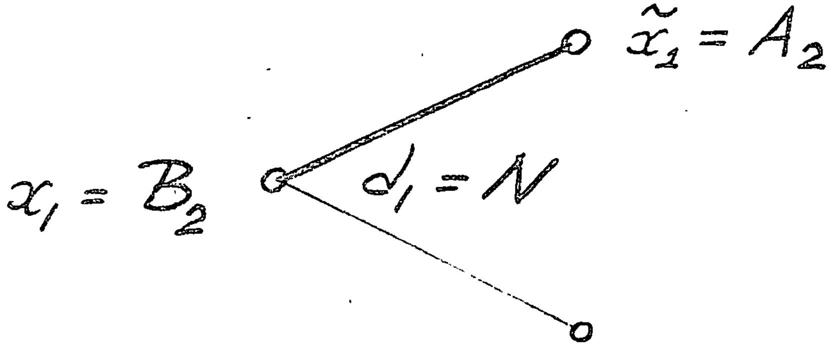
* Finalmente, observese que el estado final $\tilde{x}_i$ de cada etapa depende del estado inicial x_i de la etapa y de la decisión que se toma, asi por ejemplo sea:

* $\tilde{x}_i = T_i(x_i, d_i)$ 14

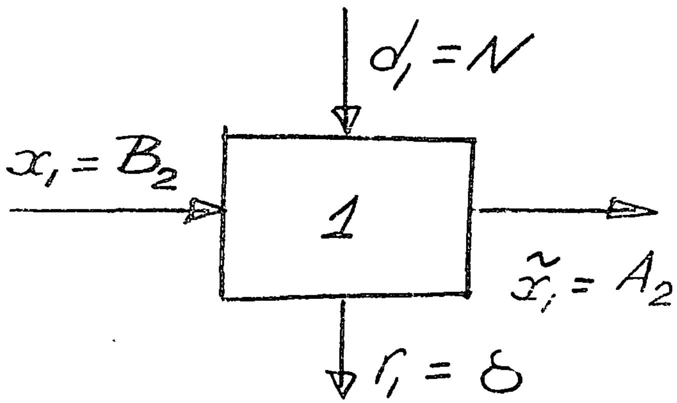
$x_1 = B_2$

$d_1 = N$ ⟶

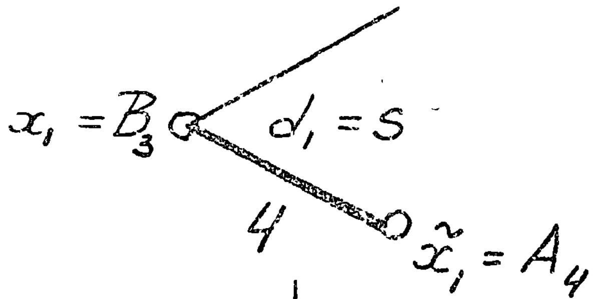
$\tilde{x}_1 = T_1(B_2, N) = A_2$

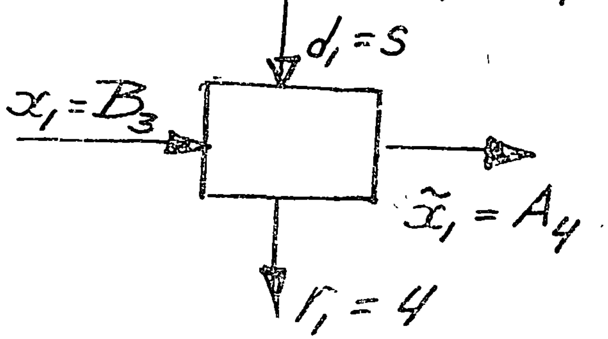


Los resultados anteriores pueden indicarse en el diagrama de bloque de la siguiente manera:



Para otro estado inicial, por ejemplo $x_1 = B_3$ y otra descición, $d_1 = S$, se tendria la representación siguiente:





* En resumen se emplean para establecer formalmente el modelo de p.d. cinco variables y dos funciones, a saber:

* Variables y funciones para representar un modelo de p.d

x_i : Estado inicial

$\tilde{x}_i$: Estado final

d_i : Descición

r_i : Beneficio ó Costo

$r_i = R_i(x_i, d_i)$

$\tilde{x}_i = T_i(x_i, d_i)$

$F_i(x_i)$: Beneficio optimo de las primeras i etapas, correspondiente al estado inicial x_i

La primer etapa de solución

ha terminado con la determinación, para cada posible estado inicial, del costo mínimo (ó beneficio máximo) correspondiente a cada posible estado inicial

En el ejemplo, lo posibles estados iniciales de la primer etapa son:

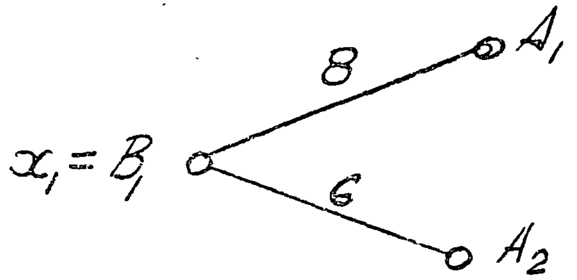
$$x_1 = B_1 \text{ ó } B_2 \text{ ó } B_3$$

y se encontró:

$$F_1(B_1) = 6 \quad d_1^* = S$$
$$F_1(B_2) = 8 \quad d_1^* = N$$
$$F_1(B_3) = 4 \quad d_1^* = S$$

incluyendose además las desciciones que llevaron a esos valores optimos.

Con los simbolos estudiados, la busqueda del optimo correspondiente a la primer etapa puede finalizarse de la siguiente manera para el estado inicial $x_1 = B_1$



$F_1(B_1) = \min\{8, 6\}$

pero: $8 = r_1(B_1, N)$

y: $6 = r_1(B_1, S)$

$\rightarrow F_1(B_1) =$

$\min\{r_1(B_1, N) ; r_1(B_1, S$

* Nótese que la única variable que cambio durante la busqueda es la descición d_1

*Único cambio de valor:

$d_1 = N$ ó S $\longrightarrow$

generalizando se tiene :

$$F_1(B_1) = \min_{d_1}\{r_1(B_1, d_1)\}$$

Empleando la simbologia introducida, la busqueda de la trayectoria optima para la primer etapa puede resumirse en un tabla como la 7.1.1.

Estado inicial x_1	Descición d_1	Longitud de la trayectoria $r_1(x_1,d_1)$	Longitud de la tray. optima $F_1(x_1)$	Descición optima d_1^*
B_1	N S	8 6	6	S
B_2	N S	8 9	8	N
B_3	N S	6 4	4	S

Tabla 7.1.1. Tabla para encontiar el optimo de la primer etapa del problema de p. d.

En resumen para encontiar el optimo en la primer etapa, correspondiente a cada posible estado inicial se empleo:

$$F_1(x_1) = \underset{d_i}{opt}\left(r_1(x_1, d_1)\right) \quad (7.1.1)$$

* El principio de optimalidad establece que si la trayectoria óptima llegase a pasar por B_1, de ahí en adelante sigue de B_1 a A_2 y no de B_1 a A_1, si llegase a pasar por B_2 continuaría a A_2 y si pasase por B_3 continuaría a A_4. Pueden

* El principio de optimalidad establece que cualquiera que fuese la ruta optima, no pasará por los tramos descartados en la primera etapa de analisis (última del recorrido)

descartarse las trayectorias B_1 A_1, B_2 A_3 y B_3 A_4 de futuras alternativas, ya que la ruta más corta no pasaría por esos segmentos.

El problema en este momento es que se ignora si la trayectoria más corta pasa por B_1, B_2 ó B_3. Continuando con la metodología de la programación dinámica se pasa a decidir que hay que hacer al pasar por las poblaciones C_1 y C_2.

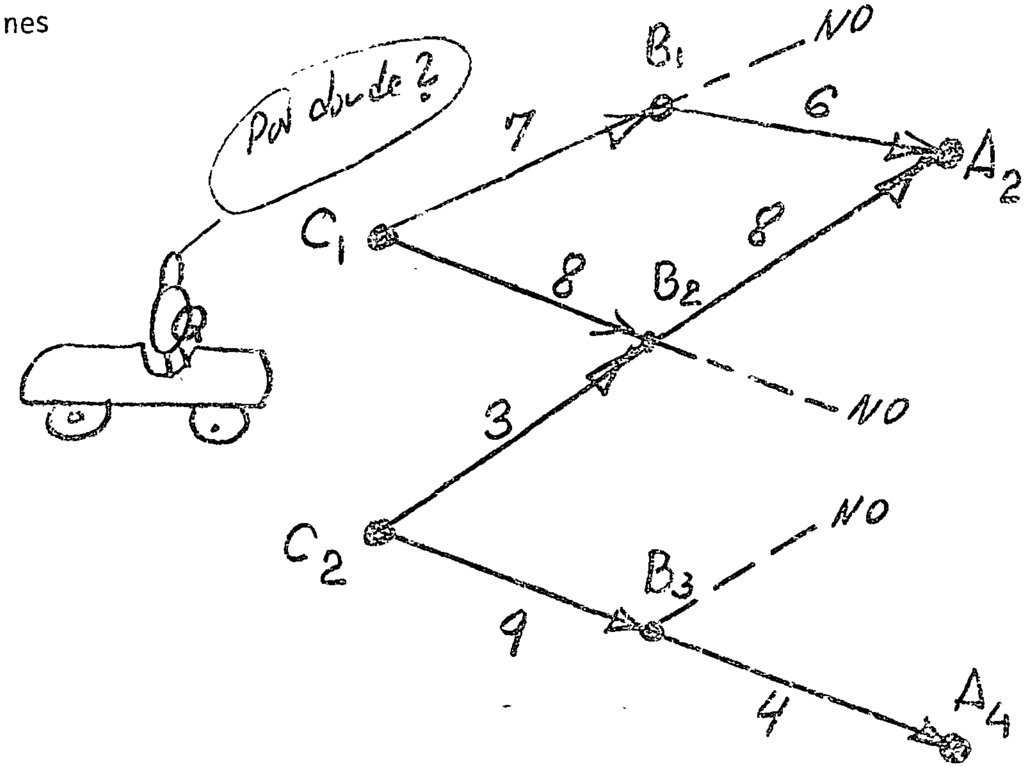
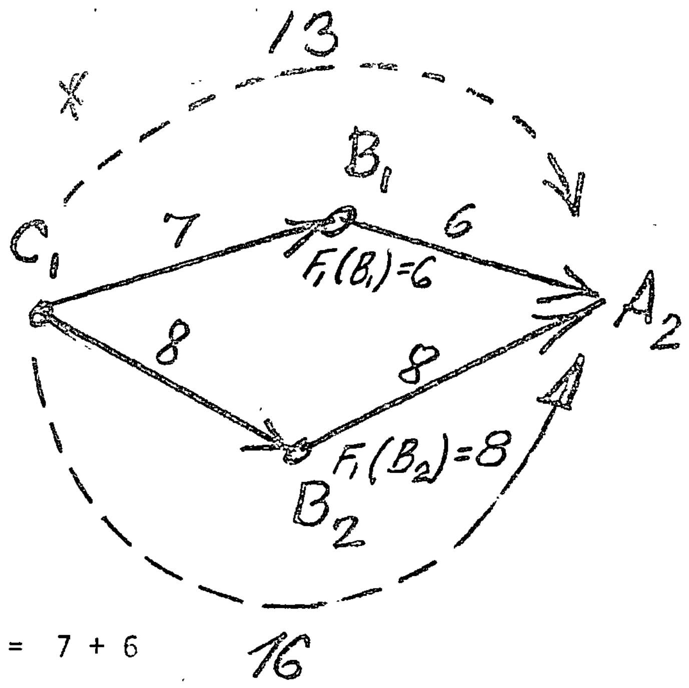


Fig. 7.15 Posibles trayectorias al litoral desde

Si la trayectoria óptima pasase por C_1 de ahí en adelante debe ser la más corta posible hasta el litoral. Para determinar esta trayectoria se hace el siguiente razonamiento:

*Si sigo de C_1 a B_1 la longitud es 7 y de B_1 al litoral lo más corto es $B_1\ A_2$ con 6 de longitud, por lo tanto la ruta $C_1\ B_1$ litoral tiene una longitud de 13. Si se sigue de C_1 a B_2 igual razonamiento lleva a concluir que lo más corto es $C_1\ B_2\ A_2$ con longitud de 16. Obsérvese que la decisión fué entre:

*



$$\begin{aligned} 7 + F_1(B_1) &= 7 + 6 \\ y &= 13 \\ 8 + F_1(B_2) &= 8 + 8 \\ &= 16 \end{aligned}$$

Si se designa con $F_2(C_1)$ al camino más corto de C_1 al litoral puede escribirse:

$$F_2(C_1) = \min\left\{ C_1\ B_1 + F_1(B_1);\ C_1\ B_2 + F_1\ (B_2) \right\} \qquad (7.1.2)$$

y concluirse que

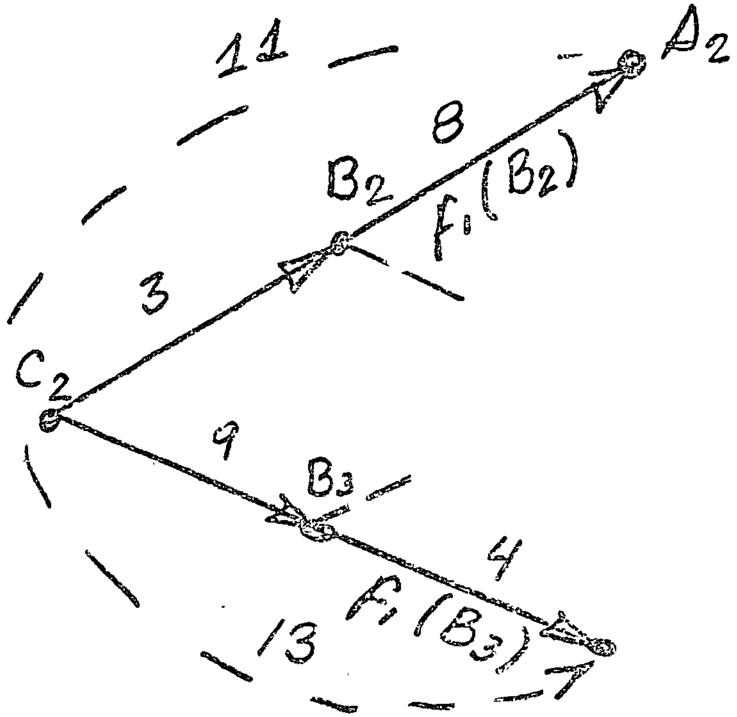
$$F_2(C_1) = \min\ (13,\ 16) = 13$$

el camino más corto de C_2 al litoral, $(F_2(C_2)$, es:

$$F_2(C_2) = \min\left\{ C_2\ B_2 + F_1(B_2);\ C_2\ B_2 + F_1(B_3) \right\} \qquad (7.1.3)$$

y en este caso

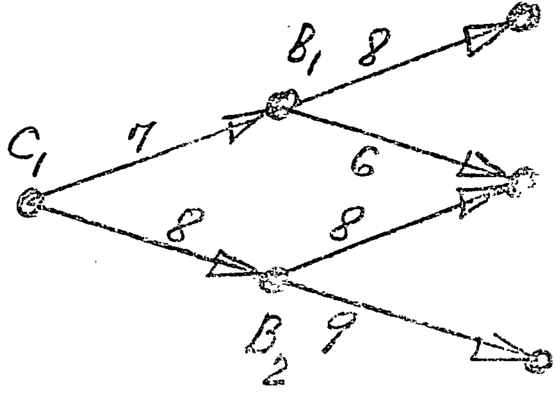
$F_2(C_2) = min(11, 13) = 11$



Nótese que el principio de optimalidad ha simplificado la búsqueda del camino más corto de C_1 ó C_2 al litoral.

*Si no se hubiese empleado el principio de optimalidad, la mínima longitud de C_1 al litoral debería de haberse seleccionado entre los 4 caminos mostrados:

*Si se desconoce el principio de optimalidad el camino de C_1 al litoral requiere analizar:

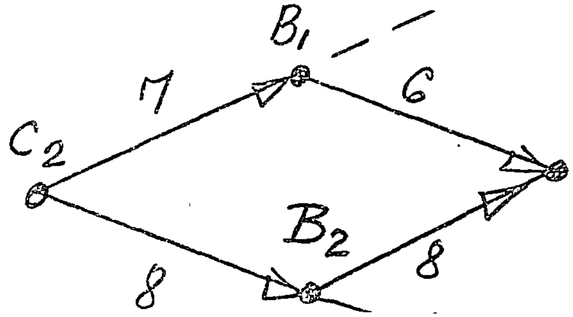


es decir:

$$F_2(C_1) = \min \left\{ 7 + 8,\ 7 + 6,\ 8 + 8,\ 8 + 9 \right\}$$

*Gracias al principio de optimalidad la búsqueda del camino más corto se redujo a 2 posibles trayectorias.

*Por el principio de optimalidad solo requiere buscar entre:



es decir:

$$F_2(C_2) = \min \left\{ 7 + 6,\ 8 + 8 \right\}$$
$$= 13$$

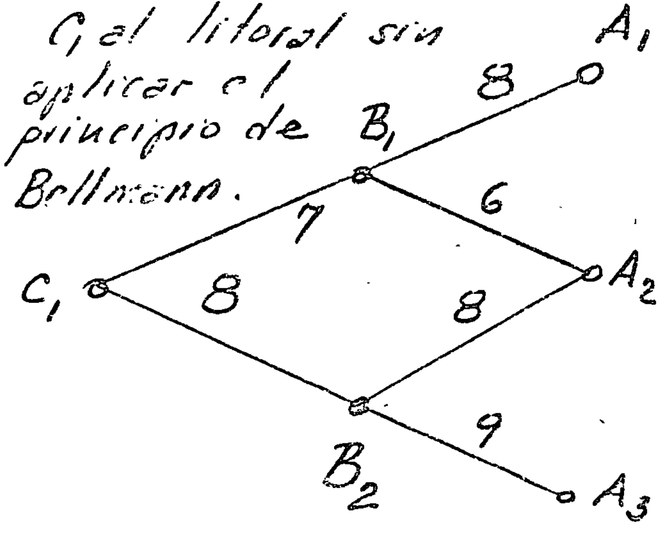
* Con estos resultados termina la segunda etapa

* Fin de la segunda etapa

Antes de continuar se haran unos comentarios sobre las implicaciones que ha tenido el principio de optimalidad en la busqueda del optimo en esta segunda etapa de solución y como se verá, en todas la posteriores de solucion

* Si se se conociese el principio de optimalidad la distancia más corta de la población C_1 a la costa tendría que haberse encontrado de entre las siguientes alternativas:

* Camino más corto de C_1 al litoral sin aplicar el principio de Bellmann.



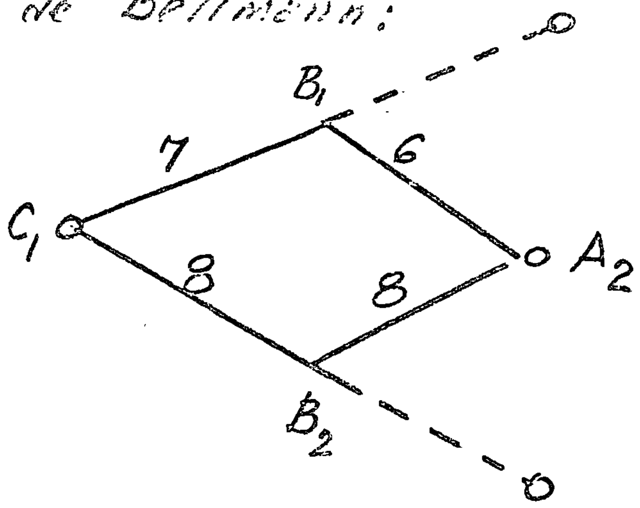
$F_1(C_1)$ es igual al mínimo de las siguientes cuatro sumas:

$$7 + 8 = 15$$
$$7 + 6 = 13$$
$$8 + 8 = 16$$
$$8 + 9 = 17$$

Sin embargo durante la primer etapa ya se descartaron como posibles caminos por los que pudiese peser el óptimo,* los que aparecen con trazo punteado en la fig. 7.1.3, es decir, la busque da se reduce a:

En este ejemplo, al buscar el camino más corto de C_1 al litoral, es decir de dos etapas, el principio de optimalidad de Bellmann permitió reducir las alternativas de búsqueda de

*Camino más corto de C_1 al litoral aplicando el principio de Bellmann:



$F_1(C_1)$ es igual al mínimo de las siguientes dos sumas

$7+6=13$

$8+8=16$

cuatro a dos.

En general, el principio de optimalidad de Bellmann permite reducir en forma sensible, sobre todo en problemas con muchas decisiones, el número de alternativas entre las que hay que seleccionar el óptimo, reduciéndose de esta manera el tiempo de cálculo y las necesidades de memoria de computadora que se requieren para realizar la búsqueda.

*Si se emplea el principio de optimalidad se reduce el número de posibles alternativas

Antes de continuar se haran unos comentarios sobre la información que debe irse conservando al ir resolviendo el problema.

* Para encontrar los resultados correspondientes a la segunda etapa se emplearon las siguientes relaciones:

* Para encontrar los resultados de la etapa dos se empleo:

$$F_2(C_1) = \min\{C_1B_1 + F_1(B_1); \; C_1B_2 + F_1(B_2)\} \quad (7.1.2)$$

$$F_2(C_1) = \min\{C_2 B_2 + F_1(B_2);\ C_2 B_2 + F_1(B_3)\} \quad (7.1.3)$$

* Es decir, fué necesario contar con los siguientes resultados de la etapa primera:

Ademas es necesario retener en memoria las desciciones que llevaron a estas longitudes minimas, que aparecen inmediatamente abajo, (Ver fig. 7.1.3) de las longitudes optimas. Es decir, de la información de la tabla 7.1.1. construida durante la primer etapa fué necesario conservar lo que aparece en la tabla 7.1.2, hasta terminar con la segunda etapa.

* Resultados de la primer etapa empleados para calcular la segunda:

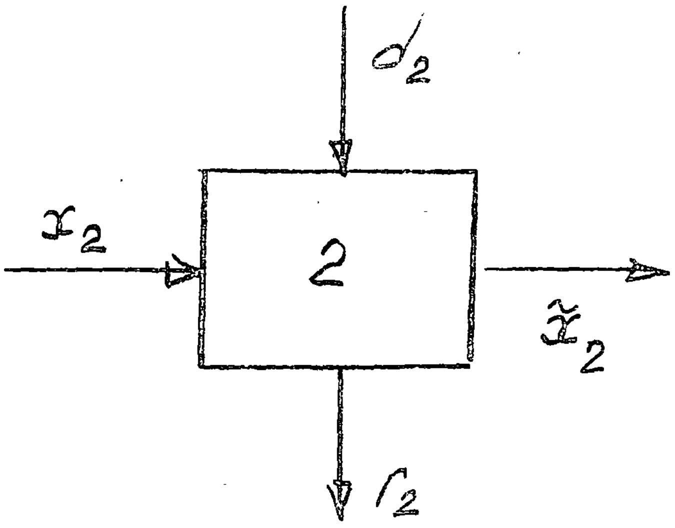
$F_1(B_1)$; $F_1(B_2)$ y $F_1(B_3)$

$d_1^* = S$; $d_1^* = N$ y $d_1^* = S$

Estado inicial x_1			Longitud de la tray. optima $F_1(x_1)$	Descrición optima d_1^*
B_1			6	S
B_2			8	N
B_3			4	S

Tabla 7.1.2 Valores encontrados en la primer etapa, que se requieren para encontrar el optimo durante la segunda etapa.

Empleando la simbología introducida anteriormente puede establecerse el siguiente diagrama de bloque para la segunda etapa de solución, que se acaba de analizar.



* Con los siguientes valores posibles del estado inicial x_2 y final $\tilde{x}_2$ de la etapa segunda.

Si se compara los valores posibles del estado final de la segunda etapa $\tilde{x}_2$ con los iniciales de la primer etapa x_1 se tiene que:

El resultado anterior puede generalizarse:

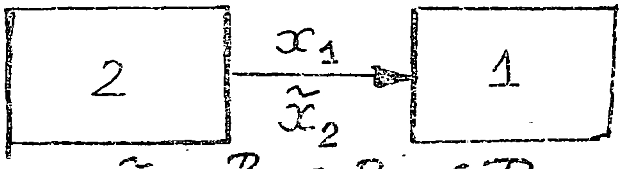
Es decir:

*Valores posibles del estado inicial y final de la segunda etapa.

$x_2 = C_1$ ó C_2

$\tilde{x}_2 = B_1$ ó B_2 ó B_3

$x_1 = B_1$ ó B_2 ó B_3



$\tilde{x}_2 = B_1$ ó B_2 ó B_3

$\tilde{x}_2 = x_1$

$$\tilde{x}_i = x_{i-1} \qquad (7.1.4)$$

El estado final de una etapa de solución coincide con el inicial de la anterior; es decir la estructura del problema es serie.

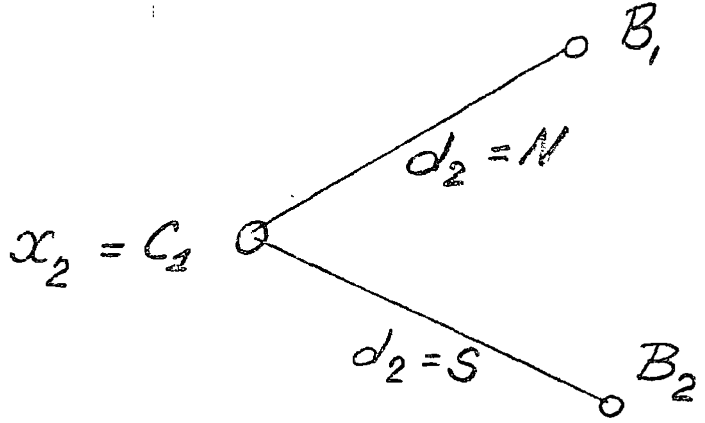
Para poder continuar con el establecimiento formal del algoritmo de p.d. se analiza la relación (7.1.2)

$$F_2(C_1) = \min\{C_1B_1 + F_1(B_1);\ C_1B_2 + F_1(B_2)\} \quad (7.1.$$

empleando las variables y funciones ya señaladas. Recuerdese que:

$$C_1B_1 = r_2(C_1,\ d_2 = N)$$

$$C_1B_2 = r_2(C_1,\ d_2 = S)$$



ya que la distancia C_1B_1 corresponde a la segunda etapa, iniciada en el estado $x_2 = C_1$ y tomando la descición de ir al norte es decir; $d_2 = N$. y en forma similar para C_1B_2

Sustituyendo: $F_2(C_1) = \min\{r_1(C_1, N) + F_1(B_1); r_1(C_1, S) + F_1(B_2)\}$ (7.1.

Además, dada la estructura serie B_1 y B_2 son estados iniciales de la etapa 1 y finales de la dos (3

$$\tilde{x}_2 = x_1$$

Recordando la relación entre estados finales e iniciales:

$$\tilde{x}_2 = T_2(x_2, d_2)$$

Como parte de la etapa dos se tiene para los estados finales $\tilde{x}_2 = B_1$ ó B_2

$$B_1 = T_2(x_2 = C_1; d_2 = N)$$

$$B_2 = T_2(x_2 = C_1; d_2 = S)$$

Sustituyendo estas relaciones en (7.1.5) y no dando un valor especifico ni al estado inicial x_2 ni a la descicion d_2 se tiene:

$$F_2(x_2) = \min_{d_2}\{r_2(x_2, d_2) + F_1(T_2(x_2, d_2))\} \quad (7.1.6,$$

Nótese que en la relación $T_2(x_2, d_2) = \tilde{x}_2 = x_1($ permite calcular el estado final de la etapa dos, con estado inicial x_2, y correspondiendo a la descición d_2 Este estado es el inicial de la etapa anterior es decir:

$$\tilde{x}_2 = x_1$$

Puede por lo tanto tambien escribirse:

$$F_2(x_2) = \min_{d_2} \{ r_2(x_2, d_2) + F_1(\tilde{x}_2 = x \qquad (7.1.7)$$

* También debe observarse que para tomar la descición, para alcanzar el optimo, solo se varia la variable de descición d_2, tal como aparece en la fórmula (7.1.6)

* Al calcular $F_2(x_2)$, beneficio optimo, so se varia d_2.

En forma explicita la relación (7.1.6) ó (7.1.7) establece:

Para cada posible estado inicial de la etapa dos, debe buscarse el optimo correspondiente a las dos primeras etapas, entre los posibles valores de la suma de los siguientes términos:

a) el costo o beneficio de la etapa dos $r_2(x_2, d_2)$

b) el optimo de la etapa uno, correspondiente al estado final de la etapa dos inicial de la etapa uno, que resulte de la decición tomada en la etapa dos

Estas sumas deben encontrarse para todas las posibles desciciones d_2.

La tabla 7.1.3 ilustra la aplicación de este algoritmo para la etapa dos

Posibles estados iniciales de la etapa dos x_2	Posibles valores de la descición d_2	Longitudes de la etapa dos $r_2(x_2,d_2)$	Estados finales de la etapa dos $\tilde{x}_2=x_1$	Valores de la tabla correspondiente a la etapa uno		$C_2(x_2,d_2)$ + $F_1(\tilde{x}_2)$	Distancia optima para las dos primeras etapas $F_2(x_2)$	Desciciones optimas	
				$F_1(\tilde{x}_2=x_1)$	d_1^*			d_1^*	d_2^*
C_1	N	7	B_1	6	S	13	13	S	N
	S	8	B_2	8	N	16			
C_2	N	3	B_2	8	N	11	11	N	N
	S	9	B_3	4	S	13			

Tabla 7.1.3. Tabla para encontrar el óptimo durante la segunda etapa (columnas a doble marco contienen información que se empleará en la etapa tres)

Para encontrar el camino más corto de D al litoral, puede extenderse el algoritmo (7.1.7) a una tercer etapa, es decir:

$$F_3(x_3) = \min_{d_3}\{r_3(x_3, d_3) + F_2(\tilde{x}_3 = x_2)\} \quad (7.1.8)$$

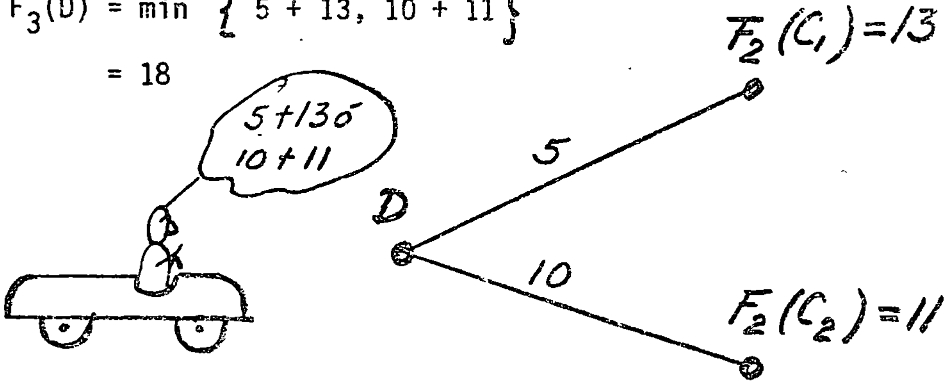
Por lo tanto el camino óptimo de D al litoral $F_3(D)$ estará dado por:

$$F_3(D) = \min \left\{ DC_1 + F_2(C_1); \; DC_2 + F_2(C_2) \right\}$$

es decir:

$$F_3(D) = \min \left\{ 5 + 13, \; 10 + 11 \right\}$$

$$= 18$$



* El camino más corto de D al litoral tiene una longitud de 18.

* Camino más corto al litoral → longitud = 18

Además de conocer la longitud del camino es necesario encontrar que poblaciones cruza.

Para saber por donde pasa dicho camino e recuerdese que:

* De este razonamiento se concluye que el camino pasa por C_1 y de ahí en adelante sigue por la trayectoria cuya longitud es:

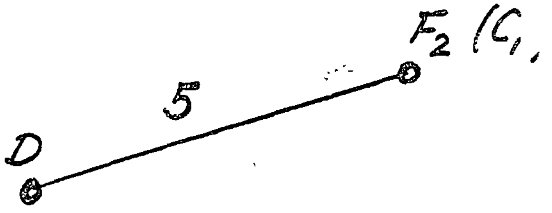
reconstruyendo el proceso se sabe que:

*es decir el camino lleva de C_1 a B_1 y finalmente se sabe que

y*este trayecto de 6 de longitud y que parte de B_1 llega a A_2. *Por lo tanto el camino más corto es:

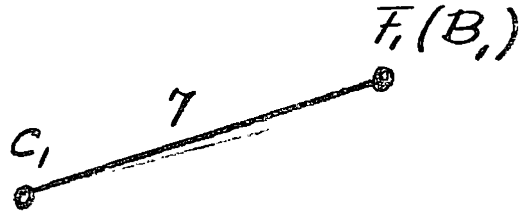
tal como se había concluido con la búsqueda exhaustiva ilustrada en la fig. 7.1.2

Antes de formalizar este método de optimización estableciendo un algoritmo de búsqueda conviene hacer hincapié sobre los aspectos más relevantes de este procedimiento.



$18 = 5 + F_2(C_1)$

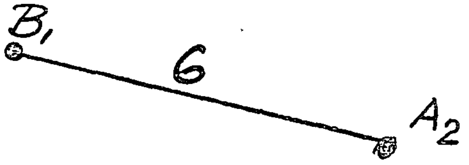
* Paso por C_1



$F_2(C_1)$

$F_2(C_1) = 7 + F_1(B_1)$

*Pasa por C_1 B_1

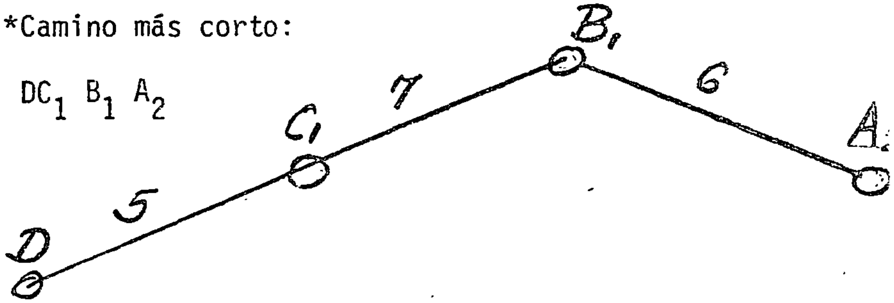


$F_1(B_1) = 6$

$*F_1(B_1) = B_1\ A_2$

*Camino más corto:

$DC_1\ B_1\ A_2$

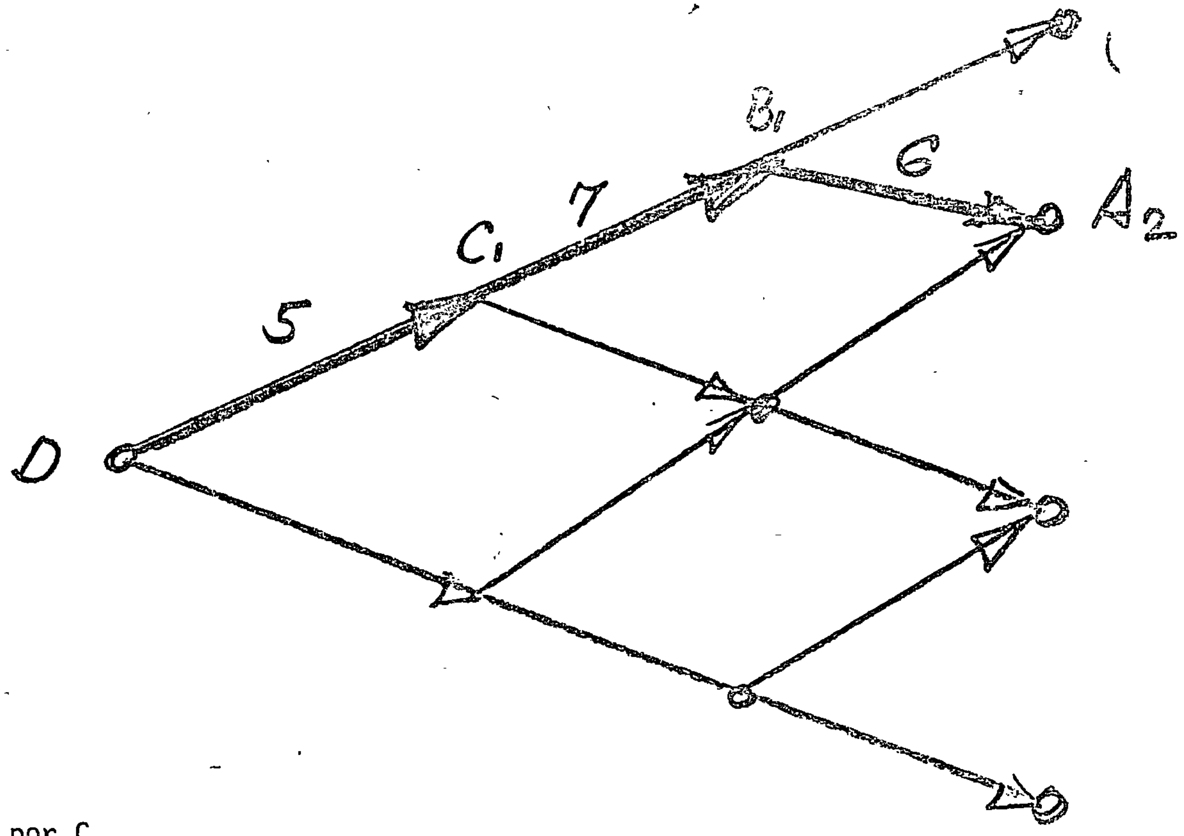


*Aspectos relevantes de la p.d.

Se trata de un procedimiento de enumeración de alternativas y posterior búsqueda del óptimo entre éstas. El principio de optimalidad reduce el número de posibles alternativas entre las que se encuentra el máximo ó mínimo reduciendo el tiempo de cómputo y los requisitos de memoria de maquinaria. *A pesar de esta reducción, estos últimos son la principal limitante que se presenta al aplicar esta metodología.

Recuérdese que la trayectoria óptima en este ejemplo fué reconstruída a partir del dato sobre longitud de dicha trayectoria de 18, en la forma que esquematiza la figura 7.1.5

*Los requisitos de memoria limitan la aplicación de la programacinón dinámica



18 = 5 + $F_2(C_1)$ → la trayectoria pasa por C_1

$F_2(C_1)$ = 13 = 7 + $F_1(B_1)$ → la trayectoria pasa por B_1

$F_1(B_1)$ = 6 → la trayectoria pasa por A_2

Fig. 7.1.5 Obtención de la trayectoria óptima.

* Hasta no haber encontrado el óptimo es necesario conservar la siguiente información:

* además hay que saber como se originaron estas trayectorias de longitud mínima, así por ejemplo se sabe que:

*Hay que conservar en memoria:

$F_2(C_1)$ y $F_2(C_2)$

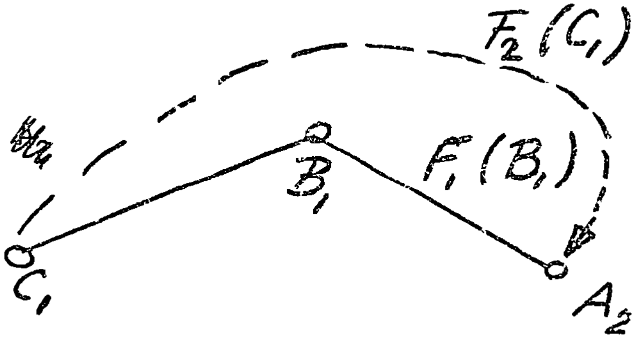
*¿Que trayectorias son?

$F_2(C_1) = C_1\ B_1 + F_1(B_1)$

es decir la trayectoria de longitud $F_2(C_1)$ de C_1 llega a B_1 y:

*parte de C_1 y llega a B_1, y de B_1 al litoral tiene

como longitud $F_1(B_1) = 6$ de B_1 a A_2



este camino llega a A_2

Además de recordar que $F_2(C_1) = 13$

Es necesario tener en memoria que esta trayectoria que

parte de C_2 y tiene una longitud de 13 pasa por :C_1 B_1 A_2

En resumen es necesario conservar en memoria los siguientes datos:

Longitud de 13 de la trayectoria óptima $F_2(C_1)$ que pasa por C_1 y recorrido del camino C_1 B_1 A_2 y longitud de 11 de $F_2(C_2)$ que pasa por C_2 B_2 A_2

Esta información aparece en doble marco en la tabla 7.1.3.

El lector puede vislumbrar facilmente que en problemas de mayor dimensión la cantidad de datos que hay que conservar en memoria puede llegar a ser muy grande.

*Finalmente conviene aclarar que en la búsqueda exhaustiva fué necesario explorar las 8 posibles trayectorias que aparecen en la fig. 7.1.2 para encontrar el óptimo.

*Búsqueda exhaustiva: 8 alternativas

Aplicando el principio de optimalidad la búsqueda no tiene que incluir las trayectorias que aparecen punteadas en la fig. 7.1.3 en la primer etapa. Durante la siguiente etapa se descartan además los que aparecen en la fig. 7.1.6.

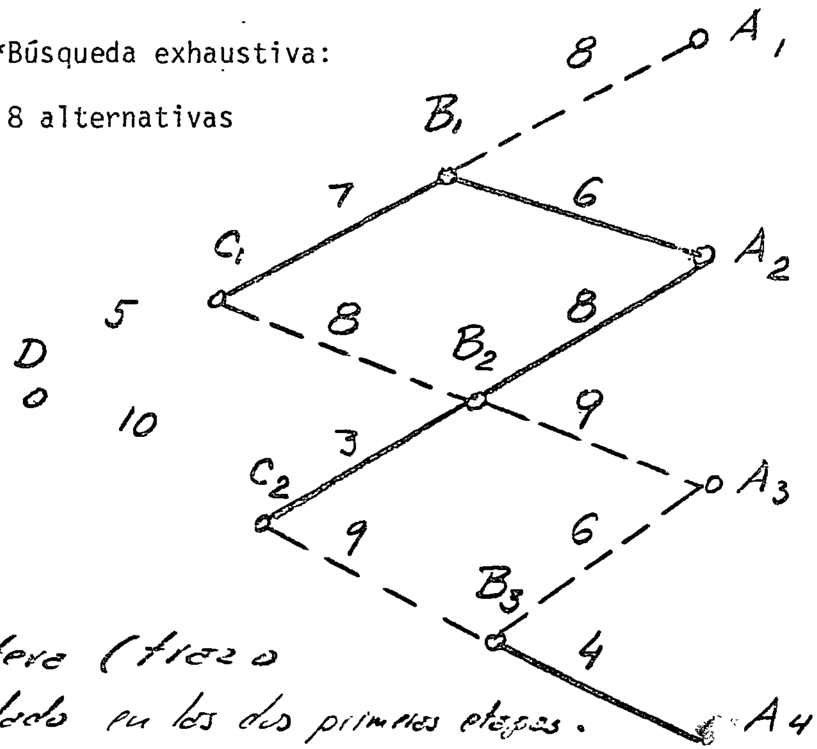


Fig. 7.1.6 Tramos de carretera (trazo punteado) que se han descartado en las dos primeras etapas.

En problemas de gran dimensión la reducción de alternativas entre las que es necesario buscar el óptimo es mucho más sensible que en este ejemplo.

La tabla 7.1.4 muestra el procedimiento de busqueda durante la tercer etapa

Posibles valores iniciales de la etapa tres x_3	Posibles valores de la descrición d_3	Longitudes de la etapa tres $r_3(x_3,d_3)$	Estados finales de la etapa tres $\tilde{x}_3=x_2$	Valores optimos de encontrados durante la etapa dos (tabla 7.1.3)			$r_3(x_3,d_3)$ + $F_2(\tilde{x}_3)$	Optimos			
				Distancias $F_2(\tilde{x}_3=x_2)$	Descriciones			Distancia $F_3(x_3)$	Descriciones		
					d_1^*	d_2^*			d_1^*	d_2^*	d_3^*
D	N	5	C_1	13	S	N	18	18	S	N	N
	S	10	C_2	11	N	N	21				

Tabla 7.1.4 Tabla para encontrar el optimo durante la tercer etapa

[illegible] Pg 390

2 LLOCATION PROCESSES

!.1 GENERAL

; allocation problem is an example of a single-period, static (deterministic) *ltiactivity process* that can be transformed by dynamic programming into a ıltistage process with a finite number of stages. The allocation problem nonstrates that a "stage" need not be related to time.

Allocation of fixed resources among some potential recipients is a major)blem of organizations. How to define and measure the return on allocated estment seems to be one of the major obstacles for the decision maker. ıenever the returns can be quantified in some way, the problem can be 'sented as a programming problem. In the rare case where the return (or ,ective function) is linear, the problem may be presented as a linear-)gramming problem. However, in many real cases the return function is ılinear, or even discontinuous. Dynamic programming offers a way to ıdle complicated nonlinear allocation problems (for example, problems h discrete or nonconvex objective functions). (See Simone [27]).

.2 ONE DIMENSIONAL ALLOCATION PROCESSES—FORMULATION

ıne-dimensional allocation problem involves the following characteristics ! assumptions:

Characte-ristic

Characteristics

1. A certain (limited) quantity x of an economic resource (such as labor, land, machines, or water) is to be allocated.
2. The resource is used in the production of certain products or services.
3. The limited resource can be used in two or more alternative ways. Each such possible way is called an *activity*.
4. Each single activity, where the resource is used, yields a *return* (or reward).
5. The process may involve stochastic elements (which will not be discussed here).

Assump-tion

4ssumptions

1. Returns from different allocations can be *compared*; that is, they can be measured in a common unit (dollar, utility, share of the market, and so on).
2. The return from any allocation is independent of the allocations to other activities.
3. The total return that can be obtained is the sum of individual returns: that is, additivity or common unit is essential.

'he problem is how to allocate the resource to the alternative activities o users) in such a way that the total return (or reward) is maximized.

a. General Formulation

The most general mathematical formulation of the one-dimensional problem involves maximizing an objective function (total return) as follows:

$$\max R(x_1, x_2, \ldots, x_n) = g_1(x_1) + g_2(x_2) + \cdots + g_n(x_n) \tag{8.3}$$

subject to one constraint—that is, to the total availability (capacity) of the resource x, which may assume any positive value:

$$x_1 + x_2 + \cdots + x_n = x = \sum_{i=1}^{n} x_i \tag{8.4}$$

where

x is the total amount of the resource
x_i is the quantity of the resource assigned to the ith activity[6]
$g_i(x_i)$ is the return from the ith activity
n = number of possible activities (n may assume any positive integer value)

If the objective function is linear, then we have a linear-programming problem. However, for the more general case, where the objective function can take any form, we can use the following dynamic-programming approach: First, we have to convert the problem to a dynamic process, which is done as follows:

1. The first allocation goes to the nth activity.
2. Then, we allocate to activity $(n-1)$.
3. Then, we allocate to activity $(n-2)$.
4. We proceed in this manner until, finally, we allocate to activity $n-(n-1) = (n-n+1)$, which is the first activity. This successive allocation results in a dynamic process.

b. Recurrence Relation

We now proceed to illustrate how the allocation problem given in (8.3) and (8.4) can be solved by developing a sequence of recurrence relations.

Let $f_n(x)$ = optimal return from an allocation of x to n activities. Assuming $g_i(0) = 0$ for all i, which is usually the case, it follows that

$$f_n(0) = 0 \tag{8.5}$$

Also

$$f_1(x) = g_1(x) \tag{8.6}$$

Let x_n be the allocation made to the nth activity, where $0 \le x_n \le x$. The remaining quantity $x - x_n$ will be used in the $(n-1)$ remaining activities.

Let us assume that we have already allocated $x - x_n$ to $(n-1)$ activities in the *optimal* (best) way. This allocation yielded a return of $f_{n-1}(x - x_n)$. By

[6] We use here the notation x_i, instead of the x_j used previously, to be in line with most literature on dynamic programming.

definition, the return from the allocation of x_n to the nth activity is $g_n(x_n)$. Thus, the total return of allocating x to all n activities is

$$R = g_n(x_n) + f_{n-1}(x - x_n) \tag{8.7}$$

Usually there are several ways of allocating x_n to the nth activity. Obviously the optimal one is that which maximizes R; that is,

$$f_n(x) = \max R = \max_{0 \le x_n \le x} \{g_n(x_n) + f_{n-1}(x - x_n)\} \tag{8.8}$$

for $n = 2, 3, \ldots,$ and $x \ge 0$. Equation (8.8) is known as the *recurrence relation*.

Thus, the allocation problem given by Equations (8.3) and (8.4) has been reduced from the original problem to that of (8.8). We now have two subproblems:

1. How to maximize (8.8).
2. How to obtain $f_{n-1}(x - x_n)$.

Answering these two problems will enable us to solve Equation (8.8), which is equivalent to the original problem (remember that $g_n(x_n)$ is given). The answer to subproblem 1 is that (8.8) is maximized by one of several possible techniques of maximization (see 8.1.6). The answer to subproblem 2 is that we can write

$$f_{n-1}(x) = \max_{0 \le x_{n-1} \le x} \{g_{n-1}(x_{n-1}) + f_{n-2}(x - x_{n-1})\} \tag{8.9}$$

where x_{n-1} is the amount allocated to the $(n-1)$th activity. Note that, as in (8.8), we are asked in (8.9) to maximize a function in which it is required that we find $f_{n-2}(x - x_{n-1})$. Here too we can use one of the maximization techniques, and we shall again need the results of the previous stage, $f_{n-3}(x - x_{n-2})$. We must continue in this process backward until we arrive at the second stage. In the second stage we will use the optimal results of the first stage $f_1(x)$. But $f_1(x)$ is given according to Equation (8.6). Thus we can solve the entire process. Note that $f_1(x)$ determines $f_2(x)$, $f_2(x)$ determines $f_3(x)$, and so on.

8.2.3 AN ILLUSTRATIVE EXAMPLE

The management of the ABC Corporation is considering the allocation of 5 million dollars among its three plants. It was decided that the allocation per plant will be either 0, 1, 2, 3, 4, or 5 million dollars.

Each plant submitted the expected returns for the next 4 years corresponding to different levels of money invested. The data on expected returns were discounted to time zero and are given in Table 8.2. For example, an initial investment of \$2 million in plant A will yield a total discounted return of \$0.5 million. (In this case, the assumed returns were: 0.1 million after 1 year, 0.15 million after 2 years, 0.2 million after 3 years, and 0.15 million after 4 years. Using an interest rate of 6 percent, this stream of returns, discounted back to time zero, yields \$0.5 million.) In Table 8.2, we read 0.5 million in the column for plant A and in the row where $K = 2$. All numbers under the columns for plants A, B, and C are subject to similar interpretation. Let T (\$5 million) be the total amount available for allocation and let K designate the total amount that is set for allocation at a given stage.

Table 8.2

AMOUNT ALLOCATED (K), IN MILLIONS OF DOLLARS	EXPECTED RETURN $g_i(K)$		
	PLANT A	PLANT B	PLANT C
0	0	0	0
1	0.2	0.3	0.4
2	0.5	0.4	0.8
3	1.9	1.2	1.1
4	1.8	2.0	1.5
5	2.5	2.2	2.0

Our problem is to determine the optimal allocation to each plant in order to maximize the overall expected return.

Solution: In order to visualize this *single-period* allocation problem as a sequential problem, let us view stage 1 as the decision point at which allocation to plant A alone is determined; and stage 2 as the decision point at which allocation to plants A and B (and none to C) is determined; and stage 3 as the decision point at which allocation to all three plants is determined.[7] In each stage we have six possible *states*—that is, plants or combination of plants that may receive 0, 1, 2, 3, 4, or 5 million dollars.

Let x_i be the amount allocated to the ith plant, and $g_i(x_i)$ be the return (reward) expected from the allocation of x_i to the ith plant. The problem of maximizing the total expected return ER may be stated as

$$\max ER = \sum_{i=1}^{3} g_i(x_i) \tag{8.10}$$

Since we face limited resources, our objective function is subject to the constraint

$$\sum_{i=1}^{3} x_i \le T \tag{8.11}$$

where $x_i \ge 0$ and is an integer, and T is the total amount we have for allocation.

[7] We have arbitrarily made stage 1 as the decision point at which allocation to plant A is determined, and stage 2 as the decision point at which allocation to plants A and B is determined, and so on. Of course, stage 1 could have been designated as the decision point at which allocation to B (or C) is determined. Depending on the first allocation decision, stage 2 would be the decision point at which allocation to either A and B, or A and C, or B and C, is made.

Let K be the amount considered for allocation (K is not necessarily equal to T; in some cases the best policy may turn out to be an allocation of $K<T$). The expected return is a function of K and the relationship can be formally expressed as

$$f_n(K)=\max_{0\leq xn\leq K}\{g_n(x_n)+f_{n-1}(k-x_n)\} \tag{8.12}$$

where $f_n(K)$ is the maximum (optimal) return.

We will now present a step-by-step dynamic programming solution to this problem.

Stage 1

In this stage we consider the allocation of K dollars to plant A only and we designate this amount by x_1. The optimal expected return $f_A(K)$ in this case is:

$$f_A(K)=\max_{0\leq x_1\leq K}\{(g_1(x_1)\} \tag{8.13}$$

where $g_1(x_1)$ is the expected return from investment in plant A.[8] These values are given in the column for plant A in Table 8.2. We have, in our case,

$$\begin{array}{lll} g_1(0)=0 & \text{and} & f_A(0)=0 \\ g_1(1)=0.2 & & f_A(1)=0.2 \\ g_1(2)=0.5 & & f_A(2)=0.5 \\ g_1(3)=1.9 & & f_A(3)=1.9 \\ g_1(4)=1.8 & & f_A(4)=1.9 \\ g_1(5)=2.5 & & f_A(5)=2.5 \end{array}$$

Table 8.3 gives a complete enumeration of $g_1(x_1)$ and $f_A(K)$ values for stage 1 analysis.

Table 8.3

	x_1						
K	0	1	2	3	4	5	$f_A(K)=\max_{0\leq x_1\leq K}\{g_1(x_1)\}$
0	0						0
1	0	0.2					0.2
2	0	0.2	0.5				0.5
3	0	0.2	0.5	1.9			1.9
4	0	0.2	0.5	1.9	1.8		1.9
5	0	0.2	0.5	1.9	1.8	2.5	2.5

[8] Equation) differs from Equation (8.6) because $g_1(x_1)$ is *not* a monotonically increasing functi....

Note that when we set $K=4$ and search

$$f_A(4)=\max_{0\leq x_1\leq K}\{g_1(x_1)\}$$

we find that $g_1(0)=0$, $g_1(1)=0.2$, $g_1(2)=0.5$, $g_1(3)=1.9$, and $g_1(4)=1.8$. In other words, the expected return is maximized for $x_1=3$; and thus $f_A(4)=g_1(3)=1.9$, which is the highest value among $g_1(0)$ through $g_1(4)$. This means that we should allocate only \$3 million of the \$4 million set for allocation. The reader can further notice that an allocation of \$3 million will yield more than the investment of \$4 million, which is an unusual, but possible, case.

Stage 2

At this stage we split the dollars to be allocated (K) between plants A and B. We allocate a certain amount x_2 to B and the remaining $(K-x_2)$ to A. Note that from our analysis of stage 1 we already know the optimal allocation to A for any amount K.

Since x_2 is the amount allocated to plant B, and $(K-x_2)$ to plant A, the optimal allocation for the two-stage process, according to the principle of optimality, is given by

$$f_{AB}(K)=\max_{0\leq x_2\leq K}\{g_2(x_2)+f_A(K-x_2)\} \tag{8.14}$$

where $g_2(x_2)$ is the return from investment in plant B. The values here can be computed by enumeration, as illustrated below.

For $K=0$:

$$f_{AB}(0)=0$$

For $K=1$ we have the following alternatives:

(a) Allocate 1 to plant B and 0 to plant A

$$g_2(1)+f_A(0)=0.3+0=0.3$$

(b) Allocate 0 to plant B and 1 to plant A

$$g_2(0)+f_A(1)=0+0.2=0.2$$

Note that the values $g_2(x_2)$ are obtained from the "plant B" column of Table 8.2, whereas the values $f_A(K)$ are taken from the results of stage 1 as summarized in Table 8.3. We can write this manipulation as

$$f_{AB}(1)=\max_{0\leq x_2\leq 1}\begin{Bmatrix} g_2(1)+f_A(0)=0.3 \\ g_2(0)+f_A(1)=0.2 \end{Bmatrix}=0.3$$

Similarly, for $K=2$ we get

$$f_{AB}(2)=\max_{0\leq x_2\leq 2}\begin{Bmatrix} g_2(0)+f_A(2)=0.0+0.5=0.5 \\ g_2(1)+f_A(1)=0.3+0\text{:} \quad).5 \\ g_2(2)+f_A(0)=0.4+0.0=0.4 \end{Bmatrix}=0.5$$

In this case we have two equivalent alternatives.

For $K=3$ we get

$$f_{AB}(3)=\max_{0\le x_2\le 3}\left\{\begin{array}{l} g_2(0)+f_A(3)=0\ \ +1.9=1.9\\ g_2(1)+f_A(2)=0.3+0.5=0.8\\ g_2(2)+f_A(1)=0.4+0.2=0.6\\ g_2(3)+f_A(0)=1.2+0\ \ =1.2\end{array}\right\}=1.9$$

Clearly, the best allocation is 3 to plant A.

For $K=4$ we get

$$f_{AB}(4)=\max_{0\le x_2\le 4}\left\{\begin{array}{l} g_2(0)+f_A(4)=0\ \ +1.9=1.9\\ g_2(1)+f_A(3)=0.3+1.9=2.2\\ g_2(2)+f_A(2)=0.4+0.5=0.9\\ g_2(3)+f_A(1)=1.2+0.2=1.4\\ g_2(4)+f_A(0)=2.0+0\ \ =2.0\end{array}\right\}=2.2$$

The best allocation is 1 to plant B and 3 to plant A.

For $K=5$ we get

$$f_{AB}(5)=\max_{0\le x_2\le 5}\left\{\begin{array}{l} g_2(0)+f_A(5)=0\ \ +2.5=2.5\\ g_2(1)+f_A(4)=0.3+1.9=2.2\\ g_2(2)+f_A(3)=0.4+1.9=2.3\\ g_2(3)+f_A(2)=1.2+0.5=1.7\\ g_2(4)+f_A(1)=2.0+0.2=2.2\\ g_2(5)+f_A(0)=2.2+0\ \ =2.2\end{array}\right\}=2.5$$

To sum up, for the second stage we get the following optimal allocation policy:

$f_{AB}(0)=0$: allocate nothing
$f_{AB}(1)=0.3$: 1 to plant B and 0 to plant A
$f_{AB}(2)=0.5$: either 1 to B and 1 to A, or 0 to B and 2 to A
$f_{AB}(3)=1.9$: 0 to B and 3 to A
$f_{AB}(4)=2.2$: 1 to B and 3 to A
$f_{AB}(5)=2.5$: 0 to B and 5 to A

A summary of the analysis for stage 2 is given in Table 8.4.

Stage 3

Here we divide dollars to be allocated among all three plants. We allocate a certain amount x_3 to plant C, and allocate the remaining $(K-x_3)$ between plants A and B according to the optimal policy $f_{AB}(K)$ derived in stage 2. The optimal policy for the three-stage process, according to the principle of optimality, is given by

$$f_{ABC}(K)=\max_{0\le x_3\le K}\{g_3(x_3)+f_{AB}(K-x_3)\} \qquad (8.15)$$

where $g_3(x_3)$ is the return from investment in plant C. Let us enumerate values of f_{ABC} corresponding to different allocation policies for specified levels of K.

Table 8.4

	x_2						$f_{AB}(K)=\max\limits_{0\le x_2\le K}\{g_2(x_2)+f_A(K-x_2)\}$
K	0	1	2	3	4	5	
0	0						0
1	0.2	0.3					0.3
2	0.5	0.5	0.4				0.5
3	1.9	0.8	0.6	1.2			1.9
4	1.9	2.2	0.9	1.4	2		2.2
5	2.5	2.2	2.3	1.7	2.2	2.2	2.5

For $K=0$, obviously, $f_{ABC}(0)=0$

For $K=1$ we get

$$f_{ABC}(1)=\max_{0\le x_3\le 1}\left\{\begin{array}{l} g_3(0)+f_{AB}(1)=0\ \ \div 0.3=0.3\\ g_3(1)+f_{AB}(0)=0.4+0\ \ =0.4\end{array}\right\}=0.4$$

For $K=2$ we get

$$f_{ABC}(2)=\max_{0\le x_3\le 2}\left\{\begin{array}{l} g_3(0)+f_{AB}(2)=0\ \ +0.5=0.5\\ g_3(1)+f_{AB}(1)=0.4+0.3=0.7\\ g_3(2)+f_{AB}(0)=0.8+0\ \ =0.8\end{array}\right\}=0.8$$

For $K=3$ we get

$$f_{ABC}(3)=\max_{0\le x_3\le 3}\left\{\begin{array}{l} g_3(0)+f_{AB}(3)=0\ \ +1.9=1.9\\ g_3(1)+f_{AB}(2)=0.4+0.5=0.9\\ g_3(2)+f_{AB}(1)=0.8+0.3=1.1\\ g_3(3)+f_{AB}(0)=1.1+0\ \ =1.1\end{array}\right\}=1.9$$

For $K=4$ we get

$$f_{ABC}(4)=\max_{0\le x_3\le 4}\left\{\begin{array}{l} g_3(0)+f_{AB}(4)=0\ \ +2.2=2.2\\ g_3(1)+f_{AB}(3)=0.4+1.9=2.3\\ g_3(2)+f_{AB}(2)=0.8+0.5=1.3\\ g_3(3)+f_{AB}(1)=1.1+0.3=1.4\\ g_3(4)+f_{AB}(0)=1.5+0\ \ =1.5\end{array}\right\}=2.3$$

For $K=5$ we get

$$f_{ABC}(5)=\max_{0\le x_3\le 5}\left\{\begin{array}{l} g_3(0)+f_{AB}(5)=0\ \ +2.5=2.5\\ g_3(1)+f_{AB}(4)=0.4+2.2=2.6\\ g_3(2)+f_{AB}(3)=0.8+1.9=2.7\\ g_3(3)+f_{AB}(2)=1.1+0.5=1.6\\ g_3(4)+f_{AB}(1)=1.5+0.3=1.8\\ g_3(5)+f_{AB}(0)=2.0+0\ \ =2.0\end{array}\right\}=2.7$$

Table 8.5 Analysis for stage 3

K	x_3						$f_{ABC}(K) = \max\limits_{0 \leq x_3 \leq K} \{g_3(x_3)+f_{AB}(K-x_3)\}$
	0	1	2	3	4	5	
0	0						0
1	0.3	0.4					0.4
2	0.5	0.7	0.8				0.8
3	1.9	0.9	1.1	1.1			1.9
4	2.2	2.3	1.3	1.4	1.5		2.3
5	2.5	2.6	2.7	1.6	1.8	2.0	2.7

The analysis for stage 3 is summarized in Table 8.5. Table 8.6 summarizes the values under the last columns of Tables 8.3, 8.4, and 8.5. Several ements of valuable information can be retrieved from the data in Table 8.3 ırough 8.6. First we note that for every value of K, one can immediately etermine the optimal expected return and identify the plants among which ıe investment must be divided. Second, we can determine the marginal exected return for a given allocation policy as K is increased in units of $1 iillion. Third, as soon as we have chosen a specific value for K, we can utilize ıe information of Table 8.6 to determine the optimal allocation policy.

Searching for the highest value of Table 8.6, we note that the optimal ›pected return is $2.7 million. Hence the investment must be allocated etween plants A, B, and C. An examination of Table 8.5 (for A, B, and C) ıows that an expected return of $2.7 million requires that $x_3=2$, or $2 iillion must be allocated to plant C, and $3 million must be allocated etween plants A and B. We now examine Table 8.4 and note that the ptimal allocation of $3 million between A and B requires $x_2=0$ (allocate 0 to ant B) and $3 million to plant A. Hence our overall optimal allocation in iis case is: Allocate $2 million to plant C, allocate $0 million to plant B, ıd allocate $3 million to plant A.

Table 8.6 Optimal solution

K	$f_A(K)$	$f_{AB}(K)$	$f_{ABC}(K)$
0	0	0	0
1	0.2	0.3	0.4
2	0.5	0.5	0.8
3	1.9	1.9	1.9
4	1.9	2.2	2.3
5	2.5	2.5	2.7

Some Comments and Generalizations

1. For m plants the ıecurrence relation will be
$$f_n(K) = \max_{0 \leq x_n \leq K} \{(g_n(x_n)+f_{n-1}(K-x_n)\} \qquad n=2, 3, \ldots, m \tag{8.16}$$
where n designates the stage number.
2. Sensitivity analysis can be easily performed. For example, if management cut the available funds to $4 million then it is easy to observe that the best policy is to allocate $1 million to C, and $3 million to A at a profit of $2.3 million {policy $f_{ABC}(4)$}.
3. The dynamic-programming solution can give us indirectly the second-best alternative. In our case, if we allocate $5 million, we get for the second-best allocation: 1 to C, 1 to B, and 3 to A, at an expected profit of $2.6 million (see Table 8.5) Similarly, we can get the third-best solution, and so on.
4. Adding a new plant to the problem merely adds an additional stage.
5. It is customary to summarize the results of the optimal policies of all stages in one table, as shown in Table 8.7.

Table 8.7 Tabular solution for the allocation problem

K	x_1	$f_A(K)$	x_2	x_1	$f_{AB}(K)$	x_3	x_2	x_1	$f_{ABC}(K)$
0	0	0	0	0	0	0	0	0	0
1	1	0.2	0	1	0.3	1	0	0	0.4
2	2	0.5	0	2[a]	0.5	2	0	0	0.8
3	3	1.9	0	3	1.9	0	0	3	1.9
4	4	1.9	1	3	2.2	1	0	3	2.3
5	5	2.5	0	5	2.5	2	0	3	2.7

[a] For stage 2, and $K=2$; $x_2=1$, $x_1=1$ is an alternative solution.

8.2.4 MULTIDIMENSIONAL ALLOCATION PROCESSES

a. General

The one-dimensional process involved an allocation of one resource subject to one constraint. Multidimensional allocation processes involve one of the following:

1. Allocation of one resource subject to two or more constraints.
2. Allocation of two or more resources subject to two or more constraints.

We shall state here the two simplest possible cases—namely, the allocation of one resource subject to two constraints, ~~and the allocation of two resources~~ subject to two constraints.

b. Allocation of One Resource to n Activities Subject to Two Constraints

Such an allocation problem can be presented as:

$$\max_{x_i} ER(x_1, x_2, \ldots, x_n) = g_1(x_1)+g_2(x_2)+\cdots+g_n(x_n) \tag{8.17}$$

subject to:

$$\sum_{i=1}^{n} a_i(x_i) \le x$$

$$\sum_{i=1}^{n} b_i(x_i) \le y$$

and

$$x_i \ge 0$$

where

1. x and y are the capacities of the two constraints (equivalent to b_1 and b_2 in the general linear-programming formulation).
2. x_i is the quantity of the resource allocated to activity i.
3. $g_i(x_i)$ is the return from the ith activity.
4. $a_i(x_i)$ and $b_i(x_i)$ are monotonically increasing functions of x_i (they approach ∞ when $x_i \to \infty$).

The general recurrence relation in this case is

$$f_n(x, y) = \max_{\substack{x_n,\ s/t \\ a_n(x_n) \le x \\ b_n(x_n) \le y}} \{g_n(x_n) + f_{n-1}[x - a_n(x_n), y - b_n(x_n)]\} \tag{8.18}$$

Example: A ship is to be loaded with several items varying in *weight*, *ize*, and *value* (all known). Also the ship's maximum capacity in tonnage ind cubic feet is known. The problem is to find which items, and in what uantities, to include in the cargo in order to maximize total value. This rototype problem is an extension of the well-known cargo-loading problem ubject to weight constraint. The solution of this problem is left as homework see Problem 8.21).

c. Allocation of Two Resources Subject to Two Capacity Constraints

\ straightforward extension of the allocation of one resource to n activities s the allocation of two resources to n activities. Let (1) x and y be the available quantities of the two resources, (2) x_i and y_i be the quantities of these esources allocated to activity i, and (3) $g_i(x_i, y_i)$ be the return from the ith ctivity resulting from the allocation of x_i and y_i to that activity. The problem n this case is to maximize total returns subject to the availability (capacity) of he resources. Formally,

$$\max_{x_i, y_i} \sum_{i=1}^{n} g_i(x_i, y_i) \tag{8.19}$$

t

$$\sum_{i=1}^{n} x_i \le x; \text{ and } \quad \sum_{i=1}^{n} y_i \le y$$

d

$$x_i, y_i \ge 0$$

The dynamic-programming approach to this two-dimensional allocation process is the same as in the one-dimensional allocation process. The recurrence relations are

$$f_n(x, y) = \max_{0 \le x_n \le x} \ \max_{0 \le y_n \le y} \{g_n(x_n, y_n) + f_{n-1}(x - x_n, y - y_n)\} \tag{8.20}$$

and for the case $n = 1$, we have

$$f_1(x, y) = g_1(x, y)$$

An example of such a process is the allocation of limited land and labor among various vegetables. We have introduced and solved a similar example in Chapter 3 by linear programming. However, the reader can note that the assumption of linearity, which is essential in linear programming, is not required in dynamic programming. In other words, the objective function (8.19) can take any form, continuous or discrete. ~~We will not illustrate and solve such a problem here, but rather leave it as homework (see Problem 8.20).~~ Dynamic programming can also treat problems that have stochastic aspects (for example, a problem in which the demand for a product is described by a known Poisson distribution). This ability to deal with stochastic aspects is, in fact, one of the great advantages of dynamic programming.

d. Computation

Multidimensional allocation processes can be solved in various ways. Problems with two variables and/or two constraints with a small number of states can be solved by using recurrence relations in a way similar to that used in solving the one-dimensional problem. For large problems we can use Lagrange multipliers, and for even larger problems we can use an approximation approach (see Bellman and Dreyfus [6]).

Use of Recurrence Relations

As in the case of the one-dimensional problem, we can break the unknowns x and y into intervals (say at integers). For each pair of x_i and y_i we have a reward or cost function, usually given in a matrix form. Using basically the same approach as employed in Section 8.2, we write the general recurrence relation as given in Equations (8.18) and (8.20) and then, by successive allocation, find the optimal value.

The major drawback of this method is that when we have more than two variables and/or when we have many states, we encounter computational difficulties, such as exceeding the memory and storage capacities of today's computers. The reader should remember that we must simultaneously retain the function $f_{n-1}(x, y)$ and compute the return function $f_n(x, y)$ and the policy function. In small problems the method is quite effective. Although not illustrated here, stochastic aspects can be incorporated into this approach.

Lagrange Multipliers

In solving multidimensional allocation processes, Lagrange multipliers λ_i (see Appendix D) can be used as a means of reducing the dimensionality of dynamic-programming problems. For example, examine the case of allocation of two resources in (8.19). Suppose that the second constraint is an equality

$$\sum_{i=1}^{n} y_i = y.$$

Then we can include the objective function and the equality constraint in the Langrangian function, reducing the problem to a one-constraint problem. Formally,

$$\max g_1(x_1, y_1) + g_2(x_2, y_2) + \cdots + g_n(x_n, y_n) - \lambda\left(\sum_{i=1}^{n} y_i - y\right) \tag{8.21}$$

s/t

$$x_1 + x_2 + \cdots + x_n \leq x \qquad x_i \geq 0 \quad \text{and} \quad y_i \geq 0$$

We then maximize over y_i independently of the maximization over x_i; that is,

$$h_i(x_i, \lambda) = h_i(x_i) = \max_{y_i \geq 0} \{g_i(x_i, y_i) - \lambda y_i)\} \tag{8.22}$$

Thus we reduce the problem to

$$\max_{x_i} h_1(x_1) + h_2(x_2) + \cdots + h_n(x_n) \tag{8.23}$$

s/t

$$x_1 + x_2 + \cdots + x_n \leq x$$

Now (8.23) is equivalent to the one-dimensional problem presented previously (see Section 8.2.2.). The solution to (8.23) will be of the form $x_i(\lambda, x)$, which is a function of λ. Similarly, the values of $y_i = y_i(\lambda)$ resulting from $h_i(x_i)$ as given in Equation (8.22), are a function of λ. We thus vary λ in such a way that the following restriction is met:

$$\sum_{i=1}^{n} y_i = y \tag{8.24}$$

We can treat problem (8.17) in a similar manner; that is, assuming an equality for the second constraint

$$\sum_{i=1}^{n} b_i(x_i) = y$$

we form a Lagrangian function:

$$g_1(x_1) + g_2(x_2) + \cdots + g_n(x_n) - \lambda[b_1(x_1) + b_2(x_2) + \cdots + b_n(x_n) - y] \tag{8.25}$$

to be maximized subject to

$$a_1(x_1) + a_2(x_2) + \cdots + a_n(x_n) \leq x$$

Here we have the recurrence equations:

$$f_n(x) = \max_{\substack{x_n,\ s/t \\ a_n(x_n) \leq n}} \{g_n(x_n) - \lambda b_n(x_n) + f_{n-1}(x - a_n(x_n))\}$$

Again, the results depend on λ, which should be varied until the following constraint is met.

$$\sum_{i=1}^{n} b_i(x_i) = y$$

Approximation

In several cases, approximation can be used as a device to save computational time. The major problem with approximation is that it does not guarantee an optimal solution. In some cases it can guarantee the local maximum but not the global one. In our discussion we shall use the following notation:

$\mathring{x} = (\mathring{x}_i) =$ a set of allocations of resource x to activities i at the starting stage
$\mathring{y} = (\mathring{y}_i) =$ a set of allocations of resource y to activities i at the starting stage
$\mathring{x}_1 = (\mathring{x}_{1i}) =$ a set of allocations of resource x to activities i at the second search cycle
$\mathring{y}_1 = (\mathring{y}_{1i}) =$ a set of allocations of resource y to activities i at the second search cycle
and so on.

Let us examine the allocation of a two-resource example. In such a case we can employ the following *successive approximation:* We start with guessing initial values for x_i. Let these values be such that $\mathring{x} = (\mathring{x}_i)$. For this set we then determine:

$$R_n(x, y) = \max_{y_i} \sum_{i=1}^{n} g_i(\mathring{x}_i, y_i) \tag{8.26}$$

s/t

$$\sum_{i=1}^{n} y_i \leq y$$

This is done by the following one-dimensional recurrence relation:

$$f_n(y) = \max_{0 \leq y_n \leq y} \{g_n(\mathring{x}_n, y_n) + f_{n-1}(y - y_n)\} \tag{8.27}$$

where $n = 2, 3, \ldots$ and $f_1(y) = g_1(\mathring{x}_1, y)$. This approach yields $\mathring{y} = (\mathring{y}_i)$.

Next we take $\mathring{y}$ and introduce it into the objective function. Then our next step is to

$$\max_{x_i} \sum_{i=1}^{n} g_i(x_i, \mathring{}_i) \tag{8.28}$$

s/t

$$\sum_{i=1}^{n} x_i \leq x$$

This problem again is solved by a one-dimensional recurrence relation. Now we get a solution $\mathring{x}_i = (\mathring{x}_{1i})$. This solution is plugged as a constraint into a new problem similar to (8.26), and a new solution $\mathring{y}_1 = (\mathring{y}_{1i})$ is found.

The process is repeated and the value of the objective function is monotonically increasing. We continue the process until we can achieve no further improvement in the objective function. Schematically, the successive approximation method can be represented as shown below:

$$\begin{tikzcd}
\text{Starting stage} & \mathring{x} \arrow[r] & \circ \arrow[dl] \\
\text{Second cycle} & \mathring{x}_1 \arrow[r] & \mathring{y}_1 \arrow[dl] \\
\text{Third cycle} & \mathring{x}_2 \arrow[r] & \mathring{y}_2 \arrow[dl, dashed] \\
\text{and so on} & &
\end{tikzcd}$$

This method can be used to find a *local maximum*. There is no guarantee for a global maximum. The method can also be used to test the optimality of a proposed solution. As in the case of the Lagrange-multiplier approach, we can always identify a nonoptimal solution, but a solution that will pass our test may be a local maximum and not necessarily a global maximum.

8.3 NETWORKS AND DECISION TREES

8.3.1 INTRODUCTION

One of the newest and most promising prototype dynamic-programming problems is the one involving trajectories. The major use of models involving trajectories is in the areas of space research and commercial and military aircraft. One important segment of trajectories, namely networks and decision trees, is receiving increasing attention from management. In this section we shall introduce the major concepts of networks, and then show their use in managerial decision making. Next, we shall show the use of dynamic programming to solve both deterministic and stochastic decision trees and, finally to solve the well-known management control problems of PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method).

Before introducing network problems and their solution, let us define certain basic terms.

A *network* is a model of a system consisting of interrelated activities, such as construction projects, research and development programs, and maintenance programs. Networks are usually represented graphically as in Figure 8.2, which shows a simple network consisting of *events* (or nodes) and *activities* (or arcs or branches).

An *event* is an identifiable point of progress during the completion of the project. The circled numbers 1 through 7 in Figure 8.2 are events or nodes. The beginning node, 1, is called the *source*, or start, and the last node, 7, is called the *sink*, or destination.

An *activity* represents a task requiring a certain period of time for its completion. In Figure 8.2, 1-2 and 4-6 are examples of two activities. Activity 1-2 connects "nodes" 1 and 2 and it takes two weeks for completion; that is, its duration is two weeks.

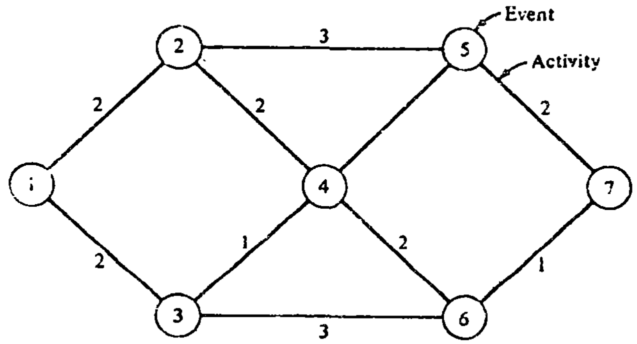


FIGURE 8.2

We note that a "node" occurs at the junction of certain activities. Depending upon the type of work, the numbers along the arcs (activities) can represent units of time (duration) or units of money or some other measure of effectiveness. Networks sometimes employ arrows to indicate the direction of progress between nodes. When no arrows are used, we assume that the network progresses from left to right and that no loops are permitted. The objective in most network problems is to find the shortest or longest path through the network.

When networks are employed to depict sequential decision processes, they are usually called *decision trees*. Similar to the graphical representation of networks, decision trees consist of nodes and branches.[9] Any time a node is connected to more than one other node, the decision maker must choose for progressing along a specific arc to reach the next stage. Two types of nodes can be identified in decision trees: decision nodes (usually designated by a square □), where the choice for direction exists, and chance nodes (usually designated by a circle ○), where the progression is by chance rather

[9] One distinguishing characteristic between networks and decision trees is that although different time sequences for network nodes exist, all activities are performed and we pass through all nodes as the work progresses. In decision trees, on the other hand, action choices result in skipping several branches and nodes.

than by choice. A decision tree in which no chance events are included is called a deterministic decision tree.

A decision tree portrays various possible courses of action, and chance determined outcomes, along with their respective *payoffs*. The payoffs or rewards are either constants or are determined by chance or other uncontrollable factors, in which case they are represented by probability distributions.

8.3.2 DETERMINISTIC DECISION TREE, (NONDYNAMIC-PROGRAMMING SOLUTION)

To illustrate the use of a decision tree let us assume that the management of a firm is facing a machine replacement problem, with different paths and their associated rewards. Figure 8.3 indicates that if the machine is replaced at this time ($T=0$), we will gain a net profit of \$50,000 during the first year, and \$70,000 during the second year. On the other hand, if we do not replace the machine at this time, we will enjoy a net profit of \$70,000 during the first year and, after one year at $T=1$, we will again face a replacement decision with the rewards during the second year shown in Figure 8.3 (55,000 if we replace and 40,000 if we do not replace). We have, in effect, three different alternatives. Replace now, replace after one year, or do not replace at all.

The solution to the problem is achieved through simple enumeration of the three possible alternatives, and by comparing their associated rewards. The results are summarized in Table 8.8, from which it is obvious that the optimal decision is to replace the machine after one year (assuming that all data are already discounted to time zero).

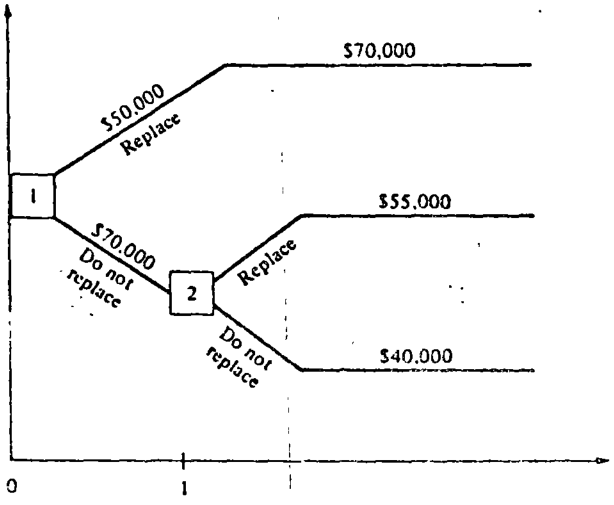


FIGURE 8.3

Table 8.8

ALTERNATIVE	REWARD, DOLLARS
1. Replace now	50,000 + 70,000 = 120,000
2. Replace after one year	70,000 + 55,000 = 125,000
3. Do not replace	70,000 + 40,000 = 110,000

We solved the replacement problem by actually identifying all possible paths through the network, calculating the projected profit for each path, and then selecting the path with the highest profit. This type of approach is all right for a small problem, but a complete manual enumeration of all the possible paths of a large network would be extremely time-consuming and costly. Dynamic programming provides an elegant and efficient way to solve large network problems.

8.3.3 DYNAMIC-PROGRAMMING APPROACH TO DETERMINISTIC NETWORKS

A cost-minimization problem in the form of a network is depicted in Figure 8.4. Our objective is to find the shortest path[10] (equivalent to cost minimiza-

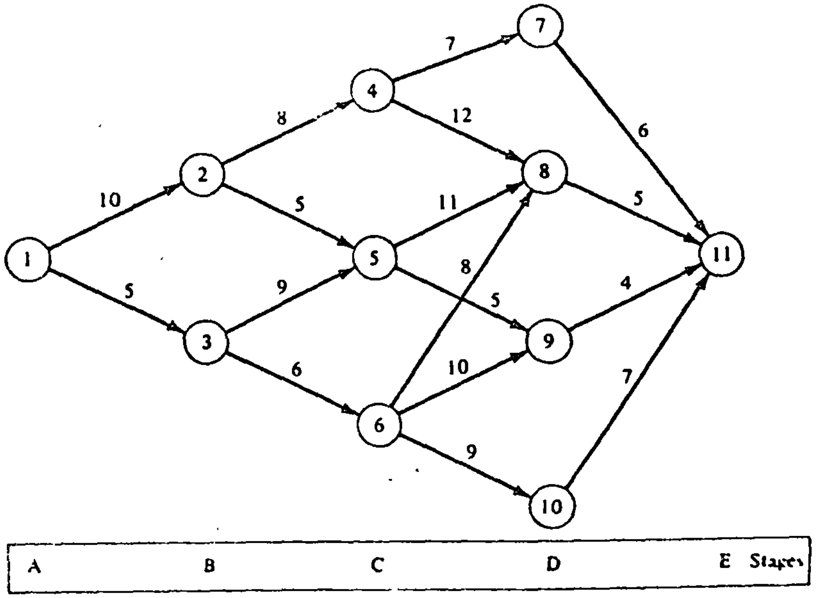


FIGURE 8.4

[10] The reader should note that in any network we can look at a " problem—that is, finding the longest path (equivalent to a maximization problem).

tion) from node 1 to node 11 by the application of dynamic programming.

We solve the problem backwards. The first time a decision problem exists is at stage D.[11]

Solution, Step I: Check Stage D

Nodes 7, 8, 9, and 10 are the four possible states in stage D. There is only one branch linking node 11 to each of these states.

The cost involved in going from each of the nodes 7, 8, 9, and 10 to node 11 is computed below.

Let us adopt the following notations:

$f_n(d)$ = *minimum* cost involved in proceeding from the nth node to the last node (along the shortest path)

d_{ij} = actual cost involved in moving from the ith node in one stage to jth node in the next stage

Then for our example, we have

$$\begin{aligned} f_7(d) &= d_{7\text{-}11}=6\\ f_8(d) &= d_{8\text{-}11}=5\\ f_9(d) &= d_{9\text{-}11}=4\\ f_{10}(d)&=d_{10\text{-}11}=7 \end{aligned}$$

Solution, Step II: Check Stage C

Nodes 4, 5, and 6 represent the three states of stage C. Our problem at this stage is to find the minimum cost (shortest path) between stage C and stage E. We could check all possible paths of progression between stage C and stage E, compare the associated costs, and then choose the least-cost path. Starting from node 6 in stage C, for example, we can reach stage D via nodes 8, 9, or 10, with costs of 8, 10, or 9 respectively. To these costs must be added the optimal costs of proceeding from nodes 8, 9, 10 to node 11.

This can be accomplished by utilizing the principle of optimality. The total cost of proceeding from each node in stage C to stage E, (TC_{CE}), is made up of two components:

$$TC_{CE}=d_{ij}+f_j(d)$$

where

d_{ij} = actual cost of proceeding from the ith node in stage C to the jth node in stage D

$f_j(d)$ = the minimum cost of proceeding from the jth node in stage D to the last (E) stage

Now we would like to find the lowest possible value of TC_{CE}. This is done by simple enumeration. The lowest possible cost of progressing from node 6 in stage C to stage E is designated by $f_6(d)$:

$$f_6(d)=\min\begin{Bmatrix} d_{6\text{-}8} +f_8(d) = 8+5=13\\ d_{6\text{-}9} +f_9(d) = 10+4=14\\ d_{6\text{-}10}+f_{10}(d)= 9+7=16\end{Bmatrix}=13$$

Note that there are three alternative ways to proceed from node 6 in stage C to stage E. The least cost $f_6(d)$, however, involves proceeding from 6 to 8, and then from 8 to 11, with a cost of 13.

Calculations for proceeding from nodes 4 and 5 in stage C to stage E are as follows:

$$f_4(d)=\min\begin{Bmatrix} d_{4\text{-}7}+f_7(d)= 7+6=13\\ d_{4\text{-}8}+f_8(d)=12+5=17\end{Bmatrix}=13$$

$$f_5(d)=\min\begin{Bmatrix} d_{5\text{-}8}+f_8(d)=11+5=16\\ d_{5\text{-}9}+f_9(d)= 5+4= 9\end{Bmatrix}=9$$

Now we can find, by enumeration, the lowest value among $f_4(d)$, f_5(d), and $f_6(d)$. This value represents the lowest cost of moving from stage C to stage E.

It is evident the shortest path from stage C to E is 5-9-11, at a cost of 9.

Solution, Step III: Check Stage B

Our next task is to find the least-cost path from stage B to stage E. The procedure is similar to the one in step II. The required calculations are as follows:

$$f_2(d)=\min\begin{Bmatrix} d_{2\text{-}4}+f_4(d)=8+13=21\\ d_{2\text{-}5}+f_5(d)=5+ 9=14\end{Bmatrix}=14$$

$$f_3(d)=\min\begin{Bmatrix} d_{3\text{-}5}+f_5(d)=9+ 9=18\\ d_{3\text{-}6}+f_6(d)=6+13=19\end{Bmatrix}=18$$

This step illustrates the economy of effort made possible by using dynamic programming. Instead of calculating the costs of seven possible paths from stage C to stage E, we make only four sets of calculations.

Note that the best path from stage B to stage E is 2-5-9-11, with a cost of 14.

Solution, Step IV: Check the Final Stage (A)

The rationale in this step is the same as explained in the earlier steps. The actual calculations of this step are as follows:

$$f_1(d)=\min\begin{Bmatrix} d_{1\text{-}2}+f_2(d)=10+14=24\\ d_{1\text{-}3}+f_3(d)= 5+18=23\end{Bmatrix}=23$$

Thus the optimal solution is 1-3-5-9-11, with a cost of 23.

[11] The breakdown into stages here is arbitrary. The analysis can run with each node being a stage.

Evaluation

We can see that in addition to solving the original minimization problem we have information now as to which is the shortest path *from any given state* i to the final stage.

The recurrence relations for the problem are given by

$$f_i(d) = \min_j \{d_{ij} + f_j(d)\} \tag{8.29}$$

The application of dynamic programming to networks can be extended to solving stochastic networks.

8.3.4 A STOCHASTIC DECISION TREE

The decision tree in Figure 8.5 portrays the decision problem of the ABC Corporation, facing a machine replacement problem.

The management has two alternatives: repair the old machine at a cost of \$1000 or purchase a new machine at a net cost of \$10,000. Each of these alternatives takes us, along different branches, to chance nodes 2 and 3. Each chance node may result in one of two different payoffs with given probabilities. All the relevant data are given in the decision tree of Figure 8.5. We solve this replacement problem by calculating and comparing the "expected value" of each alternative.

In any discrete probability distribution, the expected value is calculated as:

$$\text{Expected value} = p_1K_1 + p_2K_2 + \cdots + p_nK_n \tag{8.30}$$

where p_i = probability of ith outcome and K_i = numerical value of ith outcome.

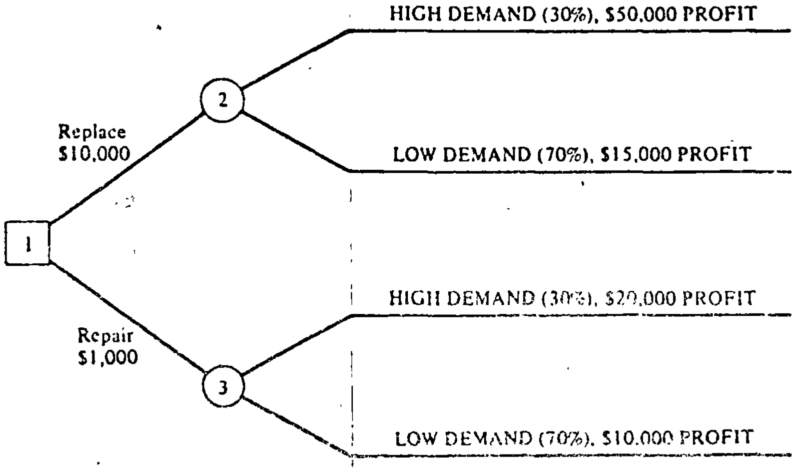

HIGH DEMAND (30%), $50,000 PROFIT
LOW DEMAND (70%), $15,000 PROFIT
HIGH DEMAND (30%), $20,000 PROFIT
LOW DEMAND (70%), $10,000 PROFIT
Replace
$10,000
Repair
$1,000
1
2
3

FIGURE 8.5

In our example, the expected value of the "replace" alternative is given as follows:[12]

$$(-10{,}000) + \{0.30(50{,}000) + 0.70(15{,}000)\} = \$15{,}500$$

The expected value of the "repair" alterrfative is

$$-1000 + \{0.30(20{,}000) + 0.70(10{,}000)\} = \$12{,}000$$

Our optimal decision in this case is to replace the machine.

Conceptually, once the expected-value calculations have been made, our probabilistic decision tree of Figure 8.5 can be represented as the equivalent deterministic tree of Figure 8.6, in which it can be seen that the decision choice is simple and straightforward (replace). Similarly, large probabilistic decision trees can be changed into equivalent deterministic models that can then be solved by dynamic programming.

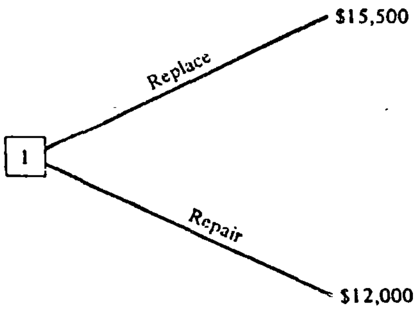


FIGURE 8.6

8.3.5 APPLICATION OF DYNAMIC PROGRAMMING TO PERT AND CPM

Dynamic programming can solve PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method) problems. The objective in PERT and CPM is to determine the longest path in the network. Each node in a PERT or CPM network is a stage in itself.

PERT and CPM are planning and control techniques based on network theory (see Moder and Phillips [22]). Both techniques are used in large projects (such as construction, research and development, and equipment overhaul) involving many interrelated activities. In both techniques, the major objective is to identify the *critical path*—that is, to identify the bottleneck activities.

The major idea of both CPM and PERT is a graphical presentation of the

[12] In this case there is an outcome of $-10{,}000$ (that is, cost [illegible] certainty; in other words, the probability is equal to 1.

project using a network, where the nodes represent events and the branches represent activities. A usual distinction between CPM and PERT is that CPM deals with deterministic cases whereas PERT handles probabilistic cases. The duration of an activity labeled t_e, in the PERT approach, is computed as an average of the following estimates:

a = optimistic estimate[13]
m = most likely estimate
b = pessimistic estimate

according to the following formula:

$$t_e = \frac{6}{a+4m+b} \tag{8.31}$$

and it is this number that is written along the branches in the PERT network. The CPM, on the other hand, considers *single* estimates for the time required to perform different activities in the network.

An Illustrative Example

Find the longest path of the PERT network of Figure 8.7. The numbers along the branches are the average expected duration of the activities (t_e).

Solution

A. Stage 8

Let node 8 be the first stage to be considered, working backward. The longest path from node 8 to node 9 is 6 days. Formally, we write this information as:[14]

$$f_8 = d_{8\text{-}9} = 6$$

B. Stage 7

We have two alternative ways of proceeding from node 7 to node 9. The direct path 7-9 takes 9 days; the other path, 7-8-9, takes 8 days (2+6). Formally,

$$f_7 = \max \begin{Bmatrix} d_{7\text{-}9} + 0 = 9 + 0 = 9 \\ d_{7\text{-}8} + f_8 = 2 + 6 = 8 \end{Bmatrix} = 9$$

Proceeding backward to stage 1 and analyzing the intermediate stages, we obtain the following results.

[13] The three different time estimates for completing the activity are based on the assumption that the beta distribution is the probability distribution representing the various possible completion times for the activity. Thus, a represents the optimistic time estimate (with a probability of 1 in 100), b represents the pessimistic time estimate (with a probability of 1 in 100), and m is the mode of the distribution as *estimated* by the project analyst.

[14] In this problem we shall write f_i in place of $f_i(d)$. $d_{i\text{-}j}$ denotes t_e of activity ij.

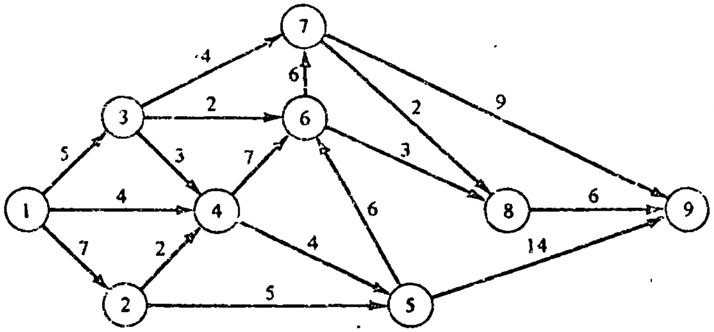


FIGURE 8.7

C. Stage 6

$$f_5 = \max \begin{Bmatrix} d_{6\text{-}8} + f_8 = 3 + 6 = 9 \\ d_{6\text{-}7} + f_7 = 6 + 9 = 15 \end{Bmatrix} = 15$$

Thus the longest path between 6 and 9 is 6-7-9 with 15 days.

D. Stage 5

$$f_5 = \max \begin{Bmatrix} d_{5\text{-}9} + 0 = 14 + 0 = 14 \\ d_{5\text{-}6} + f_6 = 6 + 15 = 21 \end{Bmatrix} = 21$$

E. Stage 4

$$f_4 = \max \begin{Bmatrix} d_{4\text{-}6} + f_6 = 7 + 15 = 22 \\ d_{4\text{-}5} + f_5 = 4 + 21 = 25 \end{Bmatrix} = 25$$

F. Stage 3

$$f_3 = \max \begin{Bmatrix} d_{3\text{-}7} + f_7 = 4 + 9 = 13 \\ d_{3\text{-}4} + f_4 = 3 + 25 = 28 \\ d_{3\text{-}6} + f_6 = 2 + 15 = 17 \end{Bmatrix} = 28$$

G. Stage 2

$$f_2 = \max \begin{Bmatrix} d_{2\text{-}4} + f_4 = 2 + 25 = 27 \\ d_{2\text{-}5} + f_5 = 5 + 21 = 26 \end{Bmatrix} = 27$$

H. Stage 1

$$f_1 = \max \begin{Bmatrix} d_{1\text{-}3} + f_3 = 5 + 28 = 33 \\ d_{1\text{-}4} + f_4 = 9 + 25 = 34 \\ d_{1\text{-}2} + f_2 = 7 + 27 = 34 \end{Bmatrix} = 34$$

As the last set of calculations shows, we have two equal longest paths; that is, we have two optimal solutions.

The first is

1-4-5-6-7-9 at 34 days

and the second is

1-2-4-5-6-7-9 at 34 days

The recurrence relation for this problem is given by

$$f_n = \max_j \{d_{n \to j} + f_j\} \tag{8.32}$$

where $n \to j$ represents all possible direct paths from node n to connecting nodes j.

8.4 ONE-DIMENSIONAL SMOOTHING AND SCHEDULING PROCESS

8.4.1 INTRODUCTION

In Section 8.2 a static (occurring in a single time period) allocation process was portrayed as a dynamic process and then solved by using dynamic programming. In Section 8.3 we applied dynamic programming in solving problems that dealt with either single-decision situations (such as finding the shortest path in a network) or situations involving multiple decision points (such as decision-tree types of problems). The problems of Section 8.3, though of a multistage nature, did not necessarily represent multitime periods. In this section we turn our attention to processes involving more than a single time period. These dynamic, rather than static, processes take place in such business problems as inventory control, replacement, and production smoothing and scheduling.

8.4.2 SMOOTHING PROCESSES

A smoothing process is one in which two opposing costs are balanced in order to achieve an optimal least-cost solution. To illustrate: Assume that we are dealing with a system that should operate in a certain specified state. If the system is *not* operating according to the specified state, a known cost c_1 is incurred. This cost is a function of the magnitude of the deviation from the specified state. A second cost c_2 is incurred when we attempt to transform the system into the desired state. *A smoothing process* balances c_1 against c_2 in such a way that the overall objective of operating the system is optimized. Examples of such processes are employment-level determination in view of a fluctuating demand for manpower (where the cost of idle employees is balanced against costs of hiring and firing) and economic order quantity in inventory problems, where the set-up cost is balanced against holding cost. Many inventory, replacement, and production scheduling problems also fall into this category. Several complicated engineering problems, such as feedback control (see Bellman and Dreyfus [6]), can also be considered as smoothing processes.

In the remainder of this section we will illustrate several typical smoothing processes in business and economics and solve them by dynamic programming.

8.4.3 OPTIMIZING EMPLOYMENT LEVEL

Let us consider a manpower scheduling situation in which

1. Fluctuating manpower requirements per time period are known with certainty.
2. The penalty for being "out of stock" is prohibitive (that is, all demands must be met).
3. No overtime work is permitted (because of a three-shift schedule and limited facilities).

Since we are dealing with human resources, manpower cannot be stored in the sense that physical goods are stored.

An Illustrative Example

The ABC chemical plant is being operated around the clock. Manpower requirements for plant maintenance are assumed to be known with certainty. Because of minor and major overhauls in different quarters, yearly manpower requirement varies as shown in Table 8.9. The problem is to find the optimal level for the working force during the year.

Table 8.9

QUARTER	1	2	3	4
MEN REQUIRED (r_i)	54	60	120	80

At the outset, we can suggest these alternative solutions to this problem:

Alternative 1. Keep the employment level *exactly* equal to the demand level, for each quarter. This can be accomplished by hiring (or laying off) as the need occurs. This approach will probably be quite costly, due to excessive recruitment, training, and layoff costs.

Alternative 2. For the entire planning period, keep the employment level equal to the highest demand level. This means that our crew size will remain constant and although we avoid high costs associated with layoffs, we incur the costs of idle crew.

Alternative 3. Vary the crew size, but not necessarily in each quarter. The objective of this policy is to find the optimum employment level to balance the opposing costs of idle crew on the one hand and costs of hiring and layoff on the other.

Dynamic programming is used to determine such an optimum employment policy.

Let us adopt the following notations and/or assumptions:

1. Cost per idle employee per quarter = $2500.
2. x_i = level of employees in the ith stage.
3. c_i = total costs associated with changeover from the ith to the next stage [illegible]

4. d_i = number of employees hired or laid off in the ith stage. This number is given by the relation

$$d_i = |x_i - x_{i+1}| \tag{8.33}$$

5. Part-time employees are available. This means that crew sizes involving fractional answers are permissible.
6. s = policy employment level in a given quarter.
7. r_i = manpower required for a given quarter i.

Our objective is to find the maintenance crew size for each period that will minimize total costs for the planning period.

Solution

At any given stage (quarter), the decision about the optimal employment level will be based on the manpower requirement in that quarter and on the level of employment in the previous quarter. In each quarter we have an upper limit to employment, which is given by the highest demand level during the entire planning horizon (120 in our case). The state variable of the system in our case is the policy employment level s (the only unknown variable).

In order to solve this problem by dynamic programming we shall assume that our process continues for several years with a constant yearly demand, and with the same quarterly fluctuations as shown in Table 8.9. For computational purposes, we shall use a planning horizon of seven quarters[15] (see Table 8.10).

Table 8.10

STAGE	3	2	1	4	3	2	1
QUARTER	1	2	3	4	5	6	7
MEN REQUIRED (r_i)	54	60	120	80	54	60	120

We start our analysis with quarter 7 and proceed *backward*. Since the data from quarter 7 to quarter 4 forms a complete cycle (one year), we can conclude our analysis when we have analyzed the fourth quarter. The problem will then have been solved and the answer obtained will be valid for any number of years, so long as none of the conditions are changed. Let

x_1 = the employment level at quarter 7, (as well as at quarter 3)
x_2 = the employment level at quarters 6 and 2
x_3 = the employment level at quarters 5 and 1
x_4 = the employment level at quarter 4
$\mathring{x}_i$ = the optimal employment level at stage i

[15] The last quarter should be the one with the *highest* demand. This simplifies computations considerably. Also note that we have used a slightly different notation in this illustration. In the earlier examples, the last stage was given the highest numerical value. Here, the latest stage is being denoted as stage 1. Of course, the method and the sequence of analysis does not change. Here, as in previous examples, we start with the "last" stage and work backward.

The total cost at each stage is composed of two parts:

1. Idle crew, whose size is given by the difference of employment level and the manpower requirement—that is, $(x_i - r_i)$—and whose cost is given by

$$2500(x_i - r_i) \tag{8.34}$$

2. Changeover cost, which is given by

$$c_i = 250(x_i - x_{i+1})^2 \tag{8.35}$$

Once the employment level for a given quarter is decided, this level, s, is the *state* entering the next quarter which is the previous stage.

Our functional equation in this case is, as usual, based on the principle of optimality and is given by

$$f_n(s) = \min_{x_n \geq r_n} \{250(x_n - s)^2 + 2500(x_n - r_n) + f_{n-1}(x_n)\} \tag{8.36}$$

Stage 1

For quarters 7 and 3, obviously, $\mathring{x}_1 = r_7 = 120$. Thus

$$f_1(s) = 250(120 - s)^2 + 2500(120 - 120) = 250(120 - s)^2 \tag{8.37}$$

In other words, the only cost here is the changeover cost, which depends on s, our policy decision on employment level in quarter 6 (the next stage).

Stage 2

Similarly, for quarter 6 we have:

$$\begin{aligned} f_2(s) = \min g_2(s, x_2) &= \min_{x_2 \geq 60} \{250(x_2 - s)^2 + 2500(x_2 - r_6) + f_1(x_2)\} \\ &= \min_{x_2 \geq 60} \{250(x_2 - s)^2 + 2500(x_2 - 60) + 250(120 - x_2)^2\} \end{aligned} \tag{8.38}$$

We shall attempt to find the minimum of this function, for any fixed value of s, by the classical calculus method.

1. Take the first partial derivative of $g_2(s, x_2)$ and equate it to zero:

$$\frac{\partial g_2(s, x_2)}{\partial x_2} = 500(x_2 - s) + 2500 - 500(120 - x_2) = 0 \tag{8.39}$$

Solving (8.39) for x_2, we obtain the optimal value:

$$\mathring{x}_2 = \frac{57,500 + 500s}{1000} = 57.5 + 0.5s \tag{8.40}$$

2. Check the identity of point $\mathring{x}_2$. The second partial derivative of $g_2(s, x_2)$ yields

$$\frac{\partial^2 g_2(s, x_2)}{\partial x_2^2} = 1000 \tag{8.41}$$

Since the second derivative is positive, we have a global minimum at $\mathring{x}_2$.

3. We now check values of s in order to satisfy the constraint $x_2 \geq 60$. We know from Equation (8.40) that $\hat{x}_2 = 57.5 + 0.5s$ is a global minimum. Even if $s \leq 5$, x_2 must be 60 because of the stated constraint. For $\hat{x}_2$ to be greater than 60, s must be greater than 5.

It is not necessary to consider $s \leq 5$ because it is constrained from below by 54. Thus we wish to examine only $s > 5$, for which $\hat{x}_2 = 57.5 + 0.5s$.

Please note that s is limited from above by 120. We now introduce all this information into $f_2(s)$ and obtain the following:

$$f_2(s) = 250(57.5 + 0.5s - s)^2 + 2500(57.5 + 0.5s - 60) + 250(120 - 57.5 - 0.5s)^2$$

Again, in this last expression, the optimal policy is a function of s; therefore the value for s must come from the next stage.

Stages 3 and 4

We continue in a similar way and express the optimal policy at stage 3 as a function of s. The procedure is repeated in stage 4, where the optimal policy is achieved when $s = 120$. When this value of $s = 120$ is introduced in the earlier stages, we can calculate $f_3(s)$, $f_2(s)$, and $f_1(s)$. The reader can verify that this will yield the following optimal values:

$$x_1 = 120 \qquad x_3 = 115$$
$$x_2 = 117.5 \quad x_4 = 112.5$$

Note that this solution indicates excessive idle time. This is because of the large changeover cost.

8.4.4 EQUIPMENT REPLACEMENT POLICY

Of the smoothing-process types of problems, the replacement and maintenance problems are of special interest because they are almost unsolvable by other analytical techniques. Here, we shall illustrate the dynamic-programming approach to a simple replacement problem.

There are several circumstances in which individuals as well as firms must make periodic replacement decisions. The replacement of the family automobile is perhaps the best illustration of this type of an individual or family decision process in our society. Many replacement and maintenance problems are multistage problems involving periodic preventive maintenance and/or replacement decisions. In addition, maintenance and/or repair scheduling can involve many variables in complex functional relations.

An Illustrative Example (Single Machine Replacement Problem)

Let us assume that each year a new model of a certain machine is available for use on the first day of January. The manager of a manufacturing department using this type of machine faces the problem of replacing the old machine by [illegible] new model in order to [illegible] Let us assume that sufficient data on costs and revenues are available to enable our manager to set up a planning horizon that covers four years.

The replacement cost is a function of the age of the machine to be replaced and the year in which the "new" machine is produced. As shown in Table 8.11, if a 1969 machine is replaced in 1971, the replacement cost is \$10,000. The replacement of the 1969 machine by a 1972 model will cost \$15,000. A complete schedule of replacement costs, covering the four years, 1969 through 1972, is given in Table 8.11.

Table 8.11 Replacement Cost, R_{ij}

		YEAR OF OLD MACHINE			
		1969	1970	1971	1972
	1969	1973	[illegible]	1977	1972
YEAR OF DECISION (NEW MACHINE)	1970	7			
	1971	10	9		
	1972	15	11	10	
		18	13	12	14

Let us adopt the following notation:

R_{ij} = cost of replacing old machine of year j by new machine in year i, in thousands of dollars
i = year of decision (new machine)
j = year of old machine
I_{ij} = revenues generated in the ith year when the machine model is of the jth year; $i \geq j$
M_{ij} = machine operations and maintenance costs for the ith year when the machine model is of the jth year; $i \geq j$

Assuming that both I_{ij} and M_{ij} for the planning period are known, our manager can calculate the expected net returns $(I_{ij} - M_{ij})$. Table 8.12 contains the net returns data, $P_{ij} = (I_{ij} - M_{ij})$.

$= I_{ij} - M_{ij}$

Table 8.12 Net Return, P_{ij}

5
115
115
3
5
65

			1 YEAR OF OLD MACHINE 2 3 4		
		1969	1970	1971	1972
1	1969	19			
YEAR OF DECISION (NEW MACHINE) 2	1970	15	22		
3	1971	12	18	23	
4	1972	10	12	15	26

Solution

The problem can be solved either by an enumeration approach or by the application of dynamic programming. If the problem size is small, the enumeration approach is quite practical. For large-size problems, however, the dynamic-programming technique is preferable.

Enumeration of All Possible Alternatives

We start with the knowledge that the old machine had been replaced on January 1, 1969, with the 1969 model. Then, our manager has only eight available alternatives[16] for the planning horizon. Let R = replace the machine, K = keep the machine.

The eight alternatives are listed in Table 8.13.

It is easy to enumerate and arrive at the payoffs given in Table 8.13. Obviously, alternative 2, with the highest payoffs, is the best strategy for our manager; that is, replace in 1970 and in 1972, and keep the 1970 model during 1971.

Table 8.13

ALTERNATIVE NUMBER	REPLACEMENT POLICY			PAYOFF, DOLLARS
	1970	1971	1972	
1	R	R	R	64,000 ✓
2	R	K	R	67,000
3	R	K	K	64,000
4	R	R	K	63,000 ✓
5	K	K	K	56,000
6	K	K	R	57,000
7	K	R	K	62,000
8	K	R	R	63,000

The Dynamic-Programming Approach

Let $f_i(j)$ = Maximum total net payoff from beginning of year i to the end of the horizon, when the equipment on hand was purchased during the year j ($j < i$).

Then, at each decision year i, with equipment purchased during j, the two choices are:

Payoff

Keep: $P_{ij} \quad + f_{i+1}(j)$ (8.42)

Replace: $P_{ii} - R_{ij} + f_{i+1}(i)$ (8.43)

[16] Two alternatives each at the beginning of 1970, 1971, and 1972. Hence the total available alternatives are $2 \times 2 \times 2 = 8$.

Therefore, the recurrence relation becomes:

$$f_i(j) = \max\begin{Bmatrix} P_{ij} \quad + f_{i+1}(j) \\ P_{ii} - R_{ij} + f_{i+1}(i) \end{Bmatrix}, \qquad i = 1,2,3,4 \tag{8.44}$$

Since the fourth year is assumed to be the end of the planning horizon, we set

$$f_5(j) = 0 \tag{8.45}$$

This initial condition allows a backward induction using the recurrence relation.

Stage 4 (1972)

The recurrence relation at this stage is

$$f_4(j) = \max\begin{Bmatrix} P_{4j} \\ P_{44} - R_{4j} \end{Bmatrix}, \qquad j = 1,2,3 \tag{8.46}$$

From Tables 8.11 and 8.12 we obtain:

$$f_4(1) = \max\begin{Bmatrix} 10 \\ 26 - 15 \end{Bmatrix} = 11$$

$$f_4(2) = \max\begin{Bmatrix} 12 \\ 26 - 11 \end{Bmatrix} = 15$$

$$f_4(3) = \max\begin{Bmatrix} 15 \\ 26 - 10 \end{Bmatrix} = 16$$

Stage 3 (1971)

The recurrence relation now becomes

$$f_3(j) = \max\begin{Bmatrix} P_{3j} \quad + f_4(j) \\ P_{33} - R_{3j} + f_4(3) \end{Bmatrix}, \qquad j = 1,2 \tag{8.47}$$

Substituting again from Tables 8.11 and 8.12,

$$f_3(1) = \max\begin{Bmatrix} 12 + 11 \\ 23 - 10 + 16 \end{Bmatrix} = 29$$

$$f_3(2) = \max\begin{Bmatrix} 18 + 15 \\ 23 - 9 + 15 \end{Bmatrix} = 33$$

Stage 2 (1970)

$$f_2(j) = \max\begin{Bmatrix} P_{2j} \quad + f_3(j) \\ P_{22} - R_{2j} + f_3(2) \end{Bmatrix}, \qquad j = 1 \tag{8.48}$$

Hence

$$f_2(1) = \max\begin{Bmatrix} 15 + 29 \\ 22 - 7 + 33 \end{Bmatrix} = 48$$

Stage 1 (1969)

Here there is no decision since a new equipment is purchased. Hence

$$\begin{aligned} f_1(j) &= P_{11} + f_2(1) \\ &= 19 + 48 = 67. \end{aligned}$$

From stage 2 we see that the equipment should be replaced in 1970. Stage 3 is thus entered with equipment purchased at $j=2$. From $f_3(2)$ we observe that during 1971 it should be kept. Stage 4 is entered with $j=2$ also, and from $f_4(2)$ it can be seen that it should be replaced during 1972. The total payoff is $f_1(j)=67$.

8.4.5 THE WAREHOUSE PROBLEM

a. Introduction

The warehouse problem involves the purchasing of a single commodity at specified periods or stages, storing for some time period, and then selling to the customers. In this sense the warehouse problem can be viewed as an inventory-control problem; it is also an extension of the transportation problem.

The warehouse problem is a classical example of linear dynamic programming. The decision maker must make periodic decisions. Each specific decision depends on the "state of the system" as determined by the preceding decisions. Several complex production scheduling, inventory, and allocation problems can be formulated in terms of the warehouse problem. In its simplest form, the warehouse problem can be solved by linear programming (Dantzig [10], p. 55).

Some characteristics and assumptions of the warehouse model can be noted.

An upper limit exists to the buy, store, and sell transactions involved in a typical warehouse problem. Some of the factors determining the upper limits are available capital, available supply, available storage capacity, and size of demand. We assume that both costs and prices are constant during the planning horizon. We assume, further, that the demand is known with certainty.

Our problem is to maximize profits by determining the optimum level to buy (or produce), store, and sell in each period of the planning horizon.

b. An Illustrative Example

Let us assume that a young man has decided to enter the nonferrous-metal brokerage business. He starts by renting for five months a small warehouse with a storage capacity of 150 tons. Assume, further, that the cost and price schedule is available to our entrepreneur, as given in Table 8.14. Other known facts and assumptions are as follows:

1. Handling, storage, and all other costs are negligible and can be assumed to be included in the cost and price schedule of Table 8.14.
2. A monthly purchase order is placed on the last day of each month (at the current price).
3. The monthly shipment is received on the first day of the following month (that is, a lead time of one day).
4. Sales are made from the second day of each month through the last day of that month.
5. The warehouse contains 50 tons at the beginning of the first month.
6. The warehouse must be empty at the end of the planning period (fifth month).

Our problem is to determine a buying, storage, and selling strategy in order to maximize profits.

Table 8.14 Cost-price schedule for the planning horizon

	MONTH, i	COST PER TON, c_i	PRICE PER TON, p_i
5	1	\$850	\$800
4	2	800	900
3	3	750	750
2	4	650	700
1	5	750	800

Solution

Our small-size problem can be solved by three different approaches: (1) by enumeration, (2) by linear programming, and (3) by dynamic programming. The last two are the only practical approaches for solving large-size problems.

c. The Warehouse Problem—A Linear Programming Formulation

Let

k = warehouse capacity ($k=150$) in tons
x_i = number of tons ordered on the last day of month i
y_i = number of tons sold during month i
s_i = stock, in tons, on the first day of month i, after arrival of shipment
c_i = ordering (or buying) price in month i
p_i = selling price during month i
z_i = unused storage capacity (slack) in month i
a_i = artificial variable in month i
w_i = stock carried over to next month

We can now make the following observations:

1. Our objective is to maximize total profit, where total profit = total revenues − total costs; that is,

$$\text{Total profit} = \sum_{i=1}^{5} p_i y_i - \sum_{i=1}^{5} c_i x_i \tag{8.49}$$

2. We cannot store more goods than permitted by the capacity of our warehouse. This means that

$$s_i \leq k \tag{8.50}$$

or

$$s_i + z_i = k \tag{8.51}$$

As noted previously, the slack variable z_i represents the unused storage capacity in month i. Also, $k = 150$ tons.
3. The stock (in tons) on the first day of each month equals the stock on the first day of the previous month less sales during the previous month, plus stock ordered during the previous month. This means that

$$s_{i+1} = s_i - y_i + x_i \tag{8.52}$$

or

$$s_{i+1} - s_i + y_i - x_i = 0$$

or

$$s_{i+1} - s_i + y_i - x_i + a_i = 0$$

or

$$s_{i+1} + y_i - x_i + a_i = s_i \tag{8.53}$$

Note that $s_1 = 50$.
4. Since we cannot order more than the level required to fill the warehouse completely during month i, we have the requirement that

$$x_i \leq k - (s_i - y_i) \tag{8.54}$$

5. Since we cannot sell more than the stock on hand, we have the requirement that

$$y_i \leq s_i \tag{8.55}$$

or

$$y_i + w_i = s_i$$

From these observations we can state our problem in linear-programming erms as

$$\max \left(\sum_{i=1}^{5} p_i y_i - \sum_{i=1}^{5} c_i x_i \right) \tag{8.56}$$

/t[17]

$$s_i + z_i = k$$
$$s_{i+1} + y_i - x_i + a_i = s_i$$
$$y_i - s_i + w_i = 0$$

nd all variables ≥ 0.

At this stage it appears that we have an objective function and three tructural constraints. However, note that since our planning horizon covers ive periods ($i = 5$), there are actually 15 structural constraints in this probem. The limitation imposed at the end of the horizon (no inventory left) educes the problem somewhat, by reducing the number of variables. That is, $x_5 = 0$, $y_5 = s_5$, $a_5 = 0$, and $w_5 = 0$. We now have sufficient information to lerive a basic feasible solution and solve the problem by the simplex method.

[17] Note that we have only three active (nonredundant) constraints per period.

d. The Warehouse Problem—Solution by Dynamic Programming

Conceptually, we are dealing here with a five-stage problem, as shown in Figure 8.8. We will now proceed, stage by stage, *backward* from stage 5 to stage 1.

Stage $n = 5$ (the First Stage to Be Considered)

On the first day of the last month we receive the quantity x_4 ordered on the last day of the preceding (fourth) month. Our stock on hand at the beginning of stage 5 (first stage to be considered) is s_5. We have to make two decisions during the last month: (1) how much to sell during the month (y_5), and (2) how much to order on the last day (x_5).

Stages	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
Months	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$

FIGURE 8.8

Because of our requirement that the venture end at the conclusion of the fifth period, x_5 must equal zero. While deciding on the level of y_5, we must obey the constraint:

$$y_5 \leq s_5$$

The problem requires, however, that no stock be on hand at the end of the fifth month. Hence y_5 must equal s_5; that is, *sell all stock*.

The maximum profit at this stage is given by

$$f_5(s_5) = \max (p_5 y_5 - c_5 x_5) \tag{8.57}$$

Since $x_5 = 0$,

$$f_5(s_5) = \max p_5 y_5$$

and since $y_5 = s_5$, the maximum profit is given by

$$f_5(s_5) = p_5 s_5 = 800 s_5 \tag{8.58}$$

Stage $n = 4$ (the Second Stage to Be Considered)

On the first day of the fourth month our stock on hand equals s_4. Again, we have to make two decisions: (1) how much to sell during the fourth month (y_4), and (2) how much to buy on the last day of that month (x_4). Using the principle of optimality we get

$$f_4(s_4) = \max \{\underbrace{p_4 y_4 - c_4 x_4}_{\substack{\text{return during} \\ \text{the 4th month}}} + \underbrace{f_5(s_5)}_{\substack{\text{optimal returns} \\ \text{in 5th month}}}\} \tag{8.59}$$

Obviously,

$$s_5 = s_4 + x_4 - y_4 \tag{8.60}$$

Also,[18]

$$f_5(s_5) = p_5 s_5 \tag{8.61}$$

If we substitute (8.60) in (8.61), then

$$f_5(s_4 + x_4 - y_4) = p_5(s_4 + x_4 - y_4) \tag{8.62}$$

If we substitute (8.62) in (8.59), and rearrange terms, we obtain

$$f_4(s_4) = \max \{x_4(p_5 - c_4) + y_4(p_4 - p_5) + p_5 s_4\} \tag{8.63}$$

In addition, we have the following upper limits (constraints):

$$y_4 \le s_4 \quad \text{(for sales)} \tag{8.64}$$

$$x_4 \le k - (s_4 - y_4) \quad \text{(for ordering)} \tag{8.65}$$

In other words, analysis of stage 4 (second stage to be considered) leads us to the following linear-programming subproblem:

$$\max \{x_4(p_5 - c_4) + y_4(p_4 - p_5) + p_5 s_4\} = \max \{150x_4 - 100y_4 + 800s_4\} \tag{8.66}$$

s/t

$$y_4 \le s_4$$
$$x_4 - y_4 \le k - s_4$$

and $x_4, y_4 \ge 0$.

Since the problem involves only two independent variables, y_4 and x_4, we can easily solve it by the graphical method shown in Figure 8.9. We know that our optimal solution must lie in one of the corner points of the convex polygon $OABC$. Let us determine the value of the objective function at each of the corner points. These values are given in Table 8.15.

It is obvious from Table 8.15 that the highest value of the objective function occurs at point B. Our optimal solution, therefore, is given by

$$f_4(s_4) = \{700s_4 + 150k\} \tag{8.67}$$

Table 8.15

CORNER POINT	COORDINATES (x_4, y_4)	$f_4(s_4)$
O	$(0, 0)$	$800s_4$
A	$(0, s_4)$	$700s_4$
B	(k, s_4)	$700s_4 + 150k$
C	$(k - s_4, 0)$	$650s_4 + 150k$

[18] As dete[illegible] d in the optimal solution of stage 5 (the first stage considered)

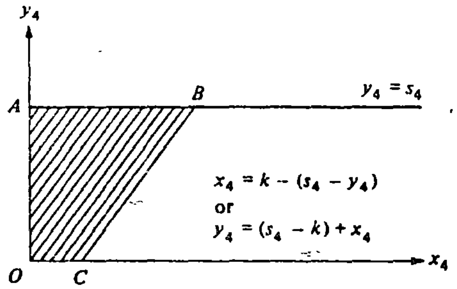


FIGURE 8.9

The optimal solution at point B corresponds to:

$$x_4 = k \quad \text{and} \quad y_4 = s_4$$

The fact that $x_4 = k$ means that we should order to full capacity, and then sell the entire stock during the fifth month. As Table 8.16 shows, this is a logical policy, since we pay only $650/ton in the fourth month and we can sell at $800/ton during the fifth month.

y x Table 8.16

	MONTH	SELL	BUY	s_i	PROFIT, DOLLARS
5	1	0	100	50	—
4	2	150	0	150	7,500
3	3	0	0	0	—
2	4	0	150	0	—
1	5	150	0	150	22,500
				TOTAL	30,000

The fact that $y_4 = s_4$ means that whatever stock we have at the beginning of the fourth month should be sold during the fourth month. The value of s_4, however, is not known at this stage of our analysis.

A similar analysis for stages 3, 2, and 1 can be made in order to determine optimal policies. The results are given in Table 8.16.

Before giving the functional relation for this problem, let us go back for a moment and examine Equations (8.61) and (8.62). Note that the optimal policy for the fifth month $f_5(s_5)$, is a function of s_4, x_4, and y_4. Also, as shown by (8.63) and (8.66), x_4 and y_4 can eventually be expressed as functions of k and s_4. As we proceed backward, stage by stage, we will find that the optimal policy for any stage can eventually be expressed as a function of k (the warehouse capacity) and s_1 (the starting inventory).

The functional equation for this problem is given by

$$f_{n-1}(s_{n-1}) = \max\{p_{n-1}y_{n-1} - c_{n-1}x_{n-1} + f_n(s_n)\} \tag{8.68}$$

Also note that

$$s_n = s_{n-1} + x_{n-1} - y_{n-1} \tag{8.69}$$

This is subject to two constraints:

$$y_n \leq s_n \tag{8.70}$$

which means we cannot sell more than our stock at a given stage, and

$$x_n \leq k - s_n + y_n \tag{8.71}$$

which means we cannot order more than our capacity to store at a given stage.

Note: Dynamic programming can solve linear-programming problems as shown here and in the allocation examples. Generally, the simplex method is much more efficient than dynamic programming. However, in certain linear programs that are dynamic in nature, such as the warehouse problem, dynamic programming may be used efficiently. Large problems are difficult to solve with most linear-programming codes. Dynamic programming in such cases is superior to linear programming.

8.4.6 AN INVESTMENT PROBLEM (BUYING CALL OPTIONS IN THE STOCK MARKET)

Several investment decisions can be multistage or multiperiod decisions. The outcome of each decision affects the decision conditions for subsequent stages.

For example, let us consider the following simplified problem:

1. An investor has the sum of \$5000 at the present time ($t = t_0$).
2. He wishes to buy six-month call options for a certain stock at \$1000 each.
3. It is assumed that there is a 60 percent chance of making a net profit of \$1000 for each six-month option. In such a case, the options will certainly be exercised.
4. It is assumed that there is a 40 percent chance of not exercising the options—that is, a net loss of \$1000 for each six-month option.

The objective of the investor is to determine an investment policy that will maximize the *chances of making a net profit* of \$3000 during the next 18 months. Considering each six-month period to be one stage, the investor's objective is to have a total of \$8000 at the end of the third stage.

Analysis

A possible investment alternative is to buy five options at time t_0 and hope for a "state" after six months that would result in a total of \$10,000. This alternative has the obvious danger that, if the stock price declines, our investor would lose all his capital at the end of the very first stage. In view of the stated objective, it is reasonable to suggest that the optimal investment policy would call for the purchase of less than five options (say two or three) at time t_0. Then, based on the outcome of the first stage, additional options would be purchased at the beginning of the second stage, and so on. In other words, the decision at the end of each stage would depend on the "state" of our investor's capital at that stage. This is a typical dynamic-programming problem.

Solution

As previously, we shall solve the problem by proceeding from the *last stage to the first stage*. Let us adopt the following notation:

k = number, in thousands of dollars, under the control of the investor, at the beginning of each stage

x_3, x_2, and x_1 = optimal number of options to be purchased, respectively, at the beginning of the third, second, and first stages

$f_3(k)$, $f_2(k)$, and $f_1(k)$ = highest possible probabilities of achieving the goal for a given k at the beginning of third, second, and first stages

We now proceed to investigate, for all possible levels of k, the respective optimal probabilities at the beginning of each stage.

Decision Conditions at the Beginning of Last (or Third) Stage

The time is one year after t_0, since each stage equals six months. Should the investor end the second stage with \$8000 ($k = 8$), he would have achieved his objective. In such a case the best policy for him would be *not* to buy any additional options (that is, $x_3 = 0$). If he ends the second stage with, say, \$10,000 ($k = 10$), he can, if he wishes, purchase one or two additional options ($x_3 = 1$ or $x_3 = 2$). Similarly, if he has \$7000 ($k = 7$), he must invest at least \$1000 ($x_3 = 1$) in order to have a 60 percent chance of achieving his goal (success), since with $k = 7$ if he does not invest, his chance of achieving his goal of \$8000 is zero.

In other words, for each level of k at the beginning of the third stage, we have an optimal x_3 and the probability of achieving the goal. These values are entered in Table 8.17.

In order to find the best policy $f_3(k)$, we select in each row of Table 8.17 the highest probability of success and the corresponding value(s) of x_3. For example, if $k = 4$ then the best policy is to buy four options, a course of action having a 60 percent chance of success.

Table 8.17

X_1 k	PROBABILITY OF SUCCESS WHEN, FOR A GIVEN k, x_3 EQUALS 0	1	2	3	4	$f_1(x_1$ $f_3(k)$	d_1 OPTIMAL x_3
0	0					0	
1	0					0	
2	0					0	
3	0					0	
4					0.6	0.6	4
5				0.6	0.6	0.6	3, 4
6			0.6	0.6	0.6	0.6	2, 3, 4
7	0	0.6	0.6	0 6	0.6	0.6	1, 2, 3, 4
8	1					1	0
9	1	1				1	0, 1
10	1	1	1			1	0, 1, 2

Decision Conditions at the Beginning of Second Stage

Table 8.17 contains the necessary information for choosing an optimal policy at the beginning of the third stage. The investor must now decide on an optimal x_2 with the objective of having at least $8000 at the end of the third stage and knowing $f_3(k)$ values from Table 8.17. The calculations are based on the following formula:

$$f_2(k) = \max_{x_2 \leq k} \{0.6 f_3(k+x_2) + 0.4 f_3(k-x_2)\} \tag{8.72}$$

For example, if $k=2$ at the beginning of the second period, we must purchase two options ($x_2=2$) in order to have a shot at $k=4$ (remember, probability of success equals 0.6) at the beginning of the third period. Also, as the values in Table 8.18 indicate, if k *indeed* equals 4 at the beginning of the third stage, the investor has a probability of 0.6 of having $k=8$ at the end of the third stage.

Now we construct joint probabilities. Thus, if $k=2$ at the beginning of the second stage, the probability of having $k=8$ at the end of the third stage is 0.36 ($0.6 \times 0.6 = 0.36$). This value is entered in Table 8.18. We apply Equation (8.72) and enter the results into Table 8.18.

Decision Conditions at the Beginning of the First Stage

At this stage ($t=t_0$), k takes on only one value, namely 5. The investor must now decide on the value of x_1, knowing $f_3(k)$ and $f_2(k)$ from Tables 8.17 and 8.18.

The calculations are based on the functional equation:

$$f_1(k) = \max_{x_1 \leq k} \{0.6 f_2(k+x_1) + 0.4 f_2(k-x_1)\} \tag{8.73}$$

Table 8.18

k	PROBABILITY OF SUCCESS WHEN, FOR A GIVEN k, x_2 EQUALS 0	1	2	3	4	5	6	7	$f_2(k)$	OPTIMAL x_2
0	0								0	
1	0	0							0	
2	0	0	0.36						0.36	2
3	0	0.36	0.36	0.36					0.36	1, 2, or 3
4	0.6	0.36	0.36	0.36	0.6				0.6	0 or 4
5	0.6	0.6	0.36	0.6	0.6	0.6			0.6	0, 1, 3, 4, or 5
6	0.6	0.6	0.84	0.6	0.6	0.6	0.6		0.84	2
7	0.6	0.84	0.84	0.84	0.6	0.6	0.6	0.6	0.84	1, 2, or 3
8	1								1	0

Since $k=5$, this relation becomes

$$f_1(5) = \max_{x_1 \leq k} \{0.6 f_2(5+x_1) + 0.4 f_2(5-x_1)\} \tag{8.74}$$

The computation is done by simple enumeration, and results are given in Table 8.19, an examination of which indicates that there are two equally good investment policies. These policies say that if we buy one or three options at the beginning of the first stage, we have a probability of 0.74 of achieving our goal of $k=8$ at the end of the third stage.

Table 8.19

k	PROBABILITY OF SUCCESS WHEN x_1 EQUALS 0	1	2	3	4	5	$f_1(k)$	OPTIMAL x_1
5	0.6	0.74	0.65	0.74	0.6	0.6	0.74	1 or 3

A schematic presentation of the two best policies at all stages is given in Figure 8.10.

The functional equation describing this problem is

$$f_{n-1}(k) = \max_{x_{n-1} \leq k} \{0.6 f_n(k+x_{n-1}) + 0.4 f_n(k-x_{n-1})\} \tag{8.75}$$

8.4.7 OTHER SMOOTHING PROBLEMS

Many business, engineering, and economics problems can be presented as one-dimensional smoothing problems. For example, consider a typical production scheduling problem involving known fluctuating demands and known

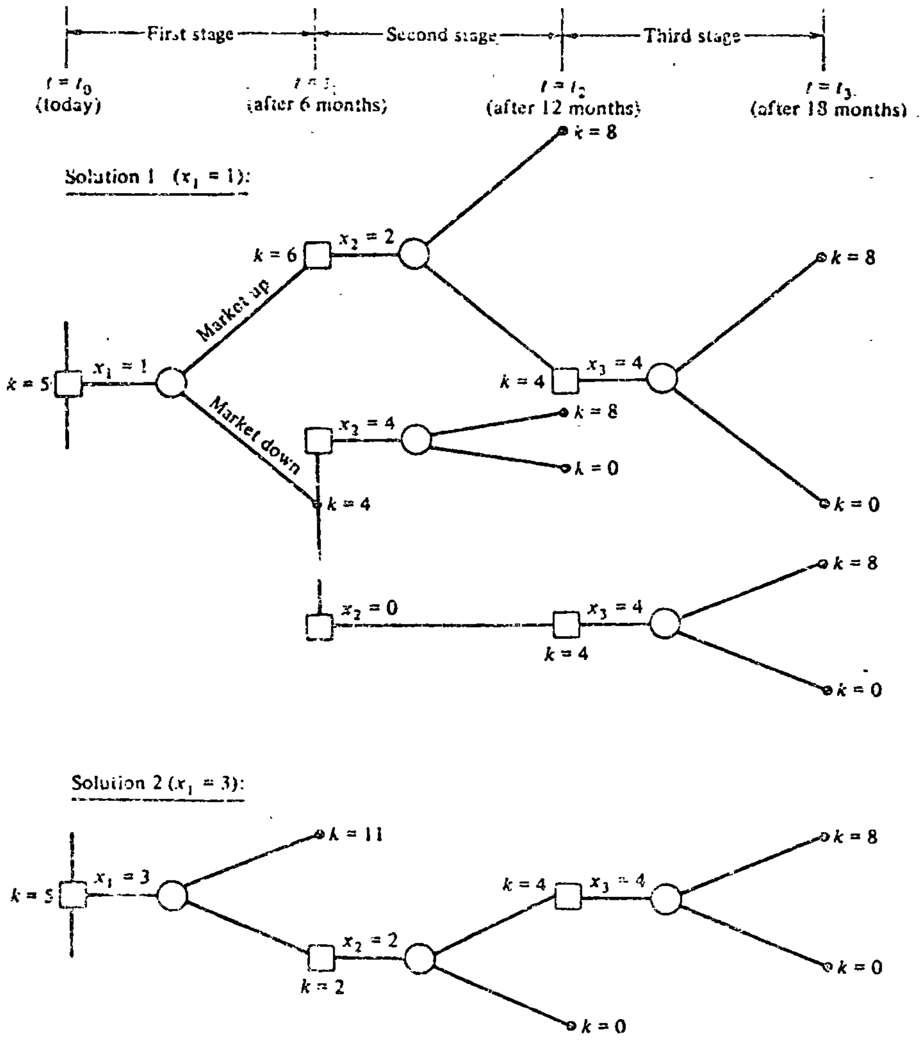


FIGURE 8.10

costs. Let us assume that the penalty for being out of stock is infinite. The objective is to satisfy all demand at minimum costs subject to the restrictions of available production resources (such as material and labor).

At the outset we can think of three approaches to finding a solution. First, regardless of the cost of hiring, training, layoff, and so on, we faithfully follow the demand curve by providing the required resources. This approach usually is not a minimum-cost approach. Second, we can schedule a constant work force for the entire planning horizon, accumulating finished-goods inventories during low-demand periods, and using excess inventory to supply the requirements of high-demand periods. Third, we can divide the planning horizon into several periods, and schedule a constant work force for each of these periods so that the excess inventories of periods of low demand can be used in periods of high demand.

Pg 409

In such a problem the structural constraints are usually linear and the objective function may be one of the following:

1. A linear cost function.
2. A convex cost function.
3. A nonconvex cost function.

When the cost function is linear, we can solve the problem by linear programming. Convex programming can be employed to solve problems with convex objective functions. In the case of a nonconvex cost function, however, dynamic programming is the only available analytical tool.

If, in addition to the structural constraints, we impose the requirement that the solution be an integer solution, the problem is very similar to the warehouse problem (Section 8.4.5). The problem is also similar to one of optimizing the employment level (Section 8.4.3). For detailed examples see Vazsonyi [30], pp. 79–87, 194–202, and 238–342.

Another problem is the *caterer problem*, which was solved by linear programming in Chapter 5 as a transportation problem. The problem has been solved as a dynamic programming problem by Bellman and Dreyfus [6].

Several other problems that were solved by other methods can also be solved by dynamic programming, sometimes more efficiently For example, the classical "*n* jobs sequencing through two machines" is presented by Bellman and Dreyfus [6] as a dynamic-programming problem.

8.4.8 MULTIDIMENSIONAL SMOOTHING AND SCHEDULING PROBLEMS

Many real-life situations can be formulated as multidimensional problems with more than one constraint. For example, in the employment-level problem, we could add another constraint by stating that the level of employment in any given month should not be less than a minimal level. In such cases, the dynamic-programming approach remains basically the same, but the solutions tend to get more complicated. (See Bellman and Dreyfus [6].)

8.5 MARKOV PROCESSES IN DYNAMIC PROGRAMMING

8.5.1 INTRODUCTION

Of the several attempts to construct a general dynamic programming formulation, the most successful model is the decision model suggested by Howard

centro de educación continua
división de estudios superiores
facultad de ingeniería, unam

APLICACIONES DE LACOMPUTADORA A LA SIMULACION Y OPTIMIZACION

CHAPTER 8. ELEMENTARY INVENTORY MODELES.

MARZO-ABRIL,1978.

chapter 8

ELEMENTARY INVENTORY MODELS

It is understandable that businessmen are concerned about the problem of inventories. It is not uncommon for a manufacturing company to have 25 percent or more of its total invested capital tied up in inventories. On December 31, 1969, the TRW Company had 26 percent of its assets in inventories and the Lockheed Aircraft Corporation had over $500,000,000, or about 39 percent of its assets represented by inventories. The General Electric Company had nearly $1,482,000,000 and the General Motors Corporation more than $3,700,000,000 in inventories in December, 1969. Naturally, if good inventory management could change any of these totals by as much as even a few percent, we are talking about really big money.

The current emphasis in management science began with the analysis of inventory systems. In 1915, F. W. Harris [9] developed the first economic lot size equation, and this was probably the beginning of the use of mathematical models to represent management problems. In 1931, F. E. Raymond published his *Quantity and Economy in Manufacture* [14] in which he developed this idea much further, attempting to account for a wide variety of conditions. In the postwar period, the management science literature has been filled with analyses of inventory and production control systems, partly because of the great interest shown by the government and the military, as well as the interest shown by such progressive companies as the Eastman Kodak Company, the Procter and Gamble Company, Johnson and Johnson, and many others.

Management Objectives and Costs

It is important that models of inventory systems reflect true incremental costs associated with alternate plans or policies. These costs represent "out-of-pocket" expenditures or foregone opportunities of profit. Cost figures derived from the normal accounting records usually do not fit the requirements. The following types of cost items are often incremental costs in inventory models: Costs depending on the number of lots, production costs, handling and storing costs, cost of shortages, and capital investment costs.

Costs Depending on Number of Lots. In deciding on purchased lot quantities, there are certain clerical costs of preparing purchase orders that are the same regardless of the quantity ordered. These costs are important in deriving economic purchase quantities as we shall see; however, the cost figure used must be the true incremental cost of order preparation. It is not correct to derive such a figure simply by dividing the total cost of the purchasing operation by the average number of purchase orders processed. A large segment of the total costs of the purchasing operation are fixed, regardless of the number of orders issued. There is, however, a variable component, and this is the pertinent figure. Quantity discounts and shipping costs are other factors which influence the quantity of materials purchased at one time and, therefore, influence the levels of material inventories. A question parallel to the purchase quantity occurs within a production system in deciding the size of production orders, that is, the number of units to process at one time. Here, the preparation costs are the incremental costs of preparing production orders, setting up machines, and controlling the flow of orders through the shop. Intraplant material handling costs affect purchase lot quantities.

Production Costs. Some of the components of production costs which have a bearing on inventory models such as set-up, change-over, and material handling costs, have been discussed in the preceding paragraph. Certain other incremental costs, however, also have a direct bearing on inventory models. For example, overtime premium and the incremental costs of production fluctuation, such as hiring, training, and separation costs need to be balanced against the cost of carrying additional inventory. In this latter context, system inventories become an important part of the development of production-inventory programs which we cover in Chapter 13.

Costs of Handling and Storing Inventory. There are certain incremental costs associated with the level of inventories. They are represented by the

costs of handling material in and out of inventory and storage costs, such as insurance, taxes, rent, obsolescence, spoilage, and capital cost. These incremental costs are commonly in proportion to inventory levels.

Cost of Shortages. An extremely important cost which never appears on accounting records is the cost of running out of stock. Such costs may appear in several ways. For example, within a production system a part shortage can cause idle labor on a production line or subsequent incremental labor cost to perform operations out of sequence, usually at higher than normal cost. There may be costs of avoiding shortages, such as expediting split lots. Shortage costs can be represented by profit foregone as when impatient customers take their business elsewhere. The realization of the importance of shortage costs raises the question, "What level of service is appropriate?"

Capital Costs. The opportunity cost of capital invested in inventory is an incremental cost of significance in designing inventory models. The cost figure itself is the product of inventory value per unit, the time that the unit is in inventory, ~~and the~~ appropriate interest rate. In general, the appropriate interest rate should reflect the opportunities for the investment of comparable funds within the organization, and, of course, it should not be lower than the cost of borrowed money. Since the funds are tied up in inventories they cannot be used for the purchase of equipment, buildings, or other profit-producing investments. There is, therefore, an opportunity cost of having funds invested in inventory, and inventory models reflect this cost.

Management Objectives. The overall objective of management is to design policies and decision rules which view inventories in a "systems" context so that the broadly construed set of costs we have discussed generally are minimized. In a production-distribution system, the functions of inventories and their effects on costs are distributed throughout the system from raw material intake through all intermediate stages to the final point of sale. The result is that there are interactions between basic inventory policy and production planning, labor policy, production scheduling, facilities planning, customer service, etc. Although there are some operations which may be regarded as almost purely inventory situations, the most usual structure involves an interaction between what we think of as the limited inventory problem and many of the broad policies for operating the enterprise as a whole. We shall begin our analysis of inventory systems with the more limited and simple concepts and attempt to build a structure of concept and technique which tries to account for many of the interactions with the environment in which inventories exist.

The Classical Inventory Model

The classical inventory model assumes the highly idealized situation shown in Figure 1. Q units are ordered or manufactured at one time. The order is placed when the inventory level falls to a point where the normal usage would just use up the inventory within the fixed procurement lead time. The receipt of the order of lot size Q is perfectly timed so that at just the point in time when the inventory balance falls to zero, the order of size Q is received, the inventory balance is increased by Q units, and the cycle repeats. We will find this model useful in establishing the overall concepts with which we will be dealing. Let us establish the following list of symbols:

TIC = total incremental cost
TIC_0 = total incremental cost of an optimal solution
Q = lot size
Q_0 = optimal lot size
R = annual requirements in units
c_H = inventory holding cost per unit per year
c_P = preparation costs per order
c_S = shortage costs per unit per year
N_0 = number of orders or manufacturing runs per year for an optimal solution

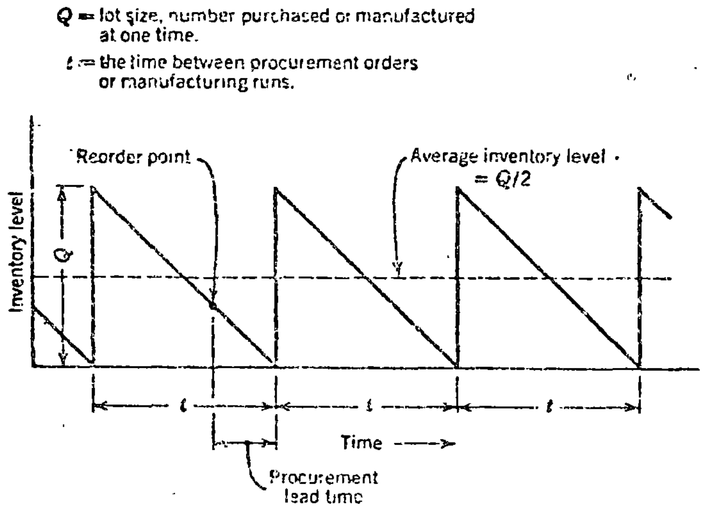


FIGURE 1. Graphic representation of inventory levels in the classical inventory model. Q = lot size, number purchased or manufactured at one time; t = the time between procurement orders or manufacturing runs. Lead time is less than order cycle time.

t = time between orders or manufacturing runs
t_0 = time between orders or manufacturing runs for an optimal solution

Objective. Our objective is to establish a mathematical model which expresses the relationship between Q, the variable under managerial control, and the incremental costs associated with the system. The incremental costs for the simple system we have defined are the costs associated with holding inventory and the costs associated with the procurement of an order of size Q. Therefore, the cost function we wish to minimize is:

$$TIC = \text{inventory holding costs} + \text{preparation costs}$$

We can see from Figure 1 that if Q is increased, the average inventory level, $Q/2$, will increase proportionately. If the inventory holding cost per unit per year is c_H, the annual incremental costs associated with inventory are

$$c_H \frac{Q}{2}$$

If the cost to hold a unit of inventory (interest costs, insurance, taxes, etc.) for a specific example was $c_H = \$0.10$, we could express the inventory holding cost function as $(0.10Q/2) = 0.05Q$. We could then plot this inventory holding cost function for different values of Q as we have done in Figure 2 curve (a).

Similarly, the annual preparation costs depend on the number of times that orders are placed per year and the cost to place an order. The number of orders required for an annual requirement of R will vary with the lot size Q of each order, or, R/Q. If it costs c_P to place an order, the annual preparation costs may be expressed as

$$c_P \frac{R}{Q}$$

If, for a specific example, $R = 1600$ units per year, and $c_P = \$5.00$, we could express the annual preparation costs as $(5.00 \times 1600/Q) = 8000/Q$. As with inventory holding costs, we can plot the preparation costs for this example for different values of Q, as we have done in Figure 2 curve (b).

Figure 2 curve (c) shows a graphic model of cost versus lot size, showing the total incremental cost curve, the calculations for which are shown in Table 1. Looking at either Table 1 or Figure 2 curve (c), we note that the minimum total incremental cost, TIC_0, occurs when 400 units are ordered

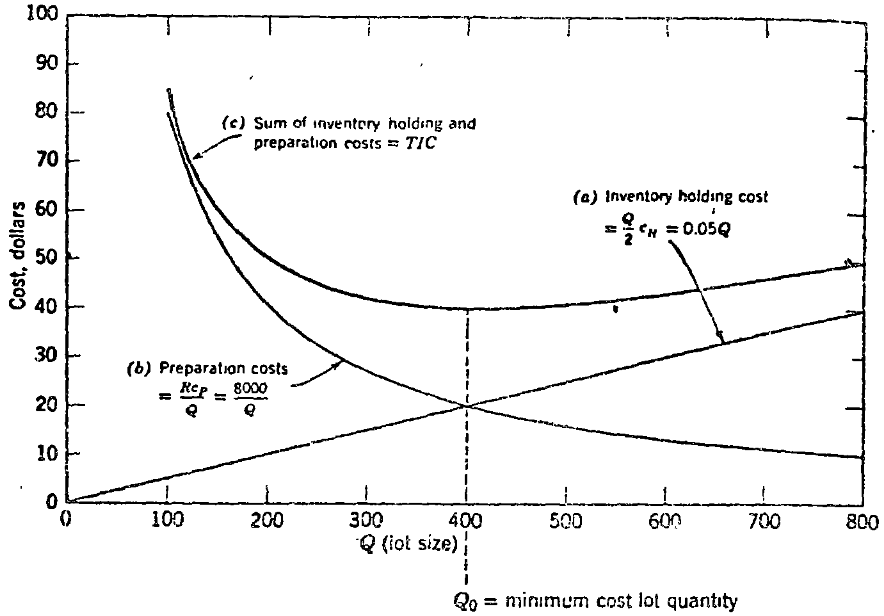


FIGURE 2. Graphic representation of classical inventory model. R = 1600 units per year, c_p = \$5.00, and c_H = \$0.10.

TABLE I. Computation of points for cost versus lot size for curves of Figure 2. R = 1600 units per year, c_p = \$5.00, and c_H = \$0.10

(1) Lot Size, Q	(2) Inventory Holding Cost $= \frac{Q}{2} \times c_H = 0.05Q$ (See Figure 2a)	(3) Preparation Costs $= \frac{Rc_P}{Q} = \frac{8000}{Q}$ (See Figure 2b)	(4) TIC = Sum of Columns (2) + (3) (See Figure 2c)
100	5.0	80.0	85.0
200	10.0	40.0	50.0
300	15.0	26.7	41.7
400 = Q_0	20.0	20.0	40.0
500	25.0	16.0	41.0
600	30.0	13.3	43.3
700	35.0	11.4	46.4

at one time. This is a solution for the specific data given, and we can see that the general form of the total incremental cost curve has a single minimum point. Note that though the intersection of the preparation and holding cost functions does correspond to the minimum point of the TIC function for this model. this is not generally true.

A General Solution. Regardless of the data used for specific examples the general form of the curves are similar to those shown in Figure 2, and we can, express the relationships in a completely general way,

$$TIC = \frac{c_H Q}{2} + \frac{c_P R}{Q} \tag{1}$$

This is an equation for the total incremental cost curve, and we wish to find a general expression for Q_0, the lot size associated with the minimum point of the total incremental cost curve. Mathematically, this may be done by finding the value of Q for which the slope of the total incremental cost curve is zero. Using the calculus, the first differential of (1) with respect to Q is:

$$\frac{d(TIC)}{dQ} = \frac{c_H}{2} - \frac{c_P R}{Q^2} \tag{2}$$

recalling that the rule for differentiation of a simple variable $x = ay^n$ is $dx/dy = nay^{n-1}$. For the first term of (1) the equivalent form which must be differentiated is $c_H Q^1/2$, where $c_H/2$ is equivalent to the constant, a. Therefore $d(TIC)/dQ = (1)(c_H/2)Q^{1-1}$, since $Q^{1-1} = Q^0 = 1$, $d(TIC)/dQ = c_H/2$.

Similarly, the equivalent form of the second term of (1) is

$$c_P R Q^{-1}$$

where $c_P R$ is equivalent to the constant, a. Therefore,

$$\frac{d(TIC)}{dQ} = (-1)(c_P R)Q^{-1-1} = -c_P R Q^{-2} = -\frac{c_P R}{Q^2}$$

The value of equation (2) is, in fact, the slope of the line tangent to the total incremental cost curve. We wish to know the value of Q when this slope is zero; therefore, we set (2) equal to zero, and solve for Q_0:

$$\frac{c_H}{2} - \frac{c_P R}{Q_0^2} = 0$$

$$Q_0^2 = \frac{2c_P R}{c_H}$$

and

$$Q_0 = \sqrt{2c_PR/c_H} \tag{3}$$

The cost of an optimal solution may be derived by substituting the value Q_0, in equation (1).

$$TIC_0 = \frac{c_HQ_0}{2} + c_P\frac{R}{Q_0}$$

$$= \frac{c_H}{2}\sqrt{2c_PR/c_H} + \frac{c_PR}{\sqrt{2c_PR/c_H}}$$

Combining the two terms with the common denominator

$$2\sqrt{2c_PR/c_H}$$

we have

$$TIC_0 = \frac{\cancel{c_H} \times \frac{\cancel{2}c_PR}{\cancel{c_H}} + \cancel{2}c_PR}{\cancel{2}\sqrt{2c_PR/c_H}} = \frac{2c_PR}{\sqrt{2c_PR/c_H}}.$$

and

$$\frac{\sqrt{(2c_PR)^2}}{\sqrt{2c_PR}} \cdot \sqrt{c_H} = \sqrt{2c_Pc_HR} \tag{4}$$

The number of orders or manufacturing runs per year N_0 and the time between orders or manufacturing runs t_0, for an optimal solution are:

$$N_0 = \frac{R}{Q_0} \tag{5}$$

$$t_0 = \frac{Q_0}{R} = \frac{1}{N_0} \tag{6}$$

If we substitute the values for R, c_P, and c_H used in our example, we obtain,

$$Q_0 = \sqrt{2 \times 1600 \times \frac{5.00}{0.10}} = \sqrt{160{,}000} = 400 \text{ units}$$

$$TIC_0 = \sqrt{2 \times 5.00 \times 0.10 \times 1600} = \sqrt{1600} = \$40$$

$$N_0 = \tfrac{1600}{400} = 4 \text{ orders or manufacturing runs per year}$$

$$t_0 = \tfrac{1}{4} = 0.25 \text{ years between orders or runs}$$

An Inventory Model with Shortage Costs

If the assumption that back orders are zero in the classical model is relaxed, we have the graphical structure of Figure 3. Our problem now is to determine the minimum cost order quantity when shortages are allowed at cost c_S. The inventory level rises to only I_{max} on the receipt of Q because the difference $Q - I_{max}$ is assumed to meet back orders instantaneously.

When shortage costs are accounted for, the classical model becomes slightly more general—the model represented by equation (3) being a special case. The rationale for derivation parallels that given for the simple

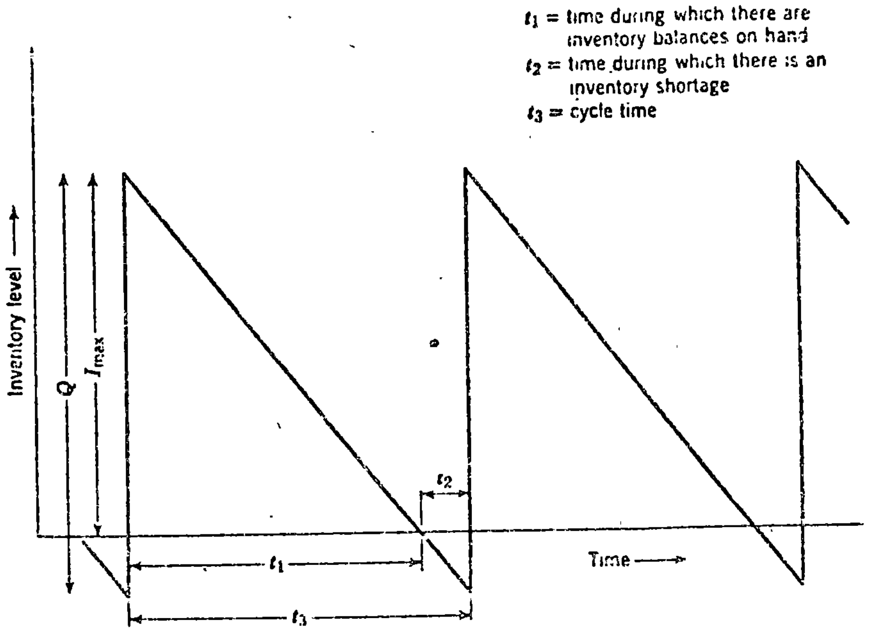

t_1 = time during which there are inventory balances on hand
t_2 = time during which there is an inventory shortage
t_3 = cycle time
Inventory level →
Q
I_{max}
t_2
t_1
t_3
Time →

FIGURE 3. Idealized structure of inventory levels with back orders of $Q - I_{max}$ allowed

case, but it is somewhat more complex mathematically. Derivations may be found in references [3, 7], and the resulting formulas are

$$Q_0 = \sqrt{2c_P R/c_H} \cdot \sqrt{(c_H + c_S)/c_S} \qquad (7)$$

$$TIC_0 = \sqrt{2c_P c_H R} \cdot \sqrt{c_S/(c_H + c_S)} \qquad (8)$$

$$I_{\max_0} = \sqrt{2c_P R/c_H} \cdot \sqrt{c_S/(c_H + c_S)} \qquad (9)$$

Note that when comparing equations (7) and (8) with the comparable formulas (3) and (4), with shortages, Q_0 is increased by the factor $\sqrt{(c_H + c_S)/c_S}$, and TIC_0 is decreased by the factor $\sqrt{c_S/(c_H + c_S)}$. The influence of shortages, then, is dependent on the relative size of c_H and c_S. If c_H is large relative to c_S, the effect of shortages on Q_0 and TIC_0 is considerable; that is, Q_0 will be increased and TIC_0 decreased compared to equations (3) and (4). If, on the other hand, c_H is small relative to c_S, minor changes in Q_0 and TIC_0 will result. The net effect of shortages costs on Q_0 and TIC_0 may at first seem to be strange. Recognize, however, that when the model permits shortages, average holding costs are reduced because of smaller average inventory balances. This will result in a larger Q_0. For the shortage case, TIC_0 is smaller than when shortages are not included because both holding costs and annual preparation costs are somewhat lower. For example, if we consider shortages in the previous example where $R = 1600$ per year, $c_P = \$5.00$ per order, $c_H = \$0.10$ per unit per year, and in addition, $c_S = \$0.50$ per unit per year, we have the following results:

$$\begin{aligned} Q_0 &= \sqrt{(2 \times 5.00 \times 1600)/0.10}\,\sqrt{(0.10 + 0.50)/0.50} \\ &= 400\sqrt{1.2} = 400 \times 1.095 = 438 \text{ units} \\ TIC_0 &= \sqrt{2 \times 5.00 \times 0.10 \times 1600}\,\sqrt{0.50/(0.10 + 0.50)} \\ &= 40\sqrt{0.833} = 40 \times 0.912 = \$36.50. \end{aligned}$$

The limiting values of c_S provide valuable insight. As c_S becomes infinitely large the factor in equation (7), $\sqrt{(c_H + c_S)/c_S}$, becomes 1 in the limit and we have the classical inventory model of equation (3). This corresponds to a policy of no shortages permitted. On the other hand if c_S is set at zero then the factor and therefore Q_0 becomes zero. This corresponds to a policy of infinite backordering, hand to mouth supply, or supply only on the basis of special order.

The Effect of Quantity Discounts

The basic economic lot size formula assumes a fixed price. When quantity discounts enter the picture, additional simple calculations will determine if there is a net advantage. As an illustration, assume the basic data of our previous example, that is, $R = 1600$ units per year, $c_P = \$5.00$ per order, and $c_H = 10$ percent per year. Recall that the economic order quantity was computed as 400 units. In addition, however, assume that the purchase prices are quoted as $1.00 per unit in quantities below 800 and $0.98 per unit in quantities above 800. If we buy in lots of 800 we save $32 per year on the purchase price plus $10 on order costs, since only two orders need to to be placed per year to satisfy annual needs. This saving of $42 per year must be greater than the additional inventory costs that would be incurred if the price discount is to be attractive. Under the 400 unit order size, average inventory is 200 units and inventory costs are 200 × 1.0 × 0.10 = $20. If orders of 800 units were placed, the inventory costs would be 400 × 0.98 × 0.10 = $39. There is a net gain of 42 − (39 − 20) = $23 by ordering in lots of 800 instead of in lots of 400. If the vendor had a second price break of $0.97 per unit for lots of 1600 or more, a similar analysis shows that the incremental inventory costs outweigh the incremental price and order savings, so that there is no net advantage in purchasing in lots of 1600. Table II summarizes the calculation for all three cases.

TABLE II. Incremental cost analysis to determine net advantage or disadvantage when price discounts are offered

R = 1600 Units per Year; c_P = $5.00 per Order, c_H = 10 percent per Year			
	Lots of 400 Units, Price = $1.00 per Unit	*Lots of 800 Units, Price = $0.98 per Unit*	*Lots of 1600 Units, Price = $0.97 per Unit*
Purchase cost of a year's supply (1600 units)	*$1600*	*$1568*	*$1552*
Ordering cost (c_P = $5.00 per order)	*20*	*10*	*5*
Inventory holding cost (avg. inv. × unit price adjustment × c_H)	*20*	*39*	*74*
	$1640	*$1617*	*$1631*

Formal Models with Price Breaks. We may generalize our ideas about the effect of quantity discounts by examining a formal model which takes price breaks into account. Recall that the lot size equation (3) did not need to consider price or value of the item because for every value of Q considered, the price was the same, that is, price was not an incremental cost. Let us now consider a lot size model which includes the value of the item as a factor. To reflect this idea, the total incremental cost associated with such a system may be expressed as follows:

$$
\begin{aligned}
TIC &= (\text{annual cost of placing orders}) \\
&\quad + (\text{annual purchase cost of } R \text{ items}) \\
&\quad + (\text{annual holding cost for inventory}) \\
&= c_P \frac{R}{Q} + kR + k\frac{Q}{2}F_H \qquad (10)
\end{aligned}
$$

where k = cost or price per unit, and F_H = *fraction* of inventory value, representing inventory holding cost on an annual basis ($kF_H = c_H$).

Following the rationale developed previously, we seek the value of Q, Q_0, which minimizes this total incremental cost equation. This leads to

$$Q_0 = \sqrt{2c_PR/kF_H} \qquad (11)$$

$$TIC_0 = \sqrt{2c_PkF_HR} + kR \qquad (12)$$

The derivations of equations (11) and (12) parallel the derivations of equations (3) and (4).

We may now use formulas (11) and (12) in the analysis of inventory systems which involve a price break. For comparison, let us assume the data of Table II for the first price break at b = 800 units. Recall that in this example, the price per unit below the break point was k_1 = \$1.00 and that above 800 units, the price was k_2 = \$0.98 per unit. To fit in with the present model, we will now express the inventory holding cost factor as F_H = 10 percent of inventory value. The other data remain the same, that is, R = 1600 units per year, and c_P = \$5.00 per order.

In the logic of our analysis, let us first note that the total incremental cost curve TIC_2 will fall below the curve TIC_1. This is shown in Figure 4. The logical thing to do, then, is to calculate q_{2_0} to see if it falls within the range P_2 where the price k_2 = \$0.98 applies. Doing this we find that q_{2_0} = 404 units, using equation (11), which is less than the break point, b = 800 units. Since 404 corresponds to the minimum [illegible] on the TIC_2 curve, we know that the lowest possible cost of TIC_2 within the range where the price k_2 applies is at the lot size b = 800 units. If it had happened that q_{2_0} was in the range P_2, this would have determined immediately that the economic lot size for the system, Q_0, was the value calculated q_{2_0}. Since this is not the case, however, we must continue our analysis to see if the minimum point on the curve TIC_1 is below TIC_2 at lot size b = 800 units. We may calculate TIC_{1_0} easily from equation (12), and its value is \$1640. Also, we may calculate TIC_b using equation (10), and this we find to be \$1617. The decision is now clear; $Q_0 = b$ = 800, since TIC_2 at lot size b is less than TIC_{1_0}.

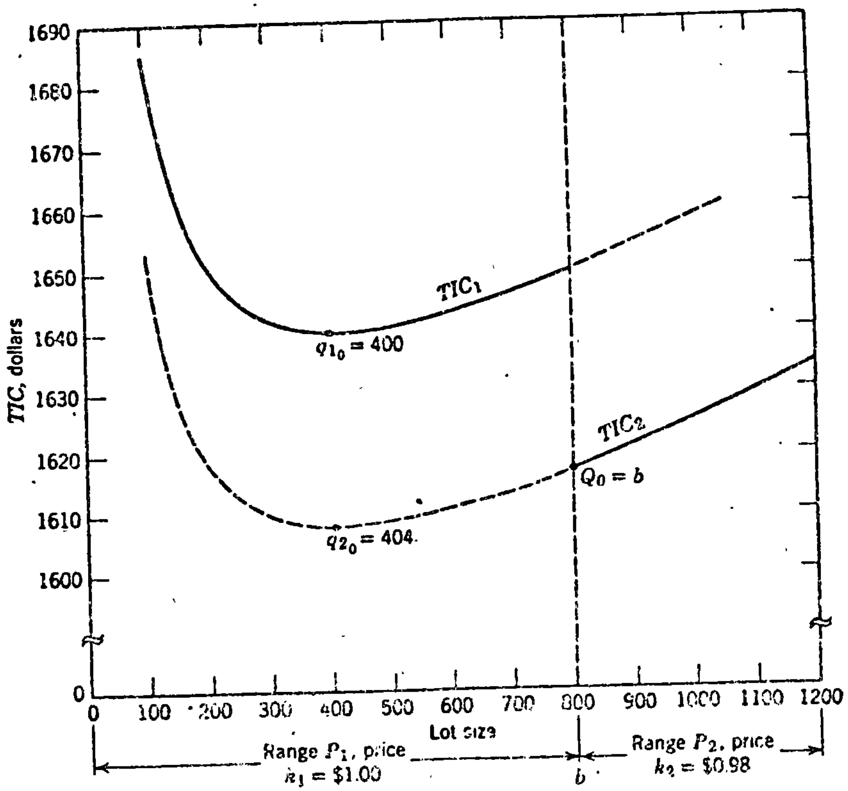


FIGURE 4. Total incremental cost curves for inventory model with one price break at b = 800 units. R = 1600 units per year, c_P = \$5.00, F_H = 10 percent of inventory value.

Compare these results with those obtained by the incremental cost analysis in Table II. This of course can be seen easily from the graphs of Figure 4. Constructing the curves for each case would be laborious, however, compared to the simple computations required to come to a decision. Figure 5 shows a decision flow chart for an inventory model with one price break, indicating the flow of calculations and resulting decisions. In some instances, the final result is obtained with one calculation, as when q_{2_0} falls in the lot size range P_2 where the price k_2 is valid. Where this is not the case, simple calculations for comparative total incremental cost yield a final result.

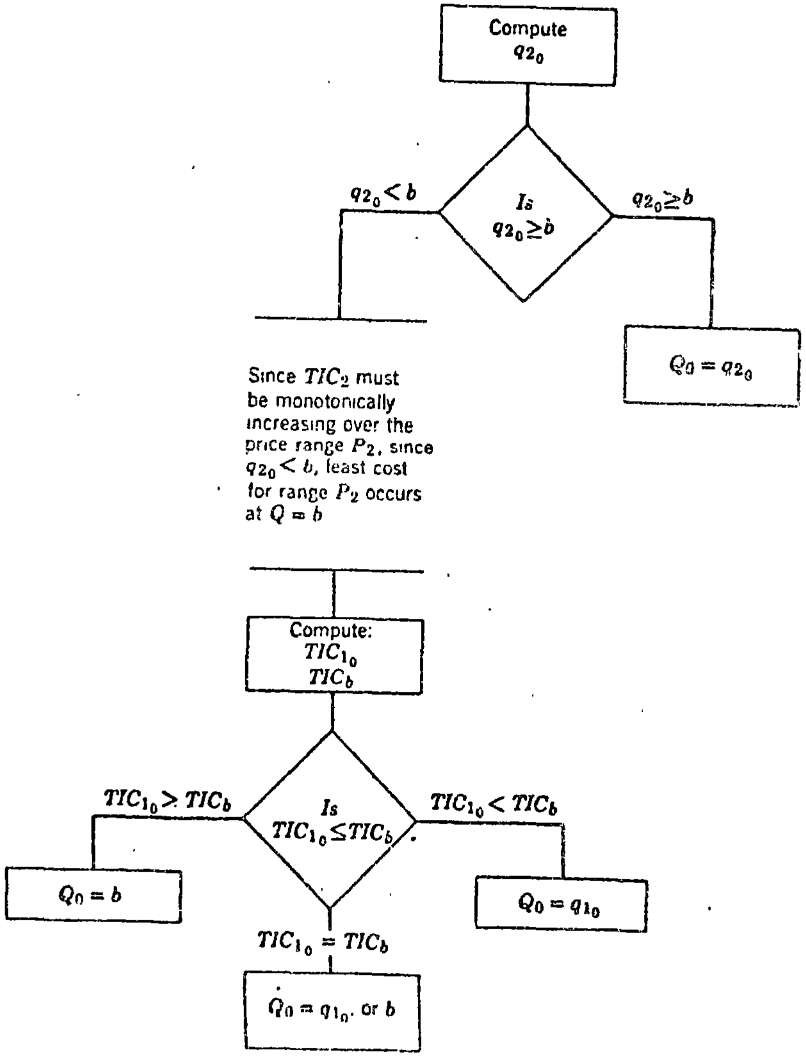


FIGURE 5 Decision flow chart for inventory model with one price break at the lot size b. Price [illegible] size range P [illegible] size range P.

Using the same general rationale we can develop decision processes for inventory models with two or more price breaks. Also, models could be constructed for quantity discount situations that also took account of other factors, such as shortage costs and the value added into inventory through the accumulation of preparation costs [4, 8].

Determining the Length of Production Runs

Production order quantities and runs are based on the same general concepts as purchase order quantities, as we have noted previously, but the assumption that the order is received and placed into inventory all at one time is often not true in manufacturing runs. For many manufacturing situations the production of the total order quantity Q takes place over a period of time, and the parts go into inventory, not in one large batch, but in smaller quantities as production continues. This results in an inventory pattern similar to Figure 6 when the run extends over a considerable period of time. When the run time is perhaps 30-60 percent of the total cycle time t shown in Figure 6, the effect on the average inventory of the system should be accounted for. Let r = daily usage rate and p = daily production rate—assuming, of course, that $p > r$. Other symbols remain as previously defined. During the production period t_p, inventory is accumulating at the rate of the difference between production rate and usage rate, $p - r$. This rate of increase continues for the production period t_p, so that the peak inventory is $t_p(p - r)$, and the average inventory is

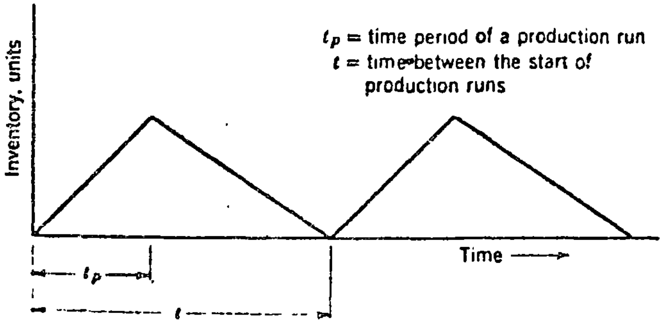


FIGURE 6 Diagram of inventory balance in relation to time when the lot Q is [illegible] over a period of time.

and since n is the same for all products, the total annual setup cost is

$$n \sum_{i=1}^{m} c_{P_i} \tag{22}$$

The total incremental cost associated with the entire set of m products is then

$$TIC = \text{annual setup cost} + \text{annual inventory holding cost}$$

$$= n \sum_{i=1}^{m} c_{P_i} + \frac{1}{2n} \sum_{i=1}^{m} c_{H_i} R_i \left(1 - \frac{r_i}{p_i}\right) \tag{23}$$

Our objective is to determine the minimum of the TIC curve with respect to n, the number of production runs. The first derivative of TIC with respect to n which we set equal to zero is

$$\frac{d(TIC)}{dn} = \sum_{i=1}^{m} c_{P_i} - \frac{1}{2n^2} \sum_{i=1}^{m} c_{H_i} R_i \left(1 - \frac{r_i}{p_i}\right) = 0$$

and the optimal number of runs, N_0 is

$$N_0 = \sqrt{\frac{\sum_{i=1}^{m} c_{H_i} R_i (1 - r_i/p_i)}{2 \sum_{i=1}^{m} c_{P_i}}} \tag{24}$$

The total cost of an optimal solution is found by substituting N_0 for n in (23), or

$$TIC_0 = N_0 \sum_{i=1}^{m} c_{P_i} + \frac{1}{2N_0} \sum_{i=1}^{m} c_{H_i} R_i \left(1 - \frac{r_i}{p_i}\right)$$

Substituting the expression for N_0 shown in (24) and simplifying leads to

$$TIC_0 = \sqrt{2 \sum_{i=1}^{m} c_{P_i} \sum_{i=1}^{m} c_{H_i} R_i \left(1 - \frac{r_i}{p_i}\right)} \tag{25}$$

Where R_i = annual requirements for the individual products, r_i = equivalent requirements per production day for the individual products, p_i = daily production rates for the individual products, c_{H_i} = [illegible] per unit, per year for the individual products, c_{P_i} = setup costs per run for the individual products, and m = the number of products.

Let us work out an example for the determination of the cycle length for the group of ten products shown in Table III, which shows the annual sales requirements, sales per production day, daily production rate, production days required, annual inventory holding costs, and setup costs.

TABLE III. Sales, production, and cost data for ten products to be run on the same equipment

(1) Product Number	(2) Annual Sales, Units R_i	(3) Sales per Production Day (250 days per year) r_i	(4) Daily Production Rate p_i	(5) Production Days Required	(6) Annual Inventory Holding Cost c_{H_i}	(7) Setup Cost per Run c_{P_i}
1	10,000	40	250	40	$0.05	$ 20
2	20,000	80	500	40	0.10	15
3	5,000	20	200	25	0.15	35
4	13,000	52	600	21.7	0.02	40
5	7,000	28	1000	7	0.30	25
6	8,000	32	800	10	0.40	37
7	15,000	60	500	30	0.02	42
8	17,000	68	500	34	0.05	50
9	3,000	12	200	15	0.35	16
10	1,000	4	125	8	0 10	12
				230.7		$292

Table IV then shows the calculation of the number of runs per year calculated by equation (24). The minimum cost number of cycles which results in four per year, each cycle lasting approximately 59 days and producing one-fourth of the sales requirements during each run. The total incremental cost from equation (25) is TIC_0 = $2420.

It is interesting to compare now the jointly determined number of runs per year with the number that would have resulted had runs been determined independently for each of the ten products. Table V summarizes these calculations. Note that products 4, 7, and 10 would have two or fewer runs per year, and products 2, 5, and 6 would have more than six runs per year. Magee [10] states a rule of thumb that if "the minimum-cost number of runs for the product alone, for any one or more products is less than half the value for all products, the product is a possible candidate

TABLE IV. Determination of the number of runs, jointly, for ten products from equation (24)

(1)	(2)	(3)	(4)	(5)	(6)
Product Number	Ratio r_i/p_i Col. 3/Col. 4 from Table III	$(1 - r_i/p_i)$	$c_{H_i}R_i$ = Col. 2 × Col. 6 from Table III	$c_{H_i}R_i$ $(1 - r_i/p_i)$ = Col. 3 × Col. 4	c_{P_i}, Col. 7 from Table III
1	0.160	0.840	$ 500	$ 420	$ 20
2	0.160	0.840	2000	1,680	15
3	0.100	0.900	750	675	35
4	0.087	0 913	260	237	40
5	0.028	0 972	2100	2,041	25
6	0.040	0.960	3200	3,072	37
7	0.120	0 880	300	264	42
8	0.136	0.864	850	734	50
9	0.060	0.940	1050	987	16
10	0.032	0.968	100	97	12
				$10,207	$292

$$N_0 = \sqrt{\frac{10{,}207}{2 \times 292}} \approx 4 \text{ cycles per year}$$

for only occasional runs." Table V also summarizes the total incremental cost which would result if the number of runs for each product were determined independently. The figure of $1932 is $488 less than the total incremental cost figure of $2420 given by equation (25) when runs are determined jointly. The apparent cost saving through individual determination of production runs is, of course, illusory because it does not take account of congestion costs or possible shortage costs that might result from independent scheduling. On the other hand, at low shop loads the interferences and schedule conflicts should not appear with independent scheduling.

TABLE V. Calculation of runs, independently for each product, from equation (17)

(1)	(2)	(3)	(4)	(5)	(6)
Product Number	$c_{H_i}R_i$ $(1 - r_i/p_i)$ from Col. 5, Table IV	c_{P_i}, from Col. 7, Table III	$\frac{\text{Col. 2}}{2 \times \text{Col. 3}}$	$N_i = \sqrt{\text{Col. 4}}$	TIC_0
1	420	$20	10 5	3.2	$ 130
2	1680	15	56.0	7.5	224
3	675	35	9.7	3.1	217
4	237	40	3.0	1.7	137
5	2041	25	40.8	6.4	101
6	3072	37	42.7	6.5	477
7	264	42	3.1	1.8	149
8	734	50	7.3	2.7	271
9	987	16	30.8	5 6	178
10	97	12	4.0	2.0	48
					$1932

SUMMARY

The models developed in this chapter are meant to build a general conceptual framework for the analysis of inventory systems. Although they may certainly be useful in some situations, they are not meant to be transplanted without modification into practical situations. Rather, they are meant to show some of the kinds of situations and factors that have been accounted for in simple inventory models. Actually, many more situations have been covered in the literature [2, 4, 6, 7, 8, 13, 15, 17, 19, 20]. With a knowledge of the general functions of inventories, management objectives, and the nature of costs which enter inventory models, we are in a position to consider the influence of variability of demand and basic systems of control which take account of these risks, as well as the effects on inventory planning of production planning and seasonal sales patterns.

REVIEW QUESTIONS

1. What is the nature of costs affected by inventories? Outline them and discuss each.

2 What are the kinds of costs related to inventories but dependent on lot quantities? In a practical situation, how do we derive these costs?

3. What are management's objectives in designing inventory systems? In the classical inventory model, which of the variables are controllable and which are outside the control of management?

4. What is the general effect of shortage costs on lot sizes?

5. Why must the classical lot size formula be modified if we are attempting to take quantity discounts into account?

6. Outline the rationale for determining the minimum cost purchase quantity Q_0 when a price discount is involved.

7. How is the determination of a production run different from the determination of a purchase lot size?

8. How does the production run problem change when a number of products share the use of the same equipment on a cyclical basis? Is the problem the same when the operating level is somewhat below capacity?

PROBLEMS

1. Compute the optimal lot size, Q_0, when R = 10,000 units per year, c_P = \$5, and c_H = \$10 per unit per year.

2. What is the total incremental cost for the conditions of Problem 1?

3. How much would Q_0 change if our estimate of c_P was in error and was actually only \$4 in Problem 1? What would be the difference in actual total incremental costs between the two solutions?

4. How much would Q_0 change if our estimate of c_H was in error, being only \$8, in Problem 1? What would be the difference in actual total incremental costs between the two solutions?

5. What is the effect on Q_0 for Problem 1 if shortages cost c_S = \$1 per unit per year? What is the total incremental cost of this solution?

6. Suppose that shortages are very expensive, perhaps \$100 per unit per year. What is the answer to Problem 5?

7. Suppose that for Problem 1 a price discount is offered so that orders placed in quantities below 125 cost \$100 each but for orders of 125 or above this quantity the price is \$95 each. Inventory holding cost is now expressed as 10 percent of the value of the item. In what quantities should the items be purchased? Use the rationale of Figure 5.

8. Determine the number of production runs for an item if R = 15,000 units per year, c_P = \$25, c_H = \$5 per unit per year, and p = 100 units per working day. There are 250 working days per year.

9. Determine the [illegible] production cycle for the following group of products assuming 250 working days per year.

Product Number	Annual Sales, Units	Daily Production Rate	Annual Holding Cost per Unit	Setup Cost per Run
1	5,000	100	\$1.00	\$40
2	10,000	75	0 90	25
3	7,000	50	0 30	30
4	15,000	80	0 75	27
5	4,000	40	1.05	80
Total				\$202

10. Carson Manufacturing Co. finds ordering costs for its materials and supplies fall into three major categories depending on urgency and the ordinary amount of follow up required. It therefore wishes to simplify its use of equation (3) for use by ordering clerks. For class 1, 2, and 3 items ordering costs are respectively \$5, \$15, and \$40.

(a) Derive formulas for the three classes of items.

(b) Further examination shows that inventory carrying cost is virtually constant at 18 percent of cost value for all items. Derive further simplified formulas for the three classes of items.

11. Carson Manufacturing Co. converted its entire ordering procedure to the EOQ basis described by problem 10b. On examining one of the Class 3 items (c_P = \$40), however, they noted very high annual freight costs under the new policy. Freight costs have been \$200 per order under the EOQ policy and would cost only \$400 for a carload lot of 500 units. R = 5000 units per year, and the average value of the item is \$222.22. Should Carson order in carload lots?

REFERENCES

1. Brown, R. G., *Decision Rules for Inventory Management*, Holt, Rinehart and Winston, 1967.
2. Buchan, J., and E. Koenigsberg, *Scientific Inventory Management*, Prentice-Hall, Englewood Cliffs, N.J., 1963.
3. Buffa, E. S., and W. H. Taubert, *Production-Inventory Systems: Planning and Control*, Richard D. Irwin, Inc., Homewood, Ill., Rev. ed., 1972.
4. Churchman, C. W., R. L. Ackoff, and E. L. Arnoff, *Introduction to Operations Research*, John Wiley & Sons, New York, 1957.
5. Eilon, S., *Elements of Production Planning and Control*, Macmillan Company, New York, 1962.
6. Fetter, R. B., and W. C. Dalleck, *Decision Models for Inventory Management*, Richard D. Irwin, Homewood, Ill., 1961.

7. Hadley, G., and T. M. Whitin, *Analysis of Inventory Systems*, Prentice-Hall, Englewood Cliffs, N. J., 1963.
8. Hannssman, F., *Operations Research in Production and Inventory Control*, John Wiley & Sons, New York, 1962.
9. Harris, F., *Operations and Cost* (Factory Management series), A. W. Shaw Company, Chicago, Ill., 1915, pp. 48-52.
10. Magee, J. F., and D. M. Boodman, *Production Planning and Inventory Control*, McGraw-Hill Book Company, New York, 2nd ed., 1967.
11. McMillan, C., and R. F. Gonzalez, *Systems Analysis—A Computer Approach To Decision Models*, Richard D. Irwin, Inc., Homewood, Ill., Rev. ed., 1968, Chap. 6.
12. Moore, F. G., and R. Jablonski, *Production Control*, McGraw-Hill Company, New York, 3rd ed., 1969.
13. Naddor, E., *Inventory Systems*, John Wiley & Sons, New York, 1966.
14. Raymond, F. E., *Quantity and Economy in Manufacture*, D. Van Nostrand Company, Princeton, N.J., 1931.
15. Starr, M. K., and D. W. Miller, *Inventory Control: Theory and Practice*, Prentice-Hall, Englewood Cliffs, N.J., 1962.
16. Thierauf, R. J., and R. A. Grosse, *Decision Making Through Operations Research*, John Wiley & Sons, Inc., New York, 1970, Chap. 7.
17. Veinott, A. F., "The Status of Mathematical Inventory Theory," *Management Science*, Vol. 12, No. 11, July 1966 (includes an exhaustive bibliography at end of paper).
18. Voris, W., *Production Control*, Richard D. Irwin, Homewood, Ill., 3rd ed., 1966.
19. Wagner, H. N., *Statistical Management of Inventory Systems*, John Wiley & Sons, New York, 1962.
20. Whitin, T. M., *The Theory of Inventory Management*, Princeton University Press, Princeton, N.J., 1953.

chapter 9

INVENTORY CONTROL SYSTEMS

Some of the major defects in the models developed in the previous chapter, so far as practical inventory policy is concerned, are the assumptions that requirements were known exactly and that the delivery of replenishment orders was perfectly timed. Also, those models did not place the inventory system in the context of the operating environment of the broader production-distribution system. In this chapter we shall attempt to introduce the idea of variability of demand and its influence on inventory policy, consider comparative systems for inventory control which take account of demand variability, and consider the impact of inventories on the problem of controlling production levels. In part V we shall develop models which focus on the impact of inventories in making aggregate production plans or programs, particularly in Chapter 13.

Variability of Demand

The source of our problem in dealing with variability of demands or requirements is focused on the lags inherent in the system for replenishment. If we could fill requirements immediately, there would be no problem. The elements of the problems caused by lags in the system were introduced in Chapter 7.

As we know, the demand for an item may vary considerably due to random causes, upward or downward trends in demand, seasonal and cyclic variations. Figure 1 shows a sales curve which demonstrates three of these factors (cyclic variations dealing with the concept of the business cycle are beyond our scope). Let us begin with a consideration of expected random variation and how realistic inventory policy might take it into account. Figure 2 abstracts from Figure 1 just the random variations in sales from average expected levels. Such a distribution could be abstracted from sales records from which the trend and seasonal factors have been removed, through commonly known statistical procedures. The residual variation is then simply the variation due to chance causes, comparable in every way to expected random variation in any process.

Buffer Stocks. The variations in demand are absorbed through the provision of buffer stocks which must be maintained because of our inability to forecast random variations in demand of the type shown by Figure 2. The size of these planned extra inventories depends on the stability of demand in relation to our willingness to run out of stock. If we are determined almost never to run out of stock, these planned minimum balances must be very high. If service requirements permit stock runouts and back ordering, the safety stocks can be moderate. Figure 3 shows the general structure of inventory balance with a fixed-order quantity system.

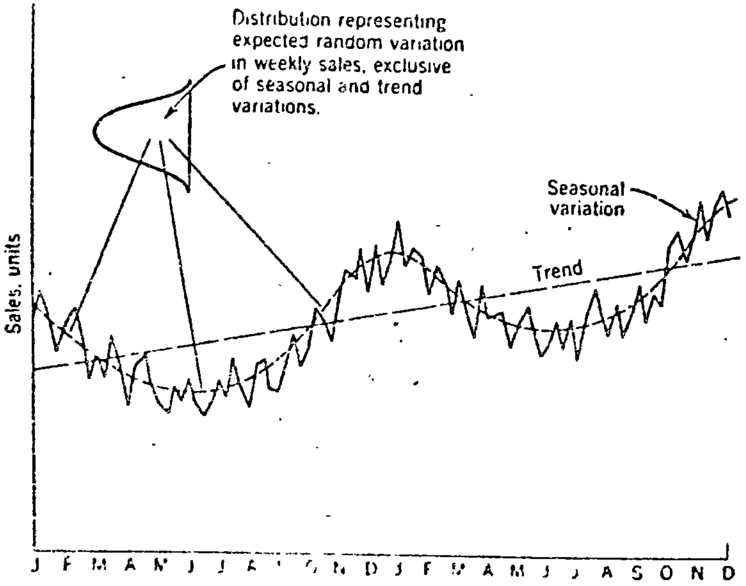


FIGURE 1 *Three [illegible] in demand which [illegible] inventory [illegible]*

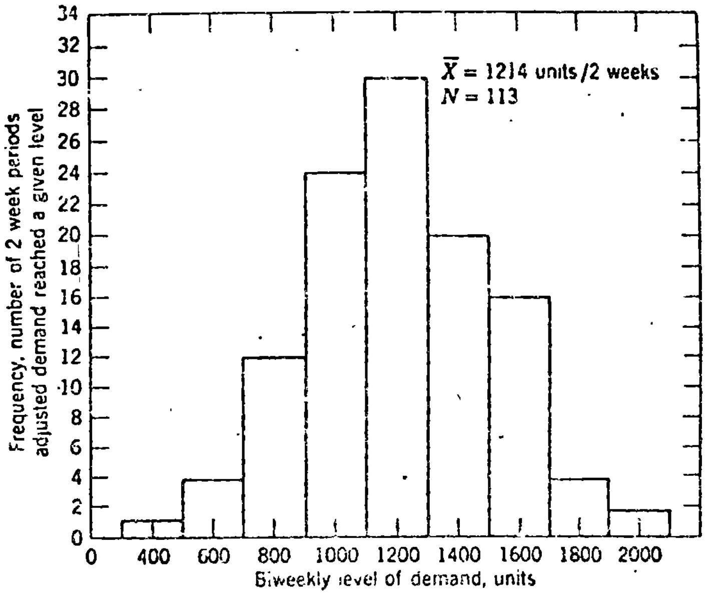


FIGURE 2. Distribution representing expected random variation in weekly sales, exclusive of seasonal and trend variations.

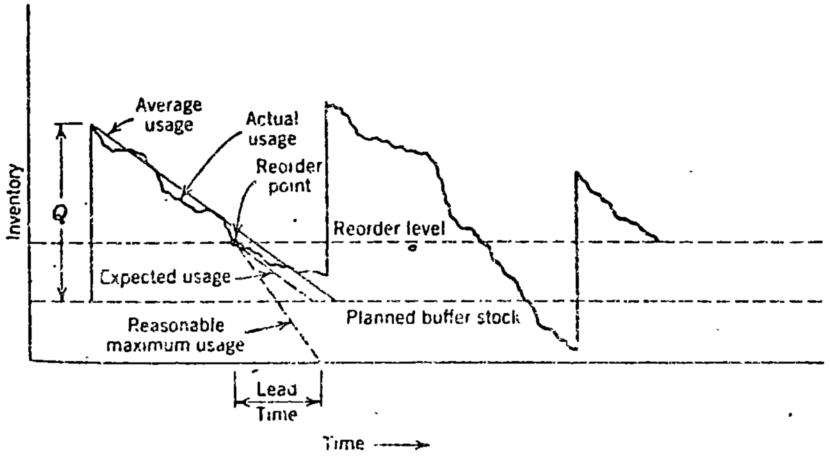


FIGURE 3 Structure of inventory balance for a fixed order quantity system, with safety stocks to absorb fluctuations in demand and in supply time. The buffer stock level is set so that a reasonable figure for maximum usage would [illegible] the inventory to zero during the lead time. Q is a fixed quantity [illegible]

The buffer stock level is set so that inventory balances would be drawn down to zero during the constant lead time for supply, if we should experience near-maximum demand.

The rational determination of buffer stocks, then, turns on a knowledge of the probability distribution of demand together with a decision regarding the risk of stock runout that we are willing to accept. To be most useful, the probability distribution of demand can be expressed in a form shown by Figure 4. Figure 4 was constructed from Figure 2, first, by plotting the number of periods that adjusted demand exceeded a given level, second, by establishing a percentage scale to represent a derived probability scale, and third, by idealizing the distribution as shown by the dashed curve of Figure 4. Since the approximate average two-week usage is 1214 units, and assuming a normal lead time of 2 weeks, we could be 90 percent sure of not running out of stock by having the 1520 units on hand when the replenishment order is placed. The buffer stock is then 1520 − 1214 = 306 units. If we wish to be 95 percent sure of not running out of stock, the buffer stock must be 1640 − 1214 = 426 units. Similarly, to be sure that we have only a 2 percent risk of running out of stock, the buffer stock level must be increased to 768 units.

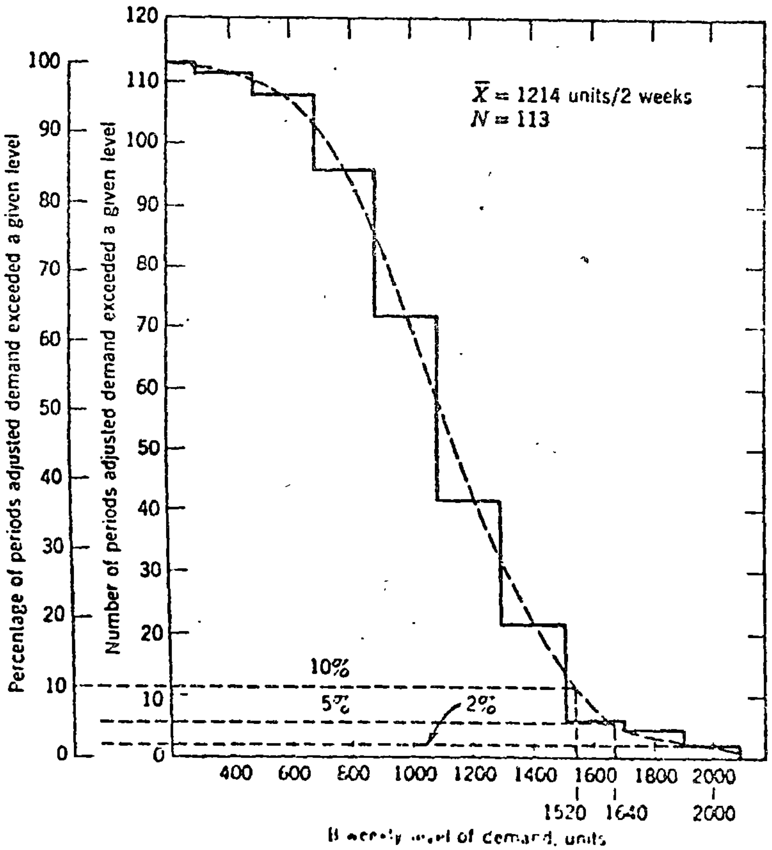


FIGURE 4 Distribution of percent of periods that demand exceeded a given level, developed from Figure 2

It is easy to see from the shape of the demand curve that, for high levels of protection, the buffer stock required goes up rapidly, and, therefore, the cost of providing this assurance goes up. This is shown by the calculations in Table I where we have assumed the demand curve of Figure 4, assigning a value of $50 to the item and inventory holding costs of 20 percent of value. The average inventory required to cover expected maximum usage rates during the lead time of 2 weeks is calculated for the three service levels shown. To offer service at the 95 percent level instead of the 90 percent level requires an incremental $1200 per year, but to move to the 98 percent level of service from the 95 percent level requires an additional $3600 in inventory cost.

The demand curve, then, provides a rational basis for the determination of buffer stock levels by helping to establish a reasonable maximum usage rate during the lead time. To establish this rate, however, management must decide what risk of stock runout is acceptable. In some instances this must be a judgment, but where a cost of shortages can be realistically assigned, a simple incremental cost analysis can determine whether additional protection is worthwhile. For example, for the data shown in Table I, there would be an incremental saving of $3156 in moving from the 90

TABLE I. Cost of providing the three levels of service shown in figure 4, when the item is valued at $50 each and inventory holding costs are 20 percent

	Service Level		
	90%	*95%*	*98%*
Expected maximum usage for 2-week replenishment time	*1520*	*1640*	*2000*
Buffer stock required	*306*	*426*	*786*
Average inventory required for service level during replenishment period = $(I_{max} - \text{Buffer})/2 + \text{Buffer}$	*913*	*1033*	*1393*
Value of average inventory at $50 per unit	*$45,650*	*$51,650*	*$69,650*
Inventory cost at 20%	*$ 9,130*	*$10,330*	*$13,930*

percent to the 95 percent level of service if the cost of a shortage was \$1 each (1214 × 26 × 0.05 × 1.00). This incremental gain exceeds the incremental cost of \$1200 shown in Table I. On the other hand, to move from the 95 percent to the 98 percent level the incremental gain is only \$950 whereas the incremental cost is \$3600 as shown in Table I. The 98 percent level of service is obviously too expensive in this instance.

In summary, we have a fairly general procedure. To determine buffer stocks, we must determine reasonable maximum usage rates during the lead time, and this requires the derivation of a demand distribution which reflects only the variation due to random fluctuations. Here, however, management must decide on a risk level for running out of stock, or if realistic shortage costs can be assigned, an incremental cost study can be made to determine the best risk level. If demand for the item is subject to seasonal variation or an upward or downward trend, the average of the distribution shifts, and it is necessary to reassess buffer stock level periodically. In such an instance, it would be better to express the demand distribution curve shown in Figure 4 in terms of deviations from expected mean values.

Practical Methods for Determining Buffer Stocks

The generalized methodology for setting buffer stocks when lead times are constant (just discussed) is too cumbersome for use in practical systems where large numbers of items may be involved. Computations are simplified considerably if we can justify the assumption that the demand distribution follows some particular mathematical function, such as the normal, Poisson, or negative exponential distributions. The general procedure is the same for all distributions: (*a*) determine the applicability of the normal, Poisson, or negative exponential distribution of demand *during lead time*, (*b*) establish a service level based on managerial policy or an assessment of the balance of costs, (*c*) define D_{max} during lead time based on the appropriate distribution and the service level, for example, if we have selected a service level of 10 percent then D_{max} is 1520 units in Figure 4, and (*d*) compute the required buffer stock from $B = D_{max} - \overline{D}$ where $\overline{D}$ is average demand and both D_{max} and $\overline{D}$ are based on the demand distribution over the constant lead time.

The three distributions have been found to be applicable in a number of situations at different stages in the supply-production-distribution system. For example, the normal distribution has been found to describe adequately many demand functions at the factory level, the Poisson distribution at the retail level, and the negative exponential distrib-

at the wholesale and retail levels [4]. When both demand and lead time are variable the determination of buffer stocks is more complex. In this situation, we are faced with an interaction between fluctuating demand and fluctuating lead times, and there is no simple mathematical analysis. Nevertheless, buffer stocks can be determined through a process of Monte Carlo simulation as long as we have a knowledge of the demand and lead time distributions. Since the simulation methodology is being used in such a situation, the distributions need not be described by any of the standard mathematical ones. Detailed examples of inventory models with variable demand and lead time are developed in reference [11].

Basic Inventory Control Systems

In attempting to develop automatic control systems for inventories, it is necessary to take account of random fluctuations in demand as just discussed and actual shifts in average demand of either a seasonal or long-term nature. The variables of the system which can be manipulated by management to develop a control system are the size of the replenishment order, the frequency of replenishment orders, the frequency of review and forecast of usage levels, and the method of information feedback on which the reviews are based. Alternate inventory control systems blend these factors in somewhat different ways.

The Fixed-Order Quantity System. This system is diagrammed in Figure 3. The system has a reorder level set which allows the inventory level to be drawn down to the buffer stock level within the lead time if average usage rates are experienced. Replenishment orders are placed in a fixed predetermined amount (not necessarily the minimum cost quantity, Q_0) timed to be received at the end of the lead time. The maximum inventory level becomes the order quantity Q plus the buffer stock I_{min}. The average inventory expected is, then $I_{min} + Q/2$. Usage rates are reviewed periodically in an attempt to react to seasonal or long-term trends of the type shown in Figure 1. At the time of the periodic reviews, the order quantity and buffer stock levels may be changed to reflect the new conditions. Demand for an item is ordinarily taken from the subsequent operation. Assume that we are considering the can of the capacitor shown in Figure 4 of Chapter 3. The capacitor is made in three sizes of electrical capacity. The can which houses the capacitor, however, is identical for all three sizes.

Figure 5 shows the chain of demand for the can as reflected back through the series of stock points and manufacturing operations. Customer orders are placed at the warehouse which maintains an inventory with controls

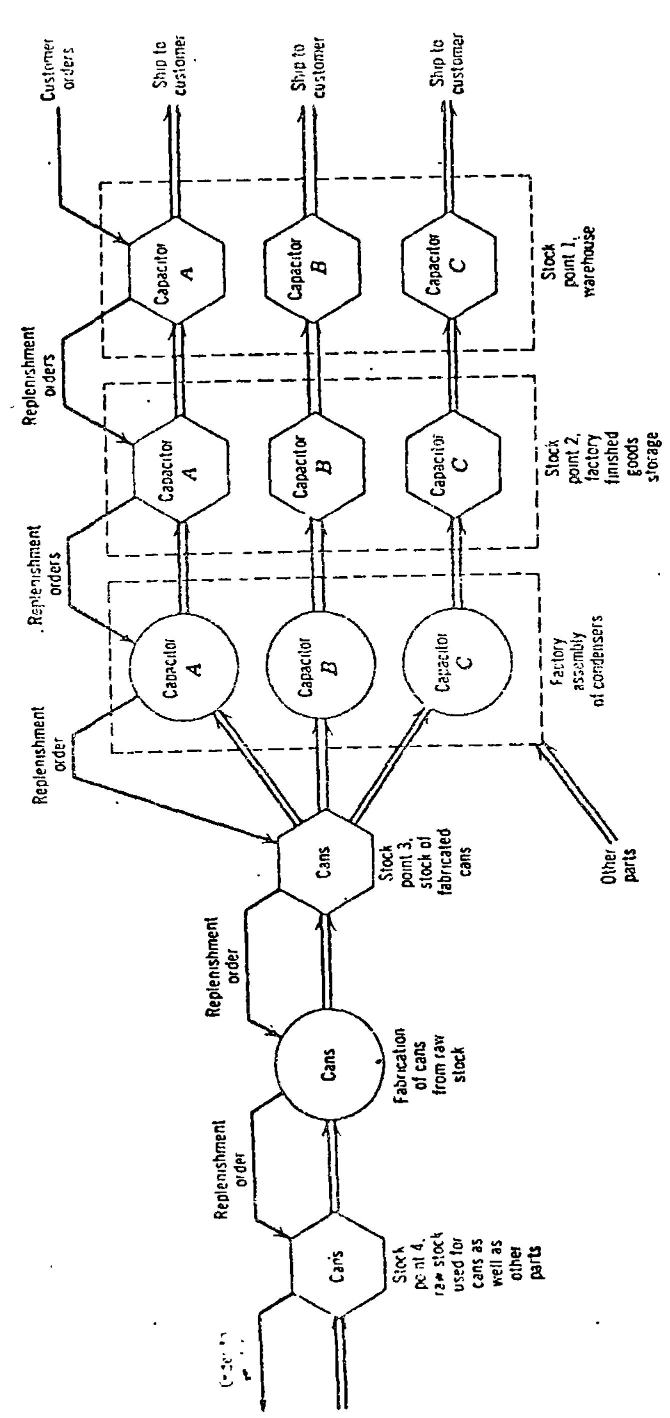


FIGURE 5 Chain of demand for the capacitor can, shown in Figure 4 of Chapter 3.

as described by Figure 3. When the warehouse inventory level falls to the reorder point, a replenishment order is sent to the factory, and the factory ships from its finished goods stock. When the finished goods inventory falls to a reorder point, however, a requisition is sent to the manufacturing department, and more condensers are assembled. To assemble the condensers, however, cans and other parts are requisitioned from stock point 3, a stock of fabricated cans. When the stock of fabricated cans falls to a reorder level, a shop order is written for a run of cans to be fabricated. The shop order requires raw stock which is drawn from stock point 4, raw material storage. When the inventory for the raw material falls to the reorder level, a purchase requisition is issued to vendors for replacement. Thus the demand for the capacitor can is reflected back in a chain involving 4 stock points and 2 factory operations. Figure 5 represents the structure of the information feedback system.

Fixed-reorder quantity systems are common where a perpetual inventory record is kept and with low-valued items such as nuts and bolts, where the inventory level is under rather continuous surveillance so that notice can be given when the reorder level is reached. One of the simplest methods for maintaining this close watch on inventory level is the use of the "two bin" system. In this system, the inventory is physically separated into two bins, one of which contains an amount equal to the reorder level. The balance of the stock is placed in the other bin, and day-to-day needs are drawn from it until it is empty. At that point it is obvious that the reorder level has been reached, and a stock requisition is issued. From that point on, stock is drawn from the second bin, which contains an amount equal to average usage over the lead time plus a buffer stock. When the stock is replenished by the receipt of the order, the physical segregation into two bins is made again and the cycle is repeated.

Fixed-reorder Cycle Systems. These systems focus control on a periodic basis, so that orders are placed weekly, monthly, or by some other cycle. The size of the order, however, is varied for each cycle to absorb the fluctuations in usage from period to period, as shown by Figure 6. The amount ordered covers normal usage during the procurement lead time plus the quantity necessary to replenish inventories to the level required for one cycle's usage plus buffer stock. This is, of course, the I_{max} level shown on Figure 6. Just as with lot size models, optimal relationships for the reorder cycle and I_{max} can be derived. See references [9, 13]. As with the fixed-quantity system, periodic reviews of usage rates are required to react to changes in the average usage rates of the type shown in Figure 1. Fixed-reorder cycle systems are prominent with higher valued items and where a large number of items are regularly ordered from the same vendor. With fixed-cycle ordering, freight cost advantages can often be gained by grouping these orders together for shipment. The common information feedback system for fixed-cycle systems is diagrammed in Figure 5, based on a chain of demand.

The main operating difficulties with the fixed cycle system described lie in the time lags in the information chain, and the apparently irresistible temptation to outguess shifts in requirement rates. The shifts in usage rates are most often simply random shifts, and the buffer stock has been designed to absorb these variations. If we respond to these random shifts in requirements we will surely drive ourselves insane. Suppose we are ordering on a monthly cycle the fabrication of cans for stock point 3 of Figure 5. Average requirements have been 500 cans monthly, but last month's requirements jumped to 600 units. If we assume that this will be a continuing requirement, we might decide to place an order for the current month which not only replenishes the 600 units drawn, but adds another 100 units to build up inventory to meet the expected continuation of 600 units per month. This makes a total order of 700 units. Suppose, however, that last months increase was simply a random fluctuation, and in a true expression of the capriciousness of random processes, requirements for this period turn out to be only 300 units. We now have a 400-unit excess inventory, and we need place an order for only 100 units for the coming period to meet average requirements. The result is that the random variations in demand from 600 units to 300 units have been translated into variations in shop orders for cans ranging from 700 units to 100 units. Demand variability has been amplified, leading to severe problems on the production floor in attempting to accommodate these wide variations.

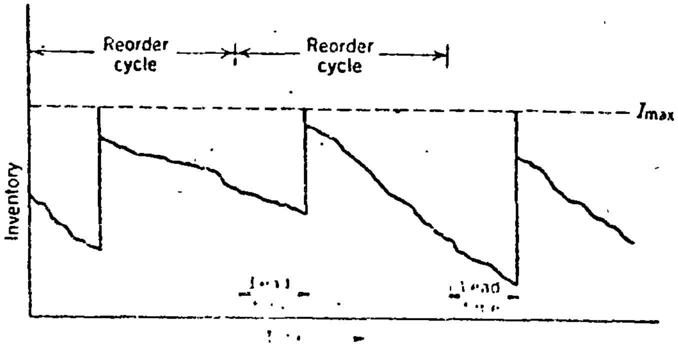


FIGURE 6 Fixed reorder cycle system of control. An order is placed at [illegible] intervals which replenishes stock based on the inventory balance on hand and on order plus the amount needed for one cycle

The question of amplification of demand variability is of extreme importance in designing stable production-inventory control systems, and we shall consider it more carefully at a later point. The immediate question is, however, "How can we tell if a change in demand is merely a random fluctuation or a true shift in average requirements?" We have an obvious application of the principles of statistical control. Appropriate control limits could be established and requirements plotted in relation to the control limits. Variations in requirements that fall within the control limits may be ignored, since buffer stocks were designed to absorb them. When points fall outside the control limits the question may be raised whether planning figures for average requirements should be revised. Even then, adjustments in planning figures for requirements should be relatively modest, taking a wait and see attitude, in order to avoid the costly results of fluctuations as in the situation described in the previous paragraph.

Control Theory Applied to Inventory Systems. Engineers have been interested in the design of automatic control systems, and the result has been the development of concepts and systems of control which have been

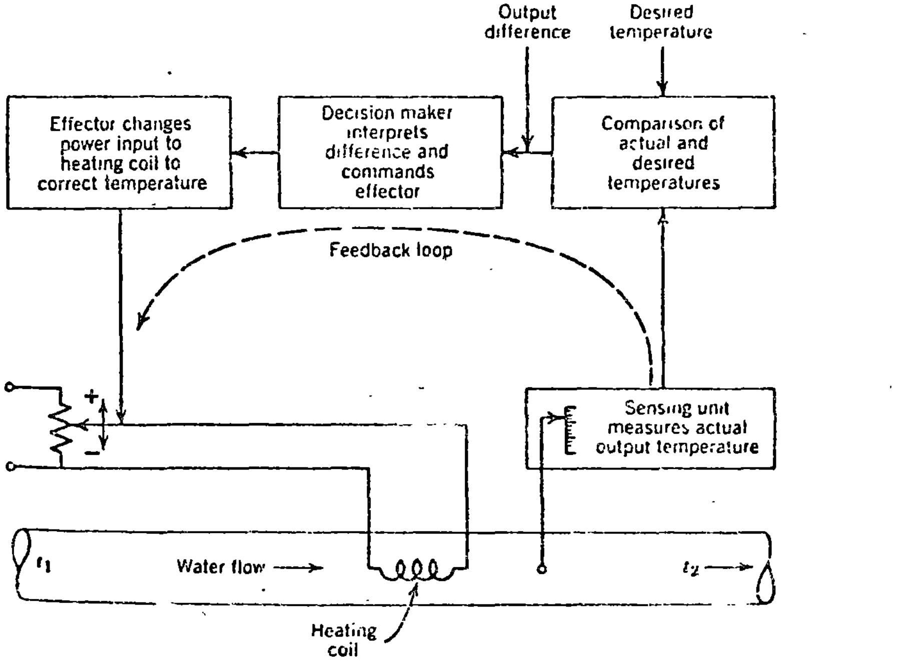

Output difference
Desired temperature
Effector changes power input to heating coil to correct temperature
Decision maker interprets difference and commands effector
Comparison of actual and desired temperatures
Feedback loop
+
−
Sensing unit measures actual output temperature
t_1
Water flow
t_2
Heating coil

FIGURE 7 Diagram of elements of an automatic system for maintaining a fixed output temperature of water flow. The basic components of the feedback loop are common to automatic control systems.

applied largely in automation and other physical systems. These self-correcting systems establish automatic control over some variable (a dimension, temperature, pressure, etc.) through a feedback loop. Conceptually, the feedback loop is comprised of some *sensing unit* which measures the output of the variable being controlled, a *comparator* which compares the actual output with the desired level, and a *decision maker* which interprets the error information and finally commands the *effector* to make a correction in the proper magnitude and direction so that output will meet standards. Figure 7 shows a schematic representation of the maintenance of the temperature of flowing water under automatic control.

Many management control problems can be viewed in the same conceptual framework. For example, Figure 8 shows a diagram for information feedback, for the control of inventories and production levels. The parallels between the physical system and the inventory system are direct. From the principles of process control, we can learn some basic control concepts of considerable value in controlling inventories. These concepts are related to time lags and their effect on the stability of the system. Let us see what actual dynamic effects we might expect from an inventory system which was originally stable and now is stimulated by a 10 percent step increase in retail sales, the new sales level remaining stable.

Forrester [8], using a dynamic simulation model of the system shown in Figure 9 demonstrates the dynamic effects dramatically. There are three levels of inventories in the system: factory warehouses, distributor, and retailer. The circled lines show the flow of orders for goods from customers to retailers, retailers to distributors, distributors to factory warehouse, and finally from the warehouse as orders for the factory to produce. The solid lines show the flow of the physical goods between each of the levels of the structure in response to the orders. The circled numbers represent the time delays in weeks for each of the activities to take place. Figure 10 shows the effect of the 10 percent step increase in retail sales on inventories at the three levels, as well as on factory production output. Whereas the sales increase was simple and orderly, the response of the inventory and production system shows wild oscillations which increase in magnitude as we go up stream in the system from the retail level to the distributor, factory warehouse, and to the actual factory output. As we will demonstrate in Chapter 21, reducing the time lags in the system, for example, by eliminating the distributor level, or reducing the time for clerical delays will reduce considerably the magnitude of the fluctuations.

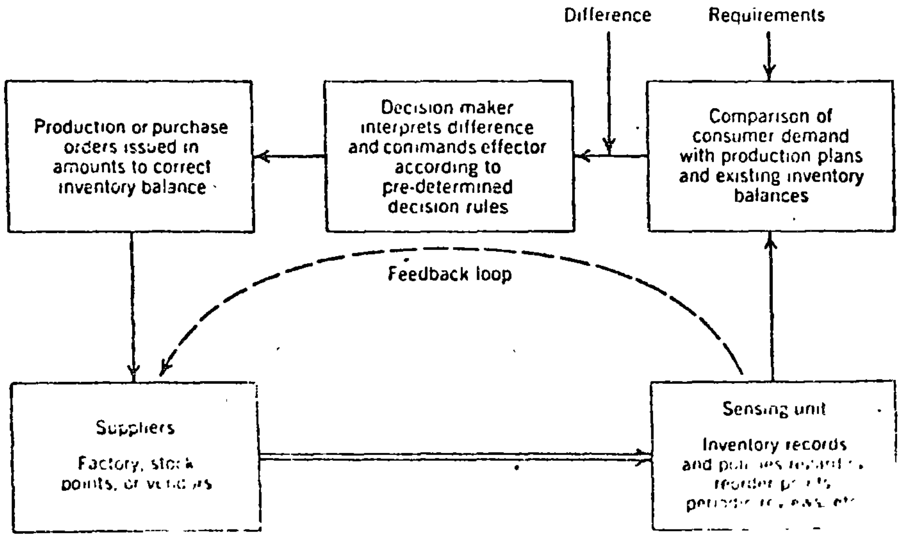

Difference
Requirements
Production or purchase orders issued in amounts to correct inventory balance
Decision maker interprets difference and commands effector according to pre-determined decision rules
Comparison of consumer demand with production plans and existing inventory balances
Feedback loop
Suppliers
Factory, stock points, or vendors
Sensing unit
Inventory records and policies regarding reorder points, periodic reviews, etc.

FIGURE 8 *Information feedback loop for an inventory control system*

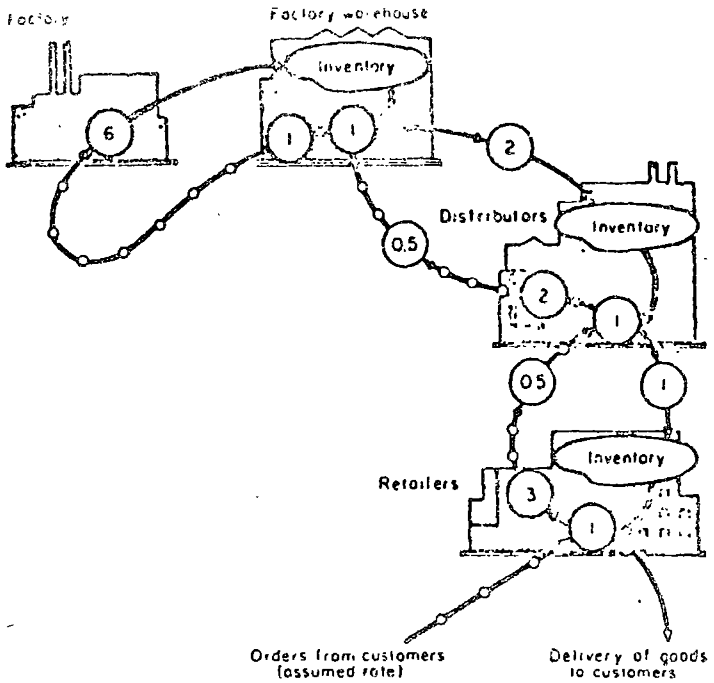
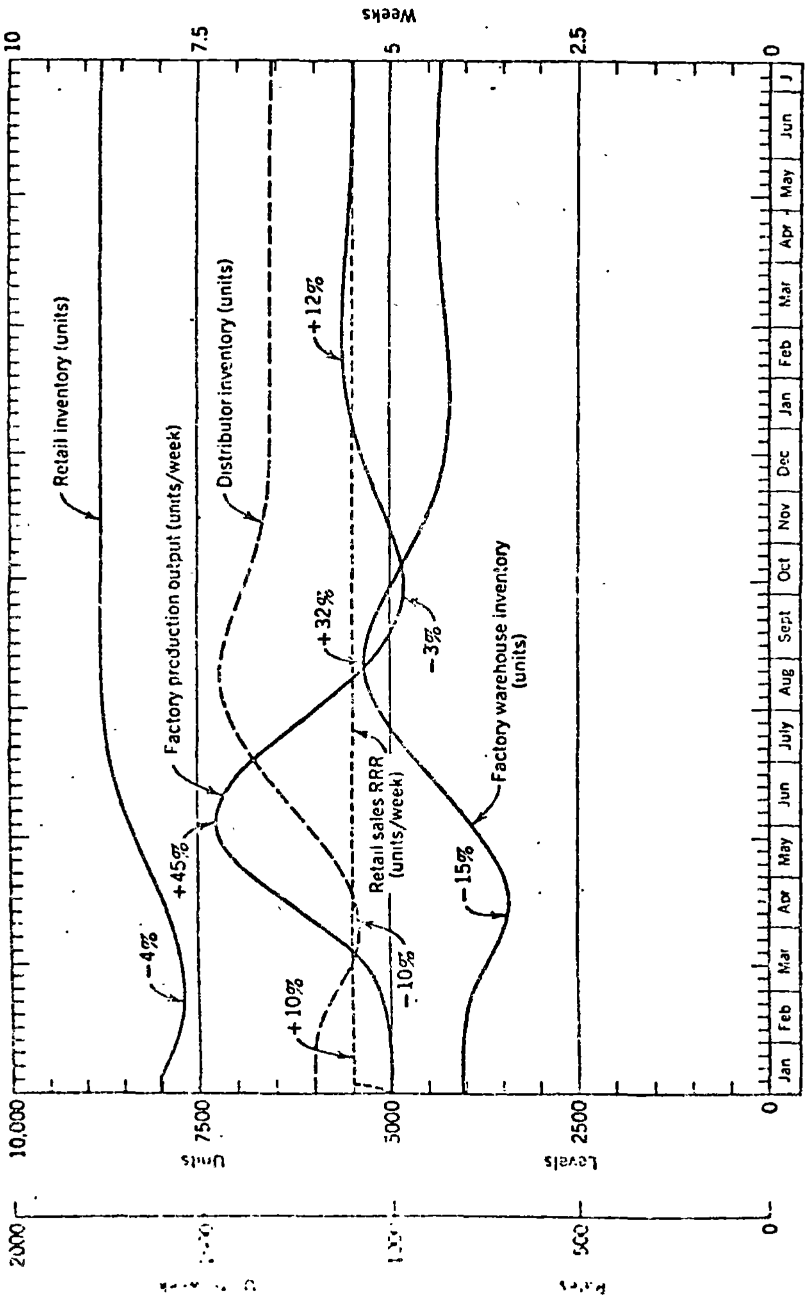


FIGURE 9. Structure of a production-distribution system Solid lines represent physical flow, lines with dots represent information flow, and circled numbers represent time delays in weeks. From J. Forrester, Industrial Dynamics, *[8].*



10 *Response of inventories at three levels and factory output to a step increase in retail sales of 10*
Adapted from J. Forrester, Industrial Dynamics, *[8].*

More direct information feedback to the various stock points instead of through the chain of demand shown in Figure 5 will have important effects in stabilizing the entire system. We consider these dynamic effects more carefully and in greater detail in our Chapter 21 discussion of large-scale system simulation. At this point, however, a conclusion we might draw is that a more direct information feedback system similar to that shown in Figure 11 for the capacitor can production-inventory system will have a stabilizing effect so that no amplification of demand *variability* will take place at stock points up stream from the consumer inventory level. At each stock point in the system, then, we are working against actual consumer demand rather than against the secondary and tertiary effects of demand as reflected back through the chain. Reducing the lag in information flow has a stabilizing effect, regardless of the inventory system used and would be appropriate for both the fixed quantity and fixed cycle systems.

Base Stock System. The base stock system [10] is a blend of the fixed quantity and fixed cycle systems which uses an information feedback system similar to that diagrammed in Figure 11. In this system, stock levels are reviewed on a periodic basis, but orders are placed only when inventories have fallen to a predetermined "reorder level." At this point an order is placed to replenish inventories to the "base stock" level, which is sufficient for buffer stock plus a fixed quantity calculated to cover current usage needs. Periodic reviews of current usage rates can result in upward or downward revisions in the base stock levels. The base stock system has the advantages of close control associated with the fixed cycle system which makes it possible to carry minimum buffer stocks. On the other hand, since replenishment orders are placed only when the reorder point has been reached, fewer orders, on the average, are placed so that order costs are comparable to those associated with the fixed quantity systems. Since all stock points are working against consumer demand, we do not have the amplification of demand variability at points up stream. Therefore, buffer stocks can be reduced even further, since the extreme levels of maximum demand are not experienced. Another result is a reduction in the cost of production fluctuations (hiring, separation, and training), since smaller production fluctuations are also associated with the type of information feedback system used.

In the sections just completed we tried to show the influence of variability of demand on inventory models, and the importance of time lags in the system as a whole. The important concept to carry over into this section is that inventory models must take account of the environment in which they are operated and cannot be considered as an isolated problem.

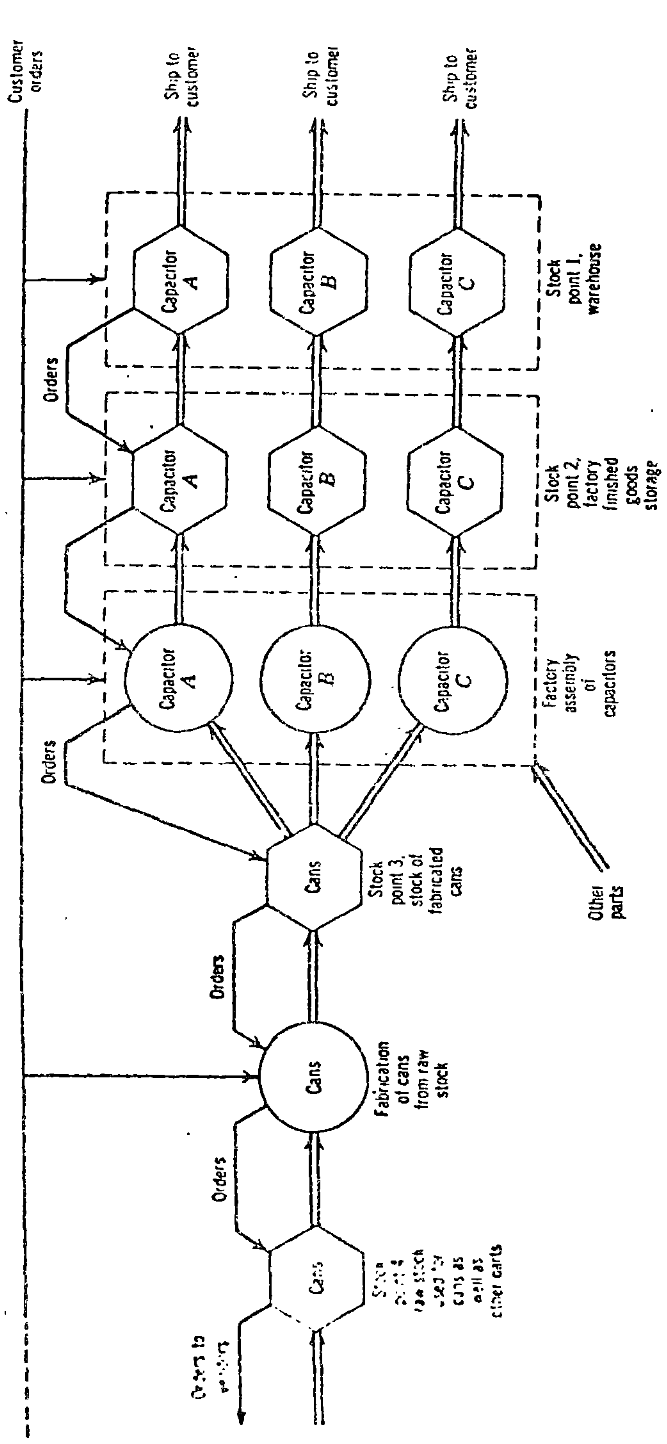


FIGURE 11. Current demand from customers fed back directly to stock points and operations so that all links in the production-inventory chain work against current demand.

We shall focus on the problem of operating a production-inventory system through the controlling of production levels. As we shall see, inventories play a major role.

Controlling Production Levels

When a basic production-inventory program has been developed, the result is a schedule of planned production levels and inventory balances based on forecasts of requirements. As sales proceed, however, we must have some system for compensating for the differences between planned and actual requirements in order to maintain inventories at proper levels. If actual requirements exceed plans, we run the risk of running out of stock, with resulting poor customer service and possible additional costs related to shortages. If actual requirements are below expectations, inventories will build up with resulting high carrying costs. Therefore, a control plan is needed which adjusts production and inventory levels in keeping with sales experience. Such a control plan might be accomplished by constructing periodically a new production program that takes into account existing inventories by adjustments in the short-run levels of production.

Our objective in this control plan is *to increase or decrease production levels in the period ahead, proportional to differences between actual and forecast sales, by an amount that minimizes the incremental costs of inventories and fluctuations of production levels.* If the planning period is fairly short, this adjustment of levels would continuously correct inventory levels to be in keeping with present demand, thus preventing stock-outs or the buildup of excessive inventories because of changes in demand. The basic elements of this control plan are comparable to those described earlier in this chapter and illustrated by Figure 8. We wish to construct a feedback control system where information on desired levels of inventories (indicated by current requirements) is compared with actual inventories to determine an error function which is fed back and compared with information on planned production levels for the coming period. By some predetermined rule, the production level is then adjusted to compensate for the demand fluctuation and bring inventories into line.

Decision Rules for Controlling Production Levels. Let us first state an obvious kind of rule for controlling production levels as actual requirements vary from forecasted requirements. The rule we will use for introductory purposes is that when actual requirements deviate from forecasts, we will add or subtract the difference as soon as possible to the amount produced in order to compensate for the variation from planned inventory levels. Let us illustrate with the forecast of requirements for 10 weeks shown in

TABLE II Calculation of production levels and inventories when the difference between forecasted and actual requirements is absorbed entirely by changes in production level, 2 weeks hence. Beginning inventory is 500 units.

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Week	Fore-casted Require-ments	Planned Pro-duction	Planned Inventories = Beginning Inventories + Col. 3 − Col. 2	Actual Require-ments	Difference Between Actual and Forecasted Requirements, Col. 5 − Col. 2	Actual Production Level = Planned Production + Difference Between Actual and Forecasted Requirements, 2 Weeks Ago, Col. 3 + Col. 6 for 2 Weeks Ago	Actual Inventory Level = Beginning Inventory + Actual Production − Actual Requirements = Beginning Inventory + Col. 7 − Col. 5
0			500				500
1	590	600	510	595	5	600	505
2	590	600	520	430	−160	600	675
3	590	600	530	590	0	605	690
4	590	600	540	1000	410	440	130
5	600	600	540	50	−550	600	680
6	600	600	540	625	25	1010	1065
7	600	600	540	570	−30	50	545
8	600	600	540	575	−25	625	595
9	610	600	530	680	70	570	485
10	610	600	530	705	95	575	355
	5980	6000		5820		5675	

Average production rate = 567.5 units per week.
Average inventory level = 573 units.

column 2 of Table II. Column 3 shows the planned production program for the product, and the planned inventories are easily calculated in column 4. As we would expect, however, actual requirements vary from forecast as shown in column 5 and the difference between actual and forecast requirements in column 6. The production lead time is 2 weeks, so that when a deviation from forecasted requirements occurs we can change production rate for the production period two weeks hence. Therefore, no change occurs from production plans in the first 2 weeks shown in column 7, but the third week's production reflects the shortage in planned requirements of five units. Similarly, the fourth week reflects the overage of 160 units which occurred in the second week, and so on. Actual inventory levels shown in column 8 are simply beginning inventory plus the amount produced during the week (column 7) less the actual requirements (column 5).

We see that this rule does indeed compensate for the variations, with a 2-week time lag, but at what cost? Actual production levels vary from 50 units to 1000 units per week in the short space of 10 weeks. But, notice that over the 10 weeks, actual total requirements were quite close to forecasted total requirements. Variation from forecast was largely week-to-week variation. As a matter of fact, the week-to-week variation reflects the random variations described in the demand distribution of Figure 4. In other words, it was variation that we should have expected to occur. Perhaps there is a better way to absorb this variation than by direct changes in the production level.

Let us test the idea just stated. Why not damp the effects of variation in actual requirements from forecast by changing production level by only 50 percent of the difference instead of 100 percent as we did previously. This is shown in Table III, under "50% Reaction." The original forecast of requirements and production and inventory plans are identical to those of Table II, but notice that violent swings in both production and inventory levels have been damped out considerably. Why not carry this idea farther? What happens with a 10 or 5 percent reaction rate? This is also shown in Table III with additional stabilizing factors in the form of simple heuristic rules. With the 10 percent reaction we have included the additional restriction that we will not respond to the variation from forecast at all unless 10 percent of the difference exceeds 10 units. In addition, with the 5 percent reaction we have included the 10-unit minimum and the restriction that larger changes in production level are made only in increments of 10 units. Therefore, if 5 percent of the difference is 27.5 units as it is in the fifth week, a change in production level of 30 units is made in the seventh week. Notice the results of progressively decreasing reaction rates in Table III. The results are more stable production and inventory levels. Also note, however, that average inventory levels have increased as reaction rate was decreased. The effect of reducing reaction rate could have been forecast. By using a

TABLE III Actual production and inventory levels when only 50 percent, 10 percent, or 5 percent of the difference between forecasted and actual requirements is absorbed by changes in production level from plan, 2 weeks hence. Buffer stocks absorb the balance of the variation. Data for forecasted and actual requirements and planned production and inventory levels are shown in Table II

	50% Reaction		10% Reaction 10-Unit Minimum		5% Reaction 10-Unit Minimum, Increments 10 Units	
Week	Actual Production Level	Actual Inventory Level	Actual Production Level	Actual Inventory Level	Actual Production Level	Actual Inventory Level
0	—	500	—	500	—	500
1	600	505	600	505	600	505
2	600	675	600	675	600	675
3	603	688	600	685	600	685
4	520	208	584	269	590	275
5	600	758	600	819	600	825
6	805	938	641	835	620	820
7	375	743	545	810	570	820
8	613	781	600	835	600	845
9	585	686	600	755	600	765
10	587	568	600	650	600	660
Average for 10 Weeks	589	655	597	684	598	688

relatively low reaction rate we are assuming that most deviations in actual requirements from forecasts are simply random deviations, so why become excited about them? If the deviation looks large, perhaps we should increase or decrease production rate *a little*, just in case it really marks the beginning of a trend. The question is, then, what should be the reaction rate for optimal cost performance? It is a good question, but it is slightly premature. Let us first discuss the general aspects of the decision rule and develop the ideas of reaction rates, review periods, and their interrelations.

Our decision rule really operates in the following context:

1. A longer-term forecast of requirements on which is based a broad production program
2. A shorter-term forecast or "review" to refine the forecasted requirements [illegible] period ahead
3. Based on this short-term review and forecast of requirements we can:
   a. Determine a production plan for these periods.
   b. Set planned inventory levels for these periods.
4. In the shortest-term planning period which is equal to the production lead time (the shortest notice used to change production levels in the period ahead), we can make a final adjustment in production level which takes account of the latest information we have regarding the comparison of actual and forecasted requirements.
5. The decision rule used is that production level in the immediate period ahead will be adjusted by some fraction k of the difference between actual and forecasted requirements for the current period.

In this context, we see that there are really two parameters we can manipulate to develop a model for the control of production levels. They are the value of k—the reaction rate—and the length of the review period mentioned in number 2 and 3 in the preceding outline. The importance of the reaction rate has already been discussed and demonstrated in the text material related to Tables II and III. In summary, k may take on values between the number 0 and 1.00, representing no reaction to deviations from forecasted requirements when $k = 0$, to 100 percent reaction and compensation when $k = 1.00$. In general terms, low values of k lead to stable production levels and relatively high buffer stock requirements, since variations from the plan must be absorbed by inventories. Conversely, high values of k lead to large production fluctuations and relatively low buffer stocks because variations from plan are absorbed by changing production levels. The significance of reaction rates in smoothing production rates is comparable to the smoothing constant α used in the exponentially smoothed forecasting methods discussed in Chapter 7.

The frequency of review also has a direct effect on both the magnitude of production fluctuations and the size of needed buffer stocks. The reason is easy to see in relation to the general principle of process control which we discussed in connection with Figures 7 and 8. The longer the period between reviews, P, the greater the chance that forecasts of requirements may not reflect the most current trends. Therefore, it is more likely that relatively large differences between actual and forecasted requirements would accumulate. For a given value of k, longer review periods lead to both relatively large production fluctuations and buffer stocks in order to provide the needed compensation. Short periods between reviews, then lead to closer control and relatively small production fluctuations and buffer stocks. [illegible] longer periods between reviews lead to looser control and larger production fluctuations and buffer stock requirements.

Determining k and P. Magee [10] derives two approximate formulas useful in solving the problem of determining the reaction rate k and the review period P for specific situations. He shows that the expected magnitude of production fluctuations is approximately proportional to

$$\sqrt{kP/(2-k)} \qquad (1)$$

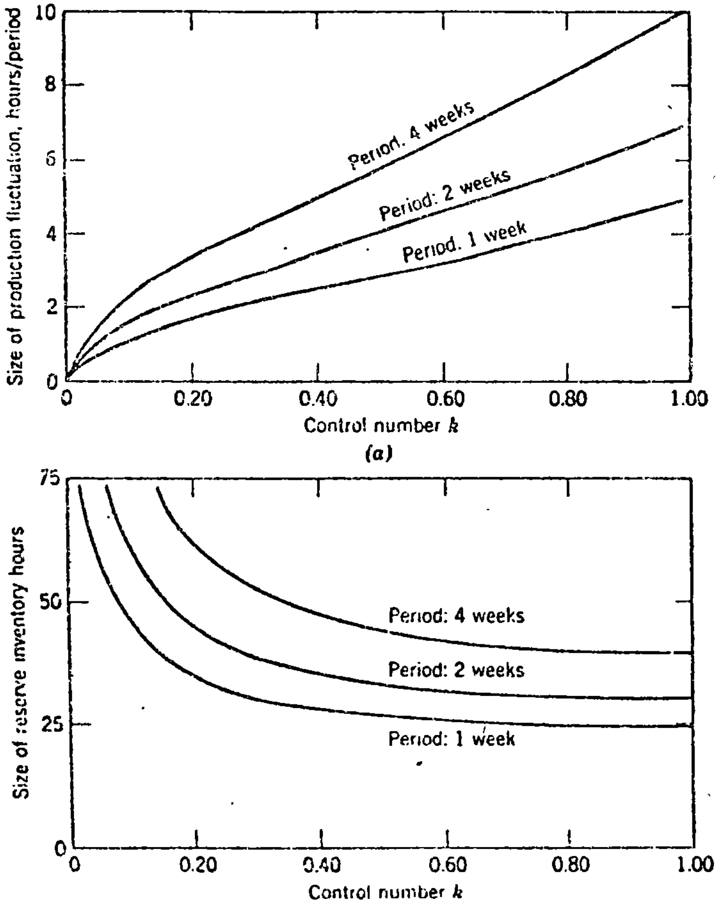


FIGURE 12. (a) Magnitude of production fluctuations versus control number and length of review period. (b) Reserve inventory required versus control number and length of review period. By permission from Production Planning and Inventory Control *by J. F. Magee and D. M. Boodman, McGraw-Hill Book Company, 2nd ed. New York, copyright 1967*

and that the required factory buffer stock will be approximately proportional to

$$\sqrt{[T(2k-k^2)+P]/(2k-k^2)} \qquad (2)$$

where T = production lead time, P = length of review period, and k = reaction rate in decimals.

The cost of production fluctuation, then, is proportional to (1) and the cost of buffer stocks are proportional to (2). Figure 12 shows the relationship of reaction rates and review period to the size of production fluctuations and reserve inventory requirements, expressed in equivalent hours. For a specific case, then, suppose that at k = 1.00 we experience a production fluctuation cost of $5000 and a buffer stock cost of $500, when the review period and production lead times are each 1 week. Using formulas (1) and (2), we can compute points for the curves shown in Figure 13 to find a value of k approximating 0.075 for minimum total incremental cost. Further similar calculations with different review periods would yield a combination of k and P which would minimize incremental costs for the entire system. Obviously, the right combination for a specific case like that shown in Figure 13 depends on the relative magnitudes of inventory carrying cost and the cost of production changes.

Let us summarize at this point some of the aspects of the control of inventories under uncertainty in a production-inventory system. Previously in this chapter we discussed systems for controlling inventories that

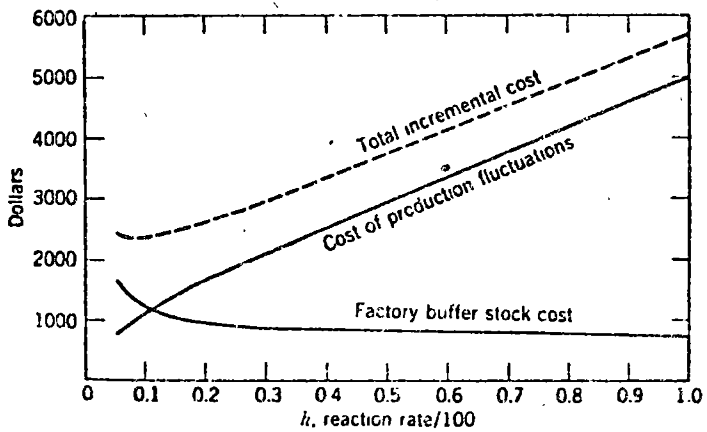


FIGURE 13 Relationship between incremental costs and k, *when the cost of production fluctuations and factory buffer stocks are $5000 and $500, respectively, at k = 1. Review period and lead time are 1 week*

involved fixing the quantity ordered at one time, letting the frequency of ordering vary, fixing the frequency of ordering, letting the quantity ordered vary, and the base stock system which was a combination of the elements of the two different systems. Also, differences in the information feedback pattern and their effects were noted. In the operation of a production-inventory system we have noted that the cost of production fluctuations is also an important factor to take into account. By way of summary, let us now consider the overall comparison of systems of control.

A Comparative Example

Magee [10] relates a hypothetical case called the Hibernian Co. which compares operation and costs for different basic systems of production and inventory control. The example considers a company that manufactures and sells about 5000 small machines per year for $100 each. The factory supplies four warehouses located in strategic areas around the country, which in turn supply the customer. We shall show the calculated results for four alternate systems of control: an economical order quantity system, a two-week fixed reorder cycle system, a base stock system with a review period of 1 week and reaction rate of 100 per cent, and a base stock system with a 1-week review period but involving a production reaction rate of 5 percent.

Each of the four branches sold an average of 25 units per week, or 1300 units per year. This average rate was, of course, subject to considerable variation, and Table IV shows distributions of demand at each of the four branches for 1-week periods, 2-week periods, etc. For example, at any given branch, sales would be expected to exceed 37 units per week only 1 percent of the time, 67 units per 2-week period 1 percent of the time, and so on. Requirements aggregated at the factory warehouse, reflecting demand from all four of the branches, are shown in Table V for eight different time groupings. Figure 14 shows the structure of the production-distribution system.

1. *Economical fixed reorder quantity system (EOQ).* Using an economical fixed reorder quantity system, we must analyze the requirements for buffer stocks, cycle stocks, transit stocks, and reordering costs for the branches, as well as, buffer stocks, cycle stocks, in-process inventory ordering costs, and the cost of production fluctuations at the factory and warehouse.

TABLE IV. Distribution of demand at each of four branches by eight different time-period groupings

Percent of Periods Exceeding Levels Given	Units of Sales Period, Weeks							
	1	2	3	4	5	6	7	8
90	19	41	64	87	111	134	158	182
60	24	46	71	95	124	144	168	193
50	25	50	75	100	125	150	175	200
20	29	56	82	108	134	160	186	212
10	31	[illegible]	86	113	139	166	[illegible]	[illegible]
1	37	67	95	123	151	[illegible]	[illegible]	[illegible]

TABLE V. Distribution of demand on factory warehouse from branches by eight different time-period groupings

Percent of Periods Exceeding Levels Given	Units of Requirements in Period, Weeks							
	1	2	3	4	5	6	7	8
90	87	182	278	374	471	569	666	764
60	95	193	291	389	488	587	686	785
50	100	200	300	400	500	600	700	800
20	108	212	314	417	519	621	722	824
10	113	218	322	426	529	631	734	836
1	123	233	341	447	553	658	762	866

Branches. At each branch, the economical quantity to be ordered at one time may be calculated if we know that $c_P = \$19$ ($6 clerical cost, $13 cost of packing, shipping, receiving, and stocking), $R = 1300$, and $c_H = \$5$. Q_0 is then,

$$\sqrt{(2 \times 19 \times 1300)/5} = 100 \text{ units}$$

Therefore, each branch would place an order for 100 units each, 4 weeks on the average, and the average *cycle stock* in each branch would be 100/2 = 50 units. The branch *buffer stock* is based on a 1 percent risk of running out of stock. Since the total

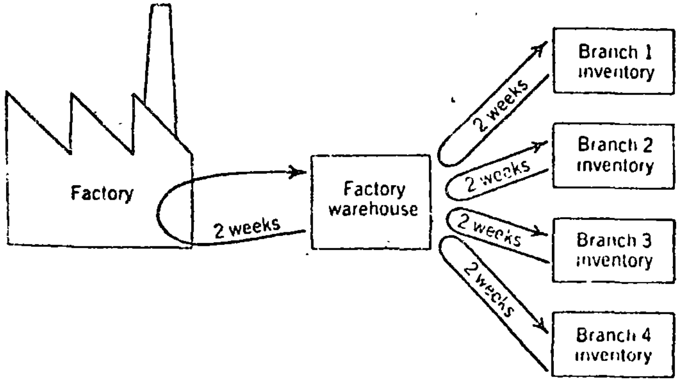


FIGURE 14 Structure of production-distribution system for Hibernian Co.

lead time was 2 weeks, we can determine the reasonable maximum demand during that period from Table IV as 67 units. Since normal demand during the 2-week lead time would be 50 units, the buffer stock is then the difference, or 17 units. Finally, the average *transit stock* is equal to the delivery time multiplied by the average demand rate, or 50 units. Average branch inventory is then as follows:

Buffer stock,	4 × 17 =	68 units
Cycle stock,	4 × 50 =	200
Transit stock,	4 × 50 =	200
		468 units

Since c_H = \$5 per unit per year, this average inventory of 468 units has an annual cost of \$2340. Since each branch places an order once every 4 weeks, on the average, there are 52 orders per year from the four branches which cost \$19 each or a total annual reordering cost of \$988.

Factory Warehouse and Factory. The factory warehouse is, of course, reflecting the aggregate demand from the four branches so that its economical order quantity reflects annual requirements, R = 5200 units, and its own inventory holding and preparation costs of c_H = \$3.50, and c_P = \$13.50. Calculating Q_0, as before, we obtain Q_0 = 200 units. Maximum 2-week demand from the branches (using a 1 percent run-out risk criterion) under the economical reorder quantity system is 233 units, so that *factory warehouse buffer stocks* are set at 233 − 200 = 33 units. *Cycle stocks are* 200/2 = 100 units, and *in-process inventories* in the factory average one-half the order quantity or 100 units. Total average inventory at the factory warehouse is therefore 233 units. On the average, 26 factory production orders per year must be issued at a cost of \$13.50 or \$351 per year. Table VI summarizes the inventory and ordering costs for the economical order quantity system. To this total we must add the *cost of production* fluctuations which occur with the economical order quantity system. Figure 15*a* shows a typical pattern of orders on the factory and the resulting factory production levels set. Note that very large fluctuations in production levels result and these fluctuations cost \$8500 per year.

TABLE VI. Summary of incremental costs of economical order quantity system for Hibernian Co. from Magee [10]

Inventory costs	
Four branches	*\$ 2,340*
Factory	*816*
Reorder costs	
Four branches	*988*
Factory	*351*
Production fluctuations	*8,500*
	\$12,995

2. *Fixed Reorder Cycle Systems.*

Branches. Under the fixed reorder cycle system, each branch warehouse maintains its inventory sufficient to fill reasonable maximum demands during the review period

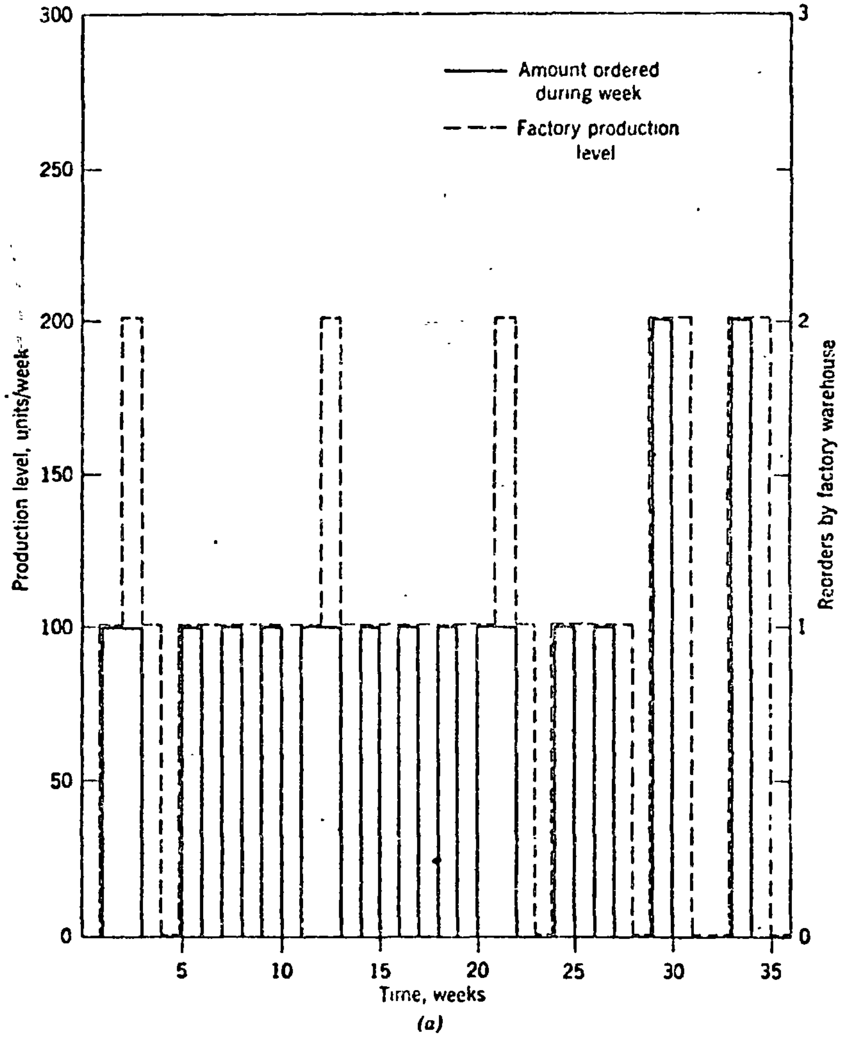


FIGURE 15. (a) *Factory orders and production level, economic reorder quantity system.* (b) *Production level, fixed reorder cycle system* (c) *Production level basestock system; reaction rate = 5 percent. Adapted by permission from* Production Planning and Inventory Control, *by J. F. Magee and D. M. Boodman, McGraw-Hill Book Company, 2nd ed., New York, copyright, 1967.*

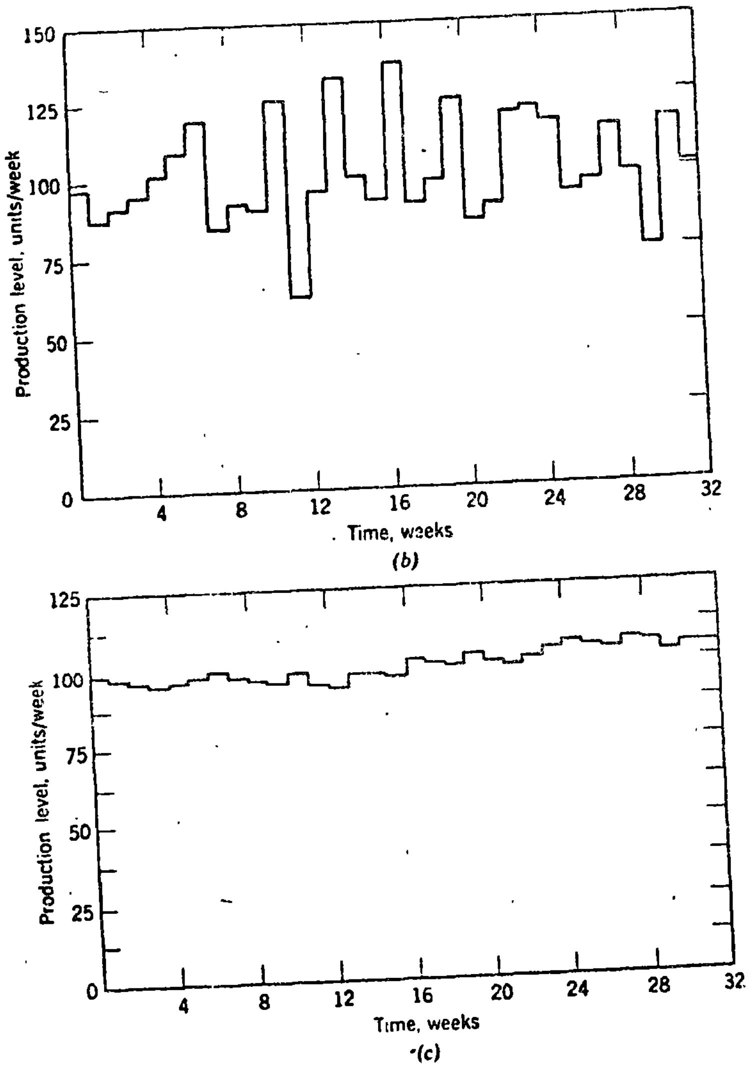


FIGURE 15. (Continued)

plus the 2-week delivery time. We must first compare system costs for several different review periods to determine the appropriate length of review period. Table IV shows the distribution of demand at each branch warehouse for eight different periods. Therefore, we can determine *buffer stock* requirements for each review period considered, at the 1 percent risk level by looking at the numbers on the row for [illegible] percent of period sales exceeding the 1 percent level. The buffer stock for [illegible] for a 1-week review period plus 2-week lead time is [illegible] 20 [illegible] requirements computed [illegible]

Table VII. *Cycle stock* would average one-half of the normal shipment. A shipment is made once each period so that the average amount shipped would be 25 × (the number of weeks in the period). Table VII shows the appropriate cycle stocks for each review period. *Transit stock* remains at 50 units for each review period. The number of orders placed varies inversely with the length of the review period, therefore, 1-week period results in 52 orders per year at $19 per order or $988 per year. The branch ordering costs for the other periods are summarized in Table VII.

*TABLE VII. Comparison of system costs for different lengths of review periods for the fixed reorder cycle system**

	Length of Review Period, Weeks					
	1	2	3	4	5	6
Branch warehouses, each branch						
Inventory						
Buffer stock	20	23	26	29	31	33
Cycle stock	12.5	25	37.5	50	62 5	75
Transit stock	50	50	50	50	50	50
Total	82.5	98	113 5	129	143 5	158
Annual inventory cost at $5	$ 412.5	$ 490	$ 567.5	$ 645	$ 717 5	$ 790
Ordering cost	990	495	330	250	195	165
Total	$1402.5	$ 985	$ 897.5	$ 895	$ 912 5	$ 955
Total, four branches	$5610	$3940	$3590	$3580	$3650	$ 3,820
Factory warehouse						
Buffer stock	41	47	53	58	62	67
Cycle stock	50	100	150	200	250	300
In-process stock	50	100	150	200	250	300
Total	141	247	353	458	562	667
Annual inventory cost at $3.50	$ 493	$ 865	$1235	$1630	$1967	$ 2,335
Ordering cost	700	350	235	175	140	120
Total factory warehouse cost	$1193	$1195	$1470	$1805	$2107	$ 2,455
Cost of changing production levels	$1600	$2250	$2760	$3180	$3560	$ 3,900
Total system costs	$8403	$7385	$7820	$8565	$9317	$10,175

*Modified from J. F. Magee and D. M. Boodman, Production Planning and Inventory Control, McGraw-Hill Book Company, New York, 2nd ed., 1967.

SUMMARY

In this chapter we have tried to develop the importance of the factor of demand variability and its impact on inventory planning. In doing this, we have developed the rational determination of buffer stocks and discussed systems inventory control which take account of the resulting risks. In connection with these systems of inventory control, the concepts of process control and information feedback were introduced and the important effects of time lags shown.

In considering the problems posed by inventories, we are forced to consider several levels of planning covering different time spans. These are as follows: .

1. *Long-range plans for plant capacity*. Plant capacity may be affected by seasonal peaks, and there are capital costs associated with this capacity. What combination of in-plant capacity, use of seasonal inventories, overtime, and subcontracting will minimize the combined capital costs, seasonal inventory costs, labor costs, production fluctuation costs, and extra costs of subcontracting? Is new capacity justified?
2. *Intermediate-range plans for a few months to a year* in advance, which attempt to determine for the expectations of sales what will be the best allocation of the resources of existing capacity. We are asking. what combination of production within periods, size of work force,, and seasonal inventories will minimize the combined costs of production fluctuation, seasonal inventory cost, labor costs, and extra subcontracting costs. We shall pay particular attention to this subject in Chapter 13.
3. *Short-range plans for the immediate period ahead.* Since actual requirements will change from forecasts, we must take a last look within the lead time to change production level, but neither can we change production levels capriciously because large costs can be involved, nor can we ignore what might develop into a huge inventory buildup. The result is that we need a control system that minimizes in the short range the cost of inventories and production fluctuations.
4. *In the shortest range of planning*, we need automatic decision rules that dispatch work to each and every workplace and machine There is no time to ponder the question at this point. We must develop an automatic rule which operates quickly and accurately indicating the best sequence in which to process orders at a [illegible] machine or machine center Here we are looking for [illegible] flow, such as those covered in Chapter 17, which will minimize inventory and idle labor costs while providing a high level of service to customers by completing their work on time.

Inventories have an important impact at all stages of planning and execution. The result is that we must view inventories in their multifunction role in the broad system from raw material input, flow through the production-distribution system, and to the consumer. They cannot be examined in isolation with realism.

REVIEW QUESTIONS

1. What are the three kinds of variations which we might expect in sales curves, which result in variability of demand?

2. Why is it that we wish to abstract just the random variations due solely to chance causes from the total variation in demand curves from all causes, for use in determining buffer stocks?

3. How can we determine what stock runout level to use for a specific situation?

4. Describe each of the three inventory control systems which take account of variability of demand which are described in this chapter.

5. What are the variables in inventory control systems that are subject to managerial control?

6. Which system has closer control over inventory levels, the fixed reorder quantity system or the fixed reorder cycle system?

7. What techniques may be applied to determine if an apparent change in demand is merely a random fluctuation or a true shift in average requirements?

8. Relate the general principles of process control to inventory control systems.

9. Describe the effects on retail inventories, distributor inventories, factory warehouse inventories, and on factory production levels when consumer demand changes, assuming the structure of a production distribution system as shown in Figure 9.

10. What is the nature of our objective in controlling production levels?

11. Compare the expected results when a production control rule is used with reaction rates of 100, 50, 10, and 5 percent.

12. In controlling production levels, what are the two main variables that are under our control?

13. What is the general relationship between reaction rates and the frequency of adjustment of production levels? Which combinations produce high costs of production fluctuation? High costs of reserve inventories?

14. How can equations (1) and (2) help to determine the best reaction rate to use [illegible]ation?

15. Make a complete analysis of the four systems of control used in the Hibernian Co. case, checking all calculations, to show exactly where the different systems have relative advantages and disadvantages.

PROBLEMS

1. Weekly demand for a product exclusive of seasonal and trend variations is represented by the empirical distribution given below. What buffer stock would be required for the item to insure that one would not run out of stock more than 15 percent of the time? Five percent of the time? One percent of the time? Normal lead time is one week.

Weekly Demand, Units	Frequency, Number of Weeks Demand Reached a Given Level
0	0
20	2
30	5
40	10
50	9
60	20
70	30
80	25
90	18
100	17
110	10
120	8
130	6
140	3
150	2
Total	165

2. If the item for which data is given in Problem 1 has a unit value of $100, shortages costs of $10 each and an annual inventory carrying cost of 25 percent of the average inventory value, which of the three levels of service would be most appropriate?

3. An organization is attempting to assess the cost of increasing its service level which is currently set at only 80 percent. Average demand during lead time is 18 units, and demand is reasonably well described by the Poisson distribution. Inventory holding costs are approximated by $c_H = \$10$ per unit per year. Calculate the buffer inventory costs required for service levels of 80, 90, 95, and 99 percent. What are the comparative costs if the distribution of demand during lead time is the negative exponential distribution? The normal distribution with $\sigma_D = 2, 4,$ and 6 units?

4. Given a control number of 0.6, a decreased demand fluctuation of 600 units in the first period, and a forecasted production level of 15,000 units in the third period, what would be the revised production quantity set for period three? (Owing to lead times, it is not possible to adjust the production level for the second period.)

5. A company manufactures a single product for which the following table represents a schedule of forecasted and actual demand in units for one year.

Month	Forecasted Demand	Actual Demand
Jan.	23,000	23,000
Feb.	24,000	25,000
Mar.	21,000	20,000
Apr.	23,000	22,000
May	20,000	22,000
June	19,000	24,000
July	17,000	22,000
Aug.	14,000	15,000
Sept.	8,000	6,000
Oct.	10,000	13,000
Nov.	9,000	10,000
Dec.	10,000	14,000
Total	198,000	216,000
Average	16,500	18,000

The initial inventory is 15,000 units. The desired ending inventory is 20,000 units. The cost of storage is $1 per unit per month. It costs $1000 to change production from zero to 3000 units and $3000 to change production from 3000 to 6000 units. No change larger than 6000 units is possible in one period. Back orders are permitted at a cost of $5 per unit per period.

(a) What is the best production plan for the forecasted demand if one wishes to minimize pertinent costs?

(b) Assuming that the year is over, what is the best production plan for the actual demand utilizing the benefit of hindsight?

(c) To correct for deviations in actual demand as compared to forecasted demand, evaluate the choice of a control number of 0.25 versus one of 0.75. Assume that at the end of a month sufficient time exists to alter the planned production for the next month.

(d) What would be the cost impact of these two control numbers if the following additional rules were formulated.

(1) Determine the planned production change.

(2) Add or subtract the additional change due to the forecast error modified by the appropriate control number factor.

DIRECTORIO DE ASISTENTES AL CURSO: APLICACIONES DE LAS COMPUTADORAS A LA SIMULACION Y OPTIMIZACION DE SISTEMAS DE INGENIERIA Y ADMINISTRACION DEL 10 DE MARZO AL 8 DE ABRIL DE 1978

	NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
1.	ING. GUILLERMO CAÑIZO LECHUGA Lamartine No. 404 Col. Polanco México 5, D.F. Tel. 545-20-41	INFONAVIT Bca. Del Muerto No. 280 México, D.F. Tel.
2.	ACT. RUBEN CHAVEZ MIZRAHI	CENTRO DE SERVICIOS DE COMPUTO U.N.A.M. Tel. 548-82-13
3.	FIS. BORIS DOBIN ROSENTHAL Cerro de Jurencia No. 79 Col. C. Churubusco México-21, D.F. Tel. 549-28-11	CENTRO DE SER. DE COMPUTO U.N.A.M. Tel. 548-82-13
. .	ING. HECTOR ENENES GAXIOLA Bca. del Muerto No. 280 México 20, D.F. Tel. 534-11-20 Ext. 150	INFONAVIT Bca. del Muerto No. 280 México 20, D.F. Tel. 534-11-20 Ext. 150
5.	SR. JAVIER ESPINOZA CACERES Londres No. 17 Depto. 203 Col. Coyoacán México, D.F. Tel.	INSTITUTO DE INGENIERIA UNAM. Cd. Universitaria México 20, D.F. Tel. 548-97-95
6.	SR. LEOPOLDO FERNANDEZ GARCIA Cerro del Vigía No. 14-6 Col. Campestre Churubusco México 21, D.F. Tel. 549-63-19	ESTUDIOS DE ING. PLANEACION Y PROGRAMAS, S.A. Av. Insurgentes Sur 559-401 Col. Nápoles México 13, D.F. Tel. 536-48-22
7.	SR. ING. JUSTINO C. GONZALEZ ALVAREZ	
8.	SR. XAVIER HARO SOLORZANO Blvd. Miguel Cervantes de Saavedra 647-9 Col Irrigación México 10, D.F. Tel.	S. A. R. H. Balderas No. 55-2°Piso Col. Centro México 1, D.F. Tel. 510-02-94

DIRECTORIO DE ASISTENTES AL CURSO: APLICACIONES DE LAS COMPUTADORAS A L SIMULACION Y OPTIMIZACION DE SISTEMAS DE INGENIERIA Y ADMINISTRACION DE 10 DE MARZO AL 8 DE ABRIL DE 1978

NOMBRE Y DIRECCION — EMPRESA Y DIRECCION

9. SRTA. MA. ANGELICA HERNANDEZ SUAREZ
Amores No. 1707-401
Col. Del Valle
México 12, D.F.
Tel. 524-19-07

S. A. H. O. P.
Insurgentes
México 12, D.F.
Tel. 567-52-21

10. ING. LUIS MEDINA CRAVIOTO
Sta. Gertrudis No. 22-1
Col. Industrial
México 14, D.F.
Tel.

I. P. N. E. S. I. A.
Edif. 4 de la Unidad
Profesional Zacatenco
Col. Lindavista
México 4, D.F.
Tel. 586-96-44

11. ACT. FCO. DAVID MEJIA RODRIGUEZ
Vicente Suárez No. 132-201
Col. Condesa
México 11, D.F.
Tel. 553-73-84

CENTRO DE SER. DE COMPUTO
U. N. A. M.
México 20, D.F.
Tel. 548-82-13

12. SR. CESAR NIEMBRO AVILA
Miguel Laurent No. 840-2° Piso
México 13, D.F.
Tel. 559-95-01

S. A. H. O. P.
Miguel Laurent No. 840
México 13, D.F.
Tel. 539-95-01

13. ING. SERGIO PEREZ USCANGA
Paseo de Italia No. 101
Col. Lomas Verdes
Naucalpan, Edo. de Méx.
Tel.

BUFETE INDUSTRIAL
Tolstoi No. 22
Col. Anzures
México 5, D.F.
Tel. 533-15-00

14. ING. DANIEL REYES BUGARIN
González de Cosio No. 24
Col. Del Valle
México 12, D.F.
Tel. 523-27-61

ICATEC, S.A.
González de Cosio No. 24
Col. Del Valle
México 12, D.F.
Tel. 536-85-60

15. SR. VICTOR MANUEL RIOS NORIEGA
Viaducto Piedad No. 580
Col. Magdalena Michuca
México 8, D.F.
Tel. 768-72-90

COMISION DE AGUAS DEL
VALLE DE MEXICO
Balderas No. 55-2o. Piso
Col. Centro
México 1, D.F.
Tel. 510-02-94

DIRECTORIO DE ASISTENTES AL CURSO: APLICACIONES DE LAS COMPUTADORAS A LA SIMULACION Y OPTIMIZACION DE SISTEMAS DE INGENIERIA Y ADMINISTRACION DEL 10 DE MARZO AL 8 DE ABRIL DE 1978

NOMBRE Y DIRECCION

EMPRESA Y DIRECCION

16. LIC. JOSE PAULINO SANCHEZ PEREZ
Instituto de Higiene No. 13
Col. Popotla
México 17, D.F.
Tel.

COMPAÑIA MEXICANA DE
EXPLORACIONES
Río Balsas No. 101-4o. Piso
Col. Cuauhtémoc
México 5, D.F.
Tel. 528-91-38

17. ING. FRANCISCO SANDOVAL PADILLA
Santander No. 76
Col. San Rafael Azcapotzalco
México 16, D.F.
Tel. 561-68-84

COMISION DE AGUAS DEL
VALLE DE MEXICO
Balderas No. 55-2o. Piso
Col. Centro
México 1, D.F.
Tel. 510-02-94

18. SR. MARCO ANTONIO TAPIA LIZARRAGA
Coyoacán No. 312-7
Col. Del Valle
México 12, D.F.
Tel. 523-49-73

DIV. EST. SUP. FAC. ING.
U.N.A.M.
Cd. Universitaria
México 20, D.F.
Tel.

19.- ING. VICENTE VAZQUEZ ZUÑIGA
Abasolo 505-5
Pachuca, Hgo.

INST. TEC. REGIONAL DE
PACHUCA No. 20
Tel.