RELACION DE ALUMNOS DEL CURSO SOBRE ASIGNACION OPTIMA DE RECURSOS

NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
1. SR. EDUARDO ALVAREZ TORRES	CARNATION DE MEXICO, S. A. Av. Insurgentes Sur No. 363 México, D. F.
2. ING. MANUEL BORREGO CALDERA Camino Viejo a Chimalhuacan s/n Junto a Granja Guadalupana Chicoloapan, Edo. de México	CIA. DE LUZ Y FUERZA DEL CENTRO, S. A. Melchor Ocampo No. 171-212 México, D. F.
3. ING. JUAN JOSE BOTELLO BARRON Isla San José No. 16 México 14, D. F.	CIA. DE LUZ Y FUERZA DEL CENTRO, S. A. Melchor Ocampo No. 171 México, D. F.
4. SR. JOSE ANTONIO CASILLAS MEJIA Calle 1857 No. 25 Col. El Parque Balbuena México, D. F.	SECRETARIA DE COMUNICACIONES Y TRANSP.
5. ING. VICTOR CHALE CURZ Playa Nornos No. 297 Col. Reforma Ixtaccihuatl México 13, D. F.	BUFETE INDUSTRIAL CONSTRUCCIONES,S.A. Tolstoi No. 22 Col. Anzures México, D. F.
6. ING. FERNANDO COLCHERO LANDA	
7. LIC. JOSE DE J. ARTURO DE ALBA M. Av. Progreso No. 186-103 Col. Escandón México 18, D. F.	FINANCIERA DEL ATLANTICO Venustiano Carranza No. 48 3er. piso México 1, D. F.
8. C. P. SERGIO M. DE URIARTE	BANCO DE LONDRES Y MEXICO Venustiano Carranza No. 65 México 1, D. F.
9. SR. LUIS FONSECA ARAUJO Fray Servando Teresa de Mier 882-C-6 Col. Jardín Balbuena México, D. F.	TENERIA TEMOLA, S. A. Calle Boleo No. 57 Col. Maza México 2, D. F.
10. SR. ARTURO FONSECA ROMERO Presa Sacinillas No. 296-3 México, D. F.	CENTRO NUCLEAR DE MEXICO Salazar, Edo. de México

RELACION DE ALUMNOS DEL CURSO SOBRE ASIGNACION OPTIMA DE RECURSOS

	NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
11.	C. P. PABLO GARCIA	CONSEJO NACIONAL DE CIENCIA Y TECNOLOGIA Av. Insurgentes Sur No. 1677 México, D. F.
12.	SR. ARMANDO GINER MARQUEZ Sinaloa No. 100 Col. Peñón de los Baños México, D. F.	CONSEJO NACIONAL DE CIENCIA Y TECNOLOGIA Insurgentes Sur No. 1677 México, D. F.
13.	ING. J. FERNANDO GOICOECHEA M. Miguel Ramos Arizpe No. 28-3 México, D. F.	INSTITUTO MEXICANO DEL CAFE Insurgentes Sur No. 421-B 1er. piso México 11, D. F.
14.	ARQ. JAIME GONZALEZ SALAZAR Edif. F34-3-12 Lomas de Plateros Mixcoac México, D. F.	COMITE ADMINISTRADOR DEL PROGRAMA FEDERAL DE CONSTRUCCION DE ESCUELAS Fresnos No. 380 Col. Florida México 20, D. F.
15.	ING. GUILLERMO GONZALEZ ESCAMILLA Valentín Gama No. 420 San Luis, Potosí, S. L. P.	SECRETARIA DE COMUNICACIONES Y TRANS. Díaz de León No. 210 San Luis Potosí, S. L. P.
16.	ING. ALEJANDRO JARAMILLO SILVA Playa Regatos No. 415 Col. Marte México, D. F.	LABORATORIOS ELILILLY DE MEXICO, S. A. Calzada de Tlalpan No. 2024 México, D. F.
17.	ING. GIL EDUARDO JIMENEZ AGUILAR Ayuntamiento No. 93-108 México 1, D. F.	SECRETARIA DE RECURSOS HIDRAULICOS
18.	ING. FRANCISCO LEAL ALCANTARA Calle 1857 No. 28 Col. El Parque Balbuena México, D. F.	TELE INDUSTRIA DE MEXICO, S. A.
19.	SR. FRANCISCO J. LEYVA GARCIA Unión No. 213 Col. Industrial México 14, D. F.	SECRETARIA DE RECURSOS HIDRAULICOS Plaza de la República No. 31 6o. y 7o. p. México 1, D. F.
20.	SRITA. MARGARITA I. LOPEZ C. Sur 124 No. 2760 Col. Villa de Cortés México 13, D. F.	SECRETARIA DE RECURSOS HIDRAULICOS Plaza de la República No. 31 México 1, D. F.

RELACION DE ALUMNOS DEL CURSO SOBRE ASIGNACION OPTIMA DE RECURSOS

	NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
21.	ING. JOSE ANTONIO LOZANO VILLAFAÑA Heriberto Frías No. 245-3 Col. del Valle México 12, D. F.	CONDUMEX, S. A. Poniente 140 No. 720 Col. Industrial Vallejo México, D. F.
22.	ING. GUILLERMO MACIAS GARCIA V. Hortensias No. 9 México 21, D. F.	COMISION NAL. COORDINADORA DE PUERTOS Av. Juárez No. 92 7o. piso México 1, D. F.
23.	ING. CARLOS A. MAIGLER	ELECTRONICA BALTEAU, S. A. Calle Escape No. 21 Naucalpan de Juárez, Edo. de México
24.	ING. SERGIO MARISCAL BELLA Luis Cabrera 58-A Circuito Economistas Cd. Satélite, Edo. de México	PETROLEOS MEXICANOS Av. Marina Nacional No. 329 México 17, D. F.
25.	ING. GONZALO MARTINEZ CORBALA Ayuntamiento No. 21 Col. Coyoacán México 21, D. F.	ORGANIZACION CONSTRUCTORA MEXICANA,S.A. Insurgentes Sur No. 1766 1er. piso México 20, D. F.
26.	ING. FERNANDO MONROY URBINA Av. Insurgentes No. 4411 Edif. 22 Depto. 403 Conjunto Residenciar Insurgentes Sur México, D. F.	COMITE NAL. ADMOR. DE PROGRAMA FEDERAL DE CONSTRUCCION DE ESCUELAS Fresnos No. 380 Col. Florida México, D. F.
27.	ING. RAUL ORTEGA SANSORES Pachuca No. 157-404 Col. Condesa México 11, D. F.	OLIVETTI MEXICANA, S. A. Calz. Vallejo No. 1029 Col. Nueva Vallejo México, D. F.
28.	ING. JOSE MANUEL ORTIZ VEGA Uxmal No. 621 Col. Narvarte México 13, D. F.	BUFETE INDUSTRIAL DISEÑOS Y PROYECTOS Tolstoi No. 22 Col. Anzures México 5, D. F.
29.	ING. GUILLERMO PEREZ NUÑEZ Bartolache No. 1804-201 México, D. F.	BUFETE INDUSTRIAL DISEÑOS Y PROYECTOS Tolstoi No. 22 México, D. F.
30.	ING. LUIS ROBERTO REYNOSO B. Calzada Obrero Mundial No. 123-5 México. D. F.	CIA. INDUSTRIAL DE PLASTICOS, S. A. Lago Xochimilco No. 121 México 21, D. F.

RELACION DE ALUMNOS DEL CURSO SOBRE ASIGNACION OPTIMA DE RECURSOS

	NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
31.	ING. CARLOS ROBLEDO PONCE Sur 73 No. 320 Col. Justo Sierra México 13, D. F.	DESPACHO ROBERTO CASAS ALATRISTE Durango No. 81 5o. piso Col.Roma México 7, D. F.
32.	SRITA. EUGENIA SANCHEZ DE GINER Sinaloa No. 100 Col. Peñón de los Baños México, D. F.	INSTITUTO POLITECNICO NACIONAL Plomeros y Peluqueros Col. Morelos México, D. F.
33.	ING. MARCO SEGOVIA MEJIA Unión No. 196-512 Col. Escandón México 18, D. F.	BUFETE INDUSTRIAL CONSTRUCCIONES, S.A Dante No. 36 5o. piso México 5, D. F.
34.	ING. KENNETH SYDNEY SMITH JACOBO Av. Melchor Ocampo No. 171 México, D. F.	CIA. DE LUZ Y FUERZA DEL CENTRO, S.A. Calzada de Explanada No. 25 México 10, D. F.
35.	ING. RAFAEL SUAREZ ALCALA Monte Albán No. 413 México, D. F.	BUFETE INDUSTRIAL CONSTRUCCIONES, S.A Tolstoi No. 22 México, D. F.
36.	ING. FRANCISCO TAVERA ESCOBAR Gutenberg No. 61-10 México 17, D. F.	PETROLEOS MEXICANOS Av. Marina Nacional No. 329 México, D. F.
37.	ING. CESAR VARGAS SANTILLAN	COMISION NAL. COORDINADORA DE PUERTOS Av. Juárez No. 92 7o. piso México 1, D. F.
38.	SR. CARLOS ANTONIO VELAZQUEZ M. Managua No. 14 Las Americas Naucalpan, Edo. de México	BANCO DE COMERCIO, S. A. Bolivar No. 34 3er. piso México 1, D. F.
39.	ING. OCTAVIO R. ZENTELLA MAYER Palenque No. 399-302 México 13, D. F.	BANCO DE COMERCIO, S. A. Bolivar No. 34 4o. piso México 1, D. F.

I. LA TEORIA DE OPTIMIZACION Y SUS APLICACIONES

por

F. Ochoa *

1. INTRODUCCION

El objetivo de este artículo es formalizar los conceptos que permiten describir la naturaleza y campo de aplicación de la *teoría de optimización*. Se define primero el problema de optimización y se discuten los aspectos complejos del mismo. Se identifican luego los pasos fundamentales del proceso de solución de un problema de optimización: a) Definición del problema. b) Formulación de un modelo de optimización. c) Selección de un método de solución. d) Aplicación del método de solución. Cada paso del proceso y lo que cada uno implica se discute en detalle.

Finalmente se presenta una clasificación tentativa tanto de los modelos de optimización como de las técnicas de solución.

* Director General, IPESA, Ingenieros Consultores.

2. EL PROBLEMA DE OPTIMIZACION

Cuando un ejecutivo o persona responsable de tomar decisiones se ve confrontado con el problema de seleccionar un curso de acción entre un conjunto de alternativas, se verá compelido a escoger la mejor, en términos de un cierto objetivo u objetivos predeterminados, según la naturaleza del problema.

Se asume en lo anterior, que el grado en el cual cada alternativa se acerca al objetivo puede evaluarse mediante un método cuantitativo. En otras palabras, una medida de la utilidad de cada curso de acción puede determinarse, permitiendo asi al que toma decisiones, seleccionar la alternativa que registre la máxima utilidad. El grado de acercamiento al objetivo en cada caso particular es la "figura de mérito".

DEFINICION - Un problema de optimización se define como la selección, entre un conjunto de varias alternativas (posiblemente un número infinito) de un problema, de aquella para la cual la figura de mérito se optimiza (se maximixa o minimiza).

3. TEORIA DE OPTIMIZACION. NATURALEZA DEL PROBLEMA DE OPTIMIZACION.

La naturaleza de los problemas de optimización es en general bastante compleja y una gran variedad de casos, presentando diversas características, se encuentran a menudo en problemas prácticos.

Para visualizar la complejidad inherente a la naturaleza del problema, considérense los siguientes ejemplos:

a) El que toma decisiones puede verse confrontado con un problema que tenga un objetivo claramente definido que debe optimizarse, sin embargo el problema puede estar sujeto o no a una serie de restricciones, las cuales a su vez, pueden no estar claramente definidas. También tendrá que tener en cuenta para la solución del problema, si el comportamiento del mismo es determinístico o estocástico.

b) El que toma decisiones podrá verse en el caso de tener que interactuar y competir con otros participantes, cada uno de los cuales trata de tomar decisiones que optimicen su propia figura de mérito.

c) Varias decisiones tendrán que tomarse en un problema de etapas múltiples, donde el objetivo es una optimización a largo plazo en contraposición con una suboptimización de una etapa particular del problema.

Es esta naturaleza tan diversa, asi como las características estructurales tan distintas de los modelos (Ver sección 6), que indican claramente la necesidad de una gran variedad de técnicas para atacar la solución de problemas de optimización. El conjunto de todas estas técnicas, o sea las incluidas dentro de los nombres específicos de *programación matemática, teoría de juegos, teoría estadística de la decisión, programación dinámica, teoría del control, cálculo de variaciones, etc.*, constituyen, junto con sus bases teóricas, la teoría general de optimización.

La teoría de optimización, en su acepción más amplia, es la rama unificada del *análisis matemático* que proporciona un enfoque formal para la solución de los problemas de optimización.

4. PROCESO DE SOLUCION

El proceso de solución para problemas de optimización puede no ser idéntico en todos los casos y puede diferir debido a la naturaleza especial del problema; sin embargo siempre será posible distinguir en el proceso los pasos básicos indicados en la Figura I.1. Las líneas de retroalimentación indican la posible revisión de las decisiones anteriores.

5. DEFINICION DEL PROBLEMA

En la etapa de definición del problema se identifican las variables de decisión o control que lo gobiernan, y se especifica a su vez la forma de interacción entre ellas. Deberá definirse una figura de mérito en términos de -- las variables de control relevantes, y el rango de los controles debe especificarse explícita o implícitamente. Finalmente, las restricciones que deben satisfacer las variables habrá que establecerlas.

6. FORMULACION DE UN MODELO MATEMATICO

Una vez que el problema ha quedado adecuadamente definido, el paso subse- - cuente es formular un modelo abstracto (usualmente un modelo matemático), - que represente fielmente la estructura esencial del problema y que pueda tener solución mediante la aplicación de un procedimiento conocido.

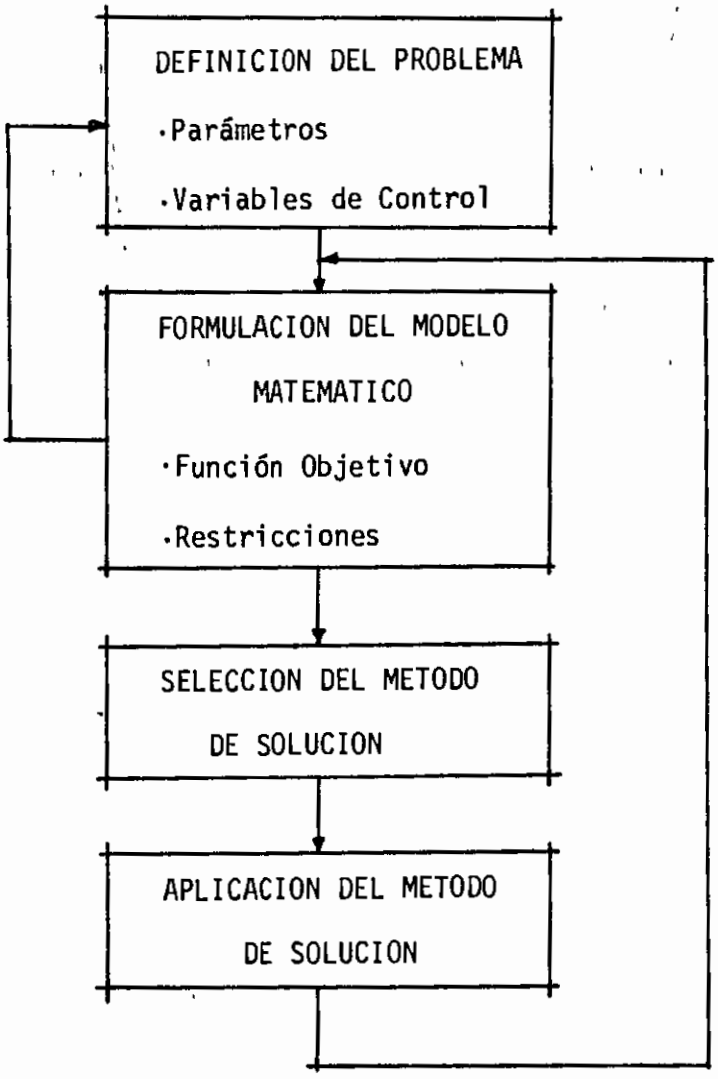


Fig. I.1 Proceso de Solución del Problema de Optimización

Cuando se haga referencia a modelos, lo entendemos en el sentido de Karlin*; "un modelo es una adecuada abstracción de la realidad que preserva la estructura esencial del problema de tal manera que su análisis proporcione información acerca de la situación concreta asi como de otras situaciones que tengan la misma estructura formal".

Es claro que la solución del modelo producirá resultados confiables solo en la medida que el modelo sea representativo del problema original. Si el problema no ha sido modelado apropiadamente, su solución puede llevar a resultados dudosos o completamente erróneos; por ejemplo considérese el caso de un modelo de programación lineal que da una solución no acotada como resultado de no haber incluido una de las restricciones dentro del modelo.

Se analizan a continuación algunas características distintivas de los modelos de optimización que permitirán su clasificación. Esto sera útil para identificación posterior de los modelos que se tratarán en las demás sesiones del seminario.

Se distinguen tres componentes principales de un modelo de optimización:

a. El conjunto de variables del problema.

b. La figura de mérito por optimizar.

c. El dominio de definición de las variables del problema (determinado por las restricciones del problema).

* Karlin, S., "Mathematical Methods and Theory in Games, Programming, and Economics", Vol. I, Addison-Wesley, 1959.

La solución óptima para cierta clase de problemas de optimización consiste - en valores numéricos que toman las variables del problema, satisfaciendo las restricciones y optimizando simultáneamente la figura de mérito.

Otras clases de problemas de optimización buscan una curva o función (problemas variacionales), que satisfaga a un conjunto de restricciones y haga óptima una cierta expresión funcional del conjunto de curvas factibles.

Para ciertos problemas el objetivo será susceptible de representación matemática como una función de las variables de control. Para otros problemas esta representación explícita puede no ser obtenible, y la figura de mérito para un conjunto dado de valores de las variables de control podrá conocerse solo después de completar un proceso complejo (como un proceso de simulación, un análisis ingenieril, la solución de un programa de computación muy elaborado, o una búsqueda en una tabla).

Más aun, el problema puede ser restringido o no restringido. Para problemas restringidos susceptibles de modelarse en forma matemática explícita, la naturaleza de las expresiones que representan a las restricciones puede ser - muy diversa. Por ejemplo pueden ser expresiones algebraicas o trascendentes, igualdades o desigualdades, lineales o no lineales y siendo el dominio de -- las variables un conjunto contínuo o discreto. También, algunas de las restricciones pueden ser ecuaciones diferenciales o integrales.

A la luz de la discusión anterior, se ha elaborado la clasificación de los - modelos de optimización en forma de estructura arbolada, mostrada en las - - Figuras I.2a y b. El árbol obviamente puede ser expandido tanto en la direc-

ción horizontal como en la vertical hasta hacerlo tan completo como se requiera o se desee.

Se distinguen ciertas ramas del árbol que representan a clases específicas de problemas, para las cuales los procedimientos de solución constituyen un desarrollo matemático bien establecido. Por ejemplo, los modelos en la rama de optimización restringida para los cuales tanto las restricciones como el objetivo puede representarse en forma matemática cerrada, constituyen aquella parte de la teoría de optimización generalmente conocida como *programación matemática.*

Como segundo ejemplo, considérese la clase de problemas para los cuales la función objetivo en forma explícita, se expresa mediante una integral definida (objetivo funcional) con o sin condiciones subsidiarias. La solución de tales modelos cae dentro del campo del *cálculo clásico de variaciones.*

Finalmente, considérense aquellos modelos con restricciones y/o objetivo que no aceptan representación matemática explícita. La optimización de tales modelos puede intentarse por medio de técnicas que se designan con el rubro general de *métodos de búsqueda directa.*

Un ejemplo de esta clase sería un cierto proceso estocástico (por ejemplo -- una línea de espera, un proceso de renovación) que está siendo analizado mediante una simulación en computadora. Los parámetros de entrada pueden variar y la simulación ejecutarse para cada conjunto de valores. Mediante -- los resultados de salida de cada corrida puede estimarse una *medida de efectividad* (MDE) de los parámetros de entrada correspondientes. Si el problema

consiste en seleccionar los parámetros de entrada que optimicen la MDE, se - requerirá una técnica de búsqueda directa para obtener el óptimo al mismo -- tiempo que se minimiza el número de experimentos simulados.

7. TECNICAS DE SOLUCION

Las técnicas de solución son los procedimientos y algoritmos desarrollados para la solución de problemas de optimización. La solución de un problema implica usualmente la determinación de los valores numéricos de las variables de control y el valor óptimo de la figura de mérito.

Los métodos de optimización se dividen generalmente en dos grandes categorías: *métodos indirectos y directos*. Con los métodos directos, la solución óptima se busca calculando directamente los valores de la función objetivo en diferentes puntos de la región factible. Los valores asi obtenidos se comparan y, por medio de un criterio auxiliar, se analiza ahora un nuevo punto que, es de esperarse, mejorará el valor de la función objetivo.

Alternativamente, los métodos indirectos buscan un conjunto de valores de las variables de control que satisfagan condiciones necesarias de optimalidad previamente conocidas. El método clásico del cálculo diferencial es un ejemplo del tipo indirecto. En efecto, se buscan valores de las variables para los cuales las primeras derivadas de la función objetivo se anulen, suponiendo que se garantice la continuidad de la función y la existencia de las derivadas en la región de interés. En esta forma, el problema de optimización ha sido transformado en un problema de búsqueda de raíces.

El algoritmo Simplex de la programación lineal exhibe aspectos tanto del método directo como del indirecto. Lleva a cabo una búsqueda directa solo entre los puntos extremos de la región factible (puntos que satisfacen la condición necesaria de optimalidad) en forma tal que la figura de mérito sea --

cuando menos tan buena como en el paso previo. Finalmente, el óptimo se detecta, de entre el conjunto de puntos extremos, cuando el criterio indirecto de factibilidad de la solución complementaria del problema dual asociado se ha satisfecho.

Para ciertos modelos matemáticos de optimización, el método de solución puede incluir la transformación del modelo original en uno equivalente que prometa ser de más fácil tratamiento que el original. Considérese la metodología de la *programación geométrica* *: en este caso, la optimización polinomial se formula en términos de su problema dual y éste es el modelo que realmente se resuelve. Otro ejemplo es la transformación en un problema de programación lineal de un programa no lineal *separable*.

Las técnicas directas pueden subdividirse en dos grupos principales: métodos *simultáneos y secuenciales*. Las técnicas de búsqueda simultánea calculan los valores de la función objetivo o superficie de respuesta para un conjunto de puntos determinados *a priori* mediante una cierta estrategia de búsqueda. Los métodos de búsqueda secuencial, por otra parte, se refieren a el examen secuencial de los soluciones de cada intento experimental, basando la localización de intentos subsecuentes en los resultados de los anteriores. En las Figuras I.3a y b se presenta un subconjunto de técnicas de solución representativas de cada una de las clases de métodos discutidos en esta sección.

* Duffin, R.J., E.L. Peterson, y C. M. Zener, *Geometric Programming*, John Wiley, 1967.

8. SELECCION DEL METODO DE SOLUCION.

La selección de un método de solución conveniente para un problema dado depende del tipo de modelo empleado, las técnicas de solución existentes para ese modelo particular y las facilidades de computación disponibles para el ingeniero analista.

En el proceso de selección se pueden considerar factores tales como linealidad del modelo, número de variables, número de restricciones, estructuras especiales, separabilidad de las variables en las restricciones y/o función objetivo, superficies representativas de las restricciones o del objetivo de rápida interpretación geométrica, etc.

La selección final de un método adecuado para un problema particulaar depende por tanto de las propiedades detalladas del modelo asi como de las técnicas de solución que formen parte del paquete de *software* de la instalación de computadora disponible.

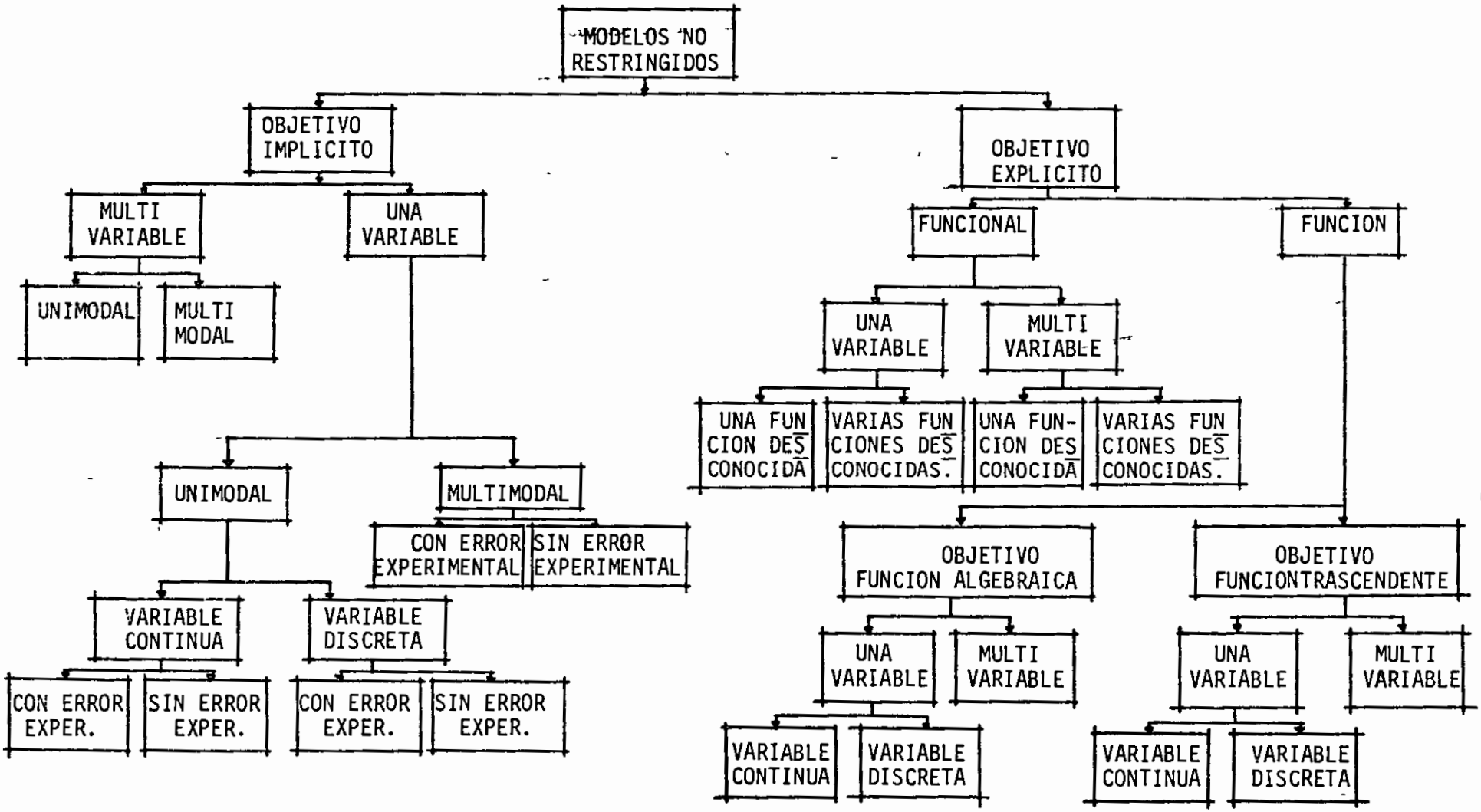


Fig. I.2a Clasificación de Modelos de Optimización

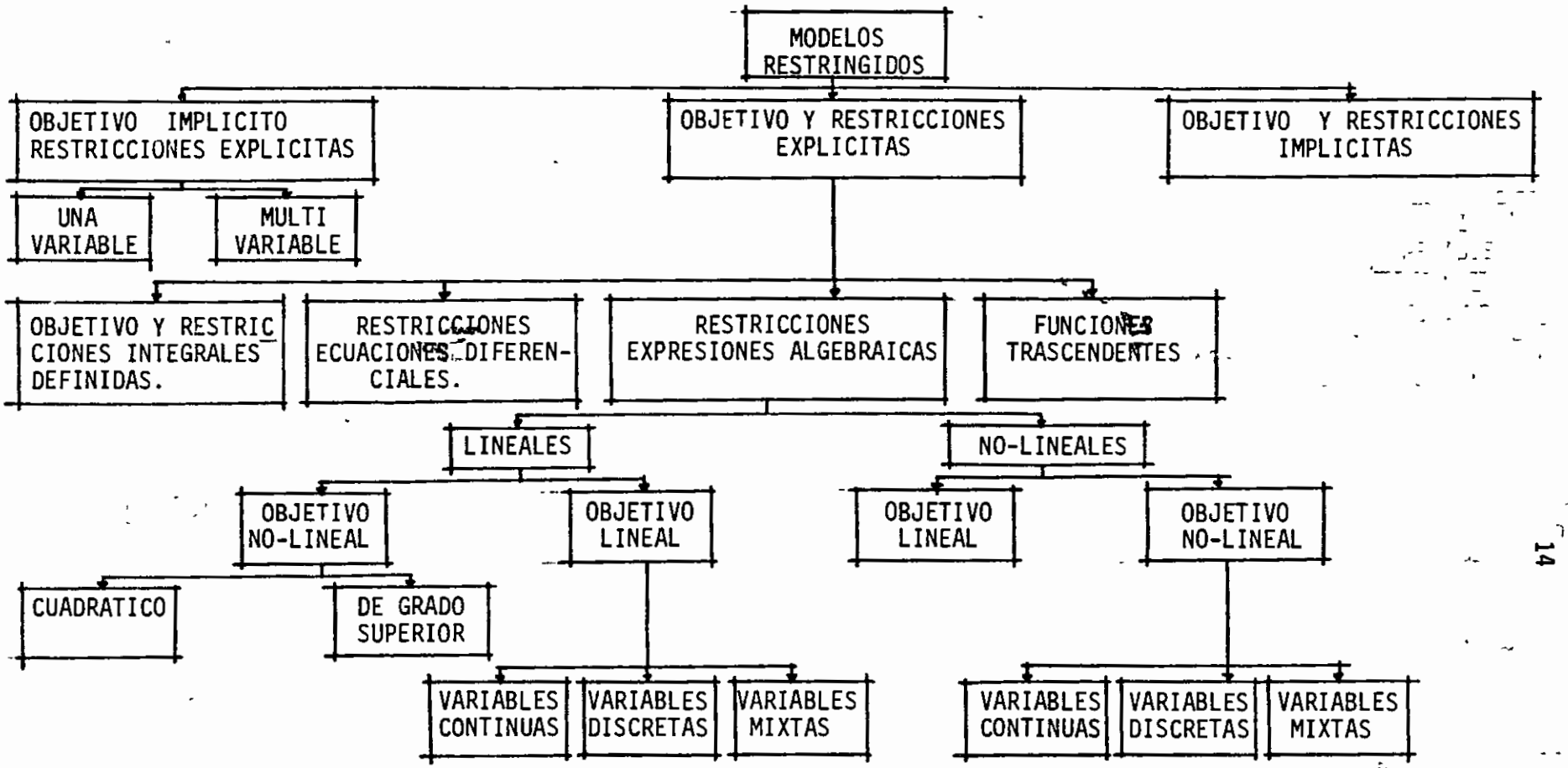


Fig. I.2b Clasificación de Modelos de Optimización

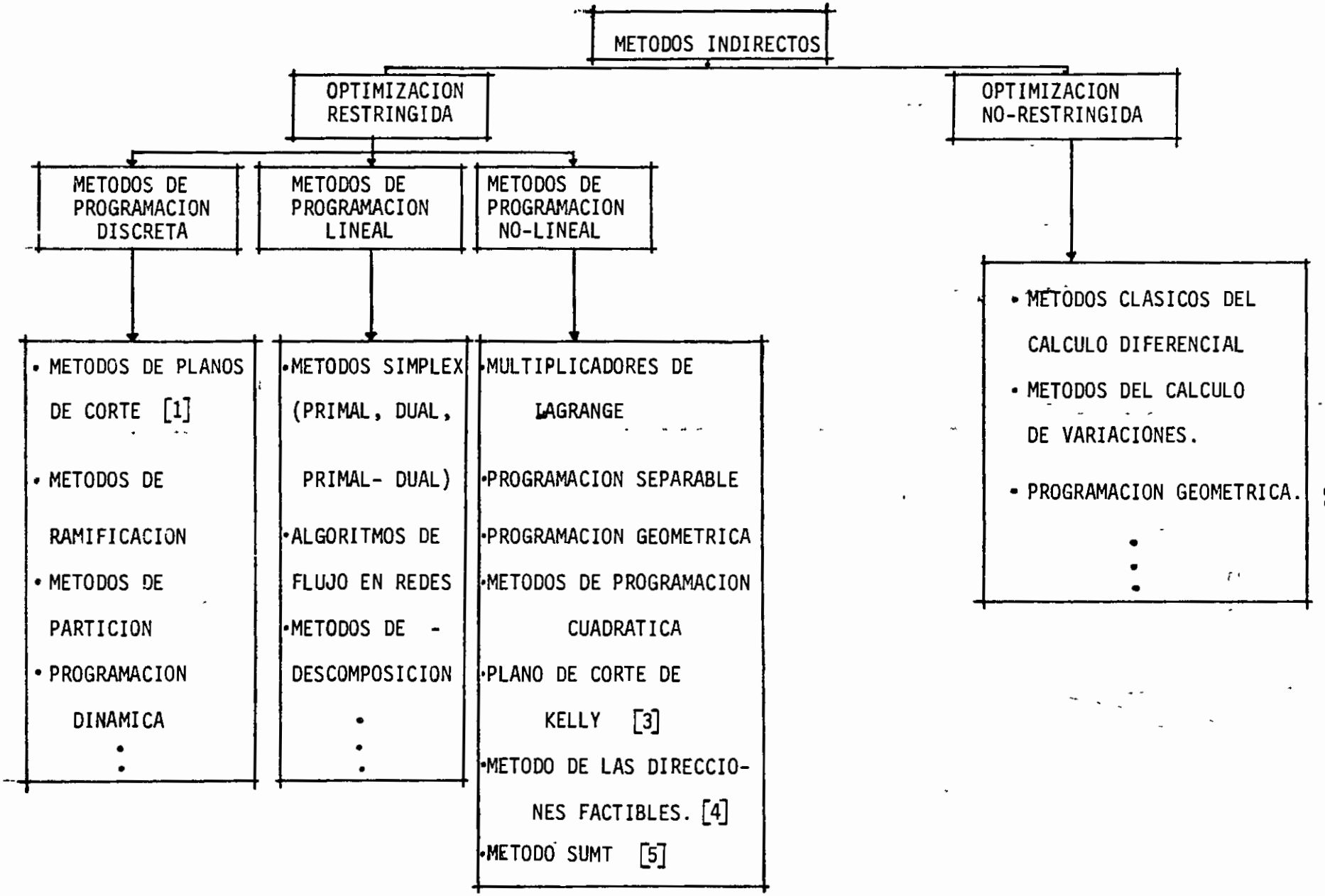
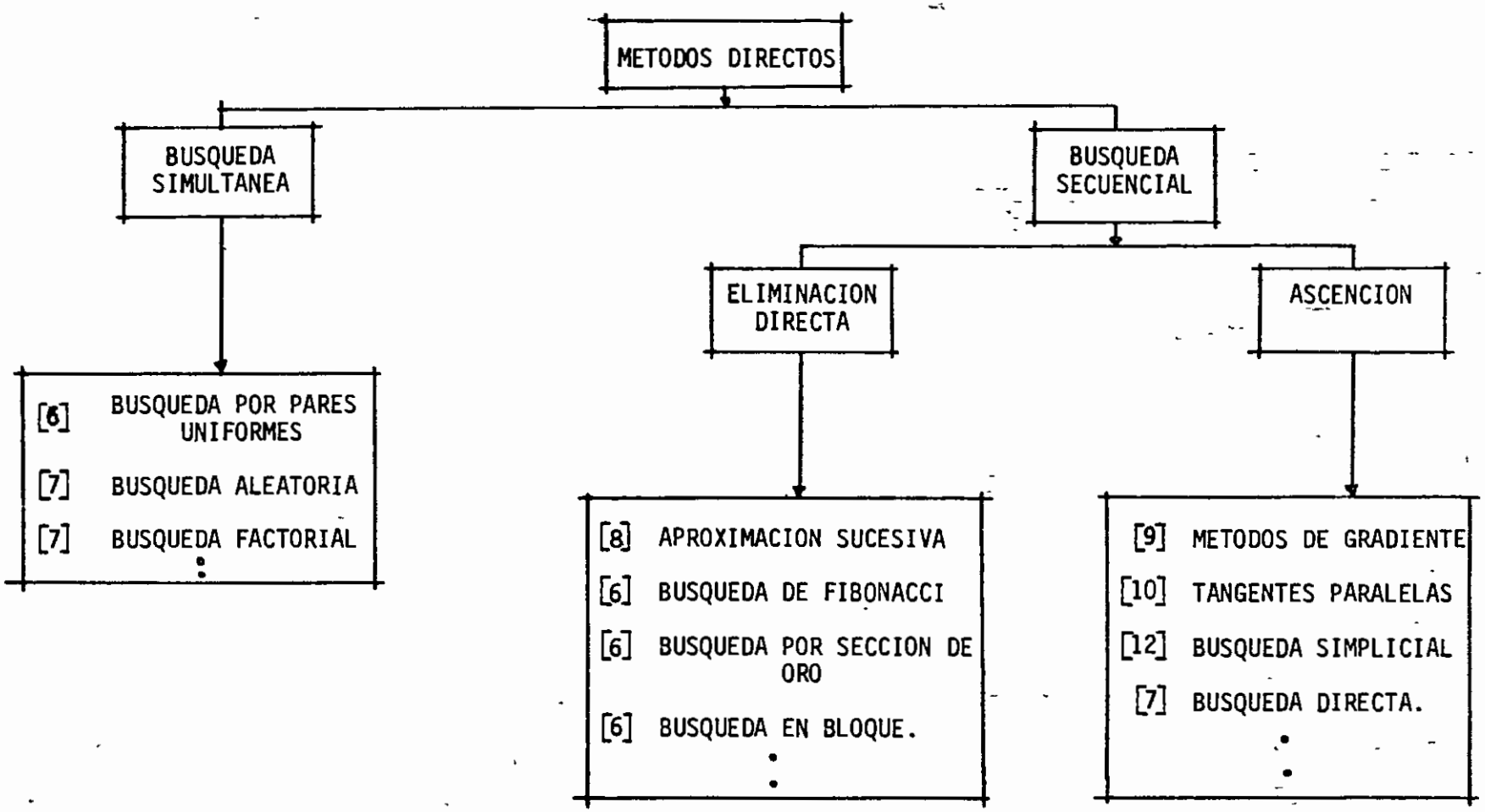


Fig. I.3a. Clasificación de Métodos de Solución



NOTA: Los números entre paréntesis corresponden a las referencias bibliográficas.

Fig. I.3b. Clasificación de los Métodos de Solución

REFERENCIAS

(1) Balinski, M.L., "Integer Programming: Method, Uses, Computation", *Management Science*, Vol. 12, No. 3, November 1965, pp. 255-264.

(2) Wolfe, P., "The Simplex Method for Quadratic Programming", *Econometrica*, Vol. 27, 1959, pp. 382-398.

(3) Kelley, J. E., Jr., "The Cutting Plane Method for Solving Convex Programs", *J. SIAM*, Vol. 8, December 1960, pp. 703-712.

(4) Zoutendijk, G., *Methods of Feasible Directions*, Elsevier Publishing Co., Amsterdam, 1960.

(5) Fiacco, A. V., and G. P. McCormick, "Computational Algorithm for the Sequential Unconstrained Minimization Technique for Nonlinear Programming", *Management Science*, Vol. 10, 1964, pp. 601-617.

(6) Wilde, D. J. and C. S. Beightler, *Foundations of Optimization*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1967.

(7) Brooks, S. H., "A Discussion of Random Methods for Seeking Maxima", *Operations Research*, Vol. 6, 1958, pp. 244-251.

(8) Berman, G., "Minimization by Successive Approximation", *J. SIAM Num. Anal.*, Vol. 3, 1966, pp. 123-133.

(9) Hadley, G., *Non Linear and Dynamic Programming*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1964, Ch. 9.

(10) Shah, B. V., R. J. Buehler, and O. Kempthorne, "Some Algorithms for Minimizing a Function of Several Variables", *J. SIAM*, Vol. 12, March 1964, pp. 74-92.

(11) Hooke, R. and T. A. Jeeves, "'Direct Search' Solution of Numerical and Statistical Problems", *J. Assn. Comp. Mach.*, Vol. 8, April 1961, pp. 212-229.

(12) Spendley, N., G. R. Hext, and F. R. Himsworth, "Sequential Application of Simplex Design in Optimization and Evolutionary Operations", *Technometrics*, Vol. 4, November, 1962, pp. 441-459.

(13) Ochoa, F. "Applications of Discrete Optimization Techniques to Capital Investment and Network Synthesis Problems", R 68-43 Massachusetts Institute of Technology, Enero 1968.

Modelo para la Evaluación de Proyectos

SINOPSIS

Se presenta en este trabajo un sistema para evaluar y seleccionar proyectos de electrificación rural Puede usarse además como herramienta para la asignación de presupuestos a zonas o regiones de un país.

La información que se requiere para su aplicación consiste de el presupuesto disponible para inversión, la política de electrificación a seguir, expresada en terminos cuantitativos mediante factores de ponderación; indicadores que permitan llegar a una medida de efectividad y el costo marginal asociado con cada proyecto

Se obtiene como resultado el conjunto de proyectos que hace máxima la función de beneficios, dentro del presupuesto disponible y las restricciones impuestas por la dependencia de algunos proyectos.

El sistema está integrado por modelos matemáticos implementados en equipos de cómputo electrónico que permiten efectuar análisis hasta de 10 000 proyectos

1.- INTRODUCCION.

1 1 - El Problema de Evaluación y Selección de Proyectos.

El sistema que aquí se presenta constituye un medio para ayudar a los funcionarios responsables de la Electrificación Rural a tomar decisiones No es un sustituto mecánico de su juicio, su experiencia y visión política sino que simplemente es una herramienta para auxiliarlos en las decisiones que deben tomar Su necesidad surge por dos hechos fundamentalmente

a) Que es necesario un mecanismo que permita conocer cuando un proyecto, es mejor que otro, sobre todo porque los beneficios que un proyecto de electrificación rural produce no tienen un valor en el mercado

b) Porque los recursos con que cuenta el gobierno siempre son menores para poder realizar todos los proyectos en un sólo período. En consecuencia, deben elegirse de esos proyectos aquellos que más contribuyan a lograr los objetivos nacionales, para que de esta forma, puedan liberarse los recursos que son escasos y valiosos y poder realizar más proyectos útiles

1.2.- Descripción del Sistema

El Sistema de Evaluación y Selección de Proyectos que se presenta esta formado por un conjunto de modelos matemáticos que, implementados en un computador electrónico, permiten obtener los proyectos que hacen máxima la función de beneficios

Los datos de entrada del sistema son

a) Información estadística, concerniente a cada proyecto, que se recolecta en formas específicamente diseñadas para tal fin.

b) Costos asociados con cada proyecto

c) La política de electrificación rural a seguir expresada mediante factores de ponderación

d) Un diagrama que ilustre la dependencia o independencia de los proyectos

c) El presupuesto disponible

Los resultados que se obtienen son

a) La lista de proyectos que hacen máxima la función de beneficios, definida en el capítulo 2

b) El valor de la función de beneficios

En el diagrama de la Figura 1 1 se ilustra la interacción de los elementos que integran el sistema

SISTEMA DE EVALUACION Y SELECCION DE PROYECTOS

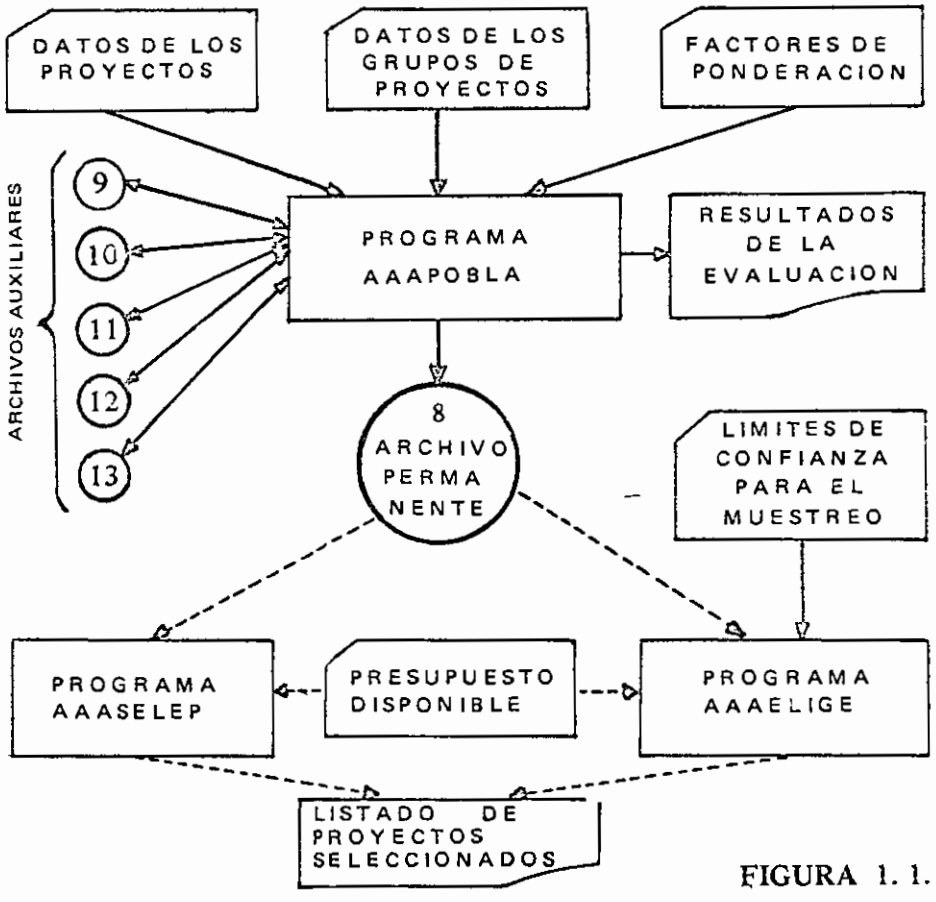


FIGURA 1. 1.

1 3 - Objetivos del Sistema

Los objetivos que se pretenden alcanzar mediante este Sistema son

a) Identificar las zonas de cada Estado, que por su potencial económico, sus condiciones sociales y por las inversiones marginales que deban efectuarse para realizar los proyectos de electrificación rural, las hagan más atractivas

b) Servir a los funcionarios como una herramienta que les permita formular programas de electrificación rural en los que se integren aquellos proyectos que mejor logren los objetivos fijados por el Gobierno Federal, y que a través de la Comisión Federal de Electricidad reciben las Juntas de Electrificación

1.4.- Elementos que Integran el Sistema

Los elementos que integran el Sistema de Evaluación y Selección de Proyectos son

a) Modelo para evaluar proyectos y zonas de un Estado, a partir de índices previamente definidos

b) Programa para computadora que calcula los índices de evaluación de las zonas

c) Programa para computador electrónico que permite transformar proyectos dependientes en proyectos independientes

d) Conjunto de programas para computador para evaluar hasta 10 000 proyectos, implementado en una maquina con 256 K Bytes Cada proyecto puede trabajarse con el número de indicadores que se desee.

e) Modelo matemático y algoritmo para resolver el problema de selección de Proyectos bajo Restricciones Presupuestales

f) Programa para computador en los que se implementa al algoritmo que resuelve el problema de seleccion Este conjunto de programas permite trabajar hasta con cinco grupos de proyectos dependientes y hasta 80 proyectos Consiste de un conjunto de subrutinas integradas con las que, a partir de los datos de costos de cada proyecto, los índices de evaluación calculados previamente (inciso d), e información sobre la dependencia entre proyectos, se obtiene el conjunto de proyectos que hace máxima la función de beneficios

g) Algoritmo para obtener, mediante métodos heurísticos, el conjunto de proyectos cuyos beneficios se aproxima al valor maximo de la funcion de beneficios, teniendo una cierta probabilidad de estar dentro de las mejores soluciones

h) Conjunto de programas en los que se implementa el algoritmo anterior Permite manejar hasta 10 000 proyectos pudiendo formar parte hasta de 400 grupos de proyectos dependientes

En las paginas siguientes se da una explicación detallada de cada uno de estos elementos En la Figura 1 2 se presenta un diagrama que muestra la secuencia del proceso de evaluación y selección de proyectos, en donde puede observarse la gran flexibilidad que se le da para su uso

2.- MODELO PARA LA EVALUACION DE PROYECTOS.

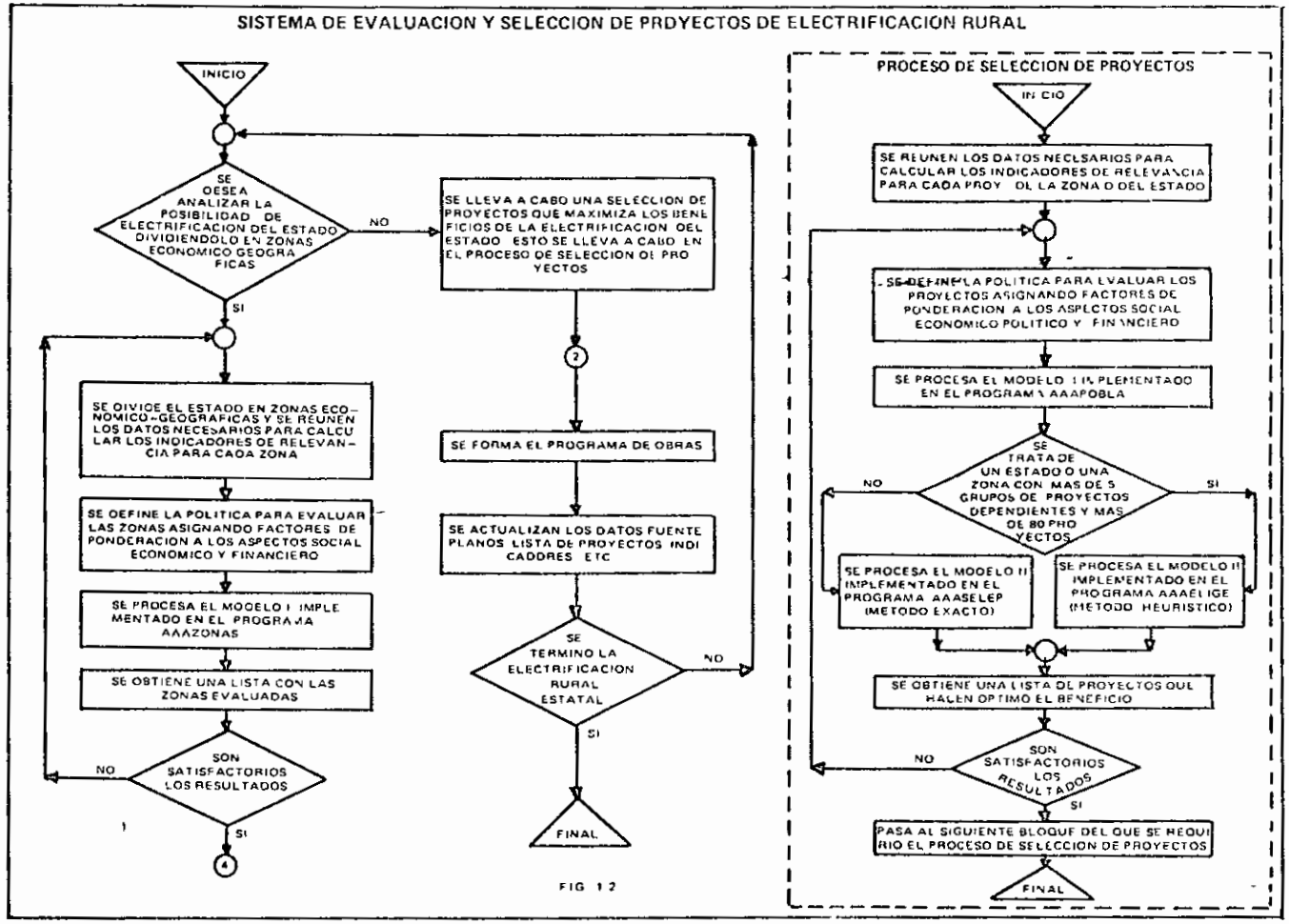
2 1 - Alcances y Limitaciones del Modelo

En el Sistema de Evaluación y Selección de Proyectos desarrollado, el objetivo de la evaluacion de proyectos es el establecimiento de jerarquías en terminos cuantitativos, de cada proyecto en relación a los demás Dado entonces un conjunto de proyectos, mediante su evaluacion se obtiene un número que representa una medida de la bondad de cada proyecto en relación a los demas

Es importante aclarar que la evaluación de proyectos en este Sistema no establece si un proyecto es o no conveniente desde un punto de vista social o económico absoluto, sino que se logra dar un número que refleja que tan bueno es un proyecto en relación a los demás Por tanto, es necesario que el sistema se aplique considerando el mayor número de proyectos posibles La aplicación de este modelo por etapas, en las que se tomen en cuenta diferentes proyectos cada vez, conducirá a resultados diferentes

Hechas estas aclaraciones se estará en posibilidad de ubicar debidamente la parte de evaluación que se abarca con el Sistema

Existen dos problemas de evaluación y selección distintos, principalmente porque cubren niveles y objetivos diferentes El primero se relaciona con los estudios que deben hacerse para definir la mejor dentro de un conjunto de alternativas con las que se pretende resolver un problema específico, en el que se trata de establecer cual es el proyecto que deberá proponerse para dotar de energía eléctrica a una localidad o a un centro de consumo potencial, mientras el segundo, que es el que se resuelve con el Sistema, intenta definir el conjunto de proyectos que deben llevarse a cabo con un presupuesto previamente determinado, de tal suerte que se logre el beneficio conjunto máximo


SISTEMA DE EVALUACION Y SELECCION DE PROYECTOS DE ELECTRIFICACION RURAL
INICIO
SE DESEA ANALIZAR LA POSIBILIDAD DE ELECTRIFICACION DEL ESTADO DIVIDIENDOLO EN ZONAS ECONOMICO GEOGRAFICAS
NO
SI
SE LLEVA A CABO UNA SELECCION DE PROYECTOS QUE MAXIMIZA LOS BENEFICIOS DE LA ELECTRIFICACION DEL ESTADO ESTO SE LLEVA A CABO EN EL PROCESO DE SELECCION DE PROYECTOS
2
SE DIVIDE EL ESTADO EN ZONAS ECONOMICO-GEOGRAFICAS Y SE REUNEN LOS DATOS NECESARIOS PARA CALCULAR LOS INDICADORES DE RELEVANCIA PARA CADA ZONA
SE FORMA EL PROGRAMA DE OBRAS
SE DEFINE LA POLITICA PARA EVALUAR LAS ZONAS ASIGNANDO FACTORES DE PONDERACION A LOS ASPECTOS SOCIAL ECONOMICO Y FINANCIERO
SE ACTUALIZAN LOS DATOS FUENTE PLANOS LISTA DE PROYECTOS INDICADORES ETC
SE PROCESA EL MODELO I IMPLEMENTADO EN EL PROGRAMA AAAZONAS
SE TERMINO LA ELECTRIFICACION RURAL ESTATAL
NO
SI
SE OBTIENE UNA LISTA CON LAS ZONAS EVALUADAS
SON SATISFACTORIOS LOS RESULTADOS
NO
SI
4
FINAL
FIG 1 2
PROCESO DE SELECCION DE PROYECTOS
INICIO
SE REUNEN LOS DATOS NECESARIOS PARA CALCULAR LOS INDICADORES DE RELEVANCIA PARA CADA PROY DE LA ZONA O DEL ESTADO
SE DEFINE LA POLITICA PARA EVALUAR LOS PROYECTOS ASIGNANDO FACTORES DE PONDERACION A LOS ASPECTOS SOCIAL ECONOMICO POLITICO Y FINANCIERO
SE PROCESA EL MODELO I IMPLEMENTADO EN EL PROGRAMA AAAPOBLA
SE TRATA DE UN ESTADO O UNA ZONA CON MAS DE 5 GRUPOS DE PROYECTOS DEPENDIENTES Y MAS DE 80 PROYECTOS
NO
SI
SE PROCESA EL MODELO II IMPLEMENTADO EN EL PROGRAMA AAASELEP (METODO EXACTO)
SE PROCESA EL MODELO II IMPLEMENTADO EN EL PROGRAMA AAAELIGE (METODO HEURISTICO)
SE OBTIENE UNA LISTA DE PROYECTOS QUE HALEN OPTIMO EL BENEFICIO
SON SATISFACTORIOS LOS RESULTADOS
NO
SI
PASA AL SIGUIENTE BLOQUE DEL QUE SE REQUIRIO EL PROCESO DE SELECCION DE PROYECTOS
FINAL

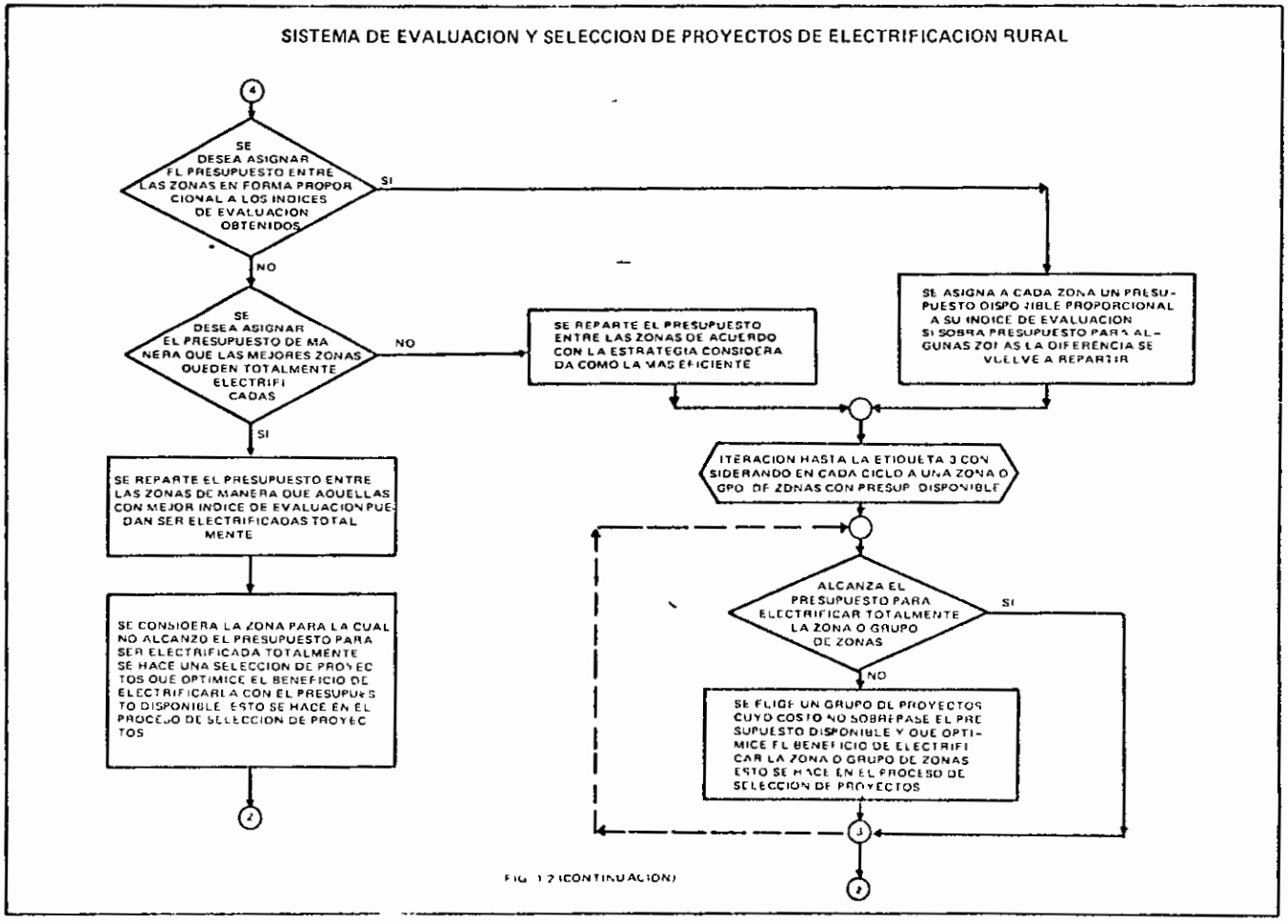


FIG. 1.2 (CONTINUACION)

2 2 - Formulacion del Modelo de Evaluación

(Modelo I en la figura 1 2)

Las características que debe tener un modelo que permita evaluar (en términos relativos) los proyectos de electrificación rural son

a) Deberá permitir obtener un índice general de evaluación al menor costo posible, ya que las obras de electrificación rural por su tamaño, no ameritan estudios elaborados y costosos

b) Deberá ser capaz de medir la efectividad de un proyecto en relación a cada uno de los objetivos que pretendan lograrse mediante la electrificación rural Para tal fin se usarán indicadores que midan el grado con que cada proyecto contribuye a lograr un objetivo específico

c) Deberá ser lo suficientemente flexible para llevar a cabo una política de desarrollo diferente en cada Estado o región del País Es decir, debe permitir asignar prioridades a los proyectos de electrificación atendiendo a diversas condiciones Así, en algunos Estados o Regiones sera el aspecto social el que predomine, mientras que en otras, el económico o el financiero

El modelo que se propone para calcular el índice general de evaluación de un proyecto y que cumple con las características arriba enunciadas es el que se da a continuación.

NOTACION·

E_i = índice de evaluación del proyecto i-ésimo

I_{ij} = valor del indicador j-ésimo en el proyecto i-ésimo

K_j = factor de ponderación del indicador j-ésimo

I^*_{ij} = valor máximo que alcanza el indicador j en los n proyectos

$\overset{\Lambda}{I}_{ij}$ = valor mínimo que alcanza el indicador j en los n proyectos

n = número de proyectos

m = número de indicadores

Se tiene entonces

$$E_i = \sum_{j=1}^{m} \frac{I_{ij} - \Lambda I_{ij}}{I^*_{ij} - \Lambda I_{ij}} \times 100\, K_j \qquad (1)$$

De esta forma se logra que todos los indicadores queden medidos dentro de un rango fijo entre 1 y K_j, independientemente de sus valores originales.

3 - SELECCION DE PROYECTOS

3 1 - Modelos de Selección Optima de Inversiones

La naturaleza de los modelos matemáticos asociados con problemas de inversión es muy diversa Sin embargo, las componentes principales que son comunes a todos ellos son

a El conjunto de proyectos, i = 1,2, , n propuestos para inversión

b La erogación asociada con cada proyecto y su distribución en el tiempo

c El beneficio relativo total asociado con la aceptación de cada proyecto

d El número de años que cubre la planeación (horizonte de planeación) y el numero de períodos considerados para la inversión

e Los presupuestos disponibles para cada período

El conjunto de proyectos puede estar ligado o no por determinados tipos de dependencia que pueden ser tecnológicas, operacionales o económicas Los casos de dependencia e independencia entre los proyectos de electrificacion se analizarán someramente dentro de este trabajo Sin embargo, se dan las siguientes definiciones

Un proyecto es independiente si los costos y beneficios que su ejecución involucra no se ven afectados por la realización de otros proyectos El beneficio total, en consecuencia, es igual a la suma de los beneficios de cada proyecto del conjunto, como si se llevara a cabo solo

El modelo que se presenta en este trabajo fue realizado bajo la hipótesis de proyectos independientes No obstante, mediante algunas adaptaciones puede llegar a manejar cierto tipo de proyectos dependientes Estos se discuten el el inciso 3 3

3 2 - Formación de Programas de Obras

Para integrar en un programa los proyectos por ejecutar, es necesario establecer un criterio que considere

a Cuándo un proyecto, con el que pretenden lograrse objetivos múltiples, es mejor que otro

b Qué proyectos deben formar parte del programa, si los recursos disponibles no permiten la construccion de todos

El problema planteado en el inciso a, puede resolverse en términos relativos mediante el modelo de evaluación propuesto. Para el inciso b, es importante hacer notar que los recursos disponibles no son suficientes para llevar a cabo todos los proyectos y por tanto será necesario escoger entre la lista de proyecto evaluados y jerarquizados, aquellos que deberán ejecutarse o a pasar a formar parte de la lista de espera para ser considerados en la formacion de los siguientes Programas

Puesto que, como resultado de la evaluación se obtiene un índice cuyo valor señala que tan bueno es cada proyecto en relación a los demás, en el proceso de seleccion deberá tratarse que con el presupuesto disponible, los Proyectos que se elijan para formar el Programa de Obras, conduzcan al maximo valor posible de la suma de sus índices de evaluación (beneficios)

Los modelos que se formulan para seleccionar entre un conjunto de proyectos, aquellos que hacen máxima la función de beneficios para un presupuesto dado pertenecen a la rama de la programación matemática llamada programación entera, pues no es posible elegir fracciones de proyecto

Los primeros algoritmos desarrollados en los últimos años para resolver este problema, presentan una serie de inconvenientes y limitaciones que los hacen practicamente inoperables Sin embargo, aprovechando la estructura de la matríz de coeficientes del problema, el Dr Felipe Ochoa Rosso (Ref 1) desarrollo un algoritmo fundamentado en el método de ramificación, con el que es posible resolver este tipo de problemas en forma muy eficiente

Este algoritmo implementado en la computadora de la Comisión Federal de Electricidad (IBM-360-50) permite llegar a resolver problemas hasta de 6 etapas de inversión y 80 proyectos Para problemas que involucran un mayor numero de proyectos (10 000), se desarrolló un método de solucion heuristico, que permite encontrar una solución factible con la que se tiene una cierta probabilidad de estar dentro de un porcentaje determinado de las mejores soluciones posibles Ambas

cifras, la probabilidad y el porcentaje definen el tamaño de una muestra, que representa el numero de pruebas a realizar para verificar que el valor de la funcion objetivo encontrado primeramente no es superado

3 3 FORMULACION DE LOS MODELOS MATEMATICOS DE SELECCION (modelo II en la figura 1 2)

En este inciso se formularán los modelos aplicables a proyectos de electrificacion rural, bajo distintas condiciones de dependencia

3 3 1 Notación

Se adoptará la siguiente notación

B_i = Beneficio relativo del proyecto i-ésimo

C_i = Costo total involucrado en la ejecución del proyecto i-ésimo

P = Presupuesto disponible

X_i = Variable de decisión que adopta unicamente valores 0 ó 1

Z = Función de beneficios

G = Conjunto de indices correspondientes a los grupos de proyectos dependientes

M = Conjunto de parejas (j', j) de proyectos pertenecientes a grupos de proyectos dependientes en las que el proyecto (j') depende directamente del proyecto (j)

N_k = Conjunto de índices correspondientes a los proyectos múltiples generados por el grupo k $(k \in G)$ de proyectos dependientes (ver párrafo 3 3 4)

N = Numero total de proyectos considerados

3 3 2 Modelo para proyectos independientes

En este caso, como se muestra en la figura 3 1, los proyectos son independientes unos de otros La electrificacion de cualquier poblado se puede llevar a cabo independientemente de la electrificacion de los demas poblados

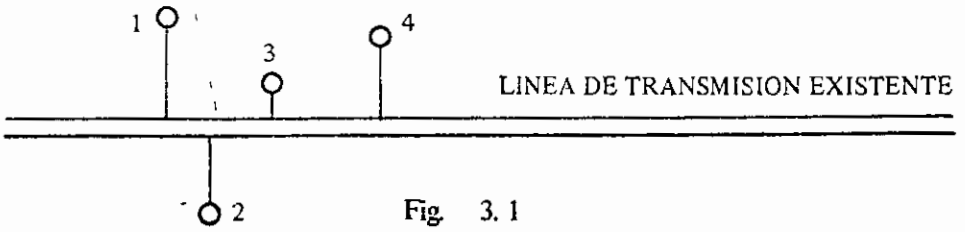


Fig. 3. 1

El modelo propuesto para este caso es el siguiente

$$\max \qquad Z = \sum_{i=1}^{n} B_i \; X_i \qquad (2)$$

$$\text{sujeto a} \qquad \sum_{i=1}^{n} C_i \; X_i \leqslant P \qquad (3)$$

$$\text{y} \qquad x_i = 0, 1 \quad \forall_i \qquad (4)$$

El problema corresponde al problema clásico de programación todo entero con una sola restricción la restriccion presupuestal

3 3 3 Modelo para proyectos dependientes en secuencia (a)

En este caso, además de proyectos independientes, se tienen grupos de proyectos en los que los proyectos dependen unos de otros para su realización

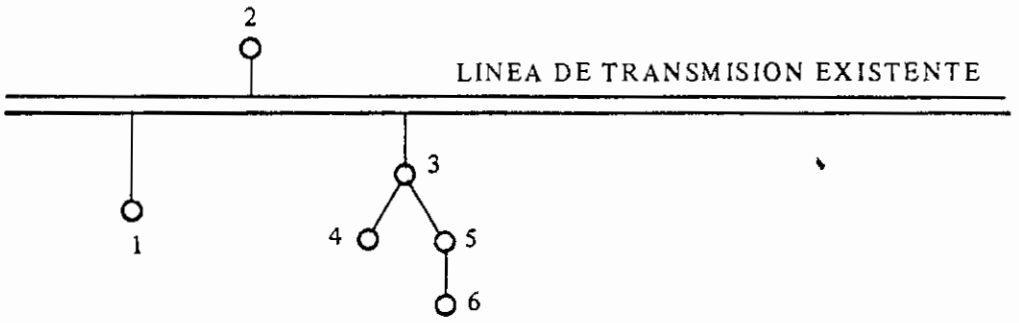


Fig. 3. 2

Así, por ejemplo, en la figura 3 2 el poblado-4 se puede electrificar solo cuando ya ha sido electrificado el poblado 3, etc

El modelo propuesto para este caso es como sigue

$$\max \quad Z = \sum_{i=1}^{n} B_i X_i \qquad (5)$$

$$\text{sujeto a} \quad \sum_{i=1}^{n} C_i X_i \leq P \qquad (6)$$

$$X_{j'} \leq x_j \qquad (j', j) \in M \qquad (7)$$

$$\text{y} \quad X_i = 0, 1 \qquad \forall_i \qquad (8)$$

Para el problema de la figura 3 2 se tendrían tres restricciones del tipo (7) correspondientes a las parejas $(j' j)$ de proyectos
$(4, 3)$, $(5, 3)$ y $(6, 5)$

3 3 4 Modelo para proyectos dependientes en secuencia (b).

Se presenta a continuacion un modelo alternativo al presentado en el inciso 3 3 3

La diferencia consiste en la forma de tratar los proyectos que forman parte de los grupos de proyectos dependientes. En vez de considerar estos proyectos como proyectos individuales, en cada grupo se generan proyectos múltiples de manera de considerar todas las formas de electrificar el grupo de poblados

El modelo para este caso es el siguiente

$$\max \; Z = \sum_{i=1}^{n} B_i X_i \qquad (9)$$

$$\text{sujeto a} \; \sum_{i=1}^{n} C_i X_i \leq P \qquad (10)$$

$$\sum_{j \in N_k} X_j \leq 1 \quad k \in G \qquad (11)$$

$$\text{y} \quad X_i = 0,1 \qquad \forall_i \qquad (12)$$

Es necesario hacer notar que el número n de proyectos considerados para el modelo no coincide con el número de proyectos individuales propuestos para la electrificación En el problema de la figura 3 2 se tienen 6 poblados por electrificar mientras que para el modelo se considerarían los siguientes 8 proyectos

Proyecto para el modelo	Formado por los proyectos individuales
1	1
2	2
3	3
4	3,4
5	3,5
6	3,4,5
7	3,5,6
8	3,4,5,6

Para el mismo ejemplo se tendría una sola restricción del tipo (11) en la que N_1 = 3,4,5,6,7,8 Todos los proyectos que se analizan son independientes, pero aquellos que pertenecen a N_1 son mutuamente exclusivos

4 Ejemplo.

Para ilustrar la aplicación del sistema expuesto, se presentan enseguida los resultados obtenidos mediante el mismo

En la figura **4** 1 se muestra un conjunto de poblados La única instalacion existente es la línea de transmisión Asimismo, los poblados A y B están electrificados Para cada proyecto se consideraron 25 indicadores con sus respectivos factores de ponderación Los resultados de la evaluación se muestran en la figura 4 2 A partir de estos resultados se procesó el modelo descrito en el inciso 3 3 4 con lo que se obtuvo una solucion óptima Z = 273 426 para el presupuesto disponible de 4 000 00 Los proyectos elegidos se muestran con línea gruesa en la figura 4 1

Finalmente, en la figura **4** 3 se muestra la curva de efectividad-costo correspondiente al problema

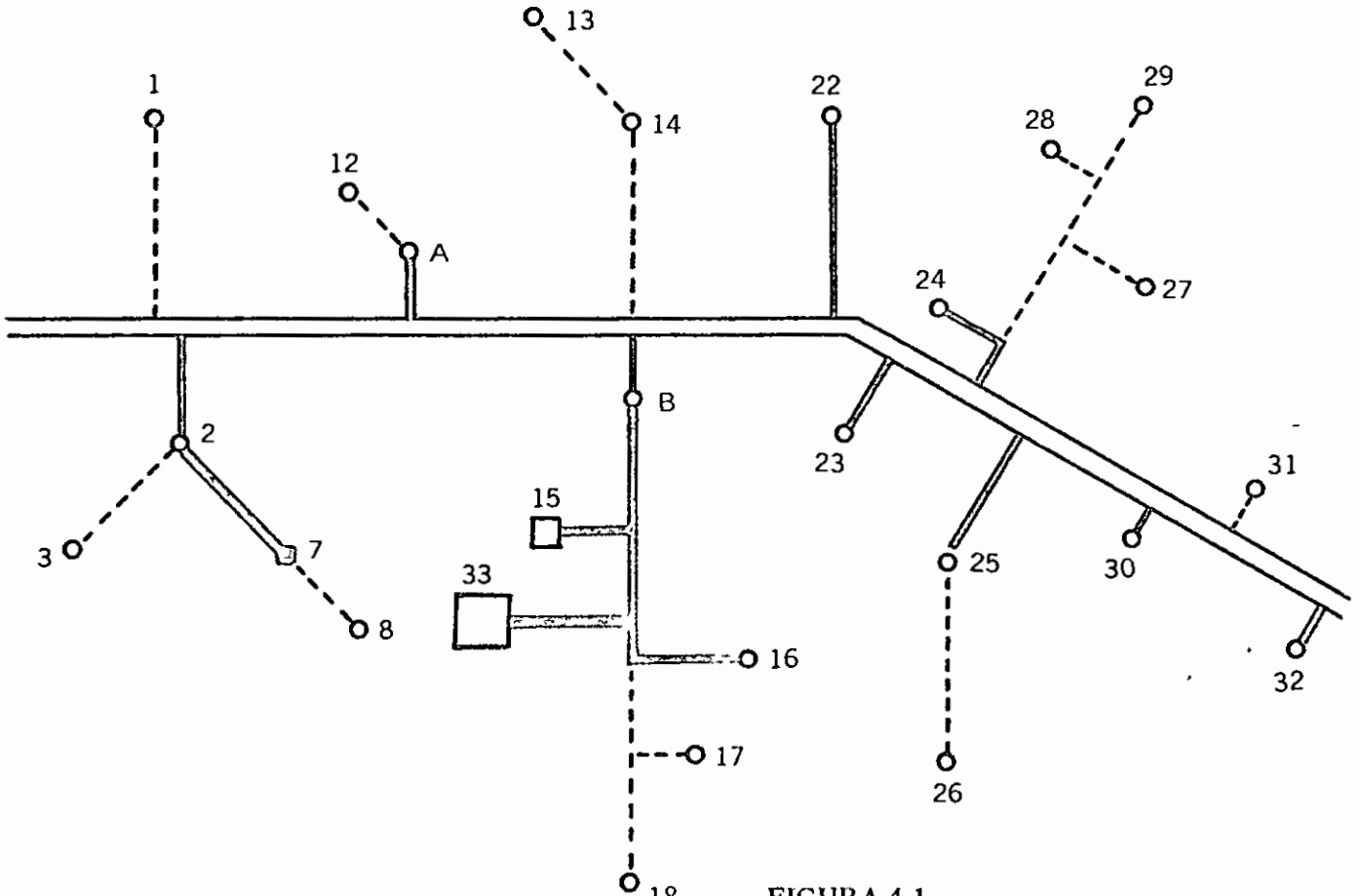


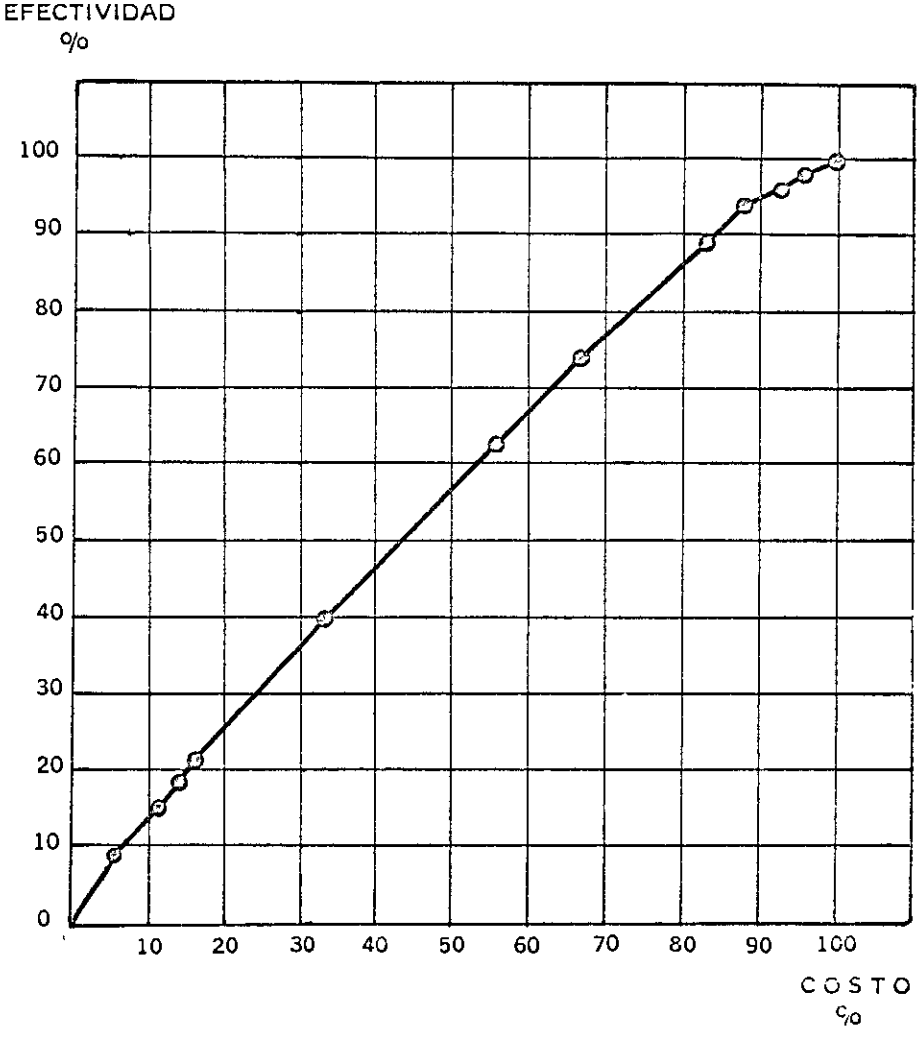
FIGURA 4 1

FIGURA 4–2

VALORES DE LOS INDICADORES

NUM DEL PROYECTO	SOCIAL	ECONOMICO	POLITICO	FINANCIERO	GENERAL (BENEF)	COSTO	BENEF/COSTO (POR CIEN)	POBLADOS QUE FORMAN PARTE DEL PROYECTO
1	16 832	8 487	1 106	2 081	28 505	330 000	8 638	22
2	10 648	6 022	1 055	1 731	19 457	240 000	8 107	23
3	8 427	1 524	1 169	1 464	12 583	169 000	7 446	32
4	3 113	1 048	1 093	1 090	6 344	90 000	7 048	30
5	47 375	9 414	1 144	5 246	63 178	900 000	7 020	2 7
6	55 341	10 148	1 144	5 550	72 182	1043,000	6 921	2 7 8
7	35 645	24 636	4 760	8 442	73 483	1096 000	6 705	15 16 33
8	22 500	4 601	4 648	3 288	35 072	535 000	6 556	14 13
9	58 795	25 753	5 151	10 301	100 000	1606 000	6 227	15 16 33 17 18
10	61 808	10 250	1 144	6 236	79 438	1279 000	6 211	2 3 7 8
11	53 842	9 516	1 144	5 933	70 435	1136 000	6 200	2 3 7
12	39 086	9 376	1 144	5 723	55 328	892 000	6 203	25
13	41 156	25 154	4 974	8 984	80 268	1300 000	6 174	15 16 33 17
14	31 451	8 764	1 257	5 578	47 050	788 000	5 971	15
15	6 431	1 752	1 030	1 505	10 718	184 000	5 825	12
16	45 535	9 931	1 156	6 690	63 302	1118 000	5 663	25 26
17	9 780	1 578	1 106	2 084	14 548	257 500	5 650	24
18	12 205	1 605	1 118	1 972	16 901	300 000	5 634	2
19	35 645	8 815	1 320	5 951	51 731	941 000	5 497	15 16
20	5 228	3 773	4 583	1 997	15 581	293 000	5 318	14
21	38 930	2 830	1 245	5 726	48 730	1035 000	4 708	24 27 28 29
22	1 030	1 030	1 169	1 030	4 259	92 000	4 630	31
23	18 672	1 708	1 118	2 658	24 157	536 000	4 507	2 3
24	23 138	2 194	1 182	4 220	30 733	721 000	4 263	24 27 28
25	16 024	1 656	1 144	2 824	21 647	517 500	4 183	24 27
26	2 867	1 040	1 093	1 398	6 398	184 000	3 477	1

FIGURA 4.3


EFECTIVIDAD
%
100
90
80
70
60
50
40
30
20
10
0
10
20
30
40
50
60
70
80
90
100
COSTO
%

5 Reconocimientos

Los autores desean expresar su reconocimiento al Dr Felipe Ochoa Rosso por sus sugerencias y contribución mediante el paquete de programas elaborados para resolver el problema de Seleccion de Proyectos, al Ing Enrique Contreras, Presidente de la Junta de Jalisco por sus valiosos comentarios y su paciencia durante las entrevistas que concedió en la primera etapa de este trabajo, y al Ing Romárico Arroyo por sus opiniones y puntos de vista.

Francisco Monteverde Zubirán

Salomón Camhaji Samra

Enrique de la Mora Askinasky

6 Bibliografía

1 Felipe Ochoa Rosso "Applications of Discrete Optimization Techniques to Capital Investment and Network Synthesis Problems" Tésis Doctoral MIT Press

2 F Hanssmann, "Operations Research Techniques for Capital Investment". John Wiley editor

3 Salomón Camhaji, Francisco Monteverde "Evaluación de Proyectos de Electrificación Rural" III CLER

4 Felipe Ochoa Rosso "Aplicaciones de la Teoría de Optimización a la Selección de Inversiones" Revista del Colegio de Ingenieros Civiles

5 James W Davis "Politics, Programs and Budgets" Prentice Hall

6 Raymond A Baner y Kenneth J Gergen "The Study of Policy Formation" Collier-Macmillan

7 Praeger. "Cost Effectiveness Analysis New Approaches in Decision Making Work"

8 Varios autores "PPBS and Benefit-Cost Analysis"

9 G Hadley "Nonlinear and Dynamic Programming" . Addisson Wesley

10 North-Halland "Integer and Nonlinear Programming" J Abadie Editor

11 North-Holland "Nonlinear Programming" J. Abadie Editor

ASIGNACION OPTIMA DE RECURSOS

En esta parte se discutirán los problemas genéricos mos - trados en relación con:

- Planteamiento del problema
- Procedimientos (algoritmos) de solución
- Aspectos teóricos
- Interpretaciones económicas

1. Problemas de transporte
2. Problemas de asignación (assignment)
3. Problema general de asignación (allocation) lineal (problemas de programación lineal)
4. Programación paramétrica
5. Programación entera

PROBLEMAS A DISCUTIR

I. EJEMPLOS DE PROBLEMAS DE TRANSPORTE

1. Pag. 143 del texto.
2. Problemas del transporte (descripción general): Las cantidades de recursos $b_1, b_2, \ldots, b_i, \ldots, b_m$ disponibles en m "origenes", deben asignarse a n des - tinos 1, 2, ..., s, ..., n, los cuales tienen res - pectivamente demandas de estos recursos en las cantidades $a_1, a_2, \ldots, a_j, \ldots, a_n$. Sea c_{ij} el costo por asignar una unidad de recurso del origen i al destino j. Suponiendo que la cantidad total de re - cursos es igual a la demanda total (transporte ba -

lanceado), encontrar para cada origen i y cada destino j la cantidad x_{ij} de recursos del origen i que debe asignarse al destino j, de modo que se satisfagan las demandas en todos los destinos a un costo mínimo.

Matemáticamente: dados C_{ij} $i = 1, 2, .., m, j = 1, 2, .., n$

$$b_i \geq 0$$

$$a_i \geq 0$$

encontrar $X_{ij} \geq 0$

de modo que sea mínimo $\sum_{j=1}^{n} \sum_{i=1}^{m} C_{ij} X_{ij}$

cumpliéndose $\sum_{j=1}^{n} X_{ij} = a_i, i = 1, .., m$

$$\sum_{i=1}^{m} X_{ij} = b_j, j = 1, .., n$$

En forma de tabla escribimos

ORIGENES	DESTINOS				RECURSOS
	1	2	... j	... n	
1	$C_{11}(X_{11})$	$C_{12}(X_{12})$	$\cdots C_{1j}(X_{1j})$	$\cdots C_{1n}(X_{1n})$	b_1
2	$C_{21}(X_{21})$	$C_{22}(X_{22})$	$\cdots C_{2j}(X_{2j})$	$\cdots C_{2n}(X_{2n})$	b_2
.					.
.					.
.					.
i	$C_{i1}(X_{i1})$	$C_{i2}(X_{i2})$	$\cdots C_{ij}(X_{ij})$	$\cdots C_{in}(X_{in})$	b_i
.					.
.	.	.	... .	... .	.
m	$C_{m1}(X_{m1})$	$C_{m2}(X_{m2})$	$\cdots C_{mj}(X_{mj})$	$\cdots C_{mn}(X_{mn})$	b_m
DEMANDAS	a_1	a_2	a_j	a_n	

TABLA DE COSTOS Y REQUERIMIENTOS. EN LOS PARENTESIS UNA SOLUCION X_{ij}.

3. Una compañía debe producir un cierto producto en cantidad suficiente para cumplir con las ventas contratadas por los siguientes cuatro meses. La capacidad de producción para este producto está limitada en cantidades diferentes para los respectivos meses. Los costos unitarios de producción también varian de acuerdo a la disponibilidad de personal. El producto puede ser producido en un mes y retenido para venderse en un mes posterior, pero a un costo de almacenamiento de $ 1.00 por unidad por mes. No se incurre en costos de almacenamiento para producto producido en ese mismo mes. Por razones de caducidad, la compañía desea tener existencia nula de este producto al final de los cuatro meses. Según los datos dados en la siguiente tabla, ¿Cuánto debe producirse en cada uno de los cuatro meses con el fin de hacer mínimo el costo total? .

MES	VENTAS CONTRATADAS	PRODUCCION MAXIMA	COSTO UNITARIO DE PRODUCCION	COSTO UNITARIO DE ALMACENAMIENTO POR MES
1	20	30	8	1
2	30	50	12	1
3	50	20	11	1
4	40	50	14	1
	(140)	(150)		

Aparentemente este no es un problema de transporte. Antes de hacer ver que así es, consideremos el siguiente modelo de nuestro problema, el cual probablemente sería el que se plantearía en un primer análisis:

x_i = producción en el mes i (i=1, 2, 3, 4)

Minimizar $Z = 8x_1 + 12x_2 + 11x_3 + 14x_4 + (x_1 - 20) + (x_1 + x_2 - 50) + (x_1 + x_2 + x_3 - 100) = 8x_1 + 14x_2 + 12x_3 + 14x_4 - 170$

con las restricciones

$x_1 \leqslant 30$		$x_1 \geqslant 20$	
$x_2 \leqslant 50$	limitación de	$x_1 + x_2 \geqslant 50$	cumplir
$x_3 \leqslant 20$	la producción	$x_1 + x_2 + x_3 \geqslant 100$	ventas
$x_4 \leqslant 50$		$x_1 + x_2 + x_3 + x_4 \geqslant 140$	contratadas

$x_1 \geqslant 0$	
$x_2 \geqslant 0$	producción no
$x_3 \geqslant 0$	negativa
$x_4 \geqslant 0$	

Este es un problema de programación lineal, el cual requiere para su solución de mucho más operaciones aritméticas que el correspondiente modelo de transporte que veremos a continuación:

Para obtener esta formulación es necesario identificar los "orígenes" y los "destinos" así como las X_{ij}, C_{ij}, a_i y b_j:

origen i = producción en el mes i (i = 1, 2, 3, 4)

destino j= ventas en el mes j (j = 1, 2, 3, 4)

X_{ij} = cantidad producida en el mes i para vender en el mes j ($i \leqslant j$)

C_{ij} = costo unitario (producción + almacenamiento) aso-

ciado con X_{ij} $(i \leq j)$

a_i = ?

b_1 = ventas contratadas en el mes j

En la siguiente tabla se da finalmente el planteamiento en un problema balanceado de transporte, agregando un destino "ficticio" (el número 5) con costos de transporte cero.

	DESTINO					
ORIGEN	1	2	3	4	5	RECURSOS
1	8	9	10	11	0	30
2	M	12	13	14	0	50
3	M	M	11	12	0	20
4	M	M	M	14	0	50
DEMANDA	20	30	50	40	10	(150)

M es un número arbitrariamente grande

TABLA DE COSTOS Y REQUERIMIENTOS

EJERCICIOS DE TAREA

(TRANSPORTE)

1. Resolver el ejercicio 1(a) del texto.

2. Una compañía ha decidido iniciar la producción de algunos o todos, de cinco nuevos productos en tres plantas con capacidad de producción excedida. Estos productos se venden por peso; y con el fin de poder hacer comparaciones, consideramos como unidad de un producto al peso correspondiente a un precio de venta de $ 100.00. La capacidad de producción disponible (por unidad de tiempo) en cada una de las plantas es:

Planta	capacidad en número de unidades*
1	40
2	60
3	90

* De cualquiera de los cinco productos o combinación entre ellos.

El departamento de ventas ha previsto las siguientes ventas potenciales por unidad de tiempo:

Producto	Ventas potenciales
1	30
2	40
3	70
4	40
5	60

Los costos variables unitarios para la producción de estos productos en cada una de las plantas son:

	Producto				
Planta	1	2	3	4	5
1	20	19	14	21	16
2	15	20	13	19	16
3	18	15	18	20	X

La X de la tabla indica que la planta 3 no puede producir el producto 5.

Se desea saber que cantidades de los respectivos productos debe producir cada una de las tres plantas:

a) Hacer el planteamiento como un problema de transporte balanceado.

b) Resolver el problema de transporte.

Solución b: Las cantidades óptimas que debe producir cada planta están dadas por:

	Producto				
Planta	1	2	3	4	5
1			10		30
2			60		
3	30	40		20	

PROGRAMACION DINAMICA Y LA ASIGNACION DE RECURSOS.

R. Carvajal.

Introducción.

La programación dinámica es una técnica matemática empleada en el estudio de procesos de decisiones múltiples.

A diferencia de programación lineal, no existe una formulación canónica del problema de programación dinámica. Más aún, no existe un conjunto de reglas simples que puedan aplicarse a diferentes problemas con objeto de extraer sus "propiedades dinámicas". El reconocimiento de las condiciones que debe satisfacer cierto problema para aplicarle las técnicas de programación dinámica requiere de experiencia. Esta puede desarrollarse a través del estudio de aplicaciones de programación dinámica y de las características comúnes a los diferentes casos.

Usos de la Programación Dinámica.

En general los modelos de programación dinámica se aplican a problemas de menor tamaño que los factibles para programación lineal. Por otro lado, la solución provista por programación dinámica contiene mucho más información que la provista por programación lineal.

A continuación se dan varias aplicaciones clásicas de modelos de programación dinámica.

1. Asignación de Proyectos.

Un empresario o funcionario público tiene la cantidad de $ 500,000 para asignarla a 5 proyectos. El beneficio derivado de cada proyecto está relacionado con la cantidad que se asigne al proyecto. Sólo es posible asignar múltiplos de $ 100,000. La relación entre la asignación y los beneficios se ilustra en la Tabla 1.

Unidades asignadas (1)	0	1	2	3	4	5
Proyecto						
1	0	10	13	14	15	16
2	0	5	7	9	12	15
3	0	0	8	16	20	21
4	0	6	9	12	14	14
5	0	8	12	14	15	15

(1) x unidades = x00 000 pesos.

Tabla 1. Beneficio derivado de la asignación de recursos a los diferentes proyectos.

El problema del funcionario es el asignar los $ 500 000 de tal forma que su utilidad sea máxima.

2. Problema de la Diligencia.

Como las diligencias no son elementos comunes en el folklore mexicano, formularemos el problema de la manera siguiente:

Un camión lleno de mercancía viene de un país del norte a la Ciudad de México. Su dueño ha "olvidado" el pagar los derechos de importación de dicha mercancía. Existen diferentes rutas a la ciudad de México. El paso entre cada dos ciudades mayores está asociado a un costo de "inspección" por parte de la policía. El problema del dueño es escoger la ruta que minimice los costos de "inspección" (Figura 1).

etapas

(6).....

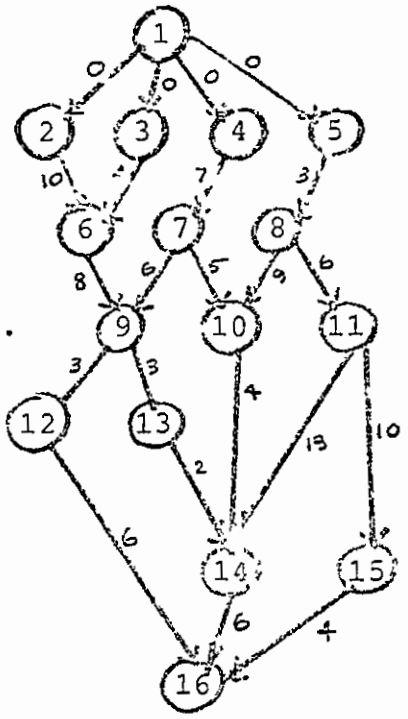
(5).....

(4).....

(3).....

(2).....

(1).....



1. Orígen
2. Mexicali
3. Nogales
4. Ciudad Juárez
5. Piedras Negras
6. Hermosillo
7. Jiménez
8. Monclova
9. Mazatlán
10. Torreón
11. Saltillo
12. Guadalajara
13. Durango
14. Zacatecas
15. San Luis Potosí
16. México

Figura 1.

3. Problema del Matrimonio.

Durante su vida don Nicandro conocerá a 4 posibles esposas, una a la vez. Don Nicandro tiene la habilidad de reconocer si la posible esposa es excepcional (E), buena (B), regular (R), o "no hay de otra" (N). El les asigna un valor relativo de 40, 30, 20 ó 10 unidades respectivamente.

La frecuencia con que los diferentes tipos de posible conyugue pueden presentarse varía con el tiempo. Así, en su temprana juventud no hay muchas posibilidades de que se presente una mujer excepcional dada la poca experiencia del jóven Nicandro. Esta frecuencia aumenta conforme don Nicandro gana experiencia, pero vuelve a decrecer conforme su juventud se desvanece.

Don Nicandro puede casarse o no con la novia en turno. Si la rechaza no hay forma de encontrarla de nuevo. El problema de Don Nicandro es el tomar las decisiones que maximicen su valor esperado tomando en cuenta los valores relativos que el asigna a cada tipo de mujer (Figura 2).

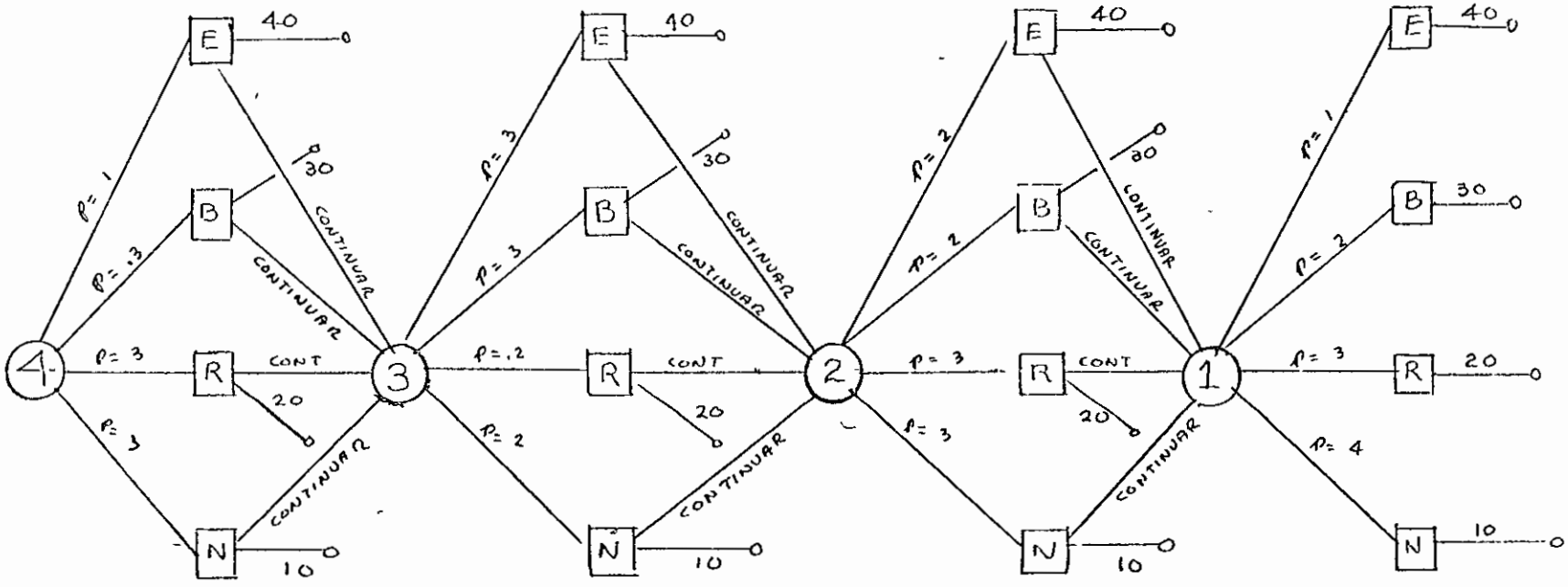


Figura 2. Red de posibilidades de Don Nicandro.

Enunciaremos brevemente, otras aplicaciones de la programación dinámica.

4. Determinación del conjunto óptimo de partes de repuesto que minímice el número de veces que una máquina esté parada por falta de repuestos.

5. Selección de los mejores medios de publicidad con objeto de obtener el máximo efecto sobre las ventas.

6. Determinación de las épocas de revisión y mantenimiento de maquinaria con objeto de tener el menor número de descomposturas.

7. Determinación de políticas óptimas de inventarios consistentes en la institución de un nivel en el cual debe ordenarse el producto y la cantidad a ordenar.

Características de los Problemas de Programación Dinámica.

i. El problema puede dividirse en etapas, con decisiones que tomar en cada etapa.

ii. Cada etapa tiene un número de estados asociados a ella.

iii. Principio de Optimalidad. Dado el estado actual, las decisiones óptimas para las etapas remanentes son independientes de las decisiones tomadas en las etapas previas. En otras palabras:

Una política óptima (conjunto de decisiones óptimas) debe tener la propiedad de que, independientemente de la forma en la que llegamos a cierto estado, las siguientes decisiones deben constituir una política óptima.

Ejemplo. En el caso del camión con mercancía, el dueño sabe que el camino óptimo para ir de Torreón a la Ciudad de México no depende de la ruta por la cual llegó a Torreón.

iv. El método de solución procede etapa por etapa, empenzando por la etapa final.

v. El problema puede expresarse por medio de una formulación recursiva, es decir, la política óptima dado que se está en el estado s con n etapas por recorrer es función de las políticas óptimas cuando tenemos n-1 etapas por recorrer.

La notación empleada en la formulación de modelos de programación lineal puede causar dificultades en un principio por loque hay que revisarla con cuidado, sobre todo la primera vez que se ve.

Formulación de Modelos de Programación Dinámica.

Denótese por

s el estado

n el número de etapas remanentes

fn(s) el valor de la política óptima cuando se está en el estado s con n etapas remanentes.

Examinaremos esta notación usando los tres primeros ejemplos:

1. Asignación de proyectos.

s número de unidades (1 unidad = $ 100 000) asignadas (0, 1, 2,3,4,5)

n número de proyectos a asignar.

fn(s) máxima utilidad cuando se asignan s unidades a n proyectos.

jn(s) número óptimo de unidades que se asignaron al proyecto n cuando se dispone de s unidades a asignar entre n proyectos.

Relacion recursiva.

Denotamos por Pn(s) el beneficio obtenido al asignar s unidades al proyecto n.

$$fn(s) = \max_{x \in \{0,1,2,3,4,5\}} \{Pn(x) + fn-1\ (s-x)\}$$

$$f0(s) = 0.$$

Esta fórmula expresa el hecho de que se debe comparar todas las formas en que se pueden asignar x unidades al proyecto n (los proyectos remanentes son n) y el resto a los otros n-1 proyectos. La mejor forma de hacer esta asignación es la que nos maximiza las ganancias.

2. Problema de la diligencia.

s ciudad en que se encuentra el camión

n etapas para llegar a México (en este caso hay varias formas de dividir el problema en etapas).

fn(s) costo mínimo de "inspección" dado que se está en la ciudad s con n etapas por recorrer.

jn(s) decisión óptima sobre la próxima ciudad a visitar dado que se está en la ciudad s con n etapas por recorrer.

Relación de recursión.

Se define como Cst el costo de "inspección"incurrido al viajar

de la ciudad s a la t.

$$f_n(s) = \min_t \{C_{st} + f_{n-1}(t)\}$$

$$f_0(s) \equiv 0$$

3. Problema del matrimonio

S tipo de novia (E,B,R,N)

n número de la novia (1,2,3,4)

$f_n(S)$ máximo valor esperado dado que el tipo de novia actual es S y hay n novias por considerar

$j_n(S)$ decisión óptima sobre el problema de casarse o no casarse dado que el tipo de novia actual es S y hay n novias por considerar.

Relación de recursión.

Llamemos $P_n(S)$ la probabilidad de que la n novia sea del tipo S y C_s el valor relativo asignado a las novias.

El valor esperado cuando hay n-1 novias por venir es $\sum_t P_{n-1}(t) f_{n-1}(t)$, es decir la suma del producto de la probabilidad de que en la etapa n-1 la novia sea de tipo t multiplicado por el mejor valor del objetivo dado que la novia es de tipo t con n-1 novias por considerar (incluyendo la actual novia cuyo tipo es t).

Por lo tanto:

$$fn(S) = \max_{t} \{C_s, P_{n-1}(t) f_{n-1}(t)\}$$

$$fo(S) \equiv 0$$

METODO DE SOLUCION

1. Problema de la asignación de proyectos.

Cuando solamente hay un proyecto remanente la solución es inmediata

$$f_1(s) = p_1(s)$$

s	$f_1(s)$	$J_1(s)$
0	0	0
1	10	1
2	13	2
3	14	3
4	15	4
5	16	5

n=2

Asignación	Asignación Proyecto 2	Asignación proyectos remanentes	Utilidad Proyecto 2	Utilidad proyectos remanentes	Total	Decisión óptima
S	x	s-x	$p_2(x)$	$f_1(s-x)$		$J_2(s)$
0	0	0	0	0	0	0
1	0	1	0	10	10	0
	1	0	5	5	0	-
2	0	2	0	13	13	-
	1	1	5	10	15	1
	2	0	7	0	7	-
3	0	3	0	14	14	-
	1	2	5	13	18	1
	2	1	7	10	17	-
	3	0	9	0	9	-
4	0	4	0	15	15	-
	1	3	5	14	19	-
	2	2	7	13	20	-
	3	1	9	10	21	3
	4	0	12	0	12	-
5	0	5	0	16	16	-
	1	4	5	15	20	-
	2	3	7	14	21	-
	3	2	9	13	22	3
	4	1	12	10	22	-
	5	0	15	0	15	-

n=3

S	x	s-x	$p_3(x)$	$f_2(s-x)$	Total	$j_3(s)$
0	0	0	0	0	0	0
1	0	1	0	10	10	0
	1	0	0	0	0	-
2	0	2	0	15	15	0
	1	1	0	10	10	-
	2	0	8	0	8	-
3	0	3	0	18	18	0
	1	2	0	15	15	-
	2	1	8	10	18	-
	3	0	16	0	16	-
4	0	4	0	21	21	-
	1	3	0	18	18	-
	2	2	8	15	23	-
	3	1	16	10	26	3
	4	0	20	0	20	-
5	0	5	0	22	22	-
	1	4	0	21	21	-
	2	3	8	18	26	-
	3	2	16	15	31	3
	4	1	20	10	30	-
	5	0	21	0	21	-

n=4

S	x	n-x	$p_4(x)$	$f_3(s-x)$	Total	$j_4(x)$
0	0	0	0	0	0	0
1	0	1	0	10	10	0
	1	0	6	0	6	
2	0	2	0	15	15	-
	1	1	6	10	16	1
	2	0	9	0	9	-
3	0	3	0	18	18	-
	1	2	6	15	21	1
	2	1	9	10	19	-
	3	0	12	0	12	-
4	0	4	0	26	26	0
	1	3	6	18	24	-
	2	2	9	15	24	-
	3	1	12	10	22	-
	4	0	14	0	14	-
5	0	5	0	31	31	-
	1	4	6	26	32	1
	2	3	9	18	27	-
	3	2	12	15	27	-
	4	1	14	10	24	-
	5	0	14	0	14	-

n=5

S	x	n-x	$p_5(x)$	$f_4(s-x)$	total	$J_5(x)$
0	0	0	0	0	0	0
1	0	1	0	10	10	0
	1	0	8	0	8	-
2	0	2	0	16	16	-
	1	1	8	10	18	1
	2	0	12	0	12	-
3	0	3	0	21	21	-
	1	2	8	16	24	1
	2	1	12	10	22	-
	3	0	14	0	14	-
4	0	4	0	26	26	-
	1	3	8	21	29	1
	2	2	12	16	28	-
	3	1	14	10	24	-
	4	0	15	0	15	-
5	0	5	0	32	32	-
	1	4	8	26	34	1
	2	3	12	21	33	-
	3	2	14	16	30	-
	4	1	15	10	25	-
	5	0	15	0	15	-

Solución óptima

$f_5(5) = 34$
$J_5(5) = 1$
$J_4(4) = 0$
$J_3(4) = 3$
$J_2(1) = 0$
$J_1(1) = 1$

2. Problema de la diligencia

Para n=1 la solución es inmediata

$f_1(14) = 6$ $\qquad$ $J_1(14) = 16$
$f_1(15) = 4$ $\qquad$ $J_1(15) = 16$

n=2

Origen	Destino intermedio	Costo	Resto del viaje	Total	Decisión
S	t	C_{st}	$f_1(t)$		$j_2(s)$
12	16	6	0	6	16
13	14	2	6	8	14

n=3

S	t	C_{st}	$f_2(t)$	Total	$J_3(s)$
9	12	3	6	9	12
	13	3	8	11	-
10	14	4	6	10	14
11	14	13	8	21	-
	15	10	4	14	15

n=4

S	t	C_{st}	$f_3(t)$	Total	$j_4(s)$
6	9	8	9	17	9
7	9	6	9	15	9
	10	5	10	15	
8	10	9	10	23	-
	11	6	14	20	11

n=5

S	t	C_{st}	$f_4(t)$	Total	$J_5(t)$
2	6	10	17	27	6
3	6	4	17	21	6
4	7	7	15	22	7
5	8	3	20	23	8

Solución óptima:

$f_6(1) = 21$
$J_6(1) = 3$
$J_5(3) = 6$
$j_4(6) = 9$
$J_3(9) = 12$
$J_2(12) = 16$

En variadas ocasiones un problema puede resolverse usando técnicas diferentes. Así, el problema de la diligencia puede resolverse empleado programación lineal como sigue:

Sea $X_{st} = \begin{cases} 0 & \text{si el camión no pasa por la ruta s a t} \\ 1 & \text{" " " " pasa " " " " " " } \end{cases}$

El problema es

$$\text{Min} \sum_{\text{arcos}} C_{st} X_{st}$$

Sujeto a

$$X_{12}+X_{13}+X_{14}+X_{15} = 1$$

$$X_{12} - X_{26} = 0$$

$$X_{13} - X_{36} = 0$$

conservación de flujo

$$X_{14}-X_{47} = 0$$

$$X_{15}-X_{58} = 0$$

$$X_{26}+X_{36}-X_{69} = 0$$

$$X_{47}-X_{79}-X_{710} = 0$$

$$X_{58}-X_{810}-X_{811} = 0$$

$$X_{69}+X_{79}-X_{912}-X_{913} = 0$$

$$X_{710}+X_{810}-X_{1014} = 0$$

$$X_{811}-X_{1114}-X_{1115} = 0$$

$$X_{912}-X_{1216} = 0$$

$$X_{913} -X_{1314}=0$$

$$X_{1314}+X_{1014} + X_{1114}-X_{1416} = 0$$

$$X_{1115}-X_{1516} = 0$$

$$X_{1216}+X_{1416}+X_{1516} = 1$$

$$X_{st} \geq 0$$

Este es un modelo de transporte cuya solución con programación lineal es entera por lo que podemos ignorar las restricciones

$$X_{st} = \{0 \text{ ó } 1\}$$

La solución que obtenemos por programación lineal nos da la ruta óptima de 1 a 16. La solución que se obtiene con programación dinámica no sólo nos da esta solución, sino las rutas óptimas de cualquier punto a la Ciudad de México. La solución más completa se obtiene pagando el precio de un número mayor de operaciones.

3. Problema del matrimonio

Llamemos $\text{suma}_1 = \sum_t P_i(t) f_i(t)$

Cuando sólo queda una novia la decisión es casarse.

n=1

S	$f_1(s)$	$p_1(s)$	Total
E	40	.1	4
B	30	.2	6
R	20	.3	6
N	10	.4	4
		Suma =	20

O sea, el valor esperado cuando falta una novia por ver es 20.

n=2

S	C_s	$Suma_1$	$f_2(s)$	$p_2(s)$	Total	Decisión
E	40	20	40	.2	8	Casarse
B	30	20	30	.2	6	Casarse
R	20	20	20	.3	6	Casarse o continuar
N	10	20	20	.3	6	Continuar

$Suma_2 = 26$

n=3

S	C_s	$Suma_2$	$f_2(s)$	$p_2(s)$	Total	Decisión
E	40	26	40	.3	12	Casarse
B	30	26	30	.3	9	Casarse
R	20	26	26	.2	5.2	Continuar
N	10	26	26	.2	5.2	Continuar

$Suma_3 = 31.4$

n=4

S	C_s	$Suma_3$	$f_3(s)$	$p_3(s)$	Total	Decisión
E	40	31.4	40	.1	4	Casarse
B	30	31.4	31.4	.3	9.42	Continuar
R	20	31.4	31.4	.3	9.42	Continuar
N	10	31.4	31.4	.3	9.42	Continuar

$Suma_4 = 32.26$

La interpretación de $suma_1$ es la siguiente:

Cuando don Nicandro tiene i novias potenciales por venir en promedio, el tendrá una con valor $suma_i$. Así, al empezar su vida en promedio él se casará con una mujer tipo B(32.26). Cuando sólo le falta conocer una, en promedio se casará con una tipo R(20).

REFERENCIAS

1. Hillier, F.S., Lieberman, G.J., Introduction to Operations Research, Holden-Day, Inc., 1967.

2. Wagner, H.M., Principles of Operations Research, Prentice-Hall, Inc., 1969.

3. Bellman, R.E., Dynamic Programming, Princeton University Press, 1957.

4. Bellman, R.E., Dreyfus, S.E., Applied Dynamic Programming, Princeton University Press, 1962.

PROGRAMACION DINAMICA Y LA ASIGNACION DE RECURSOS.

R. Carvajal.

Introducción.

La programación dinámica es una técnica matemática empleada en el estudio de procesos de decisiones múltiples.

A diferencia de programación lineal, no existe una formulación canónica del problema de programación dinámica. Más aún, no existe un conjunto de reglas simples que puedan aplicarse a diferentes problemas con objeto de extraer sus "propiedades dinámicas". El reconocimiento de las condiciones que debe satisfacer cierto problema para aplicarle las técnicas de programación dinámica requiere de experiencia. Esta puede desarrollarse a través del estudio de aplicaciones de programación dinámica y de las características comúnes a los diferentes casos.

Usos de la Programación Dinámica.

En general los modelos de programación dinámica se aplican a problemas de menor tamaño que los factibles para programación lineal. Por otro lado, la solución provista por programación dinámica contiene mucho más información que la provista por programación lineal.

A continuación se dan varias aplicaciones clásicas de modelos de programación dinámica.

1. Asignación de Proyectos.

Un empresario o funcionario público tiene la cantidad de $ 500,000 para asignarla a 5 proyectos. El beneficio derivado de cada proyecto está relacionado con la cantidad que se asigne al proyecto. Sólo es posible asignar múltiplos de $ 100,000. La relación entre la asignación y los beneficios se ilustra en la Tabla 1.

Unidades asignadas (1)	0	1	2	3	4	5
Proyecto						
1	0	10	13	14	15	16
2	0	5	7	9	12	15
3	0	0	8	16	20	21
4	0	6	9	12	14	14
5	0	8	12	14	15	15

(1) x unidades = x00 000 pesos.

Tabla 1. Beneficio derivado de la asignación de recursos a los diferentes proyectos.

El problema del funcionario es el asignar los $ 500 000 de tal forma que su utilidad sea máxima.

2. Problema de la Diligencia.

Como las diligencias no son elementos comunes en el folklore mexicano, formularemos el problema de la manera siguiente:

Un camión lleno de mercancía viene de un país del norte a la Ciudad de México. Su dueño ha "olvidado" el pagar los derechos de importación de dicha mercancía. Existen diferentes rutas a la ciudad de México. El paso entre cada dos ciudades mayores está asociado a un costo de "inspección" por parte de la policía. El problema del dueño es escoger la ruta que minimice los costos de "inspección" (Figura 1).

etapas

(6).....

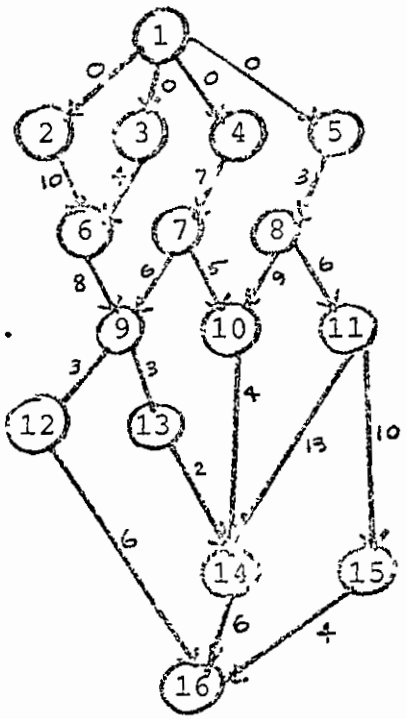
(5).....

(4).....

(3).....

(2).....

(1).....



1. Orígen
2. Mexicali
3. Nogales
4. Ciudad Juárez
5. Piedras Negras
6. Hermosillo
7. Jiménez
8. Monclova
9. Mazatlán
10. Torreón
11. Saltillo
12. Guadalajara
13. Durango
14. Zacatecas
15. San Luis Potosí
16. México

Figura 1.

3. Problema del Matrimonio.

Durante su vida don Nicandro conocerá a 4 posibles esposas, una a la vez. Don Nicandro tiene la habilidad de reconocer si la posible esposa es excepcional (E), buena (B), regular (R), o "no hay de otra" (N). El les asigna un valor relativo de 40, 30, 20 ó 10 unidades respectivamente.

La frecuencia con que los diferentes tipos de posible conyugue pueden presentarse varía con el tiempo. Así, en su temprana juventud no hay muchas posibilidades de que se presente una mujer excepcional dada la poca experiencia del jóven Nicandro. Esta frecuencia aumenta conforme don Nicandro gana experiencia, pero vuelve a decrecer conforme su juventud se desvanece.

Don Nicandro puede casarse o no con la novia en turno. Si la rechaza no hay forma de encontrarla de nuevo. El problema de Don Nicandro es el tomar las decisiones que maximicen su valor esperado tomando en cuenta los valores relativos que el asigna a cada tipo de mujer (Figura 2).

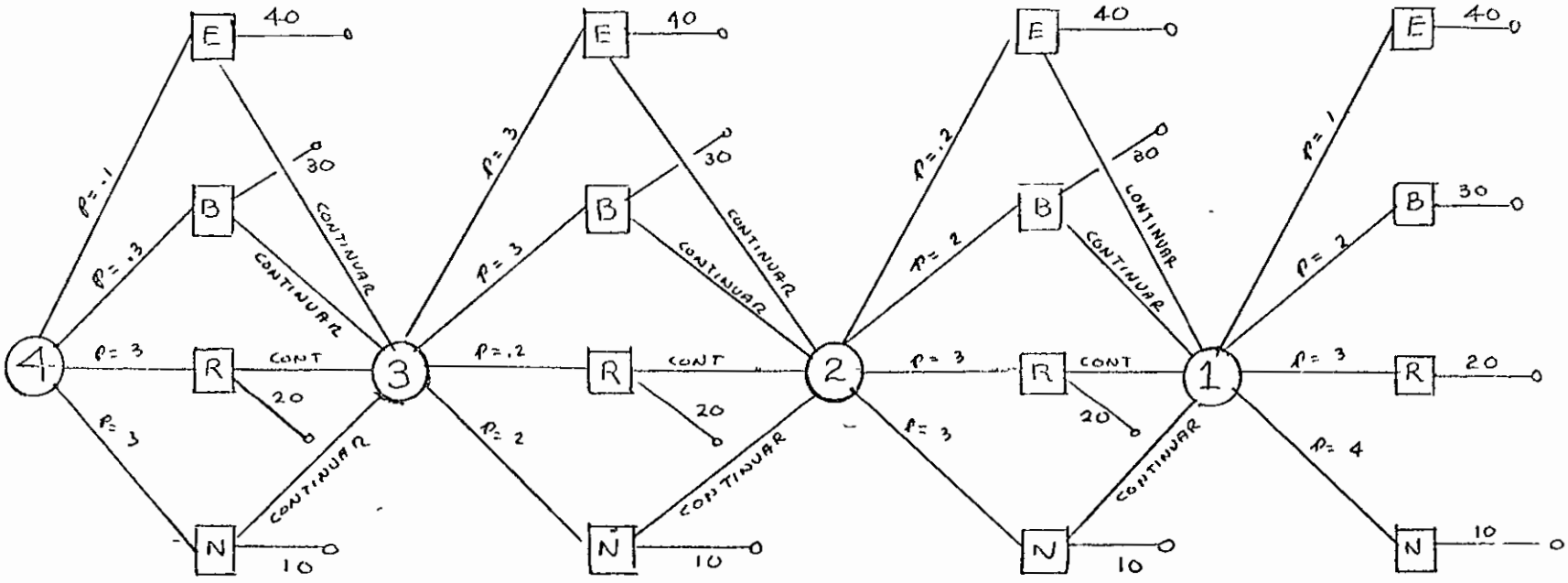


Figura 2. Red de posibilidades de Don Nicandro.

Enunciaremos brevemente, otras aplicaciones de la programación dinámica.

4. Determinación del conjunto óptimo de partes de repuesto que minimice el número de veces que una máquina esté parada por falta de repuestos.

5. Selección de los mejores medios de publicidad con objeto de obtener el máximo efecto sobre las ventas.

6. Determinación de las épocas de revisión y mantenimiento de maquinaria con objeto de tener el menor número de descomposturas.

7. Determinación de políticas óptimas de inventarios consistentes en la institución de un nivel en el cual debe ordenarse el producto y la cantidad a ordenar.

Características de los Problemas de Programación Dinámica.

i. El problema puede dividirse en etapas, con decisiones que tomar en cada etapa.

ii. Cada etapa tiene un número de estados asociados a ella.

iii. Principio de Optimalidad. Dado el estado actual, las decisiones óptimas para las etapas remanentes son independientes de las decisiones tomadas en las etapas previas. En otras palabras:
Una política óptima (conjunto de decisiones óptimas) debe tener la propiedad de que, independientemente de la forma en la que llegamos a cierto estado, las siguientes decisiones deben constituir una política óptima.

Ejemplo. En el caso del camión con mercancía, el dueño sabe que el camino óptimo para ir de Torreón a la Ciudad de México no depende de la ruta por la cual llegó a Torreón.

iv. El método de solución procede etapa por etapa, empenzando por la etapa final.

v. El problema puede expresarse por medio de una formulación recursiva, es decir, la política óptima dado que se está en el estado s con n etapas por recorrer es función de las políticas óptimas cuando tenemos n-1 etapas por recorrer.

La notación empleada en la formulación de modelos de programación lineal puede causar dificultades en un principio por loque hay que revisarla con cuidado, sobre todo la primera vez que se ve.

Formulación de Modelos de Programación Dinámica.

Denótese por

s el estado

n el número de etapas remanentes

fn(s) el valor de la política óptima cuando se está en el estado s con n etapas remanentes.

Examinaremos esta notación usando los tres primeros ejemplos:

1. Asignación de proyectos.

s número de unidades (1 unidad = $ 100 000) asignadas (0, 1, 2,3,4,5)

n número de proyectos a asignar.

fn(s) máxima utilidad cuando se asignan s unidades a n proyectos.

jn(s) número óptimo de unidades que se asignaron al proyecto n cuando se dispone de s unidades a asignar entre n proyectos.

Relación recursiva.

Denotamos por Pn(s) el beneficio obtenido al asignar s unidades al proyecto n.

$$f_n(s) = \max_{x \in \{0,1,2,3,4,5\}} \{P_n(x) + f_{n-1}(s-x)\}$$

$$f_0(s) = 0.$$

Esta fórmula expresa el hecho de que se debe comparar todas las formas en que se pueden asignar x unidades al proyecto n (los proyectos remanentes son n) y el resto a los otros n-1 proyectos. La mejor forma de hacer esta asignación es la que nos maximiza las ganancias.

2. Problema de la diligencia.

s ciudad en que se encuentra el camión

n etapas para llegar a México (en este caso hay varias formas de dividir el problema en etapas).

fn(s) costo mínimo de "inspección" dado que se está en la ciudad s con n etapas por recorrer.

jn(s) decisión óptima sobre la próxima ciudad a visitar dado que se está en la ciudad s con n etapas por recorrer.

Relación de recursión.

Se define como Cst el costo de "inspección"incurrido al viajar

de la ciudad s a la t.

$$f_n(s) = \min_t \{C_{st} + f_{n-1}(t)\}$$

$$f_0(s) \equiv 0$$

3. Problema del matrimonio

S tipo de novia (E,B,R,N)

n número de la novia (1,2,3,4)

$f_n(S)$ máximo valor esperado dado que el tipo de novia actual es S y hay n novias por considerar

$j_n(S)$ decisión óptima sobre el problema de casarse o no casarse dado que el tipo de novia actual es S y hay n novias por considerar.

Relación de recursión.

Llamemos $P_n(S)$ la probabilidad de que la n novia sea del tipo S y C_s el valor relativo asignado a las novias.

El valor esperado cuando hay n-1 novias por venir es $\sum_t P_{n-1}(t) f_{n-1}(t)$, es decir la suma del producto de la probabilidad de que en la etapa n-1 la novia sea de tipo t multiplicado por el mejor valor del objetivo dado que la novia es de tipo t con n-1 novias por considerar (incluyendo la actual novia cuyo tipo es t).

Por lo tanto:

$$fn(S) = \max_{t} \{C_s, P_{n-1}(t) f_{n-1}(t)\}$$

$$fo(S) \equiv 0$$

METODO DE SOLUCION

1. Problema de la asignación de proyectos.

Cuando solamente hay un proyecto remanente la solución es inmediata

$$f_1(s) = p_1(s)$$

s	$f_1(s)$	$j_1(s)$
0	0	0
1	10	1
2	13	2
3	14	3
4	15	4
5	16	5

n=2

Asignación	Asignación Proyecto 2	Asignación proyectos remanentes	Utilidad Proyecto 2	Utilidad proyectos remanentes	Total	Decisión óptima
S	x	s-x	$p_2(x)$	$f_1(s-x)$		$J_2(s)$
0	0	0	0	0	0	0
1	0	1	0	10	10	0
	1	0	5	5	0	-
2	0	2	0	13	13	-
	1	1	5	10	15	1
	2	0	7	0	7	-
3	0	3	0	14	14	-
	1	2	5	13	18	1
	2	1	7	10	17	-
	3	0	9	0	9	-
4	0	4	0	15	15	-
	1	3	5	14	19	-
	2	2	7	13	20	-
	3	1	9	10	21	3
	4	0	12	0	12	-
5	0	5	0	16	16	-
	1	4	5	15	20	-
	2	3	7	14	21	-
	3	2	9	13	22	3
	4	1	12	10	22	-
	5	0	15	0	15	-

n=3

S	x	s-x	$p_3(x)$	$f_2(s-x)$	Total	$j_3(s)$
0	0	0	0	0	0	0
1	0	1	0	10	10	0
	1	0	0	0	0	-
2	0	2	0	15	15	0
	1	1	0	10	10	-
	2	0	8	0	8	-
3	0	3	0	18	18	0
	1	2	0	15	15	-
	2	1	8	10	18	-
	3	0	16	0	16	-
4	0	4	0	21	21	-
	1	3	0	18	18	-
	2	2	8	15	23	-
	3	1	16	10	26	3
	4	0	20	0	20	-
5	0	5	0	22	22	-
	1	4	0	21	21	-
	2	3	8	18	26	-
	3	2	16	15	31	3
	4	1	20	10	30	-
	5	0	21	0	21	-

n=4

S	x	n-x	$p_4(x)$	$f_3(s-x)$	Total	$j_4(x)$
0	0	0	0	0	0	0
1	0	1	0	10	10	0
	1	0	6	0	6	
2	0	2	0	15	15	-
	1	1	6	10	16	1
	2	0	9	0	9	-
3	0	3	0	18	18	-
	1	2	6	15	21	1
	2	1	9	10	19	-
	3	0	12	0	12	-
4	0	4	0	26	26	0
	1	3	6	18	24	-
	2	2	9	15	24	-
	3	1	12	10	22	-
	4	0	14	0	14	-
5	0	5	0	31	31	-
	1	4	6	26	32	1
	2	3	9	18	27	-
	3	2	12	15	27	-
	4	1	14	10	24	-
	5	0	14	0	14	-

n=5

S	x	n-x	$p_5(x)$	$f_4(s-x)$	total	$J_5(x)$
0	0	0	0	0	0	0
1	0	1	0	10	10	0
	1	0	8	0	8	-
2	0	2	0	16	16	-
	1	1	8	10	18	1
	2	0	12	0	12	-
3	0	3	0	21	21	-
	1	2	8	16	24	1
	2	1	12	10	22	-
	3	0	14	0	14	-
4	0	4	0	26	26	-
	1	3	8	21	29	1
	2	2	12	16	28	-
	3	1	14	10	24	-
	4	0	15	0	15	-
5	0	5	0	32	32	-
	1	4	8	26	34	1
	2	3	12	21	33	-
	3	2	14	16	30	-
	4	1	15	10	25	-
	5	0	15	0	15	-

Solución óptima

$f_5(5) = 34$
$J_5(5) = 1$
$J_4(4) = 0$
$J_3(4) = 3$
$J_2(1) = 0$
$J_1(1) = 1$

2. Problema de la diligencia

Para n=1 la solución es inmediata

$f_1(14) = 6$ $\qquad$ $J_1(14) = 16$
$f_1(15) = 4$ $\qquad$ $j_1(15) = 16$

n=2

Origen	Destino intermedio	Costo	Resto del viaje	Total	Decisión
S	t	C_{st}	$f_1(t)$		$j_2(s)$
12	16	6	0	6	16
13	14	2	6	8	14

n=3

S	t	C_{st}	$f_2(t)$	Total	$J_3(s)$
9	12	3	6	9	12
	13	3	8	11	-
10	14	4	6	10	14
11	14	13	8	21	-
	15	10	4	14	15

n=4

S	t	C_{st}	$f_3(t)$	Total	$J_4(s)$
6	9	8	9	17	9
7	9	6	9	15	9
	10	5	10	15	
8	10	9	10	23	-
	11	6	14	20	11

n=5

S	t	C_{st}	$f_4(t)$	Total	$J_5(t)$
2	6	10	17	27	6
3	6	4	17	21	6
4	7	7	15	22	7
5	8	3	20	23	8

Solución óptima:

$f_6(1) = 21$
$J_6(1) = 3$
$J_5(3) = 6$
$J_4(6) = 9$
$J_3(9) = 12$
$J_2(12) = 16$

En variadas ocasiones un problema puede resolverse usando técnicas diferentes. Así, el problema de la diligencia puede resolverse empleado programación lineal como sigue:

Sea $X_{st} = \begin{cases} 0 & \text{si el camión no pasa por la ruta s a t} \\ 1 & \text{" " " " pasa " " " " " " } \end{cases}$

El problema es

$$\text{Min} \sum_{\text{arcos}} C_{st} X_{st}$$

Sujeto a

$$X_{12}+X_{13}+X_{14}+X_{15} = 1$$

$$X_{12} - X_{26} = 0$$

$$X_{13} - X_{36} = 0$$

$$X_{14}-X_{47} = 0$$

$$X_{15}-X_{58} = 0$$

$$X_{26}+X_{36}-X_{69} = 0$$

$$X_{47}-X_{79}-X_{710} = 0$$

$$X_{58}-X_{810}-X_{811} = 0$$

$$X_{69}+X_{79}-X_{912}-X_{913} = 0$$

$$X_{710}+X_{810}-X_{1014} = 0$$

$$X_{811}-X_{1114}-X_{1115} = 0$$

$$X_{912}-X_{1216} = 0$$

$$X_{913} -X_{1314}=0$$

$$X_{1314}+X_{1014} + X_{1114}-X_{1416} = 0$$

$$X_{1115}-X_{1516} = 0$$

$$X_{1216}+X_{1416}+X_{1516} = 1$$

conservación de flujo

$$X_{st} \geq 0$$

Este es un modelo de transporte cuya solución con programación lineal es entera por lo que podemos ignorar las restricciones

$$X_{st} = \{0 \text{ ó } 1\}$$

La solución que obtenemos por programación lineal nos da la ruta óptima de 1 a 16. La solución que se obtiene con programación dinámica no sólo nos da esta solución, sino las rutas óptimas de cualquier punto a la Ciudad de México. La solución más completa se obtiene pagando el precio de un número mayor de operaciones.

3. Problema del matrimonio

Llamemos $\text{suma}_1 = \sum_t P_i(t) f_1(t)$

Cuando sólo queda una novia la decisión es casarse.

n=1

S	$f_1(s)$	$p_1(s)$	Total
E	40	.1	4
B	30	.2	6
R	20	.3	6
N	10	.4	4
			Suma = 20

O sea, el valor esperado cuando falta una novia por ver es 20.

n=2

S	C_s	$Suma_1$	$f_2(s)$	$p_2(s)$	Total	Decisión
E	40	20	40	.2	8	Casarse
B	30	20	30	.2	6	Casarse
R	20	20	20	.3	6	Casarse o continuar
N	10	20	20	.3	6	Continuar
					$Suma_2 = 26$	

n=3

S	C_s	$Suma_2$	$f_2(s)$	$p_2(s)$	Total	Decisión
E	40	26	40	.3	12	Casarse
B	30	26	30	.3	9	Casarse
R	20	26	26	.2	5.2	Continuar
N	10	26	26	.2	5.2	Continuar
					$Suma_3 = 31.4$	

n=4

S	C_s	$Suma_3$	$f_3(s)$	$p_3(s)$	Total	Decisión
E	40	31.4	40	.1	4	Casarse
B	30	31.4	31.4	.3	9.42	Continuar
R	20	31.4	31.4	.3	9.42	Continuar
N	10	31.4	31.4	.3	9.42	Continuar
					$Suma_4 = 32.26$	

La interpretación de $suma_1$ es la siguiente:

Cuando don Nicandro tiene i novias potenciales por venir en promedio, el tendrá una con valor $suma_i$. Así, al empezar su vida en promedio él se casará con una mujer tipo B(32.26). Cuando sólo le falta conocer una, en promedio se casará con una tipo R(20).

REFERENCIAS

1. Hillier, F.S., Lieberman, G.J., Introduction to Operations Research, Holden-Day, Inc., 1967.

2. Wagner, H.M., Principles of Operations Research, Prentice-Hall, Inc., 1969.

3. Bellman, R.E., Dynamic Programming, Princeton University Press, 1957.

4. Bellman, R.E., Dreyfus, S.E., Applied Dynamic Programming, Princeton University Press, 1962.

T R A N S P O R T E

METODO DE SOLUCION (MINIMIZACION)

PROBLEMAS BALANCEADOS NO DEGENERADOS

(0) Construir una solución factible básica inicial. Ir a (2).

(1) Construir una nueva solución factible:

a) Incrementar X_{rs} en θ

b) "Propagar" $\pm\ \theta$ para conservar factibilidad

c) Dar el mayor valor posible a θ (lo que anulará a una de las variables, antes distinta de cero).

(2) Tabla de seudo-costos:

a) Para $X_{ij} \neq 0$ encontrar u_i, v_j tales que

$$\bar{C}_{ij} = u_i + v_j = C_{ij} \quad (i = 1, 2, .., m; \quad j = 1, 2, .., n)$$

(en caso que no fuera posible ésto, una de las $X_{ij} \neq 0$ puede tratarse como si fuera cero:)

b) Para $X_{ij} = 0$ completar la tabla de seudo-costos

$$\bar{C}_{ij} = u_i + v_j .$$

(3) Encontrar $d_{rs} = \underset{ij}{\text{Min}}\left\{C_{ij} - \bar{C}_{ij}\right\}$ Se presentan dos casos:

1) $d_{rs} \geq 0$, 2) $d_{rs} < 0$

en el 1) la solución factible actual es óptima y por tanto se termina el proceso de cálculo;

en el 2) no se ha llegado al óptimo (por cada unidad que se aumente X_{rs} el costo disminuirá en $|\, d_{rs} \,|$). Ir a (1).

-TRANSPORTE-
EJEMPLO No 1

Destino

Origen	1	2	3	4	Recursos
1	19	30	50	10	7
2	70	30	40	60	9
3	40	8	70	20	18
Demanda	5	8	7	14	

C_{ij}

Tabla de costos y requerimientos (Datos)

Solución factible básica inicial ($m+n-1=3+4-1=6$ variables x_{ij} distintas de cero) en este caso por la regla de la "esquina noroeste". Los cuadros vacios representan valores cero.
Costo inicial: $5(19)+2(30)+6(30)+3(40)+4(70)+14(20)=\underline{1015}$

1º

	1	2	3	4	
1	5	2			7
2		6	3		9
3			4	14	18
	5	8	7	14	

	1	2	3	4	u_i
1	(19)	(30)	40	-10	19
2	19	(30)	(40)	-10	19
3	49	60	(70)	(20)	49
v_j	0	11	21	-29	

	1	2	3	4
1	0	0	10	20
2	51	0	0	70
3	-9	-52	0	0

La posición del más negativo determina dónde hay que aumentar intercambios para la siguiente solución factible; y al tratar de hacer este aumento lo máximo posible (en θ 0), resulta:
1°estamos obligados de modificar algunas variables en $\pm\ \theta$
2°a θ la determina la variable que limita su aumento

2º

	1	2	3	4	
1	5	2			7
2		$6-\theta$ (2)	$3-\theta$ (7)		9
3		θ (4)	$4-\theta$ (0)	(14)	18
	5	8	7	14	$\theta=4$

	1	2	3	4	u_i
1	(19)	(30)	40	42	19
2	19	(30)	(40)	42	19
3	-3	(8)	18	(20)	-3
v_j	0	11	21	23	

	1	2	3	4
1				-32
2				
3				

Solo anotamos los negativos

3º

	1	2	3	4	
1		5	$2-\theta$	θ 2	7
2			2	7	9
3		$4+\theta$	6	$14-\theta$ 12	18
	5	8	7	14	$\theta=2$

	1	2	3	4	u_i
1	(19)	-2	8	(10)	19
2	51	(30)	(40)	42	51
3	29	(8)	18	(20)	29
v_j	0	-21	-11	-9	

	1	2	3	4
1				
2				
3				

No hay negativos .°. la solución factible es óptima.

Costo: $5(19)+2(30)+6(8)+7(40)+2(10)+12(20)=\underline{743}$

(1) Solución factible (2) Sol. de Costos (3) $C_{ij} - \bar{C}_{ij}$

$\bar{C}_{ij}$

EJEMPLO No. 2

Destino — Origen — Recursos — Demanda — C_{ij}

Origen \ Destino	1	2	3	4	Recursos
1	19	30	50	10	7
2	70	30	40	60	9
3	40	8	70	20	18
Demanda	5	8	7	14	

Solución factible básica inicial mejorada (Pag. 146 del texto). Los paréntesis en las variables indican el orden en que se fueron encontrando; y los paréntesis en las restricciones, el orden en que se fueron cumpliendo.

1º

(1) solución factible

	1	2	3	4	
1				7	(2) 7
2	(6) 2		(4) 7		(6) 9
3	(5) 3	(1) 8		(3) 7	(5) 18
	5	(1) 8	(4) 7	(3) 14	

(2) seudo costos $\bar{C}_{ij}$

	1	2	3	4	u_i
1	30	-2	0	10	30
2	70	38	40	50	70
3	40	8	10	20	40
v_j	0	-32	-30	-20	

(3) $C_{ij} - \bar{C}_{ij}$

	1	2	3	4
1	-11			
2		-8		
3				

$C_{11}-C_{14}+C_{34}-C_{31}=C_{11}(u_1+v_4)+(u_3+v_4)$
$-(u_3+v_1)= C_{11}-(u_1+v_1)= -30=-11$

.˙. por cada unidad que aumente x_{11}, se ahorran 11 unidades de costo

2º

(1) solución factible

	1	2	3	4	
1	θ 3			7-θ 4	7
2	2		7		9
3	3-θ	8		7+θ 10	18
	5	8	7	14	θ=3

(2) seudo costos $\bar{C}_{ij}$

	1	2	3	4	u_i
1	19	-2	-11	10	19
2	70	49	40	61	70
3	29	8	-1	20	29
v_j	0	-21	-30	-9	

(3) $C_{ij} - \bar{C}_{ij}$

	1	2	3	4
1				
2		-19		-1
3				

Sólo se pudieron ahorrar 11x3 = 33 unidades de costo.

$C_{22}-C_{21}+C_{11}-C_{14}+C_{34}-C_{32}=C_{22}-(u_2+v_2)=$
$30-49=-19$

.˙. por cada unidad que aumente x_{22}, se ahorran 19 unidades de costo

3º

(1) solución factible

	1	2	3	4	
1	3+θ 5			4-θ 2	7
2	2-θ	θ 2	7		9
3		8 θ 6		10+θ 12	18
	5	8	7	14	θ=2

(2) seudo costos $\bar{C}_{ij}$

	1	2	3	4	u_i
1	19	-2	8	10	19
2	51	30	40	42	51
3	29	8	18	20	29
v_j	0	-21	-11	-9	

(3) $C_{ij} - \bar{C}_{ij}$

	1	2	3	4
1				
2				
3				

Sólo se pudieron ahorrar 19x2=38 unidades de costo respecto a la solución anterior.

No hay negativos .˙. no se puede mejorar esta solución

(1) solución factible (2) seudo costos $\bar{C}_{ij}$ (3) $C_{ij} - \bar{C}_{ij}$

-TRANSPORTE

PROBLEMA DEGENERADO

Tabla de costos y requerimientos (datos) →

	1	2	3	
1	7	3	4	2
2	2	1	3	3
3	3	4	6	5
	4	1	5	

La solución factible básica inicial es degenerada: cuatro $X_{ij} \neq 0$ en vez de $m + n - 1 = 3 + 3 - 1 = 5$.

Con $\epsilon << 1$ perturbamos al problema

Solución factible básica inicial

1º

	1	2	3	
1	2			2
2	2	1	ϵ	$3+\epsilon$
3			5	5
	4	1	5 + 6	

	1	2	3	u_i
1	⑦	6	8	0
2	②	①	③	-5
3	5	4	⑥	-2
v_j	7	6	8	

	1	2	3
1		-3	-4
2			
3	-2		

2º

	1	2	3	
1	$2+\epsilon$		ϵ	2
2	$2+\epsilon$	1		$3+\epsilon$
3			5	5
	4	1	$5+\epsilon$	

	1	2	3	u_i
1	⑦	6	④	0
2	②	①	-1	-5
3	9	8	⑥	2
v_j	7	6	4	

	1	2	3
1		-3	
2			
3	-6	-4	

3º

	1	2	3	
1	$2-\epsilon-\theta$		② $\epsilon+\theta$	2
2	$2+\epsilon$	①		$3+\epsilon$
3	θ $2-\epsilon$		$5-\theta$ $3+\epsilon$	5
	4	1	$5+\epsilon$	$\theta=2-\epsilon$

	1	2	3	u_i
1	1	0	④	0
2	②	①	⑤	1
3	③	2	⑥	2
v_j	1	0	4	

	1	2	3
1			
2			-2
3			

4º

	1	2	3	
1			②	2
2	$2+\epsilon-\theta$	①	θ $2+\epsilon$	$3+\epsilon$
3	$2-\epsilon+\theta$ 4		$3+\epsilon-\theta$ ①	5
	4	1	5 + 6	$\theta=2+\epsilon$

	1	2	3	u_i
1	1	2	④	0
2	0	①	③	-1
3	③	4	⑥	-2
v_j	1	2	4	

	1	2	3
1			
2			
3			

← No hay negativos ∴ la solución es óptima haciendo $\epsilon = 0$

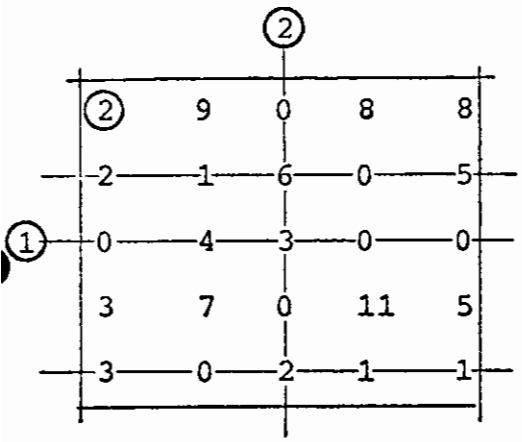
	1	2	3	
1			2	2
2		1	2	3
3	4		1	5
	4	1	5	

← Solución óptima al problema degenerado

PROBLEMA DE ASIGNACION

	1	2	3	4	5	restando
1	11	17	8	16	20	8
2	9	7	12	6	15	6
3	13	16	15	12	16	12
4	21	24	17	28	26	21
5	14	10	12	11	15	10

3	9	0	8	12	
3	1	6	0	9	
1	4	3	0	4	
4	7	0	11	9	
4	0	2	1	5	
restando 1	1	0	0	4	



	1	2	3	4	5
1	[0]	7	0	6	6
2	2	1	8	[0]	5
3	0	4	5	0	[0]
4	1	5	[0]	9	3
5	3	[0]	4	1	1

SOLUCION OPTIMA:

asignar

renglón	con columna	costo
1	1	11
2	4	6
3	5	16
4	3	24
5	2	10
costo total		67

ALGORITMO DEL TRANSPORTE

1. DESCRIPCION

El problema general del transporte, consiste en que, dado un producto homogéneo, este debe ser enviado en las cantidades A_1, A_2, ..., Am desde cada uno de los m puntos de origen y recibirse en las cantidades B_1, B_2, ..., Bn en cada uno de los n destinos. El costo de transportar una unidad desde el origen i hasta el destino j es C_{ij} y es conocido para todas las combinaciones X_{ij} que deben ser transportadas a través de todas las rutas (i,j) con objeto de minimizar el costo total del transporte.

El problema anterior se ha resuelto a través del algoritmo número 293 de Comunication of A. C. M. (Vol. 9, N°12, Dic. 1966). El programa fué escrito en lenguaje ALGØL y está constituido básicamente por un PRØCEDURE llamado - TRANSPØRTE cuyos argumentos se explican a continuación. La versión en FORTRAN que se presenta es una traducción del original en ALGOL.

TRANSPØRTE (N, M, A, C, INF, X, KW)

donde:

M Número de cantidades disponibles (orígenes).

N Número de cantidades requeridas (destinos).

A(1:M) Vector con las cantidades disponibles desde cada origen.

B(1:N) Vector con las cantidades requeridas en cada destino.

C(MxN) Matriz de costos unitarios, para todas las combinaciones origen-destino.

INF Número entero positivo (el más grande que acepte la computadora).

X(MxN) Matriz de flujos; contiene las cantidades que deben ser transportadas por todas las rutas i, j (respuesta del algoritmo).

WK Costo total óptimo (respuesta).

NOTA: Durante la ejecución del programa el contenido de los arreglos A, B y C es alterado. Además la condición

$$A_i = B_i \quad \text{para} \quad \begin{matrix} i = 1, 2, \ldots, M \\ j = 1, 2, \ldots, N \end{matrix}$$

debe cumplirse.

2. REFERENCIAS

1. S. I. Gass, "Programación Lineal", pp. 243-267, CECSA (1961).
2. I. S. Hiller, G. J. Lieberman, "Introduction to Operation Research", pp. 172-193, Holden Day Inc. (1967).
3. M. Simonnard, "Linear Programming", Cap. 11, 12, 13, 14 y 15, Prentice Hall (1962).

El algoritmo como se presenta a continuación, está acompañado de instrucciones adicionales que se encargan de:

1. Leer los datos
2. Hacer uso del PROCEDURE
3. Imprimir resultados

A continuación se presenta un instructivo de uso del programa, esto es, la forma como se preparan los datos.

3. PREPARACION DE DATOS

Para resolver el problema general del transporte son necesarios los siguientes datos. Su formato y las columnas, que ocupa en la tarjeta se dan entre paréntesis.

1. Título del problema
2. Número de cantidades disponibles
   (formato I5, columnas 1 a 5)
   Número de cantidades requeridas
   (formato I5, columnas 6 a 10)
3. Vector de cantidades disponibles
   (formato 10I8, columnas 1 a 80), en tantas tarjetas como sea necesario.
4. Vector de cantidades requeridas
   (formato 10I8, columnas 1 a 80), en tantas tarjetas como sea necesario.
5. Matriz de costos unitarios. Sus datos deben proporcionarse por renglones.
   (formato 10F.2 columnas 1 a 80)

Cada una de las tarjetas debe contener 10 datos si es necesario (excepto la última tarjeta del renglón que puede contener menos); esto es, cada renglón debe empezar en el primer campo de una tarjeta.

3.1 Ejemplo: (Corresponde al primero del inciso 5.)

Supóngase un problema donde se tienen tres fuentes de producción (M = 3) y cuatro destinos (N = 4).

Las cantidades de que se dispone son:

25, 24, 50.

y las requeridas en los destinos son:

15, 20, 30, 35.

La matriz de costos unitarios para las diferentes combinaciones de i y j es:

i/j	1	2	3	4
1	10	5	6	7
2	8	2	7	6
3	9	3	4	8

Las tarjetas de datos son las siguientes:

PROBLEMA DE EXPOSICION

DATOS NUMI

SERVICE BUREAU

IBM DE MEXICO, S A

SISTEMA IBM 7070

PROBLEMA DE EXPOSICION

SOLUCION GENERAL PROBLEMA DEL TRANSPORTE

VECTOR DE CANTIDADES DISPONIBLES

25.000	25.000	50.000

VECTOR DE CANTIDADES REQUERIDAS

15.000	20.000	30.000	35.000

MATRIZ DE COSTOS UNITARIOS

RENGLON 1			
100.000	50.000	60.000	
RENGLON 2			
80.000	20.000	70.000	
RENGLON 3			
90.000	30.000	40.000	
RENGLON 4			
0.000	0.000	0.000	

MATRIZ DE FLUJOS

RENGLON 1			
0.000	0.000	0.000	25.000
RENGLON 2			
5.000	10.000	0.000	10.000
RENGLON 3			
10.000	10.000	30.000	0.000

COSTO TOTAL OPTIMO = 0.53500E 04

B 6 7 0 0 F O R T R A N C O M P I L A T I O N M A R K I

```
$ SET SINGLE ITST
$ SET SINGLE  BCD
FILE  5=NUM1, UNIT=READER
FILE  6=NUM2, UNIT=PRINT
      INTEGER Z,Y,S1,S2,D
      DIMENSION A(100),B(100),C(100,100),X(100,100),COM(80)
C     ALGORITMO DEL TRANSPORTE
C     DESCRIPCION DE LAS VARIABLES QUE INTERVIENEN
C     M          =NUMERO DE CANTIDADES DISPONIBLES
C     N          =NUMERO DE CANTIDADES REQUERIDAS
C     A(1,M)     =VECTOR DE CANTIDADES DISPONIBLES
C     B(1,N)     =VECTOR DE CANTIDADES REQUERIDAS
C     C(M X N)=MATRIZ DE COSTOS UNITARIOS
C     X(MXN)     =MATRIZ DE FLUJOS  (RESPUESTA DEL ALGORITMO)
C     INF        =ENTERO POSITIVO MAS GRANDE QUE PERMITA USAR
C                 LA COMPUTADORA
C     KW    =COSTO TOTAL OPTIMO    (RESPUESTA)
C     LOS VECTORES 'A' Y 'B' , ASI COMO LA MATRIZ 'C' SON
C     ALTERADOS AL EJECUTARSE EL ALGORITMO
C     LA SUMA DE LOS ELEMENTOS DE LOS VECTORES 'A' Y 'B'
C     DEBEN SER IGUALES
C     REFERENCIAS)
C     ALGORITMO DERIVADO DEL NO. 293 DE) COMMUNICATIONS DE
C     THE A.C.M.    VOL  9/NO. 12/DICIEMBRE 1966  PP. 869-871
C     G. HADLEY,LINEAR PROGRAMMING,READING LONDON,1962 P 351
  200 READ (5,500,END=900) COM
      WRITE (6,128)
      WRITE (6,500)  COM
      READ(5,120) M,N
  500 FORMAT(80A1)
  120 FORMAT (I5,I5)
      READ(5, 7 ) (A(I),I=1,M)
      READ(5, 7 ) (B(J),J=1,N)
  121 FORMAT(8I10)
C 122 FORMAT(8I10)
  122 FORMAT(10F8.2)
  123 FORMAT ( 8F10.3 )
      DO 1969 I= 1 , M
 1969 READ(5,122)  (C(I,J),J=1,N)
      S1 = 0
      S2 = 0
      D = 0
      DO 124 I=1,M
  124 S1 = S1 + A(I)
      DO 125 J=1,N
  125 S2 = S2 + B(J)
      D = S1 - S2
      IF(0-D) 250,300,260
  250 N = N+ 1
      M = M + 1
      B(N) = D
      A(M) = 0
      GO TO  251
  260 N = N+ 1
      M = M + 1
      A(M) = - D
```

```
      B(N) = 0
  251 DO 126 I =1,M
  126 C(I,N) = 0
      DO 127 I=1,N
  127 C(M,J) = 0
      C(M,N)=0
  300 CONTINUE
    1 FORMAT (    33H VECTOR DE CANTIDADES DISPONIBLES )
    2 FORMAT ( /,    32H VECTOR DE CANTIDADES REQUERIDAS )
    3 FORMAT (//,    27H MATRIZ DE COSTOS UNITARIOS )
    4 FORMAT (///,    17H MATRIZ DE FLUJOS )
    5  FORMAT(///,    22H COSTO TOTAL OPTIMO =  ,E12.5)
    6 FORMAT (///,44H SOLUCION GENERAL    PROBLEMA DEL TRANSPORTE ,///)
    7 FORMAT (8I10)
    8 FORMAT (2I3,I10)
    9 FORMAT (10F10.3,/)
   10 FORMAT (15I8)
   11 FORMAT (8H RENGLON,I4)
      INF =5A0755813870
  128 FORMAT (1H1)
      WRITE(6,6)
      WRITE(6,1)
      WRITE (6,9)(A(I),I=1,M)
      WRITE (6,2)
      WRITE (6,9)  (B(J),J=1,N)
      WRITE (6,3)
      DO 1000 I=1,N
      WRITE (6,11) I
      WRITE (6,1010)(C(I,J),J=1,M)
 1000 CONTINUE
      CALL TRANSP      (N,M,A,B,C,INF,X,WK)
 1010 FORMAT (10F10.3)
      WRITE(6,4)
      DO 1002 I=1,M
      WRITE (6,11) I
      WRITE (6,9) (X(I,J),J=1,N)
 1002 CONTINUE
      WRITE (6,5) WK
      GO TO 200
  900 CALL EXIT
      END
```

```
      SUBROUTINE TRANSP     (N,M,A,B,C,INF,X,KW)
      COMMON ISO, ISVO
      LOGICAL ZG
      REAL KW
      DIMENSION A(100),B(100),C(100,100),X(100,100)
      INTEGER S,U,V,T,GO,H,P,CIJ,XIJ,AI,BJ,O
      DIMENSION G(100),LISTU(100),R(100),LISTV(100),NLV(100)
      DIMENSION NLS(100),NL(999),ISV(100),LIST(100),LS(999)
C     EN LA DECLARACION DO U... DESPUES DEL PASO 33 (S33),SE OPERA CON TO
C     DOS LOS PARES I,J DE C(I,J)=0. EN UN TIEMPO RAPIDO EL ARREGLO (NL)
C     INSPECCIONA ESTOS CEROS,LOS INDICES J DE CEROS EN LA FILA I SON
C     GUARDADOS EN NL(I-1)*N+1)....NL(NLV(I). EN LA DECLARACION DO V...
C     DESPUES DEL PASO 33 OPERA CON TODOS LOS PARES I,J CON X(I,J)=0 Y
C     C(I,J)=0),IS INSPECCIONA ESTOS CEROS INICIALES,LOS INDICES I DE CE
C     ROS ESENCIALES DE LA COLUMNA J SON GUARDADOS EN LS(ISV(J-1)+1)....
C     LS(LSV(J)).EL PROCEDIMIENTO NI AGREGA A LA LISTA LS,EL PROCEDIMIEN
C     TO TUO SACA DE LA LISTA LS UN CERO ESENCIAL EN POSICION I,J
      O = N
      DO 58 I=1,M
      DO 58 J=1,N
   58 X(I,J) =0
      DO 60 I=1,M
   60 NLV(I) =(I-1)*N
      LSVO = 0
      DO 61 J=1,N
      LISTV(J) = 1
   61 LSV (J) = 0
 1000 GO = 0
      KW = GO
      DO 62 I=1,M
      H = INF
      DO 65 J=1,N
      IF (C(I,J).LT. H)  H = C(I,J)
   65 CONTINUE
      DO 63 J=1,N
      C(I,J) = C(I,J) - H
      CIJ = C(I,J)
      IF(CIJ.EQ.0) GO TO 64
      GO TO 63
   64 LISTV(J) = 0
      NLV(I) = NLV(I) + 1
      NLVI = NLV(I)
      NL(NLVI) = J
   63 CONTINUE
      KW = H * A(I) + KW
   62 CONTINUE
      DO 1020 J=1,N
      IF (LISTV(J).EQ.0) GO TO 1020
      H = INF
      DO 67 I=1,M
      IF(C(I,J). LT.H)   H = C(I,J)
   67 CONTINUE
      DO 68 I=1,M
      C(I,J) = C(I,J) - H
      CIJ = C(I,J)
      IF(CIJ.EQ.0) GO TO 69
      GO TO 68
   69 NLV(I) = NLV(I) + 1
      NLVI = NLV(I)
```

```
      NI(NLVT) = J
   68 CONTINUE
      KW = H * B(J) + KW
 1020 CONTINUE
C     EN EL INCREMENTO DE 1 LA REDUCCION USUAL DE LA MATRIZ DE COSTOS
C     SE LLEVA A CABO,PROBLEMA DUAL,LOS CEROS SON LISTADOS EN NI
 1040 DO 70 I=1,M
      AI= A(I)
      NLVI = NLV(I)
      HKT3 = (I-1)*N +1
      DO 1030 U=HKT3,NLVI
      IF (AI.EQ.0) GO TO 1050
      J = NI(U)
      BJ = B(J)
      IF (BJ.EQ.0) GO TO 1030
      IF(AI.LT.BJ) GO TO 73
      X(I,J) = BJ
      GO TO 74
   73 X(I,J) = AI
   74 H = X(I,J)
      AI = AI - H
      B(J) = BJ - H
      CALL NI (J,0,LSVJ,LSV,IS,I)
 1030 CONTINUE
 1050 A(I) = AI
      GO = GO + AI
   70 CONTINUE
 1060 IF (GO.EQ.0) GO TO 1016
 1070 DO 75 I=1,N
   75 B(I) = 0
      K = 0
      DO 76 I=1,M
      IF(A(I).NE.0)  GO TO 77
      GO TO 78
   77 K = K +1
      LISTU(K) = I
      G(I) = INF
      GO TO 76
   78 G(I)=0
   76 CONTINUE
 1080 I = 0
      DO 79 U=1,K
      I = LISTU(U)
      NLVI = NLV(I)
      HKT5 = (I-1)*N + 1
      DO 1090 S=HKT5,NLVI
      J = NI(S)
      IF (B(J) .NE. 0 ) GO TO 1090
      B(J) = I
      L = L + 1
      LISTV(L) = J
      IF(B(J).GT. 0 ) GO TO 1013
 1090 CONTINUE
   79 CONTINUE
      IF (L.EQ.0) GO TO 1017
      K = 0
      DO 81 V=1,L
      J = LISTV(V)
      LSVJ = LSV(J)
      IF(J.EQ.1) GO TO 999
```

```
      HKT8 = LSV(J-1) + 1
      GO TO 777
  999 HKT8=1 +LSVO
  777 DO 82 S=HKT8,LSVJ
      I = LS(S)
      IF(I.GE.1) GO TO 88
 8000 FORMAT(I10)
      GO TO 82
   88 CONTINUE
      IF(G(I).EQ.0) GO TO 84
      GO TO 82
   84 G(I) = J
      K = K +1
      LISTU(K) = I
   82 CONTINUE
   81 CONTINUE
      IF(K.EQ.0) GO TO 1017
      GO TO 1080
 1013 H = B(J)
      P = J
 1010 I = R(J)
      J = G(I)
      IF(J.EQ.INF) GO TO 86
      GO TO 85
   86 IF (A(I).LT. H ) H = A(I)
      GO TO 1011
   85 IF ( X(I,J).LT. H ) H = X(I,J)
      GO TO 1010
 1011 J = P
      B(J) = B(J)-H
      A(I) = A(I)-H
      GO = GO - H
 1012 I = R(I)
      XIJ = X(I,J)
      X(I,J) = XIJ + H
      IF(XIJ.EQ.0) GO TO 90
      GO TO 89
   90 CALL NT (J,Q,LSVJ,LSV,LS,I)
   89 J=G(I)
      IF(J.EQ.INF) GO TO 1060
      X(I,J) = X(I,J)-H
      XIJ = X(I,J)
      IF(XIJ.EQ.0) GO TO 92
      GO TO 1012
   92 CALL TUO(J,Q,LSVJ,LSV,LS,I)
      GO TO 1012
 1017 K = 0
      L = N + 1
      DO 93 I=1,N
      IF(R(I).EQ.0) GO TO 95
      GO TO 94
   95 K = K + 1
      LISTV(K) = I
      GO TO 93
   94 L = L - 1
      LISTV(L) = I
   93 CONTINUE
      H = INF
      DO 1018 I=1,M
      IF(G(I).EQ.0 ) GO TO 1018
```

```
//
      DO 97 S=1,K
      J = LISTV(S)
      IF (C(I,J).LT.H)  H = C(I,J)
   97 CONTINUE
 1018 CONTINUE
      DO 99 I=1,N
      IF (G(I)) 400,500,400
  400 ZG = .TRUE.
      GO TO 600
  500 ZG = .FALSE.
  600 NLVI = (I-1) * N
      DO 100 S=1,N
      J= LISTV(S)
      IF (ZG) GO TO 700
      C(I,J) = C(I,J) + H
  700 CIJ = C(I,J)
      IF(CIJ.EQ.0) GO TO 221
      GO TO 100
  221 NLVI = NLVI + 1
      NL(NLVI) = J
  100 CONTINUE
      DO 102 S=1,K
      J = LISTV(S)
      IF (ZG) GO TO 222
      GO TO 230
  222 C(I,J) = C(I,J) - H
  230 CIJ = C(I,J)
      IF (CIJ.EQ. 0) GO TO 250
      GO TO 102
  250 NLVI = NLVI + 1
      NL(NLVI) = J
  102 CONTINUE
      NLV(I) = NLVI
   99 CONTINUE
      KW= H*GO + KW
      GO TO 1070
 1016 RETURN
  900 FORMAT(10I10)
      END
```

```
      SUBROUTINE NT (J,Q,LSVJ,LSV,LS,T)
      COMMON LSO, LSVO
      DIMENSION LSV(100),LS(999)
      INTEGER Q,T
      IF(J.EQ.0) LSVJ=LSVO
      IF(J.GT.0) LSVJ=LSV(J)
      IF(LSV(0) .EQ. 0) GO TO 100
      TT=LSV(0)
      MT=LSVJ
      IF(LSVJ.EQ.0) MT=1
      DO 52 M=MT,TT
      T=TT-M+MT
   52 LS(T+1)=LS(T)
      IF(LSVJ.EQ.0)LS(1)=LSO
  100 IF(J.EQ.0) GO TO 80
      K=1
      GO TO 85
   80 K=1
      LSVO=LSVO+1
   85 DO 53 T=K,0
   53 LSV(T)=LSV(T)+1
      LS(LSVJ+1)=T
      RETURN
      END
```

```
      SUBROUTINE TWO (J,A,LSV,I,ISV,IS,T)
      RANMAN ISA, LSVA
      ATMENSION ISV(100),IS(999)
A     NEXT = 1021
A     EX   = 1022
      INTEGFR A,T
      IF(J,EA,A) ISV.I=LSVA
      IF(J,GT,A) ISV.I=LSV(J)
      IF(J,EA,1) IT=ISVA+1
      IF(J ,GT, 1) IT=LSV(J-1)+1
      DO 1021 T=IT,LSVJ
      IF(LS(T),NE,I) GO TO 1021
      S = T
      GO TO 1022
 1021 CONTINUE
 1022 IF(J,EA,A) GO TO DO
      K=1
      GO TO 85
   DO K=1
      ISVA=ISVA+1
   85 DO 56 T=K,A
   56 ISV(T) = LSV(T) + 1
      ISVJ = ISV(A)
      IF(S,EA,A) GO TO 90
      K=S
      GO TO 95
   90 K=1
      LSA=LS(1)
   95 DO  57  T=S,LSVJ
   57 LS(T)= IS(T+1)
      RETURN
      END
```

INTRODUCCION A LA PROGRAMACION LINEAL

Servio T. Guillén[1]

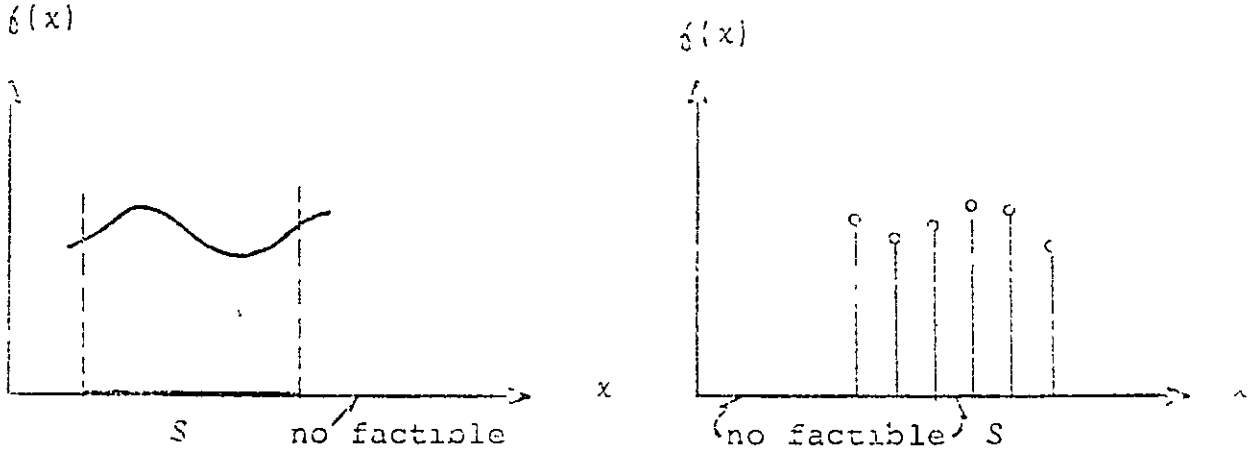
1. Generalidades

Sean $x_1, x_2, \ldots, x_n$ las variables de decisión, que abreviamos con x, S el conjunto de todas las decisiones x que son factibles (aceptables), y $f(\cdot)$ una función real, llamada función objetivo, que evalúa las decisiones x. El problema de la programación matemática es encontrar una decisión x° factible que minimiza[2] la función objetivo $f(\cdot)$; es decir, encontrar x° en S tal que $f(x^\circ) \leq f(x)$ para todo x en S. Este problema se acostumbra plantear así:

$$\min f(x)$$

sujeto a

$$x \text{ en } S$$



[1] Instituto de Ingeniería, UNAM

[2] Maximizar

2. El problema del transporte

Un recurso homogéneo se encuentra disponible en m orígenes 1, 2, ..., m, en cantidades $b_1, b_2, \ldots, b_m$ respectivamente, y se le requiere en n destinos 1, 2, ...n en cantidades $a_1, a_2, \ldots a_n$. El costo por transportar x_{ij} unidades del origen i al destino j es $c_{ij}\ x_{ij}$ donde c_{ij} es el costo unitario correspondiente. Suponiendo que las cantidades totales disponible y requerida son iguales (problema balanceado), encontrar los transportes x_{ij} que hacen mínimo el costo total, de modo que todos los destinos queden satisfechos. Matemáticamente se puede plantear el problema como:

Dados: $c_{ij},\ b_i \geq 0,\ a_j \geq 0,\quad i=1,\ldots,m;\ j=1,\ldots,n$

con $\sum_{i=1}^{m} b_i = \sum_{j=1}^{n} a_j$, encontrar $x_{i,j}$ tal que

$$\min \sum_{j=1}^{n} \sum_{i=1}^{m} c_{ij}\ x_{ij} \qquad 2.1$$

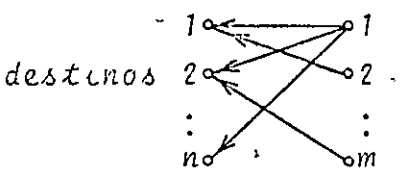
sujeto a

$$\sum_{j=1}^{n} x_{ij} = b_i, \qquad i=1,\ldots,m$$

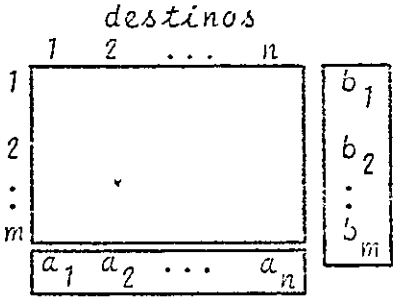
$$\sum_{i=1}^{m} x_{ij} = a_j, \qquad j=1,\ldots,n \qquad 2.2$$

$$x_{ij} \geq 0$$

Este conjunto de restricciones puede manejarse fácilmente en una tabla-matriz en que cada renglón representa un origen y cada columna a un destino. Una solución x_{ij} es factible si y solo si la suma de los elementos del renglón i vale b_i, la suma de los elementos de la columna j vale a_j, y todos los elementos de la matriz son no negativos.



orígenes



Para generar una solución factible básica[1] (a lo más $n+m-1$ transportes distintos de cero) se puede usar la regla "esquina noroeste", que se ilustra con el ejemplo no. 1:

Poner en el espacio (1,1) el mínimo de $(a_1,b_1) = (5,7)$, que es 5. Moverse en el espacio en la dirección del máximo de (a_1,b_1), que es la posición (1,2) (primer renglón, segunda columna). Poner en esta posición el mínimo de $(a_2, b_1-5) = (8,7-5)$, que es 2. Moverse en la dirección del máximo de (a_2,b_1-5), que es la posición (2,2). Poner en esta posición el mínimo de $(a_2-2,b_2)=(8-2,9)$, que es 6. Continuando con este proceso se obtiene una solución factible de $3+4-1=6$ transportes distintos de cero. En el ejemplo no. se muestra una solución factible básica "mejorada" que sigue reglas similares, pero que toma en cuenta la matriz de costos.

---

1) En la parte de programación lineal se justifica este nombre.

3. Ejemplo

Consideremos el problema 1 del apéndice 1, en que de acuerdo a capacidades variables de producción, una fábrica debe cumplir a costo mínimo, sus compromisos de ventas por los próximos cuatro meses etc. Los datos se dan en la siguiente tabla.

MES	VENTAS CONTRATADAS	PRODUCCION MAXIMA	COSTO UNITARIO DE PRODUCCION	COSTO UNITARIO DE ALMACENAMIENTO POR MES
1	20	30	8	1
2	30	50	12	1
3	50	20	11	1
4	40	50	14	1
	140	150		

Si x_α es la producción en el mes α ($\alpha=1, 2, 3, 4$), el problema se puede plantear como

$$\text{minimizar } z = 8x_1+12x_2 + 11x_3 +14x_4 + (x_1 - 20) + (x_1 + x_2 - 50) + x_1+ x_2 - x_3 - 100) = 8x_1 + 14x_2 + 12x_3 + 14x_4 - 170$$

con las restricciones

$x_1 \leq 30$
$x_2 \leq 50$ limitación de
$x_3 \leq 20$ la producción
$x_4 \leq 50$

$x_1 \geq 20$
$x_1 + x_2 \geq 50$ cumplir
$x_1 + x_2 + x_3 \geq 100$ ventas
$x_1 + x_2 + x_3 + x_4 \geq 140$ contratadas

$x_1 \geq 0$
$x_2 \geq 0$ producción no
$x_3 \geq 0$ negativa
$x_4 \geq 0$

Este es un programa lineal (PL) que puede ser resuelto por métodos simples que se verán después, sin embargo, con una representación adecuada, este puede reducirse a un problema de transporte.

Para exhibir una estructura de transporte, se deben identificar cada uno de los elementos:

x_{ij} = cantidad producida en el mes i para vender en el mes j $(i \leq j)$

c_{ij} = costo unitario (producción + almacenamiento) asociado con x_{ij} $(i \leq j)$

a_j = ventas contratadas en el mes j

b_i = capacidad de producción en el mes i

Agregando un destino "ficticio" (el número 5) con costos de transporte cero, y dando un costo de transporte M muy grande para "transportes" no factibles, nuestro problema toma el planeamiento de un problema <u>blanceado</u> de transporte:

	D E S T I N O					
ORIGEN	1	2	3	4	5	RECURSOS
1	8	9	10	11	0	30
2	M	12	13	14	0	50
3	M	M	11	12	0	20
4	M	M	M	14	0	50
DEMANDA	20	30	50	40	10	(150)

4. Algoritmo de transporte (problemas balanceados no degenerados)

(0) Construir una solución factible básica inicial. Ir a (2)

(1) Construir una nueva solución factible:

a) Incrementar x_{rs} en θ

b) "Propagar" $\pm\ \theta$ para conservar factibilidad

c) Dar el mayor valor posible a θ (lo que anulará a una de las variables, antes distinta de cero)

(2) Tabla de seudo-costos:

a) Para $x_{ij} \neq 0$ encontrar u_i, v_j tales que

$$\bar{c}_{ij} = u_i + v_j = c_{ij} \quad (i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n)$$

(en caso que no fuera posible ésto, una de las $x_{ij} \neq 0$ puede tratarse como si fuera cero:)

b) Para $x_{ij} = 0$ completar la tabla de seudo-costos

$$\bar{c}_{ij} = u_i + v_j$$

(3) Encontrar $d_{rs} = \min_{ij} \ c_{ij} - \bar{c}_{ij}$ Se presentan dos casos:

1) $d_{rs} \geq 0$. 2) $d_{rs} < 0$

en el 1) la solución factible actual es óptima y por tanto se termina el proceso de cálculo;

en el 2) no se ha llegado al óptimo (por cada unidad que se aumente x_{rs} el costo disminuirá en d_{rs}). Ir a (1).

-TRANSPORTE-

EJEMPLO No 1

Destino

Origen	1	2	3	4	Recursos
1	19	30	50	10	7
2	70	30	40	60	9
3	40	8	70	20	18
Demanda	5	8	7	14	

C_{ij}

Tabla de costos y requerimientos (Datos)

Solución factible básica inicial ($m+n-1=3+4-1=6$ variables x_{ij} distintas de cero) en este caso por la regla de la 'esquina noroeste'. Los cuadros vacios representan valores cero.

Costo inicial: $5(19)+2(30)+6(30)+3(40)+4(70)+14(20)=\underline{1015}$

1º

	1	2	3	4	
1	5	2			7
2		6	3		9
3			4	14	18
	5	8	7	14	

	1	2	3	4	u_i
1	(19)	(30)	40	-10	19
2	19	(30)	(40)	-10	19
3	49	60	(70)	(20)	49
v_j	0	11	21	-29	

	1	2	3	4
1	0	0	10	20
2	51	0	0	70
3	-9	-52	0	0

La posición del más negativo determina dónde hay que aumentar intercambios para la siguiente solución factible y al tratar de hacer este aumento lo máximo posible (en θ 0), resulta:

1°estamos obligados de modificar algunas variables en $\pm\theta$

2°a θ la determina la variable que limita su aumento

2º

	1	2	3	4	
1	5	2			7
2		$6-\theta$ (2)	$3+\theta$ (7)		9
3		θ (4)	$4-\theta$ (0)	(14)	18
	5	8	7	14	

$\theta=4$ v_j

	1	2	3	4	u_i
1	(19)	(30)	40	42	19
2	19	(30)	(40)	42	19
3	-3	(8)	18	(20)	-3
v_j	0	11	21	23	

	1	2	3	4
1				-32
2				
3				

Solo anotamos los negativos

3º

	1	2	3	4	
1		5	$2-\theta$	θ 2	7
2		2	7		9
3		$4+\theta$ 6		$14-\theta$ 12	18
	5	8	7	14	

$\theta=2$ v_j

	1	2	3	4	u_i
1	(19)	-2	8	(10)	19
2	51	(30)	(40)	42	51
3	29	(8)	18	(20)	29
v_j	0	-21	-11	-9	

	1	2	3	4
1				
2				
3				

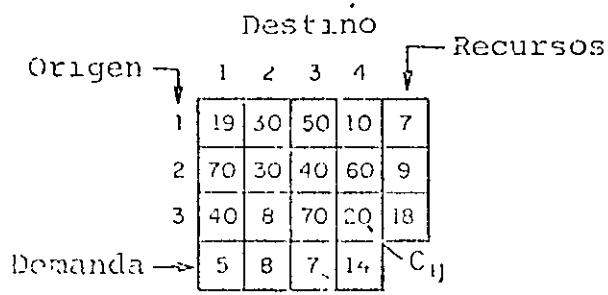
No hay negativos ∴ la solución factible es óptima.

Costo: $5(19)+2(30)+6(8)+7(40)+2(10)+12(20)=\underline{743}$

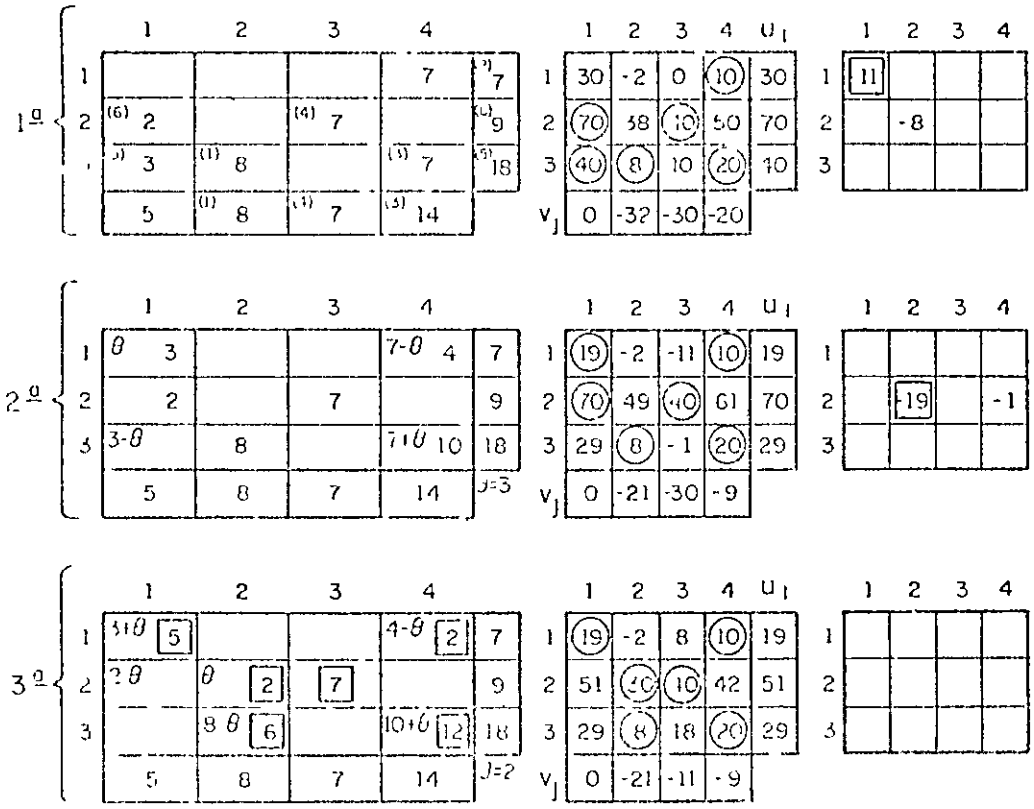
(1) Solución factible (2) Seudo-Costos (3) $C_{ij} - \bar{C}_{ij}$

$\bar{C}_{ij}$

EJEMPLO No. 2



Origen \ Destino	1	2	3	4	Recursos
1	19	30	50	10	7
2	70	30	40	60	9
3	40	8	70	20	18
Demanda	5	8	7	14	



(1) solución factible (2) seudo costos $\bar{C}_{ij}$ (3) $C_{ij} - \bar{C}_{ij}$

Solución factible básica inicial mejorada (Pag. 146 del texto). Los paréntesis en las variables indican el orden en que se fueron encontrando; y los paréntesis en las restricciones, el orden en que se fueron cumplicado.

$C_{11}-C_{14}+C_{34}-C_{31}=C_{11}-(u_1+v_4)+(u_3+v_4)-(u_3+v_1)=C_{11}-(u_1+v_1)=\ +30=-11$

∴ por cada unidad que aumente x_{11}, se ahorran 11 unidades de costo

Sólo se pudieron ahorrar 11x3 = 33 unidades de costo.

$C_{22}-C_{21}+C_{11}-C_{14}+C_{34}-C_{32}=C_{22}-(u_2+v_2)=30-49=-19$

∴ por cada unidad que aumente x_{22}, se ahorran 19 unidades de costo

Sólo se pudieron ahorrar 19x2=38 unidades de costo respecto a la solución anterior.

No hay negativos ∴ no se puede mejorar esta solución

-TRANSPORTE

PROBLEMA DEGENERADO

Tabla de costos y requerimientos (datos)

	1	2	3	
1	7	6	4	2
2	2	1	3	3
3	3	4	6	5
	4	1	5	

La solución factible básica inicial es degenerada: cuatro $X_{ij} \neq 0$ en vez de $m + n - 1 = 3 + 3 - 1 = 5$.

Solución factible básica inicial

Con $\epsilon << 1$ perturbamos al problema

1ª

	1	2	3	
1	2			2
2	2	1	ϵ	$3+\epsilon$
3			5	5
	4	1	$5+\epsilon$	

	1	2	3	u_i
1	(7)	6	8	0
2	(2)	(1)	(3)	-5
3	5	4	(6)	-2
v_j	7	6	8	

	1	2	3
1		-3	-4
2			
3	-2		

2ª

	1	2	3	
1	$2-\epsilon$		ϵ	2
2	$2+\epsilon$	1		$3+\epsilon$
3			5	5
	4	1	$5+\epsilon$	

	1	2	3	u_i
1	(7)	6	(4)	0
2	(2)	(1)	-1	-5
3	9	8	(6)	2
v_j	7	6	4	

	1	2	3
1		-3	
2			
3	-6	-4	

3ª

	1	2	3	
1	$2-\epsilon-\theta$		$(2)^{+\theta}$	2
2	$2+\epsilon$	(1)		$3+\epsilon$
3	θ $2-\epsilon$		$5-\theta$ $3+\epsilon$	5
	4	1	$5+\epsilon$	$\theta=2-\epsilon$

	1	2	3	u_i
1	1	0	(4)	0
2	(2)	(1)	(5)	1
3	(3)	2	(6)	2
v_j	1	0	4	

	1	2	3
1			
2			-2
3			

4ª

	1	2	3	
1			(2)	2
2	$2+\epsilon-\theta$	(1)	θ $2+\epsilon$	$3+\epsilon$
3	$2-\epsilon+\theta$ 4		$3+\epsilon-\theta$	5
	4	1	$5+6$	$\theta=2+\epsilon$

	1	2	3	u_i
1	1	2	(4)	0
2	0	(1)	(5)	-1
3	(3)	4	(6)	-2
v_j	1	2	4	

	1	2	3
1			
2			
3			

No hay negativos ∴ la solución es óptima haciendo $\epsilon = 0$

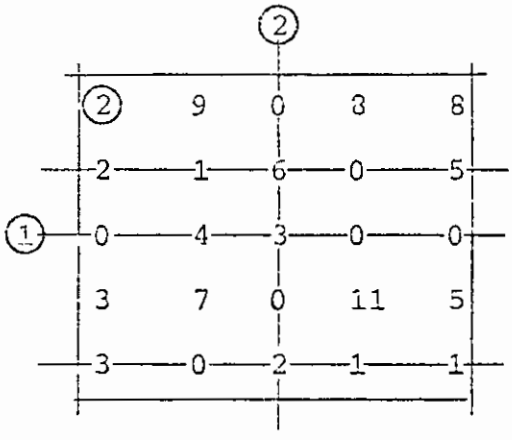
	1	2	3	
1			2	2
2		1	2	3
3	4		1	5
	4	1	5	

Solución óptima al problema degenerado

PROBLEMA DE ASIGNACION

	1	2	3	4	5	restando
1	11	17	8	16	20	8
2	9	7	12	6	15	6
3	13	16	15	12	16	12
4	21	24	17	28	26	17
5	14	10	12	11	15	10

3	9	0	8	12	
3	1	6	0	9	
1	4	3	0	4	
4	7	0	11	9	
4	0	2	1	5	
1	0	0	0	4	restando



	1	2	3	4	5
1	[0]	7	0	6	6
2	2	1	8	[0]	5
3	0	4	5	0	[0]
4	1	5	[0]	9	3
5	3	[0]	4	1	1

SOLUCION OPTIMA:

asignar

renglón	con columna	costo
1	1	11
2	4	6
3	5	16
4	3	17
5	2	10
	costo total	60

6. Eliminación de Gauss-Jordan

Sea S el conjunto de soluciones $(x_1,\ldots x_n)$ del siguiente sistema de m ecuaciones con n variables:

$$\begin{aligned} &a_{11}x_1+ a_{12}x_2+\ldots+ a_{1n}x_n=b_1\\ &a_{21}x_1+a_{22}x_2+\ldots a_{2n}x_n=b_2\\ &a_{m1}x_1+a_{m2}x_2+\ldots a_{mn}x_n=b_m \end{aligned} \qquad (6.1)$$

Se sabe que si a una ecuación cualquiera se le suma una combinación lineal arbitraria de las m ecuaciones, entonces se obtiene un "nuevo" sistema de ecuaciones cuyo conjunto de soluciones es idéntico al original, es decir, S. Si alguna ecuación se puede obtener como combinación lineal de las demás, entonces las ecuaciones son linealmente dependientes (hay al menos una ecuación redundante). En caso contrario las ecuaciones son linealmente independientes.

Supongamos que (6.1) tiene alguna solución (S no vacío) y que las m ecuaciones (6.1) son linealmente independientes. Si $m < n$, por eliminación de Gauss-Jordan se puede escribir (6.1) como

$$\begin{aligned} &x_1 + a'_{1\,m+1} + \ldots + a'_{1n}x_n = b_1'\\ &x_2 + a'_{2\,m+1} + \ldots + a'_{2n}x_n = b_2'\\ &\qquad\qquad\vdots\\ &x_m + a'_{m\,m+1} + \ldots + a'_{mn}x_n = b_m' \end{aligned}$$

en donde se exhibe la solución $x_j = b'_j$ para $j=1,2,\ldots m$; $x_j=0$ para $j=m+1,\ldots,n$.

Por definición esta es una <u>solución básica</u> por tener $n-m$ variables iguales a cero; a las variables x_j con $1\le j\le m$ se les llama variables

básicas (o en la base) y al resto, variables no básicas. Si en una solución básica una de las variables básicas es cero, entonces se dice que dicha solución es degenerada.

La eliminación de G-J puede considerarse como una sucesión de "pivoteos" sobre diferentes ecuaciones y diferentes variables. Así (6.2) si puede obtener de (6.1) como sigue: se despeja x_1 de la primera ecuación de (6.1) y se elimina (x_1) de las demás ecuaciones por sustitución. Se despeja x_2 de la segunda ecuación resultante y se elimina del resto de ecuaciones; y así sucesivamente. Este proceso se realiza más eficientemente por medio de una tabla-matriz.

Consideremos el sistema

z	x_1	...	x_r	...	x_m	x_s	...	
1	0		0		0	$-c_s$		H
0	1		0		0	a_{1s}		b_1
...								...
0	0		1		0	a_{rs}		b_r
...								...
0	0		0		1	a_{ms}		b_m

en el que se está exhibiendo la solución básica $z=H$, $x_i=b_i$ $i=1,\ldots m$

Si se toma a_{rs} como elemento pivote, se obtiene

z	x_1	...	x_r	...	x_m	...	x_s	...	
1	0		c_s/a_{rs}		0		0		$H-c_s b_s/a_{rs}$
0	1		$-a_{1s}/a_{rs}$		0		0		$b_1-b_r a_{1s}/a_{rs}$
...									...
0	0		$1/a_{rs}$		0		1		b_r/a_{rs}
...									...
0	0		$-a_{ms}/a_{rs}$		1		0		$b_m-b a_{ms}/a_{rs}$

observándose que después del pivoteo se tiene:

a) una nueva solución básica en que salió de la base x_r, y en su lugar entró x_s

b) para renglones distintos del pivote, los lados derechos son:

$a_{is}\ (b_i/a_{is} - b_r/a_{rs})$ $\quad i \neq r$

$+ a_s\ b_r/a_{rs}$ para la primera ecuación

7. El problema de la programación lineal

Se dispone de las cantidades $b_1, b_2, .., b_i ... b_m$ (datos) de recursos para producir n productos $x_1, x_2, .., x_j, ..., x_n$ (incógnitas) los cuales dan utilidades unitarias $c_1, c_2, c_j, .., c_n$ respectivamente (datos). Para producir una unidad del producto j usando recursos b_i, se requiere una cantidad a_{ij} de dichos recursos (datos). El problema es determinar las cantidades x_i de productos que deben producirse de acuerdo con las limitaciones de recursos, de modo que las utilidades sean máximas.

Matematicamente, dados:

(a) $b_1, b_2, .., b_i, ... b_m$

(b) $c_1, c_2 .., c_j, ... c_n$

(c) a_{ij} $(i = 1, 2, .., m;\ j = 1, 2, .., n)$

encontrar x_j $(j = 1, 2, .., n)$ tal que sea máximo

$$Z = \sum_{j=1}^{n} c_j x_i$$

cumpliendose las restricciones.

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, 2, .., m \text{ (limitación de recursos)}$$

$x_j \geq 0$ $j = 1, 2, .., n$ (producción no negativa)

Usando matrices:

max $Z = cx$ con $Ax \leq b$, $x \geq 0$

EJEMPLO 1

PRODUCTO	POR UNIDAD		CAPACIDAD MAXIMA DE PRODUCCION
	HORAS DE FABRICACION	UTILIDAD	
1	3	3	4
2	2	5	6

Horas hábiles/día = 18
¿Qué cantidades x_1 y x_2 deben fabricarse por día?.

Planteamiento:

Max $Z = 3x_1 + 5x_2$ (0)

con $x_1 \leq 4$ (1)

$x_2 \leq 6$ (2)

$3x_1 + 2x_2 \leq 18$ (3)

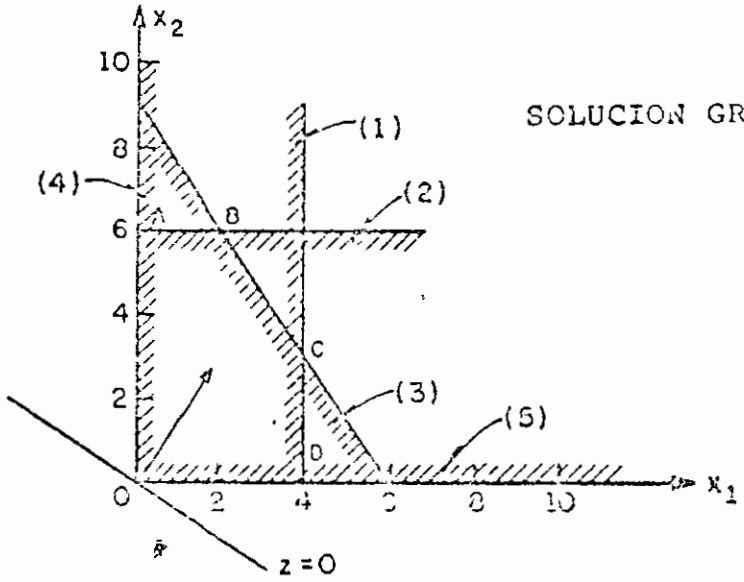
$x_1 \geq 0$, $x_2 \geq 0$ (4), (5)

Introduciendo las variables de holgura x_3, x_4, x_5, el problema queda: Max $Z = 3x_1 + 5x_2$ con las restricciones

$x_1 + x_3 = 4$, $x_2 + x_4 = 6$, $3x_1 + 2x_2 + x_5 = 18$,

$x_i \geq 0$ para $i = 1, 2, 3, 4, 5$.

SOLUCION GRAFICA



SOLUCION POR SIMPLEX PRIMAL

		(Y_4	Y_5	Y_1	Y_2	Y_3)		
	Base	z	x_1	x_2	x_3	x_4	x_5	Der.
1ª (0)	z	1	-3	(-5)	0	0	0	0
1ª (1)	x_3	0	1	0	1	0	0	+4
1ª (2)	x_4	0	0	[1]	0	1	0	(+6)
1ª (3)	x_5	0	3	2	0	0	1	+18
2ª (0)	z	1	(-3)	0	0	5	0	30
2ª (1)	x_3	0	1	0	1	0	0	4
2ª (2)	x_2	0	0	1	0	1	0	6
2ª (3)	x_5	0	[3]	0	0	-2	1	(6)
3ª (0)	z	1	0	0	0	3	1	36
3ª (1)	x_3	0	0	0	1	$\frac{2}{3}$	$-\frac{1}{3}$	2
3ª (2)	x_2	0	0	1	0	1	0	6
3ª (3)	x_1	0	1	0	0	$-\frac{2}{3}$	$\frac{1}{3}$	2

(0) la. solución factible básica: x_3, x_4, y x_5 en la base y elementos derechos no negativos.

(1) $\text{Min}\{-3,-5\}=-5<0$.˙. x_2 entrada base.

*(2) $\text{Min}\left\{\frac{+6}{1}, \frac{+18}{2}\right\}=\frac{6}{1}$.˙. sale x_4 de la base.

(3) Resolver para las nuevas var. bas. (eliminac. de Gauss-Jordán), lo que queda la 2a. soluc. fact. bas.

(1) $\text{Min}\{-3\}=-3<0$.˙. x_1 entra a la base.

(2) $\text{Min}\left\{\frac{+4}{1}, \frac{+6}{3}\right\}=\frac{6}{3}$.˙. sale x_5 de la base.

(3) eliminación G-J para construir 3a. solución fact. bas.

(1) no hay negativos en el renglón (0) .˙. la 3a. solución factible básica es la óptima con un costo de MAX Z = 36.

* Si todos los elementos de una columna no básica (excluido el elemento (0)) son negativos, entoces (no podemos ejecutar el paso (2)) el problema es no acotado: $Z = \infty$.

D U A L I D A D

PROBLEMA PRIMAL	PROBLEMA DUAL	FORMA
$\text{Max } Z = cx$ $Ax \leqslant b$ $x \geqslant 0$	$\text{Min } w = b'y$ $A'y \geqslant c'$ $y \geqslant 0$	canónica
$\text{Max } Z = cx$ $Ax \leqslant b$ $x \geqslant 0$	$\text{Min } w = b'y$ $A'y = c'$ $y \geqslant 0$	estandard

ALGUNAS RELACIONES ENTRE AMBOS PROBLEMAS:

1. El problema dual del dual es el primal.

2. Si x y y son soluciones factibles del primal y del dual respectivamente entonces $Z = cx \leqslant b'y = w$.

3. Teorema de dualidad: Dados los problemas primal y dual en su forma canónica, entonces uno y solo uno de los siguientes casos se cumple:

   a) Ambos problemas tienen soluciones óptimas finitas y $Z = w$ para tales soluciones.

   b) Un problema no tiene solución factible, y el otro tiene al menos una solución factible óptima no finita.

   c) Ninguno de los dos problemas tiene una solución factible.

E J E M P L O 2

Llamemos problema primal a
Max $w = -4y_1 - 6y_2 - 18y_3$

$$\begin{pmatrix} -1 & 0 & -3 \\ 0 & -1 & -2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \leq \begin{pmatrix} -3 \\ -5 \end{pmatrix}$$

SOLUCION POR METODO SIMPLEX DUAL

(x x x x x)
3 4 5 1 2

	Base	w	y_1	y_2	y_3	y_4	y_5	Der
1º (0)	w	1	+4	+6	+18	0	0	0
1º (1)	y_4	0	-3	0	-3	1	0	-3
1º (2)	y_5	0	0	[-1]	-2	0	1	(-5)
2º (0)	w	1	4	0	6	0	6	-30
2º (1)	y_4	0	-1	0	[-3]	1	0	(-3)
2º (2)	y_2	0	0	1	2	0	-1	5
3º (0)	w	1	2	0	0	2	6	-36
3º (1)	y_3	0	$\frac{1}{3}$	0	1	$\frac{-1}{3}$	0	1
3º (2)	y_2	0	$\frac{-2}{3}$	1	0	$\frac{2}{3}$	-1	3

(0) Solución factible básica del problema dual· una solución básica para el primal, no necesariamente factible (por los negativos de la columna derecha), y en (0) coeficientes no negativos para las y's no básicas.

(1) $\text{Min}\{-3, -5\} = -5$ . . y_5 sale de la base

*(2) $\text{Max}\left\{\frac{6}{-1}, \frac{18}{-2}\right\} = \frac{6}{-1}$ . . y_2 entra a la base con lugar de y_5

(3) Eliminación de Gauss-Jordán para obtener 2a. solución factible básica del dual.

(1) $\text{Min}\{-3\} = -3$ . . y_4 sale de la base

(2) $\text{Max}\left\{\frac{4}{-1}, \frac{6}{-3}\right\} = \frac{6}{-3}$ . . y_3 entra a la base en lugar de y_4

(3) Eliminación de G - J para obtener 2a. solución factible básica del dual

(1) Lado derecho no negativo . . solución básica del primal es factible . . solución óptima·

$y_3 = 1$ $y_2 = 3$ $y_1 = y_4 = y_5 = 0$ $w = -36$

* Si todos los elementos de renglón con lado derecho negativo (excluido el elemento w) son no negativos . entonces (no podemos ejecutar el paso (2)) el problema primal no tiene solución factible.

Apéndice 1: Problemas de Programación Lineal

1. Una compañía debe producir un cierto producto en cantidad suficiente para cumplir con las ventas contratadas para los siguientes cuatro meses. La capacidad de producción de este producto está limitada según los diferentes meses. Los costos unitarios de producción varían de acuerdo a la disponibilidad de personal. El producto puede ser producido en un mes y retenido para venderse en un mes posterior, pero a un costo de almacenamiento de $ 1.00 por unidad por mes. No se incurre en costos de almacenamiento para producto producido en ese mismo mes. Por razones de caducidad, la compañía desea tener existencia nula de este producto al final de los cuatro meses. Según los datos dados en la siguiente tabla, ¿Cuánto debe producirse en cada uno de los cuatro meses con el fin de hacer mínimo el costo total?

MES	VENTAS CONTRATADAS	PRODUCCION MAXIMA	COSTO UNITARIO DE PRODUCCION	COSTO UNITARIO DE ALMACENAMIENTO POR MES
1	20	30	8	1
2	30	50	12	1
3	50	20	11	1
4	40	50	14	1

2.Un fabricante de muebles estima que tendrá la siguiente demanda: cuando menos 40 mesas, 130 sillas, 30 escritorios y no más de 10 libreros. y para satisfacerla dispone de dos tipos de madera: 1,500 metros de tabla de tipo A y 1,000 del tipo B. Dispone además de 800 horas-hombre para efectuar el trabajo. Las cantidades de madera, las horas-nombre y las utilidades correspondientes a cada cantidad de artículo, se indican en el siguiente cuadro:

Artículo	Madera A	Madera B	horas hombre	demanda estimada	utilidad
Mesas	5	2	3	cuando menos 40	$ 12.00
Sillas	1	3	2	cuando menos 130	$ 5.00
Escritorios	9	4	5	cuando menos 30	15.00
Libreros	12	1	10	cuando menos 10	10.00
TOTAL	1500	1000	800		

Qué número de mesas, sillas, escritorios y libreros debe producir para maximizar sus ganancias.

3.Una industria produce dos artículos: A y B,la elaboración de una unidad del artículo A requiere de los siguientes insumos:

$ 20.00 de mano de obra

$ 10.00 de materia prima

5.00 por depreciación de equipo

La utilidad que se obtiene por cada unidad del artículo A, es $ 8.00.

Para manufacturar una midad del artículo B, se requieron los siguientes insumos:

$ 10.00 de mano de obra

$ 30.00 de materia prima

1.00 por depreciación de equipo

La utilidad que se obtiene por cada unidad del artículo B es $ 5.00

Los recursos para producir dichos artículos son:

$ 100,000.00 para salarios

$ 180,000.00 para materia prima

y además, se requiere que la depreciación del equipo no sea mayor de $ 40,000.00

¿Qué cantidades de artículos A y B deben producirse para que la empresa obtenga las máximas utilidades?

4. Una fábrica de automóviles y camiones consta de los siguientes departamentos;

1. Estampado de planchas metálicas
2. Armado de motores
3. Montaje de automóviles
4. Montaje de camiones

El departamento 1 puede estampar, por mes, las planchas necesarias para 25,000 automóviles o 35,000 camiones, o las correspondientes combinaciones de automóviles y camiones. El departamento 2 puede armar, por mes, 33,333 motores de automóviles ó 16,667 motores de camiones, o las correspondientes combinaciones de motores de automóvil y camión. El departamento 3 - puede montar y terminar 22,500 automóviles y 15,000 camiones el departamento 4. Si cada automóvil deja una utilidad de 300 pesos y cada camión de 250 ¿qué cantidades de automóviles y camiones deben producirse, para que las utilidades que se obtengan sean las máximas posibles?

5. Un industrial desea decerminar el programa óptimo para tres mezclas distintas que hace diferentes proporciones de pistaches, avellanas y cacahuates. Las especificaciones de cada una de ellas son: la mezcla 1 debe contener 50 por ciento de pistaches como mínimo y 25 por ciento de cacahuates cuando mas; el kilo de esta mezcla se vende a 50 centavos. El segundo tipo debe contener 25 por ciento de pistaches, por lo menos y 50 por ciento de cacahuates, cuando más y se vende a 35 centavos el kilo. El tercer tipo no tiene especificaciones y se vende a 25 centavos el kilo.

Las máximas cantidades de materias primas que puede conseguir el industrial para cada período son : 100 kilos de pistache, 100 kilos de cacahuate y 60 kilos de avellana. Cada kilo de pistache le cuesta 65 centavos, la de cacahuate 25 centavos y

35 cortezos de avellanas. Se trata de determinar cuántos kilos se deben preparar de cada mezcla, para obtener las utilidades máximas.

6. Una fábrica de papel recibe el siguiente pedido: 800 rollos de 50 centímetros de ancho, 500 rollos de 45 centímetros de ancho y 1,000 de 56 centímetros de ancho. Si en el almacén solamente existen rollos de 108 centímetros de ancho ¿cómo deben cortarse los rollos para surtir el pedido con el mínimo desperdicio de papel?

7. Una empresa tiene tres plantas con exceso de producción de un cieto producto. Se ha decidido utilizar el exceso de producción en la obtención de ese producto en forma óptima.

Este producto puede ser fabricado en tres tamaños (grande, media no y chico A,B y C). Que dan una utilidad por unidad de $ 12, $ 10 y $ 9 respectivamente. Las plantas 1, 2y 3 tienen capacidad para producir 500, 600 y 300 unidades del producto por día respectivamente, independientemente del tamaño o combinación, sin embargo, el espacio disponible para el almacenamiento, impone una limitación. Las plantas 1, 2 y 3 tienen 9,000, 8,000 y 3,500 M^2, y cada unidad del producto A, B y C requieren 20. 15 y 2 M^2 respectivamente.

Las ventas indican que hasta 600, 800 y 500 unidades A, B y C respectivamente pueden ser vendidas por día.

Con objeto de mantener uniforme la carga de trabajo entre las plantas, se ha decidido que la producción adicional asignada a cada planta sea en porcentaje la misma.

Se desea conocer la cantidad de cada artículo que se debe producir con el objeto de maximizar las utilidades.

8. Una compañía de aviación, al considerar la compra de aeroplanos de pasajeros de 3 tipos: pequeño, mediano y grande, dispone de los siguientes datos Los precios de compra por unidad son $ 3,500.000.00, $ 5,000.000.00 y $ 6,700.000.00, respectivamente la cantidad de dinero disponible para estas compras es de $ 150,000.000.00 según cálculos de la empresa, se sabe que cada avión será utilizado casi siempre a su máxima capacidad; que existirán suficientes pilotos para manejar 30 aeroplanos; que si se compran aviones de tipo pequeño solamente se podrá disponer de mantenimiento para 40 unidades, en tanto que el mantenimiento para aeroplanos de tipo mediano o grande será de 4/3 o 5/3 del que se requiere para el tipo pequeño.

La utilidad neta anual por unidad se estima que será de $ 250,000.00 para el tipo pequeño, $ 300,000.00 para el tipo mediano y $ 420.000.00 para el tipo grande.

Utilizando la información anterior en un análisis preliminar sin considerar que el número de aeroplanos debe ser entero, la compañía desea conocer cuantos aviones de cada tipo deben comprarse para maximizar las ganancias.

9. Un inversionista tiene varias posibles actividades (A, B, C y D). Las actividades A y B realizables al principio de cada uno de los cinco años siguientes. Por cada peso invertido en la actividad A, obtendrá $ 1.40 ($0.40 de utilidad) dos años después, los cuales pueden ser reinvertidos inmediatamente. Al invertir un peso en B, obtendrá $ 1.70 ($0.70 de utilidad) tres años después. Por cada peso invertido en C al principio del segundo año obtiene $ 2.00 cuatro años después. Por cada peso invertido en D al principio del quinto año obtiene $ 1.30 un año después.

El inversionista tiene $ 10,000.00 y quiere conocer qué plan de inversión maximiza la cantidad de dinero que él puede tener al principio del sexto año.

10. Un proveedor tiene contratado el servicio de n banquetes que están programados para n días consecutivos. Se sabe que el j-ésimo banquete se deben utilizar n_j $(j=1,2,\ldots,n)$ servilletas limpias, para lo cual dispone de las siguientes alternativas:

	costo/servilleta	plazo de entrega en días
compra de servilletas nuevas	a	0
servicio normal de lavandería	b	q (entero)
servicio rápido de lavandería	c	p (entero)

teniendose, $a > b > c$, $q < p$

Las servilletas pueden usarse depués de ser enviadas a la lavandería un número indefinido de veces. El valor de salvación de

las servilletas usadas es nulo. Suponiendo que inicialmente no se tienen servilletas nuevas ni en la lavandería, ¿cómo debe programar su negocio el proveedor para tener un costo mínimo al final del contrato?

PROGRAMACION LINEAL

EJEMPLO 1

PRODUCTO	POR UNIDAD		CAPACIDAD MAXIMA DE PRODUCCION
	HORAS DE FABRICACION	UTILIDAD	
1	3	3	4
2	2	5	6

Horas hábiles/día = 18.
¿Qué cantidades x_1 y x_2 deben fabricarse por día?.

Planteamiento:

Max $Z = 3x_1 + 5x_2$ (0)

con $x_1 \leq 4$ (1)

$x_2 \leq 6$ (2)

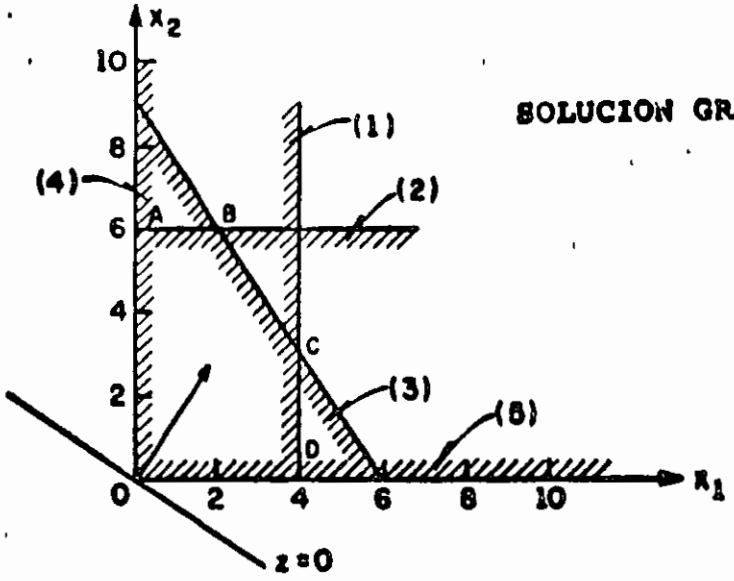
$3x_1 + 2x_2 \leq 18$ (3)

$x_1 \geq 0$, $x_2 \geq 0$ (4), (5)

Introduciendo las variables de holgura x_3, x_4, x_5, el proble - ma queda: Max $Z = 3x_1 + 5x_2$ con las restricciones

$x_1 + x_3 = 4$, $x_2 + x_4 = 6$, $3x_1 + 2x_2 + x_5 = 18$,

$x_i \geq 0$ para $i = 1, 2, 3, 4, 5$.



SOLUCION GRAFICA

SOLUCION POR SIMPLEX PRIMAL

		(y_4	y_5	y_1	y_2	y_3)		
	Base	z	x_1	x_2	x_3	x_4	x_5	Der.
1ª (0)	z	1	-3	(-5)	0	0	0	0
1ª (1)	x_3	0	1	0	1	0	0	+4
1ª (2)	x_4	0	0	[1]	0	1	0	(+6)
1ª (3)	x_5	0	3	2	0	0	1	+18
2ª (0)	z	1	(-3)	0	0	5	0	30
2ª (1)	x_3	0	1	0	1	0	0	4
2ª (2)	x_2	0	0	1	0	1	0	6
2ª (3)	x_5	0	[3]	0	0	-2	1	(6)
3ª (0)	z	1	0	0	0	3	1	36
3ª (1)	x_3	0	0	0	1	$\frac{2}{3}$	$-\frac{1}{3}$	2
3ª (2)	x_2	0	0	1	0	1	0	6
3ª (3)	x_1	0	1	0	0	$-\frac{2}{3}$	$-\frac{1}{3}$	2

(0) 1a. solución factible básica: x_3, x_4, y x_5 en la base y elementos derechos no negativos.

(1) $\text{Min}\{-3,-5\}=-5<0$.˙. x_2 entrada base.

*(2) $\text{Min}\left\{\frac{+6}{1}, \frac{+18}{2}\right\}=\frac{6}{1}$.˙. sale x_4 de la base.

(3) Resolver para las nuevas var. bas. (eliminac. de Gauss-Jordán), lo que queda la 2a. soluc. fact. bas.

(1) $\text{Min}\left\{-3\right\}=-3<0$.˙. x_1 entra a la base.

(2) $\text{Min}\left\{\frac{+4}{1}, \frac{+6}{3}\right\}=\frac{6}{3}$.˙. sale x_5 de la base.

(3) eliminación G-J para construir 3a. solución fact. bas.

(1) no hay negativos en el renglón (0) .˙. la 3a. solución factible básica es la óptima con un costo de MAX Z = 36.

* Si todos los elementos de una columna no básica (excluido el elemento (0)) son negativos, entoces (no podemos ejecutar el paso (2)), el problema es no acotado: Z = ∞.

manejar 30 aeroplanos; que si se compran aviones de tipo pequeño solamente se podrá disponer de mantenimiento para 40 unidades, en tanto que el mantenimiento para aeroplanos de tipo mediano o grande será de 4/3 o 5/3 del que se requiere para el tipo pequeño.

La utilidad neta anual por unidad se estima que será de $230,000.00 para el tipo pequeño, $300,000.00 para el tipo mediano y $420,000.00 para el tipo grande.

Utilizando la información anterior en un análisis preliminar, sin considerar que el número de aeroplanos debe ser entero, la compañía desea conocer cuantos aviones de cada tipo deben comprarse para maximizar las ganancias.

8. Un inversionista tiene varias posibles actividades (A, B, C y D), las actividades A y B realizables al principio de cada uno de los cinco años siguientes. Por cada peso invertido en la actividad A, obtendrá $ 1.40 ($ 0.40 de utilidad) dos años después, los cuales pueden ser reinvertidos inmediatamente. Al invertir un peso en B, obtendrá $ 1.70 ($0.70 de utilidad) tres años después. Por cada peso invertido en C al principio del segundo año obtiene $ 2.00 cuatro años después. Por cada peso invertido en D al principio del quinto año obtiene $ 1.30 un año después.

El inversionista tiene $ 10,000.00 y quiere conocer qué plan de inversión maximiza la cantidad de dinero que él puede tener al principio del sexto año.

6, Una empresa tiene tres plantas con exceso de producción de un cierto producto Se ha decidido utilizar el exceso de producción en la obtención de ese producto en forma óptima.

Este producto puede ser fabricado en tres tamaños (grande, mediano y chico A, B y C). Que dan una utilidad por unidad de $ 12, 10 y 9 respectivamente. Las plantas 1, 2 y 3 tienen capacidad para producir 500, 600 y 300 unidades del producto por día respectivamente, independientemente del tamaño o combinación, sin embargo, el espacio disponible para el almacenamiento, impone una limitación. Las plantas 1, 2 y 3 tienen 9,000, 8,000 y 3,500 ft^2, y cada unidad del producto A, B y C requieren 20, 15 y 2 ft^2 respectivamente.

Las ventas indican que hasta 600, 800 y 500 unidades A, B y C respectivamente pueden ser vendidas por día.

Con objeto de mantener uniforme la carga de trabajo entre las plantas se ha decidido que la producción adicional asignada a cada planta sea en porcentaje la misma.

Se desea conocer la cantidad de cada artículo que se debe producir con el objeto de maximizar las utilidades.

7 Una compañía de aviación, al considerar la compra de aeroplanos de pasajeros de 3 tipos: pequeño, mediano y grande, dispone de los siguientes datos: Los precios de compra por unidad son $ 3,500,000.00, $ 5,000,000.00 y $ 6,700,000.00, respectivamente; la cantidad de dinero disponible para estas compras es $ 150,000,000.00 según cálculos de la empresa, se sabe que cada avión será utilizado casi siempre a su máxima capacidad; que existirán suficientes pilotos para -

[illegible] 21 [illegible] 3 puede montar y [illegible] 22,500 automóviles y 1[illegible],000 camiones el Departamento 4. Si cada automóvil deja una utilidad de 300 dólares y cada camión de 250, ¿qué cantidades de automóviles y camiones deben producirse, para que las [illegible] [illegible] las utilidades [illegible]?

Un industrial desea determinar el programa óptimo para tres mezclas distintas que hace con diferentes proporciones de pistachos, avellanas y cacahuates. Las especificaciones de cada una de ellas son: la mezcla 1 debe contener 50 por ciento de pistachos como mínimo y 25 por ciento de cacahuates cuando más; la libra de esta mezcla se vende a 50 centavos. El segundo tipo debe contener 25 por ciento de pistachos, por lo menos, y 50 por ciento de cacahuates, cuando más, y se vende a 35 centavos la libra. El tercer tipo no tiene especificaciones y se vende a 25 centavos la libra.

Las máximas cantidades de materias primas que puede conseguir el industrial para cada período son: 100 libras de pistacho, 100 libras de cacahuate y 60 libras de avellana. Cada libra de pistacho le cuesta 65 centavos, la de cacahuate 25 centavos y [illegible] centavos la de avellanas. Se trata de determinar cuántas libras se deben preparar de cada mezcla, para obtener las utilidades máximas.

[illegible] fábrica de papel recibe el siguiente pedido: 800 rollos de 30 pulg. de ancho, 500 rollos de 45 pulg. de ancho y 1,000 de 56 pulg. de ancho. Si en el almacén solamente existen rollos de 108 pulg. de ancho ¿cómo deben cortarse los rollos para surtir el pedido con el mínimo desperdicio de papel?

EJEMPLOS DE PROBLEMAS DE PROGRAMACION LINEAL

PLANTEAMIENTO DE PROBLEMAS

1. Un fabricante de muebles estima que tendrá la siguiente demanda: cuando menos 40 mesas, 130 sillas, 30 escritorios y no más de 70 libreros; y para satisfacerla dispone de dos tipos de madera: 1,500 pies de tabla de tipo A y 1000 del tipo B. Dispone, además, de 800 horas hombres para efectuar el trabajo. Las cantidades de madera, las horas-hombres y las utilidades correspondientes a cada unidad de artículo, se indican en el siguiente cuadro.

ARTICULO	Madera A	Madera B	Horas hombre	Demanda estimada	Utilidad por unidad
mesa	5	2	3	Cuando menos	$ 12.00
silla	1	3	2	Cuanto menos	$ 5.00
Escritorio	9	4	5	Cuando menos	$ 15.00
Librero	12	1	10	No más de	$ 10.00
TOTAL	1500	1000	800		

Qué número de mesas, sillas, escritorios y libreros debe producir para maximizar sus ganancias

2. Una industria produce dos artículos: A y B. La elaboración de una unidad del artículo A requiere de los siguientes insumos:

   $ 20.00 de mano de obra

   $ 10.00 de materia prima

   $ 5.00 por depreciación de equipo

La utilidad que se obtiene por cada unidad del artículo A, es $ 8.00

Para manufacturar una unidad del artículo B, se requieren los siguientes insumos:

$ 10.00 de mano de obra

$ 30.00 de materia prima

$ 1.00 por depreciación de equipo

La utilidad que se obtiene por cada unidad del artículo B, es $ 9.00

Los costos para producir dichos artículos son:

$ 400,000.00 para salarios

$ 480,000.00 para materia prima

Y además, se requiere que la depreciación del equipo no sea mayor de $ 50,000.00.

¿Qué cantidades de artículos A y B deben producirse para que la empresa obtenga los máximos utilidades?

La planta de automóviles y camiones consta de los siguientes departamentos:

1. Estampado de planchas metálicas
2. Armado de motores
3. Montaje de automóviles
4. Montaje de camiones

El departamento 1 puede estampar, por mes, las planchas necesarias para 25,000 automóviles o 35,000 camiones, o las correspondientes combinaciones de automóviles y camiones. El departamento 2 puede armar, por mes, 33,333 motores de automóviles o 16,667 motores de camiones o las correspondientes combinaciones de motores de auto-

DESCRIPCION PROBLEMA DE PROGRAMACION LINEAL ("PRIMAL")

Se disponen de las cantidades $b_1, b_2, .., b_i ... b_m$ (datos) de recursos para producir n productos $x_1, x_2, .., x_j, ..., x_n$ (incógnitas) los cuales dan utilidades unitarias $c_1, c_2, c_j, .., c_n$ respectivamente (datos). Para producir una unidad del producto j usando recursos b_i, se requiere una cantidad a_{ij} de dichos recursos (datos). El problema es determinar las cantidades x_i de productos que deben producirse de acuerdo con las limitaciones de recursos, de modo que las utilidades sean máximas.

Matemáticamente, dados:

(a) $b_1, b_2, .., b_i, ... b_m$

(b) $c_1, c_2, .., c_j, ... c_n$

(c) a_{ij} $(i = 1, 2, .., m; j = 1, 2, .., n)$

encontrar x_j $(j = 1, 2, .., n)$ tal que sea máximo

$$Z = \sum_{j=1}^{n} c_j x_i$$

cumpliéndose las restricciones:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, 2, .., m \text{ (limitación de recursos)}$$

$x_j \geq 0 \quad j = 1, 2, .., n$ (producción no negativa)

Usando matrices:

Max $Z = cx$ con $Ax \leq b$, $x \geq 0$

E J E M P L O 2

Llamemós problema primal a

Max $w = -4y_1 - 6y_2 - 18y_3$

$$\begin{pmatrix} -1 & 0 & -3 \\ 0 & -1 & -2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \leq \begin{pmatrix} -3 \\ -5 \end{pmatrix}$$

SOLUCION POR METODO SIMPLEX DUAL

(x x x x x)
3 4 5 1 2

		Base	w	y_1	y_2	y_3	y_4	y_5	Der.
1º	(0)	w	1	+4	+6	+18	0	0	0
	(1)	y_4	0	-1	0	-3	1	0	-3
	(2)	y_5	0	0	[-1]	-2	0	1	(-5)
2º	(0)	w	1	4	0	6	0	6	-30
	(1)	y_4	0	-1	0	[-3]	1	0	(-3)
	(2)	y_2	0	0	1	2	0	-1	5
3º	(0)	w	1	2	0	0	2	6	-36
				1			-1		
	(1)	y_3	0	$\frac{1}{3}$	0	1	$\frac{1}{3}$	0	1
				-2			2		
	(2)	y_2	0	$\frac{-2}{3}$	1	0	$\frac{2}{3}$	-1	3

(0) Solución factible básica del problema dual: una solución básica para el primal, no necesariamente factible (por los negativos de la columna derecha), y en (0) coeficientes no negativos para las y's no básicas.

(1) $\text{Min}\{-3, -5\} = -5$.·. y_5 sale de la base

*(2) $\text{Max}\left\{\frac{6}{-1}, \frac{18}{1-2}\right\} = \frac{6}{-1}$.·. y_2 entra a la base con lugar de y_5

(3) Eliminación de Gauss-Jordán para obtener 2a. solución factible básica del dual.

(1) $\text{Min}\{-3\} = -3$.·. y_4 sale de la base

(2) $\text{Max}\left\{\frac{4}{-1}, \frac{6}{-3}\right\} = \frac{6}{-3}$.·. y_3 entra a la base en lugar de y_4

(3) Eliminación de G - J para obtener 2a. solución factible básica del dual

(1) Lado derecho no negativo .·. solución básica del primal es factible .·. solución óptima:

$y_3 = 1 \quad y_2 = 3 \quad y_1 = y_4 = y_5 = 0 \quad w = -36$

* Si todos los elementos de renglón con lado derecho negativo (excluido el elemento w) son no negativos entonces (no podemos ejecutar el paso (2)) el problema primal no tiene solución factible.

D U A L I D A D

PROBLEMA PRIMAL	PROBLEMA DUAL	FORMA
$\text{Max } Z = cx$ $Ax \leqslant b$ $x \geqslant 0$	$\text{Min } w = b'y$ $A'y \geqslant c'$ $y \geqslant 0$	canómica
$\text{Max } Z = cx$ $Ax \leqslant b$ $x \geqslant 0$	$\text{Min } w = b'y$ $A'y = c'$ $y \geqslant 0$	estandard

ALGUNAS RELACIONES ENTRE AMBOS PROBLEMAS:

1. El problema dual del dual es el primal.

2. Si x y y son soluciones factibles del primal y del dual respectivamente entonces $Z = cx \leqslant b'y = w$.

3. Teorema de dualidad: Dados los problemas primal y dual en su forma canónica, entonces uno y solo uno de los siguientes casos se cumple:

   a) Ambos problemas tienen soluciones óptimas finitas y $Z = w$ para tales soluciones.

   b) Un problema no tiene solución factible, y el otro tiene al menos una solución factible óptima no finita.

   c) Ninguno de los dos problemas tiene una solución factible.

PROGRAMACION DE REQUERIMIENTOS DE PERSONAL

Prof. Ariel Kleiman

El Dopartamento de Planeación de una importante empresa de fabricación de relojes, está tratando de decidir cuantas operarias se contratarán y entrenarán durante el primer semestre del año próximo para el Departamento de Armado.

El número de horas de trabajo varía de acuerdo con los planes mensuales de producción, por lo que se estimaron las siguientes cifras:

Mes	Horas necesarias
Enero	8 000
Febrero	7 000
Marzo	8 000
Abril	10 000
Mayo	9 000
Junio	12 000

Dos factores complican la programación del reclutamiento de operarias: 1) Se requiere todo un mes para entrenar una operaria, antes de que adquiera la habilidad necesaria para trabajar sin errores; por lo que la contratación debe efectuarse con un mes de anticipación; 2) El entrenamiento de una operaria nueva está a cargo de una operaria experta, y exige 100 horas de enseñanza y control durante el mes de capacitación.

Para el mes de enero no habría problemas, pues se contaría con 60 operarias. De acuerdo con el contrato colectivo de trabajo, el número máximo de horas mensuales de actividad no debe superar de 150, por lo que se tendrían 9 000 horas laborables disponibles.

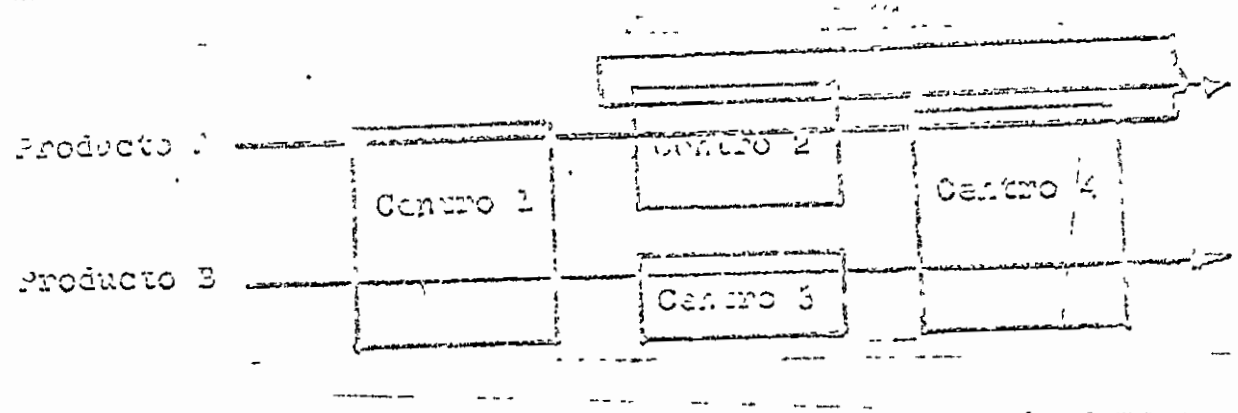
El sueldo mensual de una operaria entrenada, incluyendo prestaciones sociales, es de $1,600.00. El sueldo de una operaria nueva, es de $800.00 durante el período de entrenamiento.

Finalmente el Departamento de Personal ha estimado en el 10% la tasa mensual de renuncias de operarias, (por razones familiares, de salud, cambio de trabajo y otras causas).

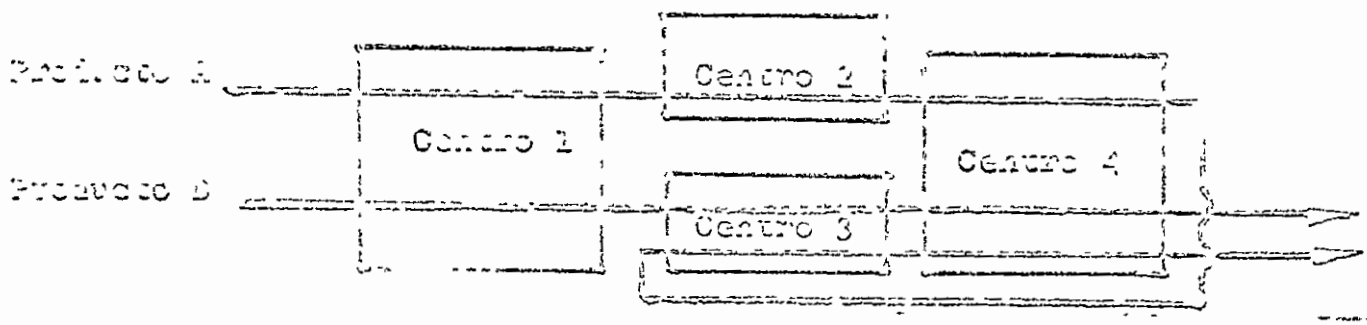
Se pide:
a) Indicar las variables del problema
b) Plantear y explicar cada restricción
c) Formular la función objetivo
d) Comentar el modelo elaborado

PROGRAMACION DE INSUMOS EN PROCESOS DE PRODUCCION

Una pequeña empresa química produce dos tipos de solventes, el A y el B. La planta está organizada en cuatro centros de procesamiento: 1, 2, 3 y 4. El diagrama usual del proceso es el siguiente:



El centro de procesamiento 3 presenta capacidad en exceso a la requerida por el producto B. Por lo tanto, es posible procesar el producto A pasando por el centro 3, en vez de pasarlo dos veces por el centro 4. En tal caso, la utilización de la capacidad estaría mejor balanceada aunque los costos por unidad serían algo mayores. El diagrama de proceso sería el siguiente:



Los centros 1 y 4 operan 16 horas diarias; los centros 2 y 3 sólo 12 horas diarias. No hay limitaciones en la capacidad de almacenaje, pero la capacidad de los ductos limita el volumen diario producido de ambos solventes a un máximo de 2500 litros.

En cuanto a los costos de materia prima, el precio de venta por litro, las ventas máximas diarias y los requerimientos de materia prima por hora de operación, para ambos productos en cada proceso. Los datos se dan a continuación:

Producto	Costo de materia prima (por litro)	Precio de venta (por litro)	Ventas máximas diarias (en litros)
A	3	10	1700
B	6	20	1500

Producto	Centro de procesamiento	Insumos litros por hora)	% de recuperación	Costo de operación ($ por hora)
A	1	300	90	150
	2 (ler. paso)	450	95	200
	4	250	85	180
	2 (2° paso)	400	80	220
	3	350	75	250
B	1	500	90	300
	3	480	85	250
	4	400	80	240

Se pide formular un modelo lineal de programación de la producción que permita maximizar las utilidades. El programa óptimo de producción debe responder a las siguientes preguntas:

a) ¿Cuánta materia prima se utilizará diariamente en la producción del solvente A por el proceso usual?

b) ¿Cuánta materia prima se utilizará diariamente en la producción del solvente A por el proceso opcional?

c) ¿Cuánta materia prima se utilizará diariamente en la producción del solvente B?.

PROGRAMACION DE INSUMOS EN PROCESOS DE PRODUCCION

Se presenta a consideración un plan integral para las operaciones de una empresa internacional de producción, refinación, transporte y distribución de petróleo crudo, gasolinas y productos petroquímicos. La denominaremos "Petroleos Interoceánicos, Sociedad Anónima" (P.I.S.A.).

La empresa P.I.S.A. está organizada como un consorcio de tipo "holding," que es propietario total o parcial de varias compañías subsidiarias, las que son responsables por las operaciones específicas. El problema fundamental de la empresa P.I.S.A. es lograr un plan integral que coordine las actividades de las diversas subsidiarias, optimice los resultados de la operación, y no restrinja excesivamente la autonomía de dichas subsidiarias en la planeación de las actividades que les sean propias.

El primer plan integral se puso en práctica a principios de 1968, coincidiendo con la asamblea anual del Consejo Directivo de la empresa "holding", en la que se analizaron los resultados económicos y financieros del ejercicio anterior, y se aprobó el esquema de plan integral.

El plan fue elaborado durante el segundo semestre de 1967 por el Departamento de Estudios Económicos de la empresa, con el asesoramiento de algunos funcionarios ejecutivos de las empresas subsidiarias, de ingenieros petroleros del Departamento de Estudios de Operaciones, y de algunos viejos directores cuya iniciativa había salvado a la empresa en los difíciles años de la posguerra. En lo que respecta a sus características fundamentales, el plan recogía la experiencia de los diversos grupos de intereses, incorporaba apreciaciones subjetivas, y evitaba una excesiva rigidez que perjudicara la toma de decisiones a nivel de empresa subsidiaria. Era más bien un marco general de estrategia a corto plazo, que proporcionaba las orientaciones globales de la empresa, en la inteligencia de que las

experiencias que se acumularían durante el proceso de aplicación del plan, permitirían efectuar revisiones anuales a fin de pulir detalle afinar procedimientos, incorporar nuevos criterios y mas que nada, adecuarse a las condiciones cambiantes del mercado internacional de crudos y refinados.

Al finalizar el segundo año de operaciones bajo el nuevo plan, se notó que las cosas no marchaban como se había pensado. A pesar del proceso anual de revisión de metas, planes y políticas, la empresa comenzó a perder terreno frente al avance de empresas competidoras mas agresivas. Además los resultados generales de operación no eran muy brillantes, ya que en promedio se repetían casi los mismos valores de años anteriores en los índices financieros y económicos del análisis de resultados. Y por si esto fuera poco, los ejecutivos de las empresas subsidiarias se quejaban de que por lo general, el plan no era suficientemente detallado y flexible como para tomar en cuenta adecuadamente las condiciones específicas de operación de cada una de las mismas. En algunos casos, había ocurrido que iniciativas interesantes para el desarrollo de operaciones a nivel local, que hubieran permitido ganar mercados específicos en razón de crisis políticas internacionales, no pudieron ser aprovechadas oportunamente porque requerían inversiones que excedían los marcos generales delineados por el plan. El proceso de análisis de las nuevas circunstancias, hubiera requerido un lapso de tiempo tan dilatado, que la oportunidad se habría perdido de todas maneras.

Asi es que a mediados de 1970 se reunió el Consejo de Directores y resolvió contratar algunos especialistas para que se enfocara el problema integral de acuerdo con la técnica de análisis de sistemas, que se construyera un modelo de descripción, análisis y predicción, y que se recurriera a los métodos cuantitativos mas adecuados para la solución del modelo y elaboración del plan integral de operaciones.

Los especialistas en técnicas cuantitativas trabajaron en la realización de un diagnóstico detallado de la estructura de la empresa, las interrelaciones entre los diversos subsistemas, los principales problemas a encarar, etc., y sugirieron que la programación lineal parecía ser la técnica más adecuada para englobar todos los factores involucrados, optimizar los resultados de operación, tomar en cuenta condiciones específicas de operación de cada subsidiaria, y analizar nuevos planes y políticas cuando cambiaran las condiciones del mercado o variaran algunos supuestos iniciales del análisis.

El Consejo de Directores autorizó se continuara adelante en la elaboración de un modelo integral, por lo que se decidió aplicar la programación lineal a nivel global, resolver el modelo obteniendo un plan de operaciones, y analizar la sensitividad de la solución ante variaciones en los parámetros estimados. El plan se entregó para su aplicación al iniciarse el ejercicio económico de 1971; se propuso un período de un año para verificar resultados, comparar con el sistema anterior, recibir críticas y nuevas iniciativas, y pulir detalles no suficientemente elaborados. En el transcurso del año se perfeccionaría el modelo mediante el análisis de postoptimalidad, tratando de adecuarse progresivamente a la realidad, obteniendo la mayor información posible del plan óptimo propuesto, y analizando las modificaciones para adecuarse a cambios en las condiciones de operación. En el año siguiente, se tomarían en consideración los problemas específicos de la relación entre el plan integral y los planes de cada subsidiaria, se evaluarían todos los proyectos particulares dentro del encuadre global, y se obtendría el plan definitivo mediante un proceso de mutuo ajuste, hasta el logro de una compatibilidad total de todas las condiciones y restricciones.

En las páginas siguientes se presentan las consideraciones referentes a: estructura general de la empresa, operaciones de refinación, operaciones de distribución y operaciones de transporte.

Estructura y Operaciones Generales

La empresa P.I.S.A. cuenta con dos fuentes importantes de aprovisionamiento de petroleo crudo: Persia e Indonesia. El petróleo de Persia es el más pesado cuesta dos dólares por barril, se carga en el puerto petrolero de Abadan, y se tienen asegurados cien mil barriles diarios durante todo el año.

El petróleo de Indonesia se obtiene de los campos petroleros de Brunei. en la isla de Borneo; es mas liviano que el persa, cuesta 2.60 dólares por barril, y se han contratado cuarenta mil barriles diarios con la Compañía Petrolera Neerlandesa.

Hay dos empresas subsidiarias dedicadas exclusivamente a operaciones de refinación. Una de ellas tiene su sede en Australia, con una planta de refinación en Sidney, cuya capacidad es de cincuenta mil barriles diarios de proceso. La empresa distribuye sus refinados en el mercado australiano, y cuenta con excedentes disponibles que exporta a los demás mercados de la empresa.

La otra subsidiaria para operaciones de refinación está situada en el Japón, cerca de Nara, y su capacidad de refinación es de treinta mil barriles diarios. Abastece el mercado japonés y sus excedentes se exportan tambien a otros mercados.

Hay dos expresas subsidiarias dedicadas exclusivamente a operaciones de distribución: una de ellas en las Filipinas y la otra en Nueva Zeelandia. Los mercados que estas subsidiarias surten se proveen de productos refinados en Japón o en Australia, o en casos especiales de escasez transitoria, de la empresa subsidiaria de Santa Bárbara, California, U.S.A., que atiende regularmente el mercado de los Estados Unidos.

Finalmente, la empresa cuenta con una flota de buques-tanque, que utiliza para transportar crudos o refinados entre

las diversas subsidiarias de acuerdo con las necesidades específicas de cada lugar.

Operaciones de refinación

La operación de una refinería es un proceso complejo. Conviene utilizar un modelo de programación lineal para tomar en consideración todas las interrelaciones posibles. Los principales factores que influyen en la complejidad del proceso son las características de los petróleos crudos, la tecnología específica de la refinería, los diversos productos finales y subproductos del proceso, los niveles de producción, y las restricciones tecnológicas y económicas del proceso. En realidad, y a efectos de programar las operaciones sobre una base semanal (o a veces diaria), ambas refinerías deberían implantar un modelo de programación lineal de sus procesos. Los especialistas estimaron que esos modelos estarían integrados por aproximadamente 300 variables y unas 100 restricciones.

No obstante, con fines de planeación anual (y de mantener este problema dentro de un rango manejable) se pueden aceptar algunos supuestos simplificadores. Por ejemplo, supondremos sólo dos tipos de insumo: el petróleo crudo persa y el indonesio. Además reduciremos la gama de posibles productos del proceso a dos grandes variedades: productos de gasolina y productos de destilados (como el fuel-oil y aceites minerales diversos). En lo referente a tecnologías de proceso, si bien hay por lo general una amplia flexibilidad de selección de la intensidad del proceso, supondremos que a efectos de planeación se consideran sólo los casos extremos, es decir la tecnología de más alta intensidad y la de más baja intensidad de proceso.

En la tabla siguiente se indican los parámetros del proceso de refinación para ambas refinerías, de acuerdo con el tipo de petróleo crudo utilizado y con el producto final del proceso.

Tabla I

Parámetros tecnológicos del proceso de refinación de petróleo

Refinería, insumo y proceso utilizados	Producto final (en barriles de producto por barril de insumo)	
Refinería australiana:	Gasolinas	Destilados
Indonesio crudo, baja intensidad	.259	.688
Indonesio crudo, alta intensidad	.365	.573
Persa crudo, baja intensidad	.186	.732
Persa crudo, alta intensidad	.312	.608
Refinería japonesa:		
Indonesio crudo, baja intensidad	.259	.688
Indonesio crudo, alta intensidad	.350	.588
Persa crudo, baja intensidad	.186	.732
Persa crudo, alta intensidad	.300	.620

Operaciones de distribución

Las operaciones de distribución se efectúan en las áreas de Australia y Japón directamente, y en las de Filipinas y Nueva Zelandia, mediante el uso de buques-tanque. Por intermedio del Departamento de Estudios Económicos se han efectuado estimaciones de la demanda probable para el año 1971 en cada una de esas áreas.

Tabla II

Estimaciones para la demanda de gasolinas y destilados para las diversas áreas de distribución en 1971

Mercado de los productos	Demanda (en miles de barriles diarios) Gasolinas	Destilados
Australia	9.00	21.00
Japón	3.00	12.00
Filipinas	5.00	8.00
Nueva Zeelandia	5.36	8.68
Totales	22.36	49.68

Se dispone tambien de los costos promedio de transporte de gasolinas o destilados desde las refinerías a los mercados de - Filipinas y Nueva Zeelandia.

Tabla III

Costos de transporte de gasolinas o destilados (en dólares por barril de destilado)

Origen \ Destino	Nueva Zeelandia	Filipinas
Australia	.10	.15
Japón	.10	.20

Operaciones de transporte

Como se ha expresado anteriormente, los buques-tanque de la flota se utilizan para realizar transporte de petróleo crudo a - las refinerías, además de los transportes de productos refinados a los mercado de consumo. Los costos de transporte de petróleo - crudo desde Persia e Indonesia a las refinerías se indican a continuación, en la tabla IV.

Tabla IV

Costos variables de transporte de petróleo crudo
(en dólares por barril de crudo)

Tipo de insumo y de proceso	Refinería de Australia	Refinería de Japón
Indonesio crudo, baja intensidad	.26	.24
Indonesio crudo, alta intensidad	.26	.24
Persa crudo, baja intensidad	.62	.59
Persa crudo, alta intensidad	.62	.59

Debemos considerar además, la capacidad de la flota de buques-tanque y los requerimientos de transporte en términos de utilización de la capacidad. Se acostumbra a convertir los buques-tanque a fracciones de un buque prototipo equivalente, cuyo tonelaje es de 47 mil, medido en unidades de tonelaje de peso muerto, (DWT). En el caso particular de la flota que consideramos, la capacidad total es de 6.9 unidades equivalentes de 47 mil DWT. Además de la capacidad disponible, es factible utilizar los servicios de buques-tanque independientes, fletados por compañías transportistas internacionales. El costo promedio de transporte resulta ser de 3200 dólares por mil barriles diarios de despacho.

En lo referente a la capacidad de transporte requerida, se tiene que ésta varía de ruta a ruta segun diversos factores, que dependen de la distancia recorrida, uso de facilidades portuarias, etc. En la tabla V se indican las fracciones de buque-tanque equivalente, de 47 mil DWT, necesarias para despachar mil barriles diarios de petróleo crudo en las rutas ya indicadas.

Tabla V

Factores de uso de capacidad de buques-tanque
(en fracciones de buque-tanque equivalente)

Origen \ Destino	Australia	Japón
Persia	.12	.11
Indonesia	.05	.045
Filipinas	.02	.01
Nueva Zelandia	.01	.06

Otras fuentes de abastecimiento

Como ya hemos mencionado, existe una posibilidad adicional de surtir los mercados de la empresa con sobrantes transitorios de petróleo refinado o de destilados, provenientes de una planta situada en California. Para el año 1971, las estimaciones de producción y demanda para los Estados Unidos indicaban que se presentaría un excedente de 17 mil barriles diarios de destilados en esa planta, y que no habría excedentes de gasolina.

El costo del destilado producido en U.S.A. es de dos dólares por barril. Los costos de transporte a los mercados de distribución, y los requerimientos de capacidad de buques-tanque se indican en la tabla VI.

Tabla VI

Costos de transporte y requerimientos de capacidad para los destilados producidos en los U.S.A.

Destino	Costo de transporte	Requerimientos de capacidad
Nueva Zealandia	.70	.18
Filipinas	.55	.15

Otros costos

A fin de completar los datos económicos y tecnológicos necesarios para armar el modelo representativo de la operación simplificada del sistema, se incluyen en la Tabla VII los costos variables de refinación, que dependen de la intensidad del proceso y del tipo de insumo utilizado.

Tabla VII

Costos variables de refinación (en dólares por barril)

	Australia	Japón
Indonesio crudo, baja intensidad	.12	.16
Indonesio crudo, alta intensidad	.23	.34
Persa crudo, baja intensidad	.15	.20
Persa crudo, alta intensidad	.30	.35

Linear Programming Transportation and Distribution Analysis

PROGRAM NAME

TPORT$

April 196-
Reprinted 5 69

INFORMATION SYSTEMS

GENERAL ELECTRIC

Preface

This program is [illegible] of an ever-expanding library of time-sharing programs for use by subscribers to the [illegible] Time-Sharing Service of General Electric's Information Service Department Ea[illegible]ram is designated by its six-character name to store and retrieve it on the system [illegible]y programs are classified as on-line and run only.

- On-line [illegible]ry programs can be accessed from any terminal that is connected to the syst[illegible] The criteria for placing a library program on-line are its general utility a[illegible] frequency of use. Unless classified as run-only (see below), on-line programs can be listed, modified, copied, or run at the discretion of the user. To retrieve a program from the on-line library, its six-character name is used, followed by three asterisks. The three-asterisk suffix is not an integral part of the program name, it is only a requirement for retrieving the program. A brief description of LINPRO and LINEP$, which are additional on-line library programs concerned with linear programming, is available by listing CATLIN. List CATLOG for an index of all library programs.

- The off-line library consists of programs not in general demand. An index to the listings of off-line programs can be obtained by listing the library program CATOFF. Off-[illegible]e library programs are available for direct placement in a specific user catalog on request from your General Electric representative.

- Run-only is a term applied to programs that cannot be listed or permanently modified by the user. It is possible to have run-only programs in either the on-line or the off-line library. Run-only programs are designated by a dollar sign ($) in the sixth character position of the program name.

The terms under which library programs are made available to subscribers may vary between programs, or they may vary with a given program from time to time. General Electric reserves the right to change these terms at its discretion. Any questions regarding use of library programs should be directed to your General Electric representative.

Contents

FIGURES

Introduction

TPORT$ solves transportation or distribution problems where it is necessary to ship a homogeneous product from each of a number of shipping points, or sources, to each of a number of receiving points, or destinations. The unit shipping cost from each source to each destination is known. For a given time period, each destination has a specified requirement for the product. TPORT$ determines the quantities to be shipped from each source to each destination so that all demands are satisfied with the minimum, or optimum, total shipping cost.

The types of transportation problems solved by TPORT$ are those in which the total demand equals the total supply, or where a dummy destination or dummy source, restricted routes, or transhipment transportation problems are involved. TPORT$ can also be used to solve assignment problems, i.e., problems in which each required activity needs one, and only one, of the available resources and needs it exclusively.

Examples illustrating each of the above problems are included in Section 3 of this Guide.

The October 1968 reprint of TPORT$ does not render the original material obsolete.

Where,

$$T = \sum_{i=1}^{M} A_i = \sum_{j=1}^{N} B_j$$

The matrix form of the solution is:

$$\begin{array}{llll} X_{1,1} & X_{1,2} & \cdots & X_{1,N} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,N} \quad A_2 \\ \vdots & \vdots & & \vdots \\ X_{M,1} & X_{M,2} & \cdots & X_{M,N} \quad A_M \\ B_1 & B_2 & \cdots & B_N \quad T \end{array}$$

The minimum cost is then computed as:

$$\text{Minimum Cost} = \sum_{j=1}^{N} \sum_{i=1}^{M} C_{ij} X_{ij}$$

The technique used in this program to solve the transportation problem is the Primal-Dual Algorithm. This algorithm uses the Maximal Flow Algorithm which is an efficient algorithm for determining the maximum flow in Network Problems. The Primal-Dual Algorithm requires fewer computations in the solution of a transportation problem than the Stepping Stone Algorithm, and this efficiency is particularly advantageous when solving large transportation problems.[1]

This algorithm can also be applied to the assignment problem. To illustrate the use of the algorithm, the following distribution and assignment problems are given.

Distribution Problem

The following example shows how TPORT$ determines the minimum total shipping costs in a distribution problem.

A manufacturer has three plants in New York, Boston, and Baltimore, and five warehouses in Chicago, Cincinnati, Dallas, Atlanta, and Akron, all handling Product Z. For a given time period, individual plant supply capacities, individual warehouse requirements, and shipping costs per unit of Product Z from each plant to each warehouse are.

Plants	Shipping Costs (In Dollars)/Unit Warehouses					Plant Supply Capacity
	Chicago	Cincinnati	Dallas	Atlanta	Akron	
New York	4	6	7	4	6	1000
Boston	7	5	8	5	8	800
Baltimore	6	4	6	7	5	600
Warehouse Requirement	400	700	300	500	500	2400

1. See Hadley, G., *Linear Programming*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1963, Pages 351-359.

1. General Description

The objective of the solution for the transportation problem is to minimize the cost function. Mathematically, the problem is formulated as follows.

PROGRAM MATHEMATICS

Consider a transportation problem with M sources and N destinations. Let i refer to source and j to destinations, and X_{ij} to the units shipped from source i to destination j. The following relations can then be written.

$$\sum_{j=1}^{N} X_{ij} = X_{i1} + X_{i2} + \ldots \qquad i = 1, \ldots, M$$

$$\sum_{i=1}^{M} X_{ij} = X_{1j} + \ldots X_{Mj} = B_j \qquad j = 1, \ldots, N$$

where,

A_i = number of product units available at source i
B_j = number of product units desired at destination j

Let C_{ij} represent the cost of transporting one unit from source i to destination j. Then the problem is one of determining the X_{ij} values that will minimize the total shipping costs, i.e., it is desired to minimize the function Z where,

$$Z = \sum_{j=1}^{N} \sum_{i=1}^{M} C_{ij} X_{ij}$$

In addition, the following relationship must be satisfied

$$\sum_{i=1}^{M} A_i = \sum_{j=1}^{N} B_j$$

The transportation problem is also represented in the following matrix form and it is the form used in this Guide.

$$\begin{matrix} C_{1,1} & C_{1,2} & \ldots & C_{1,N} & A_1 \\ C_{2,1} & C_{2,2} & \ldots & C_{2,N} & A_2 \\ \vdots & \vdots & & \vdots & \vdots \\ C_{M,1} & C_{M,2} & \ldots & C_{M,N} & A_M \\ B_1 & B_2 & \ldots & B_N & T \end{matrix}$$

where,

$$T = \sum_{i=1}^{M} A_i = \sum_{j=1}^{N} B_j$$

The matrix form of the solution is:

$$\begin{matrix} X_{1,1} & X_{1,2} & \cdots & X_{1,N} & A_1 \\ X_{2,1} & X_{2,2} & \cdots & X_{2,N} & A_2 \\ \vdots & \vdots & & \vdots & \vdots \\ X_{M,1} & X_{M,2} & \cdots & X_{M,N} & A_M \\ B_1 & B_2 & \cdots & B_N & T \end{matrix}$$

The minimum cost is then computed as:

$$\text{Minimum Cost} = \sum_{j=1}^{N} \sum_{i=1}^{M} C_{ij} X_{ij}$$

The technique used in this program to solve the transportation problem is the Primal-Dual Algorithm. This algorithm uses the Maximal Flow Algorithm which is an efficient algorithm for determining the maximum flow in Network Problems. The Primal-Dual Algorithm requires fewer computations in the solution of a transportation problem than the Stepping Stone Algorithm, and this efficiency is particularly advantageous when solving large transportation problems.[1]

This algorithm can also be applied to the assignment problem. To illustrate the use of the algorithm, the following distribution and assignment problems are given.

Distribution Problem

The following example shows how TPORT$ determines the minimum total shipping costs in a distribution problem.

A manufacturer has three plants in New York, Boston, and Baltimore, and five warehouses in Chicago, Cincinnati, Dallas, Atlanta, and Akron, all handling Product Z. For a given time period, individual plant supply capacities, individual warehouse requirements, and shipping costs per unit of Product Z from each plant to each warehouse are:

Plants	Shipping Costs (In Dollars)/Unit Warehouses					Plant Supply Capacity
	Chicago	Cincinnati	Dallas	Atlanta	Akron	
New York	4	6	7	4	6	1000
Boston	7	5	8	5	8	800
Baltimore	6	4	6	7	5	600
Warehouse Requirement	400	700	300	500	500	2400

1. See Hadley, G., *Linear Programming*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1963, Pages 351-359.

For this type of distribution problem, TPORT$ computes the minimum total shipping costs for supplying all warehouse requirements, by determining the quantities in shipments from plants, subject to supply capacity of each plant. For this example, the following computations are obtained.

Plants	Quantities To Be Shipped To Warehouses					Total Plant Shipment
	Warehouses					
	Chicago	Cincinnati	Dallas	Atlanta	Akron	
New York	400	0	0	400	200	1000
Boston	0	700	0	100	0	800
Baltimore	0	0	300	0	300	600
Warehouse Requirement	400	700	300	500	500	2400

Minimum Shipping Cost = $11,700

For an illustration of the input data format and the final printed output of this example, see Solution 1, Section 3 of this Guide.

Assignment Problem

TPORT$ also solves those problems involving the assignment of resources to jobs. The restrictions of this type of problem are such that each resource can be assigned to only one job and, conversely, each job can have only one resource. When solved by TPORT$, a dummy resource or job is added if the number of resources does not equal the number of jobs.

Mathematically, the requirement is to find a matrix such that:

$$\sum_{j=1}^{N} X_{ij} = 1 \qquad i = 1, \ldots, N$$

$$\sum_{i=1}^{N} X_{ij} = 1 \qquad j = 1, \ldots, N$$

This matrix should also minimize the function Z where,

$$Z = \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} X_{ij}$$

In this case, the cost matrix (composed of the C_{ij} elements) may represent costs or ratings. There may be different methods of determining the effectiveness associated with each combination of job and resource, i.e., the determining factor may be cost or it may be a rating matrix. The problem is to maximize the total effectiveness of doing all the jobs, e.g., minimize the total cost, where cost is the measure of effectiveness[2].

The following example illustrates how TPORT$ determines the minimum cost in an assignment problem.

Jobs A, B, and C are to be assigned to Machines 1, 2, and 3, in any combination, on a one-for-one basis. The costs associated with doing any one job on any one machine are:

2. For more detailed information, see Llewellyn, R. W., *Linear Programming*, Holt, Rinehart and Winston, New York, New York, 1966, Pages 326-335, or any standard linear programming test.

	Machine 1	Machine 2	Machine 3	Job Requirements
Job A	$220	$240	$260	1
Job B	230	280	210	1
Job C	250	290	330	1
Machine Resources	1	1	1	3

The objective is to determine the machines to which Jobs A, B, and C should be assigned simultaneously on a one-for-one basis, so as to minimize the total cost of performing all three jobs.

The solution computed by TPORT$ will be: assign Job A to Machine 2, Job B to Machine 3, and Job C to Machine 1. The total cost is $700.

For an illustration of the input data format and the final printed output of this example, see Solution 6, Section 3 of this Guide.

PROGRAM REDIMENSION

TPORT$ reserves sufficient storage for a cost matrix of 990 elements. As the program is presently written, this cost matrix is dimensioned for 30 rows and 33 columns. If required, the cost matrix can be redimensioned as long as the inequality, $M \cdot N \leq 990$, is satisfied, where M is the number of rows and N is the number of columns in the new cost matrix.

Using these definitions, the following program statements should be inserted.

```
110 INTEGER A(M), B(N), C(M,N), X"
190 INTEGER DUM(1), LS(M+N-1), DUN(1), LSV(N)
200 INTEGER A(M), B(N), C(M,N), X"
210 INTEGER G(M), LISTU", NLV", R(N), LISTV", NL(M·N)
```

For example, if the cost matrix had 20 rows and 40 columns, the above inequality would be satisfied and the program statements would be:

```
110 INTEGER A(20), B(40), C(20,40), X"
190 INTEGER DUM(1) LS(59), DUN(1), LSV(40)
200 INTEGER A(20), B(40), C(20,40), X"
210 INTEGER G(20), LISTU", NLV", R(40), LISTV", NL(800)
```

LIMITATIONS

All input data must be entered as integers, including elements of the cost matrix.

If, in the formulation of the cost matrix, costs are considered in terms of dollars and cents, the elements of the cost matrix must be multiplied by 100 and entered as integers. For example, if an element of the cost matrix is 2.83, it would be entered as 283. The program, however, will print the correct optimum cost.

No quantity, whether source or destination, or element of the cost matrix can be larger than the integer 500,000.

TPORT$ solves for one optimum solution. Multiple solutions are not considered.

The program solves those problems for which the elements of the cost matrix represent costs, not profit, i.e., the maximization problem.

A restricted route must be represented in the cost matrix by a large integer, such as 500,000 (or 5E5).

The elements of the cost matrix must represent the cost of shipping one unit from each source to each destination.

2. Operating Instructions

TPORT$ is written in Time-Sharing FORTRAN and uses the FORTRAN file capability to process the input data.

If the file were named TRAN, the program would be called and executed as follows. Supply all underlined information.

```
SYSTEM--FØRTRAN
NEW ØR ØLD--NEW
NEW FILE NAME--TRAN                    Name the file.
READY.
   :
   :                                   Create the input data file(s).
SAVE                                   Save the data file(s).

ØLD
ØLD FILE NAME--TPØRT$***               Call the program.
READY

100 +TRAN                              Recall the file(s).
RUN                                    Execute the program.
```

INPUT DATA

Data is entered using a FORTRAN file in the following format, where each line number is followed by a blank character; a comma or blank character must separate each item of data.

```
100 LABEL
```

where LABEL is an identification device limited to 45 characters, especially useful when consecutive data cases are created.

```
101 M,N
```

where,

M = number of sources
N = number of destinations

```
102 A(1),A(2),A(3),...,A(M)
```

where,

A(I),I=1,M, represent the quantity available at sources 1, 2, 3,..., M

```
103 B(1),B(2),B(3),...,B(N)
```

where,

B(J),J=1,N, represent the quantity required at destinations 1, 2, 3, . . . ,N

104 C(1,1),C(1,2),C(1,3), . . . ,C(1,N)
105 C(2,1),C(2,2),C(2,3), . . . ,C(2,N)
.
.
.
1XX C(M,1),C(M,2),C(M,3), . . . ,C(M,N)

where,

C(I,J),I=1,M and J=1,N, represent the elements of the cost matrix. Note that the elements are entered row-wise.

1YY INT

where,

INT = 0 if the elements of the cost matrix are in terms of dollars only.
INT = 1 if the elements of the cost matrix are in terms of dollars and cents and have been multiplied by 100.

1ZZ END

where,

END defines the last data case. If this label is not provided, the FORTRAN compiler diagnostic ØUT ØF DATA will be printed.

CONSECUTIVE DATA CASES

TPORT$ has the capability of processing consecutive data cases. The format for each data case is the same as described above except that the label END is the last entry in the last data file only. Figure 1 illustrates two consecutive data cases. Note that Lines 100 and 110 identify the data cases as belonging to the first and seventh sample problems described in the next section.

```
100 SAMPLE PRØBLEM 1
101 3 5
102 1000 800 600
103 400 700 300 500 500
104 4 6 7 4 6
105 7 5 8 5 6
106 6 4 6 7 5
107 0
110 SAMPLE PRØBLEM 7
111 4 7
112 1000 2000 1000 3000
113 800 1200 2500 500 600 900 500
114 251 321 256 326 5E5 5E5 0
115 325 266 330 271 335 276 0
116 5E5 5E5 300 240 305 245 0
117 5E5 5E5 180 200 190 210 0
118 1
119 END
```

Figure 1. Sample Input Data File.

Line 101 of the first data case and Line 111 of the second data case give the number of sources and the number of destinations for each problem. Lines 102 and 112 of each data case contain the quantity available at each source, while the Lines 103 and 113 contain the quantity required at each destination. The following three lines of the first data case and the following four lines of the second data case contain the elements of the cost matrix. Line 107 indicates that the elements are in dollars only, while Line 118 indicates that the elements of the cost matrix in the second data case are in terms of dollars and cents have been multiplied by 100. Computation for the file is terminated by the entry in Line 119.

OUTPUT DATA

The program first [illegible] a label to identify the solution, as provided by the first line of each input data case [illegible] from each source, the quantities to be provided for each destination are printed. Note [illegible] the program does not print zero quantities. After all quantities have been printed for [illegible] sources and destinations, the minimum cost is printed, and the program proceeds to the [illegible] consecutive case, until the data line indicating END is read.

3. Sample Problems and Solutions

In this section, sample problems and solutions are given for distribution and assignment problems handled by TPORT$. Specifically, the problems are:

- Those distribution problems in which the total demands equal the total supplies.
- Problems where a dummy destination is required.
- Problems where a dummy source is required.
- Problems of restricted routes.
- Problems of transhipment transportation.
- Problems of assignment of jobs.
- Problems of assignment of personnel.

It would be possible to enter all of these problems in a consecutive input data file; however, for purposes of discussion, these problems have been solved separately. In each solution, the underlined entries are those which must be initiated at the teletypewriter.

PROBLEM 1

In this problem the total demands equal the total supplies. The problem, in matrix form is:

DESTINATIONS (IN DOLLARS) COST MATRIX

Sources	D-1	D-2	D-3	D-4	D-5	Supply
S-1	4	6	7	4	6	1000
S-2	7	5	8	5	8	800
S-3	6	4	6	7	5	600
Demand	400	700	300	500	500	2400

SOLUTION 1

```
NEW
NEW FILE NAME--TRAN1
READY.

100 TOTAL DEMANDS EQUAL TOTAL SUPPLIES
101 3 5
102 1000 800 600
103 400 700 300 500 500
104 4 6 7 4 6
105 7 5 8 5 8
106 6 4 6 7 5
107 0
108 END

SAVE


READY.
```

```
ØLD
ØLD FILE NAME--TPØRT$***

READY.

100 + TRAN1
RUN

TPØRT$


TØTAL DEMANDS EQUAL TØTAL SUPPLIES

SØURCE        1

                                QUANTITY
   DESTINATIØN        1            400
   DESTINATIØN        4            400
   DESTINATIØN        5            200

SØURCE        2

                                QUANTITY
   DESTINATIØN        2            700
   DESTINATIØN        4            100

SØURCE        3

                                QUANTITY
   DESTINATIØN        3            300
   DESTINATIØN        5            300


MINIMUM CØST =        11700.00
```

PROBLEM 2

This problem illustrates the use of a dummy destination. In the figure below, the total of the supplies at the three sources is 2400 units, and the total of the five destinations is 2300 units It is obvious, then, that 100 units of the product will remain at one or more of the sources after all demands are satisfied. A dummy destination with a demand of 100 units is incorporated into the problem so that the problem will fit the transportation model. For this problem it is assumed that any or all of the sources may retain the excess supply without any extra assigned costs, and, therefore, the dummy destination cost coefficients are all zero.

DESTINATIONS (IN DOLLARS) COST MATRIX

Sources	D-1	D-2	D-3	D-4	D-5	Dummy	Supply
S-1	4	6	7	4	6	0	1000
S-2	7	5	8	5	8	0	800
S-3	6	4	6	7	5	0	600
Demand	400	700	300	400	500	100	2400

SOLUTION 2

```
NEW
NEW FILE NAME--TRAN2
READY.

100 PRØBLEM USING DUMMY DESTINATIØN
101 3 6
102 1000 800 600
103 400 700 300 400 500 100
104 4 6 7 4 5 0
105 7 5 8 5 3 0
106 6 4 6 7 5 0
107 0
108 END

SAVE

READY.


ØLD
ØLD FILE NAME--TPØRT$***

READY.

100 * TRAN2
RUN

TPØRT$


PRØBLEM USING DUMMY DESTINATIØN

SØURCE        1

                                QUANTITY
   DESTINATIØN          1          400
   DESTINATIØN          3          200
   DESTINATIØN          4          400

SØURCE        2

                                QUANTITY
   DESTINATIØN          2          700
   DESTINATIØN          6          100

SØURCE        3

                                QUANTITY
   DESTINATIØN          3          100
   DESTINATIØN          5          500


MINIMUM CØST =        11200.00
```

Note that the extra 100 units are assigned to Destination 6, the dummy destination, from Source 2; therefore, the extra 100 units remain at Source 2.

PROBLEM 3

This problem illustrates the use of a dummy source. In the figure below the total of the supplies at the three sources is 2300 units, and the total of the demands at the five destinations is 2400 units. It is apparent that all of the demands cannot be satisfied. Therefore, a dummy source, with a supply of 100 units, is incorporated into the problem, so the problem will fit the transportation model. Any allocations from this dummy source represent the unsatisfied demands. The cost coefficients assigned to the dummy source are assumed to be zero for this problem.

DESTINATION (IN DOLLARS) COST MATRIX

Sources	D-1	D-2	D-3	D-4	D-5	Supply
S-1	4	6	7	4	6	900
S-2	7	5	8	5	8	800
S-3	6	4	6	7	5	600
Dummy	0	0	0	0	0	100
Demand	400	700	300	500	500	2400

SOLUTION 3

```
NEW
NEW FILE NAME--TRAN3
READY.

 100 PRØBLEM USING DUMMY SØURCE
 101 4 5
 102 900 800 600 100
 103 400 700 300 500 500
 104 4 6 7 4 6
 105 7 5 8 5 8
 106 6 4 6 7 5
 107 0 0 0 0 0
 108 0
 109 END

SAVE

READY.

ØLD
ØLD FILE NAME--TPØRT$***

READY.

100 * TRAN3
RUN


TPØRT$


PRØBLEM USING DUMMY SØURCE

SØURCE          1

                                QUANTITY
   DESTINATION          1          400
   DESTINATION          4          500
```

```
SØURCE        2

                                QUANTITY
   DESTINATIØN        2          700
   DESTINATIØN        3          100


SØURCE        3

                                QUANTITY
   DESTINATIØN        3          100
   DESTINATIØN        5          500


SØURCE        4

                                QUANTITY
   DESTINATIØN        3          100


MINIMUM CØST =        11000.00
```

Note that Destination 3 requires the additional 100 units.

PROBLEM 4

This problem illustrates the use of restricted routes. The figure below indicates that certain conditions prohibit any shipments from source 1 to destinations 4 and 5 and from source 3 to destinations 1 and 2. Note that these restrictions are represented by the number 5E5 in the cost matrix. Note, also, that the other elements of the cost matrix are expressed in decimal form and that this problem illustrates the use of this program when costs are represented as dollars and cents.

DESTINATIONS (IN DOLLARS AND CENTS) COST MATRIX

Sources	D-1	D-2	D-3	D-4	D-5	Supply
S-1	4.05	6.56	7.00	5E5	5E5	900
S-2	7.23	5.66	8.07	5.00	8.26	800
S-3	5E5	5E5	6.25	7.11	5.95	700
Demand	400	700	300	500	500	2400

SOLUTION 4

```
NEW
NEW FILE NAME--TRAN4
READY.

100 PRØBLEM WITH RETRICTED RØUTES
101 3 5
102 900 800 700
103 400 700 300 500 500
104 405 656 700 5E5 5E5
105 723 566 807 500 826
106 5E5 5E5 625 711 595
107 1
108 END

SAVE

READY.
```

```
OLD
OLD FILE NAME--TPORTS***

READY.

100 * TRAN4
RUN

TPORTS

PROBLEM WITH RETRICTED ROUTES


SOURCE        1

                                QUANTITY
   DESTINATION        1          400
   DESTINATION        2          400
   DESTINATION        3          100


SOURCE        2

                                QUANTITY
   DESTINATION        2          300
   DESTINATION        4          500


SOURCE        3

                                QUANTITY
   DESTINATION        3          200
   DESTINATION        5          500


MINIMUM COST =        13367.00
```

PROBLEM 5

This sample problem illustrates the use of the transportation model to solve the transhipment transportation problem. The transhipment problem differs from the conventional transportation problem in that shipments are permissible from more than one source or destination. This difference can be illustrated as follows.

Given 3 sources and 4 destinations a diagram of possible shipment flows for the conventional transportation program would be as shown in Figure 2.

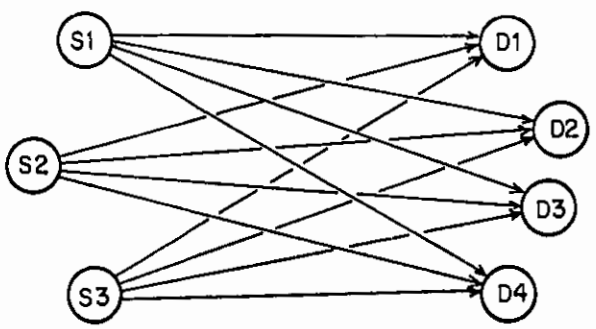


Figure 2. Conventional Transportation Problem.

Note that the flows are from sources to destinations only.

In the transportation problem where the shipment flows are permissible from any source or destination, to other sources or destinations, or to itself, the diagram of possible shipment flows would be as shown in Figure 3.

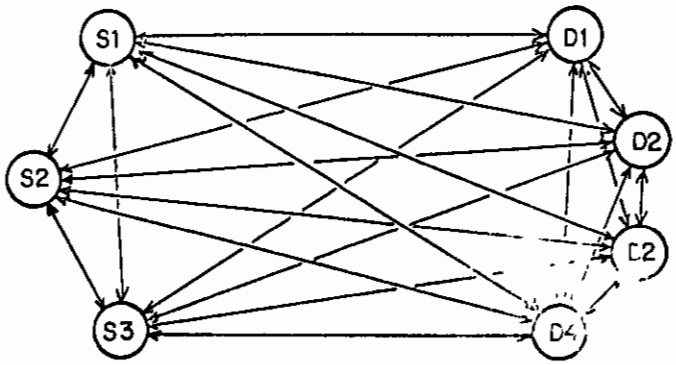


Figure 3. Diagram of Possible Shipment Flows

Note that in this case, each source and destination has 6 possible flows in either direction; the seventh flow would be from each source or destination to itself.

The following problem illustrates the transhipment problem for 3 sources and 4 destinations. The cost matrices for costs of shipping from source to destination, from source to source, and from destination to destination are:

DESTINATION (IN DOLLARS)

Sources	D-1	D-2	D-3	D-4
S-1	10	5	6	7
S-2	8	2	7	6
S-3	9	3	4	8

SOURCES (IN DOLLARS)

Sources	S-1	S-2	S-3
S-1	0	3	2
S-2	1	0	4
S-3	3	3	0

DESTINATIONS (IN DOLLARS)

Destinations	D-1	D-2	D-3	D-4
D-1	0	4	3	2
D-2	4	0	1	2
D-3	3	1	0	1
D-4	2	2	1	0

Note that shipping costs in the opposite direction are the same.

The requirements are 15 units at D-1, 20 units at D-2, 30 units at D-3, and 35 units at D-4. The supplies are 25 units at S-1, 25 units at S-2, and 50 units at S-3.

Referring to Figure 3, it is noted that the maximum number of units that might be shipped via source S-1 is the 25 units available at that source plus the number that may be transhipped via S-1. The number of units that can be conceivably transhipped via source 1 is the total number of units available at all sources. This condition applies for all sources and destinations. In this problem, the total number of units available at the 3 sources and required at the 4 destinations is 100. Therefore, 100 units must be added to the number of units required at each destination and available at each source since the shipment of all the units could pass through any source of destination.

A detailed mathematical description of the transhipment problem is given in pages 368 through 373 of Reference 1.

Condensing the above information to one matrix required as input by this program, the matrix is:

DESTINATIONS (IN DOLLARS) COST MATRIX

Sources	S-1	S-2	S-3	D-1	D-2	D-3	D-4	Supply
S-1	0	[illegible]	[illegible]	10	5	6	7	125
S-2	1	[illegible]	4	8	2	7	6	125
S-3	3	[illegible]	0	9	3	4	8	150
D-1	10	[illegible]	9	0	4	3	2	100
D-2	[illegible]	2	3	4	0	1	2	100
D-3	[illegible]	7	4	3	1	0	1	100
D-4	[illegible]	6	8	2	2	1	0	100
Demand	100	100	100	115	120	130	135	800

Note that the supplies and demands have been increased by 100 units. Note also that the main diagonal of the cost matrix contains zeroes since the cost of shipping to the same source or destination is zero.

SOLUTION 5

```
NEW
NEW FILE NAME--TRANS
READY.

100 TRANSHIPMENT TRANSPØRTATIØN PRØBLEM
101 7 7
102 125 125 150 100 100 100 100
103 100 100 100 115 120 130 135
104 0 3 2 10 5 6 7
105 1 0 4 8 2 7 6
106 3 3 0 9 3 4
107 10 8 9 0 4 [illegible]
108 5 2 3 4 0 [illegible]
109 6 7 4 3 1 [illegible]
110 7 6 8 2 2 [illegible] 0
111 0
112 END
SAVE
READY.

ØLD
ØLD FILE NAME--TPØRTS***

READY.

100 * TRANS
RUN

TPØRTS


TRANSHIPMENT TRANSPØRTATIØN PRØBLEM


SØURCE          1

                                ØUANTITY
   DESTINATIØN          1          100
   DESTINATIØN                     25
```

```
SOURCE        2

                              QUANTITY
   DESTINATION        2          100
   DESTINATION        5           25

SOURCE        3

                              QUANTITY
   DESTINATION        3          100
   DESTINATION        6           50

SOURCE        4

                              QUANTITY
   DESTINATION        4          100

SOURCE        5

                              QUANTITY
   DESTINATION        5           95
   DESTINATION        6            5

SOURCE        6

                              QUANTITY
   DESTINATION        6           75
   DESTINATION        7           25

SOURCE        7

                              QUANTITY
   DESTINATION        4           15
   DESTINATION        7           85


MINIMUM COST =          485.00
```

In matrix form, the solution is:

DESTINATIONS (QUANTITIES)

Sources	S-1	S-2	S-3	D-1	D-2	D-3	D-4	Supply
S-1	100	0	0	0	0	0	25	125
S-2	0	100	0	0	25	0	0	125
S-3	0	0	100	0	0	50	0	150
D-1	0	0	0	100	0	0	0	100
D-2	0	0	0	0	95	5	0	100
D-3	0	0	0	0	0	75	25	100
D-4	0	0	0	15	0	0	85	100
Demand	100	100	100	115	120	130	135	800

The quantities on the major diagonal of this matrix represent the difference between the maximum number of units that could have been shipped and the quantity that actually is shipped. These major diagonal quantities represent slack variables introduced into the problem in order for transhipment to occur, and they should be ignored when interpre-

ing the solution. Discarding these quantities and interpreting the above matrix row-wise, the solution is:

Ship 25 units from S-1 to D-4
Ship 25 units from S-2 to D-2
Ship 50 units from S-3 to D-3

Ship 5 units from D-2 to D-3
Ship 25 units from D-3 to D-4
Ship 15 units from D-4 to D-1

According to this schedule, the units remaining at each destination would be:

D-1 – 15 units
D-2 – 20 units
D-3 – 30 units
D-4 – 35 units

the original requirements. The minimum (or optimum) cost for shipping according to this schedule is $485.00.

In general, this program can be used to solve transhipment problems where a dummy source or destination is required, where route restrictions must be considered, and problems where partial transhipment is involved.

PROBLEM 6

This sample problem illustrates the use of the transportation model to obtain a solution to the assignment problem.

Given the assignment problem shown in the figure below, the objective is to determine the machines to which Jobs A, B, and C should be assigned simultaneously on a one-for-one basis so as to minimize the total cost of performing all three jobs.

COST MATRIX (IN DOLLARS)

Jobs	Machine 1	Machine 2	Machine 3	Job Requirements
Job A	220	240	260	1
Job B	230	280	210	1
Job C	250	290	330	1
Machine Resources	1	1	1	3

SOLUTION 6

```
NEW
NEW FILE NAME--TRAN6
READY.

100 THE ASSIGNMENT PRØBLEM
101 3 3
102 1 1 1
103 1 1 1
104 220 240 260
105 230 280 210
106 250 290 330
107 0
108 END

SAVE

READY.
```

```
OLD
OLD FILE NAME--TPORTS**

READY.

100 * TRAN6
RUN

TPORTS

THE ASSIGNMENT PROBLEM

SOURCE          1
   DESTINATION          2          QUANTITY          1

SOURCE          2
   DESTINATION          3          QUANTITY          1

SOURCE          3
   DESTINATION          1          QUANTITY          1

MINIMUM COST =          700.00
```

Relating the program solution to the original problem formulation, the following is obtained:

ASSIGNMENT MATRIX

Jobs	Machine 1	Machine 2	Machine 3	Job Requirements
Job A	0	1	0	1
Job B	0	0	1	1
Job C	1	0	0	1
Machine Resources	1	1	1	3

Minimum Cost = $700

PROBLEM 7

This sample problem illustrates the use of the transportation model to obtain a solution to the general personnel assignment problem.

In the general personnel assignment problem, it can be said that there are M personnel categories with A_i individuals in category i. There are N job types, and B_j individuals in job type j. No individual can be assigned to more than one job, but any individual can be assigned to any job. However, individuals in different personnel categories may have different degrees of efficiency in the same job. The differences in efficiency may be measured in terms of cost or a rating system might be used (Reference 1).

Given the following generalized personnel assignment problem, the objective is to determine the minimum cost of assigning the required engineers to projects

COST MATRIX (IN DOLLARS)

Category of Engineer	Project 1	Project 2	Pr	Number of Available Engineers
1	8.40	7[illegible]		10
2	7.00	[illegible].00		15
3	6.72	7[illegible].28	[illegible]2	12
Number of Engineers Required by Project	[illegible]	13	[illegible]1	37

[illegible]. It costs $[illegible]40 for any engineer in category 1 to work on project 1, it costs $6.72 for [illegible]neer in category [illegible] to work on project 2, etc.

TION 7

```
[illegible] FILE NAME--TRAN7
READY.

100 GENERAL PERSONNEL ASSIGNMENT PROBLEM
101 3 3
102 10 15 12
103 13 13 11
104 640 672 840
105 700 700 112
106 672 728 728
107 1
108 END

SAVE

READY.

OLD
OLD FILE NAME--TPORTS***

READY.

100 * TRAN7
RUN


TPORTS--

GENERAL PERSONNEL ASSIGNMENT PROBLEM


SOURCE        1

                                QUANTITY
   DESTINATION        2            10


SOURCE        2

                                QUANTITY
   DESTINATION        1             1
   DESTINATION        2             3
   DESTINATION        3            11
```

SOURCE 3

DESTINATION	1	QUANTITY 12

MINIMUM COST = 188.16

That is:

1. 10 engineers of Category 1 are assigned to Project 2.
2. 1 engineer of Category 2 is assigned to Project 1, 3 engineers assigned to Project 2, and 12 engineers of this category assigned to Project 3.
3. 12 engineers of Category 3 are assigned to Project 1.

At a minimum cost of $188.16.

904214A

USER'S GUIDE

Linear Programming Methods

LINE1$***
LINPR$***
SENSI$***

REPRINTED JANUARY 1971

GENERAL ELECTRIC

INFORMATION SERVICE DEPARTMENT

Typically, a linear programming (LP) problem involves the optimizing of a numerical linear function of a number of variables, each subject to certain constraints. A systematic procedure, called the simplex method, has been developed for solving these linear constraints This iterative method is readily adaptable to modern high-speed computers, it was first used in theoretical investigations, then expanded into the petroleum, chemical, transportation, agricultural, manufacturing, and other types of industries It has been used for solving problems of inventory, assignment, and nutrition.

In order to use the MARK I LP programs outlined in this guide, you do not need to understand the sophisticated linear programming technique, the simplex method, or to be a mathematician. You should know, however, the elements of this method (in order to formulate your problem most effectively and to interpret the results) and its limitations Sample Problems are given to illustrate the process of entering data and running the programs listed as follows:

- LINE1$ which is written in Time-Sharing FORTRAN and employs the two-phase simplex method The program can handle 28 constraints and 59 variables, including surplus A program redimension option is available.

- LINPR$ which uses the two-phase simplex algorithm and is written in Time-Sharing BASIC The program handles a maximum of 17 constraints and 29 variables, including surplus. You can run larger problems by using the program redimension option.

- SENSI$ which is written in Time-Sharing FORTRAN solves a linear programming problem and performs a sensitivity analysis on different parameters of the formulation. The program can handle a maximum of 20 constraints and 40 variables, including surplus, slack, and artificial.

Following is a list of other General Electric MARK I publications concerning linear programming.

Publication Number	Title
904213	Linear Programming Applications Reference Manual
904215	Integer Linear Programming User's Guide
804218	Transportation and Distribution Analysis User's Guide
AR-6	Zero-One Integer Linear Programming

As a further aid to your understanding of linear programming, a glossary of terms is given at the end of this guide.

This user's guide includes the programs LINEIS***, LINPR$***, and SENSI$*** which are designed primarily for the solving of LP problems using the simplex method and for performing a sensitivity analysis of the results. The programs are written in MARK I, FORTRAN, BASIC, and FORTRAN, respectively, and are part of an ever-expanding library of time-sharing programs for use by subscribers to the Time-Sharing Service.

Users need not be programmers. However, familiarity with the system is required. The Time-Sharing System Manual (publication number CPB-6-MARK I) provides such information, and the manual should be used in conjunction with this guide.

Listings of programs and routines in the library are published in the Program Library Index (publication number 800000). The extent to which library programs are made available to subscribers may vary between countries and may vary with a given program from time to time. General Electric reserves the right to distribute these terms at its discretion. Any questions regarding use of the library programs should be directed to your General Electric representative.

TABLE OF CONTENTS

LIST OF TABLES

LINE1$ and LINPR$ are written in Time-Sharing FORTRAN and BASIC languages, respectively, and seek to optimize (maximize or minimize) a numerical linear function of a number of variables, with the variables subject to certain constraints. Both of the programs use the two-phase simplex technique to solve a linear programming problem. The general idea of the two-phase method is that the problem is solved in two parts. The first phase drives all artificial variables to zero and produces a basic feasible solution In phase two, the actual objective function is optimized, starting from a basic feasible solution which contains no artificial variables.

METHOD

The simplex method proceeds in systematic steps from an initial basic feasible solution to other basic feasible solutions and, finally, in a finite number of steps to an optimal basic feasible solution in such a way that the value of the objective function at each iteration is better (or, at least, not worse) than at the preceding step.

The linear programming (LP) method determines a solution (if one exists) to the following problem:

Optimize $Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$

Subject to the constraints:

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &\leq b_2 \end{aligned}\right\} \text{"less than" constraints}$$

$$\vdots$$

$$\left.\begin{aligned} a_{g1}x_1 + a_{g2}x_2 + \ldots + a_{gn}x_n &= b_g \\ a_{h1}x_1 + a_{h2}x_2 + \ldots + a_{hn}x_n &= b_h \end{aligned}\right\} \text{" qual to" constraints}$$

$$\vdots$$

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\geq b_1 \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\geq b_m \end{aligned}\right\} \text{"greater than" constraints.}$$

The preceding problem can be rewritten in compact form as

$$\bar{A}\,\bar{X} = \bar{a}_1x_1 + \bar{a}_2x_2 + \bar{a}_3x_3 + \ldots + \bar{a}_nx_n \gtreqless \bar{B}$$

where

$\bar{a}_j = (a_{1j}, a_{2j}, \ldots, a_{mj})$ $j = 1, 2, \ldots, n$

Optimize $Z = \bar{C}\bar{X}$

The vector $\bar{B}$ is often referred to as the requirement vector, the $\bar{a}_j$ are called activity vectors and vector $\bar{c}$ is called the cost-coefficient vector.

For further details of the computational method you may refer to either of the following:

(1) Llewellyn, R.W. Linear Programming. Holt, Rinehart and Winston, Inc., New York. 1966.

(2) G. Hadley, Addison-Wesley. Linear Programming. Wesley Publishing Corp., Inc., Mass. 1963.

LIMITATIONS

LINE1$ contains the following limitations. For problems with more variables and fewer constraints (or vice versa) than the following specify, see the paragraphs under "Program Redimension Capability."

1. Tht total of less than, equal to, and greater than constraints should be equal to, or less than, 28, that is,

   $KL + KE + KG \leq 28$

2. The total of greater than constraints and the number of variables should be less than, or equal to, 59, that is,

   $KG + N \leq 59$

3. All right-hand side constants must be greater than, or equal to, zero.
   If a problem formulation results in a right-hand side constant which is negative, a reformulation after multiplying the entire equation by -1 will result in a positive right-hand side constant.

4. Values of all variables must be greater than, or equal to, zero, that is,

   $X \leq 0$

   where X is the array of variables in $A X \lesseqgtr b$
   If a particular problem formulation results in one or more variables which may take on either positive or negative values, the problem may be reformulated with any such variables replaced by the difference of two variables in the new formulation. For example, if any X(I) is unrestricted in sign, then replace X(I) by X1(I) - X2(I), that is, rewrite X(I) as the difference of two non-negative variables X1(I) and X2(I).

As an hypothetical example, consider the following problems:

$$-X_1 + X_2 \geq 4$$
$$X_2 \geq 2$$
$$X_1 \geq -2$$
$$X_1 \leq 3$$
$$\text{Max } Z = -3X_1 + 5X_2$$

The last constraint does not satisfy the fourth restriction. Thus, X_1 can be rewritten as the difference of two non-negative variables and the problem becomes

$$-(X_{11} - X_{12}) + X_2 \geq 4$$
$$X_2 \geq 2$$
$$(X_{11} - X_{12}) \geq -2$$
$$(X_{11} - X_{12}) \leq 3$$

where X_{11} and X_{12} are both greater than, or equal to, zero and

$$\text{Max } Z = -3(X_{11} - X_{12}) + .5\,X_2$$

OPERATING INSTRUCTIONS

LINE1$ uses the FORTRAN file capability to store the input data. To run the program, name the file, create and SAVE the input data, where the Data File is named LIPR, and enter the following underlined information.

```
OLD LINE1$**
READY.
100 +LIPR
RUN
```

After the program is compiled, the data from file LIPR is read in. Depending upon the value considered for LEVEL (see definitions), the value of the objective function and different basic variables, or the final basis will be printed out at each step and, finally, the value of the objective function (optimum) will be printed out. At the completion of the output, the program will loop back to read data for the next problem, if there is one.

You can run consecutive cases of input data, that is, more than one problem, by defining the input data, consecutively, on the input file.

Regardless of how many input cases you define on the input file, the last line of the file should contain the word END. If not, the program will execute properly but the message ØUT ØF DATA will be printed.

Terminology and Input Parameters

The following definitions apply to the parameters used by the program (see Sample Problem 1).

KL - Number of "less than" constraints.

KE - Number of "equal to" constraints.

KG - Number of "greater than" constraints.

N - Number of variables.

LEVEL - Indicator to obtain alternate outputs.

LG - Indicator for defining the type of problem (maximization or minimization).

A(I, J) - represent constraint-coefficients, right-hand side constants, and cost coefficients.

Output Data

Depending upon your choice of print options (value of the parameter LEVEL), the program prints a basis for each iteration and then prints the final solution. Finally, it prints the optimum value of the objective function or prints NØ FEASIBLE SØLUTIØN, or SØLUTIØN UNBØUNDED if there is no feasible solution to the problem, or if the objective function is unbounded. Then the program starts solving the next problem, if there is one.

PROGRAM REDIMENSION CAPABILITY

LINE1$ reserves sufficient storage for a data matrix of 1,860 elements This data matrix is dimensioned for 59 variables (including surplus) and 28 constraints (that is, 61 columns and 30 rows). If required the data matrix can be redimensioned as long as the inequality

$$(M + 2) \times (NN + 2) \leq 1,860$$

is satisfied, where NN is the total number of variables including surplus (if there is any), and M is the total number of constraints.

Using these definitions, the following program statement should be inserted

```
110 DIMENSION A(M+2, NN+2), H(NN+1), ID(23)
```

For example, if the problem had 37 constraints (30 less than, 4 equal to, and 3 greater than) and 42 variables, the above inequality would be satisfied and the program statement would be

```
110 DIMENSION A(39, 47), H(46), ID(23)
```

SAMPLE PROBLEM 1

Maximize the function $Z = X1 - X2 - X3 + X4$ subject to the constraints

$$\begin{aligned} X1 - 3X2 + 4X3 - 4X4 &\leq 5 \\ X1 - 2X2 &\leq 3 \\ -X1 - 3X2 + 4X3 - 4X4 &\geq 5 \\ 2X2 - X3 + X4 &\geq 4 \end{aligned}$$

Data File

The following is the Data File for Sample Problem 1. When entering identical items of data (such as 0, 0, 0, 0) you can type 4*0, using the multiplication sign to enter the information faster.

```
NEW LIPR
READY.

100 SAMPLE PRØBLEM
110 1,1,4,2,0,2
120 1,-3,4,-4
130 1,-2,2*0
140 -1,-3,4,-4
150 0,2,-1,1
160 5,3,5,4
170 1,-1,-1,1
180 END
SAVE
READY.
```

Explanation of Input File

Line 100 Identification line which may consist of name of the company, location, project number, and date of run.

Line 110 VALUE of the parameter LEVEL, value of the parameter LG, total number of variables, number of less than, or equal to, constraints, number of equations and number of greater than, or equal to, constraints.

The following option exists to get the output in alternate forms.

LEVEL = 1 Program will print basis at each iteration and final solution.
= 2 Program will print just the solution.

Depending on the type of problem, the value of the input parameter LG is given as follows:

LG = 1 for maximization problem
= 2 for minimization problem.

Line 120 to 150 Values of constraint-coefficients (row-wise).

Line 160 Values of right-hand side constants in correct format.

Line 170 Values of cost coefficients in correct format.

Line 180 END indicates no more data.

SOLUTION: SAMPLE PROBLEM 1

```
OLD LINE1$***

READY.

100 *LIPR
RUN

LINE1$    13:02    07 MON 06/15/70

     SAMPLE PROBLEM

 YOUR      VARIABLES ARE NUMBERED FROM  1 TO  4
 SURPLUS   VARIABLES ARE NUMBERED FROM  5 TO  6
 SLACK     VARIABLES ARE NUMBERED FROM  7 TO  8
 ARTIFICIAL VARIABLES ARE NUMBERED FROM  9 TO 10

 BASIS BEFORE ITERATION      1

 VARIABLE    VALUE
    7      5.000000
    8      3.000000
    9      5.000000
   10      4.000000

 OBJECTIVE FUNCTION VALUE:    .00000000

 BASIS BEFORE ITERATION      2

 VARIABLE    VALUE
    3      1.250000
    7      .0000000
    8      3.000000
   10      5.250000

 OBJECTIVE FUNCTION VALUE:   3.7500000

 BASIS BEFORE ITERATION      3

 VARIABLE    VALUE
    2      4.200000
    3      4.400000
    7      .0000000
    8      11.40000

 OBJECTIVE FUNCTION VALUE: -8.6000000

     ANSWERS:

 VARIABLE    VALUE
    1      .0000000
    2      4.200000
    3      4.400000
    8      11.40000

 OBJECTIVE FUNCTION VALUE: -8.6000000
```

SAMPLE PROBLEM 2

Western Feed, a large-scale distributor of feed for farm animals, purchases feed ingredients and mixes them for the farm community. Accordingly, the distributor receives an order from a customer raising beef cattle, of 2,000 pounds of a feed mix with the following specification:

Table 1. REQUIRED POUNDS OF NUTRIENTS PER TON OF FEED MIX

		Min	Max.
1.	Crude Protein	207.00	214.00
2.	Digestible Protein	149.00	155.00
3.	Fat	38.00	53.00
4.	Fiber	200.00	530.00
5.	Calcium	14.00	20.00
6.	Phosphorous	4.80	15.00
7.	Energy	2,476,000 KCal	2,706,000 KCal
8.	TDN	1,200.00	1,320.00

Special Limitations

Not more than 1% Animal Fat = 0.01*2,000 = 20 lbs per ton

Not less than 7% Molasses = 0.07*2,000 = 140 lbs per ton

Not less than 1.5% Premix for Western Beef = 0.015*2,000 = 30 lbs per ton

Not less than 1% Sale = 0.01*2,000 = 20 lbs per ton

On the basis of the nutrient blend required by the customer, Western Feed must select the appropriate ingredients from the nutrient breakdown presented in Table 2.

Before purchase is made of these necessary ingredients, it is required by the distributor, that he optimize the feed mix from a minimum cost standpoint, he must also satisfy the customer's nutrient requirements. The following information must therefore be obtained

1. Specification of feed ingredients (from Table 2) which will satisfy the customer's order.
2. Specification of the number of pounds of each feed ingredient, thus selected, per ton of feed mix.
3. Specification of the minimum cost per ton of feed mix.

Table 2 NUTRIENT BREAKDOWN*

Feed Integrident		Crude Protein	Digestible Protein	Fat	Fiber	Calcium	Phosphorous	Energy KCal	TDN	Cost per Pound
1. Milo	(x_1)	.110	.086	.030	.023	.0003	.0028	1,420	.840	$. [illegible]
2. Barley	(x_2)	.087	.069	.019	.057	.0006	.0033	1,580	.790	$. 027[illegible]
3. C. S. Meal	(x_3)	.456	.374	.057	.103	.0023	.0112	1,320	.751	$. 0375
4. C. S. Hulls	(x_4)	.039	.002	.009	.450	.0013	.0006	880	.437	$. 0100
5. Alf. Hay	(x_5)	.15	.106	.019	.286	.0147	0024	990	.507	$. 0180
6. Molasses	(x_6)	.067	.035	0	0	.0011	.0002	1,220	.606	$. 0175
7. Fat-Animal	(x_7)	0	0	.980	0	0	0	3,500	2 25	$. 0560
8. Screening	(x_8)	.092	.065	.015	.156	0	0	1,200	.670	$. 0160
9. Premix 80	(x_9)	.800	.530	.005	.060	.050	.027	400	.860	$. 1365
10. Premix Western	(x_{10})	.750	.570	.012	.020	.110	040	400	.820	$. 1365
11. Salt	(x_{11})	0	0	0	0	0	0	0	0	$. 0091

*All values represent the number of pounds of each nutrient in a pound of the given feed ingredient, except for energy which is represented in KCal/lb.

Mix Model-Minimize Cost

$$\text{Minimize} \quad c_1x_1 + c_2x_2 + c_3x_3 + \ldots c_nx_n$$

$$\text{S.T.} \quad a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots a_{2n}x_n \leq b_2$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 \ldots a_{mn}x_n \leq b_m$$

where x_1 = number of pounds of feed ingredient 1 per ton of feed mix.

c_1 = cost/lb of feed ingredient 1.

a_{1j} = pounds of nutrient 1 per pound of feed ingredient j.

b_1 = pounds (or KCal) of nutrient 1 required per ton of feed mix.

Mathematical Formulation

The constraints or restrictions to the problems are therefore formulated as follows

$$.11x_1 + .087x_2 + .456x_3 + .039x_4 + .15x_5 + .067x_6 + .092x_8 + .8x_9 + 75x_{10} \leq 214$$

$$\geq 207$$

$$.086x_1 + .069x_2 + .374x_3 + .002x_4 + .106x_5 + .035x_6 + .065x_8 + .53x_9 + .57x_{10} \leq 155$$

$$\geq 149$$

$$03x_1 + .019x_2 + .057x_3 + .009x_4 + .019x_5 + .98x_7 + .015x_8 + .005x_9 + .012x_{10} \leq 53$$

$$\geq 38$$

$$.023x_1 + .057x_2 + .103x_3 + .45x_4 + .286x_5 + .156x_8 + .06x_9 + .02x_{10} \leq 530$$

$$\geq 200$$

$$0003x_1 + .0006x_2 + .0023x_3 + .0013x_4 + .0147x_5 + .0011x_6 + .05x_9 + .11x_{10} \leq 20$$

$$\geq 14$$

$$.0028x_1 + .0033x_2 + .0112x_3 + .0006x_4 + .0024x_5 + .0002x_6 + .027x_9 + .04x_{10} \leq 15$$

$$\geq 4.8$$

$$1420x_1 + 1580x_2 + 1320x_3 + 880x_4 + 990x_5 + 1220x_6 + 3500x_7 + 1200x_8 + 400x_9 + 400x_{10} \leq 2766000$$

$$\geq 2476000$$

$$.84x_1 + .79x_2 + .751x_3 + .437x_4 + .507x_5 + .606x_6 + 2.25x_7 + .67x_8 + .86x_9 + .82x_{10} \leq 1320$$
$$\geq 1200$$

$x_7 \leq 20$ (not more than 1% animal fat)

$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} = 2000$ (The number of pounds of each ingredient in the feed mix totals 1 ton.)

$x_6 \geq 140$ (Not less than 7% molasses.)

$x_{10} \geq 30$ (Not less tnan 1.5% Premix for Western beef.)

$x_{11} \geq 20$ (Not less [illegible] salt.)

The objective function is:

Minimize

$$.0225x_1 + .0270x_2 + .0375x_3 + .0100x_4 + .0180x_5$$
$$+.0175x_6 + .0560x_7 + .0160x_8 + .1365x_9 + .1365x_{10}$$
$$+.0091x_{11}$$

Data File

The following is the Data File for Sample Problem 2.

```
100 FEED MIX MODEL -MINIMIZE COST
102,2,2,11,9,1,11
104,.11,.087,.456,.039,.15,.067,0,.092,.8,.75,0
106,.086,.069,.374,.002,.106,.035,0,.065,.53,.57,0
108,.03,.019,.057,.009,.019,0,.98,.015,.005,.012,0
110,.023,.057,.103,.45,.286,0,0,.156,.06,.02,0
112,.0003,.0006,.0023,.0013,.0147,.0011,2*0,.05,.11,0
114,.0028,.0033,.0112,.0006,.0024,.0002,0,0,.027,.04,0
116,1420,1580,1320,880,990,1220,3500,1200,2*400,0
118,.84,.79,.751,.437,.507,.606,2.25,.67,.86,.82,0
120,6*0,1,4*0
122,11*1
124,.11,.087,.456,.039,.153,.067,0,.092,.8,.75,0
126,.086,.069,.374,.002,.106,.035,0,.065,.53,.57,0
128,.03,.019,.057,.009,.019,0,.98,.015,.005,.012,0
130,.023,.057,.103,.45,.286,2*0,.156,.06,.02,0
132,.0003,.0006,.0023,.0013,.0147,.0011,2*0,.05,.11,0
134,.0028,.0033,.0112,.0006,.0024,.0002,0,0,.027,.04,0
136,1420,1580,1320,880,990,1220,3500,1200,2*400,0
138,.84,.79,.751,.437,.507,.606,2.25,.67,.86,.82,0
140,5*0,1,5*0
142,9*0,1,0
144,10*0,1
146,214,155,53,530,20,15,2766000,1320,20,2000,207,149,38,200
148,14,4.8,2476000,1200,140,30,20
150,.0225,.0270,.0375,.0100,.0180,.0175,.056,.016,.1365,.1365,.0091
152,END
```

SOLUTION SAMPLE PROBLEM 2

```
OLD LINEIS***

READY.

.OO *LIPR3
RUN



LINEIS     13:08    07 MØN 06/15/70

     FEED MIX MØDEL -MINIMIZE CØST

YØUR          VARIABLES ARE NUMBERED FRØM  1 TØ 11
SURPLUS       VARIABLES ARE NUMBERED FRØM 12 TØ 22
SLACK         VARIABLES ARE NUMBERED FRØM 23 TØ 31
ARTIFICIAL VARIABLES ARE NUMBERED FRØM 32 TØ 43

     ANSWERS:

VARIABLE   VALUE
    2      828.3574
    4      312.3590
    5      645.4082
    6      140.0000
    7      20.00000
   10      31.41289
   11      22.46241
   12      8.936225
   14      12.78973
   15      172.9929
   17      .9544902
   19      73.72381
   21      1.412889
   22      2.462411
   24      6.000000
   25      2.210266
   26      157.0070
   27      6.000000
   28      9.245510
   29      290000.0
   30      46.27619

ØBJECTIVE FUNCTIØN VALUE:  45.168857
```

Explanation of Output Data

The computer output shows that in order to minimize the total cost and to fulfill the nutrient requirements, 828.357 pounds of Barley, 312.359 pounds of C.S. Hulls, 645.408 pounds of Alf. Hay, 140.0 pounds of Molasses, 20.0 pounds of Fat-Animal, 31.4129 pounds of Premix Western, and 22.462 pounds of Salt per ton of feed ingredient, respectively, should be bought. No feed ingredients other than the specified should be bought. The total minimum cost will be $45.1689 dollars per ton of feed ingredients (that is, maximum cost = $45.1689 dollars).

LINPR$ is the Time-Sharing BASIC version of LINE1$. This program can handle a total of 17 constraints and 29 variables, including surplus, and does not use the Data File capability

LIMITATIONS

LINPR$ contains the following limitations. (See the Section 1, "Program Redimension Capability" for problems with more variables and fewer constraints (or vice versa) than are specified.)

1. The total of less than, equal to, and greater than, constraints should be equal to, or less than, 17, that is,

$$L + E + G \leq 17$$

2. The total of greater than constraints and the number of variables should be equal to, or less than, 29, that is,

$$G + N \leq 29$$

3. All right-hand side coefficients must be greater than, or equal to, zero.

4. All variables must be greater than, or equal to, zero.

OPERATING INSTRUCTIONS

LINPR$ can be run by typing the following lines

```
OLD LINPR$***
READY
10000 DATA ------
10001 DATA ------
```

(NOTE. Continue typing in data lines until complete)

```
RUN
```

Enter the data starting at line 10000 in the following constraint order

1. The coefficients of each of the problem variables (row-wise, including zeros for variables not appearing) in each constraint, starting with the first constraint and proceeding in order until all coefficients of all have been entered in data statements.

2. The elements of the right-hand side vector (the constants comprising the right side of all constraints) in the same order as the constraint.

3. The cost coefficients of the objective function, in the same order as used in the constraints, including zeros if needed.

You can run the consecutive cases of input data, that is, more than one problem, by defining the input data (in the same order as the previous problem), consecutively, on the data line.

The program requires that all the less than, or equal to, constraints be defined first, followed by all the equal to constraints and, finally by all the greater than, or equal to, constraints.

LINPR$ starts compiling after the command RUN is typed and the following message is given on the terminal

① HOW MANY PROBLEMS DO YOU WANT TO SOLVE? 4

The total number of problems you want to so... given ... It will be followed by:

② MAXIMIZATION = 1, M... ...ION = 2, WHICH? 2

Specify maximization or minimization by entering either "1" or "2". In this example, minimization is specified.

③ TYPE '1' FOR SOLUTION AND BASIS AT EACH ITERATION, OR '0' FOR JUST THE SOLUTION. WHICH? 1

Specify the output option by entering either "1" or "0". See "Sample Solution: LINPR$" for examples of the output options. In this example, the complete output form is specified.

④ WHAT ARE M AND N OF DATA MATRIX? 7,2

Enter the total number of constraints (m) and total number of variables (N) in that order separated by a comma. In this example, there are seven constraints and two variables.

⑤ HOW MANY 'LESS THANS', 'EQUALS', 'GREATER THANS'? 3,0,4

Enter the total number of "less than or equal to" constraints, equations, and "greater than or equal to" constraints in that order separated by commas

The program will loop through questions ②, ③, ④, and ⑤ for the number of times specified in answer to Question ①. Solutions will then be computed and printed.

Input data for answering questions can be entered by paper tape if the following precautions are observed:

- Enter each response on a separate line.
- Terminate each line with a X-OFF, carriage return, and rubout, in that order.

PROGRAM REDIMENSION CAPABILITY

LINPR$ reserves sufficient storage for a data matrix of 540 elements. This data matrix is dimensioned for 29 variables (including surplus) and 17 constraints (that is, 31 columns and 19 rows). If required, the data matrix can be redimensioned as long as the inequality

$$(M+1) \times (NN+1) \leq 540$$

is satisfied, where NN is the total number of variables including surplus (if there is any) and M is the total number of constraints.

Using these definitions, the following program statement should be selected

```
5 DIM A(M+1, NN+1), F(NN+1)
```

For example, if the problem had 25 constraints (19 less than, 3 equal to, and 3 greater than) and 16 variables, the above inequality would be satisfied and the program statement would be

```
5 DIM A(26,20), F(20)
```

SAMPLE PROBLEM: LINPR$

Maximize the function 30X1 + 45X2 while satisfying the following constraints.

```
   X1              ≤  6000
         X2        ≤  4000
2.5X1 +  2X2       ≤  24000
   X1              ≥  1000
         X2        ≥  1000
  3X1 -   X2       ≥  0
2.5X1 +  2X2       ≥  10000
```

The problem solution is

X1 = 6000 X2 = 4000

This problem has been run two times in order to illustrate all output options (by defining the data, consecutively, on the data line).

SAMPLE SOLUTION. LINPR$

```
ØLD LINPR$***
READY.
10000 DATA 1,0,0,1,2.5,2,1,0,0,1,3,-1,2.5,2
10001 DATA 6000,4000,24000,1000,1000,0,10000
10002 DATA 30,45
10003 DATA 1,0,0,1,2.5,2,1,0,0,1,3,-1,2.5,2
10004 DATA 6000,4000,24000,1000,1000,0,10000
10005 DATA 30,45
RUN
```

OPTION 0

```
LINPRS     13:39   07 MØN 06/15/70

HØW MANY PRØBLEMS DØ YØU WANT TØ SØLVE? 2

                PRØBLEM # 1

MAXIMIZATIØN=1,  MINIMIZATIØN=2.  WHICH? 1

TYPE '1' FØR SØLUTIØN AND BASIS AT EACH
ITERATIØN, ØR '0' FØR JUST THE SØLUTIØN.  WHICH? 0

WHAT ARE M AND N ØF THE DATA MATRIX? 7,2

HØW MANY 'LESS THANS','EQUALS','GREATER THANS'? 3,0,4

YØUR   VARIABLES      1    THRØUGH 2
SURPLUS  VARIABLES    3    THRØUGH 6
SLACK  VARIABLES      7    THRØUGH 9
ARTIFICIAL VARIABLES 10    THRØUGH 13

ANSWERS:

 VARIABLE        VALUE
    5            14000.
    6            13000.
    9             1000.
    4             3000.
    2             4000.
    1             6000.
    3             5000.
ØBJECTIVE FUNCTIØN VALUE      360000.
```

OPTION 1

```
                    PRØBLEM # 2

 MAXIMIZATIØN=1, MINIMIZATIØN=2. WHICH? 1

 TYPE '1' FØR SØLUTIØN AND BASIS AT EACH
 ITERATIØN, ØR '0' FØR JUST A SØLUTIØN. WHICH? 1

 WHAT ARE M AND N ØF THE DATA MATRIX? 7,2

 HØW MANY 'LESS THANS','EQUALS','GREATER THANS'? 3,0,4

 YØUR VARIABLES      1   THRØUGH  2
 SURPLUS VARIABLES   3   THRØUGH  6
 SLACK VARIABLES     7   THRØUGH  9
 ARTIFICIAL VARIABLES 10 THRØUGH 13

 BASIS BEFØRE ITERATIØN 1
 VARIABLE      VALUE
  7             6000
  8             4000
  9             24000
  10            1000
  11            1000
  12            0
  13            10000

 ØBJECTIVE FUNCTIØN VALUE      0

 BASIS BEFØRE ITERATIØN 2
 VARIABLE      VALUE
  7             6000
  8             4000
  9             24000
  10            1000
  11            1000
  1             0
  13            10000

 ØBJECTIVE FUNCTIØN VALUE      0

 BASIS BEFØRE ITERATIØN 3
 VARIABLE      VALUE
  7             5666.67
  8             3000
  9             21166.7
  10            666.667
  2             1000
  1             333.333
  13            7166.67

 ØBJECTIVE FUNCTIØN VALUE      4166.67
```

OPTION 1 - Continued

```
 BASIS BEFØRE ITERATIØN 4
 VARIABLE        VALUE
    7             5000.
    8             1000.
    9             15500.
    4             2000.
    2             3000.
    1             1000.
   13             1500.

 ØBJECTIVE FUNCTIØN VALUE          10500.

 BASIS BEFØRE ITERATIØN 5
 VARIABLE        VALUE
    7             4823.53
    8             470.588
    9             14000.
    4             2529.41
    2             3529.41
    1             1176.47
    3             176.471

 ØBJECTIVE FUNCTIØN VALUE          194118.

 BASIS BEFØRE ITERATIØN 6
 VARIABLE        VALUE
    7             4666.67
    6             1333.33
    9             12666.7
    4             3000.
    2             4000.
    1             1333.33
    3             333.333

 ØBJECTIVE FUNCTIØN VALUE          220000

 ANSWERS:

 VARIABLE        VALUE
    5             14000.
    6             13000.
    9             1000.
    4             3000.
    2             4000.
    1             6000.
    3             5000.

 ØBJECTIVE FUNCTIØN VALUE          360000.
```

Linear programming (LP) methods are used extensively in the solution of many problems, namely, allocation, transportation, assignment, and manufacturing problems. Although optimum schedules and results are easily attainable, at times a need exists for further analysis of the results. Such post analysis is referred to as sensitivity analysis.* For example, as a decision maker who uses the results of an LP program as a guide to final decision making, you may want to know the effects of tightening or of relaxing a certain product specification (that is, changing one of the cost-coefficients or right-hand side constants), or the effects of price changes for certain raw materials. You may also want to know the return on capital in several possible places before deciding to add new capital equipment. Sensitivity analysis can help in resolving some of these questions. In addition, you will have increased confidence in your application of the optimum schedules generated by the LP method.

SENSI$ is written in Time-Sharing FORTRAN and not only solves linear programming problems, but also performs a subsequent sensitivity analysis. Note that the data for this program is arranged differently from that of LINE1$ and of LINPR$.

Following are the parameters on which the sensitivity analysis can be performed

- Range of the elements of the requirement-vector (one at a time).
- Range of the cost coefficients (one at a time).

  For which the optimal solution remains unaltered although the level of the basic activities can be changed.

- Complete sensitivity analysis which shows the effect on other parameters of changing the value of a particular parameter.
- Sensitivity of profit to a change in a structural coefficient of a constraint.
- Effect on the solution by the addition of a new variable.

LIMITATIONS

SENSI$ contains the following limits.

1. The total of greater than, or equal to, equal to, and less than, or equal to constraints should be equal to, or less than, 20, that is

   $GT + E + LT \leq 20$

2. The total of equal to, less than, or equal to, and twice the number of greater than, or equal to, constraints and number of variables should be less than, or equal to, 40, that is

   $2GT + E + L + N \leq 40$

3. All right-hand side coefficients must be greater than, or equal to, zero.

4. All variables must be greater than, or equal to, zero.

*For further information concerning sensitivity analysis, see the Linear Programming Applications Manuals listed in the Introduction.

Thus SENSI$ can handle a total of 20 constraints and 40 variables (including slack, surplus, and artificial)

OPERATING INSTRUCTIONS

SENSI$ uses the FORTRAN file capability to store the input data for original problems. To name the file, create the input data, and SAVE the file, where SEN . - the name of the Data File, enter the following underlined information

```
ØLD SENSI$***
READY.
100 + SENS1
RUN
```

After the program is com p. the data from file SENS1 is read in, the following message is prin and _ program is ready to accept input

```
DØ YØU WANT BASE PRINTED AT EACH STEP, TYPE YES ØR NØ?  YES
```

If you are particularly interested in seeing the basis printed at each step, type "YES." The value of the objective function and different basic variables will be printed out at each step and finally the optimum value of the objective function will be printed out. Artificial, slack, surplus and original variables of the problem will be assigned respective numbers for the purpose of identifying them in the final solution and later in the program

Now the program will start performing the sensitivity analysis on any of the parameters of the problem by requesting

```
DØ YØU WANT REQUIREMENT-VECTØR RANGE, TYPE YES ØR NØ?  YES
```

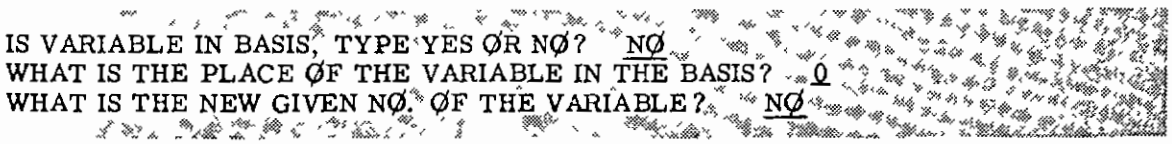
If your response to the above question is "YES" (i.e., you want to know the range of one of the elements of the right-hand side vector for which the optimum basis remains same), then the program will ask for the next input

```
WHAT IS THE CØNSTRAINT NØ.?
```

to which the constraint number is given in which the element of the right-hand side vector appears. The program will continue requesting the constraint numbers until 0 is typed, (i e., when you are no longer interested in getting the right-hand side vector range). Now the program will start performing an analysis on cost coefficients by asking for the input to the following question in turn.

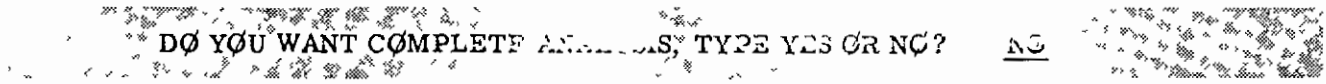
```
DØ YØU WANT CØST CØEFF. RANGE, TYPE YES ØR NØ?  NØ
```

If your response is "NO," the program will start performing the next set of analyses. If the response is "YES," the following messages will be typed



If the corresponding variable is in the final basis, then your response should be "YES" and the program will then ask for the place of that variable in the basis by printing message a. If the variable is not in the basis or your response is "NO," the program will then ask for the new given number of the variable by printing message b. The program will loop back and start asking the same question for the next variable. When you are no longer interested in getting the coefficient range, type "NO" in response to the first question and "0" in response to message b.

If you are interested in getting the complete analysis (i.e., the effect of changing the value of one parameter on other parameters of the problem), type "YES" in response to the following

⑧ 

If your response is "NO," the following message will be printed

⑨ 

This will perform a sensitivity analysis on one of the structural coefficients (one at a time) of the problem. If your response is "YES," the following message will be printed

In response to this message, give the pertinent constraint number. The program will loop back to ask the same question until you type "0," which means you are no longer interested in this analysis. The program will then print the following messages.

If your response to message a is "NO," the program will loop back to solve the next problem if any. If your response is "YES," message b will be printed and the program will ask for the structural coefficients and cost coefficient of the new variable. If the new variable does not affect the optimum solution, the program will print "SAME ANSWER AS BEFORE." Otherwise, it will print a new optimal basis and corresponding solution

You can run the consecutive cases of input data, i.e., more than one problem, by defining the input data consecutively on the input file. Regardless of how many input cases you define on the input file the last line of the file should contain the word "END."

SAMPLE PROBLEM 1

A manufacturer produces two kinds of rugs. One is of high quality and can be sold at $18 profit; the other a cheap rug yields only $3 profit. Labor used in production is somewhat skilled and is limited

to 800 man-hours per week. On the other hand, due to a union contract, if less than 800 man-hours are used, unemployment compensation of $1 must be paid for each hour wasted. A quality rug takes one man-hour to produce, a cheap rug, two man-hours. The difference in quality lies in the amount used of some special material which improves fibers and makes them more workable. The expensive rug needs three pounds of this material and the cheap rug only one pound. The availability of this material recently dropped to only 1500 pounds per week. Nevertheless, the manufacturer is obligated to produce at least 600 rugs a week of any kind. The problem is to find a production schedule which meets all requirements and results in maximum profit.

Problem Formulation

Let

X1 = the number of quality rugs to be made per week

X2 = the number of inexpensive rugs to be made per week

X3 = the slack variable representing unused labor

The constraints become

(1)	Production	$X1 + X2 \geq 600$	
(2)	Labor	$X1 + 2X2 + X3 = 800$	
(3)	Special Material	$3X1 + X2 \leq 1500$	

and the objective function is

Profit/week = $18X1 + $3X2 - $1X3

which has to be maximized.

Data File

Following is a Data File for Sample Problem 1.

```
NEW SENS1

100 SAMPLE PROBLEM
110 1,3,1,1,1
120 1,1,0
130 1,2,1
140 3,1,0
150 600,800,1500
160 18,3,-1
170 END

SAVE

READY.
```

Explanation of Data File

Line 100 Identification line may consist of the name of the company, its location, the project number, and the date of run The total number of characters must not exceed 51.

Line 110 Value of the parameter LG, number of variables, number of greater than, or equal to, constraints, number of equal to, number of less than, or equal to, constraints. Depending on the type of problem, the value of the input parameter, LG, is given as follows.

LG = 1 for maximization problem
= 2 for minimization problem

Line 120 to 140 Values of left-hand side coefficients (row-wise). First, greater than, constraints, then, strict equalities followed by less than, constraints. Include zero for variables not appearing

Line 150 Value of right-hand side constants in correct format.

Line 160 Values of cost coefficients (includes zero for variables not appearing).

Line 170 END indicates no more data.

SOLUTION: SAMPLE PROBLEM 1

```
ØLD SENSI$***

READY.

100 + SENS1
RUN

SENSI$    11.34    W1    TUE  06/17/69

DØ YØU WANT BASE PRINTED AT EACH STEP, TYPE YES ØR NØ?  NØ

     SAMPLE PRØBLEM

ARTIFICIAL VARIABLES ARE NUMBERED  FRØM  TØ    1    2
SLACK VARIABLES ARE NUMBERED       FRØM  TØ    3    3
SURPLUS VARIABLES ARE NUMBERED     FRØM  TØ    7    7

YØUR VARIABLES ARE NUMBERED        FRØM  TØ    4    6

  BASE    VALUE
    4     450.00
    5     150.00
    6      50.00
ØPTIMUM VALUE =    8.50000E+03

DØ YØU WANT REQUIREMENT-VECTØR RANGE, TYPE YES ØR NØ?  YES

WHAT IS THE CØNSTRAINT NØ.?  1
   500.00      600.00      620.00
```

```
WHAT IS THE CONSTRAINT NØ.?  2
     750.00          800.00          1.00000E+70

WHAT IS THE CONSTRAINT NØ.?  3
     1400.00         1500.00         1800.00

WHAT IS THE CØNSTRAINT NØ.?  0

DØ YØU WANT CØST CØEFF. RANGE, TYPE YES ØR NØ?  YES

IS VARIABLE IN BASIS, TYPE YES ØR NØ?  YES
WHAT IS THE PLACE ØF THE VARIABLE IN BASIS?  1
          14.00          18.00          2.00000E+60

IS VARIABLE IN BASIS, TYPE YES ØR NØ?  YES
WHAT IS THE PLACE ØF THE VARIABLE IN BASIS?  2
-6.66667E+59          3.00          4.3333

IS VARIABLE IN BASIS, TYPE YES ØR NØ?  YES
WHAT IS THE PLACE ØF THE VARIABLE IN BASIS?  3
     -1.80          -1.00          4.00000E+59

IS VARIABLE IN BASIS, TYPE YES ØR NØ?  NØ
WHAT IS THE NEW GIVEN NØ. ØF THE VARIABLE?  3
-1.00000E+70          .00          7.00

IS VARIABLE IN BASIS, TYPE YES ØR NØ?  NØ
WHAT IS THE NEW GIVEN NØ. ØF THE VARIABLE?  7
-1.00000E+70          .00          2.00

IS VARIABLE IN BASIS, TYPE YES ØR NØ?  NØ
WHAT IS THE NEW GIVEN NØ. ØF THE VARIABLE?  0

DØ YØU WANT CØMPLETE ANALYSIS, TYPE YES ØR NØ?  YES
 .5000E+00      .0000E+00      .5000E+00
-.1500E+01      .0000E+00      .5000E+00
 .2500E+01     -.1000E+01     -.5000E+00
 .2000E+01      .1000E+01     -.7000E+01

DØ YØU WANT SENSITIVITY ØF PRØFIT TØ A CHANGE IN A COEFFICIENT ØF A
CØNSTRAINT, TYPE YES ØR NØ?  YES

WITH RESPECT TØ WHICH CØNSTRAINT(NØ.)?
     4     .90000E+03
     5     .30000E+03
     6     .10000E+03

WITH RESPECT TØ WHICH CØNSTRAINT(NØ.)?  2
     4     .45000E+03
     5     .15000E+03
     6     .50000E+02
```

SOLUTION SAMPLE PROBLEM 1 - Continued

```
WITH RESPECT TØ WHICH CØNSTRAINT(NØ )?  3
      4     -.31500E+04
      5     -.10500E+04
      6     -.35000E+03

WITH RESPECT TØ WHICH CØNSTRAINT(NØ.)?  0

DØ YØU WANT TØ ADD NEW VARIABLE, TYPE YES ØR NØ?  NØ
```

EXPLANATION OF OUTPUT DATA

Original variables X1, X2, and X3 are renamed as 4, 5, and 6, respectively.

Right-hand Side Vector Range

The first constraint corresponds to the production constraint whole right-hand side value is 600 (that is, minimum rug production is 600). The output shows that this value can vary between 500 and 620 without changing the optimal basis Similarly, the number of man-hours whose original value is 800 can vary between 750 and + ∞ (infinity), the availability of material can vary between 1,400 and 1,800 pounds per week without changing the optimal basis (although the levels of different activities may change).

Cost Coefficient Range

In this analysis, the range of the cost coefficient is given for which the optimal basis remains unaltered. The cost coefficient of variable #4 (original X1), which is on the first place in the basis, can vary between $14.0 dollars and (+ ∞) (original value $18.00 dollars), cost coefficient of variable #5 (original X2) can vary between $(- ∞) and $4.333 dollars, and cost coefficient of basic variable #6 (original X3) can vary between $(-1.8) and (+ ∞) without changing the optimal basis. Similarly, the cost coefficient of the slack variable (#3, representing unused availability of material) which is not in the optimal basis can vary between $(- ∞) and $7.0 dollars without changing the final basis. In the above analysis, it has been assumed that only one parameter is changed while the others remain fixed to their original values.

After grouping the different headings, the complete analysis tableau can be rewritten as follows:

Table 3. ANALYSIS TABLEAU

		(1) Production	(2) Labor	(3) Special Material
(4)	Quality Rugs	0.5	0	-.5
(5)	Cheap Rugs	-1.5	0	.5
(6)	Unused Labor Availability	2.5	-1	-.5
(7)	Profit	2.0	1	-7.

The three columns, namely, (1), (2), and (3) represent three constraints (that is, production, labor, and special material). Rows (4)-(6) represent the different activities (or variables) in the same order as they appear in the optimal basis. The last row (7) is a Profit row

Interpretation of Results

It is clear from Table 3 that reduction in production of rugs by one unit will increase the profit by $2. The number of quality rugs will be increased by one-half, the production of inexpensive rugs will be decreased by three halves, and the unused man-hours will be increased by five halves. The second column shows the increase, or decrease, of man-hours (within its range) has no effect on the number of quality and inexpensive rugs produced. But decreasing one man-hour will decrease the unused man-hours to 49 and the profit will be increased by $1. Similarly, decreasing special material by one pound will decrease the profit by $7.

The preceding answer is given in terms of substitution rates, which are not limited to just one unit of change but which apply for a change within the range (right-side vector range) of validity of the substitution.

It is advantageous to know how the profit will change if one of the structural coefficients (the left-hand side) is slightly varied. Remember that this analysis is correct only if a small change in a coefficient of a constraint is considered. This analysis is not quite valid for larger changes.

First of the last three results (with respect to constraint 1 - production constraint) is reproduced as follows:

Table 4. RATE OF CHANGE OF PROFIT

Variables	Production
Quality Rugs	900
Inexpensive Rugs	300
Unused Man-Hours	100

Profit increases 900 times as fast as does the amount of production (of all rugs) per quantity of the quality rug increases. Profit increases 300 times as fast as does the amount of production per quantity of the inexpensive rug increases, and 100 times as fast as the amount of production per hour of unused man-hour increases.

It should be remembered that Table 4 gives the rate of change of profit (for coefficient values very close to their original values). For large increases, the rate varies, considerably.

For the last analysis (adding a new variable), see Sample Problem 2.

SAMPLE PROBLEM 2

Consider a hypothetical problem:

$$\begin{aligned} 16X1 &+ X2 + X3 \geq 4 \\ X1 &+ 5X2 - 10X3 \geq 10 \\ 3X1 &+ 2X2 + 5X3 \leq 20 \\ X1, X2, &X3, \geq 0 \end{aligned}$$

Minimize $Z = -X1 - 4X2 - 5X3$

After solving the problem, you may have omitted an important activity or variable (which we'll call X4) that has a sufficient bearing on the solution to the problem.

P[illegible] m Formulation

Let the new problem constrained be as follows.

16X1 + X2 + [illegible] - 4
X1 + 5X2 [illegible] .0
3X1 + 2X[illegible] [illegible] 30
X1, X[illegible]

Minimum Z = [illegible] - [illegible] - 2X4

After getting [illegible] to the original pro[illegible] (with variable X4 omitted), the variable X4 is added and a new [illegible]

[illegible] e

[illegible] File for Sample Problem 2 is shown as follows.

```
NEW SENS2
SAVE
READY.
100 SAMPLE PROBLEM NO. 2
110 2,3,2,0,1
120 16,1,1
130 1,5,-10
140 3,2,5
150 4,10,20
160 -1,-4,-5
170 END
READY.
```

SOLUTION: SAMPLE PROBLEM 2

```
OLD SENSI$***
READY.
100 + SENS2
RUN
SENSI$    11:31    W1    TUE  06/17/69

DO YOU WANT BASE PRINTED AT EACH STEP, TYPE YES OR NO?  NO

     SAMPLE PROBLEM NO. 2
ARTIFICIAL VARIABLES ARE NUMBERED  FROM TO   1
SLACK VARIABLES ARE NUMBERED       FROM TO   3
SURPLUS VARIABLES ARE NUMBERED     FROM TO   7   8

YOUR VARIABLES ARE NUMBERED        FROM TO   4   6
```

```
BASE   VALUE
  8    40.00
  5    10.00
  7     6.00

ØPTIMUM VALUE =   4.00000E+01

DØ YØU WANT REQUIREMENT-VECTØR RANGE, TYPE YES ØR NØ? NØ

DØ YØU WANT CØST CØEFF. RANGE, TYPE YES ØR NØ?  NØ

DØ YØU WANT CØMPLETE ANALYSIS, TYPE YES ØR NØ?  NØ

DØ YØU WANT SENSITIVITY ØF PRØFIT TØ A CHANGE IN A CØEFFICIENT ØF A
CØNSTRAINT, TYPE YES ØR NØ?    NØ

DØ YØU WANT TØ ADD NEW VARIABLE, TYPE YES ØR NØ?    YES

GIVE STRUCTURAL AND CØST CØEFF. ØF THE VARIABLE?    2, -3, 1, -2
THIS VARIABLE IS GIVEN NØ    9

      BASE      VALUE
        9        7.2727
        5        6 3636
        7       16.9091
ØPTIMUM VALUE =     4.00000E+01
```

ARTIFICIAL VARIABLE: A variable which is added externally to the problem in order to obtain an initial basis is called an artificial variable and, once it has been driven out of the basis or has achieved a zero level, the simplex procedure would not let it enter the basis again (that is, it will not become non-zero).

BASIC FEASIBLE SOLUTION: A solution [illegible] equations which has non-zero variable values [illegible] a basic solution and, if it satisfies the non-negativity restrictions is called a basic feasible solution.

NONFEASIBLE SOLUTION: Any solution which does not satisfy the non-negativity restrictions is called a nonfeasible solution.

OBJECTIVE FUNCTION: The function to be optimized [illegible]ed the objective function.

SLACK VARIABLE: An auxiliary variable [illegible] ert the inequality constraint (≤) to an equation.

SOLUTION UNBOUNDED: The problem is said to have an unbounded solution when a problem has no finite optimum value of its objective function.

SURPLUS VARIABLE: An auxiliary variable introduced to convert the inequality constraint (≥) to an equation.

202026B

BASIC Language

June 1965
Revised 3-69

INFORMATION SYSTEMS

GENERAL ELECTRIC

Preface

This manual, which combines and supersedes two other manuals--BASIC Language Reference Manual (202026A) and BASIC Language Extensions Reference Manual (802207--describes the version of the BASIC language used with the General Electric Mark I Time-Sharing Service.

Another manual, Mark I Time-Sharing Service Command System Reference Manual (229116) explains all of the system commands that are a part of the Mark I Time-Sharing Service. It should be consulted for system information.

The original development of the BASIC language was supported by the National Science Foundation under the terms of a grant to Dartmouth College. Under this grant, Dartmouth College developed, under the direction of Professors John G. Kemeny and Thomas E. Kurtz, the BASIC language compiler. Since that development, BASIC has been offered as part of the Time-Sharing Service of General Electric's Information Service Department.

General Electric has continued to expand the capabilities of BASIC, adding such versatile features as string manipulation, data files, formatted line output, and others.

Contents

Page

APPENDIXES

Introduction

A program is a set of directions that is used to tell a computer how to provide an answer to some problem. It usually starts with the given data, contains a set of instructions to be carried out in a certain order, and ends up with a set of answers

Any program must meet two requirements before it can be carried out. The first is that it must be presented in a form that the computer understands. If the program is a set of instructions for solving a system of linear equations and the computer is an English-speaking person, the program will be presented in a combination of mathematical notation and English. If the computer is a French-speaking person, the program will be presented in French rather than English. And if the computer is a high-speed digital computer, the program must be presented in a programming language that the digital computer understands.

The second requirement for any program is that it must be completely and precisely stated. This requirement is crucial when dealing with a digital computer, which has no ability to infer what you mean. It does what you tell it to do, not what you meant to tell it.

We are talking about programs that provide numerical answers to numerical problems. It would be easy to present a program in the English language, but such a program would pose insurmountable difficulties for the computer. English is rich with ambiguities and redundancies, the qualities that make poetry possible but computing impossible. Instead of using English, you present your program in a programming language that resembles ordinary matnematical notation, that has a simple vocabulary, and that permits a complete and precise specification of your program. The programming language you will use is BASIC, Beginner's All-purpose Symbolic Instruction Code. BASIC is precise, simple, and easy to understand.

An introduction to writing a BASIC program is given in Chapter 1, which includes all that you need to know to write a variety of useful and interesting programs. Chapter 2 deals with more advanced techniques. The Appendixes contain a variety of reference materials.

1. A BASIC Primer

AN EXAMPLE

The following example is a complete BASIC program for solving a system of two simultaneous linear equations in two variables:

$$ax + by = c \qquad\qquad dx + ey = f$$

and then solving two systems, each differing from this system only in the constants c and f.

If ae − bd is not equal to zero, you should be able to solve this system to find that

$$x = \frac{ce - bf}{ae - bd} \qquad \text{and} \qquad y = \frac{af - cd}{ae - bd}$$

If ae − bd is equal to zero, either there is no solution or there are infinitely many, but there is no unique solution. If you are rusty on solving such systems, take our word for it that this is correct. For now, we simply want you to understand the BASIC program for solving this system.

Study the following program carefully--the purpose of most lines in the program is self-evident--and then read the commentary and explanation.

```
100 READ A,B,D,E
110 LET G=A*E-B*D
120 IF G=0 THEN 180
130 READ C,F
140 LET X=(C*E-B*F)/G
150 LET Y=(A*F-C*D)/G
160 PRINT X,Y
170 GØ TØ 130
180 PRINT "NØ UNIQUE SØLUTIØN"
190 DATA 1,2,4
200 DATA 2,-7,5
210 DATA 1,3,4,-7
999 END
```

You can see, first, that this sample program uses only capital letters, because the teletype-writer has only capital letters.

Second, you can see that each line of the program begins with a line number. These numbers are called *line numbers;* they identify the lines, each of which is called a *statement.* A program is made up of statements, most of which are instructions to the computer. The line numbers also serve to specify the order in which the statements are to be performed by the computer. This means that you may type your program in any order. Before the program is run, the computer sorts out and edits the program, putting the statements into the order specified by their line numbers. This editing process also facilitates correcting and changing programs, as we shall explain later on.

Third, note that each statement starts, after its line number, with an English word. The word denotes the type of the statement. There are several types of statements in BASIC, nine

of which are discussed in this chapter. Seven these nine appear in the sample program we are now considering.

Note also that, although it is not obvious from the program, spaces have no significance in BASIC statements, except in messages that are to be printed out, as in line number 180 Spaces may be used or not, as will, to make program more readable. Statement 100 could have been typed as 100READA,B,C,D, E and statement 110 as 110LETG=A*E-B*D

With this preface, let's go through the program step by step. The first statement, 100, is a READ statement. It must be accompanied by one or more DATA statements. When the computer encounters a READ statement while executing your program, it will cause the variables listed after the word READ to be given values according to the next available numbers in the DATA statements. In our sample program we read A in statement 100 and assign the value 1 to it from statement 190, and similarly with B and 2, and with D and 4. At this point, we have exhausted the available data in statement 190, but there is more in statement 200 We pick up there the number 2 to be assigned E.

Next we go to statement 110, a LET statement, where we first encounter a formula to be evaluated. (The asterisk, *, is used to denote multiplication.) In this statement we direct the computer to find the value of AE – BD, and to call the result G. In general, a LET statement directs the computer to set a variable equal to the formula on the righthand side of the equal sign.

We know that if G is equal to zero the system has no unique solution. Therefore, we next ask, in line 120, whether G is equal to zero. If the computer finds a Yes answer to the question, it is directed to go to line 180. Line 180 tells it to print out NØ UNIQUE SØLUTIØN. From this point, it would go to the next statement. But lines 190, 200, and 210 give it no instructions, since DATA statements are not executed, and it then goes to line 999, which directs it to END the program.

If the answer to the question "Is G equal to zero?" is No, as it is in our sample program, the computer simply goes to the next statement, in this case statement 130. (An IF--THEN statement tells the computer where to go if the IF condition exists, but to go on to the next statement if it does not exist.) Statement 130 directs the computer to read the next two entries from the DATA statements--in this case -7 and 5, both in statement 200--and to assign them to C and F respectively. The computer is now ready to solve the system:

$$x + 2y = -7 \qquad 4x + 2y = 5$$

In statements 140 and 150, we direct the computer to find the values of x and y according to the formulas provided. Note that we must use parentheses to show that CE – BF is divided by G. Without parentheses, only BF would be divided by G, and the computer would find

$$X = CE - \frac{BF}{G}$$

The computer is told in line 160 to print the two values computed, those of X and Y. Having done so, it moves on to line 170, where it is directed back to line 130. If there are additional numbers in the DATA statements, as there are here in 210, the computer is told in line 130 to take the next number and assign it to C, and the one after that to F. The computer is now ready to solve the system:

$$x + 2y = 1 \qquad 4x + 2y = 3$$

As before, it finds the solution in 140 and 150, prints out the values in 160, and then is directed in 170 to go back to 130.

In line 130 the computer reads two more values, 4 and -7, which it finds in line 210 It then solves the system:

$$x + 2y = 4 \qquad 4x + 2y = -7$$

and prints out the solutions. It is directed back again to 130, but there are no more pairs of numbers available for C and F in the DATA statements. The computer therefore informs you that it is out of data, printing on the paper in the teletypewriter ØUT ØF DATA IN 130, and stops.

Let's look at the importance of the various statements. For example, what would have happened if we had omitted line number 160? The answer is simple. The computer would have solved the equations three times and then told us it was out of data. However, since it was not told to show us (PRINT) the answers, it would not do so, and the solutions would be the computer's secret.

What would have happened if we had left out line 120? In the problem just solved, nothing. But if G were equal to zero, we would have set the computer the impossible task of dividing by zero in 140 and 150, and it would tell us so by printing out DIVISIØN BY ZERØ IN 140 and DIVISIØN BY ZERØ IN 150. Suppose we had left out statement 170? The computer would have solved the first system, printed out the values of X and Y, and then gone on to line 180. As directed, it would print out NØ UNIQUE SØLUTIØN, and then stop.

A natural question that may arise is, why this selection of line numbers? The answer is that the particular choice of line numbers is arbitrary. The only requirement is that statements be numbered in the order that we want the computer to follow in executing the program. We could have numbered the statements 1, 2, 3, 4, . . . , 13; but we do not recommend this numbering. We would normally number the statements 100, 110, 120, . . ., 999. We put the numbers such a distance apart so that we can later insert additional statements easily. For example, if we find that we have left out two statements belonging between those numbered 140 and 150, we can give them any two numbers between 140 and 150, say 144 and 146. In the editing and sorting process, the computer will put them in the correct place.

Another question that may arise has to do with the placing of the elements of data in the DATA statements: Why place them as they are in the sample program? The choice is arbitrary. We need only arrange the numbers in the order that we want them read--the first for A, the second for B, the third for D, the fourth for E, the fifth for C, the sixth for F, the seventh for the next C, and so on. In place of the three statements numbered 190, 200, 210, we could have put

```
195 DATA 1,2,4,2,-7,5,1,3,4,-7
```

or we could have written, perhaps more logically

```
190 DATA 1,2,4,2
200 DATA -7,5
210 DATA 1,3
220 DATA 4,-7
```

putting the coefficients in the first DATA statement and the three pairs of righthand constants in the following DATA statements.

After typing the program, we type RUN followed by a carriage return. Up to this point the computer stores the program and does nothing else with it. It is the command RUN that directs the computer to execute the program.

The sample program and the resulting printout are shown now as they appear on the teletypewriter

```
100 READ A,B,D,E
110 LET G=A*E-B*D
120 IF G=0 THEN 180
130 READ C,F
140 LET X=(C*E-B*F)/G
150 LET Y=(A*F-C*D)/G
160 PRINT X,Y
170 GØ TØ 130
```

```
1?0 PRINT "   [illegible] THE SOLUTION"
1?0 [illegible]
[illegible]
2?0 DATA 1 [illegible]
2?? END
RU[illegible]

L[illegible]EAR [illegible]

 4                  [illegible]
 .666667            [illegible]6667
-3.66667            [illegible]333

OUT OF DAT[illegible]  [illegible] 1?0
```

FORMULAS

[illegible] computer can carry out a great many operations. It can add, subtract, multiply, divide, [illegible] square roots, raise a number to a power, find the sine of a number on an angle meas[illegible] in radians, and so on. We shall now learn how to tell the computer to carry out these [illegible] operations in the order that we want them done

The computer computes by evaluating formulas that are supplied in a program. The formulas are similar to those used in ordinary mathematical calculation, except that each BASIC formula must be written on a single line. Five arithmetic operations can be used to write a formula. They are listed in the following table.

Symbol	Example	Meaning
+	A + B	Addition. Add B to A.
-	A - B	Subtraction. Subtract B from A.
*	A * B	Multiplication. Multiply B by A.
/	A / B	Division Divide A by B.
↑	X ↑ 2	Raise to the power. Find X^2.

We must be careful with parentheses to make sure that we group together the things we want together. We must also understand the order in which the computer does its work.

For example, if we type A+B*C↑D, the computer will first raise C to the power D, then multiply this result by B, and then add A to the resulting product. This is the usual convention [illegible] $A + BC^D$. If this is not the intended order, we must use parentheses to indicate a different [illegible] Suppose it is the product of B and C that we want raised to the power D. We must write [illegible] Or, if we want to multiply A + B by C to the power D, we write (A+B)*C↑D. We [illegible] to B, multiply their sum by C, and raise the product to the power D by writing ((A+B)*C)↑D

[illegible] order of [illegible] computing is according to the following rules.

[illegible] parentheses is computed before the enclosed quantity is used in [illegible]

[illegible] formula that includes addition, multiplication, and [illegible] the computer first raises the number to the power, then does the [illegible] and does the addition last. Division has the same priority [illegible] the same as addition.

[illegible] formula that includes only multiplication and division (or only addition and subtraction), the computer works from left to right.

The rules are illust[illegible] sample program already consid[illegible]red The rules also tell us [illegible] -B-C, will subtract B from A and then C from their difference. [illegible] A/B/C, it will [illegible] by B and then that quotient by C Given A↑B↑C, the computer

will raise the number A to the power B and then raise the resulting number to the power C. If there is ever any question in your mind about the priority, put in more parentheses to avoid possible ambiguities.

In addition to the five arithmetic operations, the computer can evaluate several mathematical functions. The functions are given special three-letter names, as shown in the following table.

Function	Meaning	
SIN(X)	Find the sine of X.	X interpreted as a number, or as an angle measured in radians.
CØS(X)	Find the cosine of X.	
TAN(X)	Find the tangent of X.	
ATN(X)	Find the arctangent of X.	
EXP(X)	Find e^x.	
LØG(X)	Find the natural logarithm of X (ln X).	
ABS(X)	Find the absolute value of X ($\|X\|$).	
SQR(X)	Find the square root of X ($\sqrt{X}$).	

Three other mathematical functions are available in BASIC INT, RND, and SGN. These are reserved for explanation in Chapter 2.

In place of X, we may substitute any formula or number in parentheses following any of the functions listed above. For example, we may tell the computer to find $\sqrt{4 + X^3}$ by writing SQR(4+X↑3), or the arctangent of $3X - 2e^X + 8$ by writing ATN(3*X-2*EXP(X)+8).

Since we have mentioned numbers and variables, we should be sure we understand how to write numbers for the computer and what variables are allowed.

Numbers

A number may be positive or negative, and it may contain as many as nine digits, but it must be expressed in decimal form. For example, all of the following are numbers in BASIC:

2 -3.675 123456789 -.987654321 483.4156

The following are not numbers in BASIC

14/3 $\sqrt{7}$ .00123456789

The first two are formulas, but not numbers. The last one has more than nine digits. We may tell the computer to find the decimal expansion of 14/3 or $\sqrt{7}$, and to do something with the resulting number, but we may not include either in a list of DATA.

We gain flexibility by use of the letter E, which stands for "times ten to the power." Using E, we can write .00123456789 in several forms acceptable to the computer .123456789E-2 or 123456789E-11 or 1234.56789E-6. We can write ten million as 1E7 and 1969 as 1.969E3. We do not write a number as E7, but must write 1E7 to indicate that it is 1 that is multiplied by 10^7.

Variables

A variable in BASIC is denoted by any letter, or by any letter followed by a single digit. The computer will interpret E7 as a variable, along with A, X, N5, I0, and Ø1. A variable in BASIC stands for a number, usually one that is not known to the programmer at the time the program is written. Variables are given or assigned values by LET, READ, and INPUT statements. All variables have the initial value of zero, so that if you want the starting value of a variable to be zero you need not assign it that value. (Another kind of variable, the string variable, is discussed later on in Chapter 2.)

Although the computer does little in the way of correcting during computation, it will sometimes help you when you forget to indicate absolute value. For example, if you ask for the square root of -7 or the logarithm of -5, the computer will give you the square root of 7 with the error message that you have asked for the square root of a negative number, or the logarithm of 5 with the error message that you have asked for the logarithm of a negative number.

Three other mathematical symbols, symbols of relation, are available in BASIC to indicate any of six standard relations. These are used in IF--THEN statements, where values must be compared. The six possible relations are shown in the following table.

Symbol	Example	Meaning
=	A = B	Is equal to. A is equal to B.
<	A < B	Is less than. A is less than B.
<=	A < = B	Is less than or equal to. A is less than or equal to B.
>	A > B	Is greater than. A is greater than B.
>=	A > = B	Is greater than or equal to. A is greater than or equal to B.
< >	A < >B	Is not equal to. A is not equal to B.

LOOPS

We are often interested in writing a program in which one or more parts are traversed not just once, but a number of times, perhaps with slight changes each time. In order to write the simple program, one in which the part to be repeated is written just once, we use the programming device known as a loop.

The use of loops can be illustrated by two programs for the simple task of printing out a table of the first 100 positive integers together with the square root of each. Without a loop, our program would be 100 lines long:

```
100 PRINT 1,SQR(1)
105 PRINT 2,SQR(2)
110 PRINT 3,SQR(3)
     .
     .
     .
590 PRINT 99,SQR(99)
595 PRINT 100,SQR(100)
600 END
```

With the following program, using one type of loop, we can get the same table with only 5 lines of instruction instead:

```
100 LET X=1
110 PRINT X,SQR(X)
120 LET X=X+1
130 IF X<=100 THEN 110
999 END
```

Statement 100 gives the value of 1 to X, which initializes the loop. Line 110 causes the printing of both 1 and its square root. Line 120 increases the value of X by 1, to 2. Line 130 asks whether X is less than or equal to 100--a Yes answer directs the computer back to line 110. Here it prints 2 and $\sqrt{2}$, and goes to 120. Again X is increased by 1, this time to 3, and at 130 it goes back to 110. The process is repeated--line 110 (print 3 and $\sqrt{3}$), line 120 (X = 4), line 130 (since 4 is less than or equal to 100 go back to line 110), and so on--until the loop has been traversed 100 times. Then X becomes 101. The computer now finds a No answer to the question in line 130 (X is greater than 100, not less than or equal to 100). It therefore does not return to 110 but moves on to line 999, and ends the program.

All loops contain four elements: initialization (line 100 in our program), the body (line 110), modification (line 120), and an exit test (line 130). Because loops are so important, BASIC

provides a pair of statements that specify a loop even more simply than the previous program. They are the FØR and NEXT statements. Their use is illustrated in this program:

```
100 FØR X=1 TØ 100
110 PRINT X,SQR(X)
120 NEXT X
999 END
```

which does exactly the same thing as the two previous programs. In line 100, X is set equal to 1, and a test is set up, like that of line 130 above. Line 120 causes X to be increased by 1, and also carries out the test to decide whether to go back to line 110 or to go on. Thus lines 100 and 120 take place of lines 100, 120, and 130 in the previous program -- and they are easier to use.

Note that the value of X is increased by 1 each time we go through the loop. If we wanted a different increase, we could specify it by writing, for example,

```
100 FØR X=1 TØ 100 STEP 5
```

and the computer would assign 1 to X on the first time through the loop, 6 to X on the second time through, 11 on the third time, and so on the twentieth time. Another step of 5 would take X beyond 100, so the program would go on to the end after printing 96 and its square root. The step value may be either positive or negative. We could obtain the first table printed in reverse order by writing line 100 as

```
100 FOR X=100 TØ 1 STEP -1
```

In the absence of a STEP instruction, a step size of +1 is assumed.

More complicated FØR statements are allowed. The initial value, the final value, and the step size may all be formulas of any complexity. For example, if N and Z have been specified earlier in the program, we could write

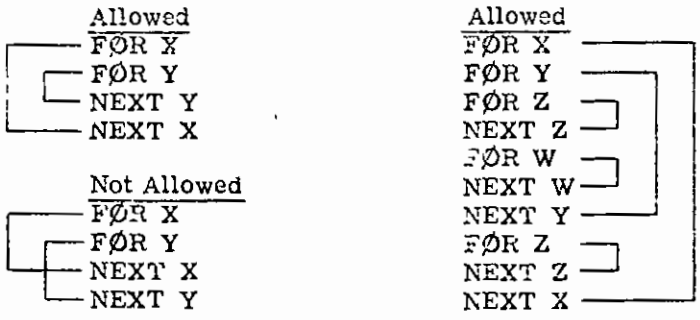
```
250 FØR X=N+7*Z TØ (Z-N)/3 STEP (N-4.Z)/10
```

For a positive step size, the loop continues as long as the control variable is *less* than or equal to the final value. For a negative step size, the loop continues as long as the control variable is *greater* than or equal to the final value

If the initial value is greater than the final value (or less than for a negative step size), then the body of the loop will not be done even once. The computer will immediately pass to the statement following the NEXT. As an example, the following program for adding up the first n integers will give the correct result 0 when n is 0.

```
100 READ N
110 FØR K=1 TØ N
120 LET S=S+K
130 NEXT K
140 PRINT S
150 GØ TØ 100
160 DATA 3,10,0
999 END
```

It is often useful to have loops within loops. These are called nested loops They can be expressed with FØR and NEXT statements. But they must actually be nested and must not cross, as the following skeleton examples illustrate.



LISTS AND TABLES

In addition to the ordinary numeric variables used in BASIC, there are variables that we can use to designate the elements of a list or table. We use these where we would ordinarily use a subscript or a double subscript, as for the coefficients of a polynomial (a_0, a_1, a_2, ...) or the elements of a matrix ($b_{i,j}$). The variables that we use in BASIC consist of a single letter, which we call the name of the list or table, followed by the subscripts in parentheses. For the coefficients of the polynomial we would write A(0), A(1), A(2), and so on; for the elements of the matrix we would write B(1,1), B(1,2), and so on.

We can enter the list A(0), A(1), ...A(10) into a program very simply with four lines:

```
100 FØR I=0 TØ 10
110 READ A(I)
120 NEXT I
130 DATA 2,3,-5,7,2.2,4,-9,123,4,-4,3
```

We need no special instruction to the computer if no subscript greater than 10 occurs. If we want larger subscripts, we must use a dimension (DIM) statement, to tell the computer to save extra space for the list or table. When in doubt, indicate a larger dimension than you expect to use. For example, if we want a list of 15 numbers entered, we might write:

```
100 DIM A(25)
110 READ N
120 FØR I=1 TØ N
130 READ A(I)
140 NEXT I
150 DATA 15
160 DATA 2,3,5,7,11,13,17,19,23,29,31,37,41,43,47
```

Statements 110 and 150 could have been eliminated by writing 120 as FØR I=1 TØ 15, but the form we used allows us to lengthen the list by changing only statement 150, so long as the number of elements in the list does not exceed 25.

We would enter a 3 x 5 table into a program by writing:

```
100 FØR I=1 TØ 3
110 FØR J=1 TØ 5
120 READ B(I,J)
130 NEXT J
140 NEXT I
150 DATA 2,3,-5,-9,2
160 DATA 4,-7,3,4,-2
170 DATA 3,-3,5,7,8
```

We may enter a table with no dimension statement, and the computer will handle all the entries from B(0,0) to B(10,10). But if you try to enter a table with a subscript greater than 10, without a DIM statement, you will get an error message telling you that you have a subscript error. This can be easily corrected by entering, for example, the line:

```
95 DIM B(20,30)
```

which will reserve space for a 20 by 30 table.

The single letter denoting a list or table name may also be used to denote a simple variable without confusion. But the same letter may not be used to denote both a list and a table in the same program.

The form of the subscript is quite flexible. You might have the list item B(I+K) or the table items B(I,K) or Q(A(3,7),B-C).

Let's look now at a sample program that uses both a list and a table. The program computes the total sales of each of five salesmen, each of whom sells the same three products.

```
SALES1     14:38

100 FØR I=1 TØ 3
110 READ P(I)
120 NEXT I
130 FØR I=1 TØ 3
140 FØR J=1 TØ 5
150 READ S(I,J)
160 NEXT J
170 NEXT I
180 FØR J=1 TØ 5
190 LET S=0
200 FØR I=1 TØ 3
210 LET S=S+P(I)*S(I,J)
220 NEXT I
230 PRINT "TØTAL SALES FØR SALESMAN ";J,"$";S
240 NEXT J
250 DATA 1.25,4.30,2.50
260 DATA 40,20,37,29,42
270 DATA 10,16,3,21,8
280 DATA 35,47,29,16,33
999 END

RUN

SALES1     14:39

TØTAL SALES FØR SALESMAN  1    $ 180.5
TØTAL SALES FØR SALESMAN  2    $ 211.3
TØTAL SALES FØR SALESMAN  3    $ 131.65
TØTAL SALES FØR SALESMAN  4    $ 166.55
TØTAL SALES FØR SALESMAN  5    $ 169.4
```

The list P gives the price per [illegible] each of the three products. The table S tells how many items of each product each man sold. As you can see from the program, product number 1 sells for $1.25 per item, number 2 for $4.30 per item, and number 3 for $2.50 per item. You can see also that salesman number 1 sold 40 items of the first product, 10 of the second, and 35 of the third, and so on. The program reads in the price list in lines 100, 110, and 120, using data in line 250. It reads in the sales table in lines 130-170, using data in lines 260-280. The same program could be used again, modifying only line 250 if the prices change, and only lines 260-280 to enter the sales in another month.

The sample program did not need a dimension statement, since the computer automatically saves enough space to allow subscripts to run from 0 to 10. A DIM statement is normally used to save more space. But in a long program, requiring many small tables, DIM may be used to save less space for tables, in order to leave more for the program.

Since a DIM statement is not executed, it may be entered into the program on any line before END. It is convenient, though, to place DIM statements near the beginning of the program.

ERRORS AND DEBUGGING

Occasionally the first run of a new problem will be free of errors and give the correct answers. Usually, though, errors will have to be corrected before the program runs right. Errors are of two types. errors of form that prevent the running of the program, and logical errors in the program that cause the computer to produce either wrong answers or no answers at all.

Errors of form will cause error messages to be printed. The various error messages are listed and explained in Appendix A. Logical errors are often much harder to find, particularly when the program gives answers that are nearly correct. In either case, after you find the errors, you can correct them by changing lines, inserting new lines, or deleting lines from the program. A line is changed by typing it correctly with the same line number. A line is inserted by typing it with a line number between those of two existing lines. A line is deleted by typing its line number and pressing the RETURN key. Notice that you can insert a line only if the original line numbers are not consecutive. For this reason, most programmers start out using line numbers that are multiples of five or ten, but that is a matter of choice.

You can make corrections at any time that you notice them, either before or after a run. Since the computer sorts lines and arranges them in order, a line may be retyped out of sequence. Simply retype the bad line with its original line number.

As with most problems in computing, we can best illustrate the process of finding errors (or bugs) in a program, and correcting (or debugging) it, by an example. Let's consider the problem of finding the value of X between 0 and 3 for which the sine of X is a maximum, and printing out this value of X and the value of its sine. If you have studied trigonometry, you know that $\pi/2$ is the correct value of X, but we shall use the computer to test successive values of X from 0 to 3 First we shall use intervals of .1, then of .01, and finally of .001.

Thus, we shall tell the computer to find the sine of 0, of .1, of .2, of .3, . . ., of 2.8, of 2.9, and of 3, and to determine which of these 31 values is the largest. It will do so by testing SIN(0) and SIN(.1) to see which is larger, and calling the larger of these two numbers M. Then it will pick the larger of M and SIN(.2), and call it M. This number it will check against SIN(.3), and so on down the line. Each time a larger value of M is found, the value of X is remembered in X0. When the computer finished the series, M will have the value of the largest of the 31 sines, and X0 will be the argument that produced that largest value. It will then repeat the search, this time checking the 301 numbers 0, .01, .02, .03, . . ., 2.98, 2.99, and 3, finding the sine of each and checking to see which sine is the largest. Finally, it will check the 3001 numbers 0, .001, .002, .003, . . ., 2.998, 2.999, and 3, to find which has the largest sine. At the end of each of the three searches, we want the computer to print three numbers the value X0 that has the largest sine, the sine of that number, and the interval of search.

Before going to the terminal, we write a program. Let's assume it is the following:

```
100 READ D
110 LET X0=0
120 FØR X=0 TO 3 STEP D
130 IF SIN(X) =M THEN 190
140 LET X0=X
150 LET M=SIN(X0)
160 PRINT X0,X,D
170 NEXT X0
180 GØ TØ 110
190 DATA .1,.01,.001
999 END
```

We shall illustrate the entire sequence on the teletypewriter, and make explanatory comments on the righthand side.

```
NEW
NEW FILE NAME--MAXSIN
READY.

100 READ D
110 LWR X0=0
120 FØR X=0 TØ 3 STEP D
130 IF SIN(X)<=M THEN 190
140 LET X0=X
150 LET M=SIN(X)
160 PRINT XØ,X,D
170 NEXT X0
180 GØ TØ 110
110 LET X0=0
190 DATA .1,.01,.001
999 END
RUN
WAIT.
```

After typing line 130, we notice that LET was mistyped [illegible], so we retype it, this time correctly.

```
MAXSIN     14:42

INCØRRECT  FØRMAT      IN  160
NEXT WITHØUT FØR       IN  170
FØR WITHØUT NEXT
UNDEFINED NUMBER     190
```

When we receive the first error message, we inspect line 160 and find that we used XØ instead of X0 for a variable. The next two error messages refer to lines 120 and 170, where we see that we mixed variables. We correct this by changing line 170 The fourth error message points out that line 130 directed the computer to a DATA statement and not to line 170 where it should go. We correct this by retyping line 130

```
USED       0.83 UNITS.
160 PRINT X0,X,D
170 NEXT X
130 IF SIN(X)<=M THEN 170
RUN
```

```
MAXSIN     14:44

 .1              .1              .1
 .2              .2              .1
 .3              .3
STØP.
READY.

110 LET M=-1
RUN
WAIT.
```

This is obviously incorrect Every value of X is being printed We stop the printing by pressing the BREAK key. Then we ponder the program for a while, trying to figure out what's wrong with it.

We notice that SIN(0) is compared with M on the first time through the loop, but we had assigned no value to M. So we wonder if giving a value less than the maximum value of the sine will do it. We give it the value of -1, by changing line 110, where we had incorrectly initialized X0 instead of M.

```
MAXSIN     14:45

 0               0               .1
 .1              .1              .1
 .2              .2              .1
 .3              .3
STØP.
READY.

160
175 PRINT X0,M,D
RUN
WAIT.
```

We are about to print out almost the same table as before. It is printing out X0, the current value of X, and the interval size each time it goes through the loop

We fix this by moving the PRINT statement outside the loop. We delete line 160, and line 175 is outside of the loop. We also realize that we want M printed and not X.

```
MAXSIN    14:46

 1.6            .999574       .1
 1.6            .999574       .1
 1.6            .999574       .1
 1.6
STOP.
READY.

180 GO TO 100
95 PRINT "X VALUE", "SINE", RESOLUTION"
RUN
WAIT.
```

We see that **we are doing the same thing over and over again, the case for D=.1.** So we stop the printing and inspect the program again.

Of course. Line 180 sent us back to line 110 to repeat the operation rather than back to line 100 to pick up a new value for D. We also decide to put in headings for our columns by a PRINT statement.

```
MAXSIN    14:47    ... 09/15/69

INCORRECT  FORMAT     IN   95
```

There is an error in our PRINT statement: no lefthand quotation mark for the third item.

```
USED       1.00 UNITS.
95 PRINT "X VALUE", "SINE", "RESOLUTION"
RUN
```

We retype line 95 with all of the required quotation marks.

```
MAXSIN    14:48

X VALUE        SINE          RESOLUTION
 1.6            .999574       .1
 1.57           1.            .01
 1.571          1.            .001

OUT OF DATA   IN 100
```

Exactly the desired results. Of the 31 numbers 0, .1, .2, .3, ..., 2.8, 2.9, 3, it is 1.6 that has the largest sine, namely .99957-. Similarly for the finer subdivisions.

```
USED       18.17 UNITS.
LIST
```

The whole process used only 18.53 computer resource units.

```
MAXSIN    14:49

95 PRINT "X VALUE", "SINE", "RESOLUTION"
100 READ D
110 LET M=-1
120 FOR X=0 TO 3 STEP D
130 IF SIN(X)<=M THEN 170
140 LET X0=X
150 LET M=SIN(X)
170 NEXT X
175 PRINT X0,M,D
180 GO TO 100
190 DATA .1,.01,.001
999 END
```

Having changed so many parts of the program, we ask for a list of the corrected program.

```
SAVE

READY.
```

We save the program for later use. This should not be done unless we do expect to use the program later.

In solving this problem, there are two common devices we did not use. One is the insertion of a PRINT statement when we wonder whether the machine is computing what we think we asked it to compute For example, if we wondered about M, we could have inserted 155 PRINT M, and we would have seen the values. The other device is used after several corrections have been made and you are not quite sure what the program now looks like. Simply type LIST or LISTNH, and the computer will type out the program in its current form for you to inspect.

SUMMARY OF ELEMENTARY BASIC STATEMENTS

In this section, we shall give a concise description of each of the types of BASIC statements you will find most useful in writing the simpler kinds of BASIC programs. For each form, we shall assume a line number and use underlining to denote a general type. Thus, variable means any variable, which is a single letter, possibly followed by a single digit.

LET

The LET statement is not a statement of algebraic equality. It is an instruction to the computer to do certain computations and to assign the answer to a certain variable. Each LET statement is of the form

LET variable = formula

Examples:

```
100 LET X=X+1
259 LET W7=(W-X[illegible])
```

READ and DATA

We use a READ statement to assign to the listed variables values obtained from a DATA statement. Neither statement is used without one of the other type. A READ statement causes the variables listed in it to be given, in order, the next available numbers in the collection of DATA statements. Before the program is run, the computer puts all of the DATA statements, in the order in which they appear, into a large data block. Each time a READ statement is encountered anywhere in the program, the data block supplies the next available number or numbers. If the data block runs out of data, with a READ statement still asking for more, the program is assumed to be done.

Since we have to read in data before we can work with it, READ statements normally are placed near the beginning of a program. The location of DATA statements is unimportant, so long as they are in the correct order. A common practice is to put all DATA statements together just before the END statement.

Each READ statement is of the form

READ sequence of variables

and each DATA statement is of the form

DATA sequence of numbers

Examples

```
150 READ X,Y,Z,X1,Y2,Q0
160 DATA 4,2,1.7
170 DATA 6.734E-5,-174.321,3.14159265

200 READ B(K)
210 DATA 2,3,5,7,9,11,10,8,6,4

300 READ R(I,J)
340 DATA -3,5,-9,[illegible]
450 DATA 2.765,5.[illegible]
```

Remember that only numbers are put in a DATA statement, and that 15/7 and $\sqrt{3}$ are formulas, not numbers

INPUT

At times it is desirable to have data entered during running of a program. This is particularly true when one person writes the program and saves it in the computer's memory, and other persons are to supply the data. This may be done by using an INPUT statement, which is very much like a READ statement, but does not draw numbers from a DATA statement. Each INPUT statement is of the form

INPUT sequence of variables

If, for example you want the user to supply values for X and Y in a program, you include the statement

```
140 INPUT X,Y
```

before the first statement that is to use either of the two values. When it encounters the INPUT statement, the computer types a question mark on the printout and waits for input. The user types two numbers, separated by a comma, and presses the return key, and the computer goes on with the rest of the program.

Frequently an INPUT statement is accompanied by a PRINT statement, to make sure that the user knows what the question mark is asking for. If you include in your program

```
120 PRINT "YØUR VALUES ØF X, Y, AND Z ARE",
130 INPUT X,Y,Z
```

the computer will type out

```
YØUR VALUES ØF X, Y, AND Z ARE?
```

Without the semicolon at the end of line 120, the question mark would have been printed on the next line. Data entered by INPUT statements is not saved with the program. Also, it may take a long time to enter a large amount of data using INPUT. INPUT should be used only when small amounts of data are to be entered, or when it is necessary to enter data during the program run, as it is with game playing programs.

PRINT

The PRINT statement has a number of different uses. It is discussed in more detail in Chapter 2. The common uses are:

A. To print out the result of some computations
B. To print out verbatim a message included in the program
C. To do a combination of A and B
D. To skip a line

We have seen examples of only A and B in our sample programs. Each type is slightly different in form, but all start with PRINT after the line number.

Examples of type A:

```
100 PRINT X,SQR(X)
135 PRINT X,Y,Z,B*B-4*A*C,EXP(A-B)
```

The first will print the value of X and then, a few spaces to the right, its square root. The second will print five different numbers X, Y, B^2-4AC, and e^{A-B}. The computer will compute the two formulas and print the values for you, if you have already given values to A, B, and C. It can print up to five numbers per line in this format.

Examples of type B:

```
100 PRINT "NØ UNIQUE SØLUTIØN"
430 PRINT "X VALUE", "SINE", "RESØLUTIØN"
```

You have seen both in the sample programs. The first prints the statement within the quotation marks. The second prints the three labels with spaces between them. The labels in 430 automatically line up with three numbers called for in the PRINT statement, as seen in the program MAXSIN.

Examples of type C.

```
170 PRINT "THE VALUE ØF X IS";X
315 PRINT "THE SQUARE RØØT ØF";X;"IS";SQR(X)
```

If the first has computed the value of X as 3, it will print out

```
THE VALUE ØF X IS 3
```

If the second has computed the value of X as 625, it will print out

```
THE SQUARE RØØT ØF 625 IS 25
```

In statements of type C, the semicolon is used to minimize space.

Example of type D:

```
250 PRINT
```

The computer will advance the paper one line when it encounters this statement.

GØ TØ

There are times in a program when you do not want all statements executed in the order in which they appear in the program. An example of this occurs in the MAXSIN program where the computer has computed X0, M, and D and printed them out in line 160. We did not want the program to go on to the END statement yet, but we wanted it to go through the same process for a different value of D. So we directed the computer to go back to line 100 with a GØ TØ statement. Each GØ TØ statement is of the form

GØ TØ line number

Example

```
150 GØ TØ 75
```

ØN—GØ TØ

The simple GØ TØ statement provides a single branched switch. The ØN--GØ TØ statement provides a multi-branched switch. The form of the statement is

ØN expression GØ TØ line number, line number, . . . , line number

The expression is any valid BASIC expression, and the line numbers are those to which the statement will transfer depending on the value of the expression.

Example:

```
230 ØN X+Y GØ TØ 575, 490, 150
```

This statement will transfer to line 575, 490, or 150 depending on whether the value of the expression X+Y is 1, 2, or 3.

The expression value will be truncated to an integer if it is not already an integer. For example, if X+Y equals 2.5, the value will be truncated to 2, and the program will branch to line 490, the second line number in the list.

Branching to a line containing a DIM, REM, or DATA statement is not allowed. As many line numbers may be included in an ØN---GØ TØ statement as will fit on one line.

IF—THEN

At times we want to jump the normal sequence of statements if a certain relationship holds. For this we use an IF--THEN statement, sometimes called a conditional GØ TØ statement. Such a statement occurred at line 130 of MAXSIN. Each IF--THEN statement is of the form

IF formula relation formula THEN line number

Examples:

```
340 IF SIN(X)<=M THEN 630
120 IF G=0 THEN 165
```

The first statement asks whether the sine of X is less than or equal to M, and tells the computer to go to line 630 if it is. The second statement asks if G is equal to zero, and tells the computer to go to line 165 if it is. In each case, if the answer to the question is No, the computer will go on to the next line of the program.

FØR and NEXT

We have already encountered the FØR and NEXT statements in loops, and have seen that they go together, one at the beginning of the loop and one at the exit, directing the computer back to the entrance. The FØR statement is of the form

FØR variable = formula TØ formula STEP formula

Most commonly, the formulas will be integers and the STEP will be omitted, which means that a step size of plus one is assumed. The accompanying NEXT statement is simple in form, but the variable must be exactly the same one as that following FØR in the FØR statement. The form of the NEXT statement is

NEXT variable

Examples:

```
130 FØR X=0 TØ 3 STEP D
180 NEXT X

120 FØR X4=(17+CØS(Z))/3 TØ 3*SQR(10) STEP 1/4
235 NEXT X4

240 FØR X=8 TØ 3 STEP -1
270 NEXT X

456 FØR J=-3 TØ 12 STEP 2
470 NEXT J
```

Notice that the step size may be a formula (1/4), a negative number (-1), or a positive number (2). In the example with lines 120 and 235, the successive values of X4 will be

.25 apart, in increasing order. In the next example, the successive values of X will be 8, 7 6, 5, 4, 3. In the last example, on successive trips through the loop J will take on values -3, -1, 1, 3, 5, 7, 9, and 11.

If the initial, final, or step size values are given as formulas, the formulas are evaluated once and for all upon entering the FØR statement. The control variable can be changed in the body of the loop. Of course the exit test always uses the latest value of this variable.

If you write 150 FØR Z=2 TØ -2 without a negative step size, the loop will not be executed, and the computer will go immediately to the statement following the corresponding NEXT statement.

DIM

Whenever we want to enter a list or table with a subscript greater than 10 we must use a DIM statement to tell the computer to save enough room for the list or table.

Examples:

120 DIM H(35)
135 DIM Q(5,25)

The first statement would enable us to enter a list of 35 items--35 if we use H(0) - and the second a table 5 x 25--or 6 x 26 if we use row 0 and column 0.

END

Every program must have an END statement, and it must be the statement with the highest line number in the program. Its form is simple: a line number with END.

Example:

99 END

2. Advanced BASIC

In Chapter 1, you learned how to write programs in BASIC. In this chapter we will discuss some capabilities of BASIC that were not discussed yet. These include:

- Alphanumeric data and string manipulation
- Files
- Matrices

We will also consider some advanced capabilities in printing output, several functions that we have not yet mentioned, and several statements either in more detail or for the first time. And, finally, we will consider two sample BASIC programs that make use of many of the advanced capabilities of BASIC.

ALPHANUMERIC DATA AND STRING MANIPULATION

Alphanumeric data, names, and other identifying information can be handled in BASIC using string variables. You can enter, store, compare, and print out alphanumeric and certain special characters in the Mark I character set.

A *string* is any sequence of alphanumeric and certain special characters in the Mark I character set not used for control purposes in the Mark I system. *String size* is limited to 15 valid characters.

A *string variable* is denoted by a letter followed by a dollar sign. For example, A$, B$, and X$ denote string variables.

DIM

Strings can be set up as one-dimensional arrays only. If you request a two-dimensional array you will receive the error message DIMENSIØN TØØ LARGE.

Examples:

```
100 DIM A(5),C$(20),A$(12),D(10,5)
200 DIM R$(35)
300 DIM M$(15),B$(15)
```

In line 100, only C$ and A$ are string variables. R$, as dimensioned in line 200, will save storage space for 35 fifteen character arrays. Any or all of the 35 strings may in fact be less than fifteen characters long.

LET

Strings and string variables may appear in only two forms of the LET statement. The first is used to replace a string variable with the contents of another string variable.

Example:

```
156 LET G$=H$
```

The second is used to assign a string to a string variable.

Example

```
160 LET J$="THIS STRING"
```

Arithmetic operations may not be done on string variables. Requests for addition, subtraction, multiplication, or division involving string variables produce the error message ILLEGAL STRING ØPERATIØN AT XXX.

The LET statement permits multiple variable replacement.

Example:

```
262 LET X=Y=Z=21*N/2
495 LET A$=G$=J$="THIS STRING"
```

The first statement places the value of the expression 21*N/2 in variables X, Y, and Z. The second statement assigns the string THIS STRING to variables A$, G$, and J$. Any valid expression or string may be used.

READ and DATA

READ statements can contain string variables intermixed with ordinary variables. In the corresponding DATA statements, every item corresponding to a string variable in the READ statement must be a valid string. If the string contains any characters that have special meaning in BASIC--such as commas, semicolons, leading or trailing spaces, and so on - it must be enclosed in quotation marks. Unquoted strings must begin with an alphabetic character.

Example:

```
100 READ A$,B$,C$,D$,A,E$
200 DATA THE," ","PEØPLE,",YES--,500,ØF THEM.
300 PRINT A$,B$,C$,B$,D$;A;E$
999 END
```

This program will print out THE PEØPLE, YES-- 500 ØF THEM. The DATA statement has quotation marks around B$ because it is a blank space, and around C$ because it includes a comma.

INPUT

Like READ and DATA statements, INPUT statements can contain string variables intermixed with ordinary variables. Every item corresponding to a string variable in the INPUT statement must be a valid string variable. If the string contains characters that have special meaning in BASIC, it must be enclosed by quotation marks. If the string begins with other than an alphabetic character it must be enclosed in quotation marks.

Example:

```
110 INPUT L$(17),M$,N$(I)
```

PRINT

The PRINT statement also can contain string variables intermixed with ordinary variables. When a string variable is encountered that has not been assigned, the PRINT statement will produce for that variable a string of 15 blank spaces. A semicolon after a string variable in a PRINT statement causes the printout of the variable following that string to be directly connected to the string variable.

Examples:

```
135 PRINT A,16,B$,C$,N
140 PRINT 100+I, "DATA",L$,M$,N$
150 PRINT S$
```

IF—THEN

Only one string variable is allowed on each side of the IF-THEN relation sign. All of the six standard relations are valid (=, <>, <, >, <=, and >=). When strings of different lengths are compared, the shorter string and the corresponding part of the longer string will be used. If they compare, the shorter string is taken to be the lesser of the two.

Examples:

```
100 IF N$="SMITH" THEN 105
200 IF A$<>B$ THEN 205
300 IF "JUNE"<=M$ THEN 305
400 IF D$>="FRIDAY" THEN 600
```

You must use quotation marks around the string to be compared, as above, unless it is referenced in the IF--THEN statement by a string variable name.

Characters are compared in their BASIC code representations. The collating sequence used in comparing is listed in Appendix C.

MORE ABOUT PRINTING

Although the format of the printout is automatically supplied for the beginner, the PRINT statement, the TAB function, and image formatted output permit a greater flexibility for the advanced programmer in setting up different formats for his output.

PRINT

The teletypewriter line is divided into five zones of fifteen spaces each. Some control of the use of these zones comes from the use of the comma. A comma is a signal to move to the next print zone or, if the fifth print zone has been used, to move to the first print zone on the next line.

Shorter zones can be made by use of the semicolon. The zones are three spaces long for 1-digit numbers, six spaces long for 2-digit, 3-digit, and 4-digit numbers, nine spaces long for 5-digit, 6-digit, and 7-digit numbers, twelve spaces long for 8-digit, 9-digit, and 10-digit numbers, and fifteen spaces long for 11-digit numbers. As with the comma, a semicolon is a signal to move to the next short print zone or, if the last such zone on the line has been used, to move to the first print zone of the next line.

The first space in any print zone is reserved for the sign, even though it is not printed out if it is plus.

If you typed the program

```
100 FØR I=1 TØ 15
110 PRINT I
120 NEXT I
999 END
```

the teletypewriter would print 1 at the beginning of the first line, 2 at the beginning of the next line, and so on, finally printing 15 at the beginning of the fifteenth line. But if you changed line 110 to read

```
110 PRINT I,
```

you would have the numbers printed in zones, reading

```
 1          2          3          4          5
 6          7          8          9          10
 11         12         13         14         15
```

If you wanted the numbers printed in the same fashion, but more tightly packed, you could change line 110 to replace the comma with a semicolon

```
110 PRINT I,
```

and the result would be printed

```
 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
```

You should remember that a label inside quotation marks is printed just as it appears, and also that the end of a PRINT statement signals a new line, unless a comma or semicolon is the last symbol. The instruction

```
150 PRINT X,Y
```

will result in the printing of two numbers and the return to the next line, but

```
150 PRINT X, Y,
```

will result in the printing of two numbers and no return. The next number to be printed will be printed in the third zone on the same line as the values of X and Y.

Since the end of a PRINT statement signals a new line, a statement such as

```
250 PRINT
```

will cause the teletypewriter to advance the paper one line. It will put a blank line in your printout, if you want to use it for vertical spacing of your results, or it will cause the completion of a partly filled line, as illustrated in the following part of a program:

```
100 FØR M=1 TØ N
110 FØR  J=0 TØ M
120 PRINT B(M,J);
130 NEXT J
140 PRINT
150 NEXT M
```

The program will print B(1,0) and next to it B(1,1). Without line 140, the teletypewriter would then go on printing B(2,0), B(2,1), and B(2,2) on the same line, and even B(3,0), B(3,1), and so on, if there were room. Line 140 directs the teletypewriter to start a new line after printing the B(1,1) value corresponding to M=1, and to do the same thing after printing the value of B(2,2) corresponding to M=2, and so on.

TAB

The print function TAB permits tabbing of the teletypewriter. Whenever the TAB function is used in the PRINT statement, it will cause the print head to move over to the position indicated by the argument of TAB.

Example:

```
150 PRINT X; TAB(10); Y, TAB(2*N), Z
```

The argument of TAB refers to a print position on the teletypewriter line. The positions are assumed to run from 0 through 74. In the example, if the value of N is 10, the print head will move to the 10th print position after printing the value of X, and to the 20th print position after printing the value of Y.

When using the TAB function, you should use the semicolon in the PRINT statement in order to minimize field width.

If the argument of TAB is less than the current teletypewriter print head position, it is ignored

All arguments of TAB are treated modulo 75.

Rules for Printing Numbers

The following rules for the printing of numbers will help you in interpreting printed results.

- If a number is an integer, the decimal point is not printed. If the integer is larger than or equal to 2^{30} (i.e. 1,073,741,824), the teletypewriter will print the first digit, followed by a decimal point (2) the next five digits, and (3) an E followed by the appropriate exponent integer. For example, it will print 32,437,580,259 as 3.24376E+10.
- For any decimal number, no more than six significant digits are printed
- For a number less than 0.1, the E notation is used unless the entire significant part of the number can be printed as a six decimal number. For example, .03456 means that the number is exactly 0345600000, but 3.45600E-2 means that the number has been rounded to .0345600.
- Trailing zeroes after the decimal point are not printed.

The following program, in which we print out the first 45 powers of 2, shows how numbers are printed.

```
100 FOR I=1 TO 45
110 PRINT 2^I;
120 NEXT I
999 END
RUN


PRINT      14:53

 2   4   8   16    32    64    128   256   512   1024  2048  4096  8192
 16384    32768    65536    131072   262144   524288   1048576  2097152
 4194304  8388608  16777216    33554432    67108864    134217728
 268435456   536870912   1.07374E+09    2.14748E+09    4.29497E+09
 8.58993E+09    1.71799E+10    3.43597E+10    6.87195E+10    1.37439E+11
 2.74878E+11    5.49756E+11    1.09951E+12    2.19902E+12    4.39805E+12
 8.79609E+12    1.75922E+13    3.51844E+13
```

PRINT USING AND IMAGE STATEMENTS

You can set up formatted line output by use of the PRINT USING and image statements.

The form of the PRINT USING statement is

PRINT USING line number, output list

where the line number is that of the image statement to be used in formatting the output line, and the output list can consist of numbers, variables, string constants, string variables, and functions.

The form of the image statement is

line number:line image

where the line number is that of the image statement in the program, and the line image consists of format control characters and printable constants.

Format control characters are

'	(apostrophe) a one-character field that is filled with the first character in an alphanumeric string regardless of the string length.
" "	(quotation marks) the replacement field of an alphanumeric string of two or more characters, the field width includes the quotation marks as well as the characters (if any) contained within the marks.
#	(pound sign) the replacement field for a numeric character.
↑↑↑↑	(four up-arrows) indicates scientific notation for a numeric field.

All other characters are treated as printable constants.

The following simple example is part of a program, showing the use of the PRINT USING statement, the line image statement, and format control characters.

Example:

```
110 PRINT USING 120,A$,"$",A,324,X
120:"   "       '###.##       #####      ##.##↑↑↑↑
```

If the values of A$, A, and X are FIRST, 12.9, and 24687, output is

```
FIRST      $ 12.90         324       2.47E+04
```

An image statement must begin with a colon. It is composed of fields which form the print line. There are five types of fields:

- Integer fields
- Decimal fields
- Exponential fields
- Alphanumeric fields
- Literal fields

Integer Fields

The following rules apply to integer fields.

- An integer field is composed of pound signs (#).
- Numbers in an integer field are right justified and truncated if they are not integral.
- The field will be widened to the right if the number is too big.
- The field must reserve a place for the algebraic sign.
- If the number is greater than 1,073,741,823 in absolute value, an asterisk will be printed.

Example:

```
100: ####   #####   ###
110 READ A,B,C
120 PRINT USING 100,A,B,C
130 GØ TØ 110
140 DATA 123.45,-34.856,45.7,457.34,-17,89.999
999 END
RUN

PRINTU    15:00

   123     -34     45
   457     -17     89

ØUT ØF DATA  IN 110
```

Decimal Fields

The following rules apply to decimal fields.

- A decimal field is a string of pound signs (#) with an imbedded period. Note that .### is not a decimal field because the period is not imbedded.
- The number will be rounded to the number of places specified by the pound signs following the decimal.
- The number is right justified, placing the decimal as given in the field definition.
- The field will be widened to the right if the number is too large.
- The field must include a place for the algebraic sign.

Example:

```
100: ####.##    ####.####    #####.    #.###
110 READ A,B,C,D
120 PRINT USING 100,A,B,C,D
130 GØ TØ 110
140 DATA 123.456,-34.856,47.7,-.0177
150 DATA 1.999,876.55,-17,.893
999 END
RUN

PRINTU    15:03

   123.46      -34.8560       48.     -.018
     2.00      876.55D0      -17.      .893

ØUT ØF DATA  IN 110
```

Exponential Fields

The following rules apply to exponential fields.

- An exponential field is a decimal field followed by four up-arrows(↑), which reserve a place for the exponent.
- The pound signs preceding the decimal represent the factor by which the exponent will be adjusted.
- The number will be rounded as with decimal fields.
- A place must be reserved for the sign.

Example:

```
100: #.#####↑↑↑↑    ##.###↑↑↑↑    ###.↑↑↑↑    #.##↑↑↑↑
110 READ A,B,C,D
120 PRINT USING 100,A,B,C,D
130 GØ TØ 110
140 DATA 123.456,-34.856,47.7,-.0177
150 DATA 1.999,876.55,-17,.893
999 END
RUN

PRINTU    15:05

   .12346E+03    -3.486E+01     48.E+00    -.18E-01
   .19990E+01     8.766E+02    -17.E+00     .89E+00

ØUT ØF DATA  IN 110
```

Alphanumeric Fields

The following rules apply to alphanumeric fields.

- The apostrophe is used to print the first character from a string variable or quoted constant.
- A field bounded by quotation marks is used to print two or more characters.
- In an alphanumeric field of two or more characters, the string is left justified within the field and blank filled or truncated on the right.

Example:

```
100:  "23456"    "THE NAME GØES HERE"    '    ''
110 READ A$,B$,C$,D$,E$
120 PRINT USING 100,A$,B$,C$,D$,E$
130 DATA ABCDEFGHI
140 DATA ABCDEF
150 DATA ABC
160 DATA ABC
170 DATA ABC
999 END
RUN

PRINTU    15:08

  ABCDEFG    ABCDEF                  A    AA
```

Literal Fields

A literal field is composed of characters or character strings that are not control characters. It will appear on the print line exactly as it appears in the image.

Example:

```
100:  THE VALUE FØR A IS    '####.##
110 LET A=100.54
120 LET A$="$"
130 PRINT USING 100,A$,A
999 END
RUN

PRINTU     15:09

  THE VALUE FØR A IS    $ 100.54
```

General Rules

The following rules apply in general to formatted line output.

- A program may contain up to 100 images.
- The list elements in the PRINT USING statement may be expressions, variables, numeric constants, and quoted literals.
- Numeric list elements must replace numeric fields, and alphanumeric elements must replace alphanumeric fields, or you will receive the error message BAD IMAGE.
- If the output list contains more elements than there are replaceable fields in the image statement, a carriage return is supplied after the last field in the image, and the image is reused. The extra elements will be printed on a second line only if they match the image fields that are to be used.

Example:

```
100 PRINT USING 120
110 PRINT
120:          I        I↑2        I↑3
130:     ###### ######## ########
140 FØR I=1 TØ 6
150 LET A(I)=I
160 LET B(I)=I↑2
170 LET C(I)=I↑3
175 NEXT I
180 FØR I=1 TØ 6 STEP 2
190 PRINT USING 130,A(I),B(I),C(I),A(I+1),B(I+1),C(I+1)
200 NEXT I
999 END
RUN

PRINTU     15:14

           I        I↑2        I↑3

           1          1          1
           2          4          8
           3          9         27
           4         16         64
           5         25        125
           6         36        216
```

The following program demonstrates one kind of application in which the formatted output line is useful.

Example:

```
100 PRINT USING 170
110 PRINT
120 FOR I=1 TO 4
130 READ A$,A,B
140 LET T=A*B
150 PRINT USING 180,A$,A,B,"$",T
160 NEXT I
170:NAME            HRS WORKED          RATE          PAY
180:"       "           ###.##       ##.###/HR    $####.##
190 DATA ANDREWS,47.5,3.987,KELLY,40,2.865,MANLEY,46,3.020
200 DATA ZUMPANO,42.34,4.255
999 END
RUN
WAIT.
```

```
FORMAT    15:19

NAME            HRS WORKED          RATE          PAY

ANDREWS             47.50        3.987/HR    $ 189.38
KELLY               40.00        2.865/HR    $ 114.60
MANLEY              46.00        3.020/HR    $ 138.92
ZUMPANO             42.34        4.255/HR    $ 180.16
```

FUNCTIONS

There are two functions that were listed in Chapter 1 but not described: INT and RND.

Three other functions that you will sometimes find useful are SGN, CLK, and TIM. And you can write your own functions by use of the DEF statement.

INT

The INT function is the one that frequently appears in algebraic computation as [x], and it gives the greatest integer not greater than x. Thus INT(2.35) equals 2, INT(-2.35) equals -3, and INT(12) equals 12.

One use of the INT function is to round numbers. We can use it to round to the nearest integer by asking for INT(X+.5). This will round 2.9, for example, to 3, by finding

INT(2.9+.5) = INT(3.4) = 3

You should convince yourself that INT(X+.5) will do the rounding guaranteed for it, that it will round a number midway between two integers up to the larger of the integers.

It can also be used to round to any specific number of decimal places. For example, INT(10*X+.5)/10 will round X correct to one decimal place; INT(100*X+.5)/100 will round X correct to two decimal places, and INT(X*10↑D+.5)/10↑D will round X correct to D decimal places

RND

The function RND is a pseudo random number generator. It requires a single argument, which has the following meanings.

- If the argument is positive, the argument is used to initiate the random number sequence.
- If the argument is negative, a random number is used to initiate the random number sequence.
- If the argument is zero, RND will supply a random number. The first use of RND(0) in a program will always yield the same random number.

A positive or negative argument would probably be used to initiate a sequence of random numbers, after which a zero argument would be used repeatedly.

If the initial value used for the argument is any power of 2, the same initial random number results as when 2 is used.

If we want the first twenty random numbers, we can use the following program to get twenty six-digit decimals.

Example:

```
100 LET X=RND(1)
110 FØR L=1 TØ 20
120 PRINT RND(0),
130 NEXT L
999 END
RUN



RNDTST     15:23

 .473599          .442519          .498805          .373168          .921321
 .123978          8.68505E-02      4.06526E-02      .341097          .468896
 2.97623E-02      .75441           .498551          .242898          9.31652E-02
 .250064          .159309          .211611          .042684          .383241
```

If, on the other hand, we want twenty random one-digit integers, we can change line 120 to read

```
120 PRINT INT(10*RND(0));
```

and we then obtain

```
RNDTST     15:24

 4  4  4  3  9  1  0  0  3  4  0  7  4  2  0  2  1  2  0  3
```

We can vary the kind of random numbers we get. For example, if we want twenty random numbers ranging from 1 to 9 inclusive, we can change line 120 as shown below

```
120 PRINT INT(9*RND(0)+1);
RUN



RNDTST     15:25

 5  4  5  4  9  2  1  1  4  5  1  7  5  3  1  3  2  2  1  4
```

Or we can obtain random numbers which are integers from 5 to 24 inclusive by changing line 120 as follows

```
120 PRINT INT(20*RND(0)+5)
RUN


RNDTST     15:26

 14     13     14     12     23     7  6  5  11     14     5  20     14     9  6
 10     8  9  5  12
```

In general, if we want our random numbers to be chosen from A integers of which B is the smallest, we would call for

INT(A*RND(0)+B)

after first having initiated the random number sequence with a positive or negative argument, as in line 100 of our sample program.

If you were to run the first version of our sample program again, you would get the same twenty numbers in the same order. But we can get a different set by throwing away some of the random numbers. In the following program we find the first ten random numbers and do nothing with them. We then find the next twenty and print them. You can see, by comparing this with the earlier program, that the first ten of these random numbers are the same as the second ten of the first program.

Example

```
100 LET Z=RND(1)
110 FOR I=1 TO 10
120 LET Y=RND(0)
130 NEXT I
140 FOR I=1 TO 20
150 PRINT RND(0),
160 NEXT I
999 END
RUN


RNDTST     15:28

 2.97623E-02      .75441          .498551          .242898          9.31652E-02
 .280064          .159309         .211611          .042684          .383241
 8.89948E-02      .140658         .643944          .829565          5.32346E-02
 .795665          .66257          .384661          .817653          .942257
```

SGN

The function SGN allows you to test for the sign of any value. The form is SGN(argument) and it yields +1, -1, or 0 depending on the value of the argument. The options are

Argument Value	Yields
Zero	0
Positive, not zero	+1
Negative, not zero	-1

Examples

SGN(0) yields 0
SGN(-1.82) yields -1
SGN(989) yields +1
SGN(-.001) yields -1
SGN(-0) yields 0

CLK

The function CLK(X), X being a dummy argument, yields the time of day in military hours.

Examples:

```
100 PRINT CLK(X)
295 IF CLK(X)>15.00 THEN 1000
130 IF A-CLK(X)<.5 THEN 1000
```

TIM

The function TIM(X), X being a dummy argument, yields the program elapsed time in seconds.

Examples:

```
100 PRINT TIM(X)
688 IF TIM(X)>10 THEN 1000
```

DEF

In addition to making use of the standard functions, you can define any other function that you expect to use several times in your program. You use a DEF statement to define such a function. The name of the defined function must be three letters, the first two of which are FN. Hence you can define up to 26 functions in one program: FNA, FNB, FNC, and so on.

The usefulness of DEF you can see in a program, for example, where you often need the function e^{-x^2}. You introduce the function by the line

```
130 DEF FNE(X)=EXP(-X↑2)
```

and later on call for various values of the function by FNE(.1), FNE(3.45), FNE(A+2), and so on. DEF can be a great time-saver when you want values of some function for a number of different values of the variable.

The DEF statement may be put anywhere in the program, and the expression to the right of the equal sign may be any formula that can be fitted on one line. It may include any combination of other functions, including ones defined by other DEF statements, and it can involve other variables besides the one denoting the argument of the function. For example, if FNR is defined by

```
170 DEF FNR(X)=SQR(2+LOG(X)-EXP(Y*Z)*(X+SIN(2*Z)))
```

and you have previously assigned values to Y and Z, you can ask for FNR(2.175). You can give new values to Y and Z before the next use of FNR, if you want to.

DEF is generally limited to cases where the value of the function can be computed within a single BASIC statement. Often much more complicated functions, or even pieces of a program, must be calculated at several different points within a program. For these functions, the GØSUB statement will frequently be useful. It is described in the next section.

ADDITIONAL BASIC STATEMENTS

Several kinds of BASIC statements were not covered in Chapter 1. These are discussed in this section. They are:

- GØSUB and RETURN
- CALL
- STØP
- REM
- RESTØRE
- CHAIN

GØSUB and RETURN

When a particular part of a program is to be used more than one time, or possibly at several different places in the overall program, it is most efficiently programmed as a subroutine. The subroutine is entered with a GØSUB statement.

Example:

```
190 GØSUB 310
```

The line number, 310, is the line number of the first statement in the subroutine.

The last line of the subroutine should be a RETURN statement, directing the computer to return to the earlier part of the program.

Example:

```
450 RETURN
```

This statement, if it is the last line in the subroutine entered in the previous example, tells the computer to go back to the first line numbered greater than 190 and continue the program there.

You may use a GØSUB statement inside a subroutine to execute yet another subroutine. This is called nested GØSUBs. It is absolutely necessary that a subroutine be left only with a RETURN statement. Using a GØ TØ or an IF--THEN to get out of a subroutine will not work correctly. You may have several RETURNs in the subroutine so long as only one of them will be used.

You should be very careful not to write a program in which a GØSUB appears inside a subroutine that refers to one of the subroutines already entered. Recursion is not allowed.

The following example, a program for determining the greatest common divisor of three integers using the Euclidean Algorithm, illustrates the use of a subroutine.

Example.

```
100 PRINT "A","B","C","GCD"
110 READ A,B,C
120 LET X=A
130 LET Y=B
140 GØSUB 230
150 LET X=G
160 LET Y=C
170 GØSUB 230
180 PRINT A,B,C,G
190 GØ TØ 110
200 DATA 60,90,120
210 DATA 38456,64872,98765
220 DATA 32,384,72
230 LET Q=INT(X/Y)
240 LET R=X-Q*Y
250 IF R=0 THEN 290
260 LET X=Y
270 LET Y=R
```

```
280 GØ TØ 230
290 LET G=Y
300 RETURN
999 END
RUN



GCD3NØ      15:30

A              B              C              GCD
 60             90             120            30
 38456          64872          98765          1
 32             384            72             8

ØUT ØF DATA  IN 110
```

The first two numbers are selected in lines 120 and 130, and their GCD is determined in the subroutine, lines 230 - 300. The GCD just found is called X in line 150, the third number is called Y in line 160, and the subroutine is entered from line 170 to find the GCD of these two numbers. This number, the GCD of the three given numbers, is printed out with the three numbers in line 180

CALL

The CALL statement is used to call an external program for use as a subroutine within the main program just as the GØSUB statement calls a subroutine inside the main program. The statement form is CALL saved program name.

Examples:

```
100 CALL HISDWN
200 CALL EQCLS*
```

You can call either previously saved programs of your own, as in line 100; common programs in your catalog library, as in line 200; or system library programs, either regular or run-only.

The standard program naming rules apply.

Examples:

```
140 CALL A B
150 CALL AB
```

Statements 140 and 150 both call a program named AB, since BASIC ignores all leading, trailing, and imbedded blanks. No arguments are permitted after the program name in the CALL statement. Subroutines may call other routines, but no program may call the main program or itself.

The return from a subroutine to the calling program is by a RETURN statement. Multiple returns are permissible. The return is always to the statement immediately following the statement in which the program was called.

All variables and defined functions are common to the main program and the called subroutines. They need not be defined separately in each program.

Example

```
NEW
NEW FILE NAME--DEFPRT
READY.

100 LET Y=4
110 LET X=7
220 LET A=FNP(3)
230 RETURN
999 END
SAVE
WAIT.

READY.

NEW
NEW FILE NAME--MAIN
READY.

100 DEF FNP(Z)=SQR(X↑2+Y↑2+Z↑2)
110 CALL DEFPRT
120 PRINT A
999 END
RUN
WAIT.

MAIN       15:35


 8.60233
```

Statement 110 calls DEFPRT, which stores 4 in Y and 7 in X, and calculates a value for A by use of a function defined in line 100 of MAIN. The function uses the value 3 for Z as defined in statement 220 of DEFPRT. DEFPRT then returns to the statement immediately following the CALL statement, and the calculated value for A, 8.60233, is printed.

An END or STØP statement terminates all execution, whether it is executed in a subroutine or in the main program.

The line numbers in the different programs are completely independent. GØ TØ and IF--THEN statements reference line numbers in their own program only.

Data is compiled from the main program first, and then from each of the called programs in the order in which the CALL statements are encountered.

Consider the following example.

Example:

```
OLD
OLD FILE NAME--SUBR


READY.

LIST


SUBR       15:38

100 READ X,Y,Z
110 DATA 6,7,8
120 RETURN
999 END
```

```
ØLD
OLD FILE NAME--MAIN

READY.

LIST

MAIN        15:39

100 READ A,B
110 CALL SUBR
120 READ C,D,E
130 DATA 1,2
140 DATA 3,4,5
150 PRINT A,B
160 PRINT X,Y,Z
170 PRINT C,D,E
999 END

RUN

MAIN        15:40

 1             2
 3             4             5
 6             7             8
```

Statement 100 reads the numbers 1 and 2 into A and B. Statement 110 transfers control to SUBR, which reads 3, 4, and 5 (not 6, 7, and 8) into variables X, Y, and Z. After the return to statement 120, 6, 7, and 8 are read into C, D, and E.

The CALL statement allows more effective use of program space available. A program referred to by a CALL statement will be compiled only once no matter how many times it is called in each of the routines. Object code generated by the called program counts toward object code limitation, but the characters in the called program do not count toward the BASIC character limitation.

As many as 10 different programs may be called.

STØP

The STØP statement is equivalent to GØ TØ XXXXX, where XXXXX is the line number of the END statement in the program. It is useful in programs having more than one natural finishing point. For example, the following two parts of programs are exactly equivalent.

Example:

```
250 GØ TØ 999              250 STØP
     :                          :
340 GØ TØ 999              340 STØP
     :                          :
999 END                    999 END
```

REM

The REM statement allows you to insert explanatory remarks in a program. The form is REM any comment. The computer completely ignores the part of the line following REM, allowing you to include directions for using the program, identifications of the parts of a long program, or anything else. Although what follows REM is ignored, you may use the line number of a REM statement in a GØSUB or IF--THEN statement.

Examples

```
100 REM INSERT DATA IN LINES 900-998.  THE FIRST
110 REM NUMBER IS N, THE NUMBER OF POINTS.  THEN
120 REM THE DATA POINTS THEMSELVES ARE ENTERED, ET


200 REM THIS IS A SUBROUTINE FOR SOLVING EQUATIONS.
      .
      .
      .
300 RETURN
      .
      .
      .
520 GOSUB 200
```

RESTORE

Sometimes it is necessary to use data in a program more than once. The RESTORE statement permits reading the data as many additional times as necessary. Whenever RESTORE is encountered in a program, the computer restores the data block pointer to the first item of data. A subsequent READ statement will then start reading the data all over again.

One word of warning if the data items you wish to use again are preceded by code numbers or parameters, superfluous READ statements should be used to pass over the numbers.

As an example, the following part of a program reads the data, restores the data block to its original state, and reads the data again. Note the use of line 570 to pass over the value of N, which is already known.

Example

```
100 READ N
110 FOR I=1 TO N
120 READ X
      .
      .
      .
200 NEXT I
      .
      .
      .
560 RESTORE
570 READ X
580 FOR I=1 TO N
590 READ X
```

CHAIN

The CHAIN statement allows you to stop the execution of the current program and begin compilation and execution of another program without direct intervention. The CHAIN statement is equivalent to giving the commands STOP, OLD, a program name, and RUN.

The statement form is

CHAIN saved program name

or

CHAIN saved program name, line number

Only one program name may appear in a statement. The name must conform to the rules used in naming BASIC programs.

Examples:

```
100 CHAIN NEXT
100 CHAIN NEXT,100
100 CHAIN PLØTER***
100 CHAIN TUT01$***
100 CHAIN PAYRØL,555
```

Notice that, as shown in the examples, BASIC library programs may be chained.

When a number appears after the saved program name, as in the second and fifth lines of the examples, the number indicates the line number of the named program at which execution is to begin. When no number appears, execution begins with the first executable statement of the named program.

Once a CHAIN statement is executed, the current program is stopped and the named program brought in. Because there is no logic path to any statements following the CHAIN statement, all needed current program statements must be executed before the CHAIN statement.

DATA FILES

An external data file is a saved program in which you can record information for later use. There are two types of data files: BCD and binary. The data in a file must all be of one type.

BCD data files may be created with a BASIC program or they may be typed in as a saved program. Thus all of the editing commands, such as LIST, EDIT DELETE, and so on, may be used in accessing and modifying BCD files.

Binary data files can be created only with a BASIC program. They cannot be created from a terminal or altered in the same manner as BCD files.

If a BCD or binary file is to be written during program execution, an area on the disc large enough to contain the entire file to be generated must be reserved in your library before program execution. This is done by renaming and saving special files available in the system library (see "Dummy Catalog Files"). These files are empty. Their purpose is to preset the size of the files to be generated.

Example:

```
ØLD
ØLD FILE NAME--CH0768***
READY
RENAME
NEW FILE NAME--WFILE
READY
SAVE
READY
```

In the example, the data file WFILE has been created, it has a storage capacity of 768 characters.

File Reference

The form of the file reference statement is

FILES name 1; name 2; ...; name n

where name 1, name 2, and so on are the names of files to be read or written by the program. *The file reference statements must precede all executable statements in the program.* The

number of files that can be used in a program depends on the size of the program, but you can always reference at least 8 files in any program.

The named files must be saved in your catalog before running the program. Files referenced but not saved produce the error message FILE NØT SAVED when you run the program.

File naming must conform to the established conventions for naming programs, except:

- File names must not contain semicolons. They will be interpreted as file name separators.
- Leading and imbedded blanks are ignored.
- The file name is left-justified.
- File names should not contain slashes or commas.

Examples

```
100 FILES A,B;C
```

or

```
100 FILES A
110 FILES B,C
```

The following file names are identical.

```
100 FILES XYZ
100 FILES XY Z
100 FILES X Y Z
```

File Designator

The file designator is used in all statements referencing files. It singles out a file named in the file reference statement. It may be an integer, an expression, a variable, or a subscripted variable.

Example:

```
100 FILES P;Q
     :
170 READ #1,A(I),B(I)
     :
195 READ #A*B,A(I),B(I)
```

Statement 170 refers to file P. Statement 195 refers to a file depending on the value of the expression A*B, which must be an integer. If the value of A*B is 1, file P is selected; if the value of A*B is 2, file Q is selected.

If the value of the file designator is less than one, non-integral, or greater than the number of files referenced, the error message ILLEGAL FILE DESIGNATØR will be printed.

Dummy Catalog Files

Since it is necessary to save a file of a size large enough to contain the data to be written, the system library contains the following six files which you can rename and save in your catalog.

Library Name	Characters	Library Name	Characters
CH0192***	192	CH1536***	1536
CH0384***	384	CH3072***	3072
CH0768***	768	CH6144***	6144

The number of data points that can be written into a file is a function of the number of characters in each data point. Line numbers and spaces must also be considered in the count.

BCD DATA FILES

BCD files are sequential access files. For each execution of the READ or WRITE statement, data is transmitted serially. BCD files contain line numbers and are listable.

If a BCD file with initial values is to be read from the disc you must prepare it before program execution and save it in your catalog.

Example:

```
NEW
NEW FILE NAME--RFILE
READY
100 1,1.5,2,2.5,3
110 3.5,4,4.5,5,5.5
SAVE
READY
```

When preparing a BCD file, you may if you wish use a blank in place of a comma as a data separator. This option allows files prepared in FØRTRAN to be processed in BASIC and conversely. RFILE in the previous example can be prepared as

```
100 1 1.5 2 2.5 3
110 3.5 4 4.5 5 5.5
```

Note that the word DATA is not needed in these files. The first number on each line is the line number.

All files are in either read mode or write mode. Initially, the FILES statement results in all files being set to the read mode. This protects you from accidentally destroying valuable files. You can later change the mode of any file by issuing a SCRATCH or RESTØRE statement. The SCRATCH statement establishes the designated file as write mode. The RESTØRE statement establishes the designated file as read mode.

File Reading

The form of the BCD read file statement is

READ #file designator, input list

where the file designator is as previously described. The pound sign denotes a BCD file.

The input list consists of the variables, separated by commas, into which the data is to be read. The list may contain non-string and string variables, and any of them may be subscripted.

Example

A file containing 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5 is to be read into A(I) and B(I).

```
100 FILES RFILE
110 FØR I=1 TØ 5
120 READ #1,A(I),B(I)
130 NEXT I
```

For each execution of the READ # statement one value is read into an A(I) and B(I) so that, at the termination of the loop

A(1) = 1	B(1) = 1.5
A(2) = 2	B(2) = 2.5
A(3) = 3	B(3) = 3.5
A(4) = 4	B(4) = 4.5
A(5) = 5	B(5) = 5.5

The file pointer will remain positioned following the last read data item (5.5 in the example) until further file statements designating the file (RFILE in the example) are executed.

File Writing

The form of the BCD write file statement is

WRITE #file designator, output list

where the file designator is as previously described.

The output list consists of the variables, separated by semicolons, from which the file is generated. The list may contain non-string variables, string variables, strings, and expressions. Subscripting is permissible in the output list.

WRITE # always generates a file beginning with line number 1000, incrementing by 10 for each new line. The line number sequencing can be modified, if you wish, by EDIT RESEQUENCE.

Each WRITE # statement generates one line of output unless the teletypewriter line limit is exceeded or the last list item is followed by a semicolon When the teletypewriter line limit is exceeded, writing continues on the next line with the next data item. When the output list is followed by a semicolon, subsequent writing occurs on the same line in a closely packed format.

Example·

```
100 FILES WFILE
110 SCRATCH #1
120 FØR I=1 TØ 25
130 WRITE #1,I,I*I
140 NEXT I
```

When listed WFILE would contain

```
1000   1    1
1010   2    4
1020   3    9
  .    .    .
  .    .    .
  .    .    .
1240   25   625
```

Following is an example showing how strings and string variables are used with files.

Example:

```
100 FILES STRING
110 SCRATCH #1
120 LET A$="STRING1"
130 WRITE #1,A$,"STRING2"
140 WRITE #1,"STRING2";A$
```

Then listing the file STRING gives

```
1000 STRING1  STRING2
1010 STRING2  STRING1
```

End-of-File and End-of-Space

The end-of-file (EOF) is a special mark written by BASIC itself that indicates the end of data in the file.

The end-of-space (EOS) is the physical end of the disc area reserved for a file. When a file is completely filled with data, EOF = EOS; otherwise EOF< EOS.

Whenever a word is generated with a WRITE # statement, an EOF mark is placed immediately following the last word written. Subsequent transmissions to the file move the EOF mark so that it always follows the last word written. When a file is generated from the teletypewriter, an EOF mark is placed immediately after the last data item in the file as soon as the file is saved.

End-of-File Test

The form of the statement that tests for an EOF mark or the EOS is

IF END #file designator THEN line number

where the file designator is as previously described.

The statement will test whether an EOF was detected by the last command reading the designated file. When writing a file, the statement will test whether an EOS was detected.

If the last READ # statement found an EOF mark, the program will go to the line number specified in the IF END # statement. Otherwise the next sequential statement is executed.

If the program continues reading or writing a file after the EOF or EOS has been detected, an error message (END ØF FILE or END ØF SPACE) will be printed and the program will continue. The error message will be printed out each time an attempt is made to exceed the EOF or EOS limit.

Example:

```
100 FILES F1;F2
110 SCRATCH #2
120 DIM X(100),Y(100),Z(100)
130 FØR I=1 TØ 100
140 IF END #1 THEN 180
150 READ #1,X(I),Y(I),Z(I)
160 WRITE #2,SQR(X(I)↑2+Y(I)↑2+Z(I)↑2)
170 NEXT I
180 STØP
```

If file F1 contained 300 data items, no EOF would be encountered; but if it contained only 150 data items, for example, the IF END # statement (line 140) would cause a transfer to line 180 following the 50th execution of the loop.

File Restoring

The form of the BCD restore file statement is

RESTØRE #file designator

where the file designator is as previously described.

The RESTØRE # statement causes the position of the designated file pointer to be moved so that the next transmission is from the beginning of the file.

If a file is already restored, the RESTØRE # statement merely sets the file to read mode.

BASIC automatically restores the referenced files before a program begins executing.

Example:

```
100 FILES UTIL
110 SCRATCH #1
120 FØR I=1 TØ 50
130 WRITE #1,I;SQR(I)
140 NEXT I
150 RESTØRE #1
      .
      .
      .
170 READ #1,X,X1
```

In the example the RESTØRE # statement (line 150) is required to restore the written file before reading it. Reading then begins at the first data item in the file.

A file being written cannot be read before it is restored. A file being read cannot be written before it is scratched. This means that if any modifications are to be made to an existing file via the program, it must be copied (written) to another file up to the modification point. The modification can then be written into the second file.

File Scratching

The form of the BCD scratch file statement is

SCRATCH #file designator

where the file designator is as previously described. This statement causes the designated file to be scratched, or made ready for writing. If the file is already reset or restored the statement merely sets the file to write mode. Following the SCRATCH # statement, transmission to the file commences at the beginning of the file.

Example.

```
100 FILES UTIL
110 SCRATCH #1
120 FØR I=1 TØ 50
130 WRITE #1,RND(0)
140 NEXT I
150 RESTØRE #1
160 READ #1,X,Y,Z
```

In the example, the SCRATCH # statement (line 110) must be executed before the write into the file. The RESTØRE # statement must be executed before reading from the written file.

File Backspacing

The form of the BCD backspace file statement is

BACKSPACE #file designator

where the file designator is as previously described. This statement is permitted only on files in the read mode.

The BACKSPACE # statement causes the position of the data pointer for the file being read to be moved backward one data item. If the data pointer is already at the beginning of the file, the backspace statement is ignored.

Some applications require a file to be processed forward and then backward. The following example illustrates how this can be done.

Example:

```
100 FILES F1
110 IF END #1 THEN 210
120 READ #1,A
130 REM CØUNT NUMBER ØF PØINTS IN FILE
140 LET N=N+1
      .
      .
      .
200 GØ TØ 110
210 FØR I=1 TØ N
220 BACKSPACE #1
230 READ #1,A
240 BACKSPACE #1
      .
      .
      .
300 NEXT I
```

BINARY DATA FILES

Binary data files are not listable. The file space normally used for line numbers in files can be used for data. They are random access files; that is, data can be written into or read from a binary file in any order. Since data conversion routines are eliminated in processing binary files, program efficiency is increased.

For binary files, the restrictions on file mode are eliminated. Binary files may be read or written without having been previously restored or scratched. The RESTØRE and SCRATCH statements serve only to move the data element pointer to the first element in the file.

File Writing

The form of the binary write file statement is

WRITE : file designator, output list

where the file designator is as previously described. The colon denotes a binary file.

The output list may contain numeric variables, string variables, or literals. The numeric variable or literal will cause a two-word entry into the file. The string variable or literal will cause a six-word entry into the file. Each two-word entry is defined as a data element. Consequently each numeric variable or numeric literal consists of one data element, and each string variable or string literal consists of three data elements.

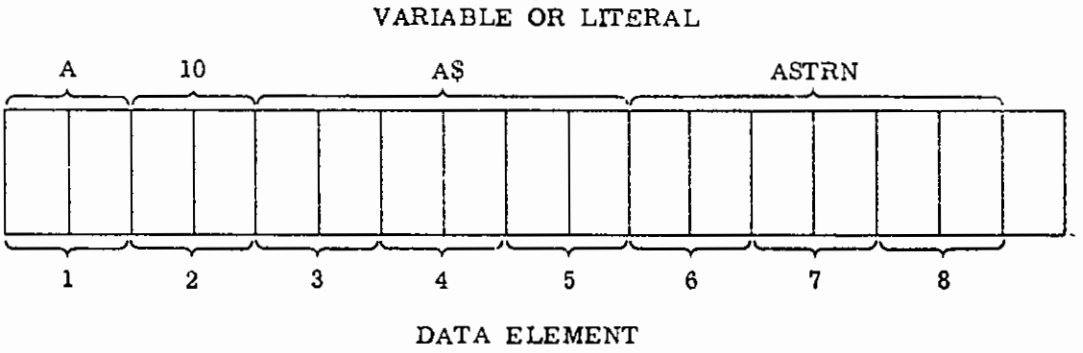
For example, a numeric variable or literal may appear in a WRITE : statement as:

```
100 WRITE:1,A          (numeric variable)
110 WRITE 1,10         (numeric literal)
```

A string variable or literal may appear in a WRITE : statement as:

```
120 WRITE:1,A$         (string variable)
130 WRITE:1,"ASTRN"    (string literal)
```

The following diagram illustrates how each of these will be entered into the file. In the diagram, each block represents one computer word.



Binary files contain no line numbers. Data may be written into the file sequentially or randomly.

The following is an example of a sequentially created file.

Example.

```
100 FILES WFILE
110 SCRATCH 1
120 FØR I=1 TØ 25
130 WRITE.1,I,I*I
140 NEXT I
```

File WFILE contain 25 pairs of numbers, or 50 successive data elements.

In the following example, string information is written into a binary file.

Example·

```
100 FILES STRING
110 SCRATCH 1
120 LET M$="MSTRING"
130 WRITE 1,M$,"NSTRING"
```

File STRING will contain two strings or six data elements, three data elements for each string.

File Reading

The form of the binary read file statement is

READ : file designator, input list

where the file designator is as previously described.

The input list may contain numeric variables or string variables. The numeric variable will cause one data element (two words) to be read from the file. The string variable will cause three data elements (six words) to be read from the file.

Data may be read sequentially or randomly from a file.

In the following example data is read sequentially.

Example:

```
100 FILES RFILE
110 FØR I=1 TØ 5
120 READ:1,A(I),B(I)
130 NEXT I
```

The first ten data elements in file RFILE will be read alternately into arrays A and B.

Random Accessing

The form of the random-access statement is

SET : file designator, variable

where the file designator is as previously described.

The SET statement will position the pointer of the designated file to the element specified by the variable. The variable may be an integer or an expression. If the value of the variable is less than one, non-integral, or greater than the length of the file, the error message ILLEGAL PØINTER will be printed out.

Example

```
100 FILES VFIL
110 SCRATCH·1
120 FØR I=1 TØ 25
130 WRITE·1,I*I
140 NEXT I
150 SET.1,7
160 READ:1,X,Y
```

File VFIL will contain 25 elements. In line 150, the pointer will be positioned to the 7th element. Reading then begins at the 7th element, and the values 49 and 64 will be read into X and Y.

If the file being processed contains string variables, the length of a string entry must be considered before the SET : statement is used.

Example.

```
100 FILES RNF
110 SCRATCH.1
120 WRITE 1,"ASTRN","BSTRN"
130 SET 1,4
140 READ.1,B$
```

When the WRITE statement is executed, ABIRN will occupy the first three entries of file RNF, and BSTRN will occupy the next three entries. To access the second string, the pointer must be set to the position where the second string begins, which is done by line 130. The string data, BSTRN, is then read into the string variable, B$.

The following uses of the SET statement are acceptable.

```
180 SET.1,A*B
195 SET·3,A(I)
```

In each case the value of the variable must be an integer within the limits of the file.

Locating the Element Pointer

Since the element pointer can be moved randomly in a binary file, it is useful to have a way of finding out where the pointer is at any time during the execution of a program.

The LØC function will provide the location of the pointer in the file referenced.

LØC(file designator)

In this statement, X represents any numeric variable, and the file designator is as previously described.

Example

```
100 FILES F1;F2
      .
      .
210 LET A1=LØC(1)
      .
      .
      .
290 PRINT LØC(2)
```

The value of the element pointer into file F1 will be stored in A1 (line 210).

The value of the element pointer into file F2 will be printed (line 290).

File Scratching

The form of the binary scratch file statement is

SCRATCH : file designator

where the file designator is as previously described.

The SCRATCH statement cause the data element pointer to be repositioned so that transmission to the file starts at the beginning of the file.

Example

```
100 FILES VFIL
115 SCRATCH 1
125 FØR I=1 TØ 20
140 WRITE 1,INT(10*RND(X))
155 NEXT I
```

The SCRATCH . statement may be replaced with a SET . statement to accomplish the same purpose:

```
115 SET:1,1
```

A file being read can be written without being scratched. This means that files can be modified during program execution.

Example:

```
100 FILES RFIL
110 LET Z1=0
120 LET S1=26
130 SET:1,S1
140 READ:1,X
150 IF X=0 THEN 180
160 SET:1,LØC(1)-1
170 WRITE.1,Z1
180 LET S1=S1+26
190 IF S1<EØF(1) THEN 130
200 END
```

This example reads every 26th data element and goes back and writes over that element with a zero, provided that the element is not already zero.

File Restoring

The form of the binary restore file statement is

RESTØRE : file designator

where the file designator is as previously described.

The RESTØRE . statement will move the element pointer to the beginning of the designated file.

Example:

```
100 FILES VFIL
110 SCRATCH:1
120 FØR J=1 TØ 50
130 WRITE:1,J,SQR(J)
140 NEXT J
150 RESTORE:1
      :
      :
180 READ:1,X
```

This program writes 100 data elements into VFIL. Then the element pointer is moved to the beginning of the file (line 150), and the first data element is read into X.

A file being written can be read before it is restored. If line 150 in the last example is omitted, reading will begin where writing stopped, and the 101st data element will be read into X.

End-of-File Test

For binary files the end-of-file test is a test for the end of file space. The IF END statement will test whether an end-of-file-space condition was detected by the last READ: or WRITE statement. When reading a partially filled binary file, the READ statement should be executed

only the number of times required to read the legitimate data in the file. The IF END test would allow such a file to be read until all the space in the file had been exhausted.

The form of the statement is

IF END file designator THEN line number

If the last READ statement or WRITE statement encountered the end of file space, the program will go to the line number specified in the IF END statement. Otherwise the next sequential statement is executed.

Example

```
100 FILES F1
110 SCRATCH 1
120 FØR J=1 TØ 1000
130 IF END:1 THEN 160
140 WRITE·1,J/2
150 NEXT J
160 END
```

An intrinsic function is also provided to test for the end-of-file condition. The function LØF will retrieve the length of the referenced file. For example:

LET X = LØF (file designator)

In this statement, X represents any numeric variable, and the file designator is as previously described.

The functions LØC and LØF can replace the IF END : statement in the previous sample program as follows

```
130 IF LØC(1)=LØF(1) THEN 160
```

When the element pointer into file F1 reaches the end of the file, writing stops.

If a program continues reading or writing a file after the end-of-file-space condition has been detected, the error message END ØF FILE SPACE will be printed, and the program will continue executing.

MATRICES

The matrix operation statements available in BASIC are among the most powerful and useful in the entire language.

Following is a list of matrix statements.

MAT READ A,B,C,	Read matrices A, B, and C, their dimensions having been previously specified. Data is read in row-wise sequence.
MAT PRINT A,B,C	Print matrices A, B, and C, with A and C in the regular format, but B closely packed.
MAT C=A+B	Add matrices A and B and store the result in matrix C.
MAT C=A-B	Subtract matrix B from matrix A and store the result in matrix C.
MAT C=A*B	Multiply matrix A by matrix B and store the result in matrix C.
MAT C=INV(A)	Invert matrix A and store the result in matrix C.

MAT C=TRN(A)	Transpose matrix A and store the result in matrix C.
MAT C=(K)*A	Multiply matrix A by the value represented by K. K may be either a number or an expression, but in either case it must be enclosed in parentheses.
MAT C=CØN	Set each element of matrix C to one. CØN means constant.
MAT C=ZER	Set each element of matrix C to zero.
MAT C=IDN	Set the diagonal elements of matrix C to one's and the non-diagonal elements to zeroes, yielding an identity matrix

MAT READ and MAT PRINT

Using the MAT READ and MAT PRINT statements, you can read data into or print data from a matrix without having to reference each element of the matrix individually.

Examples.

```
100 MAT [illegible], C
150 MAT [illegible] C
175 MAT READ Z
190 MAT PRINT A,
```

Data is read into a matrix using the DATA statement. The elements in the DATA statement are taken in row order, that is,

$A_{1,1}, A_{1,2}, \ldots, A_{1,m}, A_{2,1}, A_{2,2}, \ldots, A_{2,m}, \ldots, A_{n,m}$.

Information is read from DATA statements until the matrix array is completely filled. Partial matrices cannot be read or printed.

Example:

```
110 DIM L(2,3), M(2,2)
150 MAT READ L,M
160 LET L(2,2)=-2*L(2,2)
200 MAT PRINT L; M
500 DATA 1, 2, 3, 4, 5, 6, 8, -12, 0, 7
```

The DIM statement defines L as a 2 by 3 matrix and M as a 2 by 2 matrix. The MAT READ statement reads in row order from the DATA statement at line 500. The matrix element $L_{2,2}$ is recomputed at line 160. The two matrices are then printed to yield.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & -10 & 6 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 3 & -12 \\ 0 & 7 \end{bmatrix}$$

Matrix Addition, Subtraction, and Multiplication

You can add, subtract, and multiply matrices using the matrix arithmetic statements. The matrix dimensions must be conformable for each operation. If dimensions are not conformable, execution is stopped and you receive a dimension error message.

Matrix arithmetic statements may take the forms

MAT C=A+B MAT C=A-B MAT P=Q*R

Only one operation can be done in each statement.

Example:

Calculate [H] a,a=[E] a,a-[K] a,a +[A] a,3*[B] 3,a

```
612 MAT H=A*B
615 MAT H=H+E
618 MAT H=H-K
```

Scalar Multiplication

You can multiply a matrix by a scalar expression using a statement of the form

MAT X=(expression)*D

where X and D are matrices and the expression in parentheses is a scalar quantity. The parentheses are required to indicate scalar rather than matrix multiplication. Only one operation per statement is allowed.

Examples

```
100 MAT F=(2)*G
150 MAT Q=(2.33+M)*Q
750 MAT B=(N)*A
```

Identity Matrix

An identity matrix is defined by a statement of the form

MAT B=IDN or MAT R=IDN(expression, expression)

In the first statement, matrix B is set up as an identity matrix. If B is not defined to be square, you will receive a dimension error message. In the second statement, the size of the identity matrix R is determined at execution time by the value of the expression enclosed in parentheses

Examples:

```
190 MAT A=IDN
100 MAT V=IDN(2*N+1,2*N+1)
120 MAT B=IDN(Q,Q)
130 MAT W=IDN
140 MAT C=IDN(1,1)
```

Matrix Transposition

Matrices are transposed using the form

MAT Y=TRN(Z)

where Y and Z are both matrices. The matrix Z transpose will replace matrix Y. Y and Z must conform Matrix transposition in place (MAT A=TRN(A)) is not allowed.

Examples.

```
300 MAT G=TRN(H)
400 MAT U=TRN(V)
```

Matrix Inversion

Matrices are inverted using the form

```
MAT I=INV(J)
```

where I and J are both matrices. I will contain the matrix J inverse. I and J must conform. Matrix inversion in place (MAT A=INV(A)) is not allowed. If a matrix is singular, you will receive the message NON-UNIQUE SINGULAR MATRIX.

Example:

```
100 MAT K=INV(L)
200 MAT A=INV(B)
```

Matrix ZER and CON Functions

The ZER function is used to zero out all elements of a matrix. It may also be used to redefine the dimensions of a matrix as described in "Dimensioning." As an example:

```
MAT C=ZER
```

will zero out the elements of matrix C.

The CON function is used to set all elements of a matrix to one's. As an example

```
MAT C=CON
```

will set all elements of matrix C to one's.

Dimensioning

Every matrix variable used in a program must be given a single-letter name.

A matrix variable must be defined in a DIM statement, which sets aside the amount of storage required by the matrix variable during execution of the program. For example:

```
DIM P(3,4),Q(5,5)
```

The DIM statement defines two matrices, P and Q. P is defined as a 12 element matrix, and Q as a 25 element matrix. Note that the first element of P is P(1,1) and the last element P(3,4). The elements of Q run from Q(1,1) through Q(5,5). All matrix variables must be doubly dimensioned, as shown here.

Before any computation using the MAT statements, you must declare the precise dimensions of all matrices to be used in the computation. Four of the MAT statements are used for this purpose.

```
MAT READ C(M,N)
MAT C=ZER(M,N)
MAT C=CON(M,N)
MAT C=IDN(N,N)
```

The first three statements specify matrix C as consisting of M rows and N columns. The fourth statement specifies matrix C as a square matrix of N rows and N columns.

These same statements may be used to redimension a matrix during running. A matrix may be redimensioned to either a larger or a smaller matrix, provided the new dimensions do not require more storage space than was originally reserved by the DIM statement. To illustrate, consider the following.

Example

```
110 DIM A(8,8),B(8 8),C(8,8)
150 MAT READ A(2,2),B(2,2)
160 MAT C=ZER(2,2)

200 MAT A=IDN(3,8)
210 MAT READ B(4,1),C(4,4)
```

Note that the DIM statement reserves enough storage to accommodate three matrices, each consisting of 64 elements. The initial MAT READ specifies the dimensions of both matrices A and B as 2 rows and 2 columns.

The MAT READ also reads the number of values required by the dimensions into the storage that was reserved by the DIM statement. It reads them in row-wise sequence. In the initial MAT READ, the elements in the order read are A(1,1), A(1,2), A(2,1), A(2,2), B(1,1), B(1,2), B(2,1), and B(2,2). Statement 160 uses the ZER to specify dimensions and to zero the elements of matrix C. Statements 200 and 210 illustrate redimensioning. Matrix A is redimensioned as an 8 row, 8 column identity matrix, and matrices B and C are redimensioned as 4 row, 4 column matrices into which data is to be read.

The combination of ordinary BASIC statements and MAT statements makes BASIC very powerful, but you must be careful about dimensions. In addition to having both a DIM statement and a declaration of current dimension, you should watch your use of the MAT statements. For example, a matrix product MAT C=A*B may be illegal for either of two reasons: A and B may have such dimensions that the product is not defined, or C may have the wrong dimensions for the answer. In either case you will receive the DIMENSIØN ERRØR message.

Examples

Two programs follow that illustrate some of the capabilities of the MAT statements. In the first program, the values for M and N are read. Using these two values as indices, statement 120 sets the dimensions for matrices A, B, D, and G. The values for the elements of these four matrices are read, Then, in sequence.

- The dimensions of matrix C are specified and the elements set to zero (line 130).
- Matrix A is printed (line 150).
- Matrix B is printed (line 170)
- The sum of matrices A and B is found and stored in C (line 180).
- Matrix C is printed (line 200).
- The dimensions for matrix F, a vector, are set and the elements set to zero (line 210)
- The product of matrices C and D is computed and stored in F (line 220).
- The dimensions for matrix H (single value) are specified and the elements set to zero (line 230).
- Finally, the product of matrices G and F is found and stored in H and printed (lines 240, 260).

In the second program, a value N is read that determines the order of the Hilbert matrix segment to be computed, stored, and printed. Next the matrix is inverted and printed. Finally the Hilbert matrix is multiplied by its own inverse, and the resulting product matrix is printed. Notice that line 290 specifies N as equal to 2 to produce the first three matrices of order 2, and later returns to read in the data "3," redimensions to a larger array--larger than 2, but smaller than the original 20--and produces more output.

```
100 DIM A(5,5),B(5,5),C(5,5),D(5,5),E(5,5),F(5,5),G(5,5),H(5,5)
110 READ M,N
120 MAT READ A(M,M),B(M,M),D(M,N),E(N,N)
130 MAT C=ZER(M,M)
140 PRINT "MATRIX A OF ORDER";M
150 MAT PRINT A
160 PRINT "MATRIX B OF ORDER";M
[illegible]
260 MAT PRINT [illegible]
[illegible]
... DATA 1,2,3,4,5,6,7,8,9,9,8,7,6,5,4,3,2,1,1,2,3,3,2,1
... END
RUN
```

```
[illegible]        15:40

MATRIX A OF ORDER 3
 1   2   3

 4   5   6

 7   8   9

MATRIX B OF ORDER 3
 9   8   7

 6   5   4

 3   2   1

         C=A+B
 10       10       10

 10       10       10

 10       10       10

 360
```

MATRIX PROGRAM EXAMPLE 2.

```
100 DIM A(20,20),B(20,20),C(20,20)
110 READ N
120 MAT A=CON(N,N)
130 MAT B=CON(N,N)
140 MAT C=ZER(N,N)
150 FOR I=1 TO N
160 FOR J=1 TO N
170 LET A(I,J)=1/(I+J-1)
180 NEXT J
190 NEXT I
200 PRINT "HILBERT MATRIX OF ORDER";N
210 MAT PRINT A;
220 MAT B=INV(A)
230 PRINT "INVERSE OF HILBERT MATRIX OF ORDER";N
240 MAT PRINT B;
250 MAT C=A*B
260 PRINT "HILBERT MATRIX TIMES ITS OWN INVERSE ORDER";N
270 MAT PRINT C;
280 GO TO 110
290 DATA 2,3
999 END
RUN

MAT-2      15:46

HILBERT MATRIX OF ORDER 2
 1   .5

 .5     .333333


INVERSE OF HILBERT MATRIX OF ORDER 2
 4.    -6.

-6.     12.


HILBERT MATRIX TIMES ITS OWN INVERSE ORDER 2
 1.     0

-3.72529E-09     1.


HILBERT MATRIX OF ORDER 3
 1   .5     .333333

 .5     .333333   .25

 .333333   .25    .2


INVERSE OF HILBERT MATRIX OF ORDER 3
 9.    -36.    30.

-36.    192.  -180.

 30.   -180.   180.


HILBERT MATRIX TIMES ITS OWN INVERSE ORDER 3
 1.    -1.78814E-07     0

-2.72717E-08     1.    -5.96046E-08

-1.49012E-08    -1.78814E-07     1

OUT OF DATA  IN 110
```

EXAMPLES OF ADVANCED BASIC PROGRAMS

Following are two sample programs illustrating the use of many of the advanced capabilities of BASIC. The first program is developed in an inventory case problem, and makes use of a SCD file. The second program uses a binary file to store personnel information.

Inventory Problem

Mr. Swift, a storekeeper, would like to know how any five items in his store are selling in any given month. He would like a permanent file of the items that were sold each week over a four week period. He wants to update his file at the end of each week, and he may or may not want to get a complete written record of his sales. He may want to get a written report at any time during the month.

The record should consist of an easy to read table listing the items and the number sold in each week. The table should also show the total number of items for each week, the total number of each item sold to date, and the total number of all items sold to date.

The five items that Mr. Swift would like to check are salt, pepper, sugar, nutmeg, and coffee. The month is March.

The following program results from Mr. Swift's requirements. The program is explained by remarks included in it.

```
100 FILES STOCK
110 DIM A(4,3)
120 FOR I=0 TO 5
130 READ A$(I)
140 NEXT I
150
160 REM  W=INITIAL PASS FLAG, P=PRINTOUT FLAG
170 REM  W=0 INITIAL PASS,    P=0 PRINTOUT DESIRED
180 READ W,P
190 IF W<0 THEN 300
200
210 REM FOR THE INITIAL PASS, WRITE THREE ZEROES.  THERE IS NO
220 REM DATA INITIALLY, AND SOMETHING MUST BE WRITTEN BEFORE
230 REM IT CAN BE READ.
240 SCRATCH#1
250 WRITE#1,0,0,0
260 RESTORE#1
270
280 REM READ IN DATA WRITTEN INTO THE FILE FROM PREVIOUS WEEKS.
290 REM X...ITEM, Y...WEEK, A(X,Y)...NUMBERS OF ITEMS SOLD TO DATE.
300 READ#1,X,Y,A(X,Y)
310 IF END#1 THEN 350
320 GO TO 300
330
340 REM READ DATA FOR THIS WEEK OR DATA MISSED FROM PREVIOUS WEEKS.
350 READ X,Y,Z
360 IF X<0 THEN 430
370
380 REM WHEN X IS NEGATIVE, THE DATA READ IS FINISHED.
390 LET A(X,Y)=A(X,Y)+Z
400 GO TO 350
410
420 REM WRITE THE UPDATED INFO TO THE PERMANENT DATA FILE.
430 SCRATCH#1
440 FOR X=0 TO 4
450 FOR Y=0 TO 3
460 WRITE#1,X,Y,A(X,Y)
470 NEXT Y
480 NEXT X
490
500 REM IS A PRINTOUT WANTED?  0...YES   -1...NO
510 IF P<0 THEN 570
```

```
520
530 REM PRINT THE MONTH.
540 PRINT A$(5)
550 PRINT
560 PRINT TAB(10);
570 REM PRINT THE COLUMN NUMBER FOR EACH WEEK.
580 FOR I=0 TO 3
590 PRINT USING 610,I+1,
600 NEXT I
610:    ####
620 PRINT USING 630
630:    TOTALS
640 PRINT
650
660 REM BEGIN TO GENERATE THE TABLE OF VALUES.
670 FOR I=0 TO 4
680 PRINT A$(I);TAB(10);
690 FOR J=0 TO 3
700 REM SUM OF EACH ITEM FOR THE ELAPSED WEEKS
710 LET T(I)=T(I)+A(I,J)
720 REM SUMS FOR EACH WEEK
730 LET S(J)=S(J)+A(I,J)
740 PRINT USING 610,A(I,J),
750 NEXT J
760 PRINT USING 610,T(I)
770 NEXT I
780 PRINT
790 PRINT TAB(10);
800
810 REM PRINT SUBTOTALS FOR EACH WEEK AND THE TOTAL FOR THE PERIOD.
820 FOR K=0 TO 3
830 PRINT USING 610,S(K),
840 LET S=S+S(K)
850 NEXT K
860 PRINT USING 610,S
870 END
880 REM 0...SALT   1...PEPPER,  2..SUGAR,  3...NUTMEG,  4...COFFEE
890 DATA SALT,PEPPER,SUGAR,NUTMEG,COFFEE,MARCH
1000 DATA [illegible]
1010 DATA [illegible]
```

Suppose that [illegible] Mr. Swift wants a record of his sales to date. He runs the [illegible] line 1000 as shown above. The -1 specifies not the first week, [illegible] the [illegible] specifies [illegible] the data. Also he enters data in line 1010 as shown above. The -1 indicates the [illegible] of data, and the two final zeroes are dummy data put in to satisfy the READ. [illegible] the following results.

```
SALT        [illegible]

MARCH

                  1        2        3        4     TOTALS

SALT              3        4        0        0        7
PEPPER            7        5        0        0       12
SUGAR             4        3        0        0        7
NUTMEG            8        7        0        0       15
COFFEE            2        9        0        0       11

                 24       28        0        0       52
```

At the end of the third week, Mr Swift wants to make an update. He wants a printout. All he must do is replace line number 1010, as shown below. The BCD data file is updated, and the printout shows the entries for the third week and the resulting changes in the totals.

```
1010 DATA 0,2,7, 1,2,5, 2,2,1, 3,2,5, 4,2,12, -1,0,0
RUN



SALT        14:30

-------MARCH

                    1       2       3       4     TØTALS

SALT                3       4       7       0       14
PEPPER              7       5       5       0       17
SUGAR               4       3       1       0        8
NUTMEG              8       7       5       0       20
CØFFEE              2       9      12       0       23

                   24      26      30       0       82
```

Personnel Information

Following is a typical example of the use of binary files. There are two programs. WØRKER establishes the data base in the file INFØ. WØRK1 shows how values can be altered at will.

```
100 FILES INFØ
110 SCRATCH :1
120 READ A$,S,A1,D,S1,V
130 WRITE:1,A$;S;A1;D;S1;V
140 GØ TØ 120
150 DATAGØRDØN,9345,27,0,4,0
160 DATAPLUMMER,10200,30,4,0,5
170 DATAGARANTINØ,8600,22,0,2,7
180 DATATHØMAS,11550,29,3,0,2
190 DATACHENEY,8800,25,0,4,6

RUN

WØRKER     14:06

ØUT ØF DATA  IN 120



100 FILES INFØ
110 READ A,B,C
120 GØSUB 200
130 SET:1,A*6+B
140 READ:1,X
150 LET X=X+C
160 SET:1,LØC(1)-1
170 WRITE:1,X
180 GØ SUB 240
190 STØP
200 PRINTUSING 220
210 PRINT
220:NAME          SALARY  AGE   DEPENDENTS   SICK DAYS   VACATIØN DAYS
230:"            "$#####  ###          ###        ####            ####
240 SET:1,A*8+1
250 READ:1,A$,S,A1,D,S1,V
260 PRINTUSING 230,A$,S,A1,D,S1,V
270 RETURN
280
```

```
290 REM DATA FØRMAT --- A-NAME,  B-CØDE VALUE,  C-AMØUNT TØ BE ADDED
300 REM  NAMES---- 0-GØRDØN, 1-PLUMMER, 2-GARANTINØ, 3-THØMAS
310 REM            4-CHENEY
320
330 REM CØDE-----  1-NAME, 4-SALARY, 5-AGE, 6-DEPENDENTS, 7-SICK DAYS
340 REM            8-VACATIØN DAYS
350
360 REM  FØR DATA --- 2,4,820 --- ALTER MR. GARANTINØ'S SALARY BY $820
370
380 DATA 2,4,820


RUN


WØRK1      14:11

NAME          SALARY   AGE   DEPENDENTS   SICK DAYS   VACATIØN DAYS

GARANTINØ   $  8600    22         0            2             7
GARANTINØ   $  9420    22         0            2             7
```

The second line of the printout shows the $820 increase in Mr. Garantino's salary.

Appendix A Error Messages

Because most programs under development contain errors, a series of error messages is included in BASIC. Some of the messages are received during compilation and others during execution of a program. Many of the messages not only identify the type of error, but indicate the line number where the error occurred. In the following table, XXX means a line number.

During execution, some messages occur that do not stop execution, but inform you of irregular conditions existing in identified lines of your program. Other messages, however, point out more serious errors that cause execution to stop.

Compilation Errors

MESSAGE	MEANING
CUT PRØGRAM ØR DIMS	Either the program is too long, or the amount of space reserved by the DIM statements is too much, or both. Cut the length of the program, reduce the size of the lists and tables, reduce the length of printed labels, or reduce the number of simple variables.
DIMENSIØN TØØ LARGE IN XXX	The size of a list or table is too large. Make it smaller. Maximum dimension is A(1022).
FILE NØT DEFINED IN XXX	The file reference statement is missing.
FILES NØT FIRST IN XXX	The file reference statement is preceded by an executable statement.
FØR WITHØUT NEXT	A NEXT statement is missing. This message can occur in conjunction with NEXT WITHØUT FØR.
ILLEGAL CØNSTANT IN XXX	A number is out of bounds (> 5.78960E76).
ILLEGAL FØRMULA IN XXX	Perhaps the most common error message. May indicate missing parentheses, illegal variable names, missing multiply signs, illegal numbers, or other errors. Check the statement thoroughly.
ILLEGAL INSTRUCTIØN IN XXX	Other than one of the 26 legal BASIC instructions has been used following the line number.
ILLEGAL NUMBER IN XXX	The line number is of incorrect form or contains more than 5 digits.
ILLEGAL PRØGRAM NAME IN XXX	A program name in a CALL or CHAIN statement is a name with more than 6 characters and is other than a library program name. A program cannot call itself.

MESSAGE	MEANING
ILLEGAL RELATIØN IN XXX	Something is wrong with the relational expression in an IF--THEN statement. Check to see if you have used one of the 6 permissible relational symbols.
ILLEGAL VARIABLE IN XXX	An illegal variable name has been used.
IMAGE TABLE ØVERFLØW	More than 100 image statements have been included in the program.
INCØRRECT FØRMAT IN XXX	The format of the statement is wrong.
NEXT WITHØUT FØR IN XXX	There is an incorrect NEXT statement, perhaps with a wrong variable given. Check also for incorrectly nested FØR statements.
NØ DATA	There is at least one READ statement in the program, but there are no DATA statements.
ØVER 10 SUBRØUTINES	The program tried to call more than 10 different programs.
PRØGRAM NØT SAVED (Program Name)	A program named in a CALL statement is not saved.
PRØGRAM TØØ LØNG IN XXX	The program is too long for the available storage. Cut the size of the program or dimensions.
PRØGRAM WØN'T FIT (Program Name)	The program called is too large to fit in the remaining available space.
REDIMENSIØNED ARRAY IN XXX	An array has previously been dimensioned.
STRING TØØ LØNG IN XXX	A string has more than 15 characters.
TØØ MANY CØNSTANTS IN XXX	A statement contains constants that result in more than 75 different constants in the program. Example: LET A(4) = 1.24, 4 and 1.24 are constants.
TØØ MANY FILES	The number of files referenced caused the program size limit to be exceeded. (At least 8 files will always be allowed.)
TØØ MANY GØTØ'S IN XXX	More than 79 different line number references were made in GØTØ, IF--THEN, GØSUB, or ØN--GØTØ statements.
TØØ MANY LØØPS	There are more than 26 FØR--NEXT combinations in the program.
TØØ MUCH DATA	The program contains more than 1280 numbers, or too many numbers and strings, or too many strings as data.
UNDEFINED FUNCTIØN	A function such as FNF( ) has been used without appearing in a DEF statement. Check for typographical errors.

MESSAGE	MEANING
UNDEFINED IMAGE XXX	A PRINT USING statement references line XXX, but line XXX either does not exist or is not an image statement.
UNDEFINED NUMBER IN XXX	The statement number appearing in a GØTØ, IF--THEN, GØSUB, or ØN--GØTØ statement does not appear as line number in the program.

Execution Errors--Execution Continued

MESSAGE	MEANING
ABSØLUTE VALUE RAISED TØ PØWER IN XXX	A computation of the form (-3)↑2.7 has been attempted. The computer supplies (ABS(-3))↑2.7 and continues. Note: (-3)↑3 is correctly computed to give -27.
ATTEMPT TØ READ A NUMBER AS A STRING VARIABLE	Self explanatory.
DIVISIØN BY ZERØ IN XXX	A division by zero has been attempted. The computer supplies +∞ (about 5.78960E76) and continues running the program.
END ØF FILE IN XXX	An attempt has been made to read data from a BCD file after all data has been read. The file pointer is located at the end of the file. No data is transmitted.
END ØF FILE SPACE IN XXX	Under all physical space in a file has been used, an attempt has been made to write into a BCD file, or an attempt has been made to read from or write into a binary file. No data is transmitted.
INPUT DATA NØT IN CØRRECT FØRMAT, RETYPE IT	Self explanatory.
LØG ØF NEGATIVE NUMBER IN XXX	The program has attempted to calculate the logarithm of a negative number. The computer supplies the logarithm of the absolute value and continues.
LØG ØF ZERØ IN XXX	The program has attempted to calculate the logarithm of zero. The computer supplies -5.78960E76 and continues.
ØVERFLØW IN XXX	A number larger than about 5.78960E76 has been generated. The computer supplies ±5.78960E76 and continues running the program.
SQUARE RØØT ØF A NEGATIVE NUMBER IN XXX	The program has attempted to extract the square root of a negative number. The computer supplies the square root of the absolute value and continues.
UNDERFLØW IN XXX	A number smaller in absolute size than about 4.31809E-78 has been generated. The computer supplies zero and continues. In many circumstances, underflow is permissible and may be ignored.

MESSAGE	MEANING
ZERØ TØ A NEGATIVE POWER IN XXX	A computation of the form 0↑(-1) has been attempted. The computer supplies +∞ (about 5.7E76) and continues.

Execution Errors--Execution Terminated

MESSAGE	MEANING
BAD IMAGE IN XXX	There are syntax errors in the image statement referenced by line number XXX, or an attempt has been made to put numeric data in an alphanumeric field, or alphanumeric data in a numeric field.
CALLS ØR GØSUB NESTED TØØ DEEPLY IN XXX	Too many CALLs or GØSUBs without a RETURN. It may be that subroutines are being left by GØTØ or IF--THEN statements rather than by RETURNs. The program stops.
DATA FILE (LINE XXX) FØRMAT ERRØR	At line XXX of the data file being read, data is not in the required format.
DIMENSION ERROR IN XXX	A dimension inconsistency has occurred in connection with one of the MAT statements. The program stops.
EXPRESSIØN ØUT ØF RANGE	The range of an ØN--GØTØ statement is incorrect. Example: ØN X GØTØ 10,20,30. When the integer value of X is either minus, zero, or greater than 3, the expression is out of range.
FILE NØT BCD	An attempt has been made to do a BCD file read or write on a binary file.
FILE NØT BINARY	An attempt has been made to do a binary file read or write on a BCD file.
FILE(S) NØT SAVED (File Name)	The files indicated have been referenced but are not saved in your library.
ILLEGAL FILE DESIGNATØR IN XXX	The file designator is less than unity, non-integral, or greater than the number of referenced files.
ILLEGAL PØINTER	The element pointer is less than unity, non-integral, or greater than the length of the referenced file.
NEARLY SINGULAR MATRIX IN XXX	The INV operation in MAT has encountered a matrix with zero or nearly zero pivotal elements. The matrix being inverted is singular or nearly so. Note, however, that this check is not 100 percent reliable. For instance, this message need not occur even if the inverse is meaningless, as with high order Hilbert matrices. If this error occurs, the program stops.
ØUT ØF DATA IN XXX	A READ statement for which there is no DATA has been encountered. If this means a normal end of your program, ignore the message. Otherwise, it means that you haven't supplied enough DATA. In either case, the program stops.

MESSAGE	MEANING
READING BCD IN XXX	An attempt has been made to write into a BCD read mode file. Indicates a logic error or no SCRATCH statement encountered before read mode activity.
RETURN BEFØRE GØSUB ØR CALL IN XXX	A RETURN has been encountered before the first GØSUB or CALL in the program. Note BASIC does not require the GØSUB to have an earlier statement number--only to execute a GØSUB before executing a RETURN. The program stops.
SUBSCRIPT ERRØR IN XXX	A subscript has been called for that lies outside the range specified in the DIM statement, or, if no DIM statement applies, outside the range 0 through 10. The program stops.
WRITING BCD IN XXX	An attempt has been made to read or backspace a BCD write mode file. Indicates a logic error or no RESTØRE statement encountered before read mode activity.

Appendix B Limitations on BASIC

There are some limitations imposed on BASIC by the limited amount of computer storage. Listed below are some of these limitations, in particular, those that are [illegible] error messages in Appendix A. The reader should realize that although the [illegible] itself is fixed, in time some of these limitations may be relaxed slightly.

ITEM	LIMITATION
Source program size	The source program may not consist of more than 250 lines. It may not contain more than 6144 characters.
Constants	The total number of different constants must not exceed 75.
Data	There can be no more than 1280 data numbers.
FØR statements	There can be no more than 26 FØR statements in a program.
GØ TØ, IF--THEN, GØSUB, and ØN--GØ TØ statements	The total number of different references in these statements cannot exceed 79.
Compiled program size	Cannot exceed 4148 words.
Image statements	Maximum of 100.
Dimension of array	A singly dimensioned array cannot exceed 1022. The limitations on a doubly dimensioned array with dimensions X, Y, are: (1) X cannot exceed 1022, (2) Y cannot exceed 510, and (3) the product of X+1 and Y+1 cannot exceed 2074.
Naming of variables	The variable A is distinct from the element A(0).
Subscripting	Numeric variable names consisting of two characters may not be subscripted.

Appendix C Comparison Order for BASIC Characters

BASIC characters are compared in their BASIC code representations. The following table gives the BASIC code number for each character. Codes of nonprinting characters are enclosed in parentheses.

Code	Character	Code	Character	Code	Character
00	0	24	D	53	$
01	1	25	E	54	*
				(55)	End of Message
02	2	26	F	56	>
03	3	27	G	57	?
04	4	30	H	60	(space)
05	5	31	I	61	/
		(32)	Bell		
06	6	33	. (period)	62	S
07	7	34	" (quote)	63	T
10	8	35	?	64	U
11	9	36	<	65	V
		(37)	Carriage Ret.		
12	' (apostrophe)	40	- (minus)	66	W
13	:	41	J	67	X
14	(	42	K	70	Y
15	,	43	L	71	Z
				(72)	Line Feed
16	=	44	M	73	, (comma)
17	\	45	N	74	)
20	+	46	O	75	[
21	A	47	P	76	]
22	B	50	Q	(77)	Fill
23	C	51	R		
		(52)	Tab		

Appendix D Using the Time-Sharing System

The Mark I Time-Sharing System consists of a GE-235 computer with a number of input-output stations, currently Model 33 and Model 35 Teletypes. Those using the input-output stations are able to share the use of the computer with each other so as to suggest that each one has sole use of the computer. The teletypewriters are the devices through which the user communicates with the computer. This appendix contains elementary instructions for using the Time-Sharing System. For complete information, see the Mark I Time-Sharing Service Command System Reference Manual (229116).

The Keyboard

The teletypewriter keyboard is a standard typewriter keyboard for the most part. There are three special keys the user must be familiar with.

RETURN — The RETURN key is located at the right-hand end of the third row of keys, and does more than act as an ordinary carriage return. The computer ignores the line being typed until this key is pushed.

CTRL — The CTRL (control) key is located at the left-hand end of the third row of keys. When it is pressed along with the X key, the computer deletes the entire line being typed. This also acts as a carriage return.

The backwards arrow key is the shift of Ø. It is used to delete the character or space immediately preceding the ←. If this key is pressed N times, the N preceding characters or spaces will be deleted.

Examples.

ABCWT←←DE appears as ABCDE when RETURN is pushed.

AB C←←CDE appears as ABCDE when RETURN is pushed.

Some languages available on the Time-Sharing System use the three characters \, [, and]. They are located on the keys L, K, and M, respectively, when either SHIFT key is pushed.

Teletypewriter Operation

Besides the keyboard itself there are four control buttons necessary to operate the machine.

Button	Location	Function
ØRIG	Leftmost of six small buttons on the right.	Turns on the teletypewriter and connects it to the phone line.
CLR	Next to ØRIG.	Turns off the teletypewriter and disconnects the phone circuit.
LØC LF	Left of the space bar on Model 35 Teletypes only.	Feeds paper to permit tearing it off.

BUZ-RLS	Rightmost of six small buttons on the right.	Turns off the buzzer that signals a low paper supply.

If the teletypewriter is on a direct line to the computer, pushing the ØRIG button is all that is necessary to connect up with the computer. To disconnect from the computer, type GØØDBYE, or BYE. If that fails, push CLR.

In order to connect with the computer from a teletypewriter not on a direct line:

- Push the ØRIG button and wait for the dial tone.
- Dial one of the numbers at the Time-Sharing Center.

In order to disconnect, type GØØDBYE or BYE, and if that fails, push CLR.

Control Commands

There are a number of commands that may be given to the computer by typing the command at the start of a new line, with no line number, and following the command with a carriage return. The following table lists some of the most frequently used of these commands.

Command	Meaning
CATALØG	The computer types a list of the names of all the programs currently saved under that user number.
EDIT	Gives a brief explanation of the format used in the EDIT commands.
LENGTH	Gives you an idea of the length of the program, to the nearest 200 characters. The maximum length of one program is 6400 characters.
LIST	Causes an up-to-date listing of the program to be typed out.
LIST--XXXXX	Causes an up-to-date listing of the program to be typed out beginning at line number XXXXX and continuing to the end.
NEW	Erases from working storage the program currently being worked on, and asks for a NEW FILE NAME.
ØLD	Erases from working storage the program currently being worked on, and asks for an ØLD FILE NAME.
RENAME	Permits you to change the name of the program you are currently working on, but does not destroy the program.
RUN	Begins the computation of a program.
RUN (typed during computation)	Gives an indication that a program is running and how much machine time has elapsed since the run began.
SAVE	Saves the program intact for later use. To retrieve a saved program, type ØLD.
SCRATCH	Destroys the program currently being worked on, but leaves the user number and program name intact. It gives you a clean sheet to work on.
STATUS	Gives an indication of the status of the teletypewriter you are using (running, idle, or disconnected).
STØP	Stops the computation at once. It can be typed only when the teletypewriter is not printing.

Command	Meaning
SYSTEM	Permits you to change systems (BASIC, ALGØL, etc.) without going through the sign-on sequence again.
TTY	Supplies the following information: teletypewriter number, user number, language being used, program being used, and status of teletypewriter.
UNSAVE	Erases a saved program from memory. Since the memory of the computer is finite, this command should be used to free space in storage for other users' programs.

INDEX

EJERCICIO PARA APLICAR LAS TECNICAS DE EVALUACION DEL RENDIMIENTO DE LAS INVERSIONES

La evaluación financiera de las inversiones tiene por finalidad determinar cual es la inversión mas rentable.

Evalúe los siguientes proyectos empleando los tres siguientes métodos de evaluación:

I Plazo de recuperación

II Tasa de recuperación

III Valor presente

PROYECTO	INVERSION INICIAL	EFECTIVO POR OBTENER AÑO 1	AÑO 2	AÑO 3
A	$1,000	1,000		
B	1,000	500	500	500
C	1,000	100	500	1,500
D	1,000	1,000	100	100
E	1,000	700	300	500
F	1,000	800	800	400

Se supone una tasa de descuento requerida del 4%.

Nótese que este método no diferencia entre las inversiones A y D a pesar de que D es preferible pues se obtienen $200 más, y es que el método ignora toda entrada de efectivo después del plazo de recuperación.

II. - Tasa de recuperación

Inversión A. Aplicando nuestra fórmula:

$$I = \frac{Ua_1}{(1+r)^1}$$

I = 1,000$
Ua=1,000$
r = incógnita

Substituyendo valores:

$$1{,}000 = \frac{1000}{1+r}$$

ó $$1+r = \frac{1{,}000}{1{,}000} = 1$$

$$\therefore \quad r = 0\%$$

El resultado es lógico ya que los $1,000 invertidos, regresan al año sin redituar nada.

Inversión B.

$$I = 1{,}000 = \frac{500}{1+r} + \frac{500}{(1+r)^2} + \frac{500}{(1+r)^3}$$

Para despejar r hay que resolver una ecuación de tercer grado, sin embargo, podemos resolver por tanteo en la forma siguiente:

a) Asignar un valor cualquiera a r.

b) Buscar en las tablas de valor presente los factores $\frac{1}{1+r}$, $\frac{1}{(1+r)^2}$, $\frac{1}{(1+r)^3}$

c) Resolver el lado derecho de la ecuación.

d) Si es igual a I (en esta caso $1,000) entonces hemos encontrado r. Si no, entonces dar otro valor a r y volver al punto (b).

Por ejemplo para la inversión B

a) Supongamos que r tiene el valor 20%.

b) En las tablas de valor presente encontramos que:

$$\frac{1}{1+r} = \frac{1}{1+0.2} = 0.883$$

$$\frac{1}{(1+r)^2} = \frac{1}{(1+0.2)^2} = 0.694$$

$$\frac{1}{(1+r)^3} = \frac{1}{(1+0.2)^3} = 0.579$$

c) Aplicando a nuestra fórmula:

$$I = 0.883 \times (500) + (0.694)(500) + 0.579(500)$$

$$= \$1,028$$

d) Como \$1,028 es mayor que 1,000 r tiene que ser mayor que 20%

de aquí procedemos a tantear 25%

de donde: $I = \$978$

o sea que r está entre 20% y 25%.

Podemos resolver en forma similar las otras alternativas con los siguientes resultados:

Alternativa	r
C	32.5%
D	15.5%
E	25.2%
F	50.0%

Por lo tanto, nuestra clasificación conforme a este criterio es:

METODO DE TASA DE RECUPERACION

Inversión	Tasa de Recuperación	Clasificación
A	0.0%	6
B_1	23.5%	4
C	32.5%	2
D	15.5%	5
E	25.2%	3
F	50.0%	1

III.- Valor presente

K=tasa de descuento requerida = 4% = 0.04

Inversión A:

$$VP = \frac{1,000}{1+k} = \frac{1,000}{1+0.04} = \frac{1,000}{1.04} = 962$$

Inversión B:

$$VP = \frac{500}{1+0.04} + \frac{500}{(1+0.04)^2} + \frac{500}{(1+0.04)^3}$$

en las tablas de valor presente:

al 4% por 1 año: $\frac{1}{1+0.04} = 0.962$

al 4% por 2 años: $\frac{1}{(1+0.04)^2} = 0.924$

al 4% por 3 años: $\frac{1}{(1+0.04)^3} = 0.889$

por lo tanto $VP = 500\,(0.962) + 500\,(0.924) + (500)(0.889)$

$= \underline{\$1,387.5}$

Inversión C:

$$VP = \frac{100}{1+0.04} + \frac{500}{(1+0.04)^2} + \frac{1,500}{(1+0.04)^3} = \$1,892.2$$

Inversión D:

$$VP = \frac{1,000}{1+0.04} + \frac{100}{(1+0.04)^2} + \frac{100}{(1+0.04)^2} = \$1,143.4$$

Inversión E:

$$VP = \frac{700}{1+0.04} + \frac{300}{(1+0.04)^2} + \frac{500}{(1+0.04)^3} = \$1,395.40$$

Inversión F:

$$VP = \frac{800}{1+0.04} + \frac{800}{(1+0.04)^2} + \frac{800}{(1+0.04)^3} = \$1,865.2$$

y los proyectos de inversión estarán clasificados de la forma siguiente:

METODO DEL VALOR PRESENTE

Proyecto	VP	Clasificación
A	$ 962	6
B	$ 1,387.5	4
C	$ 1,892.2	1
D	$ 1,143.4	5
E	$ 1,395.4	3
F	$ 1,865.2	2

RESUMEN

CLASIFICACION SEGUN CRITERIO

PROYECTO DE INVERSION	PLAZO DE RECUPERACION	TASA DE RECUPERACION	VALOR PRESENTE
A	1	6	6
B	4	4	4
C	6	2	1
D	1	5	5
E	4	3	3
F	3	1	2

Puesto que ya hemos discutido que el criterio de "plazo de recuperación" tiene algunas deficiencias prácticas, la solución final se tomaría en base a cualquiera de los otros dos métodos utilizados.

En este caso tendríamos que decidir entre los proyectos C y F puesto que la tasa de recuperación dice que C es preferido y el Valor Presente prefiere a F.

El criterio a seguir es:

* Si se pueden invertir los fondos anuales a un interés mayor que el costo de capital (4%) entonces escogeríamos "F" ya que "F" nos daría una tasa de recuperación mayor que 4% (recordemos que r para "F" era igual a 50%).

* Si los fondos anuales que se reciben no se pueden invertir a más de 4% entonces escogeríamos "C" ya que "C" tiene un VP mayor que "F" al 4%.

Nomenclatura

Comenzaremos el proceso de solución, indicando explícitamente la nomenclatura a utilizar para indicar las variables de decisión del problema.

Símbolo		Significado
Petroleo crudo		
IBA	=	Producido en Indonesia y procesado a baja intensidad en Australia
IAA	=	Producido en Indonesia y procesado a alta intensidad en Australia
PBA	=	Producido en Persia y procesado a baja intensidad en Australia
PAA	=	Producido en Persia y procesado a alta intensidad en Australia
IBJ	=	Producido en Indonesia y procesado a baja intensidad en Japón
IAJ	=	Producido en Indonesia y procesado a alta intensidad en el Japón
PBJ	=	Producido en Persia y procesado a baja intensidad en el Japón
PAJ	=	Producido en Persia y procesado a alta intensidad en el Japón
Gasolinas		
GAF	=	Gasolina enviada de Australia a Filipinas
GAN	=	Gasolina enviada de Australia a N.Zeelandia
GJF	=	Gasolina enviada de Japón a Filipinas
GJN	=	Gasolina enviada de Japón a Nueva Zeelandia
Destilados		
DAF	=	Destilados de Australia a Filipinas
DAN	=	Destilados de Australia a Nueva Zeelandia

DJF	=	Destilados de Japón a Filipinas
DJN	=	Destilados de Japón a Nueva Zeelandia
DUF	=	Destilados de U.S.A. a Filipinas
DUN	=	Destilados de U.S.A. a Nueva Zeelandia

Buques-tanques

BTR	=	Buques-tanque rentados para transporte

Las primeras 18 variables se expresan en miles de barriles diarios (de petróleo crudo, gasolinas o destilados). La última variable representa el número de buques-tanque equivalentes, en unidades de 47 mil toneladas de peso muerto. Todas las variables aceptan valores reales no-negativos.

Restricciones

En base a la nomenclatura anterior comenzamos a elaborar las restricciones del modelo de programación lineal. Habrá restricciones de demanda, de oferta, de transporte, y de tecnología de las refinerias. En particular las restricciones de transporte permiten construir una matriz de transporte (unimodular), y las restricciones tecnológicas permiten construir una matriz de insumo-producto para el par de refinerías existentes.

Restricciones de oferta

El petróleo crudo que se envía de Indonesia a Japón y a Australia, para ser refinado -con cualquiera de los procesos- se ha fijado por contrato en 40 mil barriles diarios:

$$IBA + IAA + IBJ + IAJ = 40 \qquad \text{(en miles de b/d)}$$

El petróleo crudo que se envía de Persia a Japón y a Australia con la misma finalidad, puede alcanzar cualquier magnitud que no supere los 100 mil barriles diarios disponibles en el puerto de Abadán:

$$PBA + PAA + PBJ + PAJ \leq 100 \qquad \text{(en miles de b/d)}$$

Los destilados que se envían de U.S.A. a Filipinas y Nueva Zeelandia, para abastecer ocasionalmente esos mercados, pueden alcanzar cualquier magnitud que no supere los 17 mil barriles - diarios disponibles en Santa Bárbara, California:

$$DUF + DUN \leq 17 \qquad \text{(en miles de b/d)}$$

Restricciones de capacidad

La capacidad de procesamiento de petróleo crudo de ambas refinerías esta limitada: el tope es de 50 mil barriles diarios en Australia y 30 mil barriles diarios en Japón. Estas restricciones operan independientemente de donde provengan los petroleos crudos para refinación y cual sea el nivel de intensidad de proceso que se aplique.

$$IBA + IAA + PBA + PAA \leq 50 \qquad \text{(Australia)}$$

$$IBJ + IAJ + PBJ + PAJ \leq 30 \qquad \text{(Japón)}$$

Restricciones de demanda y excedentes disponibles

La demnada local del mercado australiano y japones se satisface en base a la producción local de gasolinas y destilados. Los excedentes quedan disponibles para su envío a los mercados de Filipinas y Nueva Zeelandia. El nivel de producción de refinados - es una función lineal de los parámetros teonológicos que caracterizan la operación de las refinerías. Esos parámetros contemplan el tipo de petróleo crudo que se utiliza y la intensidad de proceso seleccionada.

Para la producción de gasolinas en Australia, se tiene:

$$.259\ IBA + .365\ IAA + .186\ PBA + .312\ PAA \geq$$

$$9.0 + GAF + GAN$$

en que el miembro izquierdo de la inecuación lineal indica el volumen de oferta en función de la tecnología de la refinería australiana, y el miembro de la derecha indica la composición de la

demanda, dividida entre la demanda local y los excedentes exportables a las Filipinas y a Nueva Zeelandia.

Para la producción de destilados en Australia, se tiene análogamente:

$$.688\ IBA + .573\ IAA + .732\ PBA + .608\ PAA \geq 21.0 + DAF + DAN$$

Para la producción de gasolina en el Japón, se tiene:

$$.259\ IBJ + .350\ IAJ + .186\ PBJ + .300\ PAJ \geq 3.0 + GJF + GJN$$

Para la producción de destilados en el Japón, se tiene:

$$.688\ IBJ + .588\ IAJ + .732\ PBJ + .620\ PAJ \geq 12.0 + DJF + DJN$$

Restricciones de distribución

La demanda local de gasolinas y destilados en los mercados de Filipinas y Nueva Zeelandia se satisface en base a los excedentes de las plantas de Australia y Japón, así como de envios eventuales de los Estados Unidos. Las restricciones del mercado filipino son las siguientes:

$$GAF + GJF \geq 5.00$$

$$DAF + DJF + DUF \geq 8.00$$

Las restricciones del mercado de Nueva Zeelandia son las siguientes:

$$GAN + GJN \geq 5.36$$

$$DAN + DJN + DUN \geq 8.68$$

Restricciones de transporte

La capacidad de transporte disponible mas la que pueda rentarse eventualmente, se utiliza en el transporte de crudos y refinados de acuerdo con los coeficientes denominados "factores de uso" de la capacidad disponible. Una restricción única resume -

las relaciones de oferta y demanda de transporte, en términos de capacidad disponible y capacidad requerida, como sigue:

$$.05\ IAA + .05\ IBA + .12\ PBA + .12\ PAA +$$
$$+.045\ IAJ + .045\ IBJ + .11\ PBJ + .11\ PAJ +$$
$$+ .02\ GAF + .02\ DAF + .01\ GAN + .01\ DAN +$$
$$+ .01\ GJF + .01\ DJF + .06\ GJN + .06\ DJN +$$
$$+.015\ DUF + .18\ DUN \leq 6.9 + BTR$$

Función objetivo

De acuerdo con los datos disponibles se observa que el problema a resolver tiene por objetivo la minimización de los costos totales de refinación, distribución y transporte de todos los productos del sistema. Por lo tanto, es necesario determinar los coeficientes de costo variable que afectan a cada una de las variables del problema, considerando para ello los costos del barril de crudo, los costos de transporte por barril, y los costos de refinación por barril. La información se resume en la tabla siguiente.

Tabla VIII

Costos de refinación, transporte y materia prima

Refinería, insumo y proceso utilizado	costo del crudo	costo de transporte	costo de refinar
Refinería australiana:			
Indonesio, proceso baja intens.	2.60	.26	.12
Indonesio, proceso alta intens.	2.60	.26	.28
Persa, proceso baja intensidad	2.00	.62	.15
Persa, proceso alta intensidad	2.00	.62	.30

valores que sumados horizontalmente nos permiten obtener los siguientes costos totales, respectivamente: 2.98 , 3.14 , 2.77 y 2.92 , en dólares por barril de petróleo crudo de cada tipo.

Refinería japonesa:

Indonesio, proceso baja intens.	2.60	.24	.16	3.00
Indonesio, proceso alta intens.	2.60	.24	.34	3.18
Persa, proceso baja intensidad	2.00	.59	.20	2.79
Persa, proceso alta intensidad	2.00	.59	.39	2.98

em que en la última columna se han indicado los costos totales , por barril de petróleo crudo de cada tipo.

En la tabla III se han detallado los costos de transporte de los refinados (gasolinas o destilados) desde las refinerías a — los mercados de consumo.

De la tabla IV se obtienen los costos totales de los destilados de los Estados Unidos en los mercados de destino, que resultan ser de 2.70 dólares por barril para Nueva Zeelandia y de 2.55 dólares por barril para las Filipinas. Finalmente, se sabe que el costo de buque-tanque rentado es de 3200 dolares por mil barriles.

El conjunto de datos reunido, nos permite elaborar la función objetivo del problema, que se detalla a continuacion:

$$2.98\ IBA + 3.14\ IAA + 2.77\ PBA + 2.92\ PAA +$$
$$3.00\ IBJ + 3.18\ IAJ + 2.79\ PBJ + 2.98\ PAJ +$$
$$0.15\ GAF + .15\ DAF + .10\ GAN + .10\ DAN +$$
$$0.20\ GJF + .20\ DJF + .10\ GJN + .10\ DJN +$$
$$2.55\ DUF + 2.70\ DUN + 3200\ BTR = z$$

Es interesante advertir a efectos de la aplicación del algoritmo de descomposición de Dantzig, que las únicas dos expresiones que vinculan a todas las variables del problema son la función objetivo y la restricción de transporte. Esto implica de que existe una sola restricción " de liga", la del transporte, lo que facilita en gran medida las labores de cálculo de la solución.

A continuación se presenta la primera tabla del problema de programación lineal de este sistema.

MATRIZ DE COEFICIENTES DEL MODELO DE PROGRAMACION LINEAL (SE INDICAN LA MATRIZ DE INSUMO-PRODUCTO Y LA MATRIZ DE TRANSPORTE)

IGA	IAA	PGA	PAA	IBJ	IAJ	PBJ	PAJ	GAF	DAF	GAN	DAN	GJF	DJF	GJN	DJN	BJF	DJN	ATR	T.I.	Variable
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	18	19	17	—	Número
2.98	3.14	2.77	2.92	3.00	3.18	2.79	2.98	.15	.15	.10	.10	.20	.20	.10	.10	2.55	2.70	3.20	—	Coficientes de la Función Objetivo
1	1			1	1														= 40.00	Grado Intermedio
		1	1			1	1												≤ 100.00	Grado Puro
																1	1		≤ 17.00	Capacidad U.S.A
1	1	1	1																≤ 50.00	Capacidad Australiana
				1	1	1	1												≤ 30.00	Capacidad Japonesa
.259	.365	.196	.312					-1		-1									≥ 9.00	Demanda Australiana Granulado
.688	.573	.732	.608						-1		-1								≥ 21.00	Demanda Australiana Pastillado
				.259	.350	.186	.300					-1		-1					≥ 3.00	Demanda Japonesa Granulado
				.688	.588	.732	.620						-1		-1				≥ 12.00	Demanda Japonesa Pastillado
								1			1								≥ 5.00	Demanda Filipina Granulado
									1			1				1			≥ 8.00	Demanda Filipina Pastillado
										1				1					≥ 5.36	Demanda N.Z. Granulado
											1				1		1		≥ 8.68	Demanda N.Z. Pastillado
.059	.050	.120	.120	.045	.045	.110	.110	.020	.020	.010	.010	.010	.010	.060	.060	.150	.180	-1	≤ 6.90	Transporte

MATRIZ DE COEFICIENTES DEL MODELO DE PROGRAMACIÓN LINEAL (RESTRICCIONES ORDENADAS SEGUN REQUERIMIENTOS DE FORMATO DEL PROGRAMA DE CÓMPUTO)

IBA	IAA	PBA	PAA	IBJ	IAJ	PBJ	PAJ	GAF	DAF	GAN	DAN	GSF	DJF	GJN	DJN	BTR	DUF	DUN	T.I.	Variable
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	—	Número
2.98	3.14	2.77	2.92	3.00	3.18	2.79	2.98	.15	.15	.10	.10	.20	.20	.10	.10	3.20	2.55	2.70	—	Coeficiente de la Función Objetivo
		1	1			1	1												≤ 100.00	Crudo Persa
1	1	1	1																≤ 50.00	Capacidad Australiana
				1	1	1	1												≤ 30.00	Capacidad Japonesa
																	1	1	≤ 17.00	Capacidad U.S.A.
.05	.05	.12	.12	.045	.045	.11	.11	.02	.02	.01	.01	.01	.01	.06	.06	-1	.15	.18	≤ 6.90	Transporte
1	1			1	1														= 40.00	Crudo Indonesio
.259	.345	.126	.312					-1		-1									≥ 9.00	Demanda Australiana Gasolinas
.688	.573	.732	.608						-1		-1								≥ 21.00	Demanda Australiana Destilados
				.259	.350	.186	.300					-1		-1					≥ 3.00	Demanda Japonesa Gasolinas
				.688	.588	.732	.620						-1		-1				≥ 12.00	Demanda Japonesa Destilados
								1				1							≥ 5.00	Demanda Filipina Gasolinas
									1				1				1		≥ 8.00	Demanda Filipina Destilados
										1				1					≥ 536	Demanda N.Z. de Gasolinas
											1				1				≥ 868	Demanda N.Z. de Destilados

Solucion

La solución de este problema se obtuvo por el método simplex.

Para la elaboración del programa de cómputo se utilizó el lenguaje de programación Basic. El proceso de cómputo se efectuó en el sistema 360/67 de la Universidad de Stanford.

La solución óptima fue la siguiente:

Variables	Valores
Problema Primal	
1. IBA	0.00000
2. IAA	21.27299
3. PBA	0.00000
4. PAA	28.72697
5. IBJ	10.130947
6. IAJ	8.596004
7. PBJ	0.000000
8. PAJ	0,000000
9. GAF	4.552343
10. DAF	0.000000
11. GAN	3.175135
12. DAN	8.655457
13. GJF	0.4476541
14. DJF	0.000000
15. GJN	2.184866
16. DJN	0.024554
17. BTR	0.000000
18. DUF	8.000000
19. DUN	0.000000

Se obtienen además los siguientes valores para las variables de holgura:

20. HCP	71.27303	holgura en la oferta de petroleo crudo persa
22. HRJ	11.27302	holgura en la capacidad de refinación japonesa
23. HUD	9.00000	holgura en la oferta de destilados de U.S.A.

Problema Dual

Variables	Valores
1.	0.0000000
2.	- .1084481
3.	0.0000000
4.	0.0000000
5.	- .8333855
6.	-4.366457 E-03
7.	4.877697
8.	2.642454
9.	4.836029
10.	2.600782
11.	5.044359
12.	2.675007
13.	4.986031
14.	2.750788

La funcion objetivo alcanza un valor mínimo de 230.9845 .-

Se extraen las siguientes conclusiones generales:

a) el petróleo crudo de Indonesia se utiliza totalmente en los procesos de refinación; aproximadamente un 55% en la refinería australiana y el 45% restante en la refinería japonesa;
b) el petróleo crudo de Persia se utiliza sólo en un 28.7%;
c) el valor negativo de la sexta variable dual, asociada con la restricción de petroleo crudo de Indonesia, indica que el costo total del programa puede reducirse si se renegocia el volumen actual de 40 mil barriles diarios, para obtener un incremento. El análisis de postoptimalidad permitirá determinar el volumen mas conveniente de ese contrato;
d) la refinería australiana trabaja a plena capacidad (50 mil barriles diarios); las operaciones se efectuan a la más alta intensidad de proceso; las proporciones de utilización de petróleos crudos son de 43% de crudos indonesios y 57% de crudos persas; todo el petróleo crudo persa fluye a esta refinería;

e) la refinería japonesa trabaja a un 67% de su capacidad- de 30 mil barriles diarios; procesa exclusivamente petróleo indonesio; utiliza ambas tecnologías de proceso, la de baja intensidad en un 60% y la de alta intensidad en un 40%;

f) el valor negativo de la variable dual segunda, asociada con la restricción de capacidad de refinación de la refinería australiana, indica que el costo total del programa puede reducirse si se amplía la capacidad instalada en la misma;

g) para este programa óptimo no se utilizan buques-tanque rentados; el valor negativo de la 5a. variable dual, asociada con la restricción de transporte, indica que el costo total del programa puede reducirse aumentando la capacidad disponible o rentando unidades de transporte;

h) la demanda de destilados en Filipinas se abastece totalmente con excedentes de los Estados Unidos;

i) para abastecer la demanda local y además producir los excedentes requeridos en los demás mercados, los planes de producción deben alcanzar los siguientes volumenes:

	Gasolinas	Destilados
Refinería japonesa	5.632	12.024
Refinería australiana	16.727	29.655

y algunas otras conclusiones de relevancia menor.

Este plan óptimo calculado a partir del modelo de programación lineal puede compararse con un plan preparado por los métodos tradicionales de planeación, a fin de detectar las deficiencias que se producen al no considerar todas las interrelaciones entre el conjunto de variables de un sistema complejo. Por otra parte, si un análisis ex-post facto indica que el programa óptimo no permite alcanzar los resultados esperados, lo correcto es efectuar una revisión del modelo para detectar cuáles de los supuestos del mismo simplifican excesivamente la realidad, o implican relaciones entre variables que no reflejan la naturaleza auténtica del sistema.

Sujeto a

$$\sum_{i=1}^{n+1} C_i X_i \leq P$$

$$X_i + X_{n+1} \leq 1$$

Con este enfoque, se reducen las restricciones pero se aumentan los proyectos.

Así para el caso de la figura 3-2, se tendrían que analizar, las siguientes agrupaciones de proyectos.

Proyecto	1	1	Proyecto	8	1, 2, 3, 4 y 5
Proyecto	2	1 y 2	Proyecto	9	1, 3, 4, 5 y 6
Proyecto	3	1 y 3	Proyecto	10	1, 3, 4, 5 y 7
Proyecto	4	1, 2 y 3	Proyecto	11	1, 2, 3, 4, 5 y 6
Proyecto	5	1, 3 y 4	Proyecto	12	1, 2, 3, 4, 5 y 7
Proyecto	6	1, 2, 3 y 4	Proyecto	13	1, 2, 3, 4, 5, 6 y 7
Proyecto	7	1, 3, 4 y 5			

Así por ejemplo, el proyecto 12 esta formado por el conjunto de proyectos 1, 2, 3, 4, 5 y 7.

En este caso se analizarían, en lugar de 7, 13 proyectos pero sólo se añadiría una restricción.

Para el primer caso (A):

$$\text{Max} \quad Z = B_1 X_1 + B_2 X_2 + B_3 X_3$$

$$\text{Sujeta a:} \quad C_1 X_1 + C_2 X_2 + C_3 X_3 \leq P$$

Para el segundo caso (B)

$$\text{Max} \quad Z = B_1 X_1 + (B_1 + B_2) X_2 + (B_1 + B_2) X_3$$

$$\text{Sujeta a:} \quad C_1 X_1 + (C_1 + C_2) X_2 + (C_1 + C_2 + C_3) X_3 \leq P$$

$$X_1 + \qquad X_2 + \qquad X_3 \leq 1$$

Para el tercer caso (C).

$$\text{Max} \quad Z = B_1 X_1 + B_2 X_2 + B_3 X_3$$

$$\text{Sujeta a:} \quad C_1 X_1 + C_2 X_2 + C_3 X_3 \leq P$$

Donde: $C_1 X_1 = R_1 X_1 + L_1 X_1$

$$C_2 X_2 = R_2 X_2 + L_2 X_2$$

pero $L_2 = \hat{L}_2 + L_1 (1-X_1)$

$$C_3 X_3 = R_3 X_3 + L_3 X_3$$

donde $L_3 = \hat{L}_3 + \hat{L}_1 (1-X_1 - X_2 + X_1 X_2) + \hat{L}_2 (1-X_2)$

Sustituyendo y simplificando se obtiene la siguiente restricción:

$$X_1 (R_1 + \hat{L}_1) + X_2 (R_2 + \hat{L}_1 + \hat{L}_2) + X_3 (R_3 + \hat{L}_1 + \hat{L}_2 + \hat{L}_3) - X_1 X_2 \hat{L}_1 -$$

$$- X_1 X_3 \hat{L}_1 - X_2 X_3 \hat{L}_1 - X_2 X_3 \hat{L}_2 + \hat{L}_1 X_1 X_2 X_3 \leq P$$

Se llega entonces a un problema de programación matemática con las siguientes características.

Función objetivo lineal

SISTEMA DE EVALUACION Y SELECCION DE PROYECTOS

EJEMPLOS DE FORMULACION DE MODELOS

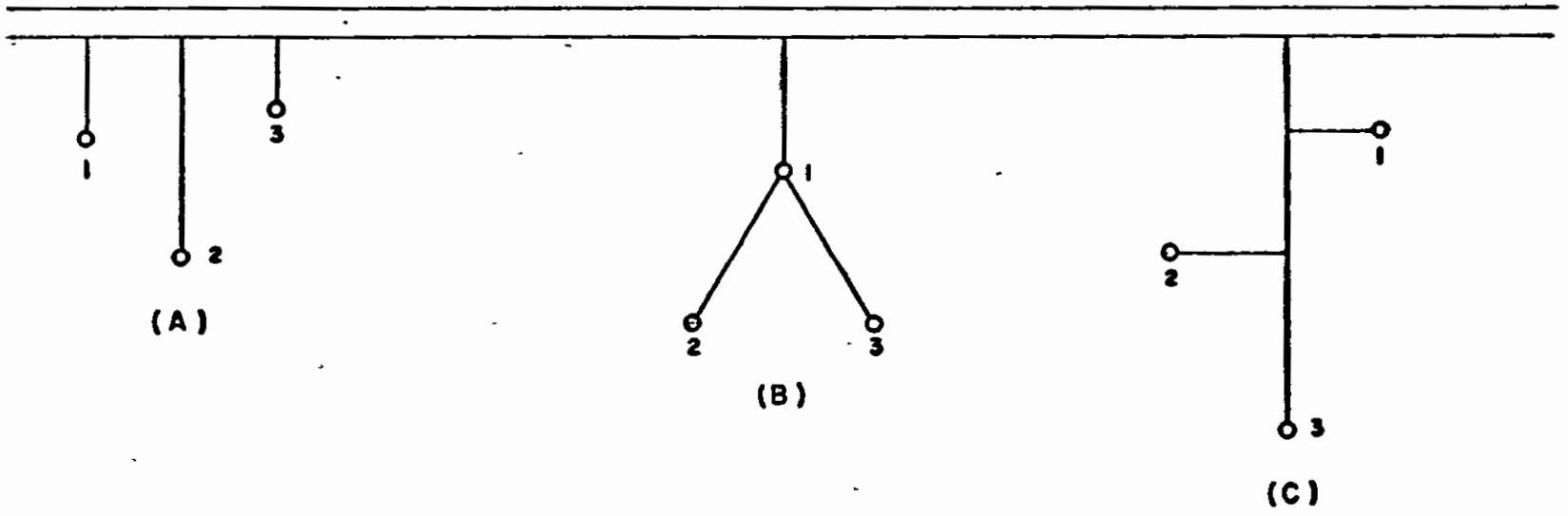


Figura 3.4

SISTEMA DE EVALUACION Y SELECCION DE PROYECTOS

DIAGRAMA QUE MUESTRA EL MANEJO
DE PROYECTOS DEPENDIENTES

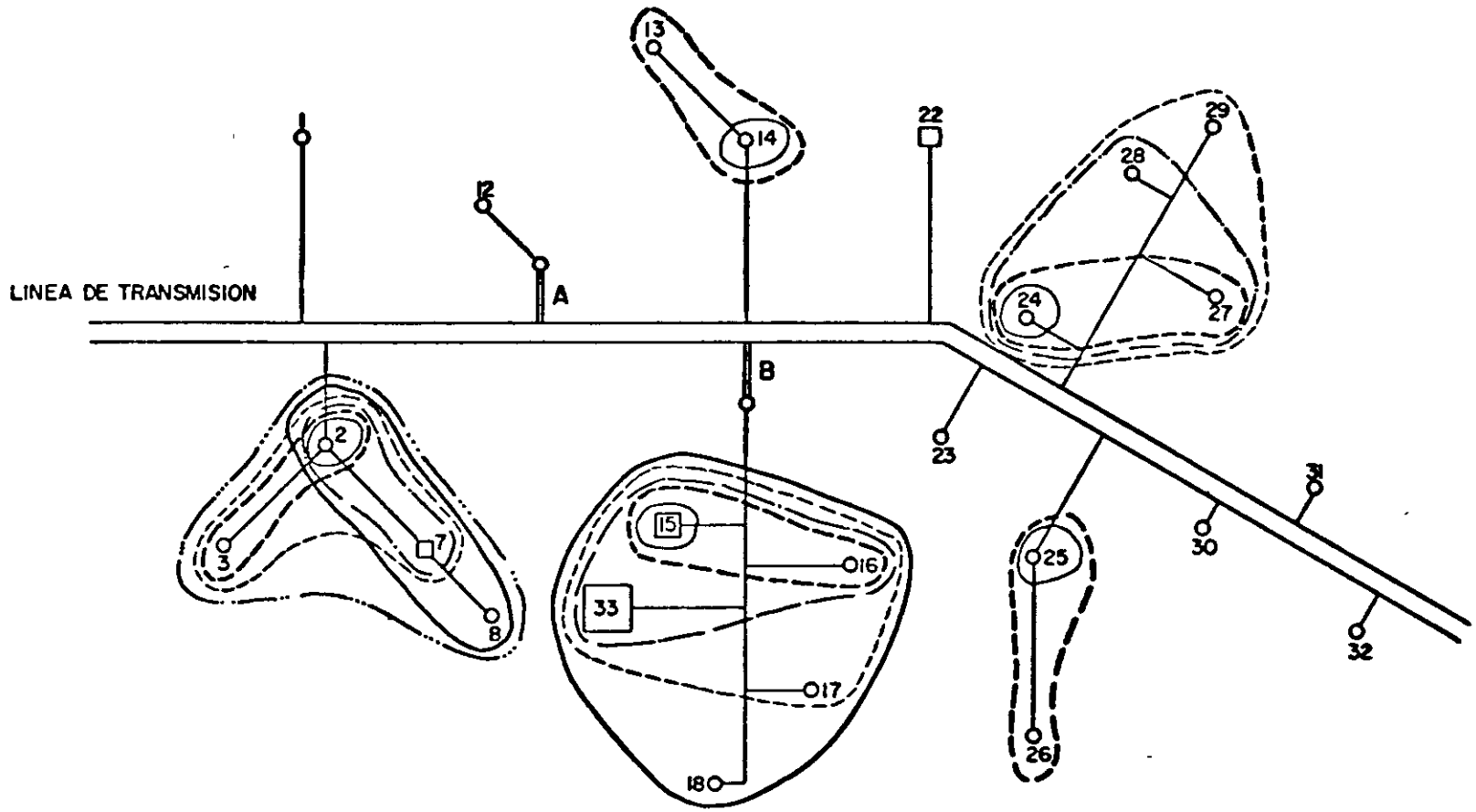


Figura 4.3

SISTEMA DE EVALUACION Y SELECCION DE PROYECTOS

POBLADOS QUE INTEGRAN LA ZONA EN ESTUDIO E INSTALACIONES ELECTRICAS EXISTENTES

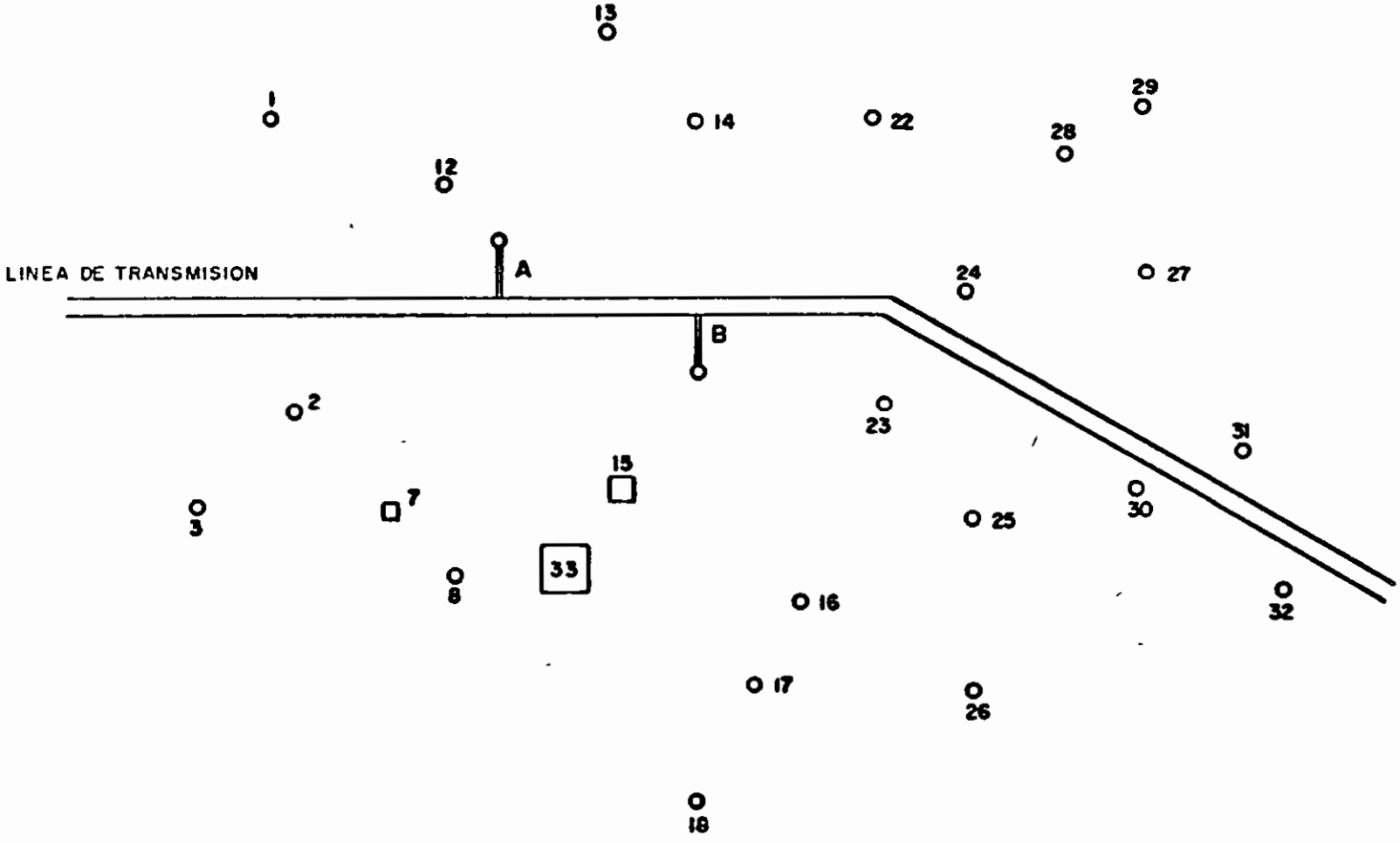


Figura 41

SISTEMA DE EVALUACION Y SELECCION DE PROYECTOS
PROYECTOS SELECCIONADOS EN LAS DIVERSAS PRUEBAS CON EL PRESUPUESTO DISPONIBLE

ALGORITMO DE SELECCION METODO HEURISTICO	PRESUPUESTO DISPONIBLE $ 4 000.00								
POBLADO NUMERO	S=100 E=100 P=100 F=100	S=100 E=0 P=0 F=0	S=0 E=100 P=0 F=0	S=0 E=0 P=100 F=0	S=0 E=0 P=0 F=100	S=70 E=20 P=0 F=10	S=60 E=25 P=5 F=10	S=25 E=60 P=5 F=10	S=40 E=30 P=20 F=10
1				6					10
2	4	3	6	9	4	4	5	6	7
3									
7	4	3	6		4	4	5	6	7
8		3	6					6	
12		7		8					8
13	1			1	6				4
14	1		4	1	6			4	4
15	2	8	3	3	1	7	6	3	3
16	2		3	3	1	7	6	3	3
17				3					
18				3					
22	3	2	1	11		1	1	1	1
23		4	2	10		2	2	2	2
24		6		7	2	8	8		9
25	5	5	5		3	6	7	7	
26									
27									
28									
29									
30				4	5	5	4	5	5
31		9		2					
32	6	1		5		3	3		6
33	2		3	3	1	7	6	3	3

SISTEMA DE EVALUACION Y SELECCION DE PROYECTOS

PROYECTOS ELEGIDOS CON LAS SIGUIENTES CONDICIONES

Presupuesto Disponible		$ 4 000 000
Factores de Ponderación:	Social	0
	Económico	100
	Político	0
	Financiero	0
Valor de la Función de Beneficio:		247.860

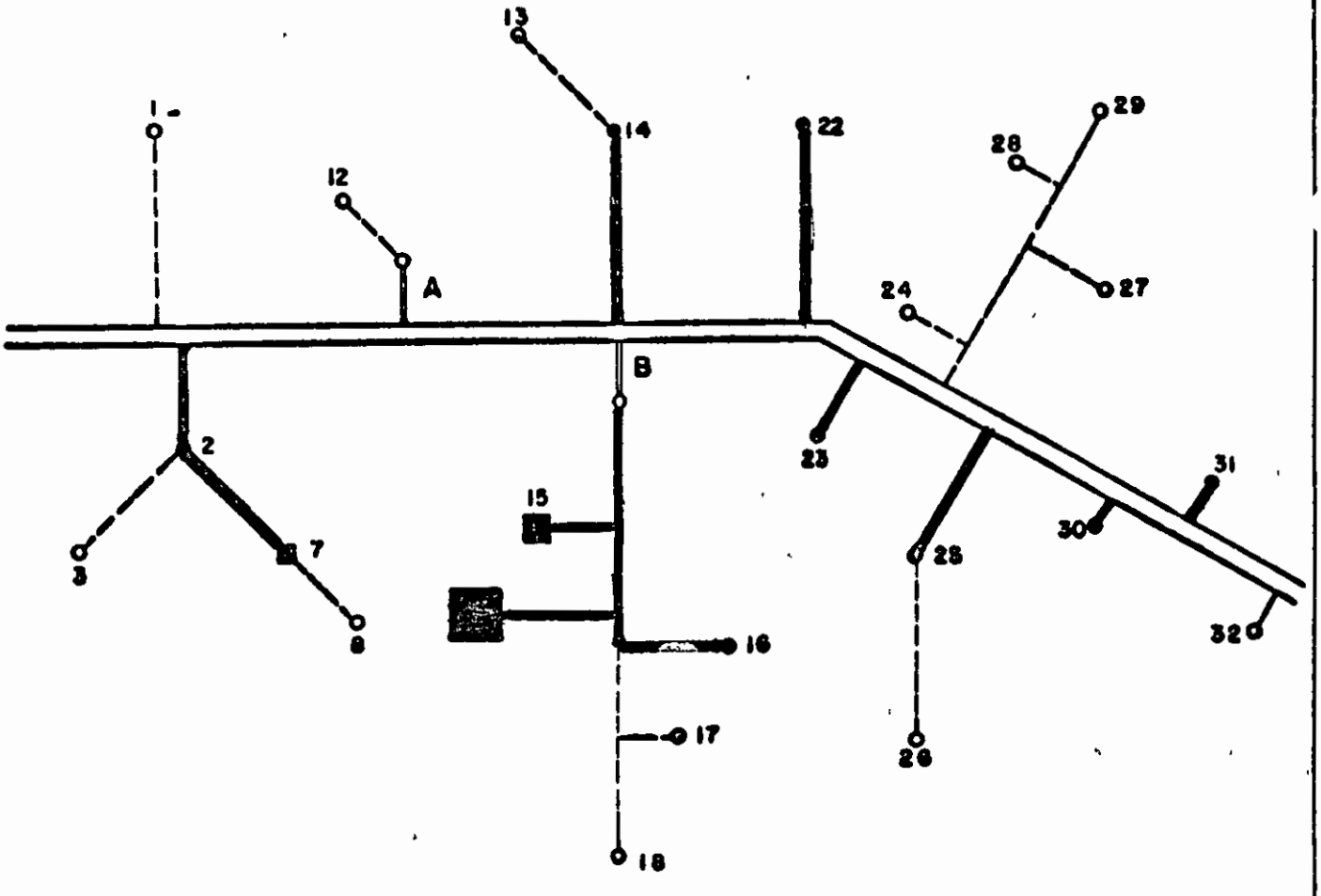
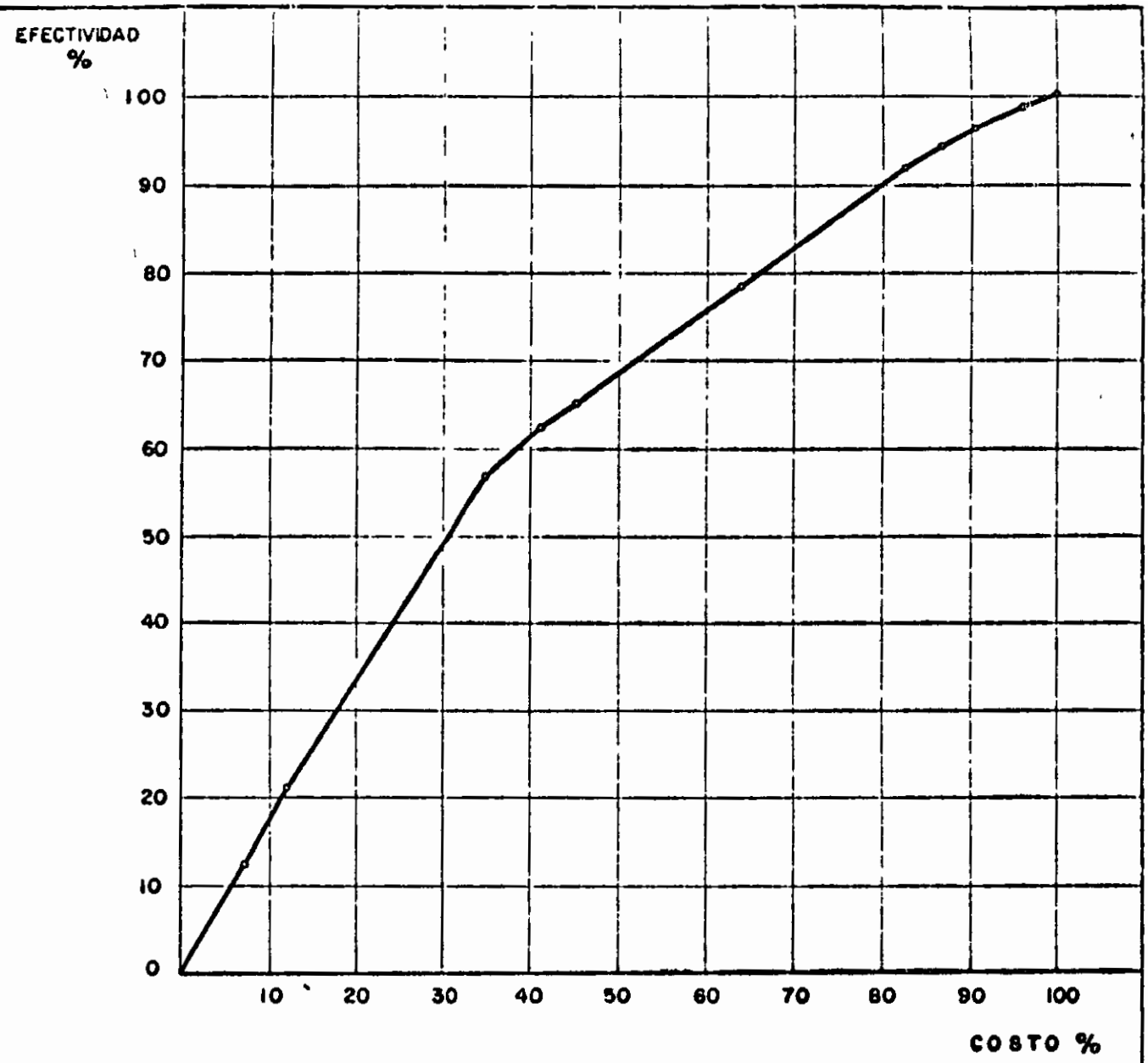


Figura 4.15



FACTORES DE PONDERACION

SOCIAL	ECONOMICO	POLITICO	FINANCIERO
0	100	0	0

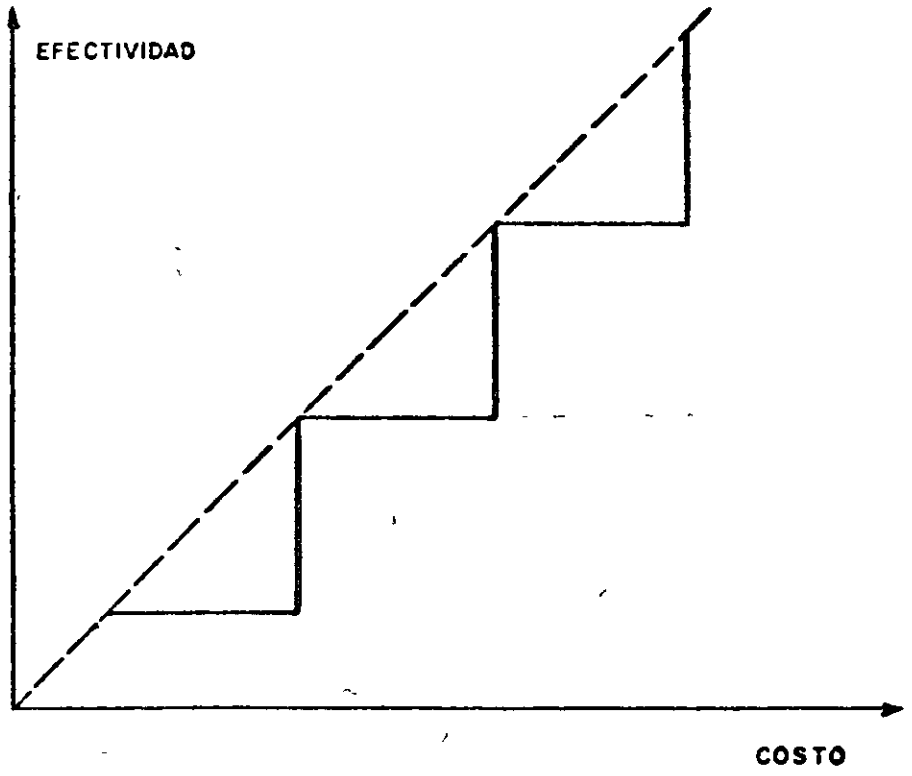
SISTEMA DE EVALUACION Y SELECCION DE PROYECTOS

ANALISIS EFECTIVIDAD-COSTO

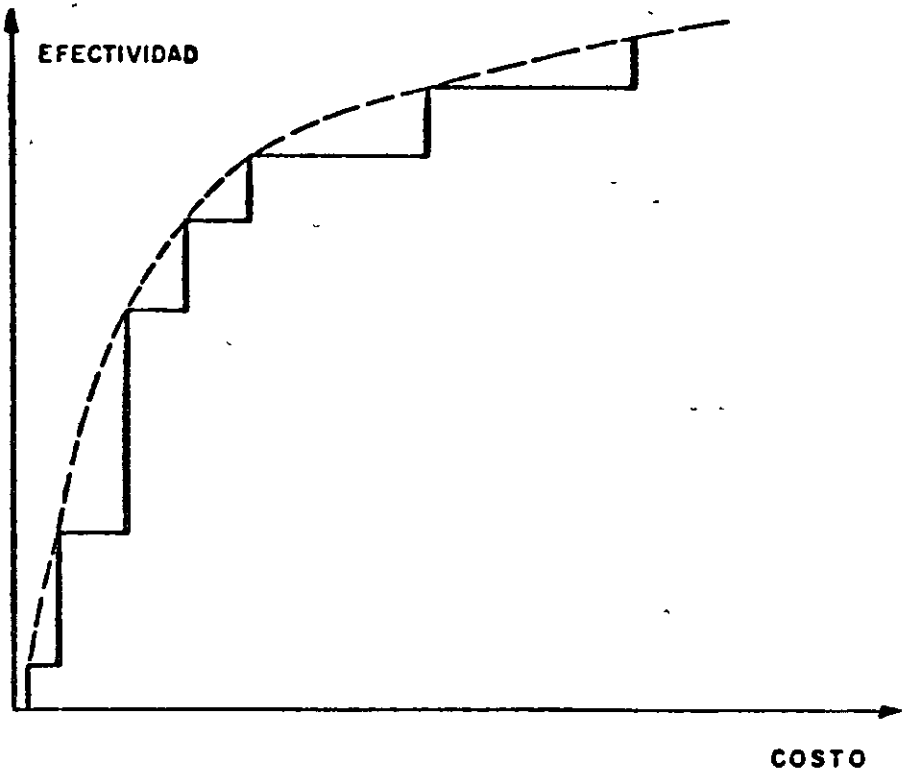
Fig. 4.60

SISTEMA DE EVALUACION Y SELECCION DE PROYECTOS

CURVAS EFECTIVIDAD-COSTO



PROYECTOS IGUALMENTE EFICIENTES



PROYECTOS EFICIENTES E INEFICIENTES

Figura 2.7

Programación de requerimientos financieros. Restricciones del problema.

1	$L + A_1 + P_1 - G_1$	≥ -20
2	$.99L - .015A_1 + A_2 - .03P_1 + P_2 + .005G_1 - G_2$	≥ 10
3	$.98L - .015A_1 - .015A_2 + A_3 - .03P_1 - .03P_2 + P_3 + .005G_1 + .005G_2 - G_3$	≥ 70
4	$.97L - .015A_1 - .015A_2 - .015A_3 + A_4 - .03P_1 - .03P_2 - .03P_3 + P_4 + .005G_1 + .005G_2 + .005G_3 - G_4$	≥ 160
5	$.9[illegible]L - .015A_1 - .015A_2 - .015A_3 - .015A_4 + A_5 - .03P_1 - .03P_2 - .03P_3 - .03P_4 + P_5 + .005G_1 + .005G_2 + .005G_3 + .005G_4 - G_5$	≥ 130
6	$.95L - .015A_1 - .015A_2 - .015A_3 - .015A_4 - .015A_5 + A_6 - .03P_1 - .03P_2 - .03P_3 - .03P_4 - .03P_5 + P_6 + .005G_1 + .005G_2 + .005G_3 + .005G_4 + .005G_5 - G_6$	≥ -20
7	A_1	≤ 52.5
8	A_2	≤ 37.5
9	A_3	≤ 52.5
10	A_4	≤ 90
11	A_5	≤ 75
12	A_6	≤ 37.5
13	P_1	≤ 30
14	P_2	≤ 90
15	P_3	≤ 100
16	P_4	≤ 60
17	P_5	≤ 40
18	P_6	≤ 50
19	L	$\leq 1[illegible]0$
20	L	≥ 40

OPERACIÓN INDUSTRIAL INTEGRADA

Se presenta a consideración un plan integral para las operaciones de una empresa internacional de producción, refinación, transporte y distribución de petróleo crudo, gasolinas y productos petroquímicos. La denominaremos "Petroleos Interoceánicos, Sociedad Anónima" (P.I.S.A.).

La empresa P.I.S.A. está organizada como un consorcio de tipo "holding," que es propietario total o parcial de varias compañías subsidiarias, las que son responsables por las operaciones específicas. El problema fundamental de la empresa P.I.S.A. es lograr un plan integral que coordine las actividades de las diversas subsidiarias, optimice los resultados de la operación, y no restrinja excesivamente la autonomía de dichas subsidiarias en la planeación de las actividades que les sean propias.

El primer plan integral se puso en práctica a principios de 1968, coincidiendo con la asamblea anual del Consejo Directivo de la empresa "holding", en la que se analizaron los resultados económicos y financieros del ejercicio anterior, y se aprobó el esquema de plan integral.

El plan fue elaborado durante el segundo semestre de 1967 por el Departamento de Estudios Económicos de la empresa, con el asesoramiento de algunos funcionarios ejecutivos de las empresas subsidiarias, de ingenieros petroleros del Departamento de Estudios de Operaciones, y de algunos viejos directores cuya iniciativa había salvado a la empresa en los difíciles años de la postguerra. En lo que respecta a sus características fundamentales, el plan resumía la experiencia de los diversos grupos de intereses, incorporaba apreciaciones subjetivas, y evitaba una excesiva rigidez que perjudicara la toma de decisiones a nivel de empresa subsidiaria. Era más bien un marco general de estrategia a corto plazo, que proporcionaba las orientaciones globales de la empresa, en la inteligencia de que las

experiencias que se acumularían durante el proceso de aplicación del plan, permitirían efectuar revisiones anuales a fin de pulir detalle afinar procedimientos, incorporar nuevos criterios y mas que nada, adecuarse a las condiciones cambiantes del mercado internacional de crudos y refinados.

Al finalizar el segundo año de operaciones bajo el nuevo plan, se notó que las cosas no marchaban como se había pensado. A pesar del proceso anual de revisión de metas, planes y políticas, la empresa comenzó a perder terreno frente al avance de empresas competidoras mas agresivas. Además los resultados generales de operación no eran muy brillantes, ya que en promedio se repetían casi los mismos valores de años anteriores en los índices financieros y económicos del análisis de resultados. Y por si esto fuera poco, los ejecutivos de las empresas subsidiarias se quejaban de que por lo general, el plan no era suficientemente detallado y flexible como para tomar en cuenta adecuadamente las condiciones específicas de operación de cada una de las mismas. En algunos casos, había ocurrido que iniciativas interesantes para el desarrollo de operaciones a nivel local, que hubieran permitido ganar mercados específicos en razón de crisis políticas internacionales, no pudieron ser aprovechadas oportunamente porque requerían inversiones que excedían los marcos generales delineados por el plan. El proceso de análisis de las nuevas circunstancias, hubiera requerido un lapso de tiempo tan dilatado, que la oportunidad se habría perdido de todas maneras.

Asi es que a mediados de 1970 se reunió el Consejo de Directores y resolvió contratar algunos especialistas para que se enfocara el problema integral de acuerdo con la técnica de análisis de sistemas, que se construyera un modelo de descripción, análisis y predicción, y que se recurriera a los métodos cuantitativos mas adecuados para la solución del modelo y elaboración del plan integral de operaciones.

Los especialistas en técnicas cuantitativas trabajaron en la realización dé un diagnóstico detallado de la estructura de la empresa, las interrelaciones entre los diversos subsistemas, los principales problemas a encarar, etc., y sugirieron que la programación lineal parecía ser la técnica más adecuada para englobar todos los factores involucrados, optimizar los resultados de operación, tomar en cuenta condiciones específicas de operación de cada subsidiaria, y analizar nuevos planes y políticas cuando cambiaran las condiciones del mercado o variaran algunos supuestos iniciales del análisis.

El Consejo de Directores autorizó se continuara adelante en la elaboración de un modelo integral, por lo que se decidió aplicar la programación lineal a nivel global, resolver el modelo obteniendo un plan de operaciones, y analizar la sensitividad de la solución ante variaciones en los parámetros estimados. El plan se entregó para su aplicación al iniciarse el ejercicio económico de 1971; se propuso un período de un año para verificar resultados, comparar con el sistema anterior, recibir críticas y nuevas iniciativas, y pulir detalles no suficientemente elaborados. En el transcurso del año se perfeccionaría el modelo mediante el análisis de postoptimalidad, tratando de adecuarse progresivamente a la realidad, obteniendo la mayor información posible del plan óptimo propuesto, y analizando lòs modificaciones para adecuarse a cambios en las condiciones de operación. En el año siguiente, se tomarían en en consideración los problemas específicos de la relación entre el plan integral y los planes de cada subsidiaria, se evaluarían todos los proyectos particulares dentro del encuadre global, y se obtendría el plan definitivo mediante un proceso de mutuo ajuste, hasta el logro de una compatibilidad total de todas las condiciones y restricciones.

En las páginas siguientes se presentan las consideraciones referentes a: estructura general de la empresa, operaciones de refinación, operaciones de distribución y operaciones de transporte.

ESTRUCTURA Y
Operaciones generales

La empresa P.I.S.A. cuenta con dos fuentes importantes de aprovisionamiento de petroleo crudo: Persia e Indonesia. El petróleo de Persia es el mas pesado, cuesta dos dólares por barril, se carga en el puerto petrolero de Abadan, y se tienen asegurados cien mil barriles diarios durante todo el año.

El petróleo de Indonesia se obtiene de los campos petroleros de Brunei, en la isla de Borneo; es mas liviano que el persa, cuesta 2.60 dólares por barril, y se tiene un contrato de cuarenta mil barriles diarios con la Compañía Petrolera Neerlandesa.

Hay dos empresas subsidiarias dedicadas exclusivamente a operaciones de refinación. Una de ellas tiene su sede en Australia, con una planta de refinación en Sidney, cuya capacidad es de cincuenta mil barriles diarios de proceso. La empresa distribuye sus refinados en el mercado australiano, y cuenta con excedentes disponibles que exporta a los demás mercados de la empresa.

La otra subsidiaria para oporaciones de refinación está situada en el Japón, cerca de Nara, y su capacidad de refinación es de treinta mil barriles diarios. Abastece el mercado japonés y sus excedentes se exportan tambien a otros mercados.

Hay dos expresas subsidiarias dedicadas exclusivamente a operaciones de distribución: una de ellas en las Filipinas y la otra en Nueva Zeelandia. Los mercados que estas subsidiarias surten se proveen de productos refinados en Japón o en Australia, o en casos especiales de escasez transitoria, de la empresa subsidiaria de Santa Bárbara, California, U.S.A., que atiende regularmente el mercado de los Estados Unidos.

Finalmente, la empresa cuenta con una flota de buques tanque, que utiliza para transportar crudos o refinados entre

las diversas subsidiarias de acuerdo con las necesidades específicas de cada lugar.

Operaciones de refinación

La operación de una refinería es un proceso complejo. Conviene utilizar un modelo de programación lineal para tomar en consideración todas las interrelaciones posibles. Los principales factores que influyen en la complejidad del proceso son las características de los petróleos crudos, la tecnología específica de la refinería, los diversos productos finales y subproductos del proceso, los niveles de producción, y las restricciones tecnológicas y económicas del proceso. En realidad, y a efectos de programar las operaciones sobre una base semanal (o a veces diaria), ambas refinerías deberían implantar un modelo de programación lineal de sus procesos. Los especialistas estimaron que esos modelos estarían integrados por aproximadamente 300 variables y unas 100 restricciones.

No obstante, con fines de planeación anual (y de mantener este problema dentro de un rango manejable) se pueden aceptar algunos supuestos simplificadores. Por ejemplo, supondremos sólo dos tipos de insumo: el petróleo crudo persa y el indonesio. Además reduciremos la gama de posibles productos del proceso a dos grandes variedades: productos de gasolina y productos de destilados (como el fuel-oil y aceites minerales diversos). En lo referente a tecnologías de proceso, si bien hay por lo general una amplia flexibilidad de selección de la intensidad del proceso, supondremos que a efectos de planeación se consideran sólo los casos extremos, es decir la tecnología de más alta intensidad y la de más baja intensidad de proceso.

En la tabla siguiente se indican los parámetros del proceso de refinación para ambas refinerías, de acuerdo con el tipo de petróleo crudo utilizado y con el producto final del proceso.

Tabla I

Parámetros tecnológicos del proceso de refinación de petróleo

Refinería, insumo y proceso utilizados	Producto final (en barriles de producto por barril de insumo)	
Refinería australiana:	**Gasolinas**	**Destilados**
Indonesio crudo, baja intensidad	.259	.688
Indonesio crudo, alta intensidad	.365	.573
Persa crudo, baja intensidad	.186	.732
Persa crudo, alta intensidad	.312	.608
Refinería japonesa:		
Indonesio crudo, baja intensidad	.259	.688
Indonesio crudo, alta intensidad	.350	.588
Persa crudo, baja intensidad	.186	.732
Persa crudo, alta intensidad	.300	.620

Operaciones de distribución

Las operaciones de distribución se efectúan en las áreas de Australia y Japón directamente, y en las de Filipinas y Nueva Zelandia, mediante el uso de buques-tanque. Por intermedio del Departamento de Estudios Económicos se han efectuado estimaciones de la demanda probable para el año 1971 en cada una de esas áreas.

Tabla II

Estimaciones para la demanda de gasolinas y destilados para las diversas áreas de distribución en 1971

Mercado de los productos	Demanda (en miles de barriles diarios) Gasolinas	Destilados
Australia	9.00	21.00
Japón	3.00	12.00
Filipinas	5.00	8.00
Nueva Zeelandia	5.36	8.68
Totales	22.36	49.68

Se dispone tambien de los costos promedio de transporte de gasolinas o destilados desde las refinerías a los mercados de - Filipinas y Nueva Zeelandia.

Tabla III

Costos de transporte de gasolinas o destilados (en dólares por barril de destilado)

Origen / Destino	Destino	
	Nueva Zeelandia	Filipinas
Australia	.10	.15
Japón	.10	.20

Operaciones de transporte

Como se ha expresado anteriormente, los buques-tanque de la flota se utilizan para realizar transporte de petróleo crudo a - las refinerías, además de los transportes de productos refinados a los mercado de consumo. Los costos de transporte de petróleo - crudo desde Persia e Indonesia a las refinerías se indican a continuación, en la tabla IV.

Tabla IV

Costos variables de transporte de petróleo crudo
(en dólares por barril de crudo)

Tipo de insumo y de proceso	Refinería de Australia	Refinería de Japón
Indonesio crudo, baja intensidad	.26	.24
Indonesio crudo, alta intensidad	.26	.24
Persa crudo, baja intensidad	.62	.59
Persa crudo, alta intensidad	.62	.59

Debemos considerar además, la capacidad de la flota de buques-tanque y los requerimientos de transporte en términos de utilización de la capacidad. Se acostumbra a convertir los buques-tanque a fracciones de un buque prototipo equivalente, cuyo tonelaje es de 47 mil, medido en unidades de tonelaje de peso muerto, (DWT). En el caso particular de la flota que consideramos, la capacidad total es de 6.9 unidades equivalentes de 47 mil DWT. Ademas de la capacidad disponible, es factible utilizar los servicios de buques-tanque independientes, fletados por compañías transportistas internacionales. El costo promedio de transporte resulta ser de 3200 dólares por mil barriles diarios de despacho.

En lo referente a la capacidad de transporte requerida, se tiene que ésta varía de ruta a ruta segun diversos factores, que dependen de la distancia recorrida, uso de facilidades portuarias, etc. En la tabla V se indican las fracciones de buque-tanque equivalente, de 47 mil DWT, necesarias para despachar mil barriles diarios de petróleo crudo en las rutas ya indicadas.

Tabla V

Factores de uso de capacidad de buques-tanque
(en fracciones de buque-tanque equivalente)

Origen \ Destino	Australia	Japón
Persia	.12	.11
Indonesia	.05	.045
Filipinas	.02	.01
Nueva Zeelandia	.01	.06

Otras fuentes de abastecimiento

Como ya hemos mencionado, existe una posibilidad adicional de surtir los mercados de la empresa con sobrantes transitorios de petróleo refinado o de destilados, provenientes de una planta situada en California. Para el año 1971, las estimaciones de producción y demanda para los Estados Unidos indicaban que se presentaría un excedente de 17 mil barriles diarios de destilados en esa planta, y que no habría excedentes de gasolina.

El costo del destilado producido en U.S.A. es de dos dólares por barril. Los costos de transporte a los mercados de distribución, y los requerimientos de capacidad de buques-tanque se indican en la tabla VI.

Tabla VI

Costos de transporte y requerimientos de capacidad para los destilados producidos en los U.S.A.

Destino	Costo de transporte	Requerimientos de capacidad
Nueva Zeelandia	.70	.18
Filipinas	.55	.15

Otros costos

A fin de completar los datos económicos y tecnológicos necesarios para armar el modelo representativo de la operación simplificada del sistema, se incluyen en la Tabla VII los costos variables de refinación, que dependen de la intensidad del proceso y del tipo de insumo utilizado.

Tabla VII

Costos variables de refinación (en dólares por barril)

	Australia	Japón
Indonesio crudo, baja intensidad	.12	.16
Indonesio crudo, alta intensidad	.28	.34
Persa crudo, baja intensidad	.15	.20
Persa crudo, alta intensidad	.30	.39

PROGRAMACIÓN DE REQUERIMIENTOS DE PERSONAL

El Departamento de Planeación de una importante empresa de fabricación de relojes, está tratando de decidir cuantas operarias se contratarán y entrenarán durante el primer semestre del año próximo para el Departamento de Armado.

El número de horas de trabajo varía de acuerdo con los planes mensuales de producción, por lo que se estimaron las siguientes cifras:

Mes	Horas necesarias
Enero	8 000
Febrero	7 000
Marzo	8 000
Abril	10 000
Mayo	9 000
Junio	12 000

Dos factores complican la programación del reclutamiento de operarias: 1) Se requiere todo un mes para entrenar una operaria, antes de que adquiera la habilidad necesaria para trabajar sin errores; por lo que la contratación debe efectuarse con un mes de anticipación; 2) El entrenamiento de una operaria nueva está a cargo de una operaria experta, y exige 100 horas de enseñanza y control durante el mes de capacitación.

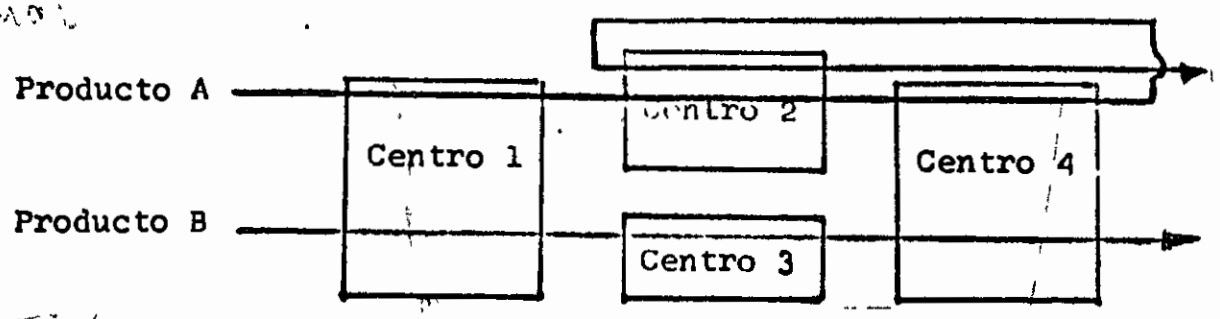
Para el mes de enero no habría problemas, pues se contaría con 60 operarias. De acuerdo con el contrato colectivo de trabajo, el número máximo de horas mensuales de actividad no debe superar de 150, por lo que se tendrían 9 000 horas laborables disponibles.

El sueldo mensual de una operaria entrenada, incluyendo prestaciones sociales, es de $1,600.00. El sueldo de una operaria nueva, es de $800.00 durante el período de entrenamiento.

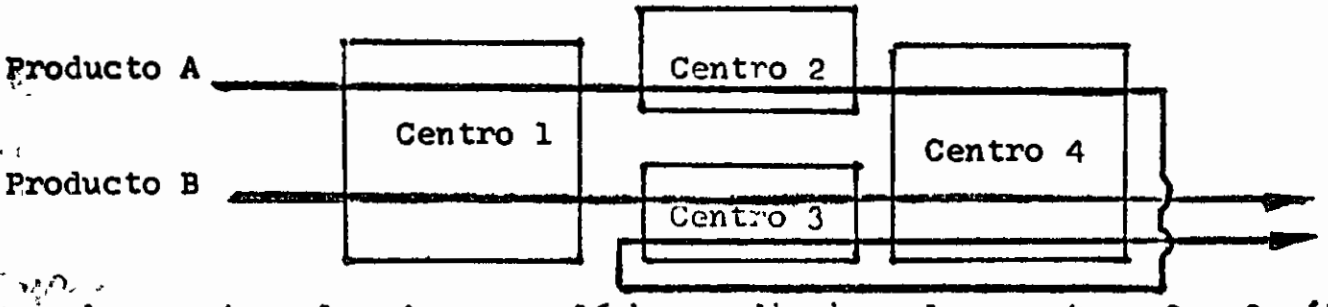
Finalmente el Departamento de Personal ha estimado en el 10% la tasa mensual de renuncias de operarias, (por razones familiares, de salud, cambio de trabajo y otras causas).

Se pide:
a) Indicar las variables del problema
b) Plantear y explicar cada restricción
c) Formular la función objetivo
d) Comentar el modelo elaborado

Una pequeña empresa química produce dos tipos de solventes, el A y el B. La planta está organizada en cuatro centros de procesamiento: 1,2,3 y 4. El diagrama usual del proceso es el siguiente:



El centro de procesamiento 3 presenta capacidad en exceso a la requerida por el producto B. Por lo tanto, es posible procesar el producto A pasando por el centro 3, en vez de pasarlo dos veces por el centro 2. En tal caso, la utilización de la capacidad estaría mejor balanceada aunque los costos por unidad serían algo mayores. El diagrama de proceso sería el siguiente:



Los centros 1 y 4 operan 16 horas diarias; los centros 2 y 3 sólo 12 horas diarias. No hay limitaciones en la capacidad de almacenaje, pero la capacidad de los ductos limita el volumen diario producido de ambos solventes a un máximo de 2500 litros.

Se conocen los costos de materia prima, el precio de venta por litro, las ventas máximas diarias y los requerimientos de materia prima por hora de operación, para ambos productos en cada proceso. Los datos son los siguientes:

Producto	Costo de materia prima (por litro)	Precio de venta (por litro)	Ventas máximas diarias (en litros)
A	5	20	1700
B	6	18	1500

Producto	Centro de procesamiento	Insumos (litros por hora)	% de recuperación	Costo de operación ($ por hora)
A	1	300	90	150
	2 (1er paso)	450	95	200
	4	250	85	180
	2 (2° paso)	400	80	220
	3	350	75	250
B	1	500	90	300
	3	480	85	250
	4	400	80	240

Se pide formular un modelo lineal de programación de la producción que permita maximizar las utilidades. El programa óptimo de producción debe responder a las siguientes preguntas:

a) ¿Cuanta materia prima se utilizará diariamente en la producción del solvente A por el proceso usual?

b) ¿Cuanta materia prima se utilizará diariamente en la producción del solvente A por el proceso opcional?

c) ¿Cuanta materia prima se utilizará diariamente en la producción del solvente B?

Programación de objetivos financieros

El contralor financiero de la empresa comercial "Viena-Bruselas" ha tratado de resolver matemáticamente el problema de la planeación financiera para el período septiembre 1971 a febrero 1972. Debido al incremento de las ventas de fin - de año, la empresa necesita disponer de una amplia disponibilidad de dinero, en especial en los meses de noviembre y diciembre. En los primeros meses del año, y en especial en febrero, hay grandes excedentes monetarios, que provienen de - los cobros de los documentos firmados por los clientes en -- los meses anteriores.

Las fuentes de financiamiento son las siguientes:

1. Descuento de documentos comerciales. El "Banco Industrial y Financiero" ha ofrecido prestar hasta un 75% de los importes mensuales de los documentos firmados por los clientes, sobre una base estrictamente mensual. El costo de la operación es del 1.5% mensual del importe descontado.
2. Prolongación de los plazos para el pago de las compras. Los proveedores de la empresa "Viena-Bruselas" aceptan que el plazo para cobrar se prolongue hasta un mes como máximo. En caso de que se recurra a ese tipo de financiamiento, la empresa pierde el descuento del 3% por pronto pago, al que normalmente tiene derecho.
3. Préstamo a corto plazo. El "Banco Servicial" ha ofrecido prestar cualquier importe comprendido entre 40 mil y 100 mil pesos sobre una base semestral. Una vez decidido el importe - de ese préstamo, queda fijo por un período de seis meses, durante el cual no se puede aumentar ese importe, ni disminuirlo mediante pagos parciales. El préstamo debe contratarse en Septiembre y reembolsarse en Febrero. El costo de esta operacion es del 1% mensual, y debe pagarse cada mes.

Por otra parte, si la empresa "Viena-Brúselas" llega a disponer de excedentes monetarios, puede invertirlos en bonos federales a corto plazo, que rinden el 0.5% mensual y son de convertibilidad inmediata.

En la formulación del modelo mas adecuado para este problema se pueden aceptar muy diversos supuestos adicionales, referentes a la interpretación de ciertas cifras y a la secuenciación de las actividades. Para facilitar el planteo se supondrá que todas las transacciones ocurren al principio de cada mes. El saldo en efectivo al comienzo del período de planeación se supone nulo. El efectivo necesario para las operaciones, y el efectivo sobrante de las operaciones corresponden a cambios que ocurren durante el mes, y no son acumulables de un mes a otro. Las cifras correspondientes a pagos por compras son valores netos, es decir despues de deducir el descuento por pronto pago. Si se recurre al financiamiento por prolongación del plazo para el pago de las compras, el importe total es igual al valor neto mas el 3% adicional.
Los importes que se reciben por descuento de documentos comerciales se reciben al inicio de cada mes y se reembolsan al inicio del mes siguiente. El efectivo sobrante se invierte en bonos federales al inicio de cada mes. El préstamo a corto plazo se obtiene el primer día del mes de Septiembre y se reembolsa el último dia del mes de Febrero.

Formule este caso de toma de decisiones sobre financiamiento a corto plazo, como un problema de asignación óptima de recursos.