CAPÍTULO 3

CARACTERIZACIÓN DEL ALGORITMO GENÉTICO

CODIFICACIÓN DE LAS SOLUCIONES

La codificación de las soluciones para un *problema de programación de actividades para un sistema de Producción intermitente* o (*JSSP)*, es la base para la programación del algoritmo. De la codificación de las soluciones dependerán los principales operadores que lo componen. Una cadena de caracteres que define una solución puede tener mucha o poca información, en general, entre más sea la información contenida en una solución será más fácil la interpretación de ésta para transformarla en una secuencia de actividades. Por otro lado, la simplicidad en la codificación de las soluciones se reflejará en la simplicidad de la programación para el cruzamiento de los individuos, de la mutación de un individuo, o más aun, en la forma en que se evalúe su desempeño. Dicho de otra forma, una vez que se conoce la codificación que se utilizará, se puede pensar en cómo se cruzaran los individuos, como mutaran y como se evaluará su desempeño. Existen en general 2 formas de codificación para las soluciones, la forma Directa y la Forma indirecta. Cuando se codifica de forma directa, la secuencia de actividades se encuentra codificada en el cromosoma que forma a nuestro individuo. Cuando se trabaja con una codificación de tipo indirecta, los genes que componen el cromosoma o individuo pueden representar no las operaciones a programar, si no una lista de reglas de despacho que se busca acomodar de manera que aplicadas estas en un orden específico, se minimice un parámetro en especial.

Para la programación del Algoritmo Genético (AG), en este trabajo se utilizará la codificación desarrollada por Gen, Tsujimura y Kubota (1994). Esta codificación establece que el cromosoma de las soluciones represente en cada uno de sus genes una operación, se trata pues, de una codificación directa. Gen y sus colaboradores nombraron a todas las operaciones de cada trabajo con un símbolo idéntico y se interpretan de acuerdo al orden de aparición en el cromosoma que representa el programa de actividades. Para un trabajo con *m* máquinas y *n* trabajos el cromosoma tendrá *nxm* genes. El símbolo de cada trabajo aparecerá exactamente *m* veces. El ejemplo del cromosoma se muestra en la siguiente figura:

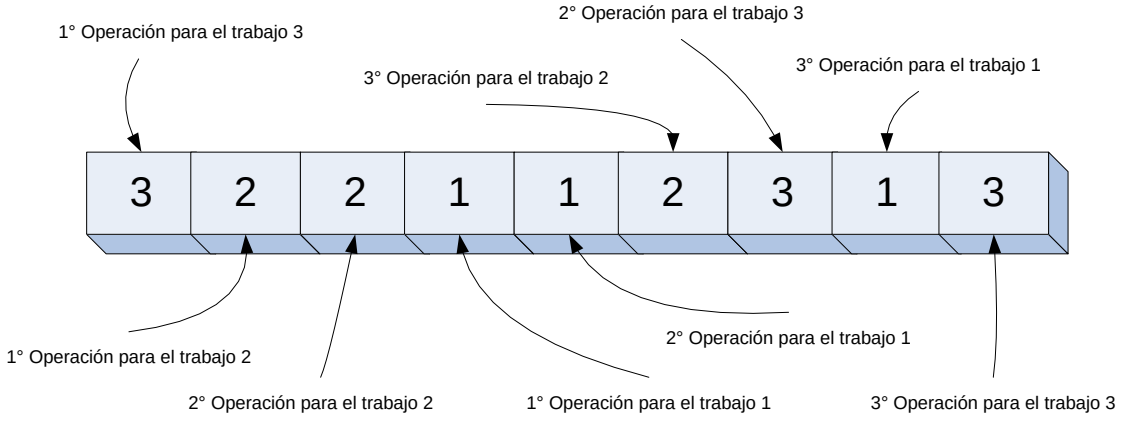

FIGURA 3.1 CROMOSOMA PARA UN ALGORITMO GENÉTICO,
REPRESENTACIÓN BASADA EN OPERACIONES PARA EL JSP

Cualquier permutación de genes en esta codificación asegura que siempre se obtenga una secuenciación de actividades válida. Esta es la ventaja operacional que muestra este tipo de codificación basada en operaciones. La secuenciación obtenida para la codificación anterior se muestra en la figura no **3.2**.

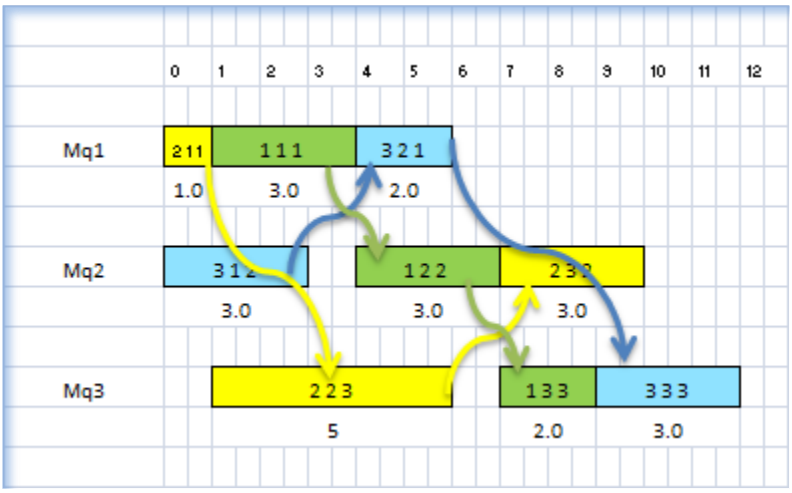

Figura 3.2 Secuencia obtenida para la solución [3 2 2 1 1 2 3 1 3]

FUNCIONES Y PARÁMETROS DEL ALGORITMO GENÉTICO

La programación del AG se realizará utilizando la herramienta *Optimtool* de MATLAB versión R2009a la cual proporciona una interfaz con la capacidad de graficar el comportamiento del algoritmo y que permite probar con diferentes parámetros y escoger los que mejor rendimiento aporten al algoritmo genético. Esta herramienta permite programar las funciones de *cruzamiento*, *mutación*, la función de evaluación del *Intervalo Operativo* para las soluciones y la función para generar la población inicial, las cuales son particulares de la codificación utilizada.

Operador *Cruzamiento*

Este operador cruzamiento realiza el cruce de secuencias parciales por dos puntos, como se describió en el capítulo 2, a semejanza de los *bloques de construcción* (*building blocks)* utilizados por Holland (1975). Estos bloques son secuencias parciales dentro de las soluciones, que suponen buenas características de una solución, el buen funcionamiento de los algoritmos genéticos se basa en que conforme realizan la búsqueda de una solución

óptima, los individuos con este tipo de bloques incrementan su número en cada generación de forma exponencial.

Las figuras 3.3 a 3.6 muestran los pasos para realizar el cruce de dichos cromosomas y la legalización de la progenie generada a partir de la operación *cruzamiento*, esta se encuentra descrita detalladamente a continuación.

Paso 1

En la figura no. 3.3 se muestran dos soluciones que representan dos secuencias validas para un JSSP con 4 máquinas y 4 trabajos, estos serán los padres **p1** y **p2** sometidos a cruzamiento.

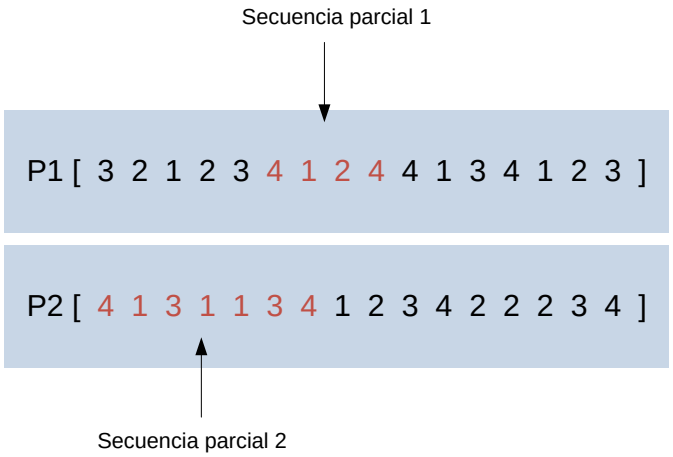

Figura 3.3 Paso 1,2 y 3: Selección de secuencias parciales

Se escoge una secuencia parcial en el primer padre de forma aleatoria. La primera posición en la sub-secuencia es la sexta. En esta posición se encuentra la indicación de una operación del trabajo 4.

Paso 2

Se encuentra el siguiente gen que represente al trabajo número 4 ,el cual se encuentra en la posición no. 9. Así, la sub-secuencia se compone de (4 1 2 4). Una secuencia parcial está identificada con el mismo trabajo en la primera y última posición del mismo segmento.

Paso 3

Se localiza la subsecuencia ahora para el padre **p2** como sigue. Esta será la primera cadena de caracteres localizada en el cromosoma **p2** que empiece y termine con el trabajo no. 4. Esta secuencia parcial 2 estaría compuesta por lo tanto de (4 1 3 1 1 3 4). Esta subsecuencia se muestra en la figura 3.3.

Paso no 4.

Realizar el intercambio de las dos secuencias parciales (4 1 2 4) y (4 1 3 1 1 3 4) para generar los nuevos individuos **o1** y **o2** como se muestra en la figura 3.4.

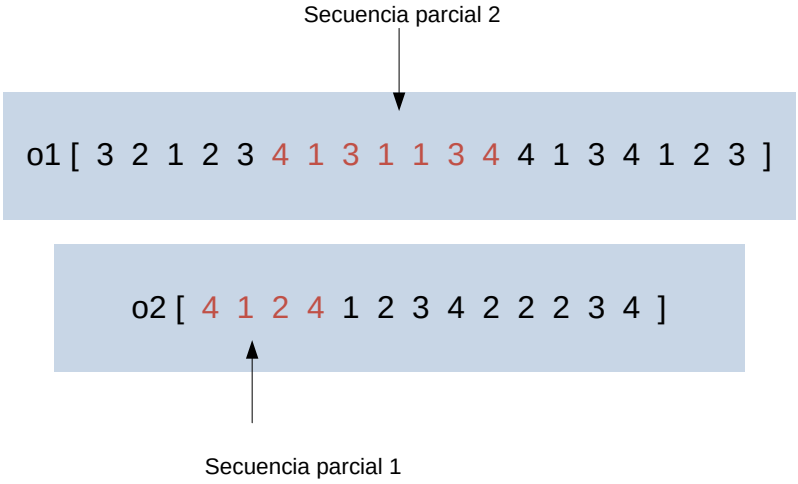

Figura 3.4 Paso 4: Intercambio de secuencias parciales

Paso no 5.

Generalmente las dos secuencias parciales encontradas tendrán un número diferente de genes, por lo tanto, los individuos generados inmediatamente después del cruce serán soluciones ilegales. Un individuo ilegal puede tener genes en exceso o carecer de algunos. Los genes faltantes y excedentes se muestran en la figura no. 3.5. El siguiente paso por tanto será reparar cada individuo borrando los genes en exceso o añadiendo los faltantes para así validar las nuevas soluciones.

En el ejemplo, a causa del *cruzamiento*, el nuevo individuo o1 ganó los genes extra 3, 1, 1 y 3, mientras ha perdido un gen 2. Por lo tanto los genes (3 1 1 y 3) tendrán que ser borrados y el gen faltante 2 tendrá que ser insertado en o1 (en la posición inmediatamente posterior a la nueva secuencia parcial) para así legalizar el nuevo individuo. Esta reparación se repite entonces con o2 para también legalizarlo.

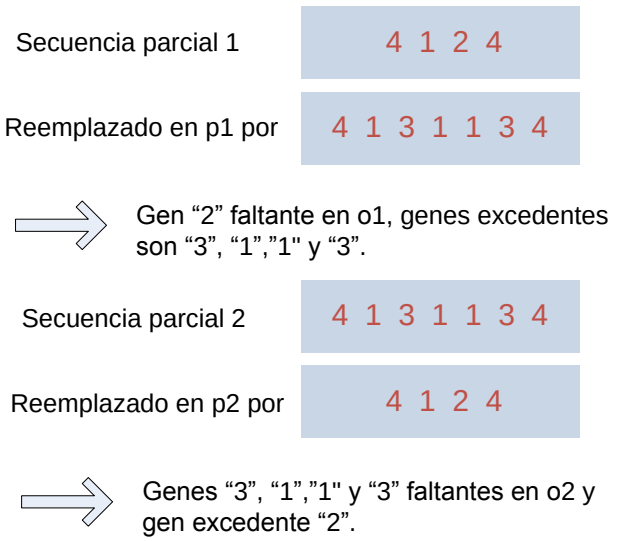

Figura 3.5 Paso 5: Identificación de genes excedentes y faltantes

Borrado aleatorio de los genes excedentes “3”,”1",”1" y “3” sin disturbar la secuencia parcial insertada.

o1 [**3** 2 **1** 2 **3** 4 1 3 1 1 3 4 4 **1** 3 4 1 2 3]

o1 [2 2 4 1 3 1 1 3 4 4 3 4 1 2 3]

Inserción de el gen faltante “2” al término de la secuencia parcial.

o1 [2 2 4 1 3 1 1 3 4 **2** 4 3 4 1 2 3]

Figura 3.6 Paso 5: Validación del nuevo individuo **o1**

Este operador se programó en MATHLAB, realizando las operaciones que anteriormente se describen. En la interfaz de MATHLAB especializada en Algoritmos Genéticos, se hace referencia a esta función que se creó como un fichero M o M-File, el cual es un bloque que contiene alguna función programada en MATHLAB por es usuario y puede ser llamada por la herramienta de *Optimtool.*

Operador *Mutación*

Una vez que son seleccionados los individuos que han de someterse a este operador, se genera un número aleatorio entre 0 y 1, el cual decidirá si es que el individuo mutará alguno de sus elementos cuando el número es menor a 0.1 o pasará idéntico a la siguiente generación en el caso contrario. Esto se debe a que el operador mutación no debe de alterar de manera definitiva a un porcentaje de la población en todas las ocasiones. Es solo en un bajo porcentaje que ocasionalmente algún individuo es sujeto al

intercambio de dos de sus elementos para generar un nuevo individuo que puede explorar un nuevo rango en el espacio solución del problema.

El operador para la mutación utilizado para este tipo de codificación es del tipo *intercambio de un par de trabajos (job pair exchange)* o *inversión de un bit,* esto es que dos trabajos son escogidos aleatoriamente para después cambiarlos de posición como se muestra en la figura 3.7. El resultado de esta operación siempre resultará en un nuevo individuo válido, por lo tanto no se requiere de alguna reparación después de la mutación.

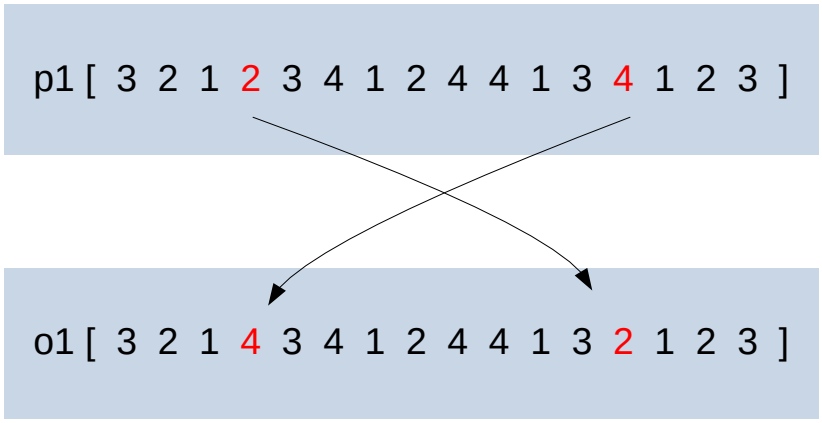

Figura 3.7 Mutación por intercambio de un par de trabajos.

Esta función también se creó como un fichero M y se muestra a continuación la programación, a diferencia del Fichero M de la función cruzamiento o el de la función de evaluación, es pequeño, por eso se muestra a continuación.

FICHERO M CON LA FUNCIÓN MUTACIÓN

```
function mutationChildren =
mymutfuncb1(parents,options,GenomeLength,FitnessFcn,state,thisScore,thisPopulation,mutation
Rate)
 if(strcmpi(options.PopulationType,'doubleVector'))
  mutationChildren = zeros(length(parents),GenomeLength);
  parents
      for i=1:length(parents)
       child = thisPopulation(parents(i),:);
       t=rand
       if t>.8

       mutationPoints=[0 0];
              while mutationPoints(1,1)==mutationPoints(1,2)
                    for j=1:2
                    m=rand(1,GenomeLength);
                    mutationPoints(1,j)=find(m==max(m));
                    end
              end

      a=child(1,mutationPoints(1,1));
      b=child(1,mutationPoints(1,2));
      child(1,mutationPoints(1,1))=b;
      child(1,mutationPoints(1,2))=a;

       mutationChildren(i,:) = child;
        else
mutationChildren(i,:) = child;
thisPopulation;
       end
      end
     end
 end
```

Función Generación de población inicial

La generación de la población inicial se tiene que programar debido a la particularidad de los vectores utilizados.

La herramienta Optimtool proporciona un campo para que el usuario pueda especificar el número de población para cada generación, así como otro campo para introducir una

población inicial si se tiene. Este proceso solo se realiza para la primera generación, ya que para la segunda, tercera y *n* generaciones en adelante, la población se compondrá de los individuos seleccionados para sobrevivir de una generación, de los individuos resultado del cruzamiento y de los individuos generados por la mutación de algún elemento.

Para generar los primeros individuos se introdujo un vector con *n "1"s, n "2"s, n "3"s, n "4"s… m "no. de trabajos", donde n es igual al número de máquinas*. Este vector se reacomodó de forma aleatoria generando una solución factible para cada iteración hasta completar la población inicial, de la cual se partirá para formar las siguientes generaciones.

Función Evaluación

De calificar que tan buena o mala es una solución se encarga la *función Evaluación.* La probabilidad de pasar a la siguiente generación intacta como parte de una población "Elite", de generar nuevas soluciones mediante el operador cruzamiento o del operador mutación, es directamente proporcional a la calificación asignada por la función evaluación a cada individuo. Esto no quiere decir que un individuo con una buena calificación no pueda de una generación a otra ser eliminado o no ser considerado para generar nuevos individuos.

La *función evaluación* será programada para evaluar qué individuo genera la secuencia de actividades con el menor *tiempo acumulado de ejecución de las máquinas o intervalo operativo* (*Makespan*), el cual se define como:

$$Cmáx = máx\{C_1, C_2, C_3, \cdots C_{n-1}, C_n\}$$ *Donde:*

C_i= tiempo de terminación del trabajo i

i=1,2,3…n-1,n

n= número de trabajos que serán procesados

Esta medida o parámetro es comúnmente monitoreado por su implicación en los costos de producción de casi cualquier industria, es por esto que se buscará minimizarlo.

La forma en que se programará la evaluación del *intervalo operativo* para una solución se describe a continuación:

Al programa se han introducido de inicio una matriz de *nXm*, llamada L, donde *n* representa el número de tareas que será variable para las diferentes simulaciones y *m* es igual a 4. De esta forma, un renglón o individuo de esta matriz, se compone de *m* columnas, las cuales indican en forma consecutiva, el número de trabajo al que pertenece la tarea, el orden en el que tienen que ser procesadas indicado por una secuencia de números enteros que van desde el 1 hasta el 7 en el caso de una operación con el número máximo de tareas, la indicación de la máquina donde será procesada, y en la cuarta y última columna el tiempo que se supone tomara el llevar a cabo la tarea.

Con el vector solución en turno se realiza una búsqueda en cada uno de sus elementos, a la vez que se busca en la matriz que contiene la información de todas las operaciones (en qué centro de trabajo se realizaran y el tiempo de operación). Una vez identificada la operación, esta se traslada a una nueva matriz que le denominamos **L**, esta matriz contendrá la misma información que la matriz **X** pero en el orden que dicta el vector solución. Una vez teniendo esta matriz, se realiza una iteración en cada uno de sus elementos. En esta búsqueda se extraerá la información de a que máquina se asignará la operación correspondiente y la información del tiempo de operación. Tomando en cuenta las precedencias entre las operaciones, y la disponibilidad en las máquinas a las que se han asignado los trabajos, se asigna un tiempo de inicio para cada operación y se suma este tiempo a la carga de trabajo de cada máquina. Esto representado en un vector que

contiene en cada uno de sus elementos el tiempo de terminación de trabajo en las máquinas, este es el vector M=[mq_1, mq_2, mq_3 … mq_n].

Finalmente el intervalo operativo será igual al valor más alto de entre los elementos del vector M, y el número de renglón de este elemento indicará en que máquina se realizará dicho trabajo. Este proceso se muestra en forma gráfica en la figura no. 3.8.

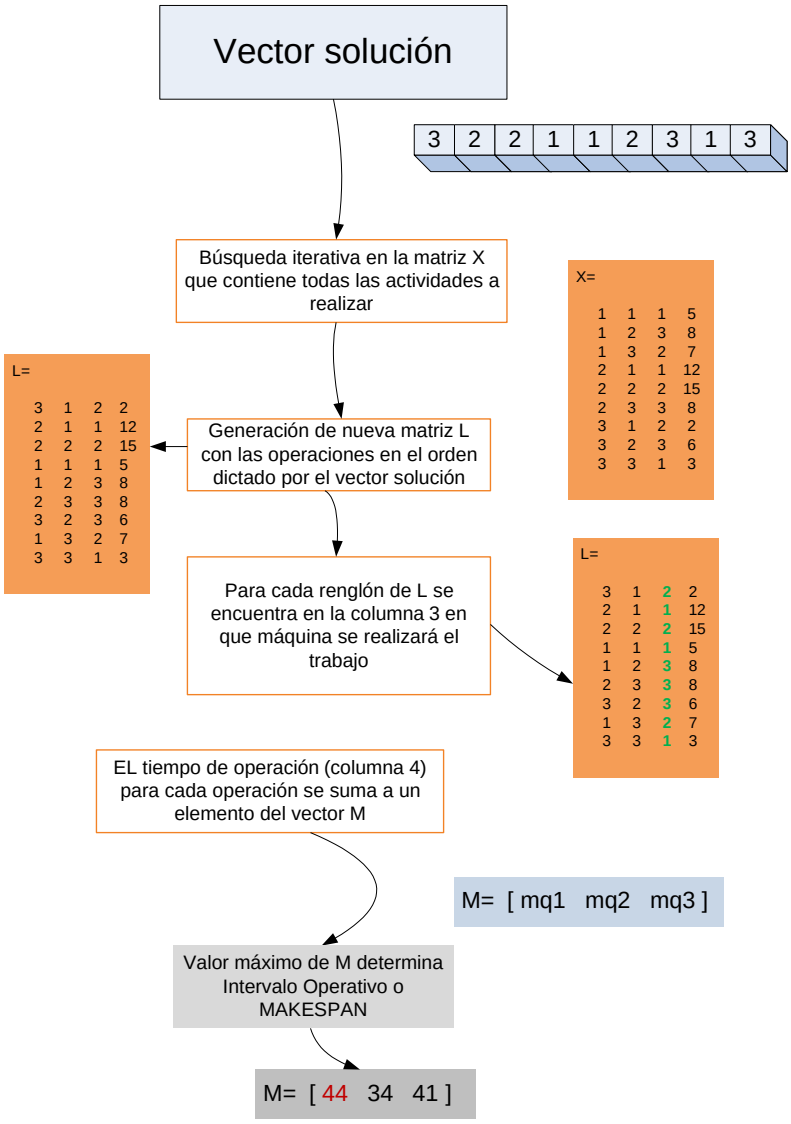

FIG 3.8 Representación de la función para evaluar el MAKESPAN

Función escalamiento

En seguida se define la función que acomodará los individuos en una escala apta para que la función de selección escoja a los que formaran la siguiente generación.

Existen opciones para esta función.

La función *proporcional* asigna una expectativa a cada uno de los individuos en proporción a su valor arrojado por la función evaluación. Esta técnica tiene debilidades si es que los valores de los individuos no se encuentran en un rango de valores amplio, de esta forma los ejemplares más aptos pueden fácilmente ser desechados y contrariamente algunos de los menos aptos, ser escogidos para formar la siguiente generación. No siendo el caso el amplio rango de valores que arroja la función evaluación que se introdujo en el algoritmo, se escogió esta función.

Selección de individuos para producir la nueva generación

Los individuos para una nueva generación serán resultado de tres tipos de procesos. El primero y más sencillo, pero no menos importante, es la selección de los individuos mejor adaptados y que sobrevivirán a la generación anterior para pasar a la nueva generación sin alguna alteración, al número de individuos de este tipo se le denomina como el numero de *Población Elite*, y es un parámetro que es posible modificar en el *Optimtool* para evaluar qué repercusiones tiene en el comportamiento general del algoritmo.
El segundo tipo de población para la nueva generación es la población resultado del proceso de *cruzamiento*. Este parámetro también influirá de forma importante sobre el rendimiento del algoritmo, y debido a que es el operador más importante para el AG se definirá de forma práctica efectuando ensayos con diferentes porcentajes. El porcentaje

restante definirá el tamaño de la población que se someterá a la mutación con alguna probabilidad de no ser alterado en ninguna forma.

La selección de los individuos que se someterán al cruzamiento y a la mutación, puede ser realizada mediante el proceso que Goldberg utilizó para su AG simple o SGA, que es el del método de selección de ruleta, también se puede utilizar el método de Torneo, o el método de selección Proporcional.
Esta función utiliza los valores escalados que arroja la anterior función de escalamiento. Existen 5 opciones prediseñadas para este propósito, selección estocástica uniforme, selección por remanente, selección uniforme, selección por ruleta, y selección por torneo. Para la simulación se decidió utilizar la selección por ruleta. Esta favorece la selección de los individuos más fuertes, sin dejar de lado la posibilidad de que se escojan algunos del menor rango, permitiendo así la diversidad a la población de soluciones.

Reproducción

Los parámetros para la reproducción permiten decidir si es que existirá una población "elite", que sobreviva a la anterior y si es que la habrá, determinar su tamaño. También se determina el porcentaje de la población que se generará por medio del operador cruzamiento. Aunque no se especifica en la ventana, el resto de la población automáticamente se generará por medio de la función mutación. En la figura 3.9 se muestra el ejemplo de una población elite de 2 individuos, que es el valor preestablecido que maneja *Mathlab*, y un porcentaje de cruzamiento de 80%.

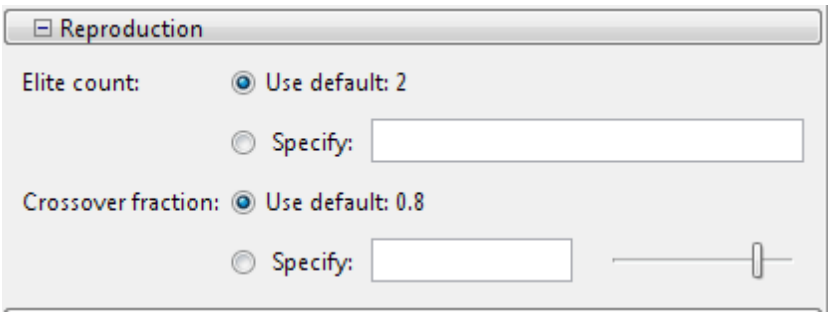

Figura 3.9 Ventana Mathlab Reproducción

Migración

La opción que nos permite la migración entre individuos de varios subgrupos dentro de la población permite evitar que el algoritmo se aloje en un óptimo local y no continúe la búsqueda de una mejor solución. Esta herramienta maneja una migración entre sub-poblaciones del tipo pasarela (*Stepping Stone*), para la cual se intercambian individuos solo con subgrupos vecinos, en nuestro caso en ambas direcciones , es decir, de la población *n* se migra en ambos sentidos, tanto a la sub-población *(n–1)* como a la (*n*+1). En la ventana se especifica que porcentaje de los mejores resultados migran y con qué frecuencia.

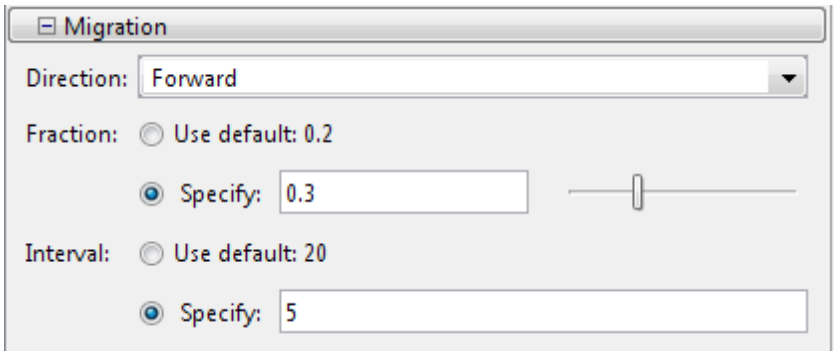

Figura 3.10, Ventana Mathlab Migración

Criterios de parada para el algoritmo

Algunos de los criterios de paro para el algoritmo que permite utilizar la herramienta *Optimtool*, se basan en el número de generaciones a crear, el tiempo de corrida del algoritmo, un límite si se desea para el parámetro que se busca optimizar, un número de generaciones o un tiempo límite en que el parámetro permanezca sin mejoras. Para la simulación actual se utilizaron como criterios de paro solo el número de generaciones a crear, siendo éste variable en diferentes situaciones dependiendo del tiempo de simulación deseado.