

**Comparación entre las técnicas metaheurísticas y los algoritmos de estimación  
de distribución en procesos estocásticos**

**Carlos Alberto Ramírez Vanegas  
Código. 93299684**

**Director:**

**Gerardo Cardona**

**Universidad Tecnológica de Pereira  
Facultad Ciencias Básicas  
2018**

**Comparación entre las técnicas metaheurísticas y los algoritmos de estimación  
de distribución en procesos estocásticos**

**Carlos Alberto Ramírez Vanegas  
Código. 93299684**

**Tesis presentada como requisito, para optar por el título de magister en  
matemáticas**

**Director:  
Gerardo Cardona**

**Universidad Tecnológica de Pereira  
Facultad ciencias básicas  
2018**

## AGRADECIMIENTOS

Al profesor Gerardo por ayudarme a culminar este último paso en la maestría y brindarme la oportunidad de trabajar con él en este documento, a los profesores de la maestría, a mis compañeros de la primera cohorte de la maestría. A la dirección del departamento de matemáticas por su apoyo durante este proceso.

## Dedicatoria

A Oro, a Mister Pompei mi gran apoyo e impulso

## INDICE GENERAL

INTRODUCCIÓN .....	6
1. PROBLEMA DE INVESTIGACIÓN .....	6
1.1 Definición del problema.....	6
1.2 Objetivo general: .....	8
1.3 Objetivos específicos: .....	8
2. MARCO DE REFERENCIA.....	8
2.1 Técnicas metaheurísticas.....	9
2.1.1 Algoritmo genético .....	11
2.1.2 Recocido simulado .....	20
2.2 Algoritmo de estimación de distribución .....	32
3. ANÁLISIS DE RESULTADOS.....	45
4. CONCLUSIONES Y RECOMENDACIONES .....	46
5. RESULTADOS EN EVENTOS ACADÉMICOS .....	46
6. TRABAJOS FUTUROS.....	47
7. BIBLIOGRAFÍA.....	47
8. Anexo.....	49

## INTRODUCCIÓN

Determinar una u otra técnica en problemas de optimización hace la diferencia al encontrar la solución a tales problemas, máxime cuando la técnica no garantiza el óptimo global, es por ello que se desea contribuir a la determinación en la elección de las diferentes técnicas existentes. Además, se presentará un panorama más matemático y menos ingenieril al presentar las técnicas tal como la concibieron sus inventores y tal como se presentan en la actualidad desnaturalizando la idea original, ante este panorama tan amplio de técnicas heurísticas se presentarán técnicas que corran menos riesgo de selección de parámetros reduciendo así la subjetividad de la técnica. Tal es el caso de la estimación por distribución la cual permite de maneja natural, encontrar la solución del problema original.

Para ello se utiliza las ventajas de los procesos estocásticos, permitiéndole a la técnica la incorporación de variables estocásticas para poder modelar el problema de manera más real.

Existe en nuestros tiempos formulación de problemas combinatoriales los cuales se presenta de manera habitual por el gran número de variables que se consideran. Tal es el caso del problema del agente viajero. Por ello el análisis numérico en las últimas décadas ha recibido gran acogida en el ámbito académico, es así como recientes técnicas basadas en procesos estocásticos y en heurísticas se acercan de manera parcial a la solución garantizando soluciones locales de buena calidad.

### 1. PROBLEMA DE INVESTIGACIÓN

#### 1.1 Definición del problema

Los problemas matemáticos en el área de la optimización tienen cada día mayor interés debido al gran volumen de datos con los que se trabajan, el creciente número de variables o el aumento de las restricciones hacen que los problemas cada día más reales dejando de lado la idealización de los mismo, pero el resultado de una mayor complejidad en el momento de

resolverlos, este tipo de problemas se apoyan fuertemente en los ordenadores los cuales cada día son más puentes en la ejecución de código que resuelva este tipo de problemas .

Como ejemplo pensemos en el problema de ruteamiento de vehículos (VRP) es una generalización del problema del agente viajero (TSP), donde se determina un conjunto de rutas para m agentes viajeros, las cuales empiezan y terminan en una ciudad depósito.

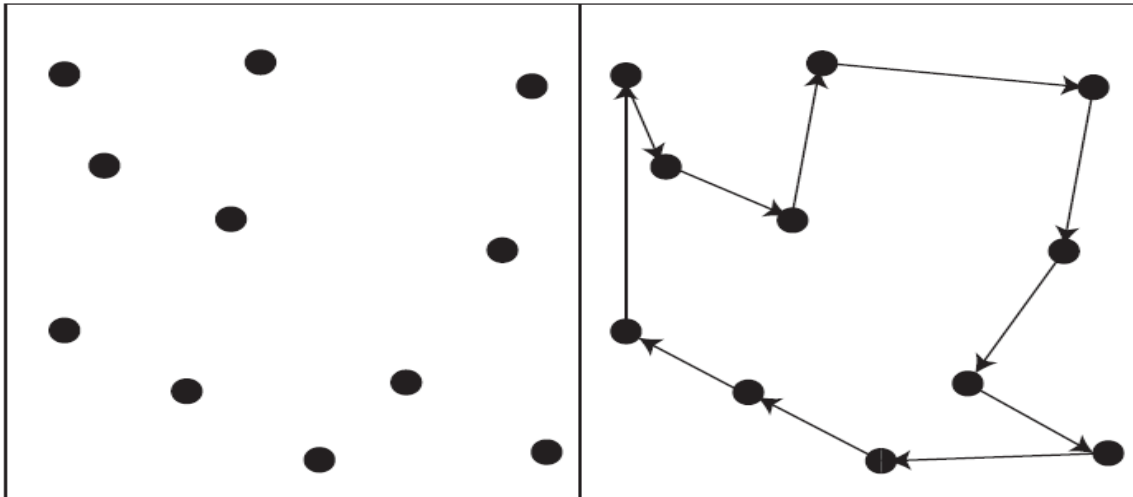


Figura 1. Solución TSP. Fuente: (Olivera, 2004)

Matemáticamente se puede formular de la siguiente manera

$$\begin{aligned} & \text{mín} \sum_{i \in V} \sum_{j \in V} C_{ij} x_{ij} \\ & \text{s.t.} \\ & \sum_{i \in V} x_{ij} = 1, \quad \forall j \in V \\ & \sum_{j \in V} x_{ij} = 1, \quad \forall i \in V \\ & \sum_{i \in S} \sum_{j \in (V \setminus S)} x_{ij} \geq 1, \quad 2 \leq |S| \leq n - 2 \\ & x_{ij} \in \{0, 1\} \quad \forall i, j \in A. \end{aligned}$$

Encontrar la solución que repite las restricciones es no-difícil. Observando la siguiente tabla verificamos lo anterior.

Ciudades	Tiempo
3	6 micro segundos
5	0.12 milisegundos
10	3.63 segundos
20	771000 años
30	2.6525E+26 siglos

Tabla 1. Ejemplo. Fuente: Elaboración propia.

Por tal motivo es necesario encontrar matemáticamente técnicas que mejoren la respuesta, en algunos casos la solución será convergente y en algunos otros se obtendrá con un margen de error, pero lo suficientemente buena como para aceptarla.

### 1.2 Objetivo general:

Comparar las técnicas metaheurísticas con técnicas estocásticas, a fin de destacar las ventajas y desventajas de cada una.

### 1.3 Objetivos específicos:

- Revisar el estado del arte alrededor de las técnicas metaheurísticas.
- Revisar el estado del arte alrededor de las técnicas estocásticas basadas en análisis bayesiano.
- Destacar ventajas y desventajas de las técnicas metaheurísticas y bayesianas.

## 2. MARCO DE REFERENCIA

Existen diversas técnicas que se pueden implementar para solucionar problemas que contienen la maldición del dimensionamiento entre ellas se encuentran las técnicas exactas y los métodos aproximados tal como se ve en la siguiente figura, de dichas posibilidades para problemas NP (no polinomiales) no es posible encontrar una solución de manera analítica, por eso es necesario el estudio de las meta heurísticas y de híbridos entre ellas. Tal como se detalla a continuación (Olivera, 2004):



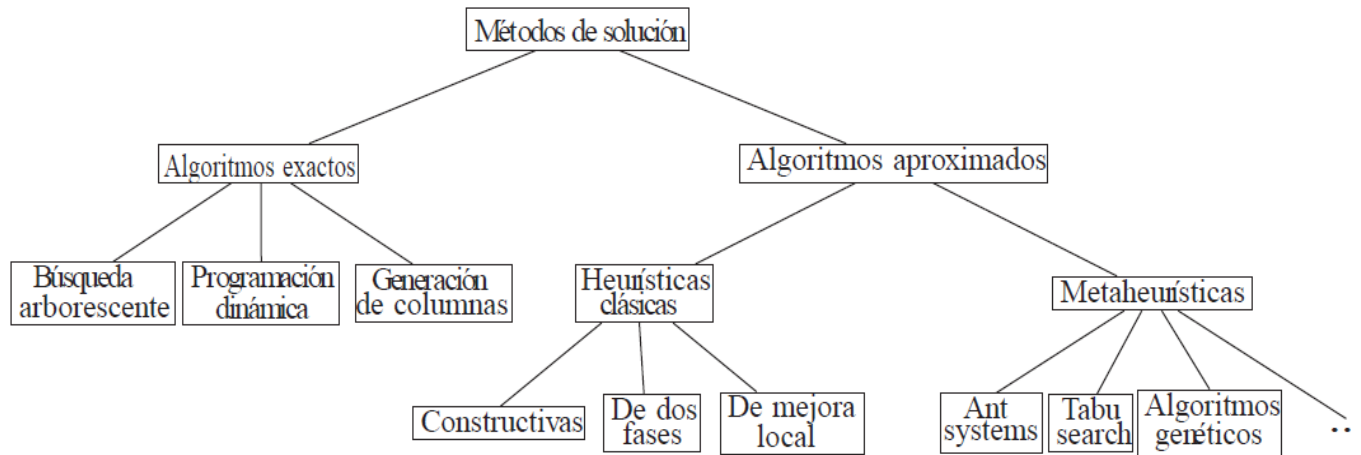


Figura 2. Métodos de solución del VRP. Fuente: (Olivera, 2004)

## 2.1 Técnicas metaheurísticas

Las técnicas metaheurísticas son nuevas herramientas que han sido desarrolladas en los últimos años para la resolución de diferentes tipos de problemas que antes no habían podido ser abordados debido a su complejidad. Consisten en algoritmos que proporcionan soluciones factibles (admisibles), que, aunque no alcancen el óptimo, al menos se acercan a su valor en un tiempo de cálculo razonable. Aunque en principio parezca lo contrario, en la mayoría de los casos nos encontramos con procedimientos muy simples, muchas veces basados únicamente en el sentido común. Son varias las situaciones en las que resulta muy recomendable la utilización de este tipo de algoritmos:

Cuando no existe un método exacto de resolución o éste requiere mucho tiempo de cálculo o memoria. Cuando no se necesita una solución óptima, sino una solución satisfactoria que oriente sobre el comportamiento del problema. Cuando los datos son poco fiables. Cuando las limitaciones de espacio, tiempo, etc., obliguen al empleo de métodos de respuesta rápida.

Como paso intermedio en la aplicación de otro algoritmo. El uso de técnicas metaheurísticas como método de resolución de problemas de optimización está, a día de hoy, muy extendido, debido entre otros, a los siguientes motivos:

- Suelen ser más fáciles de entender por parte de gente no experta el entendimiento de las heurísticas que los complejos métodos que utilizan la mayoría de las técnicas exactas.
- Mayor flexibilidad para el manejo de las características del problema.
- No suelen ser complejas incluso para funciones no lineales.

No obstante, estas técnicas también presentan inconvenientes. Uno de los más destacados es que, en la mayoría de los casos en los que estos algoritmos son utilizados, no se conoce el valor del óptimo, y dado que son procedimientos en los que no se asegura la optimalidad, no se puede tener una idea clara del nivel de calidad de la solución obtenida. Son muchas las metaheurísticas creadas hasta el día de hoy. Unas, tienen su uso limitado a problemas particulares, mientras que otras pueden ser aplicadas a problemas más diversos. Entre estas últimas podríamos destacar las siguientes como las más importantes:

- Recocido simulado.
- Algoritmos genéticos.
- GRASP.
- Búsqueda Tabú (Rosales Bello, 2005).

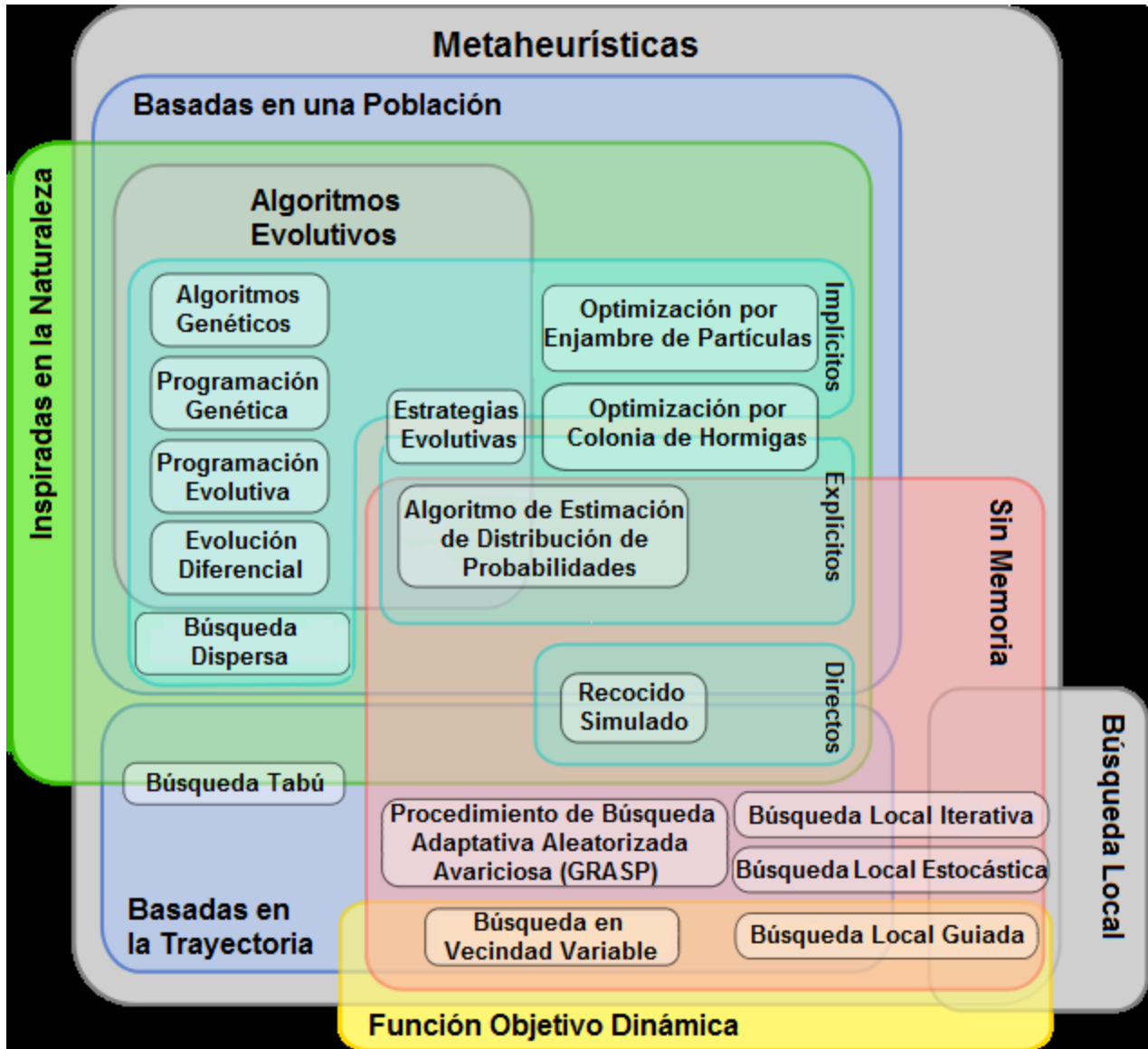


Figura 3. Resumen de las metaheurísticas. Fuente: Wikipedia

### 2.1.1 Algoritmo genético

Los Algoritmos Genéticos, algoritmos basados en poblaciones, se han utilizado en diversos problemas de programación complejos. Estos algoritmos se combinan dos procesos principales con el fin de encontrar mejores soluciones: la selección y la variación. Los operadores de cruce y mutación pertenecen al proceso de variación. El operador de cruce recombina soluciones parciales mediante la selección de los individuos que representan soluciones factibles del problema y la mutación hace cambios aleatorios en los descendientes que también son soluciones

factibles del problema y que son el resultado de la variación. Sin embargo, a veces el proceso de variación puede distorsionar las soluciones y no tiene efectos positivos sobre la aptitud de supervivencia de cada individuo (Perez Rodriguez & Hernandez Aguirre, 2015).

En la siguiente figura es posible observar el esquema general del funcionamiento de un algoritmo genético:

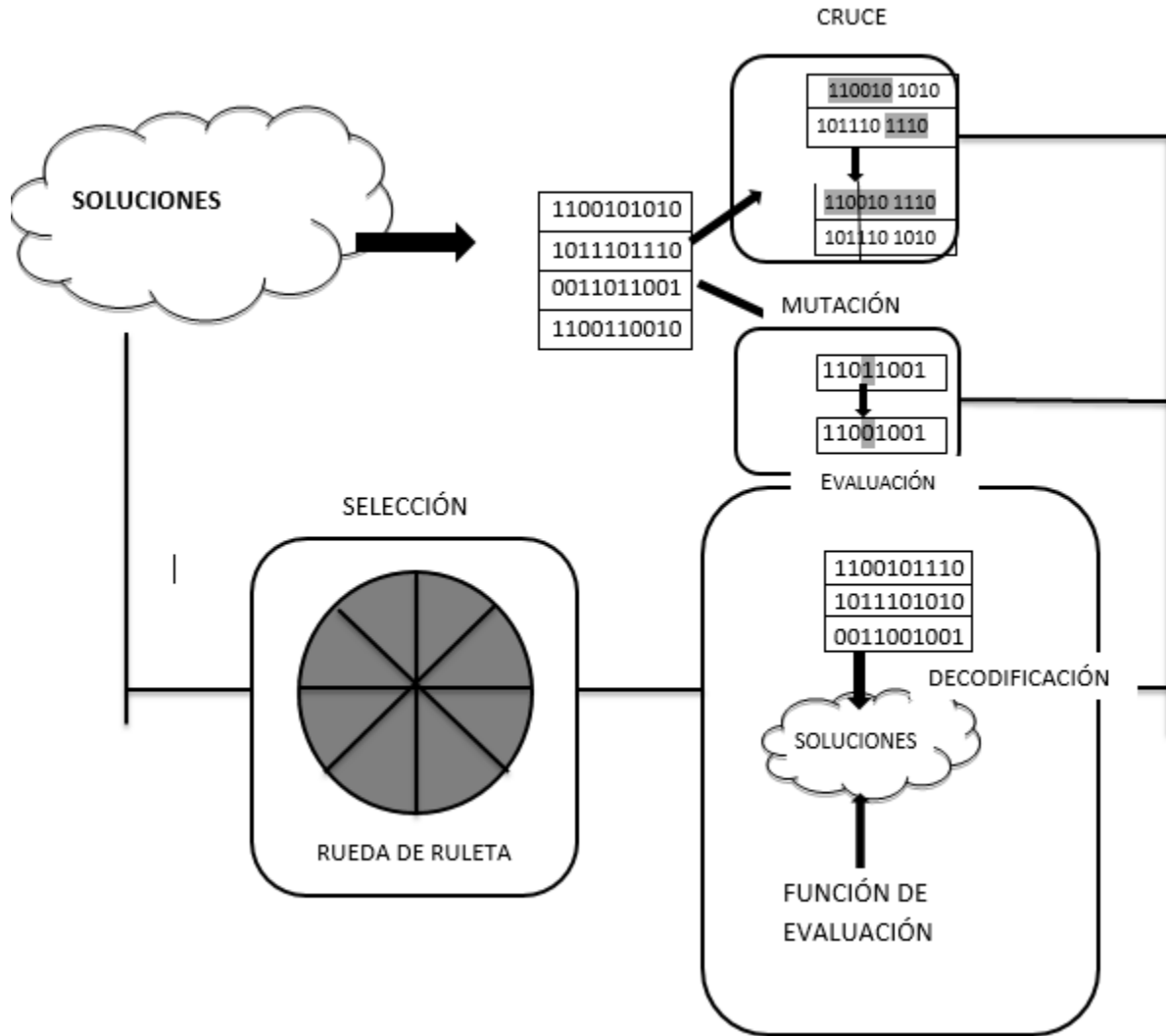


Figura 4. Esquema general AG. Fuente: (Perez Rodriguez & Hernandez Aguirre, 2015)

## **Población inicial**

Una población es una representación abstracta de un segmento o conjunto de individuos. Para el caso específico de los AG, existe una serie de poblaciones que se van generando en la medida en que el algoritmo se ejecuta. La población inicial es aquella que se toma como base para iniciar el proceso evolutivo. Con el paso del tiempo, se espera que los individuos de las nuevas generaciones tengan genes o características de los mejores individuos de las poblaciones anteriores (Gamboa Salgado & Gomez Marulanda, 2012).

## **Evaluación**

La evaluación es el proceso que permite determinar el valor de un individuo dada una función de comparación denominada función de evaluación.

La definición acertada de la función de evaluación es uno de los elementos cruciales en el comportamiento de los AG. Idealmente interesa construir funciones objetivo con “ciertas regularidades” en cuanto a su comportamiento, es decir, funciones objetivo que verifiquen que para dos individuos que se encuentren cercanos en el espacio de búsqueda, sus respectivos valores en las funciones objetivo sean similares.

Un problema habitual en las ejecuciones de los AG surge debido a la velocidad con la que el algoritmo converge, es decir, el tiempo que tarda el algoritmo en estabilizar los valores de evaluación. En algunos casos la convergencia es muy rápida, lo que suele denominarse convergencia prematura, en la cual el algoritmo converge hacia óptimos locales, mientras que en otros casos el problema es justo el contrario, es decir se produce una convergencia lenta del algoritmo. Una posible solución a estos problemas pasa por efectuar transformaciones en la función objetivo (Gamboa Salgado & Gomez Marulanda, 2012).

## **Selección**

La selección es la encargada de transmitir y conservar aquellas características de las soluciones que se consideran valiosas a lo largo de las generaciones. Existen diversos métodos de selección dentro de los más usados están la selección por ruleta y por torneo. En la ruleta, la probabilidad

que tiene un individuo de 30 reproducirse es proporcional a su valor de función de evaluación, es decir, a su adaptación (Gamboa Salgado & Gomez Marulanda, 2012).

Puesto que se trata de imitar lo que ocurre en la naturaleza, se ha de otorgar un mayor número de oportunidades de reproducción a los individuos más aptos. Por lo tanto, la selección de un individuo estará relacionada con su valor de ajuste. No se debe, sin embargo, eliminar por completo las opciones de reproducción de los individuos menos aptos, pues en pocas generaciones la población se volvería homogénea (Villar Ledo & Alonso Roque, 2013).

En cuanto a algoritmos de selección se refiere, estos pueden ser divididos en dos grandes grupos: probabilísticos y determinísticos. Ambos tipos de algoritmos basan su funcionamiento en el principio indicado anteriormente (permitir escoger una mayor cantidad de veces a los más aptos). Sin embargo, como su nombre indica, el primer tipo adjudica estas posibilidades con un importante componente basado en el azar. Es en este grupo donde se encuentran los algoritmos de selección por ruleta o por torneo que, dado su importancia por ser los más frecuentemente utilizados, se describen con detalle en esta sección. El segundo grupo engloba una serie de algoritmos que, dado el ajuste conocido de cada individuo, permite asignar a cada uno el número de veces que será escogido para reproducirse. Esto puede evitar problemas de predominancia de ciertos individuos y cada uno de estos algoritmos presentan variaciones respecto al número de veces que se tomarán los mejores y peores y, de esta forma, se impondrá una presión en la búsqueda en el espacio de estados en la zona donde se encuentra el mejor individuo (en el caso de que se seleccionen más veces los mejores), o bien que se tienda a repartir la búsqueda por el espacio de estados, pero sin dejar de tender a buscar en la mejor zona (Gestal, Rivero, Rabuñal, Dorado, & Pazos, 2010).

#### - Selección por ruleta

Propuesto por DeJong, es posiblemente el método más utilizado desde los orígenes de los Algoritmos Genéticos (Blickle & Thiele, 1995). A cada uno de los individuos de la población se le asigna una parte proporcional a su ajuste de una ruleta, de tal forma que la suma de todos los porcentajes sea la unidad. Los mejores individuos recibirán una porción de la ruleta mayor que la recibida por los peores. Generalmente, la población está ordenada en base al ajuste, por lo que las

porciones más grandes se encuentran al inicio de la ruleta. Para seleccionar un individuo basta con generar un número aleatorio del intervalo  $[0..1]$  y devolver el individuo situado en esa posición de la ruleta. Esta posición se suele obtener recorriendo los individuos de la población y acumulando sus proporciones de ruleta hasta que la suma exceda el valor obtenido. Es un método muy sencillo pero ineficiente a medida que aumenta el tamaño de la población (su complejidad es  $O(n^2)$ ). Presenta además el inconveniente de que el peor individuo puede ser seleccionado más de una vez (Gestal, Rivero, Rabuñal, Dorado, & Pazos, 2010).

#### - Selección por torneo

La idea principal de este método de selección consiste en escoger a los individuos genéticos en base a comparaciones directas entre sus genotipos.

Existen dos versiones de selección mediante torneo, el torneo determinístico y el torneo probabilístico, que a continuación pasan a detallarse.

En la versión determinística se selecciona al azar un número  $p$  de individuos (generalmente se escoge  $p=2$ ). De entre los individuos seleccionados se selecciona el más apto para pasarlo a la siguiente generación.

La versión probabilística únicamente se diferencia en el paso de selección del ganador del torneo. En vez de escoger siempre el mejor se genera un número aleatorio del intervalo  $[0..1]$ , si es mayor que un parámetro  $p$  (fijado para todo el proceso evolutivo) se escoge el individuo más alto y en caso contrario el menos apto. Generalmente  $p$  toma valores en el rango  $0.5 < p \leq 1$ .

Variando el número de individuos que participan en cada torneo se puede modificar la presión de selección. Cuando participan muchos individuos en cada torneo, la presión de selección es elevada y los peores individuos apenas tienen oportunidades de reproducción. Un caso particular es el elitismo global. Se trata de un torneo en el que participan todos los individuos de la población, con lo cual la selección se vuelve totalmente determinística. Cuando el tamaño del torneo es reducido, la presión de selección disminuye y los peores individuos tienen más oportunidades de ser seleccionados.

Elegir uno u otro método de selección determinará la estrategia de búsqueda del Algoritmo Genético. Si se opta por un método con una alta presión de selección se centra la búsqueda de las soluciones en un entorno próximo a las mejores soluciones actuales. Por el contrario, optando por

una presión de selección menor se deja el camino abierto para la exploración de nuevas regiones del espacio de búsqueda.

Existen muchos otros algoritmos de selección. Unos buscan mejorar la eficiencia computacional, otros el número de veces que los mejores o peores individuos pueden ser seleccionados. Algunos de estos algoritmos son muestreo determinístico, escalamiento sigma, selección por jerarquías, estado uniforme, sobrante estocástico, brecha generacional, etc. (Gestal, Rivero, Rabuñal, Dorado, & Pazos, 2010).

### **Cruce**

El operador de cruce permite realizar una exploración de toda la información almacenada hasta el momento en la población y combinarla para crear mejores individuos.

Es una buena idea que, tanto la codificación como la técnica de cruce, se hagan de manera que las características buenas se hereden o al menos, no sea mucho peor que el peor de los padres.

Para lo anterior no se tiene un modo formal para seleccionar la técnica de mejor adaptación a un problema específico, sin embargo es recomendable analizar la naturaleza de la cadena y los genes del individuo, con el fin de conocer que elementos se pueden cruzar para lograr la creación de un individuo que combine las mejores características de los padres (Gamboa Salgado & Gomez Marulanda, 2012).

### **Mutación**

La mutación se considera un operador básico, que proporciona un pequeño elemento de aleatoriedad en el entorno de los individuos de la población mediante la alteración de uno o más genes con una probabilidad igual a la tasa de mutación. Si bien se admite que el operador de cruce es el responsable de efectuar la búsqueda a lo largo del espacio de posibles soluciones (exploración), también parece desprenderse de los experimentos efectuados por varios investigadores que el operador de mutación realiza una explotación de la población y va ganando en importancia a medida que la población de individuos va convergiendo. El objetivo del operador de mutación es producir nuevas soluciones a partir de la modificación de un cierto número de genes de una solución existente, con la intención de fomentar la variabilidad dentro de la población. Existen diversas formas de realizar la mutación, desde la más sencilla, donde cada gen muta aleatoriamente con independencia del resto de genes, hasta configuraciones más



complejas donde se tienen en cuenta la estructura del problema y la relación entre los distintos genes (Gamboa Salgado & Gomez Marulanda, 2012).

### **Una nueva población**

Cada iteración del algoritmo arroja una nueva población en la que se espera encontrar individuos cada vez mejor adaptados al problema planteado (Gamboa Salgado & Gomez Marulanda, 2012).

### **Elitismo**

El elitismo es un caso particular del operador de copia consistente en copiar siempre al mejor, o en su caso mejores individuos de una generación en la generación siguiente. De esta manera se garantiza que el proceso de búsqueda nunca dará un paso atrás en cuanto a la calidad de la mejor solución obtenida, sino que un cambio en ésta siempre implicará una mejora. Una variación de este proceso consiste en copiar a los mejor o mejores individuos de una generación en la siguiente, únicamente cuando tras el paso de una generación no se haya mejorado con los operadores de cruce o mutación la mejor solución de la generación actual (Gestal, Rivero, Rabuñal, Dorado, & Pazos, 2010).

### **Evaluación**

Para el correcto funcionamiento de un Algoritmo Genético se debe de poseer un método que indique si los individuos de la población representan o no buenas soluciones al problema planteado. Por lo tanto, para cada tipo de problema que se desee resolver deberá derivarse un nuevo método, al igual que ocurrirá con la propia codificación de los individuos.

De esto se encarga la función de evaluación, que establece una medida numérica de la bondad de una solución. Esta medida recibe el nombre de ajuste. En la naturaleza el ajuste (o adecuación) de un individuo puede considerarse como la probabilidad de que ese individuo sobreviva hasta la edad de reproducción y se reproduzca. Esta probabilidad deberá estar ponderada con el número de individuos de la población genética.

En el mundo de los Algoritmos Genéticos se empleará esta medición para controlar la aplicación de los operadores genéticos. Es decir, permitirá controlar el número de selecciones, cruces, copias y mutaciones llevadas a cabo.

La aproximación más común consiste en crear explícitamente una medida de ajuste para cada individuo de la población. A cada uno de los individuos se le asigna un valor de ajuste escalar por medio de un procedimiento de evaluación bien definido. Tal y como se ha comentado, este procedimiento de evaluación será específico del dominio del problema en el que se aplica el Algoritmo Genético. También puede calcularse el ajuste mediante una manera ‘co-evolutiva’. Por ejemplo, el ajuste de una estrategia de juego se determina aplicando esa estrategia contra la población entera (o en su defecto una muestra) de estrategias de oposición (Gestal Pose, 2010).

A continuación se presentan los resultados del algoritmo genético en el problema de ruteamiento de vehículo:

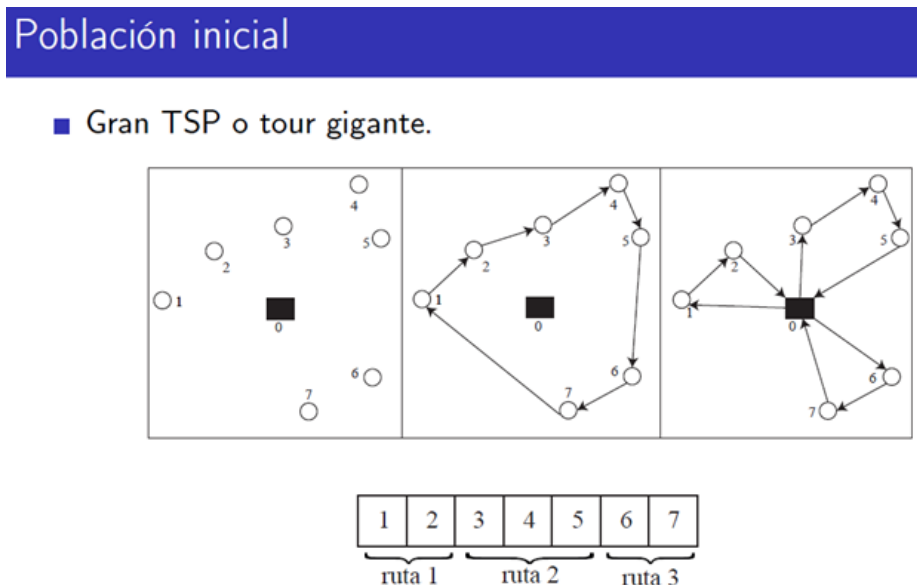


Figura 5. Población inicial. Fuente: Elaboración propia.

## Selección y recombinación

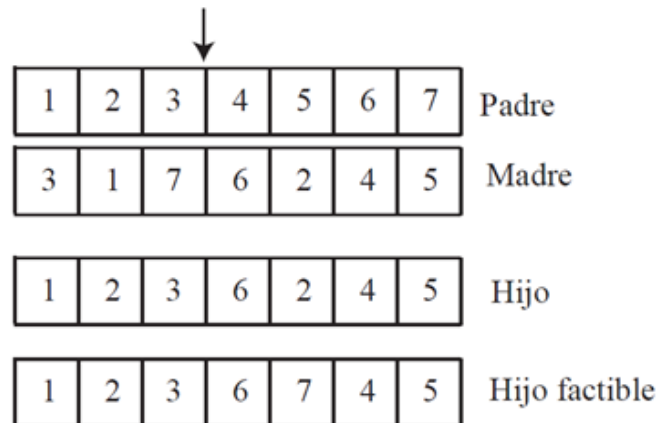


Figura 6. Selección y recombinación. Fuente: Elaboración propia.

## Mutación como etapa de mejoramiento

### ■ Estructura de vecindad 2-OPT

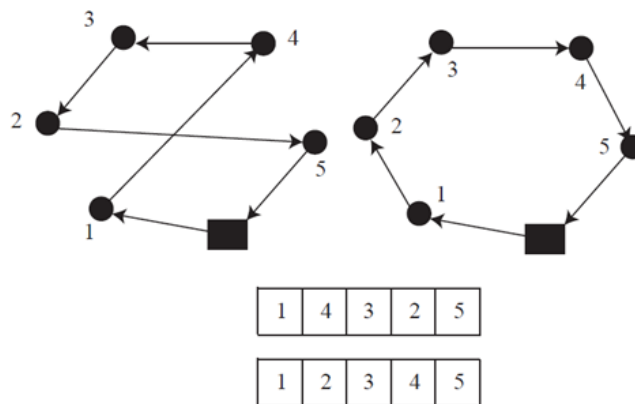


Figura 7. Mutación como etapa de mejoramiento. Fuente: Elaboración propia.

## Resultados

Datos			B&B		AG	
n	Capacidad	rutas	Fobj	T [s]	Fobj	T [s]
13	10	2	34.66	41	34.66	70
76	20	4	—	7200	159331	406

Figura 8. Resultados. Fuente: Elaboración propia.

## Resultados

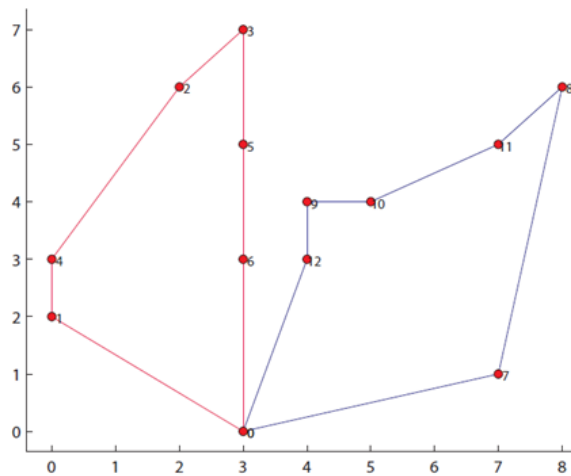


Figura 9. Resultados. Fuente: Elaboración propia.

### 2.1.2 Recocido simulado

Recocido Simulado (Simulated Annealing) es un método de optimización inspirado en el proceso de templado de metales usado desde alrededor de los 5000 años antes de Cristo. El proceso de templado de metales consiste de tres fases: una fase de calentamiento a una temperatura determinada; en la segunda fase se sostiene la temperatura alta lo cual permite a las moléculas acomodarse en estados de mínima energía; y se sigue de una fase de enfriamiento controlado para aumentar el tamaño de sus cristales y reducir sus defectos.

El algoritmo de recocido simulado (SA) es un método iterativo que inicia con un cierto estado  $s$ . Mediante un proceso particular genera un estado vecino  $s'$  al estado actual. Si la energía, o evaluación, del estado  $s'$  es menor que la del estado  $s$ , se cambia el estado  $s$  por  $s'$ . Si la evaluación de  $s'$  es mayor que la de  $s$  entonces se puede empeorar eligiendo  $s'$  en lugar de  $s$  con una cierta probabilidad que depende de las diferencias de las evaluaciones  $\Delta f = f(s) - f(s')$  y de temperatura actual del sistema  $T$ . La posibilidad de elegir un estado peor al actual es lo que le permite a SA salir de óptimos locales para poder llegar a los óptimos globales. La probabilidad de aceptar elegir un peor estado normalmente se calcula por la formula  $P(\Delta f, T) = e^{-\Delta f/T}$  (Departamento de Matemáticas, CSI/ITESM, 2018).

## - Código Recocido Simulado

El siguiente código representa una versión general del algoritmo de recocido simulado:

1.  $s := \text{GeneraUnaSolucionInicial}();$
2.  $T := T_0; g := 0;$
4. mientras ( $\text{CondicionesParoNoActivas}(g, T)$ ) hacer
5.  $s' := \text{TomaUnVecinoAleatorioDe}(s);$
6. si ( $f(s') < f(s)$ )
7.  $s := s'$ ;
8. o bien
9. si ( $\text{Random}(0, 1.0) < \exp((f(s) - f(s'))/T)$ )
10.  $s := s'$ ;
11. fin si;
12. fin si;
13.  $g := g + 1; T := \text{Actualiza}(g, T);$
15. fin mientras

La idea original que dio lugar a la metaheurística denominada Recocido Simulado (RS), se encuentra en el Algoritmo de Metrópolis. Este algoritmo se basa en la técnica de Monte Carlo para generar una secuencia de estados del sólido en su proceso de enfriamiento. Si se tiene un estado actual del sólido  $i$  con energía  $E_i$ , entonces el subsiguiente estado  $j$  con energía  $E_j$  es generado aplicando un mecanismo de perturbación consistente en provocar una pequeña distorsión en el estado actual. Si la diferencia de energía entre estados,  $(E_j - E_i)$ , es menor o igual a cero, el estado  $j$  es aceptado como el estado actual. Si la diferencia de energía es mayor a cero, el estado  $j$  tiene una cierta probabilidad,  $\Phi_{ji}$ , de ser aceptado. Dicho valor es determinado por el factor de Boltzmann, tal como se expresa en la ecuación:

$$\phi_{ji} = e^{-\left(\frac{E_j - E_i}{KBT}\right)}$$

Donde  $T$  denota la temperatura, y  $K_B$  es una constante física conocida como constante de Boltzmann. El algoritmo de RS es concebido para problemas de minimización y, por lo tanto, siempre se considera que existe un mejoramiento del estado actual cuando  $E_j - E_i \leq 0$ . Sin embargo, en términos generales, un mejoramiento existe cuando se presenta un decremento en la energía del sistema.

En optimización, un movimiento (paso de una alternativa a otra ubicada en su vecindario) será aceptado si éste mejora la función objetivo; o en caso contrario, si su probabilidad de aceptación es mayor que un número aleatorio uniformemente distribuido. Este tipo de estrategias permiten que el algoritmo escape de óptimos locales. Por otro lado, y con el fin de avanzar en la convergencia a una solución, en la medida en que evoluciona el método de optimización se disminuye la temperatura y se incrementa la longitud de la cadena. Con esto disminuye la probabilidad de aceptar soluciones de mala calidad.

La codificación del problema define el espacio de búsqueda y por consiguiente la estructura de vecindad. Este aspecto juega un papel fundamental en la eficiencia del algoritmo debido al concepto de proximidad entre estados energéticos. En un sistema real un estado energético es una transformación continua del estado energético anterior, y por lo tanto, una pequeña perturbación produce un pequeño cambio energético. En un problema de optimización combinatoria, una inadecuada definición de un vecino podría, eventualmente, producir grandes diferencias con respecto a la función objetivo, lo cual es indeseable para el algoritmo de RS.

Por esta razón, se ha demostrado que el algoritmo RS tiene gran eficiencia en algunos problemas específicos, mientras que en otros es fácilmente superado por una heurística simple.

Los aspectos fundamentales que se deben definir en el algoritmo de RS, tienen que ver con lo que se denomina el programa de enfriamiento, el cual abarca los siguientes puntos: Temperatura inicial, longitud de la Cadena, ley de enfriamiento, y temperatura final

- Temperatura Inicial ( $T_0$ ):

Es calculada únicamente al inicio del proceso junto con la longitud de la primera cadena. Su valor depende principalmente del problema tratado y de la probabilidad de aceptación  $\tau_0$ . En un problema combinatorio, la temperatura debe tener las mismas dimensiones de la función objetivo del problema. Por ejemplo, para el problema del agente viajero, la temperatura  $T$  corresponde a una distancia. Valores muy elevados, hacen que el proceso tenga un mayor esfuerzo

computacional y valores bajos pueden hacer que el algoritmo quede atrapado en soluciones de baja calidad al inicio del proceso. Aunque existen muchas propuestas, el valor de  $T_0$  puede ser calculado usando el siguiente algoritmo:

i. A partir de una alternativa inicial con una función objetivo  $E_1$ , generar  $k$  perturbaciones aleatorias (vecinos) y evaluar la función objetivo  $E_j$  de cada una. Esto puede verse como una exploración parcial del vecindario, en donde el tamaño  $k$  del mismo no es un aspecto crítico. Seguidamente se obtiene el valor promedio de la función objetivo usando la siguiente ecuación:

$$\Delta E = \frac{\sum_{j=1}^k (E_j - E_1)}{K}$$

ii. Escoger una tasa de aceptación de vecinos  $\tau_0$  basada en el porcentaje de alternativas que degraden (desmejoren) la calidad de la alternativa inicial. Valores bajos de  $\tau_0$  (por ejemplo  $\tau_0 = 20\%$ ) indican que se está asumiendo una alternativa inicial de buena calidad ya que sólo serán aceptados unos pocos vecinos que degraden la alternativa inicial. Es recomendable, con niveles altos de temperatura, usar valores altos de  $\tau_0$ , por ejemplo  $\tau_0 = 50\%$ . La siguiente ecuación puede ser usada para determinar el valor del parámetro  $T_0$ :

$$T_0 = e^{-\frac{\Delta E}{\tau_0}}$$

- Longitud de la cadena de Markov

Debe permitir al proceso alcanzar el cuasi equilibrio en cada nivel de temperatura. Más que un parámetro, la longitud de la cadena define cuando se debe realizar un cambio de estado de energía. Una manera simple de seleccionar la longitud de la cadena es hacerlo de acuerdo con el tamaño del problema. Así, por ejemplo, un cambio de estado puede tomar lugar cuando una de las siguientes dos condiciones se cumple

$12 \cdot N$  Perturbaciones son aceptadas

$P = 100 \cdot N$  Perturbaciones son realizadas

Generalizada con 100 tareas y 5 agentes,  $N = 500$ . A medida que la temperatura disminuye, menos alternativas vecinas (perturbaciones) son aceptadas, y por lo tanto la exploración del

vecindario en busca de una mejor alternativa se intensifica hasta revisar, en el peor caso, 100N alternativas vecinas. En otras palabras, esto significa que a medida que la temperatura disminuye la longitud de la cadena de Markov aumenta implícitamente.

- Ley de Enfriamiento

El enfriamiento del sistema puede hacerse de muchas formas. La más aceptada, por su simplicidad, es la ley de decremento geométrico, donde la temperatura de un nuevo estado está dada por:

$$T_{K+1} = \alpha \cdot TK$$

Donde  $\alpha$  es una constante en el intervalo [0.5-0.9]

- Temperatura final

Este parámetro define el criterio de parada del algoritmo. Puede ser usado el criterio de activar la finalización del algoritmo cuando se alcanzan más de n estados de temperatura sin ninguna aceptación de vecinos. Generalmente n es un valor entero entre 3 y 5. También se puede utilizar un número determinado de cambios de estado o iteraciones

El mecanismo de aceptación de alternativas se introduce a través de la regla de aceptación de Metrópolis: Si se tienen dos soluciones,  $\pi_i$  (alternativa actual) y  $\pi_j$  (alternativa modificada), con valores de la función objetivo  $f(\pi_i)$  y  $f(\pi_j)$ , respectivamente, entonces el criterio de aceptación determina si  $P_j$  es aceptado como alternativa actual. Para ello, se aplica la siguiente probabilidad de aceptación  $P_j$  :

$$P_j = \begin{cases} 1 & \text{Si } f(\pi_j) - f(\pi_i) \leq 0 \\ e^{-\left(\frac{f(\pi_j) - f(\pi_i)}{FK}\right)} & \text{Delocontrario} \end{cases}$$

En la anterior ecuación, es utilizada una versión levemente modificada del factor de Boltzmann, donde, para altas temperaturas este factor es cercano al valor uno, y, por lo tanto, la mayoría de vecinos son aceptados, transformando el algoritmo en una especie de exploración aleatoria en el espacio de solución. Por otro lado, a bajas temperaturas este factor es cercano a cero, y, en



consecuencia, la mayoría de movimientos que degradan la función objetivo son rechazados. En este último caso, el algoritmo se comporta como un clásico mejoramiento iterativo, mejor conocido como “Estrategia Golosa”. En una temperatura intermedia, el algoritmo tiene la capacidad de salir de óptimos

El algoritmo de RS tiene como características principales la ventaja de adaptarse fácilmente a una gran diversidad de problemas y ser una metaheurística cuya convergencia es garantizada bajo ciertas condiciones. Sin embargo, el establecimiento de estas condiciones de convergencia no ha sido aceptado por unanimidad en la comunidad científica. Algunos estudios basados en cadenas de Markov, demuestran el comportamiento asintótico en la convergencia del método. Esta demostración se basa en cumplimiento de dos propiedades: la reversibilidad y la conectividad. La propiedad de reversibilidad establece que para cualquier cambio permitido en el sistema debe permitirse también una oposición al cambio.

La propiedad de conectividad establece que cualquier estado del sistema puede ser alcanzado comenzando desde cualquier otro estado, en un número finito de cambios elementales. Estas dos propiedades permiten justificar el decremento de la temperatura en estados, de lo cual depende la velocidad de convergencia del algoritmo, y garantiza que una solución de buena calidad puede ser encontrada en un tiempo polinomial para ciertos problemas de tipo NP-hard.

En la siguiente Figura se presenta el diagrama de flujo del algoritmo descrito (Santa Chavez, Peñuela Meneses, & Granada Echeverry, 2014):

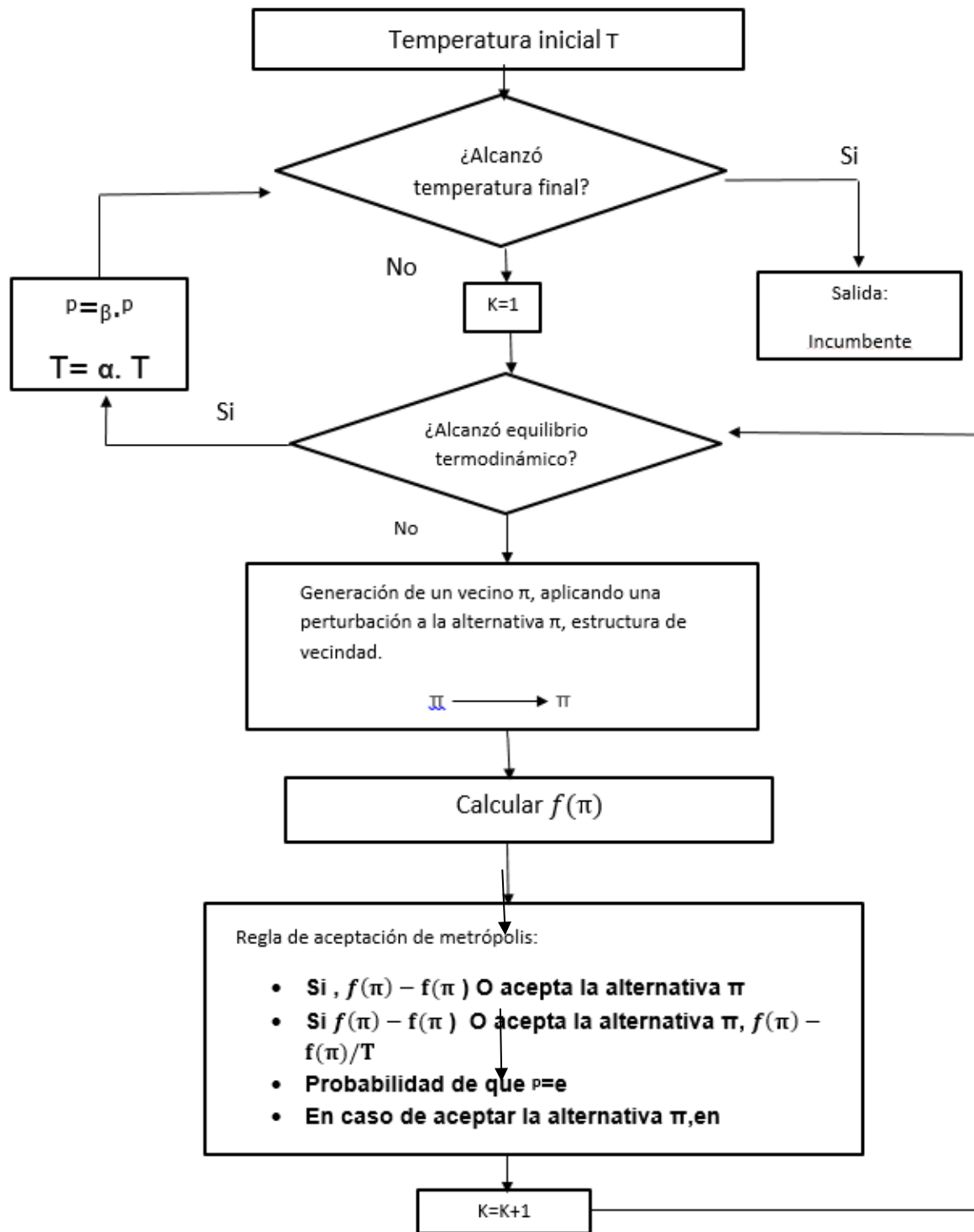


Figura 10. Fuente: Diagrama de flujo. Fuente: (Santa Chavez, Peñuela Meneses, & Granada Echeverry, 2014)

## Código

```
function TSP
global Dij
% Algoritmo de recocido simulado
% Problema agente viajero
% Problema para 11 ciudades

% La distancia entre las ciudades es simétrica  $D_{ij}=D_{ji}$ 

%% Datos del problema

% Matriz de distancias
% Problema para 6 ciudades
% La matriz es de dimensiones (notación de matlab)  $(n-1) \times n$ 
% son  $n-1$  filas y  $n$  columnas, las distancias se escriben en la matriz
% triangular superior.
% Cada elemento representa la distancia de la ciudad  $i$  a la  $j$ .
Dij=[...
    0 5 6 4 8 9
    0 0 7 9 12 9
    0 0 0 8 9 10
    0 0 0 0 7 5
    0 0 0 0 0 3];
Dij=triu(Dij,1); % Deja únicamente la matriz triangular superior
% Longitud de la cadena
Nk=6*size(Dij,2); % longitud de la cadena inicial
crit_parada=false;
ntemp=1; % contador de niveles de temperatura
%% Configuración Inicial
% Obtención del número de ciudades
```

```

long=size(Dij,2);
confg=randperm(long); %obtención de la configuración inicial
FOact=evaluar(confg); % Valor de la función objetivo inicial
FOinc=FOact; % Inicialización de la incumbente
inc=confg; % mejor configuración obtenida hasta el momento

%% Temperatura Inicial
% Utilización de una fórmula simple
phi=0.85; %aceptación 85%
u=0.01; % cambios en la FO de 1%
To=-u/log(phi)*FOact; % temperatura inicial del problema

%% Proceso de recocido simulado
it=1;
graf=FOinc;
T=To; % Primera temperatura
while ~(crit_parada) % inicio del SA
    for j=1:Nk % Evaluación de las cadenas en cada temperatura
        it=it+1;
        new=transicion(confg);
        FOnew=evaluar(new);
        prob=rand;
        if FOnew<=FOact % si mejora la solución
            confg=new;
            if FOnew<FOinc % si mejora la incumbente
                inc=confg;
                FOinc=FOnew;
            end
            FOact=FOnew; % actualización de FO
        elseif prob<exp((FOact-FOnew)/T)
            confg=new;

```

```

        FOact=FOnew;
    end
    graf=[graf FOinc];
end
ntemp=ntemp+1;
T=0.98*T;
Nk=ceil(1.1*Nk);
crit_parada=(ntemp>10);
end
%% Visualización de resultados
% grafica de la función objetivo
plot(1:it,graf)
title('incumbente')
xlabel('iteraciones')
ylabel('Costos')
% visualización de resultados
disp('    RESULTADOS OBTENIDOS    ')
disp("")
disp('Mejor configuración obtenida')
disp(inc)
disp('FO de la configuración ')
disp(FOinc)

function sal=evaluar(vector) %Evaluación de la configuración
global Dij
sal=0;
for i=1:length(vector)
    if i~=length(vector)
        if vector(i)>vector(i+1)
            sal=sal+Dij(vector(i+1),vector(i));
        else

```

```
        sal=sal+Dij(vector(i),vector(i+1));
    end
else
    if vector(end)>vector(1)
        sal=sal+Dij(vector(1),vector(end));
    else
        sal=sal+Dij(vector(end),vector(1));
    end
end
end
end
```

```
function out=transicion(vector) % Cambio de dos ciudades al azar
out=vector;
% Cambio de dos ciudades para la estructura de vecindad
i1=round(rand*(length(vector)-1))+1;
i2=round(rand*(length(vector)-1))+1;
out(i1)=vector(i2);
out(i2)=vector(i1);
```

A continuación se presenta la tabla de resultados:

```
function TSP
global Dij
%Algoritmo de recocido simulado
% Problema agente viajero
% Problema para 11 ciudades

% La distancia entre las ciudades es simétrica  $D_{ij}=D_{ji}$ 

%% Datos del problema

% Matriz de distancias
% Problema para 6 ciudades
% La matriz es de dimensiones (notación de matlab) (n-1)xn
% son n-1 filas y n columnas, las distancias se escriben en la matriz
% triangular superior.
% Cada elemento representa la distancia de la ciudad i a la j.
Dij=[...
    0 5 6 4 8 9
    0 0 7 9 12 9
    0 0 0 8 9 10
    0 0 0 0 7 5
    0 0 0 0 0 3];
Dij=triu(Dij,1); %Deja únicamente la matriz triangular superior
% Longitud de la cadena
Nk=6*size(Dij,2); %longitud de la cadena inicial
crit_parada=false;
ntemp=1; %contador de niveles de temperatura
%% Configuración Inicial
```

Figura 11. Resultados 1. Fuente: MatLab

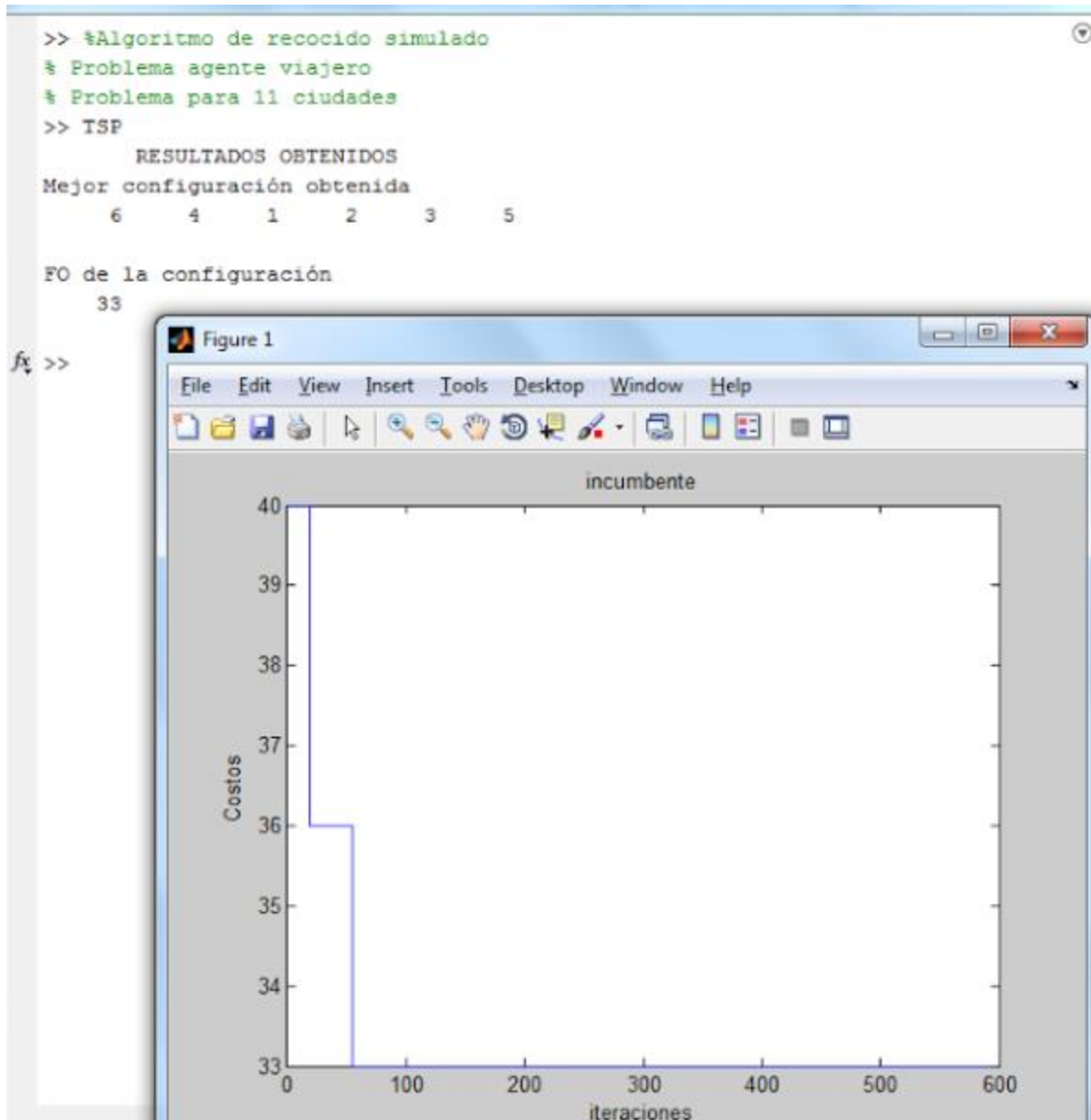


Figura 12. Resultados 2. Fuente: MatLab

## 2.2 Algoritmo de estimación de distribución

Los Algoritmos de Estimación Distribuciones también pertenecen a la clase de algoritmos de optimización basados en poblaciones. Estos están motivados por la idea de descubrir y explotar la interacción entre las variables en la solución. En estos algoritmos se estima una distribución de



probabilidad de la población de soluciones y se toman muestras para generar la siguiente población. Muchos Algoritmos de Estimación Distribuciones utilizan modelos gráficos como técnicas probabilísticas de modelado para este propósito. En particular, los modelos gráficos dirigidos (redes bayesianas) han sido ampliamente utilizados en los Algoritmos de Estimación Distribuciones (Perez Rodriguez & Hernandez Aguirre, 2015).

El comportamiento de los algoritmos genéticos depende de un buen número de parámetros: los operadores de cruce y mutación, las probabilidades de cruce y mutación, el tamaño de la población, la tasa de reproducción generacional, el número de generaciones, etc. De hecho la determinación de valores adecuados para dichos parámetros constituye por sí mismo un verdadero problema de optimización. Por otra parte una mala elección de los valores de los parámetros puede llevar a que el algoritmo obtenga soluciones alejadas del óptimo. Este es uno de los motivos por los que desde hace varios años se han venido estudiando alternativas a los métodos heurísticos estocásticos existentes que no necesiten el ajustar un número alto de parámetros (Inza, Iñiqui, & Larranaga, 2017).

Los EDAs son algoritmos heurísticos de optimización que basan su búsqueda –al igual que los algoritmos genéticos– en el carácter estocástico de la misma. También al igual que los algoritmos genéticos los EDAs están basados en poblaciones que evolucionan. Sin embargo a diferencia de los algoritmos genéticos en los EDAs la evolución de las poblaciones no se lleva a cabo por medio de los operadores de cruce y mutación. En lugar de ello la nueva población de individuos se muestra de una distribución de probabilidad, la cual es estimada de la base de datos conteniendo al conjunto de individuos seleccionados de entre los que constituyen la generación anterior. Mientras que en los algoritmos genéticos las interrelaciones entre las variables representando a los individuos se tienen en cuenta de manera implícita, en los EDAs dichas interrelaciones se expresan de manera explícita a través de la distribución de probabilidad asociada con los individuos seleccionados en cada generación. De hecho la estimación de dicha distribución de probabilidad conjunta asociada a los individuos seleccionados en cada generación es la principal dificultad de esta aproximación (Inza, Iñiqui, & Larranaga, 2017).

En el proceso de evolución de cualquier GA puede ser visto como la combinación de dos procesos; la selección y la variación. La selección se encarga de dirigir la evolución hacia mejores soluciones, mientras que los operadores de cruce y mutación conforman el proceso de variación el cual ayuda a explorar el espacio de soluciones. En un GA, el operador de cruce implícitamente recombina soluciones parciales de los individuos seleccionados a cruzarse. El operador de mutación implícitamente causa esporádicos y aleatorios cambios en los individuos que se produjeron. Por lo tanto, la contribución clave de la variación es producir mejores individuos y ayudar a mantener la diversidad en la población. Los procesos de la selección y la variación juntos forman la base para una mejor evolución. En muchos problemas de optimización, las variables en la solución pueden o no interactuar entre ellas para tener un efecto positivo en la aptitud de la solución. Por ejemplo en un GA, una solución  $\{ \}$  es codificada como un conjunto de valores,  $x_i$ . La cadena de valores es conocida como cromosoma. Dependiendo del problema a tratar, un bit, un número real o entero puede ser usado para los elementos del cromosoma, es decir, los genes. Ahora bien, el valor de aptitud es calculado bajo un criterio de optimización que es modelado en la forma de una función conocida llamada función de aptitud  $f(x)$ . La Figura 1.4 muestra un cromosoma con valores en bits. Aquí la función objetivo es simplemente la suma de todos los bits en el cromosoma (Perez Rodriguez & Hernandez Aguirre, 2015).

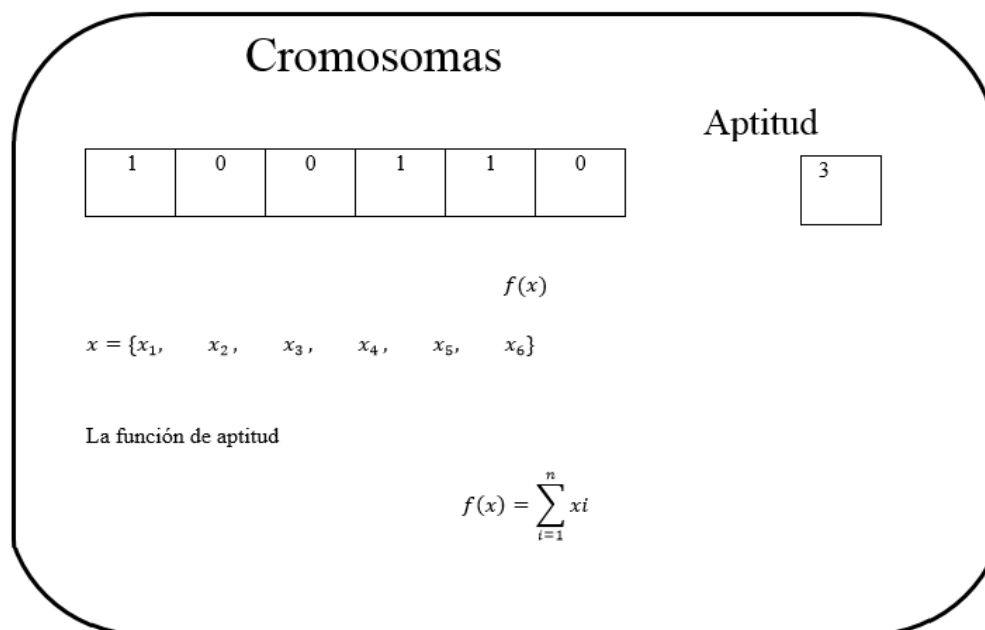


Figura 13. Función de aptitud. Fuente: (Perez Rodriguez & Hernandez Aguirre, 2015)

Sin embargo, cada gen del cromosoma contribuye individualmente e independientemente a la función de aptitud, lo que significa que no existe una interacción entre las variables de este ejemplo. Aunque la interacción puede o no estar presente, para la mayoría de los problemas de optimización y en particular de programación de operaciones esta información no es conocida de antemano. Por lo tanto, para resolver problemas de programación de operaciones es deseable tener el conocimiento de la interacción entre las variables. Según Shakya et al (2006) esto puede ser usado para dirigir la búsqueda más eficientemente. Retomando la parte de la variación de un GA, esta intenta recombinar soluciones prometedoras de la población a través de los operadores de cruce y mutación, asumiendo que estos producirán mejores

1 0 0 1 1 0 3 Cromosoma Aptitud  
 $x = \{ x_1, x_2, x_3, x_4, x_5, x_6 \}$  f(x) La función de aptitud 22 individuos.

Sin embargo, ni el operador de cruce ni el de mutación, intentan aprender o aprovechar la interacción entre variables que pudiese existir entre ellas. En vez de eso, estos operadores aleatoriamente escogen el momento de cruce y mutación definidos por la tasa de cruce y mutación respectivamente. Esta naturaleza aleatoria de los operadores puede algunas veces trastornar los valores que ofrece la interacción entre las variables evitando así obtener efectos positivos en la aptitud del individuo. Además, puede ocasionar que se requiere un mayor tiempo computacional para converger a una buena solución. Al notar este hecho, los investigadores han estado enfocados en descubrir y aprovechar la interacción entre variables asociadas al problema en estudio. Esta tarea ha originado dos enfoques principalmente. El primer enfoque está basado en el cambio de la representación del problema. La idea es manipular la representación de la cadena de soluciones para prevenir trastornos en los valores de las variables que interactúan. Los algoritmos genéticos de Goldberg et al (1989), Kargupta (1996), y Harik (1999) caen en esta categoría. El segundo enfoque está basado en cambiar el proceso de variación. La idea es aprender y aprovechar la interacción entre variables mediante la estimación de una distribución de la población y muestrear a partir de dicha distribución los descendientes. Entonces los operadores de cruce y mutación involucrados en el proceso de variación son remplazados por la estimación y el muestreo de una distribución de probabilidad (Perez Rodriguez & Hernandez Aguirre, 2015).

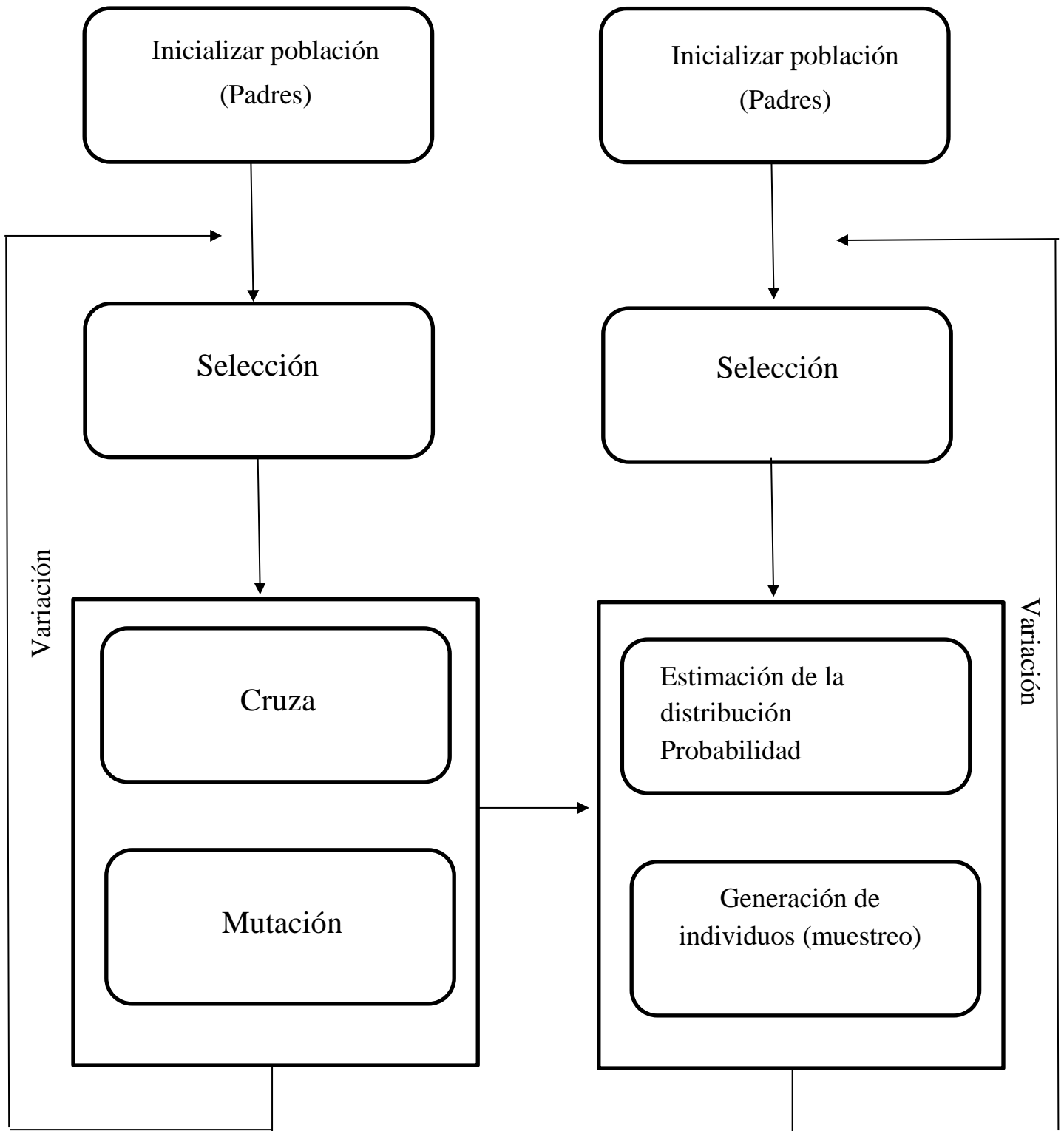


Figura 14. Comparación entre un AG y un ADB. Fuente: (Perez Rodriguez & Hernandez Aguirre, 2015)

En estadística, la distribución de un dato o conjunto de datos es la probabilidad de que se observe o se presente dicho dato o conjunto de datos en el universo de datos de donde se extrae. Por lo tanto, generalmente, la distribución se conoce como distribución de probabilidad. La estimación de la distribución es la tarea de calcular la distribución de probabilidad del dato o conjunto de datos en el universo de datos. Cuando se conoce la distribución de probabilidad asociada a los datos entonces se utiliza esta directamente. De lo contrario, la aproximación de la distribución de probabilidad puede realizarse con algún subconjunto de la población que se construyó, generalmente con los mejores individuos.

Se requiere introducir algunas notaciones para hacer más claro el hecho de que a través de una distribución de probabilidad se puede generar nuevos individuos en el proceso evolutivo. Para ello se adopta el enfoque planteado por Larrañaga et al (1999). Sea  $X_i$  una variable aleatoria, y  $x_i$  uno de sus posibles valores. Usamos  $p(X_i = x_i)$  para representar la función de densidad de probabilidad sobre un punto  $x_i$ . Ahora sea  $X = \{X_1, X_2, \dots, X_n\}$  un vector de  $n$  variables aleatorias y  $x = \{x_1, x_2, \dots, x_n\}$  ser un vector de valores que toma cada variable del vector  $X$ , entonces  $p(X = x)$  representa la función de densidad de probabilidad conjunta de  $X$ . Similarmente la función de densidad de probabilidad condicional de la variable  $X_i$  que toma el valor  $x_i$  dado el valor  $x_j$  de la variable  $X_j$ , será representado como  $p(X_i = x_i | X_j = x_j)$ .

Si el problema es de dominio discreto, por ejemplo, si cada  $X_i$  es una variable aleatoria con un conjunto finito de valores, entonces  $p(X_i = x_i)$  es la distribución marginal univariante de la variable  $X_i$ . Si todas las variables en  $X$  son de tipo discreto entonces  $p(X = x)$  será la distribución de probabilidad conjunta. De igual forma, la probabilidad condicional que  $X_i$  tomará el valor  $x_i$  dado el valor  $x_j$  de la variable  $X_j$ , y puede ser representado por  $p(X_i = x_i | X_j = x_j)$ . Sea ahora  $X_s$  un subvector de  $X$ , y sea  $x_s$  un posible conjunto de valores que puede tener  $X_s$ , entonces  $p(X_s = x_s)$  es la distribución marginal del conjunto  $X_s$ . Ahora bien sean  $X_a$  y  $X_b$  vectores disjuntos de  $X$ , y sean  $x_a$  y  $x_b$  los posibles subconjuntos de valores que puede tener  $X_a$  y  $X_b$  respectivamente, entonces  $p(X_a = x_a | X_b = x_b)$  denota la probabilidad condicional de  $X_a = x_a$  dado  $X_b = x_b$  y puede ser definido como,  $p(x_a, x_b)$  es la probabilidad conjunta de los subconjuntos  $X_a = x_a$  y  $X_b = x_b$ . La factorización de la distribución de probabilidad conjunta,  $p(X = x)$  será  $p(X = x) = p(x_1|x_2, \dots, x_n)p(x_2|x_3, \dots, x_n) \dots p(x_{n-1}|x_n)p(x_n)$  conocida también como regla de la cadena. Considerar una solución  $x = \{x_1, x_2, \dots, x_n\}$  como un conjunto de valores que toman un conjunto de

variables aleatorias  $X = \{X_1, X_2, \dots, X_n\}$  donde  $x_i \in \{0,1\}$ . Entonces, la estimación de la distribución es la aproximación de la distribución de probabilidad conjunta  $p(X = x) = p(x_1, x_2, \dots, x_n)$  de una población  $P$  de soluciones. En general, el cálculo de  $p(X = x)$  consiste en el cálculo de la probabilidad de todas las  $2^n$  combinaciones de  $x$  por lo tanto no es un enfoque factible. Sin embargo, dependiendo de las interacciones entre las variables,  $p(X = x)$  puede ser factorizada en términos de la probabilidad marginal (o condicional) calculada con las variables que interactúan. Por lo tanto, para estimar  $p(X = x)$  se requiere a) aprender la interacción entre variables y b) estimar la probabilidad marginal (o condicional) de las variables que interactúan, donde cada variable,  $X_i \in X$ , solamente interactúa con sí misma, y no interactúa con otras variables. Debido a esa falta de interacción entre variables la distribución de la probabilidad conjunta  $p(X = x)$  puede ser factorizada en términos de probabilidades marginales univariantes de las variables individuales de  $X$ , es decir,  $p(X_i = x_i)$  como

$$P(X = x) = \prod_{i=1}^n P(X_i = x_i)$$

Ahora que ya se tiene un modelo factorizado para  $p(X = x)$ , el siguiente paso es estimar las probabilidades de las variables que interactúan. Existen diversas formas de hacerlo. Una manera sencilla para el ejemplo en cuestión es, dado un conjunto de soluciones,  $D$ , de la población  $P$ , la probabilidad marginal univariante de  $x$ , siendo esta 1,  $p(x_i = 1)$ , puede ser estimada al dividir el número de soluciones en  $D$  con 1 en la  $i$ -ésima posición por el número total de soluciones  $N$  en  $D$ .

$$P(x_i = 1) = \frac{1}{N} \sum_{x \in D, x_i = 1} 1$$

$p(x_i = 0)$  puede ser calculado de manera similar. Una vez que se calcula  $p(X = x)$ , (en este caso todas las probabilidades marginales ya que se estableció de antemano que  $P(x)$  no existe interacción entre las mismas variables) se pueden generar muestras para  $x_i$  que representa un descendiente de la siguiente manera 1, si un número generado con distribución uniforme entre 0 y 1  $\leq p(x_i = 1)$   $x_i = 1$ , de otra manera Repitiendo este proceso para cada  $i \in \{1, 2, \dots, n\}$  resultará un nuevo descendiente. La misma idea se utiliza para construir la nueva generación de

individuos, es decir, una población completa, la cual puede ser utilizada para reemplazar la población padre P. la siguiente figura muestra el ejemplo descrito previamente.

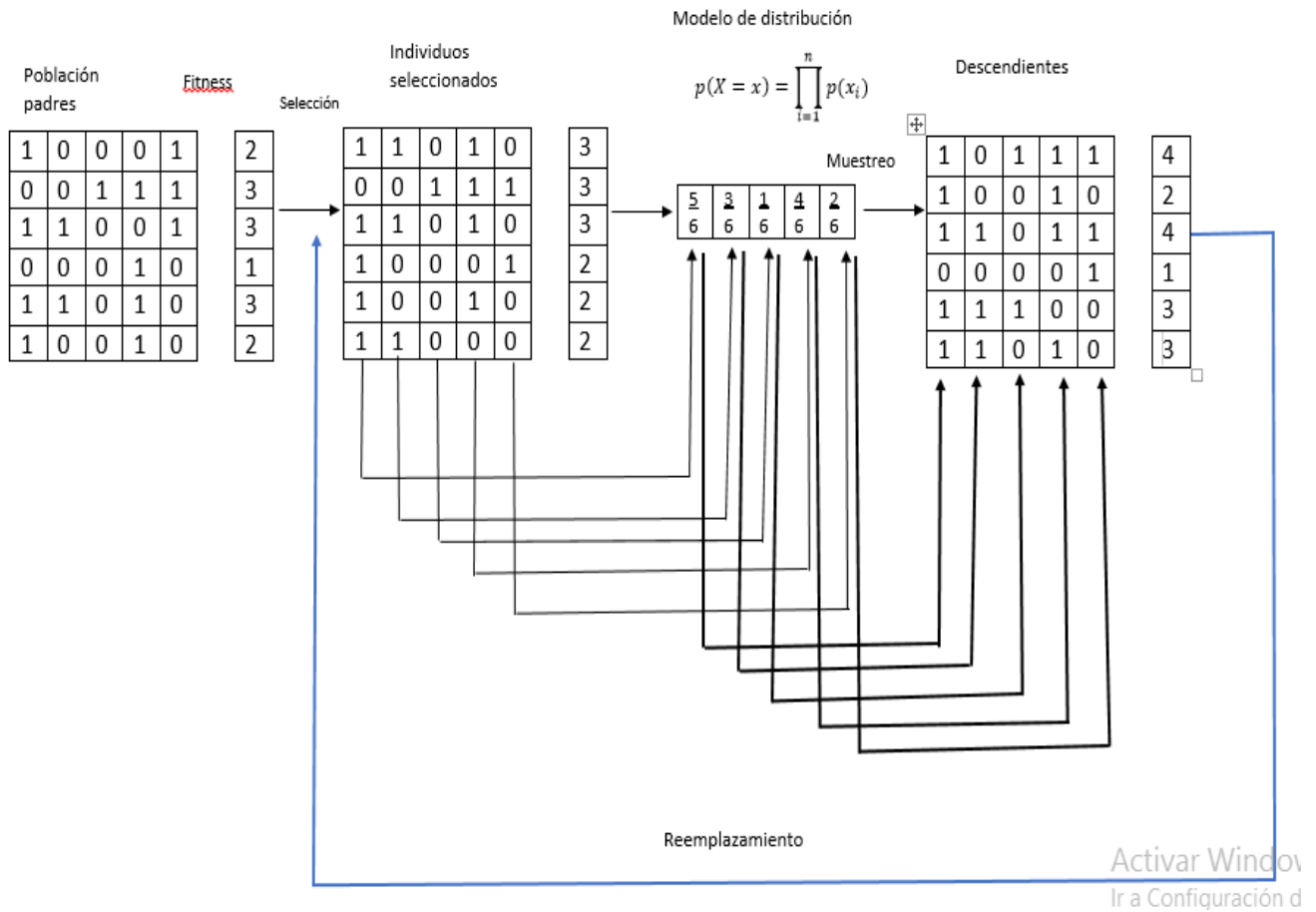


Figura 15. Modelo ADB. Fuente: (Perez Rodriguez & Hernandez Aguirre, 2015)

Aunque fue un ejemplo muy simple es suficiente para demostrar que ya no es requerido los operadores de cruce y mutación definidos en un GA. 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1 0 2 3 3 1 3 2 1 1 0 1 0 0 0 1 1 1 1 1 0 1 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 3 3 3 2 2 2 1 0 1 1 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 1 0 4 2 4 1 3 2 5 6 3 6 1 6 4 6 2 6 Selección Estimación de la Distribución Muestreo Población Padres Individuos Seleccionados Fitness Modelo de Distribución Descendientes Reemplazamiento  $p(x_5 = 1) = \frac{2}{6}$  25 La clase de algoritmos que estiman la distribución de probabilidad y realizan muestras a partir de ella para realizar el proceso de variación se les conoce como Algoritmos de Estimación de Distribuciones EDAs (por sus siglas en inglés Estimation of Distribution Algorithms). Introducidos inicialmente por Mühlenbein & Paaß (1996), estos algoritmos son un área de desarrollo en el campo del cómputo evolutivo. Se han propuesto

una amplia variedad de EDAs que utilizan diferentes técnicas para estimar una distribución de probabilidad y muestrear sobre ella y son objeto de investigación activa por la comunidad que desarrolla algoritmos evolutivos. La motivación de su creación principalmente se debe a intentar identificar y aprovechar la interacción entre las variables que participan en la solución de un problema para asistir en la evolución del algoritmo. Sin embargo, existen dos factores más que han motivado a los investigadores hacia un enfoque probabilístico de la evolución (Larrañaga & Lozano, 2002). El primero es que la ejecución de un GA depende de elegir adecuadamente los parámetros de control al menos tradicionales tales como tasa de cruce, tasa de mutación y tamaño de población. Pero esto ha llegado a ser un problema de optimización en sí mismo (Grefenstette, 1986). La motivación entonces está en minimizar los parámetros del algoritmo. El segundo factor es el hecho de que el análisis teórico de un GA es una tarea difícil. Diversas teorías han sido propuestas para explicar la evolución de un GA sin embargo una teoría convincente aún no se ha desarrollado. La motivación en este punto es lograr un mejor análisis y un mayor rigor teórico sobre el proceso de la evolución (Larrañaga & Lozano, 2002). La forma de operar un EDA inicialmente es igual que en un GA, generando una población inicial  $P$ , que consiste de  $M$  soluciones. El subconjunto  $D$  consiste de  $N$  soluciones que son seleccionadas de  $P$  acorde a un criterio preestablecido. La estimación de la distribución se hace a partir del conjunto  $D$  y se utiliza para generar nuevos descendientes que sustituyen a  $P$ . este proceso se repite hasta que un criterio de paro es alcanzado. Algoritmo de Estimación de Distribuciones 1. Generar población inicial  $P$  (padres) de tamaño  $M$  2. Seleccionar un subconjunto  $D$  de  $P$  que consiste de  $N$  soluciones, donde  $N \leq M$  3. Estimar la distribución de probabilidad  $p(X=x)$  a partir del subconjunto  $D$  4. Muestrear  $p(X=x)$  para generar descendientes 5. Ir al paso 2 hasta que el criterio de paro es alcanzado Pseudo-código de un EDA básico. De este pseudo-código se puede observar que el corazón de cualquier EDA es la estimación de la distribución y la generación de descendientes con ella. La tarea de estimar la distribución dependen de dos factores adicionales: 1) de la precisión de la interacción y/o relación entre las variables, y 2) de la precisión de la estimación de las probabilidades en el modelo de distribución. Cualquier EDA puede ser visto como la combinación de estos tres factores:

- Estimar la relación entre las variables
- Estimar las probabilidades
- Generar muestras (descendientes) a partir de la distribución



Los EDAs se pueden distinguir y clasificar por el tipo de variables con las que tratan; discretos o continuos. Aunque el objetivo real es clasificarlos según tipo de vínculo utilizado por su modelo de distribución. Dependiendo del tipo de conexión utilizado en su modelo de distribución, los EDAs puede clasificarse en univariados, bivariados y multivariados.

### **Código**

**Input:** , ,

**Output:**

Population InitializePopulation(, )

EvaluatePopulation(Population)

    GetBestSolution(Population)

**While** (StopCondition())

    Selected SelectFitSolutions(Population, )

    Model ConstructBayesianNetwork(Selected)

    Offspring

**For** ( **To** )

        Offspring ProbabilisticallyConstructSolution(Model)

**End**

    EvaluatePopulation(Offspring)

        GetBestSolution(Offspring)

    Population Combine(Population, Offspring)

**End**

**Return** ()

### **Seudocódigo**

El pseudocódigo de un AED general es el siguiente:

    Generar al azar  $M$  individuos, formando la población .

$i = 0$

    Mientras no se cumpla la condición de término, hacer:

$i = i + 1$

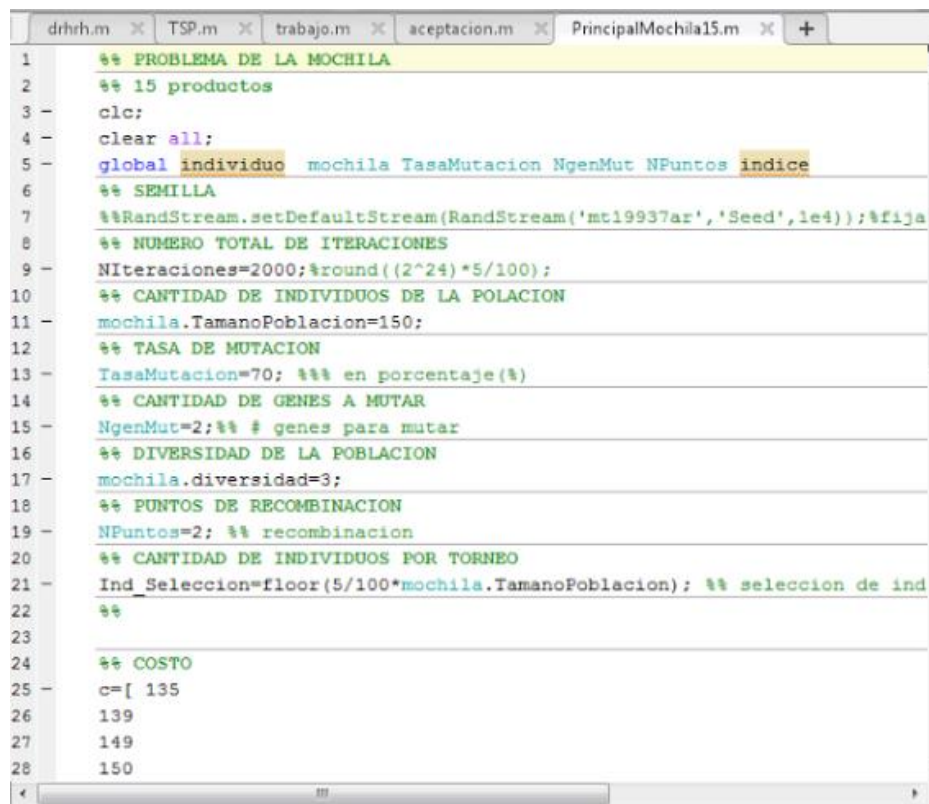
Seleccionar  $N$  individuos ( $N < M$ ) desde la población precedente ( ), formando la población:

Estimar la distribución de probabilidad de cada variable del problema, usando la población .

Generar al azar  $M$  individuos utilizando las distribuciones obtenidas , formando la población .

Fin del ciclo.

A continuación se presentan los resultados del código:



```
drhrh.m x TSP.m x trabajo.m x aceptacion.m x PrincipalMochila15.m x +
1 %% PROBLEMA DE LA MOCHILA
2 %% 15 productos
3 - clc;
4 - clear all;
5 - global individuo mochila TasaMutacion NgenMut NPuntos indice
6 %% SEMILLA
7 %%RandStream.setDefaultStream(RandStream('mt19937ar','Seed',1e4));%fija
8 %% NUMERO TOTAL DE ITERACIONES
9 - Niteraciones=2000;%round((2^24)*5/100);
10 %% CANTIDAD DE INDIVIDUOS DE LA POLACION
11 - mochila.TamanoPoblacion=150;
12 %% TASA DE MUTACION
13 - TasaMutacion=70; %%% en porcentaje(%)
14 %% CANTIDAD DE GENES A MUTAR
15 - NgenMut=2;%% # genes para mutar
16 %% DIVERSIDAD DE LA POBLACION
17 - mochila.diversidad=3;
18 %% PUNTOS DE RECOMBINACION
19 - NPuntos=2; %% recombinacion
20 %% CANTIDAD DE INDIVIDUOS POR TORNEO
21 - Ind_Seleccion=floor(5/100*mochila.TamanoPoblacion); %% seleccion de ind
22 %%
23
24 %% COSTO
25 - c=[ 135
26     139
27     149
28     150
```

Figura 16. Resultados 1. Fuente : Matlab

Elapsed time is 6.572032 seconds.

INCUMBENTE:

ans =

```
cromosoma: [0 0 0 0 0 0 1 1 1 0 0 1 1 1 1]
costo: 1453
volumen: 748
infactible: 0
```

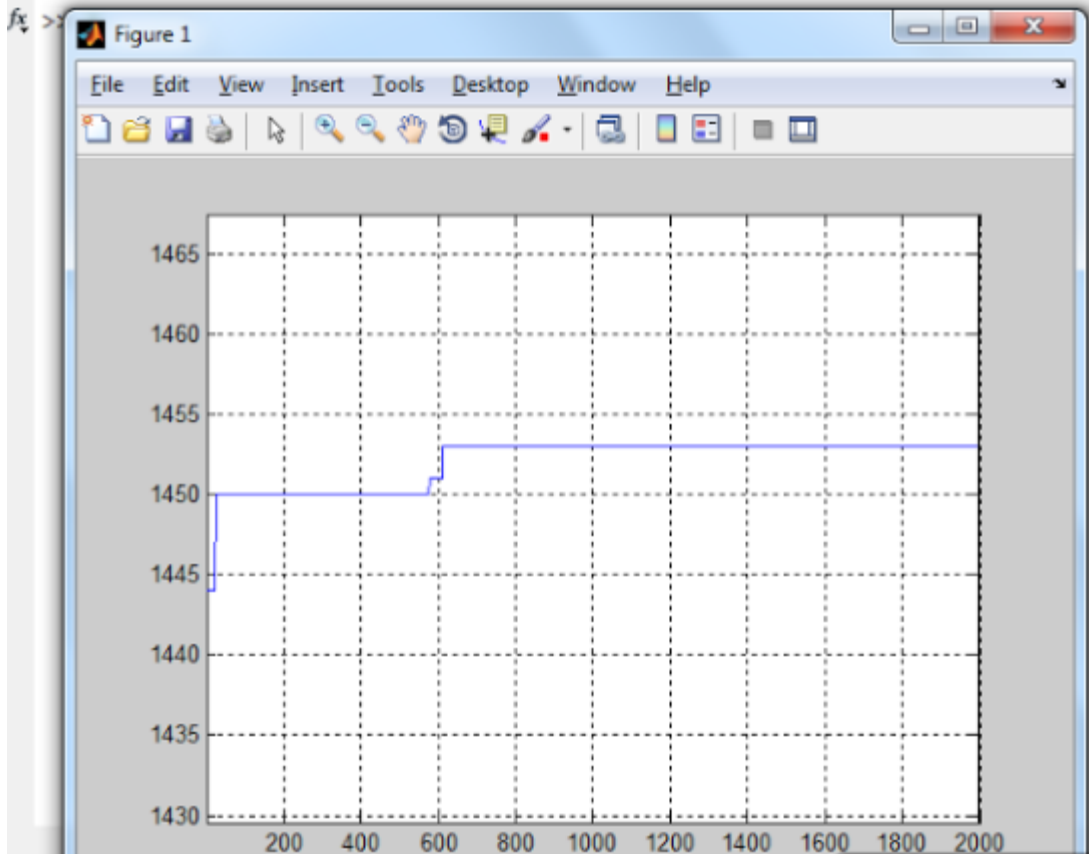


Figura 17. Resultados 2. Fuente : Matlab

```
+1 TSP.m x trabajo.m x aceptacion.m x PrincipalMochila15.m x PrincipalMochila24.m x +
1 %% PROBLEMA DE LA MOCHILA
2 %% 24 PRODUCTOS
3 - clc;
4 - clear all;
5 - global individuo mochila TasaMutacion NgenMut NPuntos indice
6 %% SEMILLA
7 %%RandStream.setDefaultStream(RandStream('mt19937ar','Seed',1500));%fij
8 - rng(1500,'mt19937ar');
9 %% parametros con los que se encuentra el optimo
10 %% NUMERO TOTAL DE ITERACIONES
11 - NIteraciones=7000;%round((2^24)*5/100);
12 %% CANTIDAD DE INDIVIDUOS DE LA POBLACION
13 - mochila.TamanoPoblacion=150;
14 %% TASA DE MUTACION
15 - TasaMutacion=15; %%% en porcentaje
16 %% CANTIDAD DE GENES A MUTAR
17 - NgenMut=1;%% # genes para mutar
18 %% DIVERSIDAD DE LA POBLACION
19 - mochila.diversidad=4;
20 %% PUNTOS DE RECOMBINACION
21 - NPuntos=10; %% recombinacion
22 %% CANTIDAD DE INDIVIDUOS POR TORNEO
23 - Ind_Seleccion=floor(5/100*mochila.TamanoPoblacion); %% seleccion por ca
24
25 %%
26 %% COSTO
27 - c=[ 825594
28      1677009
Ln 1 Col 1
```

Figura 18. Resultados 3. Fuente : Matlab

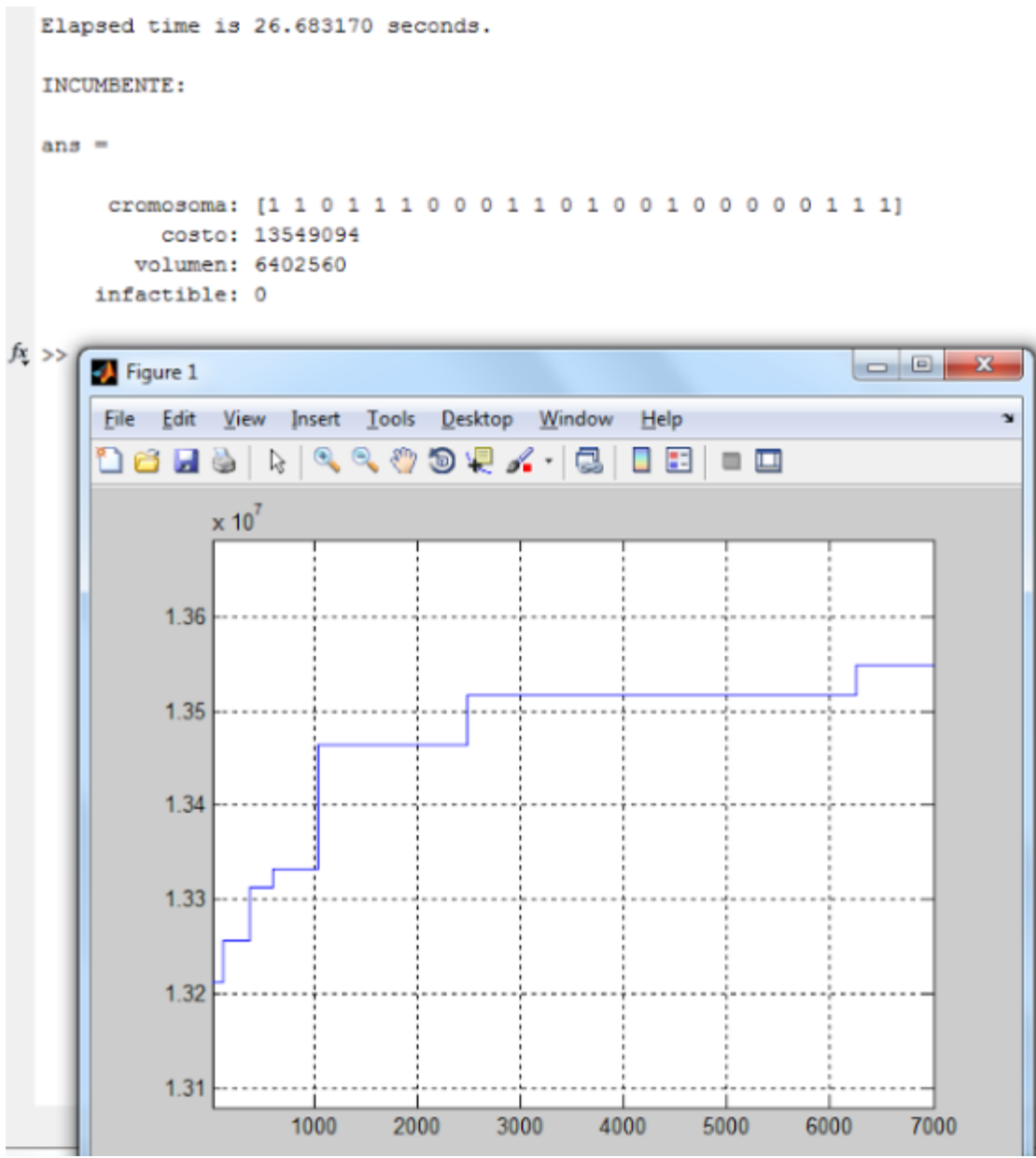


Figura 19. Resultados 4. Fuente : Matlab

### 3. ANÁLISIS DE RESULTADOS

Las diferentes técnicas utilizadas encuentran una solución factible pero en tiempo de computo diferentes esto debido a la determinación de los parámetros que lo conforman

Para evitar la determinación de parámetros la técnica bayesiana es un poco mejor que las metaheurísticas, siempre y cuando los datos se pueden representar a través de una función de probabilidad

Para Problemas de pequeño dimensionamiento se sugiere una técnica heurísticas frente a las bayesianas y cuando el dimensionamiento es menor es más adecuado las técnicas bayesianas

#### 4. CONCLUSIONES Y RECOMENDACIONES

- La solución del problema por cualquiera de las técnicas es buena de acuerdo al margen de error con el que se pretenda, así pues las técnicas metaheurísticas entregan buenos resultados, y estos dependen de la asignación de los parámetros los cuales pueden ser modificados a criterio del investigador.
- La técnica de estimación ofrece la opción de evitar la selección de parámetros, los cuales pueden ser molestos de calcular y en algunos casos calculados sin rigurosidad matemática.
- No existe una técnica mejor que otra, los resultados dependen del tipo de problema de la experiencia del programador y del conocimiento del problema.
- Ampliar el abanico de técnicas en la solución de este tipo de problemas de optimización ya que en muchas ocasiones algunos investigadores sesgan sus investigaciones a las técnicas que manejan.

#### 5. RESULTADOS EN EVENTOS ACADÉMICOS

- Presentación en el IV Congreso Latinoamericano de Estadística Bayesiana, titulada " Optimización por medio de algoritmos poblacionales y su mejoramiento por medio de estimación de distribución",

- Presentación en el congreso nacional de matemáticas convergencia de algoritmos genéticos por medio de cadenas de Markov

## 6. TRABAJOS FUTUROS

- Utilizar técnicas de ramificación a este tipo de problemas
- Realizar híbridos entre las diferentes técnicas
- Explorar un poco más las técnica estocástica las cuales reflejan un poco mejor la realidad del problema Estás pueden ser análisis bayesiano o a través de funciones de probabilidad diferentes a las gaussianas.

## 7. BIBLIOGRAFÍA

Blickle, T., & Thiele, L. (1995). *A comparison of selection schemes used in*. Zurich.

Departamento de Matemáticas, CSI/ITESM. (2018). *Recocido Simulado*.

Gamboa Salgado, J. M., & Gomez Marulanda, J. A. (2012). *FORMULACIÓN DE UN ALGORITMO GENÉTICO PARA EL PROBLEMA DE PROGRAMACIÓN DE ÓRDENES DE TRABAJO DE UNA EMPRESA DE ARTES GRÁFICAS*. Bogotá: FUNDACIÓN UNIVERSITARIA KONRAD LORENZ.

Gestal Pose, M. (2010). *Introducción a los Algoritmos Genéticos*. Universidad de Coruña.

Gestal, M., Rivero, D., Rabuñal, J., Dorado, J., & Pazos, A. (2010). *Introducción a los Algoritmos Genéticos y la Programación Genética*. Coruña: Universidade da Coruña.

- Goldberg, D., Korb, B., & Deb, K. (1989). Messy genetic algorithms: Motivation, analysis and first results. *Complex Systems*, 493-530.
- Harik, G. (1999). The compact genetic algorithm. *IEEE-EC*.
- Inza, Iñiqui, & Larranaga, P. (2017). *Tema 3. Algoritmos de Estimación de Distribuciones*.
- Karagupta, H. (1996). The gene expression messy genetic algorithm. *Proceedings of the 1996 IEEE International Conference*, 631-636.
- Olivera, A. (2004). *Heurísticas para problemas de ruteo de vehículos*. Montevideo: Universidad de la República.
- Perez Rodriguez, R., & Hernandez Aguirre, A. (2015). *UN ALGORITMO DE ESTIMACIÓN DE DISTRIBUCIONES PARA EL PROBLEMA DE SECUENCIAMIENTO EN CONFIGURACIÓN JOBSHOP FLEXIBLE*. Guanajuato: CIMAT.
- Rosales Bello, J. (2005). *PLANIFICACIÓN OPERACIONAL DE LA CAPACIDAD EN SISTEMAS LOGÍSTICOS CON RESTRICCIONES DE TIEMPO. APLICACIÓN DE LA METAHEURÍSTICA GRASP*. Sevilla: Escuela Superior de Ingenieros de Sevilla.
- Santa Chavez, J. J., Peñuela Meneses, C. A., & Granada Echeverry, M. (2014). *Algoritmo de recocido simulado aplicado al. Pereira*.
- Shakya, S., McCall, J., & Brown, D. (2006). Solving the ising spin glass problem using a bivariate EDA based on Markov. *Proceedings of IEEE Congress on Evolutionary Computation*.
- Villar Ledo, L., & Alonso Roque, Y. (2013). Algoritmos Genéticos aplicados a la optimización de antenas. *Revista Telemática*, 21-31.



8.Anexos

Códigos