# Enfoque evolutivo para problemas de localización en líneas de ensamble con backtracking

*Ninoska Maneiro Malavé* [(1)], *Jaqueline Loyo de Sardi* [(2)]
[(1)] *Departamento de Investigación Operativa, Escuela de Ingeniería Industrial, Facultad de Ingeniería, E-mail: nmaneiro@uc.edu.ve*
[(2)] *Decanato de la Facultad Experimental de Ciencias y Tecnología, E-mail: jloyo@uc.edu.ve*

*Universidad de Carabobo, Valencia, Estado Carabobo. CP 2005. Venezuela.*

## Resumen

En este trabajo, se presenta un algoritmo basado en la metodología evolutiva para obtener la asignación de M máquinas en línea que minimice el número de pasos en reversa o "backtracking" de trabajos entre las máquinas, conocido como el problema general de líneas de ensamble. Este problema combinatorio pertenece a la clase de problemas denominados Quadratic Assignment Problem (QAP) y es considerado por algunos autores como NP-completo. Se diseñó y aplicó dicho algoritmo, para varios conjuntos de parámetros en ocho casos, comparando su desempeño con el método Enumerativo y con el método Depth-First Insertion Heuristic, según el caso. Los resultados demuestran superioridad del algoritmo en términos de calidad de la solución; el entonamiento realizado al algoritmo evidenció excelente convergencia hacia mejores soluciones, robustez ante el cambio de parámetros en un tiempo de computación aceptable, lo que indica la validez del enfoque evolutivo para resolver problemas como el planteado.

**Palabras clave:** QAP, backtracking, algoritmos evolutivos.

# Evolutionary approach for location problems in joint flow line with backtracking

## Abstract

One goal of designing a generalized flow line is to assign M machines to M locations along a linear track to minimize the total backtracking movements of jobs. It can be formulated as a quadratic assignment problem which is a very difficult combinatorial optimization problem to solve, considered for some authors as NP-complete. In this paper, an evolutionary algorithm is proposed to solve this special case of the QAP. The designed algorithm was tested in eight cases, its performance was compared with Enumeration and/or Depth- First Insertion Heuristic, and it showed an excellent performance in terms of quality of the solutions in an acceptable computing time. It also showed robustness to parameter changes and fast convergence to better solutions. Compared with other methods, the evolutionary algorithm method shows ability in finding better solutions and it is a very good approach for solving the problem of generalized flow line.

**Keywords**: QAP, backtracking, evolutionary algorithms.

## 1. INTRODUCCIÓN

El problema general de líneas de ensamblaje (LE) es una línea en la cual las operaciones fluyen hacia adelante y pueden procesarse, o no, en todas las máquinas. Un trabajo en tal clase de línea puede comenzar a procesarse y completar su proceso en cualquier máquina, moviéndose siempre hacia delante (*downstream*) por operaciones sucesivas de acuerdo

con la secuencia de trabajo del proceso. Cuando dicha secuencia para un trabajo no especifica una máquina colocada delante de su localización actual, el trabajo tiene que viajar en sentido contrario *(upstream)* a fin de completar la operación requerida.

Este *"viaje en reversa"*de las operaciones, es llamado *Backtracking*, y se desvía de una línea de ensamble ideal para un trabajo específico, resultando en una estructura de trabajo menos eficiente.

En el pasado, el objetivo del problema unidimensional de localización de facilidades era minimizar los movimientos de trabajo en ambos sentidos. La minimización del *Backtracking* de trabajos en una línea de producción sirve a varias metas implícitas, tales como la reducción del tiempo de ocio de las máquinas, la simplificación del problema de la programación y carga, el incremento de la salida en la línea de producción.

Una vasta literatura trata con problemas de localización de máquinas en sistemas de manufactura clásicos, pero aparentemente ha sido poca la investigación a fin de minimizar el flujo hacia atrás o Backtracking.

El problema del Backtracking ha cobrado importancia desde finales de los 90 en los países industrializados, debido al incremento en el uso de los equipos automatizados de manejo de materiales en la manufactura de lotes pequeños debido al uso de los sistemas de Vehículos Guiados Automáticamente (AGV) que son más eficientes cuando se mueven a lo largo de caminos rectos. La Figura 1 muestra un sistema donde se usa esta clase de vehículos y la Figura 2, tipos de AGV.

Figura 1. Ejemplo de línea donde utilizan AGV´s.


Figura 2. Tipos de AGV (Vehículos Guiados Automáticamente).

La minimización del viaje en reversa o "Backtracking" en una LE puede ser formulada como un problema cuadrático de asignación de facilidades (QAP). El problema cuadrático de asignación de facilidades es la generalización y extensión de un problema tratado por el hombre tan tempranamente como el siglo XVII. Cubre una amplia clase de problemas que comprende la minimización del costo total de interacción entre pares de facilidades, nuevas y existentes.

Estos problemas involucran desde el diseño de paneles de control y teclados, balance de turbinas, asignación procesador-procesador en ambientes de procesamiento distribuido, análisis de reacciones químicas para compuestos orgánicos, hasta catalogar datos arqueológicos.

Koopmans y Beckman fueron los primeros en formular, en 1.957, el problema de localización de facilidades como un QAP. El nombre **Quadratic Assignment Problem** fue escogido porque la función objetivo suele ser un polinomio de segundo grado de las variables y las restricciones son idénticas a las del problema simple de asignación de facilidades [1].

Entre los casos particulares más conocidos del QAP pueden mencionarse:

1. El problema de **localización de una máquina nueva con respecto a un grupo de máquinas existentes**, cuyo objetivo es minimizar el costo de los movimientos bi-direccionales de trabajo.

2. El problema del **agente viajero (TSP)**, un problema clásico de rutas, en donde el agente viajero deberá planificar su itinerario para minimizar el costo total de su recorrido. El espacio de búsqueda para el TSP es un conjunto de

permutaciones de n ciudades. Cualquier permutación simple de las n ciudades produce una solución, pero la solución óptima es una permutación que produzca el mínimo costo en el recorrido.

En su forma más general, el objetivo del QAP es encontrar la asignación óptima de **N** facilidades (plantas, departamentos ó maquinarias) a **N** sitios a fin de minimizar el costo total de manejo de materiales expresado como el producto del flujo de trabajo y la distancia recorrida o simplemente la distancia. Sin embargo, en el caso general el costo de transporte entre las facilidades es conocido, pero no puede descomponerse como el producto del flujo de trabajo y la distancia [1].

El QAP se relaciona más estrechamente con el de localización de múltiples facilidades nuevas con respecto a múltiples facilidades existentes. La principal diferencia reside en el número de localizaciones; mientras que en el problema de localización de múltiples facilidades se asume un espacio de soluciones infinito, para el QAP se asume un número finito de localizaciones.

## 2. FORMULACIÓN DEL PROBLEMA

Se asume que el sistema de producción de una LE debe procesar N trabajos en el conjunto $J= \{J_1, J_2,..., J_N\}$. Cada trabajo $J_n$ *(n= 1, 2,... ,N)* debe ser procesado en un subconjunto apropiado de M máquinas, lo que se denomina secuencia de proceso. Una máquina no puede ejecutar más de una operación en secuencia sobre cualquier trabajo y puede ejecutar sólo una operación sobre un trabajo en una unidad de tiempo. Puede haber múltiples movimientos de la máquina *i* a la máquina *j* para todos los trabajos $J_n$ *Î J, n= 1, 2,... , N*. Considerando todos los posibles flujos desde la máquina i a la máquina j, una matriz de flujo llamada *matriz de requerimientos, R,* puede ser definida como:

$$R = \left[ r_{ij} \right]_{MxM} \tag{1}$$

donde $r_{ij}$ es el *flujo total* de la máquina i a la máquina j, para todo $J_n$, *n= 1, 2,... ,N*.

Si se asume que exactamente una máquina va a ser asignada a cada localización y que las localizaciones son igualmente espaciadas y numeradas, como 1, 2,..., M de modo secuencial de izquierda a derecha, como se ilustró en la Figura 1. Si un trabajo se mueve desde la localización k a la localización h donde h≤ k, la distancia de *backtrack* es *k-h* unidades.

Sea $x$=[ $x_{11}$,..., $x_{1M}$, $x_{21}$ ..., $x_{2M}$,.., $x_{i1}$,..., $x_{iM}$,..., $x_{M1}$,..., $x_{MN}$] las variables de decisión que representan las localizaciones de las máquinas definidas como sigue:

$$x_{ik} = \begin{cases} 1 & \text{Si la máquina i es} \\ & \text{asignada al sitio k,} \\ 0 & \text{En otro caso} \end{cases} \tag{2}$$

Si la distancia de Backtracking de la localización k a la h donde h≤k, es:

$$d_{kh}^{b} = \begin{cases} k-h & \text{Para } h < k \\ 0 & \text{En otro caso} \end{cases} \tag{3}$$

entonces la distancia de Backtracking entre la máquina **i** y la máquina **j**, asignadas a las localizaciones k y h, esta dada por la siguiente ecuación:

$$d_{ikjh} = r_{ij} d_{hk}^{b} \tag{4}$$

El problema de minimización de la distancia de *Backtracking* en que pueden incurrir los trabajos puede ser formulado como el modelo de QAP. Las ecuaciones planteadas en ese modelo aseguran que cada máquina es asignada a *una y sólo una localización y* que cada localización *es asignada exactamente a una máquina.* Si la máquina i es asignada a la localización k y la máquina j a la localización h, entonces ***ambos*** $x_{ik}$ *y* $x_{jh}$ *son iguales a 1,* y el término de costo $d_{ikjh}$ es incluido en la distancia total de Backtracking f(x).

$$\min \ f(x) = \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n}\sum_{h=1}^{n} d_{ikjh} x_{ik} x_{jh} \ \text{ sujeto a}$$

$$\sum_{i=1}^{n} x_{ik} = 1, \quad k = 1,\dots,n \tag{5}$$

$$\sum_{k=1}^{n} x_{ik} = 1, \quad i = 1,\dots,n \quad x_{ik} = 0, 1 \text{ para todo i, k}$$

## 3. MÉTODOS DE SOLUCIÓN DEL PROBLEMA GENERAL DE LÍNEAS DE ENSAMBLE

El problema general de Líneas de Ensamble (LE) es un QAP, un muy conocido y difícil problema de optimización combinatoria. Muchos problemas de Ingeniería Industrial emplean optimización combinatoria, es decir, la obtención de una solución dentro de un conjunto finito de alternativas. Tales problemas de optimización son notablemente difíciles de resolver; una de las principales razones de esta dificultad es que en muchas aplicaciones el número de alternativas es extremadamente grande y solo una fracción de ellas pueden ser consideradas dentro de una cantidad razonable de tiempo.

El hecho de que el QAP sea uno de los problemas NP-completos o NP-hard más difíciles, ha conducido a los investigadores a concentrarse en el desarrollo de heurísticas para resolverlos. Desde el inicio de la búsqueda de solución para el QAP, los métodos aplicados para tal fin fueron clasificados en procedimientos exactos (algoritmos óptimos: dirigidos a obtener el óptimo del problema) y procedimientos heurísticos (búsquedas dirigidas hacia la mejor solución, sea esta óptima o no).

Hasta el momento se han desarrollado numerosos métodos para resolver el QAP y sus casos particulares, basados en técnicas provenientes de la Investigación de operaciones, tal como el procedimiento de Ramificación y Acotamiento (Branch and Bound), algunas pueden ser clasificadas como métodos clásicos y otras como mezclas de métodos clásicos con nuevos procedimientos, así como técnicas de óptimos locales y de óptimos globales. Sin embargo, en los últimos tiempos se ha volcado la atención de los estudiosos del área hacia la utilización de otros métodos denominados, en términos generales, No Sistemáticos por emplear otras formas de búsqueda, tal como la búsqueda aleatoria y han sido denominados Metaheurísticas o también enfoques heurísticos inteligentes. Simulación de Recocido (SA) y Búsqueda Tabú (TS) son los métodos más populares.

Dentro de este grupo, se encuentran los algoritmos evolutivos, los cuales son métodos de búsqueda y optimización que se inspiran en el **proceso de evolución natural** para diseñar algoritmos computarizados generales, eficientes y robustos. Estos son métodos heurísticos para resolver problemas complejos de optimización y búsqueda, que han sido ampliamente utilizados en muchas disciplinas, tales como optimización combinatoria, aprendizaje de máquinas, procesamiento de imágenes, etc.

En este trabajo se utilizará el método Heurístico desarrollado por Sarker [2] para este caso particular del QAP, denominado Depth-First Insertion Heuristic (DIH) para comparar el desempeño del algoritmo evolutivo propuesto.

## 4. ALGORITMOS EVOLUTIVOS

Un algoritmo evolutivo es un proceso estocástico e iterativo que opera sobre un conjunto P de **individuos** (población) que representan una posible solución al problema que se está considerando. Cada uno de los individuos de la población recibe, a través de una función de adecuación o aptitud (**fitness**), una medida de su bondad o aptitud con respecto al problema que se desea resolver. Este valor es empleado por el algoritmo para guiar la búsqueda.

El algoritmo está estructurado en tres fases principales; Reproducción, Selección y Reemplazo las cuales se ejecutan de manera circular, y se llevan a cabo de manera repetitiva. Cada una de las iteraciones del algoritmo se denomina **ciclo reproductivo básico o generación**.

Durante la fase de **Selección** se crea una población temporal P' en la que aquellos individuos más aptos (los correspondientes a las mejores soluciones contenidas en la población) estarán representados un mayor número de veces que los poco aptos (**principio de selección natural**).

A los individuos contenidos en esta población temporal se les aplican diferentes operadores de cambio o **Reproducción** (también denominados operadores reproductivos o genéticos) en la fase de reproducción. El objetivo de esta fase es producir individuos con nuevas características, idealmente mejores (**principio de adaptación**).

Finalmente, durante la fase de **Reemplazo**, se sustituyen individuos de la población original por los nuevos individuos creados. Este reemplazo afecta a los peores individuos y tiende a conservar los mejores (**supervivencia de los más adaptados**). Todo este

proceso se repite hasta que se cumple un determinado criterio de terminación (normalmente al completar un cierto número de iteraciones, llamadas comúnmente generaciones).

Es importante destacar que el algoritmo descrito establece un compromiso entre la explotación de las buenas soluciones (fase de selección), y la exploración de nuevas zonas del espacio de búsqueda (fase de reproducción), basándonos en que el mecanismo de reemplazo puede permitir la aceptación de nuevas soluciones que no proporcionen una mejora inmediata sobre las ya existentes. La Figura 3 ilustra este proceso.

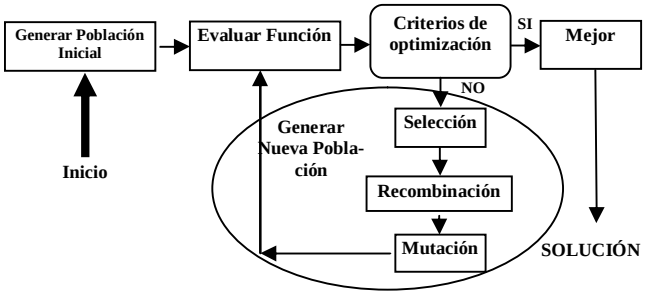


Figura 3. Estructura de un algoritmo evolutivo.

La principal característica de los algoritmos evolutivos es el uso de un operador de recombinación o cruce como mecanismo principal de búsqueda. Este operador debe recombinar los cromosomas de los padres para construir descendientes que posean características de ambos. La utilidad de este operador se fundamenta en la suposición de que diferentes partes de la solución óptima pueden ser descubiertas independientemente y luego ser combinadas para formar mejores soluciones. Adicionalmente, emplean un operador de mutación cuyo uso se considera importante como responsable del mantenimiento de la diversidad en la población, aunque secundario en relación con el operador de cruce. Sin embargo, en investigaciones recientes, se ha revisado el papel de la mutación, considerándolo a la par del operador de cruce, [3-8].

## 5. METODOLOGÍA DE LOS ALGORITMOS EVOLUTIVOS

Es importante aclarar que la metodología y los elementos presentados en el punto anterior se encuentran frecuentemente caracterizando a los Algoritmos Genéticos, pero los investigadores más tradicionales consideran como algoritmos genéticos solo aquellos cuya representación de individuos es la representación binaria, de allí que se considere el algoritmo propuesto como un algoritmo evolutivo ya que su representación es de punto flotante. Sin embargo, en la mayor parte de la literatura existente, se trata ambos términos como sinónimos.

### 5.1. Parámetros del Algoritmo

Los siguientes parámetros definen el entorno evolutivo del algoritmo.

Tam_pob: Tamaño de la población
$p_{cruce}$: Probabilidad de cruce
$p_{mutacion}$: Probabilidad de mutación
max_gen: Número de generaciones a evolucionar

### 5.2. Representación de los Cromosomas

Se asume que las máquinas se numeran 1, 2,..., M, las localizaciones de las máquinas pueden representarse usando las permutaciones de M elementos. Así que una asignación solución se representa como sigue:

$$a=[a_1, a_2,..., a_M],$$

donde $a_i \neq a_j$ para $i \neq j$, $a_i \in \{1, 2,..., M\}$ y la máquina $a_k$ esta localizada en el sitio k;

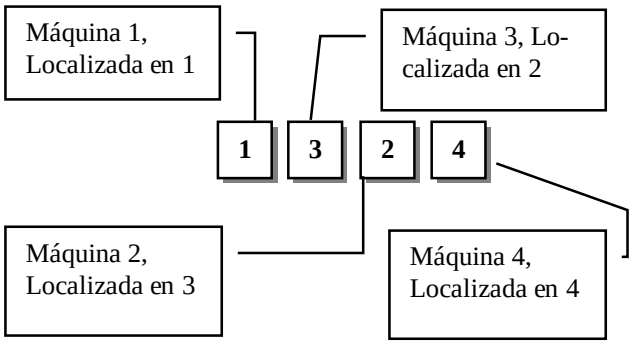


Figura 4. Representación para el problema de cuatro máquinas.

### 5.3. Evaluación

Dado un vector de asignación $a= [a_1, a_2, ..., a_n]$, donde la máquina $a_k$ esta asignada a la localización k; se define la matriz Backtracking B(a), la cual depende únicamente de las localizaciones de las máquinas, como sigue:

$$B(a) = \left[ b_{ij} \right]_{M*M} \quad (6)$$

donde $\mathbf{b_{ij}}$ es la distancia de viaje en reversa de la máquina ***i*** a la máquina ***j,*** que se calcula como sigue:

$$b_{ij} = \sum_{i=1}^{M} \sum_{j=1}^{M} d_{kh}^{b} x_{ik} x_{jh} \quad (7)$$

Por ejemplo, para M = 4 y la asignación $a_1$=[4,3,2,1], la matriz de Backtrack B($a_1$) es:

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

La distancia TOTAL de *viaje en reversa (Backtracking) f(a) o función de evaluación* puede ser calculada mediante la siguiente ecuación:

$$f(a) = \sum_{i=1}^{M} \sum_{j=1}^{M} r_{ij} b_{ij} \quad (8)$$

## 5.4. Población Inicial

Los individuos de la población inicial serán producidos mediante la generación de tam_pob permutaciones aleatorias de {1, 2,..., M}. Se probó con 4 tamaños diferentes de población inicial: 40, 60, 80 y 100 individuos.

## 5.5. Selección

La selección de individuos para aplicarle los operadores genéticos se hizo en forma equiprobable, es decir, asignándole a cada individuo de la población la misma probabilidad de ser seleccionado, combinada con una estrategia elitista que preserva el mejor individuo de la generación actual para la próxima generación reemplazando al peor individuo de la siguiente generación, como se ilustra en la Figura 5.

## 5.6 Cruce y Mutación

El operador de cruce es el propuesto en [3], especialmente diseñado para el QAP, que mantiene en el descendiente las asignaciones comunes de los padres, asignando aleatoriamente el resto de los sitios, como se muestra en la Figura 6.

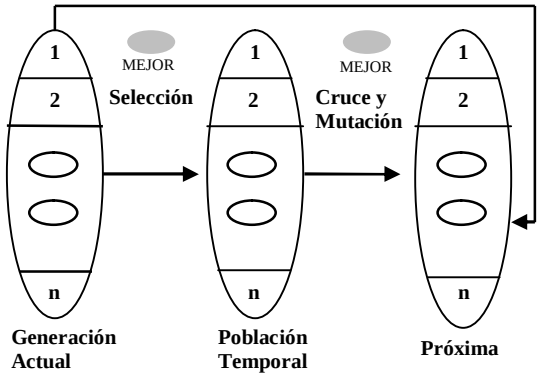


Figura 5. Estrategia elitista. Mantener el mejor para la próxima generación.

Sin embargo, cuando ambos padres son iguales esta clase de cruce originaría un descendiente idéntico, por lo que para garantizar el cruce de los individuos se aplica el clásico cruce en un punto, que se muestra en la Figura 7.

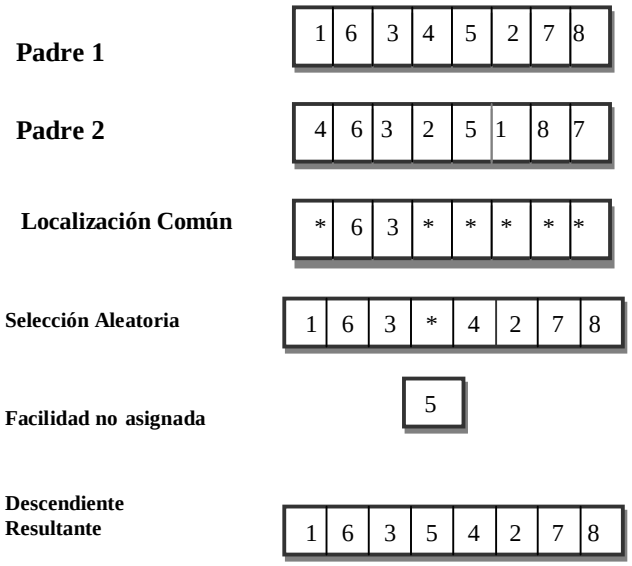


Figura 6. Primer operador de cruce utilizado.

El operador de mutación es el de intercambio de dos elementos, el cual selecciona aleatoriamente dos posiciones e intercambia los elementos que se encuentran en dichas posiciones en el descendiente mutado como se muestra en la Figura 8.

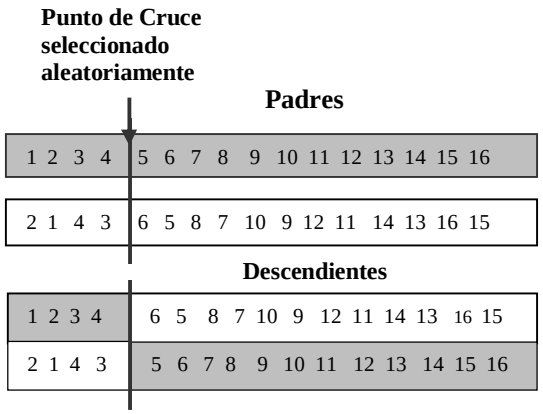


Figura 7. Operador de cruce en un punto.

**Leyenda**: cada vector (Padres) representa una posible localización. En el punto de cruce (línea negra), elegido al azar, se dividen los padres formando dos nuevos descendientes.

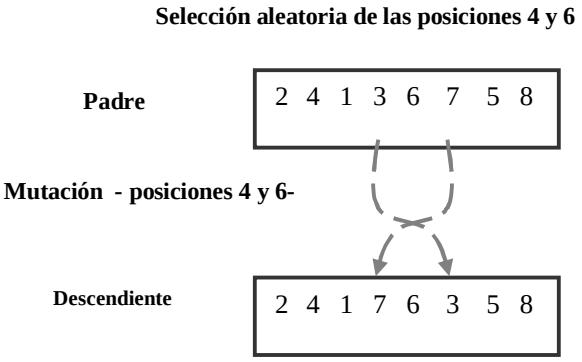


Figura 8. Operador de mutación de intercambio de dos elementos.

## 5.7. Algoritmo

- Paso 0. Establecer los parámetros del algoritmo.
- Paso 1. Generar aleatoriamente la población inicial.
- Paso 2. Hacer la evaluación. Calcular la matriz de Backtracking para cada individuo de la población.
- Paso 3. Hacer gen= 0.
- Paso 4. Si gen> max_gen entonces PARAR, en otro caso CONTINUAR.
- Paso 5. gen:= gen+1.
- Paso 6: Hacer Selección, aplicar estrategia elitista.
- Paso 7. Hacer Cruce y Mutación.
- Paso 8. Evaluar. Calcular la matriz de Backtracking e ir al Paso 4.

## 6. RESULTADOS

Se generaron aleatoriamente 8 ejemplos, que se muestran en la Tabla 1. El desempeño del algoritmo fue comparado, en el caso de los problemas hasta 12 máquinas, con la solución exacta producida por el método enumerativo, y en los problemas de tamaño mayor con el Método Depth- First Insertion Heuristic (DIH). Obsérvese el tiempo empleado por el método enumerativo en el problema de mayores de 12 máquinas, y el hecho de que no se encuentran soluciones para los problemas a partir de 16 máquinas por incapacidad de cómputo para explorar el espacio total de soluciones. Se experimentó además con varios

Tabla 1. Resultados obtenidos para los 8 casos con los tres métodos

| | | | Enumerativo | | DIH | | Algoritmo Evolutivo | |
|---|---|---|---|---|---|---|---|---|
| Caso | Máquinas | Número de Trabajos | BT | T (seg.) | BT | T (seg) | BT | T (seg) |
| 1 | 6 | 36 | 51 | 0 | 55 | 0.0 | 51* | 1 seg |
| 2 | 8 | 48 | 145 | 143 | 164 | 0 | 145* | 2 seg. |
| 3 | 10 | 64 | 6497 | 345 | 6665 | 0 | 6497* | 1 seg. |
| 4 | 12 | 100 | 11781 | $1.63\ 10^5$ | 12313 | 0 | 11781* | 2 seg. |
| 5 | 16 | 120 | - | - | 1588 | 0 | 1525** | 6 seg. |
| 6 | 24 | 160 | - | - | 58868 | 1 | 56536 | 19 seg. |
| 7 | 48 | 160 | - | - | 457301 | 2 | 443872 | 62 seg. |
| 8 | 56 | 180 | - | - | 1340253 | 3 | 1277851 | 75 seg. |

**Leyenda**: BT= Backtracking Total, T= tiempo en segundos, *= Óptimo conocido, **= Posible óptimo

conjuntos de parámetros de cruce (0.4, 0.5, 0.75) y mutación (0.2, 0.25, 0.5 y 0.75), variación del número de generaciones a reproducir (1200, 2500 y 5000), variación del número de individuos y 20 corridas del algoritmo.

Se observa que el método enumerativo sólo puede encontrar soluciones para problemas relativamente pequeños (hasta 12 máquinas).

El método DIH cuesta menos en tiempo computacional, pero la búsqueda es limitada por la matriz de requerimientos del problema, es decir el resultado obtenido no es mejorable usando el mismo método, como en el caso de los algoritmos evolutivos.

Los resultados muestran un desempeño superior del algoritmo en términos de la calidad de la solución, robustez ante el cambio de parámetros y adaptabilidad.

La Tabla 2 muestra un ejemplo de los valores de adaptabilidad para 1200, 2500 y 5000 generaciones, para la combinación cruce/mutación que arrojó el mejor valor en cada caso .

Estos métodos fueron implementados en C++ y ejecutados en un Pentium III, 500 Mhz, 384 Mb RAM. Los resultados se resumen en la Tabla 1: *BT denota el mejor valor obtenido de la función objetivo, ó dado el caso óptimo, y tiempo, medido en segundos.*

Tabla 2. Valor de adaptabilidad/ número de generaciones. Caso 24 máquinas.

| Pcruce | P mutación | Número Generac. | Resultado | Adaptabilidad |
|---|---|---|---|---|
| 0.5 | 0.25 | 1200 | 56766 | 0.011445 |
| 0.5 | 0.25 | 5000 | 56536 | 0.005021 |
| 0.75 | 0.25 | 2500 | 56583 | 0.003461 |

Leyenda: Pcruce= probabilidad de cruce.
Pmutac= probabilidad de mutación

El entonamiento del algoritmo, realizado para el caso de 24 máquinas, demostró que el número mínimo de generaciones para los problemas de 24, 48 y 56 máquinas debe ser al menos de 2500, ya que el número de generaciones a evolucionar si es influyente en la calidad de la solución.

Otro resultado interesante se refiere a las probabilidades de cruce y mutación utilizadas. Puede concluirse que este problema funciona mejor con probabilidades de cruce y mutación consideradas altas para otros problemas [4], debido a que estas altas tasas introducen mayores elementos de aleatoriedad y diversidad en la búsqueda. El cálculo del error relativo de las otras soluciones con respecto a la mejor solución obtenida, indica que no predomina ningún esquema cruce-mutación por sobre los otros, por lo que no hay un criterio que apoye la selección de una combinación común para todos los casos.

## 7. CONCLUSIONES

Los análisis de la convergencia mostraron un excelente desempeño del algoritmo, descendiendo a valores menores desde las primeras generaciones y manteniendo la búsqueda hasta las últimas generaciones o estabilizándose, en caso de una buena solución. Los valores de adaptabilidad mostrados en la Tabla 2, como medida adicional de desempeño, reflejó este comportamiento en sus resultados muy cercanos a cero. Esto se muestra en las Figuras 9 y 10.

La selección de los operadores genéticos es fundamental para el buen desempeño del algoritmo, de ello depende el desarrollo de la evolución y la calidad de las soluciones; por tal razón se experimentó con una nueva combinación de operadores ya conocidos. Se cambió la tradicional selección por el método de la Ruleta, por una selección uniforme más una estrategia elitista y se implementaron dos clases de cruce, uno de ellos diseñado para el problema. Los resultados posteriores sustentan la afirmación inicial acerca de la importancia de la escogencia de los operadores, ya que para cualquier caso estudiado el valor calculado con el algoritmo evolutivo es mejor que su correspondiente en el método DIH, y para algunos problemas grandes es capaz de encontrar el óptimo, tal como el comportamiento del algoritmo en la búsqueda de soluciones lo indica en el problema de 16 máquinas. Esto vislumbra un excelente desempeño en problemas similares donde métodos alternativos han fallado.

Los algoritmos evolutivos no son paquetes de software que pueden ser adaptados a cualquier tipo de problema de optimización.

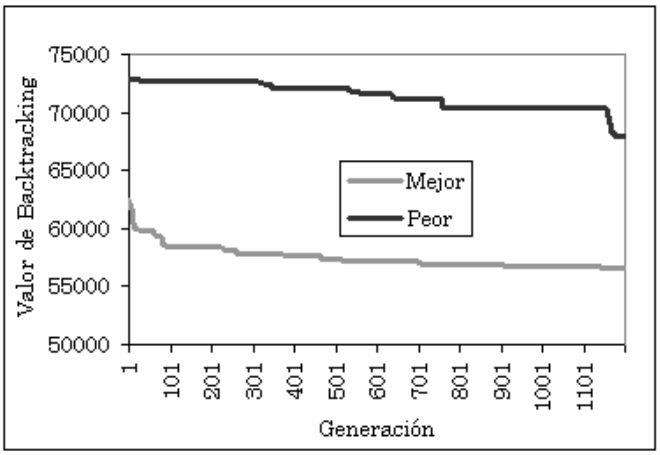


Figura 9. Convergencia del algoritmo, peor y mejor elemento de cada generación.

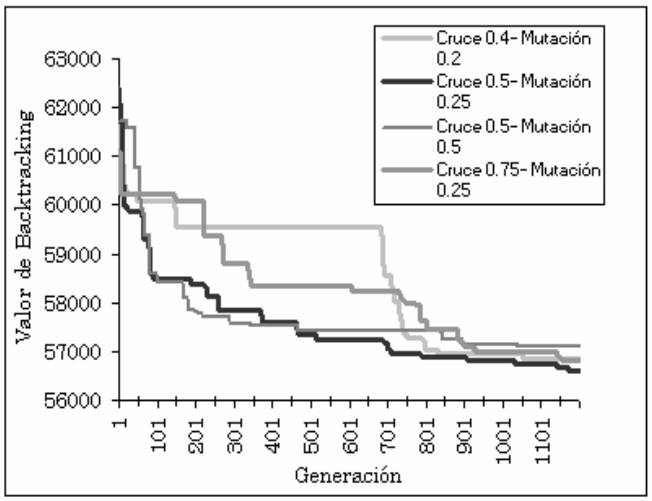


Figura 10. Convergencia para diferentes combinaciones de cruce y mutación. Caso 24 máquinas.

La razón fundamental se basa en el hecho de que algunos elementos para el diseño del algoritmo requieren conocimiento explícito del problema para que pueda tener significado el proceso de evolución y aún más los resultados arrojados, por lo que -en cierta forma- son un "traje a la medida". Esto que puede ser visto como una desventaja, debe ser sopesado con factores tales como el tiempo de respuesta relativamente pequeño en problemas intratables, demasiado grandes o muy estocásticos, robustez de la metodología y calidad de soluciones.

Desde el punto de vista industrial, lo más importante es obtener la mejor distribución de las máquinas en la línea de Ensamble, o un conjunto de buenas soluciones que permitan evaluar alternativas y tomar una decisión, que convertida en unidades monetarias es lo que finalmente se desea minimizar. Así que el análisis de tiempo/calidad de solución, toma otra perspectiva, más tiempo puede traducirse en una mejor solución y, por ende, en un menor costo. Esto evidencia que los algoritmos evolutivos son una poderosa herramienta para la resolución de problemas de asignación de facilidades, abriendo así un amplio campo de investigación para su aplicación en otros problemas o problemas similares.

Los principales aportes del trabajo son los siguientes:

1. Desarrollo de un algoritmo que permite calcular asignaciones en forma rápida y de calidad, mostrando un conjunto de soluciones posibles lo cual facilita la toma de decisiones por parte del analista.
2. Muestra un método alternativo de solución de problemas de asignación de facilidades de alta dimensionalidad y amplios espacios de búsqueda.
3. Comprueba que el desarrollo de algoritmos evolutivos no está restringido a problemas de interés teórico computacional, por lo que debe incentivarse la investigación y tratamiento de problemas de optimización en ingeniería con la ayuda de las técnicas evolutivas, que tiene como ventaja extra las amplias perspectivas que ofrece para el trabajo multidisciplinario.

## 8. REFERENCIAS

[1] Maneiro, N. (2001). "Algoritmos Genéticos aplicados a Problemas de Localización de Facilidades. Caso de Estudio: Problema de Asignación Cuadrática de Facilidades". Trabajo Especial de Grado no publicado. Área de Estudios de Postgrado de la Universidad de Carabobo.

[2] B. R. Sarker et al. (1995). "Backtracking Of Jobs In One Dimensional Machine Location Problems", European Journal Of Operational Research, 85, pp. 593-609.

[3] Tate, D.; Smith, A. (1995). "A Genetic Approach to the Quadratic Assignment Problem", Computers and Operation Research, Vol. 22, No. 1, pp 73-83.

[4] Gen, M.; Chen, R. (1997). "Genetic Algorithm and Engineering Design". Wiley, USA.

[5] Gen, M.; Chen, R. (2000). "Genetic Algorithm and Engineering Optimization". Wiley, USA.

[6] Gong, D.; Yamazaki, G.; Gen, M.; Xu, W. (1999). "A genetic algorithm method for one-dimensional machine location problems". International Journal of Production Economics. 60-61 Pág. 337-342.
[7] Reeves, C. R. (1997). "Genetic Algorithms for the Operations Researcher". INFORMS Journal on Computing. Vol. 9, Nº, Summer.
[8] Ahuja, R.; Orlin, J.; Tiwari, A. (1997). "A Greedy Genetic Algorithm for the Quadratic Assignment Problem". INFORMS, Journal of Computing, Fall.