Implementación de Búsqueda Tabú en la Solución del Problema de Asignación Cuadrática

Dagoberto Ramón Quevedo Orozco
Instituto Tecnológico de Tepic
Ingeniería en Sistemas Computacionales
Tepic, Nayarit 63175, México
dago.qvd@gmail.com

Roger Z. Ríos Mercado
Universidad Autónoma de Nuevo León
División de Posgrado en Ingeniería de Sistemas, FIME
San Nicolás de los Garza, NL 66450, México
roger@mail.uanl.mx

Resumen: Este artículo ilustra el modelado e implementación de metaheurísticas, específicamente una Búsqueda Tabú, para la solución del Problema de Asignación Cuadrática, considerado como un problema difícil de resolver en el campo de la optimización combinatoria. En la implementación computacional se utiliza el marco de trabajo de ParadisEO que facilita en gran medida el desarrollo de la aplicación. Finalmente, utilizando las instancias de QAPLIB, se realiza una experimentación computacional que ilustra la eficiencia de la Búsqueda Tabú para la solución del Problema de Asignación Cuadrática además de mostrar el comportamiento del método durante la variación de sus parámetros de ejecución.

Abstract: This paper illustrates the model and implementation of metaheuristics, specifically a Tabu Search to solve the Quadratic Assignment Problem, considered a difficult problem to solve in the field of combinatorial optimization. The use of the ParadisEO, a framework for implementing metaheuristics, greatly facilitates the development of the application. The computational efficiency of Tabu Search to solve the Quadratic Assignment Problem is illustrated over a set of instances of the well-known QAPLIB data set. The empirical work includes an evaluation of the method as a function of some of its algorithmic parameters.

Palabras clave: Investigación de operaciones, metaheurísticas, búsqueda tabú, optimización combinatoria, marco de trabajo.

Keywords: Operations research, metaheuristics, tabu search, combinatorial optimization, framework.

Introducción

Las aplicaciones de la optimización son innumerables, cada proceso tiene un potencial para ser optimizado. Las compañías e instituciones que toman sus decisiones en base a la investigación de operaciones participan en la solución de problemas de optimización. Diversas aplicaciones en la ciencia y la industria pueden ser modelados como problemas de optimización. La optimización viene dada por la reducción al mínimo de costo, tiempo, distancia y riesgo o la maximización de calidad, satisfacción y beneficios.

Un gran número de problemas de optimización en la ciencia, la ingeniería, la economía y las empresas son complejos y difíciles de resolver. No se pueden resolver de una manera exacta en un tiempo razonable. El uso de algoritmos de aproximación es la principal alternativa para resolver esta clase de problemas. En este escenario de complejidad se presentan como una opción viable el uso de metaheurísticas que si bien no garantizan la mejor solución, dan un resultado factible que satisface todas las restricciones del problema. Los objetivos principales de estos algoritmos son: la solución de problemas de una forma más rápida y la solución de grandes problemas que de una manera exacta su tiempo de cálculo es irrazonablemente alto.

La Búsqueda Tabú es una técnica que puede utilizarse en combinación con algún otro método de búsqueda para resolver problemas de optimización combinatoria con un alto grado de dificultad. Puede verse como una metaheurística que se superpone a una técnica de búsqueda y que se encarga de evitar que dicha técnica caiga en óptimos locales *prohibiendo* ciertos movimientos.

El costo en el desarrollo de la solución de un problema de optimización es también una cuestión importante. En años recientes se han distribuido software libre y código abierto que contribuyen en gran medida a la reutilización de código. Si bien, en ocasiones la naturaleza del problema requiere hacer un desarrollo *desde cero* con la finalidad de ajustarlo a la medida de los requerimientos; habrá situaciones en las que es requerido minimizar el tiempo y el costo del desarrollo, y por lo tanto se recomienda el uso

de marcos de trabajo o *framework* que incluyan diversas características genéricas de los algoritmos metaheurísticos, tal es el caso de ParadisEO [12] que permite la implementación de metaheurísticas de manera eficiente para problemas mono objetivo y problemas multiobjetivo.

El objetivo del presente artículo es ilustrar al lector acerca del modelado e implementación de la metaheurística Búsqueda Tabú para la solución de un Problema de Asignación Cuadrática y mediante una experimentación computacional determinar su eficiencia y comportamiento variando los parámetros de ejecución; entre otro de los objetivos es ilustrar los beneficios de usar marcos de trabajo como ParadisEO durante la fase de implementación.

Conceptos generales

Dada la dificultad práctica para resolver de forma exacta toda una serie de importantes problemas combinatorios para los cuales, por otra parte es importante ofrecer alguna solución, comenzaron a aparecer algoritmos que proporcionan soluciones factibles es decir que satisfacen todas las restricciones del problema.

Este tipo de algoritmos se denominan *heurísticas* [4], del griego *heuriskein*, "encontrar". Las heurísticas son procedimientos simples, a menudo basados en el sentido común, que se supone ofrecen una buena solución (aunque no necesariamente la óptima) a problemas difíciles, de un modo fácil y rápido.

Los procedimientos *metaheurísticos* [7] son una clase de métodos de aproximación que están diseñados para resolver problemas difíciles de optimización combinatoria. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos. Podemos agrupar los algoritmos heurísticos en dos principales grupos: constructivos y de búsqueda local los cuales se definen a continuación.

- *Constructivos*: Son procedimientos iterativos que, en cada paso, añaden un elemento hasta completar una solución. Pueden ser métodos deterministas y estocásticos [7].
- *Búsqueda local*: Parten desde una solución inicial, en cada iteración el movimiento se produce desde una solución actual a una de su entorno o vecindario que mejore la solución actual y finaliza cuando ninguna solución de su vecindario mejora a la actual. Esto tiene una desventaja, dado que la solución final siempre será un óptimo local; para *escapar* de óptimos locales se usan algoritmos que permiten seguir explorando el espacio de soluciones, haciendo uso de estructuras de memoria y técnicas probabilísticas.

Búsqueda Tabú

El origen de la Búsqueda Tabú (TS por sus siglas en inglés, *Tabu Search*) se atribuye a Fred Glover [5] que en sus palabras define: "la búsqueda tabú guía un procedimiento de búsqueda local para explorar el espacio de soluciones más allá del óptimo local".

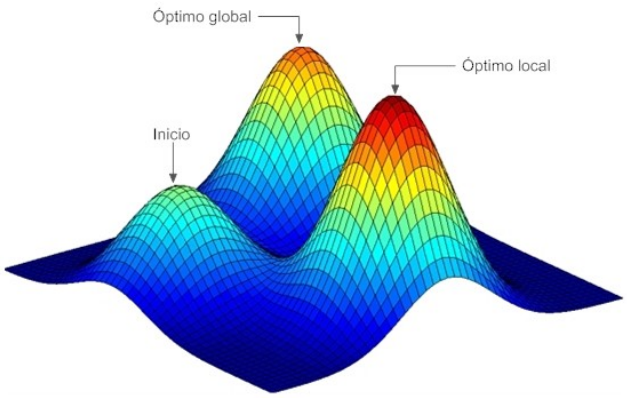

Figura 1: Superficie de función objetivo de un problema de optimización combinatoria usada en la Búsqueda Tabú.

El algoritmo toma de la Inteligencia Artificial el concepto de memoria y lo implementa mediante estructuras simples con el objetivo de dirigir la búsqueda teniendo en cuenta la historia de ésta, es decir, el procedimiento trata de extraer información de lo sucedido y actuar en consecuencia. En este sentido puede decirse que hay un cierto aprendizaje y que la búsqueda es inteligente. De esta manera permite moverse a una solución aunque no sea tan buena como la actual, de modo que se pueda escapar de óptimos locales y continuar estratégicamente la búsqueda de soluciones aún mejores.

Características

El principal distintivo de TS sobre otras metaheurísticas de tipo búsqueda local es su uso de memoria que contiene una estructura basada en una *lista tabú*, así como la implementación de mecanismos para la selección del siguiente movimiento. Los elementos básicos de TS son la estructura del vecindario, el movimiento, la lista tabú y criterio de aspiración. Un *movimiento* es una operación que, cuando se aplica a una solución *x*, genera un vecindario de $N(x)$. Una *lista tabú* es un conjunto de atributos prohibidos o *tabú*, es decir, atributos que no son permitidos ser aplicados en la solución actual.

El tamaño de la *lista tabú* o *tenencia tabú* [4] es la duración que un atributo permanece en estado tabú o tabú-activo (medido en número de iteraciones). El tamaño de la lista tabú puede adoptar diferentes formas.

- *Estático*: Puede depender del tamaño de la instancia del problema y sobre todo del tamaño de la vecindad. No hay un tamaño óptimo para todos los problemas, o incluso todas las instancias de un problema dado. Por otro lado, el valor óptimo puede variar durante el progreso de la búsqueda. Para superar este problema, se utiliza un tamaño variable de la lista tabú.
- *Dinámico*: El tamaño de la lista tabú puede cambiar durante la búsqueda sin necesidad de utilizar ninguna información sobre la memoria de la búsqueda.

Un *criterio de aspiración* es una condición que cuando es satisfecha se cancela la condición de tabú del atributo. La búsqueda se detiene cuando el *criterio de parada* (límite de tiempo, número limitado de iteraciones, etc.) se cumple.

Algoritmo Básico de Búsqueda Tabú

La Búsqueda Tabú [6] puede describirse como sigue. Dada una función $f(x)$ a ser optimizada en un conjunto X de soluciones, TS empieza de la misma manera que cualquier búsqueda local, procediendo iterativamente de un punto (solución) a otro hasta satisfacer un criterio dado de terminación. Cada $x \in X$ tiene un entorno (o vecindad) asociado $N(x) \subseteq X$, y cada solución $x' \in N(x)$ se puede alcanzar desde x mediante una operación llamada movimiento.

Sea $N^*(x) \subseteq N(x)$, donde las soluciones que son admitidas en $N^*(x)$ se determinan de varias formas. Una de ellas da a TS su nombre, identifica soluciones encontradas sobre un horizonte especificado (e implícitamente algunas soluciones identificadas con ellas), y les *prohíbe permanecer* en $N^*(x)$ clasificándolas como *tabú*. A continuación se definen las líneas esenciales en el comportamiento de TS en su esquema básico definido en el Algoritmo 1.

Paso 6: Se toma una solución del vecindario que no pertenezca a una lista tabú representada por *T*.

Paso 7: El movimiento es efectuado sin tomar en consideración si esta solución empeora o mejora a la actual, este comportamiento o movimiento no evaluado es lo que permite a TS *equivocarse* para seguir explorando en un espacio de solución mayor.

Paso 8: Se actualiza la lista tabú con el movimiento efectuado.

Algoritmo 1 Búsqueda Tabú

1: $x^* \leftarrow x : x \in X$
2: $T \leftarrow \varnothing$
3: $i \leftarrow 0$
4: **repeat**
5: $\quad i \leftarrow i + 1$
6: $\quad x' \leftarrow \arg\min f(\mu) : \mu \in N(x) \setminus T$
7: $\quad x \leftarrow x'$
8: $\quad T \leftarrow T \cup \{x\}$
9: $\quad$ **if** $f(x) < f(x^*)$ **then**
10: $\quad\quad x^* \leftarrow x$
11: $\quad$ **end if**
12: **until** $i \leq max\_iteracion$
13: **return** x^*

Paso 9: Evaluación de la solución actual respecto a la solución *incumbente* o mejor encontrada hasta el momento, dependiendo si el problema es de minimización o maximización, si la solución actual resulta ser más atractiva, entonces la solución actual es asignada a la *incumbente*.

TS en su esquema básico no contempla un *criterio de aspiración* que omita el estado tabú de una solución.

Problema de Asignación Cuadrática

El Problema de Asignación Cuadrática (QAP por sus siglas en inglés, *Quadratic Assignment Problem*), fue introducido [1] como un modelo matemático para la ubicación de un conjunto indivisible de actividades económicas.

Se considera el problema de asignación de un conjunto de facilidades a un conjunto de localidades, teniendo la distancia entre cada localidad y el flujo entre las facilidades, además de los costos asociados a la instalación en un cierto lugar [2]. Se busca que este costo, en función de la distancia y flujo, sea mínimo. El QAP es NP-duro y es considerado como un complejo problema de optimización combinatoria [8].

Formulación del problema

Para cada par de facilidades *i* y *j* se tiene el flujo $a_{ij}(i, j = 1, \ldots, n)$. Para cada par de localidades *i* y *j* se tiene la distancia $b_{ij}(i, j = 1, \ldots, n)$. Se busca asignar una facilidad a cada una de las localidades a fin de minimizar la suma de los productos de los flujos y las distancias. Más formalmente, buscamos la permutación *p* de las *n* localidades que minimice la función objetivo [10].

$$\min_{p \in P(n)} z(p) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{p_i p_j} \qquad (1)$$

donde $A=(a_{ij})$ y $B=(b_{ij})$ son matrices de $n \times n$. $P(n)$ es el conjunto de todas las posibles permutaciones de $1,\ldots,n$ y p_i representa la localidad de la facilidad i en la permutación $p \in P(n)$.

Figura 2: Solución Inicial.

La Figura 2 ilustra una solución inicial que en un contexto específico representa un conjunto de facilidades (A, B, C, D) en un conjunto de localidades (1, 2, 3, 4). La Figura 3 muestra un grafo bidireccional que define el flujo entre cada una de las facilidades.

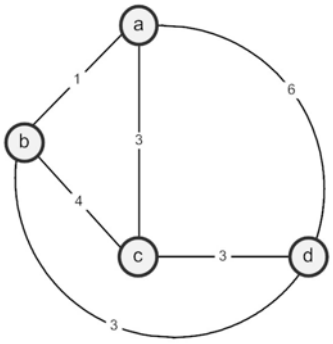

Figura 3: Flujo entre facilidades.

De igual manera la Figura 4 representa la distancia entre cada una de las localidades.

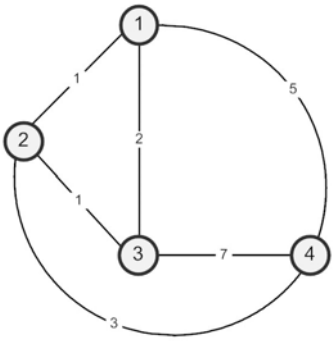

Figura 4: Distancia entre localidades.

El costo de la solución inicial representada en la Figura 2 conforme a la Ecuación (2) es 142. Contando con los datos del problema, y dada la complejidad combinatoria del mismo es posible aplicar métodos heurísticos y metaheurísticos con la finalidad de encontrar una mejor solución. Para este caso en particular se aplico una Búsqueda Tabú en su esquema básico definido en el Algoritmo 1. El resultado de la solución mejor encontrada por TS durante su procesamiento es representada en la Figura 5.

Figura 5: Solución final.

El costo de la solución final representada en la Figura 5 conforme a la Ecuación (2) es 102, obteniendo una mejora del 28.16% respecto a la solución inicial.

Búsqueda Tabú en la solución de QAP

Se describe ahora una TS adaptada para la solución de QAP; el modelo ha sido desarrollado principalmente por los trabajos de Skorin-Kapov [9] y Taillard [10], que efectivamente han reportado resultados favorables para las instancias de QAP en base un modelo de solución basado en TS.

Evaluación

La *función de evaluación* determina el costo asociado a la solución *p*. En este caso, la función permanece sin cambio respecto a la función objetivo definida en la Ecuación (1), el orden de operaciones requerido por esta función es de $O(n^2)$.

$$z(p)=\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{p_i p_j} \qquad (2)$$

Movimiento

El *movimiento* es definido por el intercambio de los elementos *r* y *s*, ubicados en p_r y p_s generando una nueva solución $\mu \in P(n)$.

Algoritmo 2 Movimiento

Require: *p, r, s*

1: $\mu_k = p_k : \forall k \neq r, s$

2: $\mu_s = p_r$

3: $\mu_r = p_s$

4: **return** μ

La evaluación incremental determina el costo de intercambiar los elementos *r* y *s* que intervienen en el

movimiento, sin necesidad de efectuar una evaluación completa de la solución.

$$\Delta(p,r,s) = 2 \times \sum_{k=1,k\neq s,r}^{n} (a_{sk} - a_{rk})(b_{p_r p_k} - b_{p_s p_k}) \qquad (3)$$

Sea [10] $\Delta(p,r,s)$ definida en la Ecuación (3) el costo de intercambiar los elementos *r* y *s* ubicados en p_r y p_s. El orden de operaciónes requerido por esta función es de $O(n)$, una mejora considerable frente $O(n^2)$ de la Ecuación (2).

Vecindario

Sea $N(p)$ el conjunto de todas las permutaciones que se pueden obtener mediante el intercambio de dos elementos diferentes de *p* o bien todos los posibles *movimientos* de *p* donde $N(p)$ es llamado el *vecindario* generado a partir de la solución actual *p* [10].

Algoritmo 3 Vecindario

Require: *p*
1: $r \leftarrow 0, s \leftarrow 0, N \leftarrow \varnothing$
2: **repeat**
3: **if** $r < n-2$ **then**
4: **if** $s < n-1$ **then**
5: $s \leftarrow s+1$
6: **else**
7: $r \leftarrow r+1$
8: $s \leftarrow r+1$
9: **end if**
10: $N \leftarrow N \cup \{\mu : \mu = p, \mu_r = p_s, \mu_s = p_r\}$
11: $m \leftarrow 1$
12: **else**
13: $m \leftarrow 0$
14: **end if**
15: **until** $m = 1$
16: **return** *N*

Atributos Tabú

Los atributos que conforman la *lista tabú*, están establecidos por el par ordenado (r,s) que intervienen en la operación de *movimiento* definida en el Algoritmo 2.

La Tabla 1 muestra un ejemplo de la ejecución de TS, donde es apreciable las actualizaciones que la lista tabú tiene durante cada iteración representada por *i*, el tamaño de la lista tabú o *tenencia tabú* es de *t*=3. Véase como en la iteración 4 la lista tabú esta completa, el par ordenado (1,4) es el siguiente en salir de la lista tabú al restarle solo una iteración con *estatus tabú* antes de ser eliminado de la lista.

i	Lista Tabú *t*=3			*p*	$z(p)$	m_{rs}
	1	2	3			
1	-	-	-	{2,4,1,3}	118	(1,4)
2	-	-	(1,4)	{3,4,1,2}	102*	(3,4)
3	-	(1,4)	(3,4)	{3,4,2,1}	104	(1,2)
4	(1,4)	(3,4)	(1,2)	{4,3,2,1}	118	(1,3)
5	(3,4)	(1,2)	(1,3)	{2,3,4,1}	130	(2,3)
6	(1,2)	(1,3)	(2,3)	{2,4,3,1}	122	(1,4)
7	(1,3)	(2,3)	(1,4)	{1,4,3,2}	114	(3,4)
8	(2,3)	(1,4)	(3,4)	{1,4,2,3}	112	(1,3)
9	(1,4)	(3,4)	(1,3)	{2,4,1,3}	118	(2,3)
10	(3,4)	(1,3)	(2,3)	{2,1,4,3}	126	(1,4)

Tabla 1: Iteraciones de TS para la solución de QAP.

Implementación en ParadisEO

El desarrollo constante de modelos de optimización y algoritmos metaheurísticos cada vez más sofisticados y complejos demanda el uso de software que integren las características requeridas para la implementación de metaheurísticas de tal manera que la curva de tiempo y costo implicado en el desarrollo sea mínima.

ParadisEO

ParadisEO [11] es un marco de trabajo que separa la lógica genérica de las metaheurísticas, del problema que se pretende resolver. Esta separación y la gran variedad de funciones de optimización aplicadas permiten una máxima reutilización de código y de diseño. ParadisEO está desarrollado en C++, y es un marco de código abierto. Es compatible con Unix, Linux, MacOS y Windows e incluye el siguiente conjunto de módulos:

- Objetos Evolutivos (EO): Esta librería ha sido desarrollada inicialmente para los algoritmos evolutivos (algoritmos genéticos, estrategias evolutivas, programación evolutiva, programación genética, algoritmos de estimación y distribución).
- Objetos con Movimiento (MO): Incluye soluciones simples basadas en metaheurísticas tales como búsqueda local, recocido simulado, búsqueda tabú y búsqueda local iterada.
- Objetos Multiobjetivos (MOEO): Incluye los mecanismos de búsqueda para resolver problemas de optimización multiobjetivo. Están disponibles los algoritmos como NSGA-II, IBEA y SPEA2.
- Objetos Paralelos Evolutivos (PEO): Incluye herramientas para metaheurísticas paralelas y distribuidas: evaluación paralela, función de evaluación paralela, diseño de distribución e hibridación.

Un aspecto importante de ParadisEO es la definición de sus componentes, ya que todos se encuentran definidos en *plantillas* (clases). El usuario implementa una metaheurística en base a plantillas que proveen la funcionalidad a los diferentes componentes del problema.

Para la implementación se hizo uso del Modulo MO, el cual incluye genéricamente el algoritmo y componentes de la Búsqueda Tabú. Si bien la implementación no explota ampliamente otros módulos de ParadisEO, da una clara visión del modelado y representación de sus componentes. Un usuario experto puede extender sin dificultad las plantillas disponibles, listas para adaptarse a su problema y obtener más eficacia en sus métodos. Sin embargo, ParadisEO-MO puede ser utilizado por principiantes, con un mínimo de código para producir diversas estrategias de búsqueda.

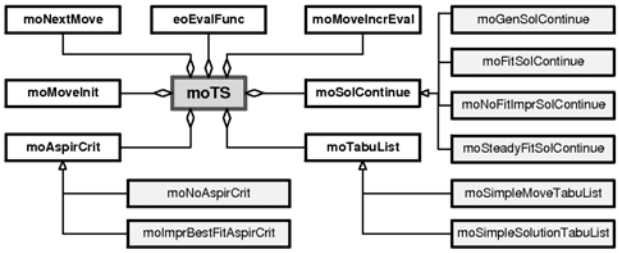

Figura 6: Diagrama UML de la plantilla de Búsqueda Tabú (moTS).

En base al esquema UML (por sus siglas en inglés, *Unified Modeling Language*) de la Figura 6, se implementa TS para la solución de QAP, segmentos importantes del código, se debe a la contribución de los desarrolladores de *INRA ParadisEO* [12] quienes implementaron de manera eficiente, la lógica y características descritas en el modelo matemático para la solución de QAP.

El código que fue utilizado para la experimentación computacional puede ser consultado en sitio web [13]. Para su análisis se recomienda que el lector tenga claro los conceptos de POO y conocimientos en programación en C/C++, así como la documentación de la API de ParadisEO [12] siempre presente para la consulta de términos y/o definiciones de clases que el código fuente utiliza y de esta manera tener una clara compresión de la implementación.

Experimentación

El objetivo del experimento es ilustrar el comportamiento de la metaheurística TS así como su sensibilidad al cambio del tamaño de la lista tabú.

Condiciones de la experimentación: El equipo de computo cuenta con las siguientes características: HP Pavilion DV5-1135 Portátil, AMD Turion X2-64 Dual Core 2.2, 3 GiB RAM, Sistema Operativo Ubuntu 9.04, Linux 2.6.28-15.

Las instancias de prueba son tomadas de QAPLIB (por sus siglas en inglés, *Quadratic Assignment Problem Library*) [3] cuya primera publicación data de 1991 y sigue siendo hoy en día el repositorio de instancias de QAP más reconocido en la comunidad científica. Las instancias utilizadas en este experimento son del grupo de É.D. Taillard clase *A*, con tamaños de n=10 a n=100. Los parámetros para la ejecución son de un valor fijo de *maxi_itera*=5000 que representa el máximo número de iteraciones, utilizado como *criterio de parada*.

Se pretende evaluar el desempeño de TS para diferentes valores fijos del tamaño de la lista tabú. Se probó con *size_tabu*=5, 10, 15, 20 y 25. Para determinar la calidad de la solución encontrada es calculado el *gap* que se define como el intervalo relativo entre la solución reportada por el algoritmo y la mejor solución conocida cuya fórmula de cálculo está definida en la Ecuación (4).

$$gap\% = \frac{s_{a\lg} - s_{opt}}{s_{opt}} \quad (4)$$

Donde $s_{a\lg}$ es la solución reportada por el algoritmo y s_{opt} es la mejor solución conocida para la instancia.

Resultados computacionales: Los resultados de la experimentación para cada instancia pueden ser consultados en el sitio web: *http://yalma.fime.uanl.mx/~roger/ftp/paradiseo*. En el despliegue de los resultados se ha omitido *size_tabu*=25 dado que los valores reportados son prácticamente similares a *size_tabu*=20.

La Figura 7 muestra el tiempo de CPU (*t*) en segundos y la Figura 8 el intervalo *gap*.

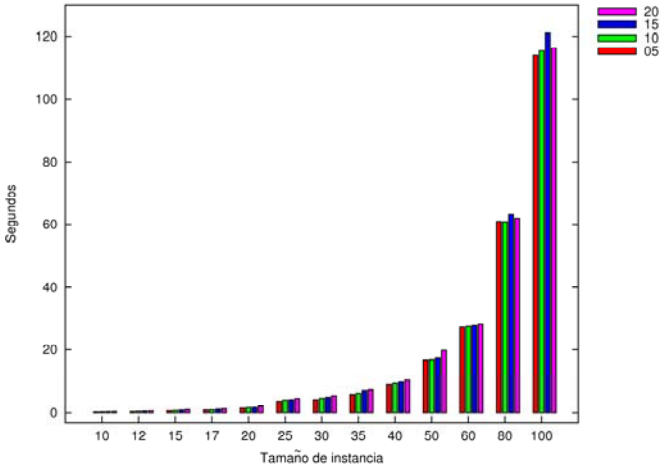

Figura 7: Variación de tiempo computable.

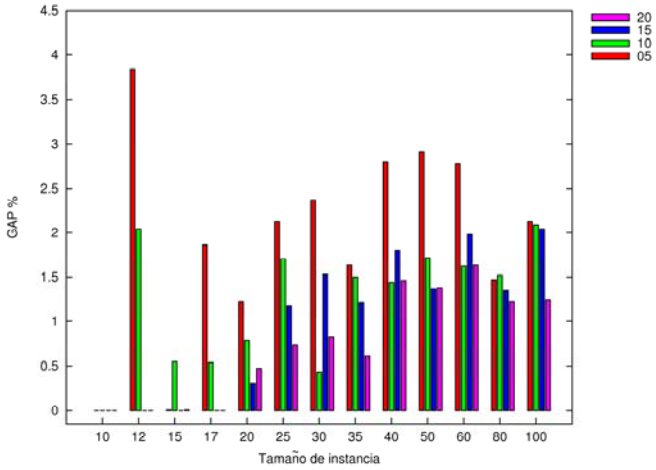

Figura 8: Variación del *gap %*.

En la Figura 9 se puede observar claramente el comportamiento típico de TS durante la solución de la instancia *tai12a*. Se muestran los puntos *A*, *B* y *C*; *A* indica la solución inicial, *B* y *C* representan los puntos relevantes durante el trayecto de la búsqueda. *B*, considerado un óptimo local, obtiene una mejora del 20.21% respecto al punto *A*, mientras que *C* el óptimo global del trayecto mejora un 6.95% respecto a *B* y un 25.75% respecto a *A*.

Cabe mencionar que de haber utilizado una búsqueda local ordinaria esta hubiera determinado a *B* como la solución mejor encontrada dada su incapacidad para seguir buscando en el espacio de soluciones, sin embargo TS *escapa* de estos óptimos locales, lo que permitió seguir analizando en un espacio de soluciones más amplio hasta finalmente llegar a la solución del punto *C* que no pudo ser mejorada por ninguna otra solución en el trayecto. Para llegar a *C* se necesitaron aproximadamente 90 iteraciones más allá de *B*.

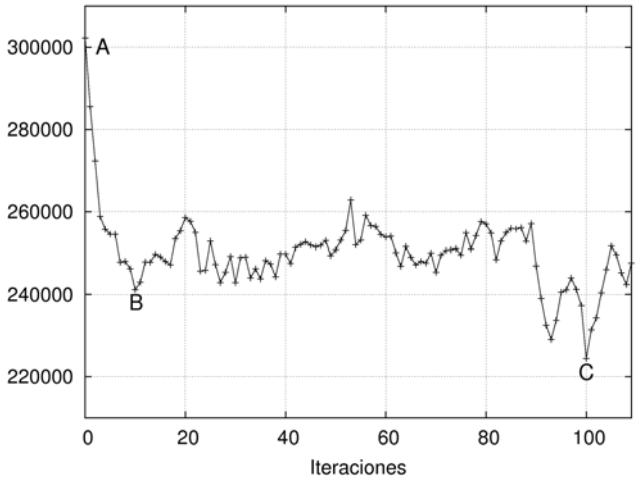

Figura 9: Comportamiento de TS en la solución de QAP.

Conclusiones

Es apreciable en la Figura 8, que al incrementar el tamaño de la lista tabú, para instancias de tamaño menor a 30 el *gap* tiende a mejorar, esto es debido a la lista tabú que mantiene a las soluciones por más iteraciones lo cual permite generar vecindarios con mayor diversidad de soluciones, evitando vecindarios previamente generados. Sin embargo para instancias de tamaño mayor a 30, la mejora respecto al incremento de la lista tabú es poco notable.

TS ciertamente no es la mejor opción a aplicar en la solución de QAP, argumentando que la calidad de sus resultados está por debajo de las mejores soluciones encontradas con metaheurísticas más sofisticadas, aunque la diferencia es tan solo notable cuando el tamaño de la instancia de QAP incrementa de manera considerable. Sin embargo es posible mejorar el desempeño de la implementación reforzando los siguientes puntos:

- Sustituir una solución inicial basada en aleatoriedad por un algoritmo de fase constructiva que determine una mejor solución inicial.
- Cambiar el esquema de la lista tabú de estático a dinámico, el cual toma en cuenta el tamaño de la instancia.

Durante el diseño de una metaheurística se deben de tomar en consideración varios puntos previos a la implementación, como la escalabilidad y flexibilidad. ParadisEO es un marco que permite dotar de complejidad y flexibilidad a las implementaciones que desarrollemos, siempre haciendo énfasis que el éxito de todo desarrollo depende inherentemente de una organizada planificación; tomando en cuenta la complejidad en la estrategia de búsqueda que se pretenda aplicar, se decidirá qué rumbo tomar durante la implementación, es decir si es conveniente realizar un desarrollo *desde cero* o bien tomar opciones como ParadisEO siempre y cuando satisfaga cada uno de los requerimientos del problema.

Agradecimientos: El primer autor fue apoyado por la Academia Mexicana de Ciencias a través de una beca otorgada dentro del XIX Verano de Investigación Científica durante el año 2009. Agradecemos también las críticas y comentarios de dos revisores anónimos que ayudaron a mejorar la presentación del trabajo.

Referencias

[1] M. Beckmann y T. Koopmans. Assignment problems and the location of economic activities. *Econometrica*, 25(1):53–76, 1957.

[2] E. Burkard, E. Çela, M. Pardalos y S. Pitsoulis. The quadratic assignment problem. En D.Z. Du y P. M.

Pardalos, editores, *Handbook of Combinatorial Optimization*, volumen 3, págs. 241–337. Kluwer, Boston, EUA, 1998.

[3] E. Burkard, S. Karisch y F. Rendl. QAPLIB - A Quadratic Assignment Problem Library. *Journal of Global Optimization*, 10(4):391–403, 1997. `http://www.opt.math.tu-graz.ac.at/qaplib/`

[4] A. Díaz, F. Glover, H. M. Ghaziri, J. L. González, M. Laguna, P. Moscato y F. T. Tseng. *Optimización Heurística y Redes Neuronales*. Paraninfo, Madrid, España, 1996.

[5] F. Glover. Future paths for integer programming. *Computers & Operations Research*, 13(5):533–549, 1986.

[6] F. Glover y M. Laguna. *Tabu Search*. Kluwer, Boston, EUA, 1997.

[7] R. Martí. *Algoritmos heurísticos en optimización combinatoria*. Departamento de Estadística e Investigación Operativa, Facultad de Ciencias Matemáticas. Universidad de Valencia, España, 2003.

[8] S. Sahni y T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

[9] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, 2(1):33–45, 1990.

[10] É. Taillard. Comparison of iterative searches for the quadratic assignament problem. *Location Science*, 3(2):87–105, 1995.

[11] E. Talbi. *Metaheuristics from Design to Implementation*. Wiley, Boston, EUA, 2009.

[12] E. Talbi, J. Boisson, J. Humeau, T. Legrand, A. Liefooghe, L. Jourdan, N. Melab, A. Tantar, M. Fatene, T. Luong y A. Khanafer. INRIA ParadisEO. `http://paradiseo.gforge.inria.fr`.