REVISTA DE MÉTODOS CUANTITATIVOS PARA LA ECONOMÍA Y LA EMPRESA (1). Páginas 25–37.
Junio de 2006. ISSN: 1886-516X. D.L: SE-2927-06.
URL: http://www.upo.es/RevMetCuant/art2.pdf

Algoritmo Tabú para un problema de distribución de espacios

G. Hernández-Díaz, Alfredo
Departamento de Economía, Métodos Cuantitativos e Historia Económica.
Universidad Pablo de Olavide
Correo electrónico: agarher@upo.es

Guerrero Casas, Flor M.
Departamento de Economía, Métodos Cuantitativos e Historia Económica.
Universidad Pablo de Olavide
Correo electrónico: fguecas@upo.es

Caballero Fernández, Rafael
Departamento de Economía Aplicada (Matemáticas). Universidad de Málaga
Correo electrónico: r_caballero@uma.es

Molina Luque, Julián
Departamento de Economía Aplicada (Matemáticas). Universidad de Málaga
Correo electrónico: julian_molina@uma.es

RESUMEN

La distribución de espacios es un problema que habitualmente se presenta en situaciones reales cuando se deben asignar simultáneamente diferentes conjuntos de espacios (despachos, habitaciones, salas, etc.) distribuidos entre edificios y/o plantas entre varios grupos de personas de tal forma que se minimicen las distancias entre los espacios asignados a cada grupo y la sede de dicho grupo. Esta situación da lugar a un problema combinatorio con una función objetivo cuadrática, lo cual complica enormemente su resolución mediante un método exacto. Por este motivo, proponemos para su resolución un metaheurístico basado en Búsqueda Tabú con dos grupos de movimientos claramente diferenciados: intercambio de despachos y reasignación de sedes. Finalmente, aplicamos dicho algoritmo a un caso real en la Universidad Pablo de Olavide de Sevilla (España).

Palabras clave: búsqueda tabú; problemas de asignación.
Clasificación JEL: C61; C63.
2000MSC: 90C59; 90C27; 90C20; 90C90.

Artículo recibido el 20 de abril de 2006 y aceptado el 12 de mayo de 2006.

Tabu Search Algorithm for a Room Allocation Problem

ABSTRACT

The distribution of spaces is a usual real problem presented when we have to assign simultaneously different sets of spaces (offices, rooms, halls, etc.). These spaces are distributed in buildings and/or floors and have to be assigned among several groups of people. The aim is to minimize the total distance among the spaces assigned to each group and its head office. This situation drives us to a quadratic combinatorial problem, so difficult to solve with exact methods. This is the reason to propose a metaheuristic method to solve it, a Tabu Search algorithm with two types of movements: the swapping of two offices and further assignment of head offices. The performance of the algorithm is demonstrated on a problem related with the Pablo de Olavide University in Seville (Spain).

Keywords: tabu search; room allocation problems.
JEL classification: C61; C63.
2000MSC: 90C59; 90C27; 90C20; 90C90.

1. Introducción y formulación

El problema de distribuir despachos entre grupos de personas simultáneamente es una tarea compleja dado que cada grupo valorará los espacios que le han sido asignadas en función de la *sede* (planta en la que se encuentra la mayoría del grupo) que le haya sido también asignada. Así, el problema es de una complejidad superior dado que debemos resolver dos problemas de asignación simultáneamente. Por un lado, buscamos el reparto óptimo de las sedes y por otro, una vez conocida ésta, la distribución óptima de los espacios. Por tanto, una solución posible debe contemplar que todos los grupos tengan asignado una planta como sede, pudiendo ocurrir que varios grupos compartan sede, que ninguna habitación sea asignada a más de un grupo y que se satisfagan las demandas de los grupos, es decir, que el total de asignaciones por grupo coincida con el tamaño de éste.

Así, la formulación del problema queda de la siguiente forma:

$$\text{Min.} \sum_G \sum_H \sum_P w_{g,h}^p X_{p,g} Y_{h,g} \tag{1}$$

$$\text{s.a.} \quad \sum_P X_{p,g} = 1, \text{ para cada } g \in G, \tag{2}$$

$$\sum_G Y_{h,g} \leq 1, \text{ para cada } h \in H, \tag{3}$$

$$\sum_H Y_{h,g} = c_g, \text{ para cada } g \in G, \tag{4}$$

$$X_{p,g}, Y_{h,g} \in \{0,1\}, \tag{5}$$

donde los conjuntos G, H y P indexan al número de grupos, de despachos y de plantas respectivamente, la variables binarias $X_{p,g}$ e $Y_{h,g}$ se definen como sigue:

$$X_{p,g} = \begin{cases} 1, & \textit{si al grupo } g \textit{ se le asigna la planta } p \textit{ como sede}, \\ 0, & \textit{en caso contrario}, \end{cases}$$

e

$$Y_{h,g} = \begin{cases} 1, & \textit{si la habitación } h \textit{ es asignada al grupo } g, \\ 0, & \textit{en caso contrario}, \end{cases}$$

$w_{g,h}^p$ representa la valoración que le da el grupo g a la habitación h si su sede es la planta p, y c_g es el número de personas que compone el grupo g. En nuestro caso, $w_{g,h}^p$ representará la distancia (por ejemplo, euclídea) desde la habitación h hasta la planta p en la que se encuentra la sede del grupo g. Este es el motivo por el que el problema está planteado como de minimización.

Así, la ecuación (1) representa el objetivo a minimizar, esto es, la suma total de todas las valoraciones que los grupos han considerado para cada una de los espacios asignados; la restricción (2) obliga a que todos los grupos tengan asignada una única sede; la ecuación (3) imposibilita que una habitación sea asignada a más de un grupo, pudiendo ocurrir además que una habitación quedase sin asignar (tomando la suma el valor cero); mientras que (4) fuerza a que el número de espacios asignados a cada grupo coincida con el número de miembros que lo forman.

Lógicamente, el problema resultaría infactible si la demanda superara a la oferta, es decir, si:

$$\sum_{G} c_g > |H|.$$

Por este motivo, en todo momento supondremos que la demanda es inferior o igual a la oferta.

En resumen, el modelo planteado es un problema combinatorio de minimización con función objetivo cuadrática. Este problema no corresponde exactamente a ninguno de los problemas clásicos de optimización combinatoria en la literatura, pero presenta cierta similitud (en cuanto a la formulación y en cuanto a la situación real a resolver) con los problemas de asignación de espacios, motivo por el cual en el siguiente epígrafe haremos una breve introducción a este tipo de problemas.

2. Problemas de Asignación de Espacios

Habitualmente nos enfrentamos a problemas de reparto de muy diversa índole (personas, tareas, dinero, objetos,…). Dichos problemas, si no son excesivamente grandes, los solemos resolver manualmente usando nuestra razón. Pero, cuando el tamaño de dichos problemas excede de nuestra capacidad, nos vemos obligados a utilizar herramientas analíticas que nos faciliten su resolución. Es por ello por lo que desde hace varias décadas se vienen estudiando este tipo de problemas dentro del campo de la Programación Matemática y la Investigación Operativa dentro de la rama comúnmente conocida como Problemas de Asignación (allocation problems).

En particular, el problema de la asignación de espacios ha sido ampliamente estudiado en los últimos años. Así, por ejemplo, Carter (1989) y Carter y Laporte (en Burke y Carter, 1998) trabajan con la problemática de asignar aulas a las distintas asignaturas de una universidad, López García et al. (2002) estudian la distribución óptima de salones entre los múltiples cursos de verano en una universidad y Ciriani et al. (2004) proponen un algoritmo para el reparto de despachos de un pasillo. Aunque en cada uno de estos problemas se trabaja con un conjunto de restricciones específico, la base de la gran mayoría de estos problemas combinatorios es el de la "concentración", es decir, cada grupo (ya sea de alumnos, de conferencias o de personas de un departamento) busca que los elementos que le han sido asignados (aulas, salas de conferencias o despachos) estén lo más próximos posible unos de otros. No obstante, tal como se ha comentado antes, ninguno de los trabajos anteriores es exactamente igual al propuesto y por ello no se pueden comparar las soluciones obtenidas con otros métodos o algoritmo de la literatura.

Debido a que la mayoría de los problemas de este tipo son NP-completos (véase Garey y Johnson (1975) para más detalles), algunos de los métodos exactos se han demostrado poco eficientes y lentos (computacionalmente hablado). Es por esto por lo que los algoritmos metaheurísticos están siendo utilizados cada día más para este tipo de problemas de asignación y otros de similares características. Distintos procedimientos de búsqueda como la Búsqueda Tabú (Laguna et al. (1995), Díaz y Fernández (2001), Kelly et al. (1994) o López García et al. (2003)), los Algoritmos Genéticos (Wang at al. (2002)), el Temple Simulado (Herault y Privault (1998)), concatenación de movimientos simples y *path relinking* (Yagiura et al. (2004)), *logic cuts* (Osorio y Laguna (2003)) o los basados en Colonias de Hormigas (Middendorf et al. (2002)) se están utilizando con gran éxito en estos tipos de problemas. Más recientemente, los

problemas de asignación están siendo estudiados con objetivos múltiples utilizando también algoritmos metaheurísticos (Gandibleux et al. (2003) y Gandibleux et al. (2004)).

Las siguientes secciones están estructuradas de la siguiente manera. En la Sección 3 recordamos los conceptos básicos de la Búsqueda Tabú y en la Sección 4 exponemos con detalle el algoritmo que proponemos. En la Sección 5 contrastamos el algoritmo propuesto con 80 instancias generadas aleatoriamente para este problema. Finalmente, en la Sección 6 aplicamos dicho algoritmo al caso real de la Universidad Pablo de Olavide y en la Sección 7 presentamos las conclusiones.

3. La Búsqueda Tabú

La búsqueda tabú (TS) es un procedimiento heurístico de memoria adaptativa propuesto por Glover en 1986 (Glover, 1986) para la búsqueda de los óptimos globales (resp., soluciones eficientes) en problemas de optimización mono-objetivo (resp., multiobjetivo). TS está siendo aplicado exitosamente en las últimas décadas en multitud de problemas complejos (continuos y discretos, lineales y no lineales, convexos y no convexos, etc.) en diferentes áreas científicas (tal y como se puede ver en Glover y Laguna (1997)).

TS explora el espacio de soluciones a través de repetidos movimientos desde una solución a la mejor de sus vecinas tratando de evitar los óptimos locales. Para un problema mono-objetivo, TS realiza una búsqueda por entornos en la cual se desplaza en cada iteración a la mejor solución no tabú del vecindario de la solución actual. Los principales atributos de cada solución visitada son almacenados en una lista tabú por un determinado número de iteraciones para evitar que estas soluciones sean revisitadas, es decir, para evitar ciclos en la búsqueda por entornos. Así, un elemento del vecindario de la solución actual es declarado tabú (es decir, es prohibido) si alguno de sus atributos está en la lista tabú. En general, un método basado en búsqueda tabú requiere de los siguientes elementos:

1. Solución inicial. La búsqueda debe comenzar desde una solución inicial que podría ser cualquier solución admisible que satisfaga las restricciones del problema. Una buena solución inicial podría acelerar la búsqueda con el consiguiente ahorro de tiempo. Dicha solución puede ser generada aleatoriamente o utilizando *funciones ávidas* o *greedy functions* (funciones que incorporan información adicional del problema utilizadas como estrategias para generar puntos de mejor calidad).

2. Movimiento. Un movimiento es un procedimiento aleatorio o determinístico por el que se genera una solución admisible a partir de la solución inicial. Usualmente, este procedimiento es sencillo para el caso de problemas combinatorios, pero mucho más complejo para el caso de problemas de optimización continuos.

3. Vecindad. Dada una solución S, la vecindad N(S) es el conjunto de todas la soluciones admisibles que pueden ser generadas por la ejecución de un movimiento sobre la solución actual S. Este conjunto suele ser numerables para problemas combinatorios y, en aquellos casos en los que N(S) sea grande, se suele operar con un subconjunto de

éste. Para problemas continuos, los posibles vecinos son no numerables y se debe ser más creativo para definir N(S).

4. Lista tabú. Es un mecanismo de memoria adaptativa que trata de evitar que la búsqueda entre en un ciclo o quede atrapada en un óptimo local. Una vez que un movimiento, que genera una nueva solución, es aceptado, su movimiento inverso se añade a la lista tabú y permanece en ésta un número determinado de iteraciones. Si el tamaño de la lista tabú es pequeño, entonces la búsqueda se intensifica en una determinada área del espacio, mientras que si el tamaño de la lista es grande se enfatiza la búsqueda en diferentes regiones del espacio de soluciones.

5. Criterio de parada. En general, la búsqueda termina después de un número determinado de iteraciones, después de un tiempo de computación predefinido o cuando se alcaza un número dado de iteraciones sin mejorar la mejor solución.

No obstante, dentro de un TS se puede encontrar una gran variedad de estrategias destinadas a mejorar la búsqueda como, por ejemplo, criterios de aspiración, que admite movimientos tabú si se satisface el criterio de aspiración (por ejemplo, se mejora el valor de la función objetivo de la mejor solución encontrada hasta ese momento) con idea de cruzar las barreras impuestas en las restricciones tratando de encontrar otras zonas factibles más prometedoras; fases de intensificación, que permite concentrar la búsqueda en aquellas zonas más prometedoras; fase de diversificación, que permite desplazarse hacia zonas no exploradas, etc. Una referencia básica acerca de TS se puede encontrar en Glover y Laguna (1997).

4. Algoritmo propuesto

A continuación mostramos los detalles del heurístico propuesto para resolver el problema planteado. El algoritmo que presentamos se basa en una búsqueda local mediante el uso de una lista tabú que inicializamos apropiadamente. El pseudocódigo se muestra en el siguiente esquema:

Estructura básica del algoritmo

- Generar una solución inicial s_0.
- Inicializamos la solución y la lista tabú: $s_{final} = s_0$, $T=\{s_0\}$.
- Fase de acercamiento: En cada iteración t:
Repetir (hasta condición de parada):
1. Generar vecindario tipo 1, $N(s_t)$, y eliminamos las soluciones tabú, $N(s_t)=N(s_t)-T$.
2. s_{t+1} es seleccionado verificando $f(s_{t+1}) = \min\{f(s) : s \in N(s_t)\}$.
3. Si $f(s_{t+1}) < f(s_{final})$, $s_{final} = s_{t+1}$.
4. Actualizar T.
Fin acercamiento.
- Fase de Intensificación:
Repetir (hasta condición de parada): En cada iteración t:
1. Generar un vecino de tipo 2, $N(s_t)$.
2. Aplicamos una Fase de Acercamiento.
Fin Intensificación.

Pasamos a continuación a describir las características específicas del algoritmo.

4.1. Generación de la solución inicial

Esta generación se realiza a través de un procedimiento ávido. Así, asignamos las sedes a los distintos grupos bajo criterios de volumen de los grupos y capacidad de los edificios. Concretamente, ordenamos los grupos en función del tamaño de mayor a menor, y siguiendo este orden, asignamos al grupo la planta con menor holgura por exceso, y si esto no es posible, la planta con menor holgura por defecto. A continuación, y respetando el orden anterior, vamos asignados los distintos espacios a cada uno de los miembros de cada grupo. Como veremos en secciones posteriores, este semillado es muy efectivo y, en algunas de las instancias, está muy próxima a la solución óptima final.

4.2. Fase de acercamiento

- Vecino tipo 1. En esta fase se intentará mejorar el valor de la función objetivo realizando intercambios del tipo 1-1, es decir, intercambiando o reasignado a dos individuos mal colocados (es decir, individuos colocados en una planta o edificio distinto de la sede de su grupo). Fijamos a un individuo mal colocado y lo incluimos en la lista tabú (es decir, no volverá a ser considerado un cierto número de iteraciones). A continuación, distinguimos tres tipos de movimientos posibles: (a) si hubiese sitio en su sede, lo reasignamos a su sede, (b) si la planta de su sede está llena pero hubiese un individuo de otro grupo mal colocado ocupando una habitación, los intercambiamos, (c) si la planta de su sede está llena de individuos bien colocados, entonces buscamos acercarlo a su sede mediante un intercambio con otro individuo también mal colocado.

4.3. Fase de Intensificación

En esta fase probamos a mejorar la solución encontrada cambiando aleatoriamente las asignaciones de las sedes iniciales, para luego reparar la solución de una forma acorde a esta nueva distribución de las sedes de cada grupo. Para ello, generaremos el siguiente tipo de vecinos:

-Vecino tipo 2. Generamos aleatoriamente un número N entre 1 y el número de grupos y reasignamos aleatoriamente las sedes a N de los grupos elegidos aleatoriamente.

De esta manera damos un salto dentro del espacio de las soluciones, pero la mayoría de los individuos estarán mal colocados. Es por esto que a continuación realizamos una búsqueda intensiva para mejorar el punto actual en el que nos encontramos. Es decir, se procede a recolocar (utilizando el esquema de movimientos de la fase de Acercamiento) los individuos que tras esta reasignación de las sedes han quedado mal colocados.

4.4. Criterios de parada

Con idea de que el algoritmo sea lo más autónomo posible, hemos utilizado criterios de parada auto-ajustables dependiendo de la oferta y la demanda. En primer lugar, el criterio de parada para la fase de acercamiento se reajusta en cada ciclo con un valor igual al número total de miembros mal colocados (siempre que el valor de la función objetivo no sea cero). De esta forma, buscamos mejorar la solución encontrada redistribuyendo a los individuos mal colocados. Para la fase de intensificación, y dado

que la holgura juega un papel relevante en el problema, tomamos como criterio de parada que el número de iteraciones no supere a:

$$\frac{Departamentos * Plantas}{Oferta - Demanda + 1}$$

(también si el valor de la función objetivo no es cero).

5. Implementación y resultados

El algoritmo presentado se implementó bajo un entorno Windows utilizando el lenguaje C++ en su versión 6.0. Dicha implementación se ha utilizado en un PC de sobremesa con el sistema operativo Windows XP con un microprocesador de Intel Pentium a 1.5 GHz. y una memoria RAM de 512 MB.

Para este problema no existen instancias en la literatura ni otros métodos con los que comparar resultados (cada uno de los trabajos comentado en la Sección 2 incorpora restricciones distintas que lo hacen diferente al nuestro). Por este motivo se ha generado un conjunto de instancias (que se describen a continuación) y que se han resuelto, aunque no se ha comparado con ningún otro método al no existir otro método por el momento para este mismo problema.

Así, para comprobar la eficiencia del algoritmo se han generado 80 instancias clasificadas en cuatro conjuntos que describimos a continuación. Si denotamos por *m* al número de grupos que debemos distribuir, por *n* al número de plantas o edificios disponibles y por *s* a la holgura total de la que se dispone, es decir, la diferencia entre la oferta total y la demanda total, denotaremos por *m_n_s* a la instancia generada para ubicar a *m* grupos en las *n* plantas (o edificios) con una holgura total de *s* espacios. Como coeficientes $w_{g,h}^{p}$ tomaremos la distancia euclídea del edificio donde se encuentra el despacho *h* al edificio *p* si éste fuese la sede del departamento *g,* una vez que asignamos aleatoriamente coordenadas en el plano a cada edificio. Los cuatro grupos de instancias generados van aumentando de tamaño, y por tanto de dificultad, en los parámetros *m* y *n*. Dentro de cada grupo de instancias hacemos variar el parámetro *s* desde 0 hasta 19. Así, los cuatro grupos de instancias corresponden a los valores (*m,n*) = (10,5), (*m,n*) = (15,7), (*m,n*) = (20,10) y (*m,n*) = (25,12).

Los resultados para cada grupo de instancias se presentan en las tablas 1, 2, 3 y 4, respectivamente. En la columna *Valor alcanzado* mostramos el mejor valor de la función objetivo obtenido por el algoritmo, en *Mal colocados* mostramos el porcentaje de individuos que finalmente quedan mal colocados (fuera de sus sedes), en *Tiempo mejor* mostramos el tiempo, en segundos, en el que el algoritmo alcanza el mejor valor y en *Tiempo total* el tiempo de ejecución del algoritmo.

Instancia	Valor alcanzado	Mal colocados (%)	Tiempo mejor	Tiempo total
10_5_0	5.012	2	0.64	2.84
10_5_1	1	1	0.61	1.71
10_5_2	1.341	3	0.56	1.33
10_5_3	0.632	0.5	0.41	0.99
10_5_4	2.166	4	0.37	0.86
10_5_5	2.088	2	0.36	0.67
10_5_6	3.298	4	0.52	0.82
10_5_7	1.253	0.5	0	0.87
10_5_8	3.893	4.5	0.62	0.78
10_5_9	0	0	0	0.38
10_5_10	0	0	0	0.50
10_5_11	0	0	0	0.66
10_5_12	2.8	3.5	0.48	0.52
10_5_13	0	0	0	0.53
10_5_14	0	0	0	0.43
10_5_15	3.354	2.5	0.46	0.62
10_5_16	0	0	0	0.64
10_5_17	0	0	0	0.72
10_5_18	1.104	0.5	0.49	0.60
10_5_19	0	0	0	0.38

Tabla 1. Resultados obtenidos para el conjunto de instancias *(m,n)=(10,5)*.

En la Tabla 1 se aprecia, como era esperable, que el valor alcanzado tiende a mejorar a medida que la holgura, *s*, aumenta. Es más, en algunos problemas el valor de la función objetivo conseguido es cero, obteniéndose por tanto un 100% de éxito en las asignaciones de los despachos. No obstante, el valor alcanzado puede empeorar para valores de *s* grandes debido a que otros parámetros como el número de despachos por planta es también determinante a la hora de obtener un reparto óptimo de los espacios disponibles.

Instancia	Valor alcanzado	Mal colocados (%)	Tiempo mejor	Tiempo total
15_7_0	4.539	2	7.83	13.23
15_7_1	2.148	1.33	0.37	6.19
15_7_2	3.267	1	0.47	4.27
15_7_3	1.683	1	0.82	3.11
15_7_4	0.962	1.33	0.51	2.94
15_7_5	0.6	0.33	0.37	2.02
15_7_6	6.974	3	0.56	1.88
15_7_7	2.543	1.33	0.42	2.08
15_7_8	0.761	0.33	0.30	1.41
15_7_9	0	0	0	0.43
15_7_10	0.5	0.33	0.5	1.93
15_7_11	0	0	0	0.87
15_7_12	0	0	0	0.52
15_7_13	0	0	0	0.80
15_7_14	0	0	0	0.55
15_7_15	0	0	0	1.06
15_7_16	0	0	0	0.36
15_7_17	0	0	0	0.46
15_7_18	1.8	3	0.76	0.91
15_7_19	0	0	0	0.46

Tabla 2. Resultados obtenidos para el conjunto de instancias *(m,n)=(15,7)*.

De nuevo, en la Tabla 2 se aprecian comportamientos similares a los obtenidos anteriormente. No obstante, merece la pena destacar que en esta ocasión los tiempos

requeridos son ligeramente superiores debido a la mayor dificultad de éstos (esto se aprecia con mayor claridad en las instancias de las Tablas 3 y 4).

Instancia	Valor alcanzado	Mal colocados (%)	Tiempo mejor	Tiempo total
20_10_0	13.466	11	19.09	85.41
20_10_1	5.683	2.25	8.80	32.04
20_10_2	5.375	3.25	17.02	21.28
20_10_3	6.200	3	11.01	13.94
20_10_4	3.002	3	4.32	11.2
20_10_5	10.136	6.5	2.55	11.33
20_10_6	3.4	2.5	0.46	8.49
20_10_7	0.6	1.5	0.4	6.058
20_10_8	1.788	2	0.42	6.32
20_10_9	1.7	1.25	0.44	5.57
20_10_10	8.9	4.75	0.38	6.64
20_10_11	13.336	5.5	0.46	6.48
20_10_12	0.9	9.5	3.54	5.41
20_10_13	0	0	0	0.32
20_10_14	5.4	2.25	0.56	3.99
20_10_15	2	4	0.49	3.95
20_10_16	0	0	0	0.41
20_10_17	0	0	0	0.31
20_10_18	0	0	0	0.45
20_10_19	10.351	6.25	1.06	4.09

Tabla 3. Resultados obtenidos para el conjunto de instancias *(m,n)=(20,10)*.

Instancia	Valor alcanzado	Mal colocados (%)	Tiempo mejor	Tiempo total
25_12_0	7.447	1.8	100.9	166.06
25_12_1	12.148	5.2	39.67	86.073
25_12_2	3.713	1.2	9.07	53.907
25_12_3	4.456	4.2	9.61	42.47
25_12_4	5.745	5.6	4.17	36.73
25_12_5	32.676	14.2	3.27	42.45
25_12_6	4.967	3	16.78	21.67
25_12_7	12.308	5.2	13.10	21.79
25_12_8	2.909	1	0.63	14.8
25_12_9	10.044	4.4	4.12	18.57
25_12_10	4.603	3.4	0.43	15.93
25_12_11	1.341	1.2	0.57	11.92
25_12_12	1.2	0.8	0.50	12.76
25_12_13	0.1	0.2	0.28	10.70
25_12_14	4.110	2.6	0.3	13.17
25_12_15	5.735	3.6	10.49	10.62
25_12_16	0	0	0	0.50
25_12_17	32.177	10.2	3.09	14.03
25_12_18	5.465	3.4	0.42	8.943
25_12_19	1.131	0.8	0.74	8.092

Tabla 4. Resultados obtenidos para el conjunto de instancias *(m,n)=(25,12)*.

Finalmente, en las Tablas 3 y 4 mostramos los resultados obtenidos para las instancias más complejas. Es apreciable tanto en los porcentajes de individuos mal colocados como en los tiempos requeridos para la obtención de la solución que a medida que el tamaño del problema crece, la complejidad aumenta debido a las múltiples combinaciones posibles. No obstante, hay que reseñar que el valor ideal deseable por todos los grupos (valor alcanzado = *0*) no siempre es posible alcanzarlo y que, por tanto, los porcentajes tan bajos obtenidos en individuos mal colocados son muy destacables.

Estos resultados nos parecen prometedores, tanto en calidad de las soluciones como en tiempo de computación. Sin embargo, como ya se ha señalado, no existen resultados previos con los que compararlos u otros métodos conocidos para resolver este problema.

6. Aplicación a la Universidad Pablo de Olavide de Sevilla

La Universidad Pablo de Olavide (UPO) se funda en 1997 en Sevilla (España) sobre los cimientos de la antigua Universidad Laboral. Inicialmente, la UPO contó con tres edificios rehabilitados donde, sin ningún criterio de concentración, se fueron alojando los profesores los primeros años. Es más, con idea de no beneficiar a ningún departamento, podíamos encontrar a profesores de cualquiera de los departamentos existentes prácticamente en todas las plantas disponibles para despachos. Varios años más tarde y una vez que otro grupo de edificios habían sido también rehabilitados para aulas y despachos, la Universidad se plantea redistribuir a su plantilla bajo el criterio de la mayor concentración posible, al menos para el profesorado que trabaja a tiempo completo en dicha Universidad. En ese momento, la UPO contaba con 6 edificios para la distribución del profesorado. Cada edificio cuenta con tres plantas para despachos y unos 25 puestos de trabajo de media por planta. Además, la UPO contaba con un total de 9 departamentos con un total aproximado de 425 profesores a tiempo completo. Finalmente, la holgura que conseguimos para nuestro problema fue únicamente de 2 despachos. Por tanto, nuestro problema era del tipo 9_6_2. Obsérvese que, dado que hay más departamentos que edificios, varios de los departamentos se verán obligados a compartir edificio.

Figura 1. Plano general de la Universidad Pablo de Olavide.

Dado que el criterio que se iba a seguir era el de la concentración del profesorado de un mismo departamento, la valoración que seguimos para cada uno de los despachos fue la de distancia a la sede. Así, $w^{p}_{g,h}$ representa la distancia del despacho h al edificio p si éste fuese la sede del departamento g.

Una vez introducidos los datos del problema, el algoritmo encontró una buena solución en 3.7 segundos, quedando la distribución de la siguiente manera: Los departamentos 1, 2, 3, 4, 5, 6 y 7 alcanzaron el 100% en la colocación del su profesorado en los edificios donde se encuentran sus sedes, mientras que los departamentos 8 y 9 alcanzaron el

71,05% y 75% respectivamente. De manera global, concluimos que el 96,94% del profesorado se encuentra en el edificio de su sede.

7. Conclusiones

Presentamos un algoritmo tabú para un problema de distribución de espacios en el que debemos distribuir habitaciones entre personas divididas en grupos de tal forma que los miembros de un mismo grupo estén lo más cerca posible a la sede de su grupo. Es por esto que la valoración que cada grupo dará a una habitación variará dependiendo de la sede de éste.

El algoritmo propuesto es un metaheurístico de búsqueda local basado en la búsqueda tabú. Intercalamos apropiadamente distintas fases de acercamiento e intensificación. En la fase de acercamiento consideramos movimientos simples para la mejora inmediata de la solución, mientras que para la fase de intensificación consideramos movimientos del tipo "cambio de sedes" para explorar otras estructuras sobre la base de una misma solución.

Contrastamos el algoritmo con 80 instancias cuya dificultad aumenta gradualmente y donde comprobamos que la eficiencia del algoritmo es alta. Lógicamente, mientras mayor holgura tenga el problema, es decir, mayor diferencia entre la oferta y la demanda, la solución que encontramos es de mejor calidad.

Finalmente, aplicamos el algoritmo al caso real que nos presenta la Universidad Pablo de Olavide de Sevilla para la óptima distribución del profesorado bajo el criterio de la concentración de éste de los departamentos que la componen. Aunque el problema tiene únicamente una holgura de dos despachos, el algoritmo nos proporciona una solución con un nivel de concentración global del profesorado del 96,24%.

Referencias

[1] Burke, E. y Carter, M., *Practice and Theory of Automated Timetabling*, Vol. II, Selected Papers of the First International Conference, Edinburgh, UK, Ed. Springer, 1998.

[2] Carter, M., *A Langangian Relaxation Approach to the Classroom Assignment Problem*, INFOR, 27, Nº 2, 1989.

[3] Ciriani, V., Pisanti, N. y Bernasconi, A., *Room allocation: a polynomial subcase of the quadratic assignment problem*, Discrete Applied Mathematics, 144, pp. 263-269, 2004.

[4] Díaz, J. A. y Fernández, E., *A Tabu search heuristic for the generalized assignment problem*, European Journal of Operational Research, 132, pp. 22-38, 2001.

[5] Gandibleux, X., Morita, H. y Katoh, N., *Use of a genetic heritage for solving the assignment problem with two objectives*. En Evolutionary Multi-Criterion Optimization (C. Fonseca, P. Fleming, E. Zitzler, K. Deb, L. Thiele Eds.). EMO 2003, Second International Conference, Faro, Portugal, April 2003 Proceedings. Lecture Notes in Computer Sciences 2632, pp. 43-57, Springer.

[6] Gandibleux, X., Morita, H. y Katoh, N., *A population-based metaheuristic for solving assignment problems with two objectives*, in Journal of Mathematical Modelling and Algorithms, por aparecer.

[7] M. Garey y D. Johnson, *Computers and Intractability*. New York: Freeman, 1979.

[8] Glover, F., *Future Paths for Integer Programming and links to artificial intelligence*, Computers & Operational Research, 5, pp. 533-549, 1986.
[9] Glover, F. y Laguna, M., *Tabu Search*, Kluwer Academic Publishers, Boston, 1997.
[10] Herault, L. y Privault, C., *Solving a Real World Assignment Problem with a Metaheuristic*, Journal of Heuristics, 4, pp. 383-398, 1998.
[11] Kelly, J. P., Laguna, M. y Glover, F., *A Study on Diversification Strategies for the Quadratic Assignment Problem*, Computers and Operations Research, 21, no. 8, pp. 885-893, 1994.
[12] Laguna, M., Kelly, J. P., González Velarde, J. L. y Glover, F., *Tabu Search for the Multilevel Generalized Assignment Problem*, European Journal of Operational Research, 82, pp. 176-189, 1995.
[13] López García, L., Pérez de la Torre, L., Rodríguez Romano, N. y Posada Bolívar, A., *El problema de asignación de salones: solución con la heurística Búsqueda Tabú*, Actas del 2º Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), Gijón, 2003.
[14] Middendorf, M., Freischle, F. y Schmeck, H., *Multi Colony Ant Algorithms*, Journal of Heuristics, 8, pp. 305-320, 2002.
[15] Osorio, M.A. y Laguna, M., *Logic Cuts for Multilevel Generalized Assignment Problems*, European Journal of Operational Research, 151, pp. 238-246, 2003.
[16] Yagiura, M., Iwasaki, S., Ibaraki, T. y Glover, F., *A very large-scale neighborhood search algorithm for the multi-resource generalized assignment problem*, Discrete Optimization, 1, pp. 87-98, 2004.
[17] Wang, Y.-Z., An application of genetic algorithm methods for teacher assignment problem, Expert Systems with Applications, 22, pp. 295-302, 2002.