Heurísticas adaptativas para el problema de asignación generalizada

Lourenço, H.R. (2002), Heurísticas adaptativas para el problema de asignación generalizada (Adaptive heuristics for the generalized assignment problem). In Proceedings of The First Spanish Congress in Evolutive and Bioinspired Algorithms, Merida, Spain, February 6-7, pp. 267-275.
ISBN 84-607-3913-9.

Heurísticas Adaptativas para el Problema de Asignación Generalizada

Helena Ramalhinho-Lourenço y Daniel Serra*

***Resumen*— Las decisiones sobre la asignación de recursos en diversos ámbitos son problemas altamente complejos y que requieren modelos y métodos sofisticados para su solución e implementación. Entre estos modelos, el Modelo de Asignación Generalizada consiste en asignar un conjunto de tareas a una serie de recursos con un coste total mínimo del sistema. Cada recurso tiene una capacidad limitada y cada tarea debe ser asignada a uno y solo a un recurso, requiriendo una cierta cantidad de esta capacidad. Este modelo tiene diversas aplicaciones en el ámbito industrial, logístico, sanitario, etc.. Se propone una heurística adaptativa basada en sistemas GRASP (*greedy randomized adaptive search heuristic*) y MMAS (*Max-Min Ant System*) para la resolución de este tipo de problemas. La ventaja de este método consiste en la fácil adaptación a nuevas restricciones o condiciones del problema al modelo y método de resolución, no siendo necesario reformular el problema. También se presentan resultados computacionales que demuestran que estos métodos son de los más eficientes de los conocidos hasta el momento, seguidas por observaciones concluyentes.**

Palabras clave— **Asignación de Recursos, Heurísticas adaptativas, GRASP, Algoritmos basados en Colonias de Hormigas.**

I. INTRODUCCIÓN

LAS decisiones sobre la asignación de recursos en distintos ámbitos de la gestión son problemas altamente complejos y que requieren modelos y métodos sofisticados para su solución e implementación. En particular, el **P**roblema de **A**signación **G**eneralizada (**GAP**) es un modelo cuantitativo muy utilizado para gestionar la asignación de recursos. Más concretamente, el GAP consiste en asignar un conjunto de tareas a una serie de recursos con capacidad limitada teniendo como objetivo obtener la asignación de coste total mínimo.

Nuestros intereses iniciales en este problema surgieron de aplicaciones reales en el área de servicios de salud, problemas de asignación de recursos de elevado coste. Además, el GAP tiene muchas otras aplicaciones, por ejemplo en áreas como informática y redes de comunicación, problemas de localización, rutas de vehículos y problemas de gestión de operaciones en producción.

El objetivo de este trabajo es presentar algoritmos híbridos que se basan en heurísticas constructivas y en la aplicación subsiguiente de la búsqueda local para resolver el GAP, y presentar los respectivos resultados computacionales. Los elementos básicos son obtenidos de una metaheurística conocida por "optimización con colonia de hormigas" *(Ant Colony Optimization Algorithm)* *MAX-MIN Ant System*, Stützle (1997,1998a,1998b), Stützle and Hoos (1999), y la metaheuristica GRASP, Feo and Resende (1995). Estas heurísticas pueden ser encajonadas en una estructura general con tres fases. En la primera fase, la solución es construida por una heurística construtiva; en la segunda fase, la búsqueda local es utilizada para mejorar esta solución inicial; y la última fase consiste en actualizar una serie de parámetros los cuales dirigen el proceso de construcción en la primera fase. Las tres fases son repetidas hasta que es verificado el criterio de parada. Las elecciones hechas en cada grado te llevan a diferentes métodos heurísticos. Para más información en metaheurística ver Adenso Días et al.(1996).

El documento está organizado del siguiente modo: Primero, presentamos el GAP y un análisis de los métodos propuestos para resolverlo. En la sección III., describimos la estructura general de la heurística de búsqueda adaptativa. En la sección IV., nos centramos en los métodos de búsqueda local, describiendo la búsqueda de descenso y la búsqueda tabú. La sección V. describe los experimentos computacionales para evaluar la heurística propuesta, presenta los resultados computacionales y ejecuta una comparación entre otros métodos. Finalmente concluimos con la sección VI. con observaciones generales de este trabajo.

II. EL PROBLEMA DE LA ASIGNACIÓN GENERALIZADA

El GAP consiste en asignar una serie de tareas a una serie de recursos con capacidad con un coste total mínimo. Cada tarea debe ser asignada a uno y uno solo recurso requiriendo una cierta cantidad de esta capacidad. Este problema es bien conocido y para un extenso análisis vea por ejemplo; Martello and Toth (1990), Cattrysse an Van Wassenhove (1992) y P. Chu and Beasley (1997). Beasley (1997). Fisher, Jaikumar y Van Wassenhove (1986) probaron que el problema es "NP-hard". Por otra parte, el problema de decidir si existe una solución factible es "NP-complete", Garey y Johnson (1979). Osman (1995) presento una perspectiva de varias aplicaciones en la vida real.

Se propone un modelo matemático basado en la programación entera, y posteriormente un método eficiente de resolución que permite la obtención de soluciones en tiempo real. Las principales respuestas del

* Departamento de Economia i Empresa - GREL, Universitat Pompeu Fabra, Barcelona, España.
helena.ramalhino@econ.upf.es ; daniel.serra@econ.upf.es

modelo si dirigen a la planificación y asignación de los recursos a tareas a realizar durante un cierto periodo, respetando la cantidad disponible de cada recurso y con el objetivo de maximizar la utilización del recurso de equipos o capital o la minimización de las ineficiencias y atrasos en la prestación de los servicios.

El Problema de Asignación Generalizada (GAP) puede ser formulado como un programa entero, como se presenta en la parte inferior.

La notación utilizada es la siguiente:

I : conjunto de tareas ($i=1,..., n$);

J : conjunto de agentes ($j=1, ..., m$);

a_j = capacidad del recurso j;

b_{ij} = necesidad de capacidad si la tarea i es asignada al recurso j ;

c_{ij} = coste de la tarea i si es asignada al recurso j;

Las variables son x_{ij} = 1, si la tarea i es asignada al agente j; = 0 si no.

Se asume que $b_{ij} \le a_j$ y que $\sum_{i=1}^{n} b_{ij} > a_j$.

$$(1)\ \min \quad f(x) = \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij} x_{ij}$$

s.a

$$(2)\quad \sum_{i=1}^{n} b_{ij} x_{ij} \le a_j, \quad j = 1,...,m$$

$$(3)\quad \sum_{j=1}^{m} x_{ij} = 1, \quad i = 1,...,n$$

$$(4)\quad x_{ij} \in \{0,1\}, \quad i = 1,...,n;\ j = 1,...,m$$

Las restricciones (2) están relacionadas con la capacidad de los recursos, las restricciones (3) garantizan que cada tarea sea asignada a un solo recurso, y puesto que las variables son binarias (restricción 4) cada tarea es asignada a uno y solo un recurso.

Varios algoritmos exactos para el GAP han sido propuestas por diversos autores: Ross y Soland (1975), Martello y Toth (1990), Fisher, Jaikumumar y Van Wassenhove (1986), Guinard y Rossenwein (1989), Karabakal et al. (1992) y Savelsbergh (1997). También, varias heurísticas han sido propuestas para resolver el GAP: Trick (1992), Cattrusse, Salomón, y Van Wassenhove (1994). Amini y Racer (1994) presentaron una heurística de profundidad variable de búsqueda motivada por el trabajo de Lin y Kernighan (1973) en el TSP (Problema del Agente Viajero). Osman (1995) presentó una comparación de algoritmos basados en búsqueda tabú y en recocido simulado. Wilson (1997) presentó un algoritmo genético para reestablecer la capacidad para una serie de soluciones cercanas a la óptima, y luego para perfeccionar la mejor solución hallada por la búsqueda local. Chu y Beasley (1997) también presentaron un algoritmo genético para el GAP que intenta perfeccionar la capacidad y simultáneamente su optimización. Laguna et al. (1995) propuso un algoritmo de búsqueda tabú basada para una generalización a diversos niveles del problema.

III. HEURÍSTICAS DE BÚSQUEDA ADAPTATIVA

Esta sección presenta la estructura general y los principales aspectos de la heurísticas de búsqueda adaptativas propuestas para resolver el GAP. La heurística de búsqueda adaptativa está basada en dos metaheurísticas para resolver los problemas de optimización combinatoria: Greedy Randomizrd Adaptive Search Heurístic (GRASP), Feo and Resende, MAX-MIN Ant System (MMAS), Stützle (1997,1998a,1998b) Stützle and Hoos (1999). Ambas técnicas incluyen un paso en el cual es aplicado un método de búsqueda local. Los métodos de búsqueda local son presentados en la siguiente sección, ya que presentan algunos aspectos innovadores.

Estructura General

La heurística de búsqueda adaptativa propuesta para resolver el GAP puede ser descrita en una estructura general incluyendo tres fases, que son aplicados repetidamente hasta que algún criterio de parada sea verificado:

Fase 1: Generar una solución usando una heurística aleatorizada tipo greedy.

Fase 2: Aplicar un método de búsqueda local.

Fase 3: Actualizar los parámetros (si hay).

El objetivo de la estructura general descrita es juntar metaheurísticas, recientemente propuestas como el GRASP o MMAS, en la misma estructura, lo cual consiste en primero generar una solución inicial, aplicarle un método de búsqueda local, luego, actualizar algunos parámetros basados en la solución obtenida. Este proceso es repetido hasta que algún criterio de parada sea verificado. Las heurísticas GRASP y MMAS serán discutidos detenidamente en las siguientes secciones.

La aproximación adoptada para el primer paso se basa en el método greedy, Adenso Días et. al.(1996). Los métodos greedy, de una forma general, construyen una solución añadiendo paulatinamente componentes individuales a la solución hasta que se obtiene una solución factible. En el caso del GAP, el método greedy funciona del siguiente modo: a cada paso, es elegida una nueva tarea para ser asignada; luego, la siguiente elección es el recurso para el cual es asignada la tarea elegida. Este procedimiento es repetido hasta que todas las tareas han sido asignadas a algún recurso.

Proponemos dos heurísticas para la primera fase: la heurística adaptativa greedy o *Greedy Ransomized Adaptive Hauristic* (GRAH) y otra heurística basada en la optimización con colonias de hormigas o *Ant System Heuristic* (ASH). La principal diferencia entre la GRAH y la ASH es el método de elegir el recurso para el cual la tarea elegida va a ser asignada. En el método greedy básico, la elección es determinista y está basada en una función de prioridades, por ejemplo la función de coste. Para el GRAH, la elección es una función aleatoria de prioridades que no depende de las soluciones obtenidas

previamente. Para el ASH, la elección es también es una función aleatoria de prioridades basada en valores ajustables que están adaptadas en un estilo de refuerzo de aprendizaje durante el curso del algoritmo y reflejan las ejecuciones anteriores. Para el segundo paso, son propuestas dos algoritmos de búsqueda local, búsqueda de descenso y búsqueda tabú. El tercer paso es solamente aplicado si hay parámetros que tienen que ser actualizados al final de cada iteración, el cual es el caso del MMAS.

Admitiremos soluciones no factibles con respeto a la capacidad de los recursos, pero soluciones no factibles serán penalizadas en la función objetiva. La principal razón para admitir soluciones no factibles es mejorar la eficiencia de la búsqueda y permitir rutas de escape fuera de la optimización local. Esta aproximación es bastante común en aplicaciones de metaheurísticas y a menudo es muy efectiva (vea Johnson et al. (1989)).

IV. MÉTODOS DE BÚSQUEDA LOCAL

Los métodos de búsqueda local son métodos extensivamente usadas para obtener buenas soluciones para problemas difíciles de optimización combinatoria. Para poder deducir el método de búsqueda local, es necesario definir un entorno (*neighborhood*), que es una función que asocia una conjunto de soluciones $N(x)$ con cada solución x. El entorno es usualmente obtenido por modificaciones específicas en x, llamadas movimientos.

La búsqueda local empieza con una solución inicial, x, y busca en el entorno por una solución con un menor coste, *Entorno(x,flag)* donde *flag* indica se si ha encontrado o no una solución vecina mejor. Luego, esta solución vecina sustituye la solución actual y la búsqueda continua hasta que un criterio de parada es verificado. El algoritmo devuelve la mejor solución encontrada con respeto a la función de coste, designada por óptimo local.

Entornos

Presentamos dos entornos para el GAP, un entorno simples basado en un movimiento donde una tarea es reasignada a un nuevo recurso y una entorno más complejo donde más tareas son reasignadas a nuevos recursos.

El movimiento en el entorno simples consiste en remover una tarea de un recurso y asignar esta tarea a otro recurso diferente. El tamaño del entorno es $n(m-1)$.

El entorno más complejo está basado en movimientos en cadena, el cual se mueve más de una tarea desde el actual recurso a un nuevo recurso. Los movimientos en cadena fueron introducidos por Glover (1992), y han sido aplicados en varios problemas, incluyendo una extensión del GAP, Laguna et al.(1995). Este entorno es más complejo que el anterior, pero lleva a una búsqueda más poderosa y eficiente sin un aumento significante del tiempo de computación.

El entorno de cadena de movimientos puede ser obtenido por la aplicación de los siguientes dos tipos de movimientos:

Movimiento A: Cambia la tarea i de un recurso j a otro recurso diferente w.

Movimiento B: Cambia la tarea i de un recurso j a otro recurso diferente w. Después, cambia una tarea k del recurso w a otro recurso, diferente de w, pero que puede ser el j.

El entorno de movimientos en cadena para una solución puede ser obtenido de una forma similar al entorno simples, pero el movimiento B solo es aplicado si el movimiento A no ha tenido éxito. El número de vecinos es del orden $O(n^2m^2)$, y, es significantemente mayor que para el entorno simples.

Búsqueda de Descenso

Un de los métodos propuestos para la búsqueda local es la búsqueda de descenso. En él, en cada iteración el movimiento se produce desde la solución actual a una de su entorno que sea mejor que ella, finalizando la búsqueda cuando todas las soluciones de su entorno sean peores. Es decir, la solución final será siempre un óptimo local. Los principales pasos de la búsqueda de descenso, *búsqueda_descenso(x)*, aplicada a una solución inicial x, son:

1. Sea *flag=false*;
2. Aplicar *Entorno(x, flag)*:
3. Si *flag=false*, detenerse (una optimo local, x, fue encontrada), si no repite el paso 2.

Estamos ahora en una posición para presentar el método GRASP, el cual consiste en una de los métodos propuestos para resolver el GAP:

1. Mientras que un criterio de parada no sea satisfecho:
   1.1. Construye una solución x usando el GRAH. En la primera iteración inicial $x_b=x$, la mejor solución encontrada por la búsqueda.
   1.2. Aplica la *búsqueda_descenso(x)*
   1.3. Si x es factible, y $f(x)<f(xb)$ sea $x_b=x$.
2. Devuelve la mejor solución encontrada, x_b.

El siguiente método para el GAP, también se basa en la estructura general, es el procedimiento MMAS, y puede ser descrito del modo siguiente:

1. Iniciar los caminos de feromeno y los parámetros.
2. Mientras que un criterio de parada no sea satisfecho:
   2.1. Construir una solución x usando el ASH. En la primera iteración inicial $x_b=x$, la mejor solución encontrada por la búsqueda.
   2.2. Aplicar *búsqueda_descenso(x, flag):*
   2.3. Actualiza parámetros relacionados con el feromone usando la actual solución x.
   2.4. Si x es factible, y $f(x)<f(x_b)$ sea $x_b=x$.
3. Devuelve la mejor solución encontrada, x_b.

En ambos métodos, el criterio de parada aplicado consiste en un número máximo de iteraciones.

Búsqueda tabú

La búsqueda tabú fue originalmente propuesta por Glover (1986), y desde entonces esta metaheurística ha

sido sujeta a extensivos estudios y aplicada en varios problemas de optimización con muy buenos resultados. La búsqueda tabú puede ser descrita como una búsqueda inteligente que usa la memoria para dirigir la pesquisa lejos de las soluciones óptimas locales, continuando la búsqueda y de esto modo para encontrar mejores resultados. Para una vista preliminar de la búsqueda tabú vea Glover y Laguna (1997).

Los ingredientes básicos de la búsqueda tabú son: el entorno, la lista tabú, el criterio de aspiración y el criterio de parada. La búsqueda tabú, *búsqueda_tabú(x)*, aplicada a una solución inicial *x*, puede ser brevemente descrita del siguiente modo:

1. Mientras el criterio de parada no es verificado:
   1.1. Generar la lista de candidatos de movimientos/vecinos;
   1.2. Escoger el mejor vecino no tabú o que verifica el criterio de aspiración, *x'*;
   1.3. Actualizar la solución actual, *x*=*x'*.
2. Output la mejor solución encontrada.

Un atributo tabú está relacionado con el movimiento de una tarea desde un recurso a otro recurso, ej. suponer una tarea *i* está asignada a un recurso *j* en la solución actual y esta tarea es reasignada al recurso *k*. Luego para un número concreto de iteraciones siguientes está prohibido (o es tabú) asignar *i* al recurso j. La lista tabú fue utilizada como una matriz, en que la entrada *(i,j)* contiene el número de la iteración donde la tarea *i* fue cambiada desde el recurso *j*.

El criterio de aspiración es considerado como el que deniega el status tabú de un movimiento, si este lleva a la mejor solución encontrada hasta el momento. Tal como se ha hecho para la búsqueda de descenso, ahora tenemos dos métodos más para el GAP, GRAH/TABU y el ASH/TABU que puede ser descrito como antes, pero en vez de usar el método de búsqueda de descenso se aplica la búsqueda tabú.

V. EXPERIMENTO COMPUTACIONAL

En esta sección, presentaremos los experimentos computacionales y los resultados obtenidos. Hemos seguido las pautas propuestas por Barr et al.(1995). El experimento computacional fue designado con tres objetivos principales:

- Entender el comportamiento de los diferentes métodos propuestos, basados en la estructura general de la heurística de búsqueda adaptativa.
- Comparar los dos métodos propuestos para el primer paso de la estructura general: GRAH y ASH.
- Comparar los métodos descritos en este trabajo con otras técnicas y metodología propuesta para resolver el GAP.

Todos los métodos descritos fueron codificados en Fortran, y fueron testados en un conjunto de problemas clasificados de 8 recursos/40 tareas en 10 recursos/60 tareas. Estos problemas de teste están disponibles en (http://www.ms.ic.ac.uk/info.html) y han sido también utilizadas por otros autores en sus experimentos computacionales, Osman (1995), Acttrysse, Salomon y Van Wassenhove (1994), Chu y Beasley (1997). Todos los tests numéricos han sido ejecutados en un PC-Pentium II con 166 MHz y 16 MB RAM.

Las medidas del comportamiento de los métodos consideradas son:

- La calidad de la solución obtenida como un tanto por ciento de la variación de la solución óptima.
- El tiempo de computación, i.e. el tiempo total y el tiempo para encontrar la mejor solución.

Los factores que pueden influir en el comportamiento de un método y sus resultados son:

- Problema específico: número de recursos (m); numero de tareas (n); capacidad de recursos (a_j), coste de la capacidad sobrecargada (α).
- Primer Paso: GRAH o ASH.
- Segundo paso: búsqueda de descenso o búsqueda tabú y el entorno utilizado.
- Criterio de parada: número total de iteraciones (NTI) y el número de iteraciones de la búsqueda tabú (NITB).
- Otros parámetros: tamaño del RCL, tamaño de la lista tabú (STL), otros parámetros: $t_{\min}$, $t_{\max}$, r y p_0.

Si queremos considerar todos los factores de arriba, la experimentación será bastante extensa. Por lo tanto para minimizar el esfuerzo computacional algunos de los factores de arriba son escogidos a priori basados en experimentos previos para el GAP o en resultados computacionales preliminares. Los siguientes valores de los parámetros para el MMAS fueron encontrados por dar buenas actuaciones en corridas preliminares:

$t_{\min} = 0.1 \times \min_{\forall i,j} t_{ij}$ y $t_{\max} = n \times \max_{\forall i,j} t_{ij}$,

$r = 0.75$ y $p_0 = \frac{n-m}{n} \times 0.8$.

Comparación entre diferentes aproximaciones

La principal utilización para estos testes iniciales es entender el comportamiento de los diferentes heurísticas de búsqueda adaptativa basadas en la estructura general. Las distintas combinaciones heurísticas consideradas están descritas en la Tabla I.

En los tres últimos métodos, antes de aplicar el método de búsqueda tabú, aplicamos un simple método de búsqueda de descenso. La razón es que muchas de las soluciones obtenidas en el primer paso no son factibles y una simple aplicación del método de búsqueda de descenso encuentra una solución factible en menor tiempo, de este modo, permite a la búsqueda tabú empezar desde una solución mejor. Con el último método intentaremos analizar el esfuerzo de usar o no la lista restringida de candidatos.

En este experimento, los siguientes factores están prefijados: NTI =30, NITB=200, α=50, STL=10. Estos valores fueron fijados en testes preliminares, y fueron fijados para controlar el tiempo corrido. Para cada problema hemos realizado 5 corridas de cada uno de los métodos para cada instancia distinta, o sea un total de 25 corridas por cada grupo de instancias gap(7,...,12).

TABLA I

LAS DISTINTAS COMBINACIONES DELAS HEURÍSTICAS

Heurística	Fase 1	Fase 2	Observaciones
GRASP	GRAH	búsqueda de descenso con el entorno movimientos en cadena	Método GRASP
MMAS	ASH	búsqueda de descenso con el entorno movimientos en cadena	Método Max-Min Ant System
GASH+TS	GRAH	búsqueda tabú con el entorno movimientos en cadena	Entorno restringido
ASH+TS	ASH	búsqueda tabú con el entorno movimientos en cadena	Entorno restringido
GRAH+LS+TS	GRAH	búsqueda de descenso con el entorno simples seguida de búsqueda tabú con el entorno movimientos en cadena	Entorno restringido
ASH+LS+TS	ASH	búsqueda de descenso con el entorno simples seguida de búsqueda tabú con el entorno movimientos en cadena	Entorno restringido
ASH+LS+CTS	ASH	búsqueda de descenso con el entorno simples seguida de búsqueda tabú con el entorno movimientos en cadena	Entorno completo

En la Tabla II presentamos la calidad media de las soluciones para cada conjunto de problemas de teste. Los mejores resultados fueron obtenidos por el ASH+TS, GRAH+TS, ASH+LS+TS y GRAH+LS+TS, cuando la búsqueda tabú fue utilizada en el segundo paso de la estructura general. También, la combinación del LS+TS mejora los resultados, ya que la búsqueda tabú empieza con una mejor solución y rápidamente mejora los resultados. El MMAS y el GRASP obtuvieron los peores resultados y usualmente se quedan clavados en malo óptimo local.

TABLA II

CALIDAD DE LA SOLUCIÓN MEDIA 6 GRUPOS DE PROBLEMAS.

prob.	m*n	GRASP	MMAS	GRAH+TS	ASH+TS	GRAH+LS+TS	ASH+LS+TS	ASH+LS+CTS
gap7	8*40	0.047%	0.017%	0.000%	0.000%	0.000%	0.000%	0.000%
gap8	8*48	0.251%	0.198%	0.092%	0.081%	0.106%	0.042%	0.124%
gap9	10*30	0.056%	0.078%	0.011%	0.000%	0.011%	0.000%	0.011%
gap10	10*40	0.143%	0.113%	0.033%	0.038%	0.038%	0.013%	0.034%
gap11	10*50	0.045%	0.021%	0.000%	0.000%	0.000%	0.000%	0.003%
gap12	10*60	0.061%	0.036%	0.003%	0.000%	0.003%	0.000%	0.008%
Promedio		**0.100%**	**0.077%**	**0.023%**	**0.020%**	**0.026%**	**0.009%**	**0.030%**

Cuando se considera la versión completa del entorno de movimientos en cadena en la búsqueda tabú, ASH+LS+CTS, la calidad de la solución no mejoró. Por lo tanto, el uso de listas restrictivas de candidatos juega un papel importante en la búsqueda, y ayuda a encontrar buenas soluciones en tiempos significantemente menores, como puede ser observado en los tiempos computacionales dados en la Tablas III. También puede ser visto como las heurísticas propuestas cumplen muy bien, encontrando la solución óptima en muchas instancias. Para aquellos testes en los que las heurísticas fallaron para encontrar la óptima, las soluciones obtenidas están muy cercanas a la solución optima.

En la Tabla III presentamos el tiempo promedio para encontrar la mejor solución para los seis problemas de teste. Para todas las heurísticas, el tiempo de CPU incrementa a razón de m/n, y también con respecto al número de tareas. Para el mismo número de iteraciones globales de la estructura general, lo métodos basados en colonias de hormigas (MMAS, ASH+TS, ASH+TS+LS), siempre requieren menos tiempo que los basados en GRAH (GRASP, GRAH+LS, GRAH+KS+TS). El tiempo computacional para encontrar la mejor solución es significantemente menor que el tiempo total corrido, y otra vez ASH encuentra la mejor solución más rápidamente. En cualquier caso al diferencia entre el ASH+TS, ASH+LS+TS y el GRAH+LS+TS no es significante. La explicación por este comportamiento es que la búsqueda tabú con el entorno de cadena de movimientos encuentra eficazmente buenas soluciones.

TABLA III

TIEMPOS COMPUTACIONALES PARA ENCONTRAR LA MEJOR SOLUCIÓN

prob.	m*n	GRASP	MMAS	GRAH+TS	ASH+TS	GRAH+LS+TS	ASH+LS+TS	ASH+LS+CTS
gap7	8*40	42.8	35.8	56.6	19.7	21.4	22.5	92.4
gap8	8*48	88.8	80.3	96.5	68.1	78.3	85.3	225.9
gap9	10*30	22.2	18.8	10.7	19.0	13.4	15.3	55.8
gap10	10*40	85.4	45.5	51.2	35.7	34.5	37.5	159.7
gap11	10*50	98.3	84.1	64.3	33.8	33.9	28.4	278.2
gap12	10*60	211.9	137.5	91.7	59.3	95.0	59.7	379.7
Promedio		**91.5**	**67.0**	**61.8**	**39.3**	**46.1**	**41.5**	**198.6**

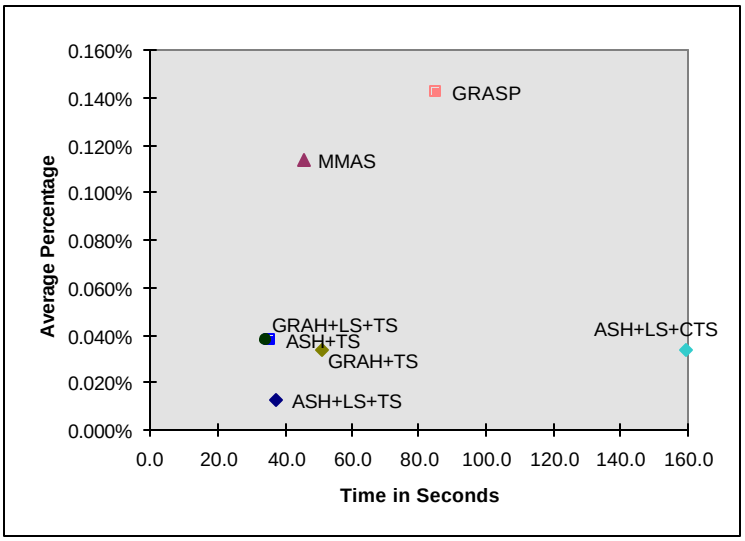

Fig. 1. Calidad de la solución versus tiempo computacional (gap10).

Para entender mejor el comportamiento de las varias heurísticas, presentamos dos figuras donde comparamos la calidad de la solución y el esfuerzo computacional. Por simplicidad, presentamos el promedio de los resultados para el problema de teste 10, Figura 1. Puede ser fácilmente observado que las heurísticas que obtienen mejores resultados en términos de calidad de solución y tiempo computacional son la AH+LS+TS, ASH+TS y la GRAH+LS+TS en este orden aproximadamente, ya que este domina las que quedan. Si tenemos que escoger solo una de entre las heurísticas aquí propuestas, nuestra elección será la ASH+LS+TS porque obtiene la mejor solución dentro de un tiempo computacional razonable.

Comparación entre las heurísticas de búsqueda adaptativas

Una segunda imposición que nos gustaría responder está relacionada con las diferentes aproximaciones propuestas para el primer paso. El GRAH está basado en la aleatoriedad para la construcción de soluciones iniciales. La otra aproximación basada en colonia de hormigas, utiliza información de las buenas soluciones visitadas en iteraciones previas para construir una solución. Queríamos estudiar si hay alguna diferencia entre estas dos aproximaciones para el GAP. Por esta razón, todos los factores fueron dejados constantes, excepto las dos heurísticas diferentes propuestas para el primer paso. Presentamos el promedio de los resultados para los seis problemas de teste cuando el GRAH y el ASH fueron usados en el primer paso de la estructura general, y combinados con la LS+TS, Figura 2 y la evolución de las soluciones, Figura 3. Uno puede ver que cuando el ASH es utilizado en el primer paso el método obtiene mejores soluciones en menos tiempo para gran parte de los problemas de teste y su evolución converge rápidamente para buenas soluciones.

La explicación de la diferencia entre el ASH y el GRAH es la calidad de la solución obtenida por estas heurísticas *greedy*. Hemos observado que las soluciones obtenidas por el GRAH son muy diferentes y no siguen ninguna pauta. A pesar de esto, para el ASH las soluciones obtenidas en las primeras iteraciones son peores o del mismo valor que las

obtenidas por el GRAH. Pero a medida que la búsqueda avanza, la ASH es capaz de obtener mejores soluciones, las cuales quieren decir menos tiempo corrido por el método de búsqueda local en el segundo paso.

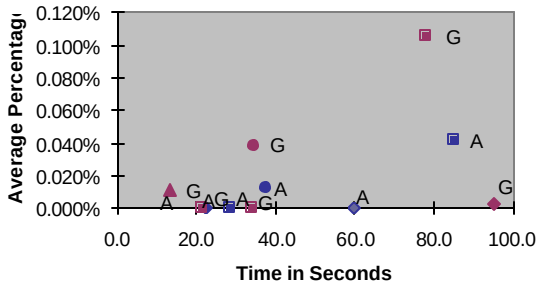

Fig. 2. Calidad de la solución versus tiempo computacional para el ASH+LS+TS (A) y GRAH+LS+TS (G).

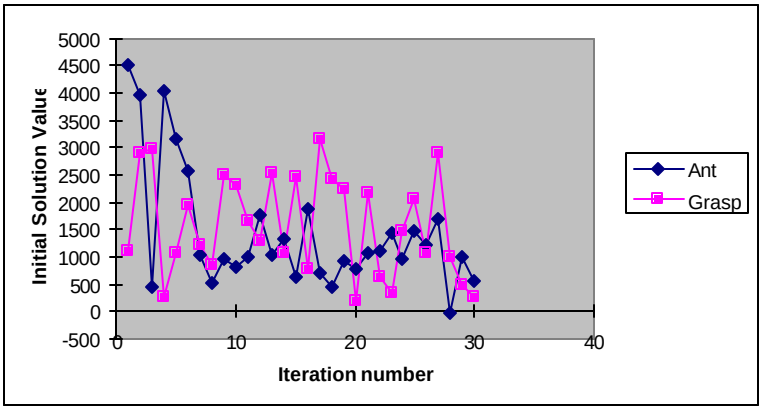

Fig. 3. Evolución de los algoritmos ASH+LS+TS (Ant) y GRAH+LS+TS (Grasp) APRA el gap9-2.

Resultados computacionales

Finalmente, en esta última sección, enseñamos la ejecución de nuestros mejores métodos para todos los problemas de teste y los comparamos con otros métodos propuestos para resolver el GAP, las metaheurísticas por Osman (1995), **TS6** y **TS1**, y Chu y Beasley (1997), **GAa** y **Gab,** Tabla IV.

Podemos observar que el ASH+LS+TS ha obtenido un promedio de mejores resultados que otras técnicas para estas instancias. Este método obtiene la solución óptima en todas las corridas para todos los problemas de teste numero 12, en los cuales otros métodos propuestos previamente no fueron capaces de encajar.

TABLA IV

COMPARACIÓN DE LOS RESULTADOS CON OTROS MÉTODOS

prob.	m*n	TS6	TS1	Ga_a	Ga_b	ASH+LS+TS	GRAH+LS+TS
gap7	8*40	0.02%	0.00%	0.08%	0.00%	0.00%	0.00%
gap8	8*48	0.14%	0.09%	0.33%	0.05%	0.04%	0.11%
gap9	10*30	0.06%	0.06%	0.17%	0.00%	0.00%	0.01%
gap10	10*40	0.15%	0.08%	0.27%	0.40%	0.01%	0.04%
gap11	10*50	0.02%	0.02%	0.20%	0.00%	0.00%	0.00%
gap12	10*60	0.07%	0.04%	0.17%	0.01%	0.00%	0.00%
Promedio		**0.08%**	**0.05%**	**0.20%**	**0.08%**	**0.01%**	**0.03%**

VI. CONCLUSIONES

La principal contribución de este trabajo es la aplicación de heurísticas de búsqueda adaptativa al problema generalizado de asignación, basadas en el GRASP y la metodología de optimización con colonias de hormigas. La estructura general tiene también algunos aspectos innovadores como la combinación del ASH y el GRAH con técnicas de búsqueda tabú, y el uso de entornos con movimientos en cadena.

La experiencia computacional mostró que los métodos híbridos basada en ideas de optimización con colonia de hormigas y el GRASP combinadas con la búsqueda tabú nos lleva a buenos resultados dentro de unos tiempos razonables. A partir de los resultados, podemos concluir que el éxito del ASH+LS+TS se debe a la combinación entre la búsqueda tabú y ASH. También, el entorno basado en movimientos en cadena y la estrategia de la lista de candidatos restringida juegan un papel importante conduciendo la búsqueda a buenas soluciones. Los resultados comparan favorablemente con los métodos existentes, en términos de tiempo empleado y calidad de la solución.

La eficiencia de los métodos propuestas es importante a la hora de aplicar estos modelos en problemas reales de asignación de tareas a recursos, en especial en el ámbito sanitario, productivos o logísticos, porque se trata de ambientes muy dinámicos y donde las decisiones se tienen de tomar en tiempo real.

VII. AGRADECIMIENTOS

Este trabajo ha sido posible gracias a la Fundación BBVA y al Centre de Recerca en Economia i Salud (CRES) de la Universitat Pomeu Fabra.

VIII. REFERENCIAS

[1] Adenso Díaz, Glover F, Ghaziri HM, González JL, Laguna M, Moscato P, and Tseng FT (1996). Optimización Heurística y redes Neurolanes: en dirección de Operaciones e Ingeniería, Editorial Praninfo, Madrid.

[2] Amini MM and Racer M (1994). A rigorous comparison of alternative solution methods for the generalized assignment problem, Mgmt Sci 40: 868-890.

[3] Barr RS, Golden BL, Kelly JP, Resende MGC and Stewart Jr WR (1995). Designing and Reporting on Computational Experiments with Heuristics Methods, Journal of Heuristics, 1: 9-32.

[4] Cattrysse DG and Van Wassenhove LN (1992). A survey of algorithms for the generalized assignment problem, Eur J of Opl Res 60: 260-272.

[5] Cattrysse DG, Salomon M and Van Wassenhove LN (1994). A set partitioning heuristic for the generalized assignment problem, Eur J of Opl Res 72: 167-174.

[6] Chu PC and Beasley JE (1997). A genetic algorithm for the generalised assignment problem, Comp Opns Res 24: 17-23.

[7] Colorni A, Dorigo M and Maniezzo V (1991a). Distributed Optimization by Ant Colonies, Proceeding of ECAL91 - European Conference on Artificial Life: Elsevier Publishing, Paris, France, 134-142.

[8] Colorni A, Dorigo M and Maniezzo V (1991b). The Ant System: Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man, and Cybernetics -Part B, 26, 1, 29-41.

[9] Dorigo M and Di Caro G (1999), The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover (eds), New Ideas in Optimization, McGraw-Hill.

[10] Dorigo M, Maniezzo V and Colorni A (1996), The ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics – Part B, 26(1): 29-42.

[11] Feo TA and Resende MGC (1995). Greedy randomized adaptive search heuristic, Journal of Global Optimization 6: 109-133.

[12] Fisher M, Jaikumar R and Van Wassenhove L (1986). A multiplier adjustment method for the generalized assignment problem, Mgmt Sci 32: 1095-1103.

[13] Garey MR and Johnson DS (1979). Computer and Intractability: a guide to the theory of NP-Completeness, Freeman, San Francisco.

[14] Glover F (1986). Future paths for integer programming and links to artificial intelligence, Comp Opns Res 5: 533-549.

[15] Glover F (1992). Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problem, University of Colorado. Shortened version published in Discrete Applied Mathematics, 65: 223:253 (1996).

[16] Glover F and Laguna M (1997). Tabu Search, Kluwer Academic Publishers: Norwell, Massachusetts.

[17] Glover, F. (1998). A Template for Scatter Search and Path Relinking, in Artificial Evolution, Lecture Notes in Computer Science 1363, J-K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.), Springer-Verlag, 13-54.

[18] Glover F (2000). Multi-start and strategic oscillation methods – Principles to exploit adaptive memory, in Computing Tools for Modeling, Optimization and Simulation, edited by Manuel Laguna and José Luis González Velarde, Kluwer Academic Publishers, 1-38.

[19] Goss S, Aron S, Deneubourg JL, and Pasteels JM (1989). Self-organized shortcuts in the Argentine Ant, Naturwissenschaften, 79: 579-581.

[20] Guignard M and Rosenwein M (1989). An improved dual-based algorithm to the knapsack problem, Eur J Opnl Res 27: 313-323.

[21] Johnson DS, Aragon CR, McGeoch LA, and Schevon C (1989). Optimization by Simulated Annealing: an experimental evaluation; part I, graph partitioning, Opns Res 39:3, 865.

[22] Karabakal N, Bean JC and Lohmann JR (1992). A steepest descent multiplier adjustment method for

the generalized assignment problem. Report 92-11, University of Michigan, Ann Arbor, MI.
[23] Laguna M, Kelly JP, González-Velarde JL and Glover F (1995). Tabu search for the multilevel generalized assignment problem, Eur J Oper Res 82: 176-189.
[24] Lin S and Kernighan BW (1973), An efficient heuristic algorithm for the traveling salesman problem, Operations Research 21: 498-516.
[25] Martello S and Toth P (1990). Knapsack Problems: Algorithms and Computer Implementations, Wiley: New York.
[26] Martin O, Otto SW and Felten EW (1992). Large-step Markov chain for the TSP incorporating local search heuristics, Operations Research Letters, 11: 219-224.
[27] Moscato P (1999). Memetic Algorithms: a short introduction, in New Ideas in Optimization, edited by D. Corne, F. Glover, and M. Dorigo, McGraw-Hill, 219-234.
[28] Osman IH (1995). Heuristics for the generalized assignment problem: simulated annealing and tabu search approaches, OR Spektrum 17: 211-225.
[29] Ross GT and Soland PM (1975). A branch and bound based algorithm for the generalized assignment problem, Mat h Prog 8: 91-103.
[30] Savelsbergh M (1997). A Branch-And-Cut Algorithm for the Generalized Assignment Problem, Opns Res 45: 6, 831- 841.
[31] Stützle T (1997). MAX-MIN Ant System for the Quadratic Assignment Problem, Technical Report AIDA-97-4, FG Intellektik,TU Darmstadt, Germany.
[32] Stützle T (1998a). Local Search Algorithms for Combinatorial Problems- Analysis, Improvements , and New Applications. PhD thesis, Departement of Computer Science, Darmstadt University of Technology, Germany.
[33] Stützle T (1998b). An ant approach for the flow shop problem, In Proceeding of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98), 3: 1560-1564, Verlag Mainz.
[34] Stützle T and Hoos H (1999). Max-Min Ant System and Local Search for Combinatorial Optimization. in: S. Vob, S. Martello, I.H. Osman and C. Roucairol (eds), Meta-Heuristics: Trends in Local Search paradigms for Optimization, Kluwer Academic Publishers, pp. 313-329.
[35] Trick MA (1992). A linear relaxation heuristic for the generalized assignment problem, Naval Res Logist 39: 137-152.
[36] Yagiura M, Yamaguchi T and Ibaraki T (1998). A variable depth search algorithm with branching search for the generalized assignment problem, Optimization Methods and Software, vol. 10, 419-441.
[37] Yagiura M, Yamaguchi T and Ibaraki T (1999). A variable depth search algorithm for the generalized assignment problem, in: S. Vob, S. Martello, I.H. Osman and C. Roucairol (eds), Meta-Heuristics: Trends in Local Search paradigms for Optimization, Kluwer Academic Publishers, 459-471.
[38] Wilson JM (1997). A genetic algorithm for the generalised assignment problem, J Opl Res Soc, 48: 804-809.