Matemática: Una publicación de FCNM – ESPOL
2014, Vol. 12, No.2

EL PROBLEMA DE LA MOCHILA, COMPLEJIDAD, COTAS Y MÉTODOS DE BÚSQUEDA EFICIENTES

Sandoya Fernando[1]

Resumen. *El problema de la mochila (KP por sus siglas en inglés), es un problema de optimización combinatoria muy referenciado en la literatura de investigación de operaciones, tanto por sus aplicaciones como por su estructura, que lo hace ideal para la evaluación del desempeño de métodos de búsqueda inteligente en problemas de optimización combinatorio. En este artículo se explora la utilización de optimizadores de propósito general, que usan métodos de solución estándar basados en algoritmos genéticos, solvers que utilizan métodos exactos, fundamentalmente métodos basados en branch&bound y se diseña dos algoritmos propios, el primero basado en la metaheurística GRASP clásica, y el segundo en una modificación a la propuesta original de la metodología GRASP, que demuestra ser muy eficiente en la solución del KP, y que es construido en base a un estudio de las cotas de este problema, con ello se propone el nuevo esquema GRASP para desarrollar algoritmos eficientes para resolver problemas de optimización combinatoria.*

Palabras Claves: Problema de la mochila, Optimización Combinatoria, Metaheurísticas, GRASP, Algoritmos Greedy.

Abstract. *The knapsack problem (KP) is a combinatorial optimization problem very referenced in the literature of Operations Research, both for its applications as its structure, making it ideal for performance evaluation of intelligent search methods useful for solving combinatorial optimization problems. In this article the use of general purpose solvers, using standard methods of solution based on genetic algorithms, and exact methods, mainly branch and bound based and two proprietary algorithms designed is explored, the first based on the GRASP classical metaheuristic, and the second in an amendment to the original proposal of the GRASP methodology, which proves to be very efficient in solving the KP, and is built on a study of the dimensions of this problem, thus the new scheme GRASP is proposed to develop efficient algorithms for solving combinatorial optimization problems.*

Keywords: Knapsack Problem, Combinatorial Optimization, Metaheuristics, GRASP, Greedy Algorithms.

Recibido: Julio2014.
Aceptado: Agosto 2014.

1. INTRODUCCIÓN

El problema de la mochila (KP, por sus siglas del inglés Knapsack Problem) es un problema muy referenciado en la literatura de investigación de operaciones, y que ha sido intensamente estudiado desde la segunda mitad del siglo XX, con el boom de la programación lineal y la programación lineal entera, este vasto interés ha sido producto por un lado de sus aplicaciones directas e indirectas, pero también ha sido originado por su uso como un campo de pruebas para ensayar la eficiencia de métodos de solución y de búsqueda inteligente en problemas de optimización combinatoria.

En este artículo se explora la utilización de optimizadores de propósito general, que usan métodos de solución estándar basados en algoritmos genéticos, solvers que utilizan métodos exactos, fundamentalmente métodos basados en branch&bound y algoritmos propios basados en la metaheurística GRASP.

2. EL PROBLEMA KP

El problema KP es un problema de optimización combinatoria de formulación sencilla, aunque su resolución es compleja, y que aparece, directamente o como un subproblema en una gran variedad de aplicaciones, incluyendo planificación de la producción, modelización financiera, muestreo estratificado, planificación de la capacidad de instalaciones, etc., como se observa en [1]. Además de sus potenciales aplicaciones, el KP es de particular interés por sus características combinatorias y su estructura sencilla, que lo vuelven un problema ideal para el diseño de métodos de búsqueda inteligente. Con el tratamiento del problema KP se pueden evaluar las ventajas y desventajas de estos algoritmos, sobretodo su robustez, precisión y rapidez.

El KP es un problema combinatorio que es NP-duro, como se puede ver en [2], con respecto a la codificación binaria estándar y consta en la lista de 21 problemas NP-completos de Karp, por tanto es improbable que, en algún momento, pueda ser encontrado un algoritmo que pueda resolverlo en tiempo polinomial. Sin embargo KP no es del tipo fuertemente NP-duro, y por tanto puede ser resuelto en tiempo seudo-polinomial, particularmente por programación dinámica, como se demuestra en [3]; es decir, que en su codificación uniaria el problema es resoluble polinomialmente. De esta manera, incluso grandes instancias del problema pueden ser resueltas de manera exacta con solvers que utilizan métodos exactos como la búsqueda dinámica o branch&bound (en particular Cplex), y el planteamiento de métodos aproximados, como los algoritmos basados en metaheurísticas, se da para probar la eficiencia de estos procedimientos.

[1] *Sandoya Sánchez Fernando, Ph.D., Profesor, Departamento de Matemáticas, Facultad de Ciencias Naturales y Matemáticas, Escuela Superior Politécnica del Litoral (ESPOL)*

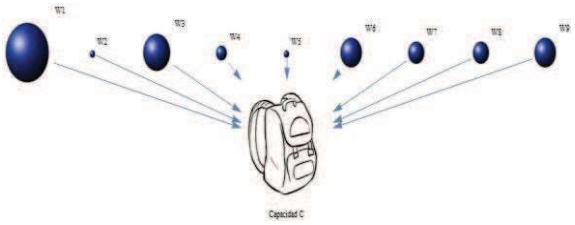
2.1 FORMULACIÓN

Dado un conjunto de objetos $V = \{1, 2, 3, \dots, n\}$, con utilidades unitarias $p_1, p_2, \dots, p_n$, y con pesos $w_1, w_2, \dots, w_n$, respectivamente, y dado que se tiene un recipiente (la "mochila"), de capacidad c, el problema KP consiste en determinar qué objetos debo seleccionar para incluir en la mochila, de tal manera que la utilidad total de los objetos que se cargan sea la máxima posible, como se puede ver en la FIGURA 1.

FIGURA 1

El problema de la mochila, complejidad, cotas y métodos de búsqueda eficientes

Como cargar óptimamente la mochila sin rebasar su capacidad



Así, el KP puede ser representado sencillamente por la siguiente formulación de programación binaria:

$$Max \sum_{i=1}^{n} p_i x_i$$

$$s.t. \sum_{i=1}^{n} w_i x_i \leq c$$

$$x_i \in \{0, 1\}; i = 1, 2, \dots, n$$

En donde: $x_i = 1$ si decidimos incluir el objeto i en la mochila, y $x_i = 0$ sino.

2.2 COTAS

Para el problema KP se pueden establecer fácilmente cotas, inferiores y superiores para la carga óptima.

Primero, establecemos el parámetro b, denominado el "artículo de quiebre", definido de la siguiente manera:

- Ordenamos los artículos en sentido decreciente a su importancia marginal: $e_j = {}^{p_j}/_{w_j}$.
- Al cargar la mochila con un algoritmo voraz de acuerdo a este orden, b es el primer artículo que no puede ser incluido en la mochila.

Propiedad:

Si representamos con $\overline{p_j} = \sum_{i=1}^{j} p_i$, $\overline{w_j} = \sum_{i=1}^{j} w_i$ y $r = c - \overline{w_{b-1}}$, entonces:

1. $\overline{p_{b-1}}$ es una cota inferior para el óptimo de KP
2. $\overline{p_{b-1}} + r\,\frac{\overline{p_b}}{w_b}$ es una cota superior para el óptimo de KP

Demostración:

Si representamos con $\overline{p_j} = \sum_{i=1}^{j} p_i$, $\overline{w_j} = \sum_{i=1}^{j} w_i$, $j = 0, \dots, n$, el algoritmo glotón incluirá los artículos 1, 2,… mientras se cumpla que:

$$w_j \leq c - \overline{w_{j-1}}$$

Es claro que: $w_{b-1} \leq c - \overline{w_b}$.

La solución de quiebre $x' = \{x'_1, x'_2, \dots, x'_n\}$ es la solución en la que $x'_j = 1$ para $j = 1, \dots, b-1$, y $x'_j = 0$ para $j = b, \dots, n$.

Por otro lado, se define la capacidad residual $r = c - \overline{w_{b-1}}$; es decir, el volumen de la mochila que queda desocupado cuando se aplica el algoritmo glotón sobre la lista ordenada en orden decreciente de importancia marginal de los artículos.

Así, una cota superior, u, para el KP es:

$$u = \left\lfloor \overline{p_{b-1}} + r\,\frac{\overline{p_b}}{w_b} \right\rfloor$$

Por tanto, la solución del KP está acotada en el intervalo: $[\overline{p_{b-1}}, u]$.

3. DISEÑO DE UN ALGORITMO EFICIENTE PARA RESOLVER EL KP

La idea en este estudio es comparar la eficiencia de algunos procedimientos de búsqueda inteligente para resolver el problema KP. Para esta comparación se utilizarán las instancias de prueba de Pisinger, que son problemas de tipo KP con un diseño que los vuelve particularmente difíciles de resolver, de esta manera si un algoritmo tiene buen desempeño al resolver estas instancias, resolverá con gran probabilidad un problema real. Estas instancias se pueden encontrar en el sitio web de Optsicom, [4], y corresponden a conjuntos de datos fuertemente correlacionados, no correlacionados y débilmente correlacionados.

En estas instancias de Pisinger se consideran 4 tipos de ejemplos para comprobar la eficiencia de algoritmos sobre KP: en cada tipo se considera diferentes rangos de datos: R = 100, 1000, 10000 para diferentes tamaños de problemas: n = 1,000 y 3,000, donde la capacidad c es elegida de tal manera que $c = \frac{1}{2}\overline{w_n}$, estos tipos son:

- Instancias de datos no correlacionados (UC), donde p_j , w_j están distribuidas uniformemente en $[1, R]$.
- Instancias de datos débilmente correlacionados (WC), donde w_j están distribuidas uniformemente en $[1, R]$ y p_j distribuidas uniformemente en $\left[w_j - \frac{1}{10}R, w_j + \frac{1}{10}R\right]$, y tal que $p_j \geq 1$;

- Instancias de datos fuertemente correlacionados (SC), donde w_j están distribuidas uniformemente en [1, R], y $p_j = w_j + 10$;
- Instancias de datos subset-sum (SS), donde w_j están distribuidas aleatoriamente en [1, R], y $p_j = w_j$.

Para resolver el problema utilizaremos:

1. Un optimizador de propósito general, específicamente EVOLVER, que se presenta como el optimizador comercial que utiliza métodos genéticos más eficiente del mercado.
2. Un optimizador MIP exacto, específicamente Cplex, que utilizando programación dinámica permite hallar la solución óptima en todos los casos.
3. Un algoritmo basado en la metodología clásica propuesta por GRASP, denominado GRASP_1, diseñado a partir de las características del KP, principalmente de las cotas establecidas en la propiedad.
4. Un algoritmo basado en una modificación a la metaheurística GRASP clásica, denominado GRASP_KP, en el cual se altera el orden de la ejecución aleatorizada y voraz del algoritmo.

3.1 ALGORITMO DETERMINÍSTICO VORAZ

Un algoritmo voraz es un algoritmo determinístico, es decir si se lo ejecuta repetidas veces el resultado siempre será el mismo. Actúa de la siguiente manera: en cada iteración el algoritmo evalúa cada candidato a formar parte de la solución según su aporte a la función objetivo, de tal forma que si el elemento es seleccionado pasa a formar parte de la solución, y si es descartado, no se lo incluye en la solución y no volverá a ser considerado para una próxima iteración, por lo que deja de ser un elemento candidato.

Los algoritmos voraces tienden a ser bastante rápidos en su ejecución, llegando la mayoría de las veces a alcanzar una complejidad de orden lineal, y su diseño es relativamente sencillo. Sin embargo, pueden tener poca precisión y quedar lejos de la solución óptima.

Para el problema del KP un algoritmo voraz sería el siguiente:

1. Inicio: $M_0 = \emptyset$ = solución parcial con los elementos seleccionados,
   $CL = \{1, 2, \ldots, n\}$ = conjunto inicial de elementos no seleccionados.
2. Dada una solución parcial M_k con k elementos seleccionados, seleccionar el elemento con mayor aporte marginal:
   $i^* \in CL: \max_{i \in CL} {}^{p_i}/_{w_i}$
3. Si hay suficiente capacidad remanente en la solución parcial actual para incluirlo; esto es, si
   $\sum_{j \in M_k} w_j x_j + w_{i*} \leq c$, $M_{k+1} = M_k \cup \{i^*\}$;.
   Y se actualiza $CL = CL \setminus \{i^*\}$.
   En caso contrario, si $\sum_{j \in M_k} w_j x_j + w_{i*} > c$, se elimina i^* del conjunto de elementos no seleccionados, $CL = CL \setminus \{i^*\}$ y se regresa a 2.

En la sección 5 se reportan los resultados obtenidos al aplicar este método voraz y su comparación con las heurísticas propuestas.

3.2 BASES DE LA METAHEURÍSTICA GRASP

El método GRASP (acrónimo del inglés "Greedy Randomized Adaptative Search Procedure") es una metaheurística introducida a finales de la década del 80 del siglo XX por Feo y Resende, en [5]. De manera general se puede describir la operación de GRASP de la siguiente manera:

En una primera fase se aplica un procedimiento para construir una "buena" solución factible, la cual después en una segunda fase es mejorada por una técnica de búsqueda local. Estas dos fases son repetidas de manera iterativa durante cierto número máximo de iteraciones, recordando la mejor solución explorada. En su forma clásica la fase constructiva de GRASP se realiza mediante un procedimiento glotón adaptativo; es decir, un método de construcción entrenado para considerar la aleatoriedad y que se adapta en cada paso de la construcción.

Las características de un algoritmo heurístico que ha sido diseñado en base a la metaheurística GRASP son:

- Su carácter adaptativo: porque en cada iteración del algoritmo se procede a actualizar el beneficio que se obtiene por añadir el nuevo elemento seleccionado a la solución parcial actual. De esta manera, el beneficio de agregar un elemento dado a la solución parcial en la iteración i, no será igual, necesariamente, con el beneficio que se tenga al agregar el mismo elemento en la iteración $i + 1$.
- Su carácter aleatorizado: El algoritmo así diseñado no utiliza la función voraz para seleccionar el mejor candidato, por el contrario, buscando diversificar la búsqueda y corregir la miopía de la selección voraz, y así no repetir las mismas construcciones en corridas diferentes del algoritmo, se construye una lista de elementos, denominada lista de candidatos restringida (LCR), con los mejores candidatos a formar parte de la solución, y de entre los que finalmente se seleccionará uno al azar.

Al igual que ocurre en la mayoría de métodos diseñados en base a metaheurísticas, las soluciones que se generan en la fase de construcción de un algoritmo basado en GRASP no suelen ser óptimos locales, por lo que, como fase siguiente, se suele aplicar algún

procedimiento de búsqueda local en la vecindad de la solución generada para mejorarla.

Durante la fase de construcción de un algoritmo basado en GRASP, se desarrolla la lista de candidatos restringida, LCR, de la siguiente manera:

1. Se determinan todos los elementos factibles de ingresar a la solución actual
2. Se determina una función que describa el aporte que tendría en la solución la inclusión de cada uno de los elementos factibles, esto da la dimensión glotona al método (Greedy).
3. Se escoge los $\alpha\%$ mejores elementos factibles, esta es la lista LCR
4. Aleatoriamente se escoge un elemento de LCR para añadirlo a la solución actual, esto da la dimensión probabilística (Randomized) del método.

La eficiencia del método depende en gran medida del valor de α que se use, por lo que este parámetro debe ser bien calibrado en los experimentos numéricos. Por definición α debe tomar valores en el intervalo $[0,1]$. Con $\alpha = 1$ el método se vuelve de búsqueda totalmente aleatoria, pues cualquier elemento tendría probabilidad de incorporarse a la solución en cada paso del algoritmo, mientras que con $\alpha = 0$ el método se aproxima a un procedimiento totalmente glotón, pues solo el mejor elemento sería incorporado en cada paso a la solución. Con esta filosofía diseñamos un algoritmo denominado GRASP_1 para resolver el problema KP.

Una modificación importante que se reporta últimamente en la literatura a la metaheurística GRASP, es la siguiente: las construcciones originales basadas en GRASP primero evalúan cada elemento candidato a seleccionarse por la función voraz (Greedy) que mide su aporte a la solución que está construyéndose, y de ahí se construye RCL, de la cual se selecciona finalmente un elemento aleatoriamente (Randomized) para ser incorporado a la solución. En [6] se demuestra que un diseño alternativo, en el cual primero aplicamos la aleatorización (Randomized) y luego la acción voraz (Greedy) puede ser más eficiente. La heurística que se propone en esta investigación, denominada GRASP_KP está construida con base a esta nueva filosofía.

Con esta última modificación en este artículo se logra demostrar, estadísticamente, que la distribución de las soluciones generadas por GRASP_KP tiene un patrón que, incluso en la peor de las construcciones, es en promedio mejor al obtenido por un algoritmo voraz, y por tanto determinista; de esta manera, la mejor de las soluciones generadas supera a la del procedimiento determinista con una alta probabilidad. Este hecho es significativo pues en [7] se demuestra que bajo el esquema clásico de GRASP los valores muestrales obtenidos son en promedio inferiores a los obtenidos por un procedimiento determinístico.

3.3 ALGORITMO GRASP_1

Con el comportamiento observado de las cotas para el óptimo del problema KP, se diseñó un algoritmo constructivo basado en la propuesta clásica de la metaheurística GRASP, que denominamos GRASP_1, en el cual intentamos añadir un nuevo elemento a la solución parcial bajo construcción siempre que quede espacio remanente en la mochila, es decir si:

$$r = c - \overline{w_{b-1}} > 0$$

El esquema del método de construcción es el siguiente:

1. Inicio: $M_0 = \emptyset$ = solución parcial con los elementos seleccionados,
   no_selec = $\{1, 2, \dots, n\}$ = conjunto de elementos no seleccionados.
2. Dada una solución parcial M_k con k elementos seleccionados, la lista de candidatos CL = no_selec.
3. Evaluar el aporte marginal
   $\forall i \in CL: e_i = {}^{p_i}/_{w_i}$
4. Se construye la lista de candidatos restringida, RCL, seleccionando la fracción α $(0 < \alpha < 1)$ de los mejores elementos de CL, en función de su aporte marginal, donde α es un parámetro que debe elegirse adecuadamente.
5. Se selecciona aleatoriamente un elemento i^* de RCL si hay suficiente capacidad remanente en la solución parcial actual para incluirlo; esto es, si $\sum_{j \in M_k} w_j x_j + w_{i*} \leq c$, y lo añade a la solución parcial, $M_{k+1} = M_k \cup \{i^*\}$;
6. En caso contrario, si $\sum_{j \in M_k} w_j x_j + w_{i*} > c$, no se actualiza M_k y se elimina i^* del conjunto de elementos no seleccionados, no_selec=no_selec \ $\{i^*\}$
7. Regresar a 2.

3.4 ALGORITMO GRASP_KP

Con base a la nueva propuesta de la metaheurística GRASP, en el que alteramos la secuencia de la aplicación de la parte voraz y aleatoria del algoritmo, desarrollamos un nuevo algoritmo que se muestra más eficiente. El esquema del método de búsqueda es el siguiente:

1. Inicio: $M_0 = \emptyset$ = solución parcial con los elementos seleccionados.
   no_selec= $\{1, 2, \dots, n\}$ = conjunto de elementos no seleccionados.
2. Dada una solución parcial M_k con k elementos seleccionados, la lista de candidatos CL=no_selec.
3. La lista de candidatos restringida,RCL, se construye seleccionando aleatoriamente una

fracción α $(0 < \alpha < 1)$ de los elementos de CL, donde α es un parámetro que debe elegirse adecuadamente.

4. Cada elemento $i \in RCL$ es evaluado de acuerdo a su aporte marginal:
$$e_i = {}^{p_i}/_{w_i}$$
5. Se selecciona el mejor candidato i^* en RCL si hay suficiente capacidad remanente en la solución parcial actual para incluirlo; esto es, si $\sum_{j \in M_k} w_j x_j + w_{i*} \leq c$, y lo añade a la solución parcial, $M_{k+1} = M_k \cup \{i^*\}$;
   En caso contrario, si $\sum_{j \in M_k} w_j x_j + w_{i*} > c$, no se actualiza M_k y se elimina i^* del conjunto de elementos no seleccionados, no_selec = no_selec \ $\{i^*\}$
6. Regresar a 2.

4. SOLUCIÓN DEL KP EN UN OPTIMIZADOR DE PROPÓSITO GENERAL

Cuando se resuelve un problema de optimización, una práctica común es desarrollar un método dependiente del contexto (es decir un procedimiento que explote explícitamente la estructura y propiedades del problema para conducir una búsqueda eficiente), tal como los algoritmos GRASP_1 y GRASP_KP desarrollados aquí, muchas implementaciones de heurísticas y metaheurísticas están basadas en este paradigma. Por otro lado, como proponen Gortázar y otros en [8], algunos investigadores proponen crear procedimientos generales que permitan hallar soluciones de calidad razonable para una amplia clase de problemas de optimización basados en un paradigma independiente del contexto, y es bajo esta filosofía que se han desarrollado programas de software comercial relevantes que se han popularizado en los últimos años en esta área. La mayoría de estos programas no explotan la estructura específica de la función objetivo del problema que se está resolviendo, sino que funcionan a manera de una caja negra en cuanto a que lo importante en ellos es poner énfasis en tener bien definidas las entradas que recibe el optimizador y saber interpretar las salidas o respuestas que produce, sin tener en cuenta en mayor medida su funcionamiento interno. Dicho de otro modo, en estos optimizadores de caja negra nos interesará describir cómo está modelado el problema y entenderemos qué es lo que hace el optimizador, pero sin dar mucha importancia a cómo lo hace. No se precisa definir ni conocer los detalles internos de su funcionamiento. Salvo ciertos aspectos que están disponibles para calibración por parte del usuario, el optimizador de caja negra se encarga de obtener una solución.

Uno de los optimizadores de caja negra más populares actualmente es Evolver, de Palisade Corp., que como se muestra en [9], utiliza la tecnología de los algoritmos genéticos para resolver rápidamente problemas de optimización.
En este artículo comparamos nuestras heurísticas basadas en GRASP con el optimizador Evolver, en las corridas se otorgó más tiempo de ejecución al optimizador Evolver como una ventaja frente a nuestras heurísticas GRASP. A pesar de esta ventaja en el tiempo de cómputo otorgada al optimizador Evolver, los resultados que se muestran en la sección 5 indican que los algoritmos basados en GRASP siempre fueron mejores, y que existe un gran deterioro de la solución hallada por Evolver conforme se utilizan problemas de talla más grande.

5. RESULTADOS NUMÉRICOS

En esta sección se presentan los resultados obtenidos a partir de la aplicación de los métodos planteados; la comparación de los diferentes algoritmos se lo realizó en función de su calidad y eficiencia. Para determinar la eficiencia de los algoritmos se calculó el tiempo promedio de resolución (en segundos), y para la calidad se reporta el porcentaje de desviación Promedio al óptimo (diferencia entre la solución heurística y la solución óptima).
Para cada prueba se han realizado 50 ejecuciones. Los experimentos computacionales se realizaron con dos tipos de problemas: problemas de tamaño medio n=1,000 y problemas de tamaño grande n= 3,000.
Los resultados de la calidad y el esfuerzo computacional de cada algoritmo se muestran en la TABLA I el algoritmo que menos esfuerzo computacional (tiempo en segundos) requiere es el Glotón, el que más esfuerzo computacional requirió fue Evolver. El optimizador de propósito general, incluso para tiempos de ejecución muy superiores a los de los otros algoritmos (600 segundos) no permitió mucha precisión en los resultados, obteniéndose un gap promedio de 3.5%.
En las heurísticas GRASP_KP se tiene que el valor de α para el cual el algoritmo tuvo mejor rendimiento es $\alpha = 0.3$.
En la TABLA I se muestra la desviación promedio porcentual hasta la mejor solución encontrada, la fila "# Factibles" indica en cuántos de los problemas de prueba se obtuvieron soluciones factibles (que respetan la restricción de capacidad), y la fila "# óptimo" indica cuántas veces se alcanzó el óptimo en el tiempo límite. De la tabla se deduce que para todos los casos estudiados, el algoritmo que más se acerca al óptimo es el GRASP_KP, observando que da muy buenos resultados y con un esfuerzo

computacional aceptable, habiendo alcanzado incluso el óptimo exacto el 50% de las veces.

TABLA I

El problema de la mochila, complejidad, cotas y métodos de búsqueda eficientes

Comparación de los métodos GRASP_1, GRASP_KP, Evolver_KP y Glotón, en problemas de talla n=1,000

Comparación de calidad y esfuerzo computacional de los métodos, instancias medianas de n =1000					
	GRASP_KP	**GRASP_1**	**Glotón**	**Evolver (300 seg)**	**Evolver (600 seg)**
Desviación promedio porcentual	0.021%	0.102%	5.100%	4.740%	3.470%
# Factibles	24	24	24	24	24
# Óptimo	12	8	4	5	6
Tiempo promedio (seg.)	39.37	198.94	0.792	300	600

Analizando los resultados para problemas de talla más grande, $n = 3000$, se obtienen conclusiones similares. Así, en la TABLA II, podemos observar que el método que generó soluciones con el menor porcentaje de desviación promedio fue GRASP_KP. El tiempo promedio de resolución más alto fue encontrado usando el GRASP_1, equivalente al tiempo usado en Evolver. Así, GRASP_KP fue el más eficiente, habiendo incluso alcanzado la solución óptima exacta en casi la mitad de las veces.

Los resultados demuestran que el GRASP_KP brindan al problema planteado, soluciones muy buenas cercanas al óptimo, con un esfuerzo computacional aceptable. El valor de α que se recomienda es de 0,3 debido a que a partir de ese valor no varía mucho su desviación con la solución óptima.

TABLA II

El problema de la mochila, complejidad, cotas y métodos de búsqueda eficientes

Comparación de los métodos GRASP_1, GRASP_KP, Evolver_KP y Glotón, muestras medianas de talla n=3,000

Comparación de calidad y esfuerzo computacional de los métodos, instancias medianas de n =3000				
	GRASP_KP	**GRASP_1**	**Glotón**	**Evolver_KP**
Desviación promedio porcentual	0.012%	0.099%	5.05%	8.212%
# Factibles	24	24	24	24
# Óptimo	11	6	5	4
Tiempo (seg.)	511.20	693.15	6.726	600

Los perfiles de búsqueda de estas dos heurísticas basadas en GRASP muestran también la superioridad de GRASP_KP frente al basado en el diseño tradicional GRASP1, en la Figura 2 y la Figura 3 se muestra este perfil de búsqueda para un problema de tamaño n=1,000 y n=3,000 respectivamente. Este patrón de comportamiento se repitió en todos los problemas testeados, se puede observar que GRASP_KP construye sistemáticamente buenas soluciones, cercanas al óptimo, mientras que con GRASP_1 la búsqueda da construcciones de baja calidad y se requiere tiempo para que vayan mejorando.

FIGURA 2

El problema de la mochila, complejidad, cotas y métodos de búsqueda eficientes

Perfil de búsqueda de GRASP_1 y GRASP_KP en un problema de talla grande n=1000, en función del tiempo de ejecución y del número de iteraciones.

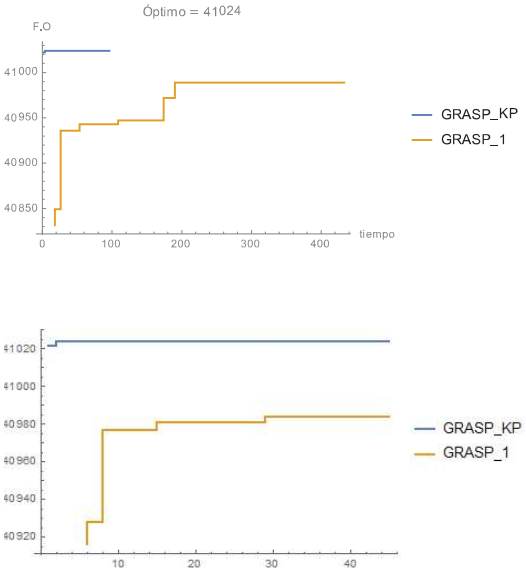
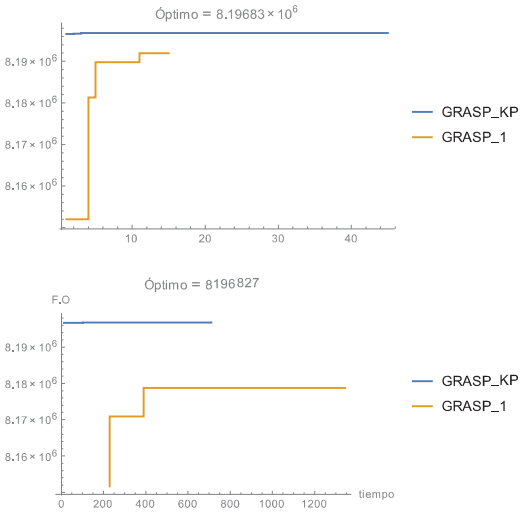


FIGURA 3

El problema de la mochila, complejidad, cotas y métodos de búsqueda eficientes

Perfil de búsqueda de GRASP_1 y GRASP_KP en un problema de talla grande n=3000, en función del tiempo de ejecución y del número de iteraciones

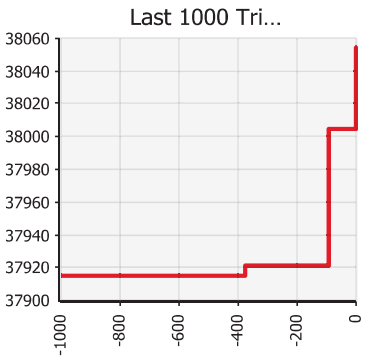


En cuanto al perfil de búsqueda del optimizador de propósito general Evolver, cuyos ejemplos se muestra en la FIGURA 4 y en la FIGURA 5, que reflejan la mejora en la búsqueda para los problemas de talla 1000 y 3000 de las figuras anteriores, indican claramente que hay un excesivo requerimiento de tiempo computacional, a pesar de lo cual los resultados encontrados son muy pobres en términos de la precisión, nótese que la peor de las construcciones con cualquiera de los métodos basados en GRASP es superior a la mejor de las soluciones alcanzadas por el optimizador Evolver, a pesar de que sistemáticamente se le otorgó más tiempo a la ejecución de este último.

FIGURA 4

El problema de la mochila, complejidad, cotas y métodos de búsqueda eficientes

Perfil de búsqueda del optimizador de propósito general Evolver para el problema de prueba knapsack_uncorrelated_n_1000_r_100_3



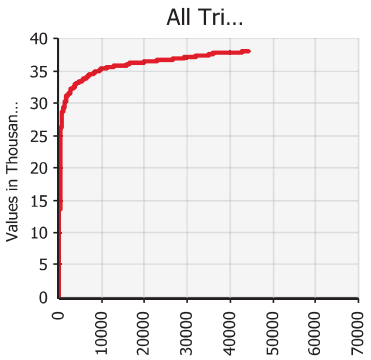
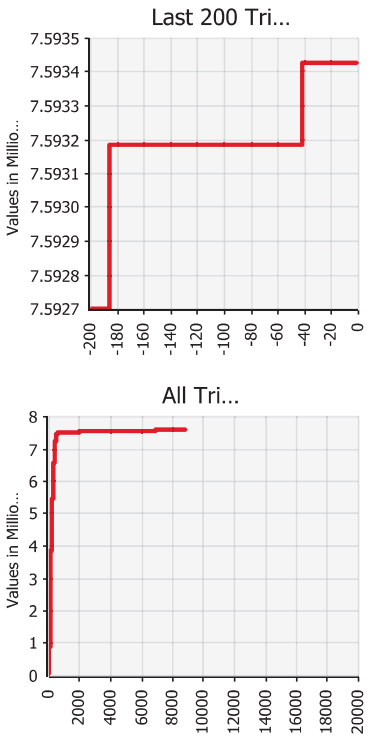


FIGURA 5

El problema de la mochila, complejidad, cotas y métodos de búsqueda eficientes

Perfil de búsqueda del optimizador de propósito general



Para completar este estudio se aplicó el test de Student para muestras aparejadas, ya que se trata de la aplicación de procedimientos distintos sobre los mismos problemas de prueba a las soluciones obtenidas. El nivel de significancia p resultante tiene un valor cercano al cero, lo cual indica claramente que hay una significativa diferencia estadística entre los métodos GRASP desarrollados. Un análisis adicional consiste en ordenar los métodos que se comparan de acuerdo al promedio de los valores calculados con este test. De acuerdo a esto, el mejor método es GRASP_KP (que superó en promedio a GRASP_1, como se observa en la columna 4 de la TABLA I y la TABLA II, seguido de GRASP_1.

Un resultado interesante que se puede observar aquí es que, tanto con GRASP_1 como GRASP_KP, el peor resultado que se obtuvo en alguna iteración supera al valor encontrado por un

algoritmo glotón (columnas 2 y 3 de la TABLA III. Esto quiere decir que hay una gran probabilidad que con una sola iteración de estos algoritmos (que se correrían muy rápido) se supere a la solución encontrada por un método determinístico.

TABLA III

El problema de la mochila, complejidad, cotas y métodos de búsqueda eficientes

Prueba estadística t para comparar la eficiencia de los métodos GRASP_1, GRASP_KP y glotón, muestras medianas de talla n =1,000

Prueba t para medias de dos muestras emparejadas: instancias medianas $n = 1000$			
	Peor construcción GRASP_KP vs. Glotón	Peor construcción GRASP_1 vs. Glotón	Mejor construcción GRASP_KP vs. Mejor construcción GRASP_1
Observaciones	24	24	24
Estadístico t	2.4151	1.9747	1.7199
$P(T <= t)$ una cola	0.0120	0.0302	0.0494
Valor crítico de t (una cola) al 95%	1.7139	1.7139	1.7139

6. CONCLUSIONES

El algoritmo desarrollado en base al nuevo esquema de la metaheurística GRASP muestra un mejor desempeño que el algoritmo desarrollado en base a la propuesta clásica de GRASP, en ese sentido conjeturamos que puede ser una mejor manera de aplicar esta metaheurística para la solución de otros problemas de optimización combinatoria.

El algoritmo GRASP_KP se muestra como el de mejor rendimiento, considerando velocidad, eficiencia y calidad de las soluciones obtenidas, esto es algo que destacar, porque además va de la mano con el hecho, demostrado empíricamente, de la sencillez de su formulación.

Los optimizadores de propósito general, basados en algoritmos genéticos, tienen la característica de que facilitan enormemente el proceso de modelación, y el proceso de búsqueda no tiene que ser diseñado, sino que es determinado por el propio programa, pero esta aparente ventaja es a costa de una mala calidad de las soluciones encontradas, por tanto no hay un equilibrio entre precisión, eficiencia y calidad.

El algoritmo GRASP utilizando su versión modificada; es decir GRASP_KP, permite hallar soluciones que incluso en su peor construcción son mejores que las obtenidas por un procedimiento voraz determinista, esto implica que el algoritmo es altamente eficiente y robusto, y es probable que con una sola construcción se pueda obtener una solución de excelente calidad.

REFERENCIAS BIBLIOGRÁFICAS Y ELECTRÓNICAS

[1]. **K. BRETTHAUER Y B. SHETTY**. (2002). "*The nonlinear knapsack problem - algorithms and aplications",* European Journal of Operational Research, vol. 138, nº 3, pp. 459-472.

[2]. **M. R. GAREY Y D. S. JOHNSON,** Computers and Intractability: A Guide to the Theory of NP-Completeness, New York: W.H. Freeman, 1979.

[3]. **C. H. Papadimitriou, «***On the Complexity of Integer Programming***,»** Journal of the Association for Computing Machinery, vol. 28, nº 4, pp. 765-768, 1981.

[4]. **«Optsicom.es,»** [En línea]. Available: http://www.optsicom.es/binaryss/knapsack.zip. [Último acceso: 20 4 2014].

[5]. **T. FEO Y M. RESENDE**, «A probabilistic heuristic for a computationally difficult set covering problem,» Operations Research Letters, vol. 8, pp. 67 - 71, 1989.

[6]. **M. RESENDE Y R. WERNECK,** «A hybrid heuristic for the p-median problem,» Journal of heuristics, vol. 10, nº 1, pp. 59-88, 2004.

[7]. **R. MARTÍ**, Procedimientos Metaheurísticos en Optimización Combinatoria, Universitat de València, 2008.

[8]. **F. GORTÁZAR, A. DUARTE, M. LAGUNA Y R. MARTÍ**, «Black box scatter search for general classes of binary optimization problems,» Computers & Operations Research, vol. 37, nº 11, pp. 1977 - 1986, 2010.

[9]. **PALISADE**, «Evolver -Para optimización por medio de algoritmos genéticos- Palisade,» 16 05 2011. [En línea]. Available: http://www.palisade-lta.com/evolver/. [Último acceso: 16 05 2011].

[10]. **PALISADE CORPORATION**, «Palisade Corporation,» [En línea]. Available: http://www.palisade-lta.com/. [Último acceso: Julio 2013].

[11]. **J. DRÉO, A. PÉTROWSKI, P. SIARRI Y E. TAILLARD,** Metaheuristics for Hard Optimization, Berlin: Springer, 2006, pp. 169 - 170.

[12]. **V. CAMPOS, M. LAGUNA Y R. MARTÍ,** «Context-independient scatter and tabu search for permutation problems,» INFORMS Journal on Computing, vol. 17, nº 1, pp. 111 - 122, 2005.