GRASP: Procedimientos de búsqueda miopes, aleatorizados y adaptativos*

José Luis González Velarde

Tecnológico de Monterrey
Monterrey, NL 64849 México
gonzalez.velarde@itesm.mx

1 Introducción

Consideremos un problema de optimización combinatoria definido por un conjunto base finito E = {1, ..., n}, un conjunto de soluciones factibles $F \subseteq 2^E$, y una función objetivo $f : 2^E \longrightarrow \mathbb{R}$. En la versión de minimización, buscamos una solución óptima $S^* \in F$ tal que $f(S^*) \leq f(S)$; $\forall S \in F$. El conjunto base E, la función de coste f, así como el conjunto de soluciones factibles F se definen para cada problema específico. Por ejemplo, en el caso del problema del agente viajero, el conjunto base E consta de todas las aristas que conectan las ciudades a ser visitadas, $f(S)$ es la suma de los costes de todas las aristas $e \in S$, y F está formado por todos los subconjuntos de aristas que determinan un ciclo Hamiltoniano.

Un procedimiento de búsqueda miope aleatorizado y adaptativo (GRASP por sus siglas en inglés) [42, 43] es una metaheurística para encontrar soluciones aproximadas (i.e. sub-óptimas de buena calidad, pero no necesariamente óptimas) a problemas de optimización combinatoria. Se basa en la premisa de que soluciones

*El autor agradece el apoyo recibido por el Tecnológico de Monterrey a través de la cátedra CAT025 para la preparación de este manuscrito.

iniciales diversas y de buena calidad juegan un papel importante en el éxito de métodos locales de búsqueda.

Un GRASP es un método multi-arranque, en el cual cada iteración GRASP consiste en la construcción de una solución miope aleatorizada seguida de una búsqueda local usando la solución construida como el punto inicial de la búsqueda local. Este procedimiento se repite varias veces y la mejor solución encontrada sobre todas las iteraciones GRASP se devuelve como la solución aproximada. El seudo-código en la Figura 1 ilustra un GRASP básico para minimización.

```
        procedure GRASP
f* ← ∞
for i ≤ imáx do
        x ← GreedyRandomized();
        x ← LocalSearch(x);
        if f(x) < f* then
                f* ← f(x);
                x* ← x;
        end if
        i ← i + 1
end for
return x*
```

Figura 1.

En este capítulo, nos centraremos primero en las dos componentes más importantes de GRASP. A saber, construcción y búsqueda local. Después examinaremos cómo el reencadenamiento de trayectorias puede ser usado en GRASP como un mecanismo de memoria e intensificación. El trabajo termina con una lista parcial de aplicaciones exitosas de GRASP.

Reseñas recientes de GRASP pueden encontrarse en [102, 93], una extensa bibliografía comentada está en [50] y una actualización en el URL

http://graspheuristic.org/annotated.

2 Construcciones GRASP

En esta sección describimos varios mecanismos de construcciones miopes aleatorizadas. Estos procedimientos mezclan miopía con aleatorización de diferentes formas. Todos los mecanismos de construcción considerados construyen una solución incorporando un elemento a la vez. En cada paso del proceso de cons-

trucción, se tiene a la mano una solución parcial. Un elemento que pueda seleccionarse como parte de una solución parcial se llama elemento candidato. Consideremos un problema de cubrimiento de conjuntos, donde se tiene una matriz $A = [a_{ij}]$ de ceros y unos, un coste c_j para cada columna j, y se quiere determinar un conjunto J de columnas con el menor coste total $\sum_{j \in J} c_j$ tal que para cada renglón i, al menos una columna j del conjunto tenga una entrada $a_{ij} = 1$. En este problema, una solución parcial es un conjunto de columnas que no necesariamente forman un cubrimiento. Cualquier columna que no se haya seleccionado previamente es un elemento candidato. El conjunto solución J se construye incorporando un elemento (columna) a la vez hasta que el conjunto J sea un cubrimiento.

Para determinar qué elemento candidato seleccionar enseguida para incluirse en la solución, generalmente se hace uso de una función miope. Una función miope mide la contribución local de cada elemento a la solución parcial. En el caso del cubrimiento de conjuntos, una función miope plausible es la razón entre el número p_j de filas sin cubrir, que quedarían cubiertas si la columna j se elige y la contribución c_j al coste total de elegir la columna j para la solución, esto es p_j/c_j. La elección miope sería agregar la columna con el mejor valor de la función miope.

Procedure Construcción-C
Input: k, E, $c(\_)$;
$x \leftarrow \emptyset$;
$C \leftarrow E$;
while $C \neq \emptyset$ **do**
Calcular el costo miope $c(e)$; $\forall e \in C$;
RCL $\leftarrow \{k$ elementos $e \in C$ con el menor $c(e)\}$;
Seleccionar un elemento s de RCL al azar;
$x \leftarrow x \cup \{s\}$;
Actualizar el conjunto candidato C;
end while
return x;

Figura 2

Existen varias formas posibles de introducir aleatoridad a este procedimiento. Una de las primeras ideas fue el uso de una lista restringida de candidatos (RCL) [42]. Tal lista contiene un conjunto de elementos candidatos con los mejores valores de la función miope. El siguiente candidato a ser agregado a la solución se selecciona al azar de la lista restringida de candidatos. Dicha lista puede consistir de un número fijo de elementos (restricción por cardinalidad) o elementos con los valores de la función miope dentro de un rango dado. La Figura 2 muestra

un seudo-código para un procedimiento de construcción GRASP basado en restricción por cardinalidad. Por ejemplo, denotemos por c^* y c_*, respectivamente, los valores mayor y menor de la función miope para los elementos candidatos, y sea α un número real tal que $0 \leq \alpha \leq 1$. En una lista restringida de candidatos basada en el valor, la RCL consiste en todos los elementos candidatos e cuyo valor de función miope $c(e)$ es tal que $c(e) \leq c_* + \alpha(c^* - c_*)$. Nótese que si $\alpha = 0$, entonces este esquema de selección es un algoritmo miope, mientras que si $\alpha = 1$, entonces es totalmente aleatorio. La Figura 3 muestra un seudo- código para un procedimiento de construcción GRASP basado en el valor. Más tarde será discutida la forma de determinar valores para α.

procedure Construcción-V
Input: $\alpha, E, c(\_)$;
$x \leftarrow \emptyset$;
$C \leftarrow E$;
while $C \neq \emptyset$ **do**
Calcular el costo miope $c(e)$; $\forall e \in C$;
$c_m = \min\{c(e) \mid e \in C\}$;
$c^M = \max\{c(e) \mid e \in C\}$;
RCL $\leftarrow \{e \in C \mid c(e) \leq c_m + \alpha(c^M - c_m)\}$;
Seleccionar un elemento s de RCL al azar;
$x \leftarrow x \cup \{s\}$;
Actualizar el conjunto candidato C;
end while
return x;

Figura 3

Se puede también mezclar una construcción al azar con una construcción miope de la siguiente manera. Elegir secuencialmente un conjunto parcial de elementos candidato al azar y después completar la solución usando un algoritmo miope [95]. La Figura 4 muestra un seudo-código para tal procedimiento de construcción.

procedure Construcción-RG
input: k, E, c( );
$x \leftarrow \emptyset$;
$C \leftarrow E$ **for** $i = 1, 2, ..., k$ **do**
    if $C \neq \emptyset$ **then**
        Elegir el elemento e al azar de C;
        $x \leftarrow x \cup \{e\}$;
        Actualizar el conjunto de candidatos C;
    end if
end for
while $C \neq \emptyset$ **do**
    Calcular el costo miope $c(e)$; $\forall e \in C$
    $e_+ \leftarrow \text{argmin}\{c(e) \mid e \in C\}$;
    $x \leftarrow x \cup \{e_+\}$;
    Actualizar el conjunto de candidatos C;
    Calcular el costo miope $c(e)$;$\forall e \in C$;
end while
return x;

Figura 4

Otro enfoque es mediante perturbación de costes. Aquí, los datos de costes se perturban aleatoriamente y se aplica un algoritmo miope [27]. La Figura 5 muestra un seudo-código para este procedimiento de construcción.

procedure Construcción-PG
input: $E, c(\_)$;
$x \leftarrow \emptyset$;
$C \leftarrow E$;
Perturbar aleatoriamente los datos del problema;
while $C \leftarrow E$; **do**
    Calcular el costo miope perturbado $\underline{c}(e)$; $\forall e \in C$;
    $e^* \leftarrow \text{argmin}\{c(e) \mid e \in C\}$;
    $x \leftarrow x \cup \{e^*\}$;
    Actualizar el conjunto de candidatos C;
end while
return x;

Figura 5

Un ejemplo final de un procedimiento de construcción de GRASP es una

variación sobre el enfoque de RCL basado en el valor. En este procedimiento, llamado función de sesgo [25], en vez de seleccionar el elemento de la RCL al azar con iguales probabilidades asignadas a cada elemento, se asignan diferentes probabilidades, favoreciendo elementos bien evaluados. Los elementos de la RCL se ordenan de acuerdo a los valores de la función miope.

La probabilidad $\pi(r(e))$ de seleccionar el elemento e es

$$\pi(r(e)) = \frac{sesgo(r(e))}{\sum_{e' \in RCL} sesgo(r(e'))},$$

donde $r(e)$ es la posición del elemento e en la RCL. Se han propuesto varias alternativas para asignar sesgos a los elementos. Por ejemplo,

- sesgo aleatorio: $sesgo(r) = 1$;
- sesgo lineal: $sesgo(r) = 1/r$;
- sesgo exponencial: $sesgo(r) = e^{-r}$.

En la siguiente sección, discutimos cómo determinar el valor de α para usarse en los esquemas basados en RCL anteriormente discutidos. Recordemos que si $\alpha = 0$, entonces estos esquemas de selección se convierten en un algoritmo miope, mientras que si $\alpha = 1$, son totalmente aleatorios.

3 Búsqueda local

Un algoritmo de búsqueda local explora repetidamente la vecindad de una solución en busca de una mejor solución. Cuando no se encuentra una solución que mejora la actual, se dice que la solución es localmente óptima. La Figura 6 muestra un seudo-código para un procedimiento de búsqueda local.

procedure BúsquedaLocal
input: $x_0, N(\_), f(\_)$;
$x \leftarrow x_0$;
while x no es localmente óptimo con respecto a $N(x)$ **do**
    Sea $y \in N(x)$ tal que $f(y) < f(x)$;
    $x \leftarrow y$;
end while
return x;

Figura 6

La búsqueda local juega un papel importante en GRASP ya que sirve para buscar soluciones localmente óptimas en regiones prometedoras del espacio de

soluciones. Esta es la diferenciación de GRASP con respecto del algoritmo semi-miope de Hart y Shogan [55]. Por definición su desempeño nunca será peor que semi-miope, y casi siempre producirá mejores soluciones en menos tiempo.

Aunque los algoritmos miopes pueden producir buenas soluciones razonables, su principal desventaja como generador de soluciones iniciales para búsquedas locales es su falta de diversidad. Aplicando repetidamente un algoritmo miope, una sola o muy pocas soluciones pueden generarse. Por otra parte, un algoritmo totalmente aleatorio produce una gran cantidad de soluciones diversas. Sin embargo, la calidad de estas soluciones generalmente es muy pobre y usarlas como soluciones iniciales para búsquedas locales generalmente conduce a una convergencia lenta hacia un mínimo local.

Para beneficiarse de la convergencia rápida del algoritmo miope y de la gran diversidad del algoritmo aleatorio, se acostumbra usar un valor de α estrictamente contenido en el interior del rango $[0; 1]$. Ya que no se conoce a priori qué valor usar, se han propuesto diferentes esquemas. La primera referencia en la literatura en la cual se propone una variación del parámetro α fue [66]. Los autores proponen un valor inicial del parámetro, $\alpha = 1$, con este valor se efectúan las construcciones, una vez que han transcurrido un cierto número de iteraciones sin que se haya construido una solución mejor, este valor se disminuye en una cantidad $\Delta\alpha$, esto es $\alpha = \alpha - \Delta\alpha$. Esto se repite mientras el parámetro α, no sea negativo. Otra estrategia razonable es seleccionar al azar un valor diferente en cada iteración GRASP. Esto puede hacerse usando una probabilidad uniforme [92] o usando el esquema de GRASP reactivo [88].

En el esquema de GRASP reactivo, sea $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ el conjunto de valores posibles para α. Las probabilidades asociadas con la elección de cada valor se fijan todas inicialmente iguales a $p_i = 1/m;\ i = 1; :::; m$. Más aún, sea z^* el valor de la solución incumbente, esto es, la mejor solución encontrada hasta el momento, y sea A_i el valor promedio de todas las soluciones halladas usando $\alpha = \alpha_i\ i = 1; :::; m$. Las probabilidades de selección se reevalúan periódicamente tomando $p : i = q_i / \sum_{j=1,\ldots,m} q_j$, con $q_i = z^*/A_i$ para $i = 1; :::; m$. El valor de q_i será mayor para valores de $\alpha = \alpha_i$ que produzcan las mejores soluciones en promedio. Mayores valores de q_i corresponden a valores del parámetro α más adecuados. Las probabilidades asociadas con estos valores más apropiados se incrementarán cuando sean reevaluadas.

En el contexto de GRASP, se han usado esquemas de búsqueda local más elaborados. Por ejemplo, búsqueda tabú [66, 35, 1, 101], recocido simulado [70], vecindades variables [28, 49], y vecindades extendidas [3].

4 Estructuras de memoria en GRASP

Quizás una de las principales desventajas del planteamiento original de GRASP es su falta de estructuras de memoria. Las iteraciones de GRASP son independientes y no utilizan las observaciones hechas durante iteraciones previas. Una consecuencia de esto es el hecho de que una solución previamente construida puede aparecer nuevamente, ya que esta situación es equivalente a un muestreo con reemplazo, invirtiendo tiempo de cómputo en construir soluciones repetidas.

Un remedio ha sido sugerido por Fleurent y Glover [51] quienes usan un diseño de memoria adaptativo tal como se propone en búsqueda tabú con el fin de retener y analizar características de ciertas soluciones seleccionadas y almacenadas en un conjunto élite S, y proporcionar una base para mejorar las ejecuciones futuras dentro del proceso constructivo.

Para esto definen una función de evaluación $E(e) = F(valor(e), intensidad(e))$, para cada candidato en la lista C. F es una función monótona no decreciente en sus argumentos, donde $valor(e)$ está asociada con la función objetivo. Mayores valores de esta función corresponden a mejores elecciones (así, en un problema de minimización, $valor(e)$ se incrementa si el cambio en el coste disminuye). Mientras que $intensidad(e)$, se vuelve más grande cuando e aparece con más frecuencia en los mejores miembros del conjunto élite S.

Esta función es utilizada entonces para determinar las probabilidades de elección de cada miembro de la lista C, $p(e) = E(e)/\sum_{e' \in C} E(e')$. La función de evaluación generalmente tiene la forma $E(e) = \lambda valor(e) + intensidad(e))$. Valores mayores de λ dan más énfasis a valor, que a intensidad, esto es deseable al inicio de la búsqueda, ya que no se cuenta con información suficiente para que el factor intensidad pueda ser significativo. A medida que avanza la búsqueda y se tiene información dentro del conjunto de soluciones élite, el valor de λ puede disminuirse. Ya que la intensificación de alguna manera enfatiza la calidad de las soluciones a costa de la aleatorización, generalmente se acompaña con una componente de diversificación que periódicamente conduce la búsqueda hacia la exploración de diferentes regiones del espacio de búsqueda. En este caso, la diversificación se puede lograr incrementando λ, lo cual se hace cuando la diversidad de las soluciones generadas es muy baja.

Otra alternativa es el uso del reencadenamiento de trayectorias (path relinking) con GRASP. El reencadenamiento de trayectorias fue propuesto originalmente por Glover [53] como una forma de explorar las trayectorias entre soluciones élite obtenidas por búsqueda tabú o búsqueda dispersa. Usando una o más soluciones élite, se exploran las trayectorias en el espacio de soluciones que conducen a otras soluciones élite para buscar mejores soluciones. Para generar trayectorias, los movimientos se seleccionan para introducir atributos en la solución actual que estén presentes en la solución élite guía.

El reencadenamiento de trayectorias en el contexto de GRASP fue introducido por Laguna y Martí [70]. Desde entonces han aparecido numerosas extensiones, mejoras, y aplicaciones exitosas [5, 27, 94, 97, 95, 49]. Ha sido usado como un esquema de intensificación, en el que las soluciones generadas en cada iteración de GRASP se reencadenan con una o más soluciones de un conjunto élite de soluciones, o como en una fase de post-optimización, donde se reencadenan pares de soluciones del conjunto élite.

Consideremos dos soluciones x_s y x_t en las cuales queremos aplicar reencadenamiento de trayectorias desde x_s hacia x_t. La Figura 7 ilustra el procedimiento de reencadenamiento de trayectorias mediante su seudo-código. El procedimiento se inicia calculando la diferencia simétrica $\Delta(x_s; x_t)$ entre las dos soluciones, i.e. el conjunto de movimientos necesarios para alcanzar x_t desde x_s. Se genera entonces una trayectoria de soluciones encadenando a x_s con x_t. El algoritmo devuelve la mejor solución encontrada en esta trayectoria. En cada paso, el procedimiento examina todos los movimientos $m \in \Delta(x; x_t)$ desde la solución actual x y elige aquel que resulta en la solución menos costosa, i.e. aquel que minimiza $f(x \oplus m)$, donde $x \oplus m$ es la solución resultante de aplicar el movimiento m a la solución x. Se efectúa el mejor movimiento m^* produciendo $x \oplus m^*$ y el movimiento m^* se elimina de la diferencia simétrica de $\Delta(x \oplus m; x_t)$. En caso necesario, la mejor solución x^* se actualiza. El procedimiento termina cuando se alcanza x_t, i.e. cuando $\Delta(x; x_t) = \emptyset$.

procedure PR
input: x_s, x_t;
Calcular la diferencia simétrica $\Delta(x_s, x_t)$;
$x \leftarrow x_s$;
$f^* \leftarrow \text{mín}\{f(x_s), f(x_t)\}$;
$x^* \leftarrow \text{argmín}\{f(x_s), f(x_t)\}$;
while $\Delta(x_s, x_t) \neq \emptyset$; **do**
    $m^* \leftarrow \text{argmin}\{f(x \oplus m); \forall m \in \Delta(x, x_t)\}$;
    $\Delta(x \oplus m^*, x_t) \leftarrow \Delta(x, x_t) \setminus \{m^*\}$;
    $x \leftarrow x \oplus m^*$;
    if $f(x) < f^*$ **then**
        $f^* \leftarrow f(x)$;
        $x^* \leftarrow x$;
    end if
end while
return x^*;

Figura 7

La figura 8 ilustra el reencadenamiento de trayectorias.En el grafo de esta

figura, los nodos corresponden a soluciones, y los arcos corresponden a movimientos que permiten que una solución se alcance a partir de otra. Supóngase que dos soluciones, A y D, se reencadenan. Sea A la solución inicial y D la solución meta, y supóngase que la diferencia simétrica $\Delta(A; D) = 3$. Existen tres posibles movimientos partiendo de A. Se elige el mejor movimiento, el cual produce la solución B. En este punto, la diferencia simétrica $\Delta(B; D) = 2$ y por lo tanto existen dos movimientos posibles. De nuevo, se elige el mejor movimiento, el cual produce la solución C. Finalmente, en este punto hay un solo movimiento posible, el cual conduce a la solución meta D. Este esquema de reencadenamiento de trayectorias produjo una "trayectoria" $A \rightarrow B \rightarrow C \rightarrow D$ la cual puede ahora ser evaluada.

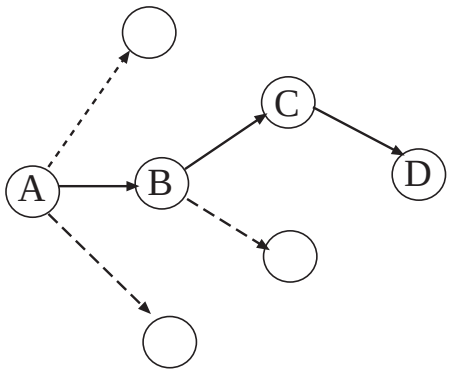

Figura 8

El reencadenamiento de trayectorias mantiene un conjunto P de soluciones élite halladas durante la optimización [51]. Las primeras $|P|$ soluciones distintas que se encuentran se insertan en el conjunto élite. Después de eso, una solución candidato x^* se agrega a P si su costo es menor que el costo de todas las soluciones del conjunto élite, o si su costo es mayor que el mejor, pero menor que la peor solución élite y es suficientemente diferente de todas las soluciones del conjunto élite. Si se acepta su entrada al conjunto élite, la nueva solución reemplaza a la solución más similar a ella entre el conjunto de soluciones élite con un costo peor que ella [95]. El conjunto élite puede ser renovado periódicamente [4] si no se observan cambios en el conjunto élite durante un número especificado de iteraciones GRASP. Una forma de hacer esto es fijar en infinito los valores de la función objetivo de la peor mitad del conjunto élite. De esta forma se crearán nuevas soluciones del conjunto élite.

Se han propuesto varios esquemas alternativos para el reencadenamiento de trayectorias. Ya que este procedimiento puede ser demandante en recursos computacionales, no necesita ser aplicado después de cada iteración GRASP. Es más

conveniente hacerlo periódicamente. Usualmente las trayectorias desde x_s hasta x_t y desde x_t hasta x_s son diferentes y ambas pueden explorarse. Ya que las trayectorias pueden ser largas, no es necesario seguir la trayectoria completa. Se puede restringir siguiendo una trayectoria truncada que inicie en x_s y otra que inicie en x_t.

5 GRASP Continuo

Recientemente Hirsch et al [57, 58, 59, 60], introdujeron un método de optimización global el cual extiende GRASP del dominio de la optimización discreta al de la optimización global continua. El dominio de la función se supone inmerso en un hiper-rectángulo de dimensión n, donde n es el número de variables. En el inicio de cada iteración GRASP, se genera de forma aleatoria una solución x dentro del rectángulo. La fase de construcción se inicia con esta solución dejando en libertad de variar todas sus coordenadas, en cada una de las coordenadas libres i se lleva a cabo una búsqueda lineal manteniendo las otras $n-1$ coordenadas de x en sus valores actuales. El valor de z_i de la i-ésima coordenada, que minimice el valor de la función objetivo, así como el valor de la función objetivo g_i se almacenan. Una vez que se ha llevado a cabo para cada una de las coordenadas libres la búsqueda local, se forma una lista restringida de candidatos (LRC) la cual contiene la coordenadas libres i cuyos valores g_i son menores o iguales a $\alpha\max + (1-\alpha)\min$, donde max y min son, respectivamente los valores máximo y mínimo de g_i sobre todas las coordenadas libres de x, y $\alpha \in [0;1]$. De la LRC se elige una coordenada al azar, digamos $j \in$ LRC, y x_j se fija a z_j, quedando $n-1$ variables libres. Eligiendo una coordenada de esta manera se asegura la aleatoreidad en la fase de construcción. Se continúa con el procedimiento anteriormente descrito hasta que todas las n coordenadas de x se hayan fijado. En este punto se ha obtenido x de la fase de construcción. Para la fase de post-procesamiento la cual consiste en una búsqueda local, iniciando desde un punto $x \in \mathbb{R}^n$, el algoritmo genera un conjunto de direcciones y determina en qué dirección, si es que hay una, mejora la función objetivo. Es fácil ver que existen $3^n - 1$ direcciones posibles y aun para valores moderados de n este número puede ser muy grande. Es por esto que se fija un cierto número máximo de direcciones a explorar, Ndir, y se construyen al azar este máximo número de direcciones. Una vez construida la dirección d, se genera el punto de prueba $x^* + hd$ donde h es un parámetro de discretización. Si el punto de prueba x es factible y es mejor que x^*, entonces a x^* se le asigna el valor de x y el proceso vuelve a comenzar con x^* como la solución inicial. Es importante notar que el conjunto de direcciones puede cambiar cada vez durante el proceso, así como el orden en el cual estas direcciones se consideran. Una vez que se encuentra un punto con $f(x^*) \leq f(x^* + hd)$ para cada una de las Ndir d elegidas, se declara a x^* localmente óptima y el procedimiento regresa

esta solución. Esto corresponde a una iteración de construcción de GRASP, el procedimiento se repite *maxiter* veces y se regresa la mejor de todas las soluciones construidas.

6 Aplicaciones

La primera aplicación de GRASP fue a cubrimientos de conjuntos [42] en 1989, y, a partir de entonces, ha sido aplicado a un amplio rango de tipos de problemas. Referimos al lector a Festa y Resende [50] y al URL

http://graspheuristic.org/annotated

para una extensa bibliografía comentada de GRASP. Concluimos este capítulo con una lista parcial de aplicaciones de GRASP, mostrando su amplia aplicabilidad.

- enrutamiento [11, 14, 19, 29, 64];
- lógica [36, 86, 90, 91];
- cubrimiento y partición [11, 12, 42, 52, 54];
- localización [1, 35, 70, 102, 103];
- árbol mínimo de Steiner [28, 75, 76, 77, 97];
- optimización en grafos [2, 44, 65, 84, 89, 93, 96, 48, 65, 32];
- asignación [41, 51, 66, 70, 78, 81, 85, 88, 87, 100, 56];
- horarios, programación, y manufactura [16, 17, 18, 21, 33, 37, 38, 39, 40, 45, 46, 66, 98, 99, 105, 6];
- transporte [11, 38, 41, 71, 20];
- sistemas de potencia [22, 23, 15];
- telecomunicaciones [2, 13, 64, 70, 88, 89, 94, 26, 82, 104, 65, 72];
- diseño de redes [8], [34];
- dibujo de grafos y mapas [47, 67, 93, 96, 73, 74, 24];
- lenguaje [31];
- estadística [79, 80];
- biología [9];
- programación matemática [83];
- empaquetado [30]; y
- VLSI [10], entre otras áreas de aplicación.

7 Referencias bibliográficas

[1] S. Abdinnour-Helm and S.W. Hadley. Tabu search based heuristics for multi-floor facility layout. International Journal of Production Research, 38:365-383, 2000.

[2] J. Abello, P.M. Pardalos, and M.G.C. Resende. On maximum clique problems in very large graphs. In J. Abello and J. Vitter, editors, External Memory Algorithms and Visualization, volume 50 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 199-130. American Mathematical Society, 1999.

[3] R.K. Ahuja, J.B. Orlin, and D. Sharma. New neighborhood search structures for the capacitated minimum spanning tree problem. Mathematical Programming, 91:71-97, 2001.

[4] R.M. Aiex, S. Binato, and M.G.C. Resende. Parallel GRASP with path relinking for job shop scheduling. Parallel Computing, 29:393- 430, 2003.

[5] R.M. Aiex, M.G.C. Resende, P.M. Pardalos, and G. Toraldo. GRASP with path relinking for the three-index assignment problem. Technical report, AT&T Labs-Research, 2000. To appear in INFORMS J. on Comp.

[6] M.S. Akturk and D. Ozdemir. A new dominance rule to minimize total weighted tardiness with unequal release dates. European Journal of Operational Research, 135:394-412, 2001.

[7] Álvarez, A., González Velarde, J.L., de Alba, K. GRASP Embedded Scatter Search for the Multicommodity Capacitated Network Design Problem. Journal of Heuristics, 11:233-257, 2005.

[8] A. Andreatta and C.C. Ribeiro. Heuristics for the phylogeny problem. Journal of Heuristics, 8:429-447, 2002.

[9] S. Areibi and A. Vannelli. A GRASP clustering technique for circuit partitioning. In J. Gu and P.M. Pardalos, editors, Satisfiability Problems, volume 35 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 711-724. American Mathematical Society, 1997.

[10] M.F. Argüello, J.F. Bard, and G. Yu. A GRASP for aircraft routing in response to groundings and delays. Journal of Combinatorial Optimization, 1:211-228, 1997.

[11] M.F. Argüello, T.A. Feo, and O. Goldschmidt. Randomized methods for the number partitioning problem. Computers and Operations Research, 23:103-111, 1996.

[12] M. Armony, J.C. Klincewicz, H. Luss, and M.B. Rosenwein. Design of stacked selfhealing rings using a genetic algorithm. Journal of Heuristics, 6:85-105, 2000.

[13] J.B. Atkinson. A greedy randomised search heuristic for time- constrained vehicle scheduling and the incorporation of a learning strategy. Journal of the Operational Research Society, 49:700-708, 1998.

[14] L. Bahiense, G.C. Oliveira, and M. Pereira. A mixed integer disjunctive model for transmission network expansion. IEEE Transactions on Power Systems, 16:560-565, 2001.

[15] J.F. Bard and T.A. Feo. Operations sequencing in discrete parts manufacturing. Management Science, 35:249-255, 1989.

[16] J.F. Bard and T.A. Feo. An algorithm for the manufacturing equipment selection problem. IIE Transactions, 23:83-92,1991.

[17] J.F. Bard, T.A. Feo, and S. Holland. A GRASP for scheduling printed wiring board assembly. IIE Transactions, 28:155-165, 1996.

[18] J.F. Bard, L. Huang, P. Jaillet, and M. Dror. A decomposition approach to the inventory routing problem with satellite facilities. Transportation Science, 32:189–203, 1998.

[19] J.F. Bard, G. Kantoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. Transportation Science, 36:250-269, 2002.

[20] S. Binato, W.J. Hery, D. Loewenstern, and M.G.C. Resende. A GRASP for job shop scheduling. In C.C. Ribeiro and P. Hansen, editors, Essays and Surveys in Metaheuristics, pages 59-79. Kluwer Academic Publishers, 2002.

[21] S. Binato and G.C. Oliveira. A reactive GRASP for transmission network expansion planning. In C.C. Ribeiro and P. Hansen, editors, Essays and Surveys in Metaheuristics, pages 81-100. Kluwer Academic Publishers, 2002.

[22] S. Binato, G.C. Oliveira, and J.L. Araújo. A greedy randomized adaptive search procedure for transmission expansion planning. IEEE Transactions on Power Systems, 16:247-253, 2001.

[23] C. Binucci, W. Didimo, G. Liotta, and M. Ñonato. Labeling heuristics for orthogonal drawings. In Proceedings of GD'98 - Symposium on Graph Drawing, volume 2265 of Lecture Notes in Computer Science, pages 139-153. Springer-Verlag, 2002.

[24] J.L. Bresina. Heuristic-biased stochastic sampling. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pages 271-278, Portland, 1996.

[25] M. Brunato and R. Battiti. A multistart randomized greedy algorithm for traffic grooming on mesh logical topologies. Technical report, Department of Mathematics, University of Trento, Trento, Italy, 2001.

[26] S.A. Canuto, M.G.C. Resende, and C.C. Ribeiro. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. Networks, 38:50-58, 2001.

[27] S.A. Canuto, C.C. Ribeiro, and M.G.C. Resende. Local search with perturbations for the prize-collecting Steiner tree problem. In Extended Abstracts of the Third Metaheuristics International Conference, pages 115-119, Angra dos Reis, July 1999.

[28] C. Carreto and B. Baker. A GRASP interactive approach to the vehicle routing problem with backhauls. In C.C. Ribeiro and P. Hansen, editors, Essays and Surveys in Metaheuristics, pages 185- 199. Kluwer Academic Publishers, 2002.

[29] P. Chardaire, G.P. McKeown, and J.A. Maki. Application of GRASP to the multiconstraint knapsack problem. In E.J.W. Boers et al., editor, Evo-Workshop 2001, pages 30–39. Springer-Verlag Berlin Heidelberg, 2001.

[30] H. Fraçois and O. Boëffard. The greedy algorithm and its application to the construction of a continuous speech database. In Proceedings of LREC-2002, volume 5, pages 1420-1426, May 29-31 2002.

[31] A. Corberán, R. Martí, and J.M. Sanchís. A GRASP heuristic for the mixed Chinese postman problem. European Journal of Operational Research, 142:70-80, 2002.

[32] P. De, J.B. Ghosj, and C.E. Wells. Solving a generalized model for con due date assignment and sequencing. International Journal of Production Economics, 34:179-185, 1994.

[33] De Alba, K. Álvarez, A., González Velarde, J.L. GRASP with Adaptive Memory for finding good starting solutions for the capacitated network design problem, en el libro Computational Modeling and Problem Solving in the Networked World:, pp 121 - 137. Series Interfaces in Computing and Optimization Editors: Hemant K. Bhargava and Nong Ye, Kluwer 2003

[34] H. Delmaire, J.A. Díaz, E. Fernández, and M. Ortega. Reactive GRASP and Tabu Search based heuristics for the single source capacitated plant location problem. INFOR, 37:194-225, 1999.

[35] A.S. Deshpande and E. Triantaphyllou. A greedy randomized adaptive search procedure (GRASP) for inferring logical clauses from examples in polynomial time and some extensions. Mathematical Computer Modelling, 27:75-99, 1998

[36] A. Drexl and F. Salewski. Distribution requirements and compactness constraints in school timetabling. European Journal of Operational Research, 102:193-214, 1997.

[37] T.A. Feo and J.F. Bard. Flight scheduling and maintenance base planning. Management Science, 35:1415-1432, 1989.

[38] T.A. Feo and J.F. Bard. The cutting path and tool selection problem in computer aided process planning. Journal of Manufacturing Systems, 8:17-26, 1989.

[39] T.A. Feo, J.F. Bard, and S. Holland. Facility-wide planning and scheduling of printed wiring board assembly. Operations Research, 43:219-230, 1995.

[40] T.A. Feo and J.L. González-Velarde. The intermodal trailer assignment problem: Models, algorithms, and heuristics. Transportation Science, 29:330-341,1995.

[41] T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally dificult set covering problem. Operations Research Letters, 8:67–71, 1989.

[42] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. Journal of Global Optimization, 6:109-133, 1995

[43] T.A. Feo, M.G.C. Resende, and S.H. Smith. A greedy randomized adaptive search procedure for maximum independent set. Operations Research, 42:860-878, 1994.

[44] T.A. Feo, K. Sarathy, and J. McGahan. A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties. Computers and Operations Research, 23:881-895, 1996.

[45] T.A. Feo, K. Venkatraman, and J.F. Bard. A GRASP for a difficult single machine scheduling problem. Computers and Operations Research, 18:635-643, 1991.

[46] E. Fernández and R. Martí. GRASP for seam drawing in mosaicking of aerial photographic maps. Journal of Heuristics, 5:181-197, 1999.

[47] P. Festa, P.M. Pardalos, and M.G.C. Resende. Algorithm 815: FORTRAN subroutines for computing approximate solution to feedback set problems using GRASP. ACM Transactions on Mathematical Software, 27:456-464, 2001.

[48] P. Festa, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Randomized heuristics for the MAX-CUT problem. Optimization Methods & Software, 7:1033-1058, 2002.

[49] P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In C.C. Ribeiro and P. Hansen, editors, Essays and Surveys in Metaheuristics, pages 325-367. Kluwer Academic Publishers, 2002.

[50] C. Fleurent and F. Glover. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. INFORMS Journal on Computing, 11:198-204, 1999.

[51] J.B. Ghosh. Computational aspects of the maximum diversity problem. Operations Research Letters, 19:175-181, 1996.

[52] F. Glover. Tabu search and adaptive memory programing : Advances, applications and challenges. In R.S. Barr, R.V. Helgason, and J.L. Kennington, editors Interfaces in Computer Science and Operations Research, pages 1-75. Kluwer, 1996.

[53] P.L. Hammer and D.J. Rader, Jr. Maximally disjoint solutions of the set covering problem. Journal of Heuristics, 7:131-144, 2001.

[54] J.P. Hart and A.W. Shogan. Semi-greedy heuristics: An empirical study. Operations Research Letters, 6:107-114, 1987.

[55] M. Hasan, I. Osman, and T. AlKhamis. A meta-heuristic procedure for the three dimension assignment problem. International Journal of Applied Mathematics, 8:365-380, 2002.

[56] Hirsch, M.J., Meneses, C.N., Pardalos, P.M., and Resende, M.G.C., Global Optimization by Continuous Grasp, to appear in Optimization Letters, 2006.

[57] M. J. Hirsch, P. M. Pardalos, and M. G. C. Resende, Solving systems of nonlinear equations using Continuous GRASP. Submitted to Journal of Heuristics, 2006.

[58] M. J. Hirsch, P. M. Pardalos, and M. G. C. Resende. Speeding up Continuous GRASP. Submitted to European Journal of Operational Research, 2006.

[59] M. J. Hirsch, C. N. Meneses, P. M. Pardalos, M. A. Ragle, M. G. C. Resende A continuous GRASP to determine the relationship between drugs and adverse reactions. Optimization on Line, http://www.optimization-online.org/DB_HTML/2006/10/1495.html, 2006

[60] J.G. Klincewicz. Avoiding local optima in the p-hub location problem using tabu search and GRASP. Annals of Operations Research, 40:283-302, 1992.

[61] J.G. Klincewicz. Enumeration and search procedures for a hub location problem with economies of scale. Annals of Operations Research, 110:107-122, 2002.

[62] J.G. Klincewicz and A. Rajan. Using GRASP to solve the component grouping problem. Naval Research Logistics, 41:893-912, 1994.

[63] G. Kontoravdis and J.F. Bard. A GRASP for the vehicle routing problem with time windows. ORSA Journal on Computing, 7:10-23, 1995.

[64] M. Laguna, T.A. Feo, and H.C. Elrod. A greedy randomized adaptive search procedure for the two-partition problem. Operations Research, 42:677-687, 1994.

[65] M. Laguna and J.L. González Velarde. A search heuristic for just-in-time scheduling in parallel machines. Journal of Intelligent Manufacturing, 2:253-260, 1991.

[66] M. Laguna and R. Martí. GRASP and path relinking for 2- layer straight line crossing minimization. INFORMS Journal on Computing, 11:44-52, 1999.

[67] M. Laguna and R. Martí. A GRASP for coloring sparse graphs. Computational Optimization and Applications, 19:165-178, 2001.

[68] Y. Li, P.M. Pardalos, and M.G.C. Resende. A greedy randomized adaptive search procedure for the quadratic assignment problem. In P.M. Pardalos and H. Wolkowicz, editors, Quadratic Assignment and Related Problems, volume 16 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 237-261. American Mathematical Society, 1994.

[69] X. Liu, P.M. Pardalos, S. Rajasekaran, and M.G.C. Resende. A GRASP for frequency assignment in mobile radio networks. In B.R. Badrinath, F. Hsu, P.M. Pardalos, and S. Rajasejaran, editors, Mobile Networks and Computing, volume 52 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 195- 201. American Mathematical Society, 2000.

[70] H. Ramalhinho Lourenço, J.P. Paixão, and R. Portugal. Multiobjective metaheuristics for the bus-driver scheduling problem. Transportation Sciences, 35:331-343, 2001.

[71] P. Mahey and C.C. Ribeiro. Optimal routing for multi- service communication networks. OR/MS Today, 29:40-43, 2002.

[72] R. Martí. Arc crossing minimization in graphs with GRASP. IIE Transactions, 33:913-919, 2001.

[73] R. Martí and V. Estruch. Incremental bipartite drawing problem. Computers and Operations Research, 28:1287-1298, 2001.

[74] S.L. Martins, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Greedy randomized adaptive search procedures for the Steiner problem in graphs. In P.M. Pardalos, S. Rajasejaran, and J. Rolim, editors, Randomization Methods in Algorithmic Design, volume 43 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 133-145. American Mathematical Society, 1999.

[75] S.L. Martins, M.G.C. Resende, C.C. Ribeiro, and P. Pardalos. A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. Journal of Global Optimization, 17:267-283, 2000.

[76] S.L. Martins, C.C. Ribeiro, and M.C. Souza. A parallel GRASP for the Steiner problem in graphs. In A. Ferreira and J. Rolim, editors, Proceedings of IRREGULAR'98 - 5th International Symposium on Solving Irregularly Structured Problems in Parallel, volume 1457 of Lecture Notes in Computer Science, pages 285-297. Springer- Verlag, 1998.

[77] T. Mavridou, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A GRASP for the biquadratic assignment problem. European Journal of Operational Research, 105:613-621, 1998.

[78] M.C. Medeiros, M.G.C. Resende, and A. Veiga. Piecewise linear time series estimation with GRASP. Computational Optimization and Applications, 19:127-144, 2001.

[79] M.C. Medeiros, A. Veiga, and M.G.C. Resende. A combinatorial approach to piecewise linear time analysis. Journal of Computational and Graphical Statistics, 11:236-258, 2002.

[80] R.A. Murphey, P.M. Pardalos, and L.S. Pitsoulis. A parallel GRASP for the data association multidimensional assignment problem. In P.M. Pardalos, editor, Parallel Processing of Discrete Problems, volume 106 of The IMA Volumes in Mathematics and Its Applications, pages 159-180. Springer-Verlag, 1998.

[81] A. Myslek. Greedy randomised adaptive search procedures (GRASP) for topological design of mpls networks. In Proceedings of the 8th Polish Teletraffic Symposium, 2001.

[82] G. Palubeckis and A. Tomkevicius. GRASP implementations for the unconstrained binary quadratic optimization problem. Information Technology and Control, 24:14-20, 2002.

[83] P. M. Pardalos, T. Qian, and M. G. C. Resende. A greedy randomized adaptive search procedure for the feedback vertex set problem. Journal of Combinatorial Optimization, 2:399-412, 1999.

[84] P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A parallel GRASP implementation for the quadratic assignment problem. In A. Ferreira and J. Rolim, editors, Parallel Algorithms for Irregularly Structured Problems - Irregular'94, pages 115-133. Kluwer Academic Publishers, 1995.

[85] P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A parallel GRASP for MAXSAT problems. Lecture Notes in Computer Science, 1184:575-585, 1996.

[86] L.S. Pitsoulis, P.M. Pardalos, and D.W. Hearn. Approximate solutions to the turbine balancing problem. European Journal of Operational Research, 130:147-155, 2001.

[87] M. Prais and C.C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. INFORMS Journal on Computing, 12:164-176, 2000.

[88] M.G.C. Resende. Computing approximate solutions of the maximum covering problem using GRASP. Journal of Heuristics, 4:161-171, 1998.

[89] M.G.C. Resende and T.A. Feo. A GRASP for satisfiability. In D.S. Johnson and M.A. Trick, editors, Cliques, Coloring, and Satisfiability: The Second DIMACS Implementation Challenge, volume 26 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 499-520. American Mathematical Society, 1996.

[90] M.G.C. Resende, L.S. Pitsoulis, and P.M. Pardalos. Approximate solution of weighted MAX-SAT problems using GRASP. In J. Gu and P.M. Pardalos, editors, Satisfiability Problems, volume 35 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 393-405. American Mathematical Society, 1997.

[91] M.G.C. Resende, L.S. Pitsoulis, and P.M. Pardalos. Fortran subroutines for computing approximate solutions of MAXSAT problems using GRASP. Discrete Applied Mathematics, 100:95-113, 2000.

[92] M.G.C. Resende and C.C. Ribeiro. A GRASP for graph planarization. Networks, 29:173-189, 1997.

[93] M.G.C. Resende and C.C. Ribeiro. A GRASP with path- relinking for private virtual circuit routing. Networks, 41(1):104-114, 2003.

[94] M.G.C. Resende and R.F. Werneck. A GRASP with path- relinking for the p-median problem. Technical report, Internet and Network Systems Research Center, AT&T Labs Research, Florham Park, NJ, 2002.

[95] C.C. Ribeiro and M.G.C. Resende. Algorithm 797: Fortran subroutines for approximate solution of graph planarization problems using GRASP. ACM Transactions on Mathematical Software, 25:342-352, 1999.

[96] C.C. Ribeiro, E. Uchoa, and R.F. Werneck. A hybrid GRASP with perturbations for the Steiner problem in graphs. INFORMS Journal on Computing, 14:228-246, 2002.

[97] R.Z. Ríos Mercado and J.F. Bard. Heuristics for the flow line problem with setup costs. European Journal of Operational Research, pages 76-98, 1998.

[98] R.Z. Ríos Mercado and J.F. Bard. An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup costs. Journal of Heuristics, 5:57-74, 1999.

[99] A.J. Robertson. A set of greedy randomized adaptive local search procedure (GRASP) implementations for the multidimensional assignment problem. Computational Optimization and Applications, 19:145-164, 2001.

[100] D. Serra and R. Colomé. Consumer choice in competitive location models: Formulations and heuristics. Papers in Regional Science, 80:439-464, 2001.

[101] T.L. Urban. Solution procedures for the dynamic facility layout problem. Annals of Operations Research, pages 323-342, 1998.

[102] T.L. Urban, W.-C. Chiang, and R.A. Russel. The integrated machine allocation and layout problem. International Journal of Production Research, pages 2913-2930, 2000.

[103] J. Xu and S. Chiu. Effective heuristic procedure for a field technician scheduling problem. Journal of Heuristics, 7:495-509, 2001.

[104] J. Yen, M. Carlsson, M. Chang, J.M. Garcia, and H. Nguyen. Constraint solving for inkjet print mask design. Journal of Imaging Science and Technology, 44:391-397, 2000.