CAPÍTULO 3

GRASP (Greedy Randomized Adaptive Search Procedures).

Los problemas de optimización surgen de las situaciones de aplicación práctica. Estos problemas se aplican en distintas áreas, tales como: telecomunicación, logística, finanzas, transportación y producción. Aunque desde el punto de vista teórico es posible enumerar todas las posibles soluciones y evaluar cada una de ellas con respecto al valor de la función objetivo, en la práctica es imposible analizar y evaluar todas las posibles soluciones ya que el número de combinaciones crece exponencialmente con el tamaño del problema. También pueden existir restricciones de tiempo para el proceso de toma de decisiones (ver [9], [10], [11]). En las últimas décadas se han realizado grandes esfuerzos para desarrollar métodos que no requieran enumerar explícitamente todas las soluciones. Sin embargo, para un gran número de aplicaciones, los métodos exactos continúan siendo de poca utilidad, debido al esfuerzo enumerativo requerido. Desde un punto de vista práctico, en esas situaciones es deseable contar con métodos de solución que puedan proporcionar soluciones de calidad con un esfuerzo computacional razonable. La calidad de las soluciones se puede probar utilizando cotas para el valor de la solución óptima. Este hecho ha motivado el desarrollo de métodos de aproximación en donde el objetivo es buscar soluciones de calidad en un tiempo razonable [9].

3.1 Métodos Heurísticos.

En los últimos años ha habido un crecimiento en el desarrollo de procedimientos heurísticos para resolver problemas de optimización combinatoria. El auge que

experimentan los procedimientos heurísticos se debe a la necesidad de disponer de herramientas que permitan ofrecer soluciones rápidas a problemas reales. Es importante destacar el hecho de que los algoritmos heurísticos no garantizan la optimalidad, aunque su propósito es encontrar una solución cercana al óptimo en un tiempo razonable.

Un método heurístico se puede describir como un procedimiento que se basa en un conjunto de ideas intuitivas, que sin ser rigurosos, puedan ser utilizadas en la búsqueda de soluciones de calidad para un problema dado. La forma más simple de los métodos heurísticos son la *búsqueda local* y los *procedimientos voraces*. El objetivo de la búsqueda local es mejorar una solución factible inicial para un problema, mientras que un procedimiento voraz es un método constructivo que permite obtener una solución factible para un problema dado. La búsqueda por mejores métodos de aproximación, combinado con el desarrollo en la tecnología de la computación, ha dado origen la que se conoce como métodos meta heurísticos [9], [12].

3.1.1 Búsqueda Local.

La Búsqueda Local es una técnica utilizada para mejorar una solución inicial para un problema de optimización. Es un proceso iterativo que consiste en lo siguiente. En cada iteración se busca una solución vecina de la solución actual que mejore el valor de la función objetivo. Para cada solución $x \in X$ se define una estructura de vecindad $N(x) \subset X$ tal que cada solución $x' \in N(x)$ puede ser alcanzada a partir de x mediante una operación simple denominada *movimiento*. El proceso iterativo se repite hasta que ya no

se puede encontrar una solución vecina que mejore el valor de la función objetivo de la solución actual. [9].

En la figura 3.1 se muestra, en su forma más general, un algoritmo de búsqueda local.

Algoritmo de búsqueda local.

Sea *f(x)* la función a minimizar, *x* una solución factible y *N(x)* una estructura de vecindad.

CriterioParada ← falso

mientras (no se satisfaga *CriterioParada*) **hacer**

$x' = \arg\min \{ f(y) : y \in N(x) \}$

si $(f(x') < f(x))$ entonces

$x \leftarrow x'$

En otro caso

CriterioParada ← cierto

fin sí

fin mientras

Figura 3.1

3.1.2 Algoritmo Voraz.

También es un procedimiento iterativo que empieza con una solución vacía. En cada iteración se añade un elemento a la solución, y el procedimiento termina cuando se obtiene una solución factible. Para la selección de los candidatos se utiliza una función voraz que mide el beneficio de añadir el elemento a la solución. En la figura 3.2 se muestra, en su forma más general, un algoritmo voraz.

Algoritmo voraz.

Sean S una solución parcial, E el conjunto factible de elementos que pueden ser añadidos a la solución parcial, e_i los elementos del conjunto E y g una función voraz.

$S \leftarrow \varnothing$

mientras (S no factible) **hacer**

Evaluar la función voraz g para cada elemento de E

Seleccionar el mejor elemento $e \in E$ de acuerdo con el valor de la función voraz g

(i.e. $e = \arg\min \{ g(e_i) \ \forall \ i = 1,.....|E| \}$)

$S \leftarrow S \cup \{e\}$

Actualizar E

fin mientras

Figura 3.2

3.1.3 GRASP.

La palabra GRASP proviene de las siglas de Greedy Randomized Adaptive Search Procedures que se puede traducir como: Procedimientos de Búsqueda Voraces Aleatorizados y Adaptativos (ver figura 3.3).

El método GRASP se introduce inicialmente por Feo y Resende en [12]. Es un procedimiento iterativo que consiste en: una fase de construcción (ver figura 3.4) y una fase de búsqueda local. Se obtiene una solución factible durante la fase de construcción aplicando un procedimiento voraz. En cada iteración del procedimiento voraz se agrega un nuevo elemento a la solución de acuerdo al valor de una función voraz. En lugar de escoger siempre el mejor elemento candidato (i.e. $e \in E$ tal que e = argmin $\{g(e_i)$ *para* $i=1.....,|E|\})$, se construye una lista con los mejores candidatos, de donde se selecciona uno aleatoriamente. El término adaptativo se refiere al hecho de que los beneficios asociados con cada elemento son actualizados en cada iteración de la fase de construcción para reflejar los cambios producidos por selecciones previas. Una vez que se construye una solución utilizando un procedimiento voraz, se realiza un procedimiento de búsqueda local.

Algoritmo GRASP.

mientras (el criterio no satisfecho) **hacer**

Construye una solución inicial usando el procedimiento voraz

Realiza una búsqueda local para mejorar la solución construida.

fin mientras.

Figura 3.3

Algoritmo de fase de construcción del GRASP.

Sea f una función voraz, α es valor del parámetro para controlar la aleatoriedad del procedimiento, x una solución parcial, C el conjunto de elementos candidato y RCL la lista restringida de candidatos.

$x \leftarrow \varnothing$

Inicializa el conjunto de candidatos C

mientras $(x \text{ infactible})$ hacer

$a = min\{f(t) | t \in C\}$

$b = max\{f(t) | t \in C\}$

$RCL = \{c \in C | f(c) \leq a + \alpha(b-a)\}$

Seleccionar aleatoriamente un elemento $c \in RCL$

$x \leftarrow x \cup \{c\}$

Actualizar C

fin mientras.

Figura 3.4