COMPARACIÓN DE TRES MÉTODOS DE BÚSQUEDA INCOMPLETOS PARA EL PROBLEMA DE LA MOCHILA

ALFONSO H., BERTONE E., MINETTI G.
Proyecto UNLPAM-09/F009[1]
Dpto. de Informática - Facultad de Ingeniería - Universidad Nacional de La Pampa
Calle 110 esq. 9 - (6360) General Pico – La Pampa – Rep. Argentina
E-mail: {alfonsoh, bertonee, minettig }@ing.unlpam.edu.ar
Phone: (02302)422780/422372, Ext. 6302

GALLARD R.
Proyecto UNSL-338403[2]
Departamento de Informática - Universidad Nacional de San Luis
Ejército de los Andes 950 - Local 106 – (5700) San Luis – Rep. Argentina
E-mail: rgallard@unsl.edu.ar
Phone: +54 2652 420823 - Fax : +54 2652 430224

Abstract

El objetivo de éste trabajo es analizar el comportamiento de distintos algoritmos de búsqueda incompletos en un problema clásico de optimización combinatorial. El problema seleccionado es el problema de la mochila, clasificado como NP-duro. Siendo los algoritmos evaluados: el genético, el tabu search y el simulated annealing.

1. Introducción

La optimización es el proceso de encontrar la mejor solución (u óptimo) de un conjunto de soluciones [Ilog97]. En forma general el problema se puede formular de la siguiente manera [Pérez Serrada96]:

$$x^0 = \arg\operatorname{opt}_{x \in \chi} f(x)$$

en el que la aplicación $f(\bullet)$ asocia a cada elemento x del conjunto χ un vector de números reales $f(x) \in \Re^{\Re}$. Naturalmente, no es necesario que $f(x)$ tenga una forma cerrada o explícita. Los candidatos a solución x son estructuras (conjuntos, árboles, grafos, etc.) de objetos los cuales se pueden caracterizar inequívocamente a través de m atributos en cierto orden. Al dominio del problema se le denota χ y no necesariamente coincide con *el espacio de búsqueda* del algoritmo (usado para encontrar la solución al problema), que se denota X.

Si estudiamos la complejidad de los problemas de optimización se dice que son *polinómicamente programables* cuando existen algoritmos de resolución que dan una respuesta tras realizar un número de operaciones que es – a lo sumo - una función polinómica del tamaño del problema. Este es un requisito mínimo de complejidad, es decir, si un problema no es polinómicamente programable todos los métodos de resolución serán inútiles para tamaños crecientes del problema ya que ninguna computadora sería capaz de afrontarlos en un tiempo razonable. Generalmente en la práctica la mayoría de los problemas de optimización son *no polinómicamente programables*. Por lo anterior puede ser imposible encontrar el óptimo global en algunas aplicaciones, o no haber una manera de chequear si una solución es un óptimo global. En muchos casos encontrar el óptimo global puede ser innecesario. Encontrar una buena solución (un óptimo local) rápidamente puede ser más deseable que encontrar la mejor solución (el óptimo global) lentamente.

Un problema de optimización clásico es el problema de la mochila que consiste en encontrar un plan de carga adecuado para llenar la misma con el mayor número de *ítems*, tratando de lograr un máximo beneficio. Para ello se dispone de un conjunto de n *ítems*, los cuales tienen un peso y un beneficio asociado, y se debe seleccionar un subconjunto de dichos *ítems* de forma que la suma de los beneficios sea máxima, y la suma total de sus pesos no supere la *capacidad* de la mochila.

---

[1] Grupo de Investigación apoyado por la Universidad Nacional de La Pampa.

[2] Grupo de Investigación apoyado por la Universidad Nacional de San Luis y la ANPCYT (Agencia Nacional para Promover la Ciencias y la Tecnología).

Este problema pertenece a la familia de los problemas NP-Completos, por lo que es de interés analizar su comportamiento mediante las siguientes técnicas de optimización: Algoritmos genéticos, Simulated annealing, Tabu-Search.

A continuación se presenta el modelo del problema:

Dado un conjunto de pesos W[i], beneficios P[i] y capacidad C, encontrar un vector, en nuestro caso será *binario*, x=<x[1],..,x[n]> tal que,

$$\text{Max } P(x)= \Sigma_{i=1..n} x[i]*P[i]$$

s.a

$$\Sigma_{i=1..n} x[i]*W[i] <= C$$

La generación de los pesos y beneficios puede ser aleatoria, pero según [Martello90], la dificultad del problema se ve afectada por la correlación que pueda existir entre los beneficios y pesos. En este caso se tomaron datos no correlacionados, dónde:

W[i]:= uniformemente aleatorio ([1..*v*]).

P[i]:= uniformemente aleatorio ([1..*v*]).

2. Breve Descripción De Las Técnicas De Optimización

2.1. Algoritmos Genéticos

Los Algoritmos Genéticos (AG) son técnicas de búsqueda inspirados en los mecanismos de selección y genética natural; que combinan la capacidad de supervivencia de individuos (posibles soluciones o reglas de inferencia), con operadores genéticos. Mantienen una población que se renueva continuamente. La nueva generación se construye probabilísticamente, a partir de los individuos más aptos de la generación anterior (operador de selección), combinando su "código genético" (a través de operadores de crossover y mutación) [Goldberg 89]. A partir de este esquema, una población puede evolucionar proponiendo nuevos y mejores individuos.

El problema de la mochila presenta una característica interesante y es la existencia de una restricción ($\Sigma_{i=1..n} x[i]*W[i] <= C$). Durante la construcción de los individuos que formarán las distintas poblaciones, se puede dar el caso de que dicha restricción sea violada, por lo que se plantea la necesidad de implementar algún mecanismo, dentro del Algoritmo Genético, que evite o disminuya el impacto que pueden tener los individuos ilegales dentro de la población. Se conoce que para disminuir el efecto de individuos ilegales se puede utilizar *funciones de penalidad*, *algoritmos reparadores* entre otras técnicas.

En este trabajo se utilizó un algoritmo genético con penalidad lineal [Michalewicz96], la cual es definida a continuación:

$$Eval(x)= \Sigma_{i=1..n} x[i]*P[i]-Pen(x)$$

Si el individuo es factible: $Pen(x)=0$

En otro caso: $Pen(x)= \rho*(\Sigma_{i=1..n} x[i]*W[i]-C)$

$\rho=max_{i=1..n}\{P[i]/W[i]\}$

$C=0.5* \Sigma_{i=1..n} W[i]$

debido a que en trabajos previamente desarrollados por el grupo se ha observado un mejor comportamiento al usar penalidad lineal en vez de utilizar penalidad cuadrática o logarítmica [Alfonso 99].

2.2. Simulated Annealing

Simulated Annealing es una técnica similar a la programación genética. Esta basado en observaciones del proceso de enfriado de metales. El término annealing está relacionado con la manera con la que los metales líquidos son enfriados lentamente para asegurar un estado de energía mínima. Se puede decir que el algoritmo de simulated annealing enfría la solución lentamente hasta alcanzar el objetivo posible mas bajo [Ilog 97].

Se chequea si el equilibrio térmico es alcanzado. La temperatura T es reducida gradualmente a través de la función $g(T,t)$, en algunos casos T es reducido cada cierto número de iteraciones t. El algoritmo finaliza con un valor pequeño de T, el criterio de detención controla si el sistema está congelado, por ejemplo no aceptará mas cambios.

2.3. Tabu Search

Este procedimiento ha sido tradicionalmente usado en problemas de optimización combinatorial, el cual guía una heurística de escalamiento descendiente con el objetivo de continuar la exploración evitando un retroceso a un óptimo local desde el cual ya se ha salido. En cada iteración un movimiento admisible se aplica a la solución actual, transformándola de ésta manera en una solución vecina con menor costo. Son permitidas las soluciones que incrementan la función de costo, mientras que el movimiento inverso está prohibido para algunas iteraciones con el objetivo de evitar la ocurrencia de ciclos.

Durante el proceso de búsqueda los movimientos son almacenados en una lista Tabu, representando la memoria de los pasos previos del algoritmo. Se cuenta con un proceso de aspiración para determinar cuando las restricciones Tabu pueden ser sobreescritas.

3. Conclusiones

Los mejores resultados se han observado en la implementación del Simulating Annealing, en primer lugar, y del Algoritmo Genético, en el segundo puesto. La ventaja obtenida por el Simulating Annealing, estimamos, es causada porque esta implementación en las etapas finales se dedica a explotar la mejor solución encontrada. Mientras que el AG, mediante elitismo, mantiene la mejor solución durante la evolución beneficiándola con una mayor fitness, aunque nada asegura que esa mejor solución sea explotada.

A diferencia del Simulating Annealing, que es un procedimiento que mantiene un solo candidato a solución, el AG permite trabajar con múltiples candidatos y considerando que la ventaja obtenida en los resultados finales son levemente superiores a las obtenidas por el AG se puede sugerir el uso de este último en situaciones donde deban contemplarse más de una alternativa para dar una solución satisfactoria al problema.

Con respecto a la implementación del tabu search, se puede concluir que este logra el mayor desaprovechamiento de la capacidad de la mochila luego de analizar los pesos de las soluciones finales.

4. Bibliografía

[Alfonso99] Alfonso H., Montaño Ortiz, A. *KNAPSACK PROBLEM: Análisis Comparativo de Diversas Implementaciones usando Algoritmos Genéticos.* III Seminario Internacional en Tecnologías de Información. 1999.

[Glover89] Glover F., *Tabu search. part I.* ORSA Journal on Computing, 1(3):190-206, 1989.

[Glover90] Glover F., *Tabu search. part II.* ORSA Journal on Computing, 2(1):4-32, 1990.

[Glover93] Glover F., Laguna M., *Tabu search. In Modern Heuristic Techniques for Combinatorial Problems.*, pages 70-150. Blackwell Scientific Publications, Oxford, Colin r. Reeves edition, 1993.

[Goldberg89] Goldberg, D., *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Wesley, Reading, MA, 1989.

[Ilog97] Ilog, Inc., *Optimization Technology White Paper: A comparative sutdy of optimization techniques*, www.ilog.com

[Martello90] Martello, S. and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester, Uk, 1990.

[Michalewicz96] Michalewicz, Z., *Genetic Algorithm + Data Structure = Evolution Programs*, 3ª ed., Springer-Verlag, New York, 1996.

[Michalewicz96] Michalewicz, Z., Dasgupta, D., Le Riche, R., Schonauer, M., *Evolutionary algorithms for constrained engineering problems.*

[Pérez Serrada96] Pérez Serrada, A., *Una Introducción a la Computación Evolutiva.*Marzo 1996.