Informe sobre Estrategias Algorítmicas

Adrián Quiroga Linares, Antonio Vilares Salgueiro

Diciembre 2024

1 Backtracking

Este algoritmo es una técnica general de búsqueda y generación de soluciones que explora todas las posibilidades de manera sistemática, retrocediendo cuando detecta que una solución parcial no puede extenderse a una solución válida. Este informe analiza la implementación de este algoritmo para resolver un problema de asignación, comparando los enfoques con y sin el uso del vector usadas. Para ello, se desarrolla un programa que resuelve el problema de asignar galeones a ciudades de manera óptima, siguiendo las instrucciones proporcionadas en el guión.

1.1 Implementación del Algoritmo

Se han implementado dos versiones del algoritmo de backtracking:

- **Sin vector usadas**: Se recorre la matriz de costes evaluando cada posible asignación sin llevar un registro explícito de las tareas ya asignadas.
- **Con vector usadas**: Se utiliza un vector auxiliar para marcar qué tareas han sido asignadas, lo que permite evitar evaluaciones redundantes.

Ambos enfoques utilizan la matriz de costes proporcionada en clase para generar el árbol de decisiones y encontrar la solución óptima.

1.2 Análisis de Funciones del Algoritmo

1.2.1 Componentes del vector s

El vector s representa la solución parcial en un momento dado del árbol de decisiones, indicando las asignaciones realizadas hasta ese punto.

1.2.2 Función de Asignación

La asignación de atacar una ciudad a un galeón se realiza en la función **Generar()**, que selecciona y evalúa posibles asignaciones basándose en la matriz de costes y las restricciones del problema.

1.2.3 Función MasHermanos

Esta función comprueba si existen más opciones válidas para el nodo actual, es decir, si hay más caminos por explorar en el árbol de decisiones.

1.2.4 Función Retroceder

El objetivo de **Retroceder()** es deshacer la última asignación realizada para regresar al nodo padre y explorar alternativas no visitadas previamente.

1.3 Resultados Obtenidos

El programa genera los siguientes resultados:

- **Mejor beneficio**: 42
  Mejor solución: 2 1 3
- **Mejor beneficio**: 111
  Mejor solución: 4 6 1 3 2 5

n	Vector usadas	Nº nodos visitados	Nº pasos Criterio	Nº pasos Generar	Nº pasos Solución	Nº pasos Más Hermanos	Nº pasos Retroceder
3	No	16	69	30	30	40	10
3	Sí	16	48	30	30	40	10
6	No	1957	37446	7422	7422	8659	1237
6	Sí	1957	11742	7422	7422	8659	1237

Table 1: Tabla de análisis de pasos.

1.4 Conclusiones

1.4.1 Análisis Teórico

La complejidad del algoritmo de backtracking es $O(n!)$ en el peor caso, ya que evalúa todas las permutaciones posibles. Sin embargo, el uso del vector usadas permite reducir el número de pasos necesarios al evitar exploraciones redundantes.

1.4.2 Análisis Práctico

- Con el vector usadas, el número de pasos en la función de criterio se reduce significativamente (por ejemplo, de 37446 a 11742 en el caso de $n = 6$).
- Aunque el tiempo de retroceso y generación es similar en ambos casos, el ahorro en la evaluación de criterios hace que la versión con vector usadas sea más eficiente en problemas grandes.

En resumen, el uso del vector usadas optimiza el algoritmo tanto en términos de pasos realizados como de tiempo de ejecución, especialmente para valores grandes de n.

2 Ramificación y Poda

La técnica de ramificación y poda permite reducir significativamente el número de nodos explorados en comparación con el backtracking tradicional. A continuación, se presentan las conclusiones derivadas del análisis de los resultados obtenidos, así como los cambios necesarios en el caso de minimizar en lugar de maximizar.

2.1 Resultados Obtenidos

El programa genera los siguientes resultados:

- **Mejor beneficio**: 42
  Mejor solución: 2 1 3
- **Mejor beneficio**: 111
  Mejor solución: 4 6 1 3 2 5

N	**Nº de nodos explorados**
3	9
6	1248

Table 2: Tabla de nodos explorados con asignación trivial.

N	**Nº de nodos explorados**
3	4
6	105

Table 3: Tabla de nodos explorados con asignación precisa.

2.2 Relación entre reducción de nodos y complejidad de cálculo:

- Con una **estimación trivial**, se exploran más nodos porque el criterio de poda es menos restrictivo. Esto conduce a menos cortes en el árbol de búsqueda, pero también implica menor complejidad de cálculo por nodo.

- Con una **estimación precisa**, el número de nodos explorados se reduce drásticamente, especialmente para tamaños mayores como $N = 6$. Este enfoque aumenta la precisión de la poda, pero también incrementa la complejidad de cálculo debido al esfuerzo adicional necesario para calcular las cotas.

2.3 Comparación con backtracking:

En comparación con el backtracking, el número de nodos explorados se reduce significativamente en ambas estimaciones. Por ejemplo:

- Para $N = 6$ con asignación precisa, se exploran **105 nodos**, mientras que el backtracking explora **7422 nodos**.

- Esto representa una reducción del **98.54%** en el número de nodos.

La asignación trivial también mejora respecto al backtracking, pero no alcanza la misma eficiencia que la asignación precisa.

2.4 Cambios en caso de minimización:

En un problema de minimización, es necesario realizar los siguientes ajustes en la estrategia de ramificación y poda:

1. **Condición de poda:**

- En lugar de comparar si un nodo tiene un beneficio inferior al beneficio óptimo actual (como se hace en maximización), la poda se realiza si el costo estimado del nodo supera el costo mínimo encontrado hasta el momento.

- Si se cumple que $CI(x) \geq C$, hay que podar.

- Esto implica ajustar la comparación del criterio de poda para mantener una minimización activa.

2. **Cálculo de C:**

- En maximización, C representa una cota superior. En minimización, debe calcularse como una cota inferior (el menor costo posible desde un nodo dado).

- Para obtener los distintos C se calculan mediante $C = min(C, Valor(y))$.

3. **Estrategia de ramificación:**

- En maximización, se priorizan los nodos con mayores beneficios esperados. En minimización, los nodos con menores costos deben ser explorados primero.

- Esto implica modificar las reglas para ordenar los nodos hijos generados cambiando el seleccionar el de mayor BE por el de menor BE.

4. **Valores iniciales:**

- El beneficio inicial se establece como $-\infty$ en maximización, pero en minimización, el costo inicial debe ser $+\infty$ para representar un valor no limitado y que la primera iteración ya cambie.

5. **Estimaciones de las cotas:**

- En maximización, las cotas representan valores altos esperados. Para minimización, debemos modficarlo para calcular los valores más bajos que reflejen el mejor costo posible desde el nodo actual.

2.5 Conclusiones finales:

La implementación de ramificación y poda ofrece una mejora sustancial en la eficiencia del algoritmo tanto en problemas de maximización como de minimización. Sin embargo, los cambios necesarios en la estrategia y los criterios de poda son esenciales para garantizar que el algoritmo siga siendo eficiente y correcto. En particular, la calidad de las estimaciones de las cotas juega un papel crucial en la reducción de la complejidad y el número de nodos explorados.

En resumen, mientras que el backtracking es una técnica general y sencilla de implementar, la ramificación y poda representa una solución más sofisticada y eficiente, especialmente en escenarios donde las cotas se pueden calcular de manera precisa.