Universidad del País Vasco
Euskal Herriko Unibertsitatea

Heurísticos de Búsqueda

Graph Partitioning Problem (GPP) mediante algoritmos de búsqueda local

Josu Ceberio e Inigo Lopez-Gazpio

04/10/2023

Índice

1. Enunciado

Como hemos podido comprobar en la segunda unidad didáctica, las búsquedas locales ofrecen un nuevo paradigma de búsqueda que mejora el rendimiento de manera considerable. A pesar de ello, no están exentas de problemas como puede ser la debilidad de quedarse atascado en los óptimos locales que existen en el espacio de búsqueda. Para contrarrestar este gran problema, en esta tercera unidad didáctica vamos a explorar múltiples maneras de mejorar nuestro algoritmo básico de búsqueda local.

¿Puedes recordar cuales son los tres grandes grupos de soluciones planteadas?
¿Qué ventajas e inconvenientes tiene cada alternativa?

Variable Neighborhood Descent (VND) es uno de los algoritmos que hemos analizado es este aspecto. Este algoritmo trata de explorar vecindades variadas para escapar de óptimos locales y tratar de encontrar nuestro tan deseado óptimo global. En este cuaderno tu tarea va a ser **implementar VND** para el problema conocido como GPP. ¿Te atreves?

1.1. El problema

GPP o Graph Partitioning Problem es un problema de particionado de grafos. En *GPP* se define un grafo con n nodos y una matriz de aristas que reflejan los pesos en las transiciones entre nodos. La matriz de aristas está definida como

$$W = [w_{ij}]_{n \times n}$$

A su vez, GPP define la función objetivo como la tarea de minimizar la suma de los pesos entre los nodos pertenecientes a distintas particiones, y se puede formalizar de una manera más matématica con la siguiente formulación:

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_i \cdot (1 - \sigma_j) \cdot w_{ij}$$

Como puedes observar, en este caso vamos a tratar el problema GPP en su formato binario, es decir, realizando un total de dos particiones balanceadas entre los nodos. De esta manera un nodo debe pertenecer a una partición o a la otra ***(0-1 balanced GPP)***. Para realizar esta práctica tendrás que descargar los ficheros proporcionados que contienen los grafos del problema (matrices de vértices), están disponibles en eGela.

Nota: Reflexiona sobre el espacio de estados y el espacio de codificación de una solución. ¿Son todos los posibles estados del espacio de codificación soluciones válidas para el problema?

1.2. El algoritmo: *Variable Neighborhood Descent*

A continuación implementaremos nuestro nuevo algoritmo, *VND*, un algoritmo muy potente para nuestro arsenal de herramientas. Recuerda que VND tiene la capacidad de analizar distintos vecindarios, incluso vecindarios más lejanos, lo que aporta una capacidad mayor de exploración.

En este ejemplo concreto nuestra implementación de VND va a utilizar dos vecindarios, uno principal, y el otro secundario. El procedimiento será el siguiente: el algoritmo seguirá un comportamiento estándar realizando búsquedas en el vecindario principal, pero en el momento en el que se quede atascado en un óptimo local realizará una nueva búsqueda en el vecindario auxiliar. Si el vecindario secundario ofrece una alternativa de mejora, el algoritmo actualizará el candidato y volverá a continuar la búsqueda en el vecindario principal. Si el vecindario auxiliar tampoco ofrece ninguna opción de mejora, entonces el algoritmo termina la ejecución.

Tu tarea consiste en implementar dos funciones de vecindario: *Hamming* e *Insert*, que corresponderán al vecindario principal y auxiliar. Espera... pero... ¿por qué? ¿no podrían ser principal y auxiliar a la inversa?

Buena pregunta, realmente el vecindario principal debería ser aquel que genere menos óptimos locales. Para ello, tendremos que realizar una estimación para poder responder a esta pregunta con más rigor.

2. Código auxiliar

Enlace código auxiliar (jupyter-notebook)

3. Entrega

Entregar un documento pdf que contenga la respuesta a todas las preguntas planteadas en el *notebook*, además del *notebook* con la implementación del código python que replique la experimentación realizada. Incluir también en el pdf la siguiente tabla completa y una gráfica en la que se compare la convergencia de los algoritmos acorde a la función de fitness (fitness vs steps para cada experimento). Puedes utilizar como guía para la elaboración de tu memoria la rúbrica planteada en la Sección 4.

Algoritmos utilizados	Grid 8X8	G124.16	U500.05	G1000.01
RS (steps) RS (best F)				
BL (best) (AVG steps) BL (best) (AVG best F)				
BL (greedy) (AVG steps) BL (greedy) (AVG best F)				
BL (random) (AVG steps) BL (random) (AVG best F)				
VND (AVG steps) VND (AVG best F)				

*Las ejecuciones de los algoritmos se deben ponderar entre un número razonable de repeticiones, dependiendo del poder de cómputo de la máquina.

4. Rúbrica

La entrega de este laboratorio tiene una calificación del 10 % de la evaluación de la asignatura, sujeta a la superación del test de realización de la práctica.

Se valorará:

- ☐ 1. Hasta un 5 % la calidad y alcance de la experimentación
  - ☐ 1.1. Calidad del código: uso adecuado de funciones, modularidad, organización, etc
  - ☐ 1.2. Alcance y organización de la experimentación
  - ☐ 1.3. Visualización y graficado de resultados
  - ☐ 1.4. Eficiencia de la solución planteada
- ☐ 2. Hasta un 5 % la claridad y razonamiento presente en la memoria
  - ☐ 2.1. Razonamiento resultados obtenidos
  - ☐ 2.2. Comparativa de algoritmos
  - ☐ 2.3. Lecciones aprendidas
  - ☐ 2.4. Calidad presentación memoria