Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Diseño de Algoritmos I

Proyecto II:

Resolviendo el PRPP con Branch and Bound

Michelle Fernández 09-10279
Dayana Rodrigues 10-10615

Febrero 2017

1. Detalles de implementación.

A. Clases definidas.

Para la implementación de la solución, se diseñaron dos clases principales, la primera clase llamada *Edge*, representa cada una de las aristas de un grafo G(V,E) almacenando 5 atributos relacionados a cada $e \in E$, estos datos corresponden a el nodo origen, nodo destino, costo de la arista, beneficio de la arista y un booleano que determina si ha sido cruzada anteriormente.

```
Edge(int n_origen, int n_detino,
                    int cost, int benefit, bool cross = false)
```

La segunda clase llamada *Graph* representa a un grafo G(V,E) de un problema PRPP, cada instancia de *Graph* almacena como atributos de la clase la cantidad de vértices $v \in V$, la cantidad de aristas $e \in E$ y dos matrices de adyacencia, una para todas las aristas $e \in E$ y una matriz llamada T para las aristas $e' \in R \cup Q$.

```
Graph(int vertex, int edges,
       vector<vector<Edge>> &eds, vector<vector<Edge>> &t_eds)
```

Dentro de la clase *Edge* se definió una función *maxBenefit* basada en el algoritmo de Dijkstra y que calcula el camino de máximo beneficio entre dos vértices del grafo.

```
vector<Edge> maxBenefit(int src, int dst)
```

B. Algoritmos utilizados.

Para la implementación de la solución mediante Branch and Bound fue necesario obtener una solución inicial al problema PRPP, para ello se implementó el algoritmo de heurística ávida propuesto en el enunciado del problema.

```
vector<Edge> maxBenefitPath(Graph graph, int deposit)
```

Finalmente se implementó el algoritmo de ramificación y acotamiento sugerido en el enunciado del problema, tomando en cuenta el tiempo de ejecución del mismo para cada una de las instancias. El tiempo mínimo de corte definido para el problema es de 1 hora por cada instancia PRPP a resolver.

2. Resultados experimentales.

A. Tablas de resultados.

Instancia	Valor óptimo	Valor de la heurística	% de desviación de la heurística	Tiempo Branch and Bound
P01	3	3	0	0
D0	109	47	56.88	0
G0	0	0	0	0
R0	1742	-7019	301.2	1

B. Análisis de resultados.

Para la gran mayoría de las instancias requeridas para la solución, el algoritmo greedy implementado no retorna un camino factible, puesto que al llegar al vértice depósito continuaba su ejecución sobre el grafo lo que producía valores negativos.

En algunas de las instancias donde el algoritmo greedy devolvía un camino con beneficio máximo negativo, el algoritmo Branch and Bound encontraba un camino con beneficio positivo que seguía sin ser el óptimo pero mejoraba los resultados.

Para las instancias P01 y G0 el algoritmo greedy retornó el camino óptimo, por lo que el algoritmo Branch and Bound sólo devolvió el mismo como resultado, mientras que para la instancia D0, el algoritmo greedy devolvió un camino con costo -66 que el algoritmo Branch and Bound proceso hasta conseguir un camino de costo 47 que se vio corrompido por las aristas incorrectas de la solución greedy (esta aristas corresponden a los caminos que iban y volvían al depósito continuando la ejecución).

Finalmente para la instancia R0 sucedió lo mismo que en el caso anterior, salvo que Branch and Bound no encontró ninguna solución mejor para esta instancia.

Concluimos que la eficacia de la solución Branch and Bound depende de que la solución greedy se lo suficientemente factible como para no corromper los resultados del Branch and Bound.