Spanning Cover Inequalities for the Capacitated Vehicle Routing Problem

Guilherme G. Arcencio[1], Matheus T. Mattioli[1], Pedro H. D. B. Hokama[2], Mário César San Felice[1]

[1]Departamento de Computação, Universidade Federal de São Carlos, Brazil

[2]Instituto de Matemática e Computação, Universidade Federal de Itajubá, Brazil

`hokama@unifei.edu.br, felice@ufscar.br`

Abstract. *In the k-Minimum Spanning Subgraph problem with d-Degree Center we want to find a minimum cost spanning connected subgraph with $n - 1 + k$ edges and at least degree d in the center vertex, with n being the number of vertices. In this paper we describe an algorithm for this problem and present correctness demonstrations which we believe are simpler than those found in the literature. A solution for the k-Minimum Spanning Subgraph problem with d-Degree can be used to formulate spanning cover inequalities for the capacitated vehicle routing problem.*

1. Introduction

The k-Minimum Spanning Subgraph problem (k-MSS) is a generalization of the Minimum Spanning Tree problem (MST) whose feasible solutions correspond to a spanning connected subgraph with $n - 1 + k$ edges, where n is the number of vertices. Moreover, we consider the k-MSS with d-Degree Center (k-MSS with d-DC), a generalization of the k-MSS whose feasible solutions have at least degree d in the center vertex, i.e., $|\delta(0)| \geq d$, with $\delta(v)$ being the set of edges incident to vertex v and 0 being the center. Our interest in these problems derive from the study of the Vehicle Routing Problem (VRP), in which we want to find routes connecting customers with the depot in a way to minimize operational costs, such as the total traveled distance or the size of the fleet. In the Capacitated Vehicle Routing Problem (CVRP), we have the additional constraint that each route cannot surpass the capacity limit of the vehicle.

Cover inequalities [Balas 1975, Wolsey 1975] are added to an Integer Linear Program (ILP) to strengthen its linear programming relaxation. Traditionally, the inequality left hand side (LHS) is the sum of several decision variables whose simultaneous occurrence violates a constraint, while its right hand side (RHS) is the number of variables minus one. A spanning cover inequality slightly generalizes those because its LHS is the sum of decision variables whose simultaneous occurrence does not violate the explicit constraints of the problem. However, no feasible or good integer solution which includes the LHS variables can be produced. Using the CVRP as example, a set of edges corresponds to a spanning cover inequality if any collection of circuits using those edges has cost greater than the best primal bound. It is NP-hard to solve the CVRP with some edges fixed. However, the solution to the k-MSS with d-DC with some edges fixed can be used to identify sets of edges which form spanning cover inequalities to the CVRP, since the optimal solutions to this problem is a lower bound to the CVRP.

While searching for Lagrangean relaxations for the CVRP, Christofides, Mingozzi and Toth [Christofides et al. 1981] solved the k-Degree Center Tree Problem (k-DCT), which corresponds to the 0-MSS with k-DC. Improving over that, Fisher [Fisher 1994] solved the k-MSS with 2k-DC, which he called k-Tree.

Our contributions. We propose an algorithm to solve the k-MSS with d-DC and prove its correctness. We note that, while our algorithm has similarities with those from Christofides et al. and Fisher, we believe its correctness proof is simpler and some differences in the algorithm together with an auxiliary result allow a more efficient implementation.

2. Lower bounds and algorithm

In a CVRP instance where each vertex $v \in V$ has a certain demand d_v and each vehicle has capacity Q, every feasible solution will use at least k vehicles, where $k = \left\lceil \frac{\sum_{v \in V} d_v}{Q} \right\rceil$.

Hence, any solution for CVRP is a collection of k or more circuits. Thus, it is a spanning subgraph with at least $n-1+k$ edges and degree at least $2k$ in the depot. Therefore a solution for the k-MSS with 2k-DC is a lower bound for the CVRP optimum. Next, we present our algorithm for the k-MSS with d-DC.

2.1. Algorithm for the k-MSS with d-DC

Our algorithm generates an MST, then greedily exchanges edges, increasing the depot's degree to the required value, and the resulting tree is then expanded to a k-spanning subgraph. Algorithm 1 shows the pseudocode of our algorithm.

Definition 2.1. *Consider a spanning tree T of $G = (V, E)$, with $c\colon E \to \mathbb{R}_+^*$. Let $0 \in V$ be the center and take an edge $e \in \delta(0) \setminus T$. The set $T \cup \{e\}$ contains exactly one cycle C. Letting e' denote the most expensive edge in $C \setminus \delta(0)$, we define $\Delta(T, e) = c(e) - c(e')$.*

Algorithm 1 K-MSS WITH D-DC(G, c, k, d)

```
Input: graph $G = (V, E)$, cost function $c\colon E \to \mathbb{R}_+^*$, number of extra edges $k$ and degree $d$ of center
Output: a minimum cost spanning subgraph with $n - 1 + k$ edges and center with degree at least $d$
 1: $T \leftarrow$ MST of $G$
 2: while $|\delta_T(0)| < d$ do
 3:     $z \leftarrow e \in \delta(0) \setminus T$ such that $\Delta(T, e)$ is minimal
 4:     $C \leftarrow$ the only cycle in $T \cup \{z\}$
 5:     $z' \leftarrow$ the most expensive edge in $C \setminus \delta(0)$
 6:     $T \leftarrow (T \cup \{z\}) \setminus \{z'\}$
 7: rename the edges in $E$ such that $c(e_i) \leq c(e_{i+1})$ for $i = 1, \cdots, |E| - 1$
 8: $r \leftarrow 0$
 9: for $i = 1$ to $|E|$ do
10:     if $r < k$ and $e_i \notin T$ then
11:         $T \leftarrow T \cup \{e_i\}$
12:         $r \leftarrow r + 1$
13: return T
```

Definition 2.2. *Letting $G = (V, E)$ and $c\colon E \to \mathbb{R}_+^*$, MST$_d$ is a spanning tree of G among those with center degree at least d which has minimum cost.*

We want to prove that, by the end of the algorithm's while loop, T is an MST$_d$.

Theorem 2.3. *Let $G = (V, E), c\colon E \to \mathbb{R}_+^*$, k and d in $\mathbb{Z}_+^*$ be an instance of the k-MSS with d-DC. A subgraph T built by Algorithm 1's first loop is an MST$_d$.*

Proof. In this proof we call T_i the tree T in the beginning of the i-th iteration of the algorithm's while loop. Let l denote $|\delta_{T_1}(0)|$. If $l \geq d$, we are done. Otherwise, suppose that T_i is an MST$_{l+i-1}$. Note that this is true for $i = 1$. We show that T_{i+1} is an MST$_{l+i}$. We have

$$c(T_{i+1}) = c(T_i) + \min_{e \in \delta(0) \setminus T_i} \{\Delta(T_i, e)\} \ . \tag{1}$$

Suppose, by contradiction, that T_{i+1} is not an MST$_{l+i}$. Then, there must exist an MST$_{l+i}$ called T^* such that $c(T^*) < c(T_{i+1})$. Since $|\delta_{T^*}(0)| \geq l+i$ and $|\delta_{T_i}(0)| = l+i-1$, by the pigeonhole principle T^* has at least one edge $z \in \delta(0) \setminus T_i$. Thus, by (1) we have $c(T_{i+1}) \leq c(T_i) + \Delta(T_i, z)$.

Let C_z be the cycle generated in T_i by inserting z. Note that $T^* \setminus \{z\}$ has 2 components and that C_z intersects both. Moreover, let $z' \in C_z$ be an edge, other than z, which reconnects these 2 components in T^*. Let $S = (T^* \cup \{z'\}) \setminus \{z\}$. Note that S is a spanning tree with $|\delta_S(0)| \geq l + i - 1$. We have $\Delta(T_i, z) = \min_{e \in C_z}\{c(z) - c(e)\} \leq c(z) - c(z')$. Thus

$$\begin{aligned} c(S) &= c(T^*) - c(z) + c(z') \\ &\leq c(T^*) - \Delta(T_i, z) \\ &< c(T_{i+1}) - \Delta(T_i, z) \\ &\leq c(T_i) + \Delta(T_i, z) - \Delta(T_i, z) = c(T_i) \ , \end{aligned}$$

which contradicts the hypothesis that T_i was an MST$_{l+i-1}$. Therefore, by induction, T will be an MST$_d$ by the end of $d - l$ iterations. □

Due to Theorem 2.3, we have that the first loop of Algorithm 1 produces an optimal solution to the 0-MSS with d-DC. The second loop of Algorithm 1 adds k least cost edges, producing a k-MSS with d-DC, as stated by the following lemma. Due to space constraints, its proof was omitted.

Lemma 2.4. *Let $G = (V, E), c\colon E \to \mathbb{R}_+^*$, k and d in $\mathbb{Z}_+^*$ be an instance of the k-MSS with d-DC. A subgraph T built by Algorithm 1's second loop is an optimal solution for k-MSS with d-DC.*

3. Implementation details and efficiency

Consider an instance of the k-MSS with d-DC, whose graph $G = (V, E)$, with costs c in the edges, has n vertices and m edges. Our algorithm takes time $O(m \log n)$ to build the MST at Line 1 and to add the extra edges in its second loop.

Now, focusing on the first loop of the algorithm, we can compute every $\Delta(\cdot)$ in the first iteration using a modified Depth First Search (DFS) beginning at the depot. This DFS must travel only through edges of tree T, and it keeps track of the most expensive edge e', not incident to the depot, in the current path. Moreover, every time the DFS finds an edge e outside of T which is incident to the depot, it computes $\Delta(e) = c(e) - c(e')$.

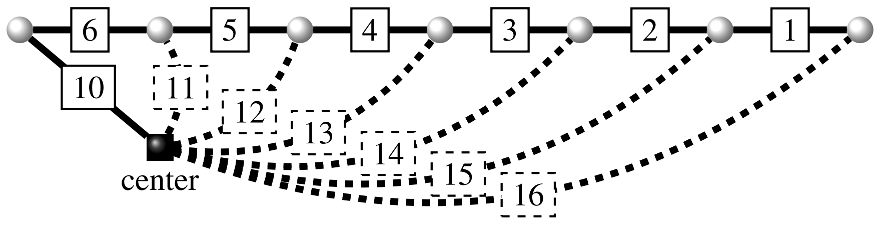

Figure 1. Worst case for the number of $\Delta(T, e)$ updates

Computing every $\Delta(\cdot)$ using this DFS approach costs at most $O(m)$ time, since the DFS still needs to consider every edge, despite only traveling through edges of T. In each iteration of the first loop an edge is added to the depot, therefore we have at most $d - 1$ iterations, once T is connected. Thus, a straightforward implementation which recomputes every $\Delta(\cdot)$ in each iteration takes time $O(dm)$. However, this might not be necessary due to the following property, whose proof we omitted due to space constraints.

Property 3.1. *In each iteration of Algorithm 1's first loop we need to update a $\Delta(T, e)$ only if the most expensive edge e' in C_e was removed from T and e was not added to T in the last iteration.*

Thus, we can keep a table indexed by the most expensive edge in each circuit and, once an edge e' is removed we can check which circuits were affected and update accordingly. While this observation does not improve the worst case of the algorithm, it can have a significant empirical impact, since the removal of an edge seems to only affect the $\Delta(\cdot)$ of others in pathological cases, as the one shown in Figure 1, where the edge removed in each iteration affects the $\Delta(\cdot)$ of all other vertices.

4. Future works

We want to use this algorithm to find spanning cover inequalities for the CVRP and evaluate its impact empirically. Moreover, we would like to improve the efficiency of this algorithm and to generalize this lower bound to variants of the CVRP.

Acknowledgments

The authors gratefully acknowledge the financial support of grants FAPESP #2020/06103-0, FAPESP #2020/06105-2, and CNPq process 435617/2018-4.

References

Balas, E. (1975). Facets of the knapsack polytope. *Mathematical programming*, 8(1):146–164.

Christofides, N., Mingozzi, A., and Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1):255–282.

Fisher, M. L. (1994). Optimal solution of vehicle routing problems using minimum k-trees. *Operations research*, 42(4):626–642.

Wolsey, L. A. (1975). Faces for a linear inequality in 0–1 variables. *Mathematical Programming*, 8(1):165–178.