Up- and downgrading the 1-center in a network

Elisabeth Gassner *

July 17, 2007

Abstract

We study budget constrained network improvement and degrading problems based on the vertex 1-center problem on graphs: Given a graph with vertex weights and edge lengths the task is to decrease and increase the vertex weights within certain limits such that the optimal 1-center objective value with respect to the new weights is minimized and maximized, respectively. The upgrading (improvement) problem is shown to be solvable in $\mathcal{O}(n^2)$ time provided that the distance matrix is given. The downgrading 1-center problem is shown to be strongly $\mathcal{NP}$-hard on general graphs but can be solved in $\mathcal{O}(n^2)$ time on trees. As byproduct we suggest an algorithm that solves the problem of minimizing over the upper envelope of piecewise linear and monotone functions in $\mathcal{O}(K)$ time where K is the total number of breakpoints.

Keywords: combinatorial optimization, location problem, vertex center, upgrading, downgrading, network improvement, network modification, complexity, tree.

1 Introduction

Location problems are - due to their practical relevance - among the best investigated problems in Operations Research. There are two classical models, the median problem (minimize the sum of weighted distances to the customers) and the center problem (minimize the maximum among the weighted distances to the customers). Both problems have received much attention during the last decades (e. g., see Kariv and Hakimi [15], Mirchandani and Francis [18] and Drezner and Hamacher [8]). However, less attention was paid to network modification problems where the goal is to change the parameters (vertex weights and/or edge lengths) in order to improve or downgrade the network in respect of the optimal location of a facility.

In this paper, we are interested in a variant of the 1-center problem on graphs where we are allowed to invest a given budget in order to increase or decrease the vertex weights. We distinguish two problems, the improvement (or upgrading) problem and the downgrading (or degrading) problem: In the improvement problem we are allowed to decrease the vertex weights within certain limits such that the optimal 1-center objective value after the weight modification is minimized. This means, we improve the network before deciding where the center should be located. In the downgrading problem we have to increase the vertex weights,

*gassner@opt.math.tu-graz.ac.at. Department of Optimization and Discrete Mathematics, Graz University of Technology, Steyrergasse 30, A-8010 Graz, Austria.
This research is partially supported by the Austrian Science Fund Project P18918-N18 *Efficiently solvable variants of location problems.*

i. e., there is an adversary who increases the vertex weights within certain limits such that the optimal 1-center objective value after the weight modification is maximized.

Both problems can be seen as bilevel problems with an actor who changes the vertex weights and a reactor who locates the center. While in the improvement model both decision makers, the actor and reactor, have the same goal, i. e., to minimize the maximum weighted distance, in the downgrading model the goals of the decision makers are conflicting.

1.1 Related Problems

Network up- and downgrading versions have been applied to several classical combinatorial optimization problems. Fulkerson and Harding [10] and Hambrusch and Tu [12] investigated how to maximally reduce the length of a shortest and longest path, respectively, by edge length modifications. Phillips [19] introduced a flow interdiction problem, i. e., the problem of attacking edges to reduce the maximum flow value. Frederickson and Solis-Oba [9], Drangmeister et al. [7] and Krumke et al.[16] investigated network improvement and degrading versions for the minimum spanning tree and Steiner tree problem. A general framework for up- and downgrading versions of 0/1-combinatorial optimization problems was investigated by Burkard, Klinz and Zhang [3] and Burkard, Lin and Zhang [4].

Network modification problems are closely related to reverse problems where the task is to maximally improve a prespecified solution within certain limits. Observe that for reverse problems a feasible solution is given while this is not the case for up- and downgrading problems. Due to the large number of publications in this area, we restrict ourselves to location problems. Burkard, Gassner, and Hatzl [1] proved $\mathcal{NP}$-hardness of the reverse 1-median problem with edge length modification on general graphs and suggested a linear time algorithm for the problem on a cycle. In [2] the same authors presented an $\mathcal{O}(n \log n)$ time algorithm for the reverse 2-median problem on trees as well as the reverse 1-median problem on unicyclic graphs. Zhang, Yang and Cai [20, 21] investigated the problem of how to shorten the lengths of edges at minimum cost (measured in different norms) such that the shortest distances between the vertices are within given bounds. For the case of the ℓ_∞-norm a strongly polynomial algorithm is developed while for ℓ_1- and ℓ_2-norm the problem is even hard to approximate.

Finally, the class of inverse problems should be mentioned. The goal of an inverse location problem is to modify parameters at minimum cost such that a given feasible solution becomes optimal. Several papers deal with inverse location problems like the inverse p-median problem on a graph and the inverse 1-median problem in the plane with ℓ_1- or ℓ_∞-norm and vertex weight modification (Burkard, Pleschiutschnig and Zhang [5]). Gassner [11] proved $\mathcal{NP}$-hardness of the inverse 1-maxian problem with variable edge lengths (even on series-parallel graphs) and suggested an $\mathcal{O}(n \log n)$ time algorithm for this problem on a tree. And Cai, Yang and Zhang [6] proved that the inverse center problem with edge length modification is $\mathcal{NP}$-hard. The reader is referred to the comprehensive survey on inverse combinatorial optimization by Heuberger [14].

1.2 Organization and contribution of this paper

In this paper, we investigate up- and downgrading versions of the 1-center problem.

The (vertex) 1-center problem is given by a graph $G = (V, E)$ with positive edge lengths $\ell_e \in \mathbb{R}_+$ for $e \in E$ and positive vertex weights $w_v \in \mathbb{R}_+$ for $v \in V$. The task is to find a vertex

$x \in V$ which minimizes

$$f(x) = \max\{w_v d(v,x) \mid v \in V\}$$

where $d(i,j)$ denotes the shortest distance between the vertices i and j with respect to edge length ℓ. The optimal objective value with respect to vertex weights w is denoted by

$$z(w) = \min_{x \in V} \max\{w_v d(v,x) \mid v \in V\}.$$

In Section 2, we give a short introduction to the 1-center problem and related problems.

The up- or downgrading 1-center problem is given by an instance of the 1-center problem and in addition we are given cost coefficients $c_v \in \mathbb{R}_+$, bounds $u_v \in \mathbb{R}_v$ for the vertices $v \in V$ and a total budget B.

A vertex modification $\delta = (\delta_v)_{v \in V}$ of the up- or downgrading problem is called feasible if $\delta \in \Delta$ with

$$\Delta = \left\{ \delta \in \mathbb{R}^{|V|} \;\middle|\; \sum_{v \in V} c_v \delta_v \leq B \text{ and } 0 \leq \delta_v \leq u_v \forall v \in V \right\}.$$

The upgrading 1-center problem, Up1Center for short, is to solve

$$\min_{\delta \in \Delta} z(w - \delta) = \min_{\delta \in \Delta} \min_{x \in V} \max\{(w_v - \delta_v) d(v,x) \mid v \in V\}.$$

The downgrading 1-center problem, Down1Center for short, is to solve

$$\max_{\delta \in \Delta} z(w + \delta) = \max_{\delta \in \Delta} \min_{x \in V} \max\{(w_v + \delta_v) d(v,x) \mid v \in V\}.$$

In Section 3, we deal with Up1Center and suggest an $\mathcal{O}(n^2)$ time algorithm for this problem on general graphs provided that the distance matrix is given. Section 4 is dedicated to Down1Center. We show that the problem is in general $\mathcal{NP}$-hard (Subsection 4.1) and suggest an efficient algorithm for trees that runs in $\mathcal{O}(n^2)$ time (Subsection 4.2).

1.3 Notation

Throughout this paper, we will use the following notation: Let $G = (V, E)$ be a graph. Then $n = |V|$ is the number of vertices and $m = |E|$ is the number of edges.

Let $T = (V, E)$ be a tree and $(i, j) \in E$. Then $T_i(i, j)$ denotes the subtree that contains vertex i after deleting edge (i, j). If no confusion is possible, we write $v \in T_i(i, j)$ instead of $v \in V(T_i(i, j))$.

A function $f(x) : X \to Y$ is called unimodal if there exists a value y such that either f is monotonically increasing for $x \leq y$ and monotonically decreasing for $x \geq y$ or f is monotonically decreasing for $x \leq y$ and monotonically increasing for $x \geq y$.

2 A short survey of the 1-center problem

The 1-center problem is a special case of the p-center problem where the task is to find a set $S \subset V$ of at most p elements which minimizes

$$\max_{v \in V} \min_{x \in S} w_v d(v,x).$$

Observe that even the unweighted p-center problem (where $w_v = 1$ for all $v \in V$) is strongly $\mathcal{NP}$-hard since it can easily be transformed to a dominating set problem.

Based on the property that the 1-center objective function is convex on every path, the 1-center problem on trees can be solved in linear time (Megiddo [17]). On general graphs the 1-center problem can be solved by determining the 1-center objective value for every vertex and then choosing the best one. Observe that the bottleneck of this procedure is the computation of the distance matrix (e. g., using Floyd-Warshall's algorithm which runs in $\mathcal{O}(n^3)$ time).

More attention was paid to so-called absolute center problems where the center may be located anywhere on the graph, i. e., on vertices or in the interior of edges. An absolute 1-center can be found in $\mathcal{O}(mn \log n)$ time provided that the distance matrix is available (Kariv and Hakimi [15]). The following lemma is crucial for solving the 1-center in a tree:

Lemma 2.1 (Kariv and Hakimi [15])**.** *Let $v \in V$ and let $s, t \in V$ such that s and t lie in different subtrees $T_a(a, v)$ and $T_b(b, v)$ $(a \neq b)$ and let $w(s)d(s, v) = w(t)d(t, v) = \max_{v' \in V} w(v')d(v', v)$. Then v is the 1-center.*

Based on this lemma Kariv and Hakimi developed an $\mathcal{O}(n \log n)$ time algorithm which was improved to a linear time algorithm by Megiddo [17].

Next we discuss some properties of the 1-center in a graph that will be of interest for the investigations of up- and downgrading versions of center problems. We will mainly focus on necessary and sufficient conditions for the optimal objective value to be at most and at least a threshold L, respectively.

Let us start with an upper bound on the optimal objective value.

Lemma 2.2. *Let (G, ℓ, w) be an instance of the 1-center problem and let $L \in \mathbb{R}_+$. Then*

$$\min_{x \in V} f(x) \leq L$$

if and only if there exists a vertex $x \in V$ such that $w_v d(v, x) \leq L$ for all $v \in V$.

The above lemma is trivial. However, we mention it in order to point out the contrast to the lower bound.

Lemma 2.3. *Let (G, ℓ, w) be an instance of the 1-center problem and let $L \in \mathbb{R}_+$. Then*

$$\min_{x \in V} f(x) \geq L$$

if and only if for every $v \in V$ there exists a vertex $q(v) \in V$ such that

$$w_{q(v)} d(v, q(v)) \geq L.$$

Proof. There exists a solution $x \in V$ with $\min_{x \in V} f(x) \geq L$ if and only if $f(x) \geq L$ for all $x \in V$ and hence $\max_{v \in V} w_v d(v, x) \geq L$ for every $x \in V$. And this is true if and only if for every $x \in V$ there exists a vertex $q(x) \in V$ with $w_{q(x)} d(q(x), x) \geq L$. □ □

Lemma 2.3 implies a kind of anti-domination property of the 1-center problem. There exists a feasible solution with objective value at least L if and only if for every vertex $v \in V$ there exists another vertex $q(v) \in V$ that is far away. If we ask for a set S of vertices such that for every $v \in V$ there exists a vertex $q(v) \in S$ such that $q(v)$ is near to v then we get a generalized domination problem. Due to this relationship, a vertex $v \in V$ is called

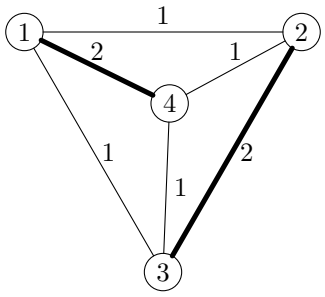

Figure 1: An instance of the 1-center problem where every vertex is a anti-domination vertex for $L = 2$.

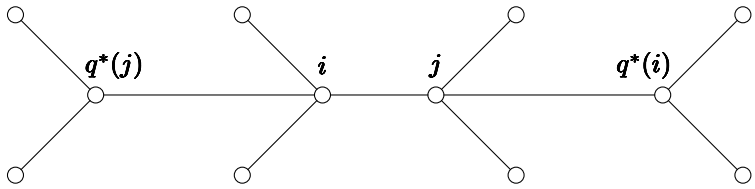

Figure 2: An illustration for the proof of Lemma 2.4.

anti-dominated by $q(v) \in V$ if $w_{q(v)}d(q(v), v) \geq L$ holds and $q(v)$ is called anti-domination vertex.

Observe that there may exist a threshold L such that every vertex is a anti-domination vertex. This is illustrated in the following example:

Consider the graph of Figure 1. The edge lengths are given in the figure, all vertex weights are equal to 1 and $L = 2$. Observe that $w_2d(2, 1) = 1 = w_3d(3, 1)$ and $w_4d(4, 1) = 2 \geq L$. Therefore, vertex 1 is uniquely anti-dominated by vertex 4. By similar arguments, we see that vertex 2 is only anti-dominated by vertex 3, vertex 3 is only anti-dominated by vertex 2 and finally vertex 4 is only anti-dominated by vertex 1. Hence, the set of anti-domination vertices is unique and contains all vertices.

However, if the underlying graph is a tree then there exists a set of two vertices that anti-dominate the rest of vertices:

Lemma 2.4. *Let (G, ℓ, w) be an instance of the 1-center problem where $G = (V, E)$ is a tree and let $L \in \mathbb{R}_+$. Then*

$$\min_{x \in V} f(x) \geq L$$

if and only if there exist two vertices $s, t \in V$ such that

$$\max\{w_s d(i, s), w_t d(i, t)\} \geq L \qquad \text{for every } i \in V.$$

Proof. Clearly, if there are two vertices $s, t \in V$ that anti-dominate all vertices, then every vertex is anti-dominated and hence Lemma 2.3 implies that the optimal objective value is at least L.

Hence, we assume that $\min_{x \in V} f(x) \geq L$, i. e., every vertex v is anti-dominated by another vertex $q(v) \in V$. Let $k \in V$ and let $q^*(k)$ be a vertex of maximum weighted distance to k, i. e.,

$$w_{q^*(k)}d(k, q^*(k)) = \max\{w_v d(k, v) \mid v \in V\}.$$

Since $k \in V$ is anti-dominated by assumption, we conclude that k is also anti-dominated by $q^*(k)$, i.e.,

$$w_{q^*(k)} d(k, q^*(k)) \geq L$$

for every $k \in V$. Now let

$$L' := w_{q^*(i)} d(i, q^*(i)) = \min\{w_{q^*(k)} d(k, q^*(k)) \mid k \in V\}.$$

Then

$$L \leq L' = w_{q^*(i)} d(i, q^*(i)) \leq w_{q^*(k)} d(k, q^*(k))$$

holds for all vertices $k \in V$.

Consider the path from i to $q^*(i)$ and let vertex j be adjacent to i on this path. We will prove that $q^*(i)$ and $q^*(j)$ anti-dominate all vertices. This means, we have to show that

$$\text{either } w_{q^*(i)} d(k, q^*(i)) \geq L \text{ or } w_{q^*(j)} d(k, q^*(j)) \geq L$$

for every $k \in V$.

We partition the set of vertices into two subsets. Let $T_i(i,j) = (V_i, E_i)$ and $T_j(i,j) = (V_j, E_j)$. Then every vertex is either in V_i or in V_j.

Let $k \in V_i$, then by definition $q^*(i) \in V_j$ and therefore $d(k, q^*(i)) = d(k,i) + d(i, q^*(i))$. Hence, we get

$$w_{q^*(i)} d(k, q^*(i)) = w_{q^*(i)} \left(d(k,i) + d(i, q^*(i))\right) \geq w_{q^*(i)} d(i, q^*(i)) = L' \geq L.$$

It remains to show that $q^*(j) \in V_i$. Assume that $q^*(j) \in V_j$ then

$$\begin{aligned} L' = w_{q^*(i)} d(i, q^*(i)) &\geq w_{q^*(j)} d(i, q^*(j)) \\ &= w_{q^*(j)} \left(d(i,j) + d(j, q^*(j))\right) \\ &> w_{q^*(j)} d(j, q^*(j)). \end{aligned}$$

Observe that the first inequality holds because of the maximality property of q^*. The above statement contradicts the minimality of vertex i and proves that $q^*(j) \in V_i$. Let $k \in V_j$ then we have $d(k, q^*(j)) = d(k,j) + d(j, q^*(j))$ and

$$w_{q^*(j)} d(k, q^*(j)) = w_{q^*(j)} \left(d(k,j) + d(j, q^*(j))\right) \geq w_{q^*(j)} d(j, q^*(j)) \geq L.$$

Hence, every vertex is either anti-dominated by $s = q^*(j)$ or by $t = q^*(i)$ which proves the lemma. □ □

The previous lemma as well as the additional properties used in its proof immediately imply that $\min_{x \in V} f(x) \geq L$ if and only if there exists an edge $(i,j) \in E$ and vertices $s \in T_i(i,j)$, $t \in T_j(i,j)$ such that

$$w_s d(s,j) \geq L \quad \text{and} \quad w_t d(i,t) \geq L.$$

Therefore, the following theorem follows:

Theorem 2.5. *Let (G, ℓ, w) be an instance of the 1-center problem where $G = (V, E)$ is a tree. Then the optimal objective value is equal to*

$$\max_{\substack{(i,j) \in E \\ s \in T_i(i,j) \\ t \in T_j(i,j)}} \min\{w_s d(s,j), w_t d(i,t)\}.$$

Observe that Theorem 2.5 provides a dual formulation of the vertex 1-center problem (which is a minimization problem). For the absolute 1-center problem a dual formulation was already known before. It is of the form

$$\max_{i,j \in V} \frac{w_i w_j}{w_i + w_j} d(i,j).$$

It should be mentioned that Theorem 2.5 and Lemma 2.1 are closely related. Nevertheless, the vertices s and t of Theorem 2.5 and Lemma 2.1 are not the same and hence our theorem is not a direct consequence of the lemma. Consider a path of four vertices $T = (V, E)$ with $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (2, 3), (3, 4)\}$ and let $\ell(1, 2) = \ell(3, 4) = 3$ and $\ell(2, 2) = 4$. Moreover, $w_1 = w_4 = 1$ and $w_2 = w_3 = 2$. Then the midpoint x of edge $(2, 3)$ is the absolute 1-center and the vertices 1 and 4 have maximum weighted distance to x ($s = 1$, $t = 4$ for the criterion of Lemma 2.1). On the other hand, the vertices 2 and 3 are vertex 1-centers and vertex 3 has maximum weighted distance to 2 and vertex 2 has maximum weighted distance to 3 ($s = 2$, $t = 3$ for the criterion of Theorem 2.5).

Finally, it should be mentioned that for the special case of trees a lower bound on the optimal vertex 1-center objective value can be verified by fixing a vertex and checking the weighted distances to it while an upper bound is verified by fixing an edge and checking the weighted distances to its endpoints.

3 Upgrading the 1-center by vertex weight modification

This section is dedicated to the network improvement problem Up1Center. The task of Up1Center is to modify the vertex weights within certain limits in such a way that the optimal 1-center objective value with respect to the new vertex weights is minimized. Recall that Up1Center is defined as follows:

$$\min_{\delta \in \Delta} \min_{x \in V} \max_{v \in V} \{(w_v - \delta_v) d(x, v)\}.$$

Clearly, we can reverse the first two minimization tasks and get an equivalent problem

$$\min_{x \in V} \min_{\delta \in \Delta} \max_{v \in V} \{(w_v - \delta_v) d(x, v)\}.$$

Let $x \in V$ be a fixed vertex, then

$$\text{P(x)} \qquad \min_{\delta \in \Delta} \max_{v \in V} \{(w_v - \delta_v) d(x, v)\}$$

is the problem of maximally improving the potential center location $x \in V$. Problem P(x) is a reverse center problem where the goal is to maximally improve a given location within certain limits. Since for upgrading problems the locations are not fixed a priori, we have to try out all locations, maximally improve them and take the best choice, i. e., Up1Center is equivalent to

$$\min_{x \in V} h(x)$$

where $h(x)$ is the optimal objective value of P(x). Observe that the optimal 1-center before changing the vertex weights and the optimal location after an optimal weight modification is

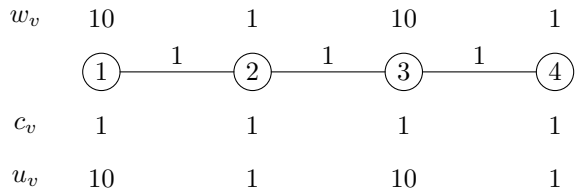

Figure 3: Instance of Up1Center where $h(x)$ is not unimodal along a path.

in general not the same. This is the reason why it is not sufficient to solve only $P(x^*)$ where x^* is a 1-center with respect to the original weights w. Moreover, $h(x)$ is not unimodal along a path. This is shown by the following example: Consider the instance given in Figure 3 with $B = 8$.

The optimal solutions of P(x) for all vertices $x \in V$ are given in the following table.

	optimal solution	optimal objective value
P(1)	$\delta_3 = 8$	4
P(2)	$\delta_1 = \delta_3 = 4$	6
P(3)	$\delta_1 = 8$	4
P(4)	$\delta_1 = 7$, $\delta_3 = 1$	9

It can be seen that vertex 2 is the unique 1-center before the weight modification while either vertex 1 or vertex 3 are 1-centers after an optimal weight modification. Moreover, the optimal objective value is not unimodal along a path.

Therefore, we determine $h(x)$ for every $x \in V$. Let us start with the following characterization of an optimal solution of P(x).

Lemma 3.1. *Let δ^* be an optimal solution of P(x) and let $h(x) = L^*$. Then $(w_v - \delta_v^*)d(v,x) = L^*$ if $w_v d(v,x) \geq L^*$ and $\delta_v^* = 0$ otherwise.*

Hence, the problem can be solved by sorting the vertices according to $w_v d(v,x)$, i. e., $w_{v_1} d(v_1,x) \geq w_{v_2} d(v_2,x) \geq \cdots \geq w_{v_n} d(v_n,x)$. Then decrease the weight of vertex v_1 until either the upper bound u_{v_1} is reached, the whole budget is used or $(w_{v_1} - \delta_{v_1})d(v_1,x) = w_{v_2} d(v_2,x)$. If one of the first two stop criteria occurs then an optimal solution is found. Otherwise decrease simultaneously the weights of v_1 and v_2 until one of the upper bounds is reached, the whole budget is used or $(w_{v_1} - \delta_{v_1})d(v_1,x) = w_{v_3} d(v_3,x)$ and $(w_{v_2} - \delta_{v_2})d(v_2,x) = w_{v_3} d(v_3,x)$, and so on. Clearly, this algorithm solves problem P(x) in $\mathcal{O}(n^2)$ time because there are at most n iterations and in iteration i we have to change i weights.

However, we can solve the problem even in linear time: For every $L \in \mathbb{R}_+$ we define the following solution:

$$\delta_v(L) = \begin{cases} w_v - \frac{L}{d(v,x)} & \text{if } w_v d(v,x) \geq L \\ 0 & \text{else} \end{cases}.$$

According to Lemma 3.1, $\delta(h(x))$ is an optimal solution of problem P(x). Therefore P(x) is equivalent to the problem of finding the smallest value L such that $\delta(L)$ is a feasible solution, i. e., satisfies the upper bounds and budget constraint. Observe that if $\delta(L)$ is feasible then $\delta(L')$ is feasible for all $L' \geq L$.

Assume that we know a lower and upper bound on $h(x)$, i. e., $h(x) \in [a, b]$. Moreover, let

$$I' = \{v \in V \mid w_v d(v, x) \geq b\}$$
$$\alpha' = \sum_{v \in I'} c_v w_v$$
$$\beta' = \sum_{v \in I'} \frac{c_v}{d(v, x)}.$$

We assume that

$$\max_{v \in I'} (w_v - u_v) d(v, x) \leq a \tag{1}$$
$$\frac{\alpha' - B}{\beta'} \leq a \tag{2}$$

holds. Observe that we know that the weights of all vertices in I' have to be changed in an optimal solution. Therefore, the upper bounds for these vertices have to be satisfied and the budget we need for these vertices must not exceed B.

Inequality (1) and $h(x) \geq a$ ensure that

$$\delta_v(h(x)) = w_v - \frac{h(x)}{d(v, x)} \leq u_v$$

holds for all $v \in I'$.

Now we investigate the budget we need for all vertices in I'. Let $L \in [a, b]$ then

$$B'(L) = \sum_{v \in I'} c_v \delta_v(L) = \sum_{v \in I'} c_v w_v - L \sum_{v \in I'} \frac{c_v}{d(v, x)} = \alpha' - L\beta'$$

is equal to the investment for vertices in I'. Due to inequality (2), we conclude that $B'(L) = \alpha' - L\beta' \leq \alpha' - a\beta' \leq B$ holds for all $a \leq L \leq b$.

The goal is to decide whether $L \leq h(x)$ or $L \geq h(x)$. If $\delta(L)$ is feasible then $h(x) \leq L$ otherwise $h(x) \geq L$. Hence, the goal is to check the feasibility of $\delta(L)$. Define

$$I'' := \{v \in V \setminus I' \mid w_v d(v, x) \geq L\}$$

the set of all vertices whose weights have to be changed for $\delta(L)$ (for $a \leq L \leq b$) in addition to those that are in I'. Now we test the feasibility of solution $\delta(L)$:

1. $w_v - \frac{L}{d(v,x)} \leq u_v$ for all $v \in I''$
2. $\sum_{v \in I''} c_v \left(w_v - \frac{L}{d(v,x)}\right) + (\alpha' - L\beta') \leq B$.

Observe that the upper bound constraint of all vertices $v \in I'$ is satisfied. The total cost of solution $\delta(L)$ consists of the cost of vertices in $I' \cup I''$ where $B'(L)$ is the part of vertices in I'.

If $\delta(L)$ is not feasible then $h(x) \geq L$. But then the weights of all vertices $v \in V \setminus (I' \cup I'')$ remain unchanged in an optimal solution. Therefore we can delete all these vertices (set

$V = I' \cup I''$). Now assume that $\delta(L)$ is feasible and therefore $h(x) \leq L$. Then the weight of all vertices in I'' are changed in an optimal solution. Therefore, we can merge I' and I'' and create a new instance:

$$\begin{aligned} I' &= I' \cup I'' \\ \alpha' &= \alpha' + \sum_{v \in I''} c_v w_v \\ \beta' &= \beta' + \sum_{v \in I''} \frac{c_v}{d(v,x)}. \\ a &= \max \left\{ a, \max_{v \in I''} (w_v - u_v) d(v,x), \frac{\alpha' - B}{\beta'} \right\} \\ b &= L \end{aligned}$$

Observe that the new instance again fulfills the inequalities (1) and (2). This procedure is now iterated until all undeleted vertices are in I' because then obviously $a = h(x)$.

Hence, we start with $a = 0$, $b = \max_{v \in V} w_v d(v,x) + 1$ and $I' = \emptyset$. In every iteration the test value L is chosen as the median of $\{w_v d(v,x) \mid v \in V \setminus I'\}$. Let $k = |V \setminus I'|$ then it takes $\mathcal{O}(k)$ time to determine the median, check the feasibility of $\delta(L)$ and to update the instance. If $h(x) \geq L$ then all vertices in $V \setminus (I' \cup I'')$ are deleted, i. e., in the new instance there are at most $\frac{k}{2}$ vertices in $V \setminus I'$. If $h(x) \leq L$ then all vertices in I'' change to I' hence in the new instance there are again at most $\frac{k}{2}$ vertices in $V \setminus I'$. Let $T(k)$ be the time needed to solve an instance with k vertices in $V \setminus I'$ then

$$T(k) = \mathcal{O}(k) + T\left(\frac{k}{2}\right)$$

and hence $T(n) = \mathcal{O}(n)$.

Since we have to determine $h(x)$ for all $x \in V$, we get an overall running time of $\mathcal{O}(n^2)$.

Theorem 3.2. *Upgrading the* 1*-center problem can be solved in* $\mathcal{O}(n^2)$ *time provided that the distance matrix is given.*

4 Downgrading the 1-center by vertex weight modification

In this section, we deal with increasing the vertex weights within certain bounds and fulfilling a budget constraint such that the optimal 1-center objective value after the modification is maximized. It is shown that the problem is in general $\mathcal{NP}$-hard but for the special case of trees there exists a polynomial time algorithm that solves Down1Center in $\mathcal{O}(n^2)$ time.

4.1 NP-hardness proof for general graphs

In this section, we show that Down1Center is strongly $\mathcal{NP}$-hard on general graphs. Lemma 2.4 already suggests a reduction from the total domination problem, which is known to be $\mathcal{NP}$-hard even on bipartite, split, line, circle graphs and several further graph classes (see Haynes, Hedetniemi and Slater [13] for a comprehensive survey).

An instance of the total domination problem is given by a (simple) graph $G = (V, E)$ and a natural number K. The task is to find a total dominating set of cardinality at most K, i. e.,

find a subset of vertices $S \subseteq V$ such that for every $i \in V$ there exists a vertex $j \in S$ such that $(i,j) \in E$. Observe that in contrast to the classical dominating set problem we have to make sure that all vertices within the set S are dominated by another vertex in S. This means, vertices in S do not dominate themselves.

Let $G = (V, E)$ and $K \in \mathbb{N}$ be an instance of the total dominating set problem. Then $\bar{G} = (V, \bar{E})$ denotes the complement of G. We construct an associated instance of Down1Center in the following way: Down1Center should be solved on graph $\bar{G} = (V, \bar{E})$, all edge lengths are equal to one and all vertex weights are equal to zero. Moreover, $c_v = u_v = 1$ for all $v \in V$ and $B = K$. Observe that the distance $d(i,j)$ between two vertices $i, j \in V$ is equal to 1 if $(i,j) \in \bar{E}$ ($(i,j) \notin E$) and at least 2 if $(i,j) \notin \bar{E}$ ($(i,j) \in E$).

We show that there exists a total dominating set of cardinality at most K if and only if there exists a feasible solution of the associated Down1Center problem with objective value at least 2.

Let S be a total dominating set of cardinality at most K then set $\delta_v = 1$ if $v \in S$ and 0 otherwise. Clearly δ is a feasible solution of Down1Center. For every $i \in V$ there exists a vertex $j \in S$ with $(i,j) \in E$, i. e., there exists a vertex j with modified weight $w_j + \delta_j = 1$ such that $(i,j) \notin \bar{E}$. Therefore, $\max\{(w_j + \delta_j)d(i,j) \mid j \in V\} \geq 2$ for every $i \in V$ and hence, we have a feasible solution of Down1Center with objective value at least 2.

Now let δ be a feasible solution of Down1Center with objective value at least 2. We conclude that for every $i \in V$ there exists a vertex $j \in V$ such that $(w_j + \delta_j)d(i,j) \geq 2$. Obviously, $i \neq j$, $\delta_j = 1$ and $(i,j) \notin \bar{E}$ because otherwise either $(w_j + \delta_j) = 0$ or $d(i,j) \leq 1$. Let $S = \{j \in V \mid \delta_j = 1\}$. Clearly, $|S| \leq B$ and for every $i \in V$ there exists a vertex $j \in S$ with $(i,j) \in E$.

Theorem 4.1. *Downgrading the* 1*-center by vertex weight modification is in general strongly* $\mathcal{NP}$*-hard.*

Observe that due to our construction of the reduction above Down1Center is $\mathcal{NP}$-hard on all graph classes $\mathcal{G}$ such that the total domination problem is $\mathcal{NP}$-hard on the complement graphs of $\mathcal{G}$.

4.2 An efficient algorithm for trees

In this subsection a quadratic time algorithm for Down1Center on trees is developed.

Let δ be a feasible solution of Down1Center then Theorem 2.5 implies that the associated optimal objective value with respect to the new weights $w + \delta$ is equal to

$$\max_{\substack{(i,j)\in E \\ s\in T_i(i,j) \\ t\in T_j(i,j)}} \min\{(w_s + \delta_s)d(s,j), (w_t + \delta_t)d(i,t)\}.$$

Therefore, Down1Center on trees can be rewritten in the following way

$$\max_{\delta\in\Delta} \max_{\substack{(i,j)\in E \\ s\in T_i(i,j) \\ t\in T_j(i,j)}} \min\{(w_s + \delta_s)d(s,j), (w_t + \delta_t)d(i,t)\}.$$

Obviously, this is equivalent to

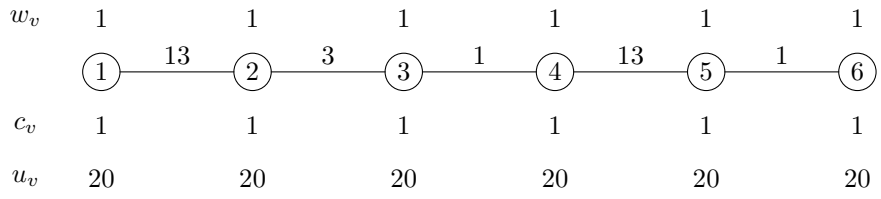

Figure 4: Instance of Down1Center with $B = 20$.

$$\max_{(i,j)\in E} \max_{\substack{\delta\in\Delta \\ s\in T_i(i,j) \\ t\in T_j(i,j)}} \min\{(w_s+\delta_s)d(s,j),(w_t+\delta_t)d(i,t)\}.$$

The last formulation of Down1Center on trees immediately leads to the following algorithm: For every edge $(i,j) \in E$ solve the inner maximization problem. Finally, take the solution associated to that edge that leads to the maximal objective value. Therefore, our task is to solve a series of the following subproblem for fixed $(i,j) \in E$:

$$\text{P(i,j)} \qquad \max_{\substack{\delta\in\Delta \\ s\in T_i(i,j) \\ t\in T_j(i,j)}} \min\{(w_s+\delta_s)d(s,j),(w_t+\delta_t)d(i,t)\}$$

Let us start with two examples. Consider a path with four vertices $V = \{1,2,3,4\}$ with $c_v = w_v = 1$ for all $v \in V$, $u_1 = u_2 = 1$, $u_3 = u_4 = 6$ and $\ell(1,2) = 2$, $\ell(2,3) = \ell(3,4) = 1$. Moreover, the total budget $B = 12$. Before modifying the vertex weights vertex 2 is the unique 1-center with objective value equal to 2. Now fix each edge and solve P(i,j):

- P(1,2): Since $u_1 = 1$ and vertex 1 is the only vertex in $T_1(1,2)$, we conclude that the optimal objective value of P(1,2) is equal to 4.
- P(2,3): It is straightforward to verify that $\delta_1 = 1$ and $\delta_4 = 6$ is an optimal solution and hence we have 6 as optimal objective value.
- P(3,4): Since $3 \in T_3(3,4)$ and $4 \in T_4(3,4)$ and both of these two vertices are allowed to be increased by 6 units, we get 7 as optimal objective value.

We conclude that 7 is the optimal objective value of Down1Center on this instance with $\delta_3 = \delta_4 = 6$. The new vertex weights are of the form $\tilde{w}_1 = 1 = \tilde{w}_2$ and $\tilde{w}_3 = 7 = \tilde{w}_4$. Therefore vertex 3 and vertex 4 are 1-centers after an optimal vertex weight modification.

The above example shows that the optimal 1-center will change after an optimal modification of the vertex weights. Next, consider the instance of Down1Center given in Figure 4:

Vertex 3 is the (vertex) 1-centers with respect to weight w and the absolute 1-center is on edge $(2,3)$. It is a well-known fact that the absolute 1-center lies on an edge which is adjacent to a vertex 1-center. If we consider the downgrading version for the absolute 1-center problem then (since all weights and cost coefficients are equal) it is an optimal solution to increase the weight of vertex 1 and of vertex 6 by 10 units each. The (downgraded) absolute 1-center with respect to the new weights lies on edge $(2,3)$.

Now let us consider the downgrading version for the vertex 1-center problem: We solve P(i,j) for every edge (i,j) and get the following optimal solutions for the subproblems:

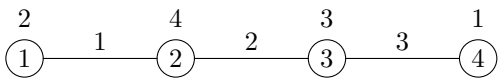

Figure 5: Example: Illustration

	optimal solution	optimal objective value
P(1, 2)	$\delta_1 = \frac{29}{2}$, $\delta_6 = \frac{11}{2}$	201.5
P(2, 3)	$\delta_1 = \frac{181}{17}$, $\delta_6 = \frac{159}{17}$	≈ 186.4
P(3, 4)	$\delta_1 = \frac{149}{16}$, $\delta_6 = \frac{171}{16}$	≈ 175.3
P(4, 5)	$\delta_1 = 6$, $\delta_6 = 14$	210
P(5, 6)	$\delta_6 = 20$	21

First, we can see that the optimally downgraded vertex 1-center is equal to vertex 4 and 5, i. e., the downgraded absolute 1-center is not adjacent to the downgraded vertex 1-center. Moreover, the optimal objective value for the subproblem P(i, j) are not unimodal along a path.

Let us start with investigating subproblem P(i, j). Observe that a feasible solution of P(i, j) is a triple (δ, s, t). Let us start with some observations about the structure of an optimal solution of P(i, j):

1. (Obs.1) We may assume without loss of generality that $c_k u_k \leq B$ for all $k \in V$. Otherwise, we may use the modified bounds $\tilde{u}_k = \min\left\{u_k, \frac{B}{c_k}\right\}$.

2. (Obs.2) There exists an optimal solution (δ, s, t) such that at most two weights are changed, i. e., $\delta_k = 0$ for all $k \neq s, t$.

3. (Obs.3) There exists an optimal solution (δ, s, t) of P(i, j) such that either $c_s\delta_s + c_t\delta_t = B$ or $\delta_s = u_s$, $\delta_t = u_t$ and $c_s u_s + c_t u_t < B$. This can be proved by assuming that $c_s\delta_s + c_t\delta_t < B$. If $\delta_s < u_s$ then δ_s can be increased without violating the feasibility of the solution while the new solution is still optimal. The same can be done for δ_t. This procedure is repeated until either $c_s\delta_s + c_t\delta_t = B$ or $\delta_s = u_s$ and $\delta_t = u_t$.

In order to solve P(i, j), we have to decide about the amount of investment into subtree $T_i(i, j)$ and $T_j(i, j)$. As soon as we know the amount of investment into the two subtrees, we invest into these vertices that are most valuable. Therefore, we introduce a profit function with parameter λ where λ represents the amount of investment into subtree $T_i(i, j)$:

$$p_k(\lambda) := \begin{cases} (w_k + \min\{u_k, \frac{\lambda}{c_k}\})d(k, j) & \text{if } k \in T_i(i, j) \\ (w_k + \min\{u_k, \frac{B-\lambda}{c_k}\})d(i, k) & \text{if } k \in T_j(i, j) \end{cases}$$

for $k \in V$.

Consider the instance given in Figure 5 with $B = 10$ and $c_1 = c_2 = 1$, $c_3 = 2$, $c_4 = 5$ and $u_1 = 3$, $u_2 = u_3 = 4$ and $u_5 = 2$. An illustration of the corresponding profit functions for fixed edge $(i, j) = (2, 3)$ is given in Figure 6.
Observe that the solution (δ, s, t) with

$$\delta_s = \min\left\{u_s, \frac{\lambda}{c_s}\right\} \qquad \delta_t = \min\left\{u_t, \frac{B-\lambda}{c_t}\right\}$$

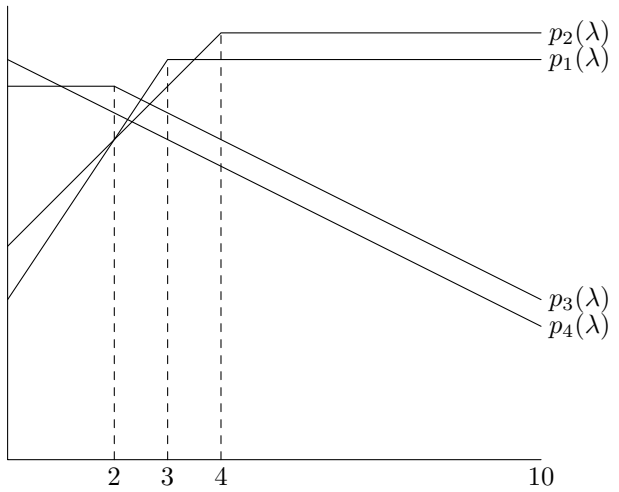

Figure 6: Example: Illustration of the functions $p_k(\lambda)$.

and $\delta_k = 0$ for all $k \neq s, t$ is a feasible solution for every $0 \leq \lambda \leq B$, $s \in T_i(i,j)$ and $t \in T_j(i,j)$. This is true because the upper bounds are fulfilled and

$$c_s\delta_s + c_t\delta_t \leq c_s\frac{\lambda}{c_s} + c_t\frac{B-\lambda}{c_k} = B.$$

Hence, given a triple (λ, s, t), we get an associated feasible solution of P(i, j) with objective value

$$\min\{p_s(\lambda), p_t(\lambda)\}.$$

Now let $0 \leq \lambda \leq B$ then we define

$$\begin{aligned} s(\lambda) &:= \arg\max_{s \in T_i(i,j)} p_s(\lambda) \\ t(\lambda) &:= \arg\max_{t \in T_j(i,j)} p_t(\lambda) \\ \alpha(\lambda) &:= p_{s(\lambda)}(\lambda) \\ \beta(\lambda) &:= p_{t(\lambda)}(\lambda) \end{aligned}$$

Observe that given a parameter value λ then $s(\lambda) \in T_i(i,j)$ and $t(\lambda) \in T_j(i,j)$, respectively, are the most valuable vertices if λ is the investment into $T_i(i,j)$ and $B - \lambda$ is the investment into $T_j(i,j)$. The functions $\alpha(\lambda)$ and $\beta(\lambda)$ are introduced in order to simplify the notation. First we will reformulate P(i, j) as optimization problem over $\alpha(\lambda)$ and $\beta(\lambda)$ and finally these two functions are again translated in terms of $p_k(\lambda)$. Observe that $\alpha(\lambda)$ is monotonically increasing since all functions $p_k(\lambda)$ for $k \in T_i(i,j)$ are increasing while $\beta(\lambda)$ is monotonically decreasing. See Figure 7 for an illustration of $\alpha(\lambda)$ and $\beta(\lambda)$.

Obviously, if $0 \leq \lambda \leq B$ and $s \in T_i(i,j)$, $t \in T_j(i,j)$ we have

$$\min\{p_s(\lambda), p_t(\lambda)\} \leq \min\{\alpha(\lambda), \beta(\lambda)\}. \tag{3}$$

It remains to show that we may restrict ourselves to the optimization over $\min\{\alpha(\lambda), \beta(\lambda)\}$. First we show that there exists a value $0 \leq \lambda \leq B$ such that the associated feasible solution for $s(\lambda)$ and $t(\lambda)$ is optimal.

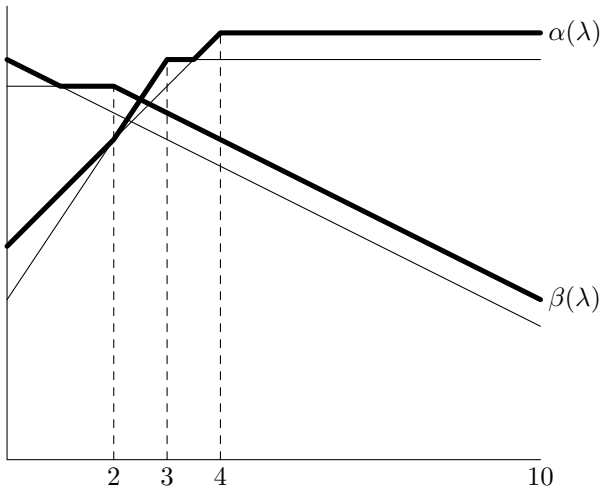

Figure 7: Illustration of the profit functions $p_k(\lambda)$.

Lemma 4.2. *Let (δ, s, t) be an optimal solution such that Obs. 3 holds. Then there exists a parameter value $0 \leq \lambda \leq B$ such that*

$$(w_s + \delta_s)d(s, j) = p_s(\lambda) \text{ and } (w_t + \delta_t)d(i, t) = p_t(\lambda).$$

Proof. Set $\lambda = c_s\delta_s$. Since Obs. 3 holds, we distinguish the following two cases:

1. Case: $c_s\delta_s + c_t\delta_t = B$
   Observe that $\lambda = c_s\delta_s \leq c_s u_s$ and $B - \lambda = c_t\delta_t \leq c_t u_t$ holds. Therefore, $\min\{u_s, \frac{\lambda}{c_s}\} = \frac{\lambda}{c_s} = \delta_s$ and $\min\{u_t, \frac{B-\lambda}{c_t}\} = \frac{B-\lambda}{c_t} = \delta_t$.
2. Case: $\delta_s = u_s$, $\delta_t = u_t$ and $c_s u_s + c_t u_t < B$
   Then $\lambda = c_s u_s$ and $B - \lambda = B - c_s u_s > c_t u_t$ holds and we conclude $\min\{u_s, \frac{\lambda}{c_s}\} = u_s = \delta_s$ and $\min\{u_t, \frac{B-\lambda}{c_t}\} = u_t = \delta_t$.

Obviously the above result immediately follows. □

Combining the previous results, we get the following implications: Let z^* be the optimal objective value of P(i, j). Then there exists an optimal solution (δ, s, t) which satisfies Obs. 3. Due to Lemma 4.2 there exists a value $\lambda' \in [0, B]$ with

$$(w_s + \delta_s)d(s, j) = p_s(\lambda') \text{ and } (w_t + \delta_t)d(i, t) = p_t(\lambda').$$

Therefore,

$$z^* = \min\{p_s(\lambda'), p_t(\lambda')\}.$$

According to (3) we know that

$$z^* = \min\{p_s(\lambda'), p_t(\lambda')\} \leq \min\{\alpha(\lambda'), \beta(\lambda')\}.$$

But since there exists a feasible solution with objective value $\min\{\alpha(\lambda'), \beta(\lambda')\}$ and z^* is the optimal objective value we conclude

$$z^* = \min\{\alpha(\lambda'), \beta(\lambda')\}.$$

Since there exists a feasible solution with objective value $\min\{\alpha(\lambda), \beta(\lambda)\}$ for every $0 \leq \lambda \leq B$ it follows that

$$z^* \geq \min\{\alpha(\lambda), \beta(\lambda)\}$$

holds for every $0 \leq \lambda \leq B$. We conclude that

$$z^* = \max_{0 \leq \lambda \leq B} \min\{\alpha(\lambda), \beta(\lambda)\}. \tag{4}$$

Observe that due to the monotonicity and continuity of α and β there exists an intersection point of $\alpha(\lambda)$ and $\beta(\lambda)$ in $[0, B]$ if and only if $\alpha(0) \leq \beta(0)$ and $\alpha(B) \geq \beta(B)$.

Lemma 4.3. *The optimal objective value z^* of P(i, j) is of the form*

$$z^* = \begin{cases} \max\limits_{s \in T_i(i,j)} p_s(B) & \text{if } \max\limits_{s \in T_i(i,j)} p_s(B) \leq \max\limits_{t \in T_j(i,j)} p_t(B) \\ \max\limits_{t \in T_i(i,j)} p_t(0) & \text{if } \max\limits_{s \in T_i(i,j)} p_s(0) \geq \max\limits_{t \in T_j(i,j)} p_t(0) \\ \min\limits_{0 \leq \lambda \leq B} \max\limits_{k \in V} p_k(\lambda) & \text{otherwise} \end{cases}$$

Before proving this lemma, we try to get an intuition about its statement. If the total investment is put into $T_i(i, j)$ and still the weighted distance from the most valuable vertex in $T_i(i, j)$ to j is at most the weighted distance from the best vertex in $T_j(i, j)$ then the best choice is to invest the whole budget into $T_i(i, j)$ because the minimum is always attained by a vertex in $T_i(i, j)$. The second case reflects the reverse situation to the first one, i. e., the whole budget is used for $T_j(i, j)$ but still the minimum is attained by a vertex in $T_j(i, j)$. Finally, if the minimum may be attained by a vertex in $T_i(i, j)$ or by a vertex in $T_i(i, j)$ then we have to minimize over the upper envelope of the piecewise linear functions $p_k(\lambda)$.

Proof. Recall that $\alpha(\lambda)$ is monotonically increasing while $\beta(\lambda)$ is monotonically decreasing. We distinguish two cases:

1. Case: There exists a $\lambda' \in [0, B]$ such that $\alpha(\lambda') = \beta(\lambda')$
   Then λ' is an optimal solution of P(i, j) because

   $$\min\{\alpha(\lambda), \beta(\lambda)\} = \begin{cases} \alpha(\lambda) \leq \alpha(\lambda') & \text{if } \lambda \leq \lambda' \\ \beta(\lambda) \leq \beta(\lambda') & \text{if } \lambda \geq \lambda' \end{cases}$$

   But due to the monotonicity of α and β we have

   $$\max\{\alpha(\lambda), \beta(\lambda)\} = \begin{cases} \alpha(\lambda) \geq \alpha(\lambda') & \text{if } \lambda \geq \lambda' \\ \beta(\lambda) \geq \beta(\lambda') & \text{if } \lambda \leq \lambda' \end{cases}$$

   Therefore, $z^* = \min_{0 \leq \lambda \leq B} \max\{\alpha(\lambda), \beta(\lambda)\}$ holds. Using the definition of α and β we get $\max\{\alpha(\lambda), \beta(\lambda)\} = \max_{k \in V} p_k(\lambda)$ which proves the third branch of the lemma.

2. Case: There is no intersection point of $\alpha(\lambda)$ and $\beta(\lambda)$ in $[0, B]$

   (a) $\alpha(B) \leq \beta(B)$: This property is equivalent to the first branch of the lemma's statement. Due to the monotonicity of α and β, we know that $\alpha(\lambda) \leq \beta(\lambda)$ for all $0 \leq \lambda \leq B$. Hence, we have

   $$z^* = \max_{0 \leq \lambda \leq B} \min\{\alpha(\lambda), \beta(\lambda)\} = \max_{0 \leq \lambda \leq B} \alpha(\lambda) = \alpha(B).$$

(b) $\alpha(0) \geq \beta(0)$: Finally, we deal with the second branch of the lemma's statement. Here we have $\alpha(\lambda) \geq \beta(\lambda)$ for all $0 \leq \lambda \leq B$ and therefore

$$z^* = \max_{0 \leq \lambda \leq B} \min\{\alpha(\lambda), \beta(\lambda)\} = \max_{0 \leq \lambda \leq B} \beta(\lambda) = \beta(0).$$

□ □

Observe that Lemma 4.3 implies the following algorithm for Down1Center on trees:

- For every $(i, j) \in E$:
  - If $\max_{s \in T_i(i,j)} p_s(B) \leq \max_{t \in T_j(i,j)} p_t(B)$ then

$$z^* = \max_{s \in T_i(i,j)} p_s(B)$$
$$\delta_k = \begin{cases} B & \text{if } k \in T_i(i,j), k = s(B) \\ 0 & \text{otherwise} \end{cases}$$

  - Else if $\max_{s \in T_i(i,j)} p_s(0) \geq \max_{t \in T_j(i,j)} p_t(0)$ then

$$z^* = \max_{t \in T_i(i,j)} p_t(0)$$
$$\delta_k = \begin{cases} B & \text{if } k \in T_j(i,j), k = t(0) \\ 0 & \text{otherwise} \end{cases}$$

  - Else solve $\min_{0 \leq \lambda \leq B} \max_{k \in V} p_k(\lambda)$

The first two cases can be solved in $\mathcal{O}(n)$ time since one has to evaluate the profit functions p at certain values and compare the function values. For the third case the problem of minimizing over the upper envelope of piecewise linear and monotone functions has to be solved. In the next section it is proved that this problem can also be solved in $\mathcal{O}(n)$ time. Therefore, we get the following theorem:

Theorem 4.4. *The Downgrading 1-center problem on a tree can be solved in $\mathcal{O}(n^2)$ time.*

5 Minimizing over the upper envelope of piecewise linear and monotone functions

In this section, we show that minimizing over the upper envelope of piecewise linear and monotone functions can be done in $\mathcal{O}(K)$ time where K is the total number of breakpoints.

Let n piecewise linear functions $f_i : [0, B] \to \mathbb{R}$ for $i = 1, \ldots, n$ be given. We consider the following problem:

$$\min_{0 \leq \lambda \leq B} \max_{i=1,\ldots,n} f_i(\lambda) \tag{5}$$

this means, we want to find a minimizer on the upper envelope.

We assume that every function has at least one breakpoint. If this is not the case then we assign breakpoint B to all affine-linear functions. Let K be the total number of breakpoints then clearly $n \leq K$ holds.

Moreover, we assume that the upper envelope of the functions f_i $(i = 1, \ldots, n)$ is unimodal. This is the case if each function f_i is either monotonically increasing or decreasing.

Our algorithm is a generalization of Megiddo's search and prune algorithm for solving linear programs with two variables which is equivalent to problem (5) where all functions are affine-linear. The main idea is to reduce the number of breakpoints successively. This is done by splitting up the current interval into two subintervals and continuing with that subinterval which contains an optimal solution. In each interval reduction procedure at least $\frac{1}{8}$ of the breakpoints are dropped. During the whole algorithm it is guaranteed that every function has at least one breakpoint in the current interval.

Let $\lambda \in [0, B]$. Then it is possible to decide whether there exists an optimal solution $\lambda^* \in [0, B]$ with $\lambda^* \leq \lambda$ or $\lambda^* \geq \lambda$ in the following way: Let $I = I^+ \cup I^-$ be the set of functions where I^+ is the set of increasing functions and I^- is the set of decreasing functions. Then determine $\max_{k \in I^+ \cup I^-} f_k(\lambda) = f_j(\lambda)$. If $j \in I^+$ then $\lambda^* \leq \lambda$ and otherwise $\lambda^* \geq \lambda$. Obviously, it takes $\mathcal{O}(|I|)$ time to decide which of the two cases is true, i. e., the time we need is linear in the number of functions.

The above described test of a fixed value λ is used several times during our algorithm. Each iteration of the algorithm consists of two steps: In the first step the number of breakpoints and in the second step the number of functions is reduced.

1. The breakpoint reduction step:

   Determine the median m among the breakpoints and test the value m. In order to simplify the notation let us assume that $m \leq \lambda^*$. Then there are $a \leq \frac{K}{2}$ breakpoints in $(m, B]$. Hence, we have halved the number of breakpoints.

2. The function reduction step:

   Observe that there are $K - a$ breakpoints in $[0, m]$ and since every function has at least one breakpoint there are at most $b \leq K - a$ functions having no breakpoint in $(m, B]$. Observe that our goal is to continue with interval $[m, B]$. Even if there are at most $a \leq \frac{K}{2}$ breakpoints in this interval, we have to make sure that every function has at least one breakpoint in $[m, B]$. Simply assigning breakpoint m to all functions that are affine-linear in $[m, B]$ will in general not decrease the number of breakpoints, e. g., if every function has exactly one breakpoint. Therefore the goal is to reduce the number of affine-linear functions in $[m, B]$. We do this on the line of Megiddo's linear time algorithm for linear programming problems with two variables:

   Take all functions that are affine-linear in $[m, B]$ and arrange them in disjoint pairs. If the number of of affine-linear functions is odd then one function remains unmachted. Then determine the intersection points of these pairs. If two affine-linear functions have no intersection point, i. e., they are parallel, then one function can be deleted since it is dominated by its partner function. Let $2b_1$ be the number functions that are matched with a parallel function and let $2b_2$ be the number of remaining matched functions. Then there are b_2 intersection points. Then the median m' of the intersection points is determined and it is tested whether $m' \leq \lambda^*$ or $m' \geq \lambda^*$. Assume that $m' \geq \lambda^*$ then there are at least $\lceil \frac{b_2}{2} \rceil \geq \frac{b_2}{2}$ breakpoints in $[m', B]$ (in the wrong subinterval). Observe

that we will continue with interval $[m, m']$. For every pair whose intersection point is in $[m', B]$ one affine-linear function can be deleted since it is dominated in $[m, m']$ by its partner function. Hence, we can delete at least $\frac{b_2}{2}$ affine-linear functions in addition to all those function that were deleted because they have a parallel partner function. Therefore the number of remaining affine-linear functions is at most

$$b_1 + \left(2b_2 - \frac{b_2}{2}\right) + 1 \leq \frac{3b+1}{4}.$$

Finally, one (arbitrarily chosen) endpoint of the new interval is assigned to the remaining affine-linear functions as breakpoint. Again every function has a breakpoint.

Next the running time of the algorithm is analyzed. We start with the breakpoint reduction step: It takes $\mathcal{O}(K)$ time to determine the median among the breakpoints. Since every function has at least one breakpoint, we have $n \leq K$ and therefore $\mathcal{O}(K)$ is an upper bound for the time needed to decide whether $m \leq \lambda^*$ or $m \geq \lambda^*$.

Next the running time of the function reduction step is investigated: First we have to find all affine-linear functions, match them and determine the intersection points. This can be done in $\mathcal{O}(b)$ time. Then the median among the intersection points is determined. This takes again $\mathcal{O}(b)$ time. To test the median m', we have to consider all functions (including all functions with at least one breakpoint). Therefore, we need $\mathcal{O}(K)$ time. And finally all intersection points are checked and several functions are deleted. This takes again $\mathcal{O}(b)$ time. Therefore, the total running time of this second step is $\mathcal{O}(K)$ time.

Putting all together the number of breakpoints in the new interval $[m, m']$ is at most

$$a + \frac{3b+1}{4} \leq a + \frac{3}{4}(K-a) + \frac{1}{4} = \frac{3}{4}K + \frac{1}{4}a + \frac{1}{4} \leq \frac{7}{8}K + \frac{1}{4}.$$

Let $T(K)$ be the time needed to solve the problem if there are K breakpoints then we conclude that the following recurrence equation holds:

$$T(K) = \mathcal{O}(K) + T\left(\frac{7}{8}K + \frac{1}{4}\right)$$

and hence $T(K) = \mathcal{O}(K)$.

6 Conclusion

In this paper two network modification problems, the upgrading 1-center problem and the downgrading 1-center problem, are investigated. The upgrading 1-center problem can be solved in quadratic time if the shortest distances are available while the corresponding downgrading version is in general strongly $\mathcal{NP}$-hard. However, if the underlying graph is a tree then both versions can be solved in quadratic time.

The suggested algorithm for the downgrading version of the 1-center problem includes an algorithm for minimizing over the upper envelope of piecewise linear and monotone functions in $\mathcal{O}(K)$ time where K is the number of breakpoints. We assume that this algorithm can be used as subroutine for several problems in different areas of optimization in order to improve the running time of existing algorithms.

References

[1] R. E. Burkard, E. Gassner, and J. Hatzl, A linear time algorithm for the reverse 1-median problem on a cycle, *Networks* **48**, No. 1, 16–23, 2006.

[2] R. E. Burkard, E. Gassner, and J. Hatzl, The reverse 2-median problem on trees, accepted for publication in *Discrete Applied Mathematics*.

[3] R.E. Burkard, B. Klinz, and J. Zhang, Bottleneck capacity expansion problems with general budget constraints, *RAIRO Recherche Operationelle* **35**, 1–20, 2001.

[4] R.E. Burkard, Y. Lin, and J. Zhang, Weight reduction problems with certain bottleneck objectives, *European Journal of Operational Research* **153**, 191–199, 2004.

[5] R.E. Burkard, C. Pleschiutschnig, and J. Zhang, Inverse median problems, *Discrete Optimization* **1**, 23–39, 2004.

[6] M.C. Cai, X.G. Yang, and J. Zhang, The complexity analysis of the inverse center location problem, *Journal of Global Optimization* **15**, 213–218, 1999.

[7] K.U. Drangmeister, S.O. Krumke, M.V. Marathe, H. Noltemeier, and S.S. Ravi, Modifying edges of a network to obtain short subgraphs, *Theoretical Computer Science* **203**, 91–121, 1998.

[8] Z. Drezner and H. W. Hamacher, *Facility location. Applications and Theory.* Springer, Berlin, 2002.

[9] G.N. Frederickson and R. Solis-Oba, Increasing the weight of minimum spanning trees, *Journal of Algorithms* **33**, 244–266, 1999.

[10] D.R. Fulkerson and G.C. Harding, Maximizing the minimum source-sink path subject to a budget constraint, *Mathematical Programming* **13**, 116–118, 1977.

[11] E. Gassner, The Inverse 1-Maxian Problem with edge length modification, accepted for publication in *Journal of Combinatorial Optimization*.

[12] S. E. Hambrusch and Hung-Yi Tu, Edge Weight Reduction Problems in Directed Acyclic Graphs, *Journal of Algorithms* **24**, Issue 1, 66–93, 1997.

[13] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, *Fundamentals of domination in graphs.* Pure and Applied Mathematics. Marcel Dekker, New York, 1998.

[14] C. Heuberger, Inverse Combinatorial Optimization: A Survey on Problems, Methods, and Results, *Journal of Combinatorial Optimization* **8**, 329–361, 2004.

[15] O. Kariv and S. L. Hakimi, An algorithmic approach to network location problems. I: The p-centers and II: The p-medians. *SIAM Journal on Applied Mathematics* **37**, No. 3, 513–560, 1979.

[16] S.O. Krumke, M.V. Marathe, H. Noltemeier, R. Ravi, and S.S. Ravi, Approximation algorithms for certain network improvement problems, *Journal of Combinatorial Optimization* **2**, 257–288, 1998.

[17] N. Megiddo, Linear-Time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM Journal on Computing* **12**, No. 4, 759–776, 1983.

[18] P. B. Mirchandani and R. L. Francis, *Discrete Location Theory.* Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Ltd., New York, 1990.

[19] C. A. Phillips, The network inhibition problem, *Annual ACM Symposium on Theory of Computing*, Proceedings of the twenty-fifth annual ACM symposium on Theory of computing table of contents, San Diego, California, United States, 776 – 785, 1993.

[20] J. Zhang, X.G. Yang and M.C. Cai, Inapproximability and a polynomially solvable special case of a network improvement problem, *European Journal of Operational Research* **155**, 251–257, 2004.

[21] J. Zhang, X.G. Yang and M.C. Cai, A network improvement problem under different norms, *Computational Optimization and Applications* **27**, 305–319, 2004.