DEPARTMENT OF OPERATIONS RESEARCH
UNIVERSITY OF AARHUS

Working Paper no. 2000/6

# Multicriteria Semi-obnoxious Network Location Problems (MSNLP) with Sum and Center Objectives

Horst W. Hamacher
Martine Labbé
Stefan Nickel
Anders J.V. Skriver

ISSN 1398-8964

# Multicriteria Semi-obnoxious Network Location Problems (MSNLP) with Sum and Center Objectives

HORST W. HAMACHER
Fachbereich Mathematik
Universität Kaiserslautern
Kurt-Schumacher-Strasse 26
67663 Kaiserslautern
Germany

MARTINE LABBÉ
Service de Mathématiques de la Gestion
Université Libré de Bruxelles
Brussels
Belgium

STEFAN NICKEL
ITWM
Universität Kaiserslautern
Ottlieb-Daimler-Strasse
Gebäude 49
67663 Kaiserslautern
Germany

ANDERS J. V. SKRIVER*
Department of Operations Research
University of Aarhus
Building 530, Ny Munkegade
DK - 8000 Århus C
Denmark

November 17, 2000

**Abstract**

Locating a facility is often modeled as either the maxisum or the minisum problem, reflecting whether the facility is undesirable (obnoxious) or desirable. But many facilities are both desirable and undesirable at the same time, e.g. an airport. This can be modeled as a multicriteria network location problem, where some of the sum-objectives are maximized (push effect) and some of the sum-objectives are minimized (pull effect).

We present a polynomial time algorithm for this model along with some basic theoretical results, and generalize the results also to incorporate maximin and minimax objectives. In fact, the method works for any piecewise linear objective functions. Finally, we present some computational results.

Keywords: MCDM, Multicriteria, Obnoxious, Semi-obnoxious, Facility Location, Networks.

## 1 Introduction

There are a number of models that deal with the problem of locating (placing) a new facility on a network. Most of these models locate a desirable facility, such as a supermarket or a

*Corresponding author.

fire station, where the objective is to keep the new facility close to its users (pull effect). There are also some models describing how to locate an obnoxious (undesirable) facility such as a nuclear power plant or a dump site which the users want to locate far away (push effect). Many facilities can, however, be thought of as semi-obnoxious. Such facilities could be airports, train stations or other noisy service facilities. It could also be the above-mentioned dump site that, with respect to transportation costs, should be located centrally, but, in the opinion of the citizens, should be located distantly. These location problems could with obvious advantages be formulated as multicriteria network location problems. In this way the trade-off between the different objectives can be revealed, making a good basis for an overall decision. Different aspects of the problem can be described by different objectives. Such objectives could be transportation costs, travel time, air pollution or minimizing the number of citizens within a certain radius of the facility. Another situation arises when we have more decision makers, each having their own objective function. When we solve a problem with more than one objective, it is highly unlikely that one solution is optimal for all objectives. Instead, the solution is the set of efficient or Pareto locations, i.e. solutions where we cannot improve any objective without at least one other objective being worsened.

Bicriterion models for the planar case of the problem is presented in Brimberg and Juel [1], Carrizosa *et al.* [2] and Andersen and Skriver [10]. In Andersen and Skriver [10] an approximation solution method for the bicriterion network location problem is also presented. A general solution method for the multicriterion median-problem is presented in Hamacher *et al.* [5].

As one notices, the terminology for location problems is not unique. Therefore we introduce in the following a classification scheme for location problems that should help get an overview over the manifold area of location problems.
We use a scheme which is analogous to the one introduced successfully in scheduling theory. The presented scheme for location problems was developed in Hamacher and Nickel [6] and Hamacher *et al.* [5].
We have the following five position classification

$$pos1/pos2/pos3/pos4/pos5 \ ,$$

where the meaning of each position is explained in Table 1:
If we do not make any special assumptions in a position, we indicate this by a $\bullet$.

The rest of the paper is organized as follows. In Section 2 we give some definitions and

| Position | Meaning | Usage (Examples) |
|---|---|---|
| 1 | number of new facilities | |
| 2 | type of problem | **P** planar location problem<br>**D** discrete location problem<br>**G** network location problem |
| 3 | special assumptions and restrictions | $w_m = 1$ all weights are equal<br>$\mathcal{R}$ a forbidden region |
| 4 | type of distance function | $l_1$ Manhattan metric<br>$d(\mathcal{V}, \mathcal{V})$ node to node distance<br>$d(\mathcal{V}, G)$ node to point distance |
| 5 | type of objective function | $\sum$ median problem<br>$\sum_{obnox}$ anti-median problem<br>max center problem<br>$\max_{obnox}$ anti-center problem |

Table 1: Classification scheme for location problems.

describe the problem. The general solution procedure is described in Section 3, and in Section 4 we present a different approach that works only in the bicriteria case. In Section 5 we discuss how the general solution procedure can also be used with center objectives. Computational results are presented in Section 6, and we conclude the paper in Section 7.

## 2 Problem formulation and definitions

We are given a (strongly) connected network $G(\mathcal{V}, \mathcal{E})$ with nodeset $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ where $|\mathcal{V}| = n$ nodes, and edgeset $\mathcal{E} = \{(v_i, v_j), (v_k, v_l), \ldots, (v_p, v_q)\}$ with $|\mathcal{E}| = m$ edges. If the underlying graph is directed it is denoted $G_D$, and the edge $e = (v_i, v_j)$ has head $v_j$ and tail $v_i$. If the underlying graph is undirected, it is just denoted $G$, and $e = (v_i, v_j) = (v_j, v_i)\ \forall e \in \mathcal{E}$. We define the set of objectives as $\mathcal{Q} = \{1, 2, \ldots, Q\}$. Each node $v_i$ carries $Q$ weights $(w_i^1, w_i^2, \ldots, w_i^Q)^t$, where $w_i^q > 0, \forall q \in \mathcal{Q}$, so we may refer to the matrix of weights by $W_{Q\times n}$. Each edge $e \in \mathcal{E}$ has length $l(e) \in R_+$.
By $d(v_h, v_k)$ we denote the distance between $v_h$ and $v_k$, is given by the length of a shortest path between $v_h$ and $v_k$. A point $x \in G(\mathcal{V}, \mathcal{E})$ can be located both at a node or on an edge. This is often referred to as absolute location.
We define a **point** $x$ on a directed edge $e = (v_i, v_j)$ as a tuple $x = (e, t), t \in [0, 1]$, with

$$d(v_k, x) = d(v_k, v_i) + tl(e) \quad \text{and} \quad d(x, v_k) = (1-t)l(e) + d(v_j, v_k)$$

for any $v_k \in \mathcal{V}$. A **point** $x$ on an undirected edge $e = (v_i, v_j)$ is defined as a touple

$x = (e,t), t \in [0,1]$, with

$$d(x, v_k) = \min\{d(v_k, v_i) + tl(e), d(v_k, v_j) + (1-t)l(e)\}$$

for any $v_k \in \mathcal{V}$. Notice that $d(v_i, x) = tl(e)$ and $d(x, v_j) = (1-t)l(e)$ for $x = (e,t)$. Since $v_i = (e,0)$ and $v_j = (e,1)$, all nodes of the network are also points of the network.
The set $\{(e,t)|t \in (t_1,t_2), t_1, t_2 \in [0,1]\}$, forming an open **subedge** on $e$, is denoted $(e,(t_1,t_2))$ for any $e \in \mathcal{E}$. Of course this set is empty, unless $t_2 > t_1$. Similarly, we define closed and half right/left open subedges.
We formulate the model with the maxisum and minisum objectives, which are obviously negatively correlated. These objective functions are often referred to as the **weighted anti-median** and **median** of a network. In Section 5 we discuss the maximin and minimax objectives. For the undirected problem the objective functions are defined by

$$f^q(x) = \sum_{i=1}^{n} w_i^q \; d(x, v_i) \quad q \in \mathcal{Q} \tag{1}$$

and for the directed case they are defined by

$$f^q(x) = \sum_{i=1}^{n} w_i^q \; (d(x, v_i) + d(v_i, x)) \quad q \in \mathcal{Q} \tag{2}$$

In (2) observe that we for each node $v_i$ make a round-trip from $x$ to $v_i$ and back to $x$. In some applications it may be more appropriate to look only at the distances out of $x$ or into $x$. The general undirected problem $1/G/\bullet/d(\mathcal{V},G)/(Q_1\text{-}\sum_{obnox}, Q_2\text{-}\sum)_{Par}$ is formulated as follows:

$$\begin{array}{ll} \max & f^q(x) \quad q \in \mathcal{Q}_1 \\ \min & f^q(x) \quad q \in \mathcal{Q}_2 \\ \text{s.t.} & \\ & x \in G(\mathcal{V}, \mathcal{E}) \end{array} \tag{3}$$

$\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$, where $\mathcal{Q}_1 \cap \mathcal{Q}_2 = \emptyset$. $\mathcal{Q}_1$ is the set of obnoxious objective functions, and $\mathcal{Q}_2$ is the set of desirable objective functions. At most one of the sets are allowed to be empty. If $\mathcal{Q}_1 = \emptyset$ we have the situation discussed in Hamacher, Labbé and Nickel [5]. $f(x) = (f^1(x), f^2(x), \ldots, f^Q(x))^t$.

For simplicity in the succeeding argumentation we multiply all objective functions in $\mathcal{Q}_1$ by $-1$ in order to minimize instead of maximize. Thus, in the remaining part of the paper we assume that $w_i^q < 0, \forall i = 1, 2, \ldots, n$ and $q \in \mathcal{Q}_1$, and $w_i^q > 0, \forall i = 1, 2, \ldots, n$ and $q \in \mathcal{Q}_2$. We now have a multicriteria minimization model:

$$\begin{array}{ll} \min & f^q(x) \quad q \in \mathcal{Q}_1 \\ \min & f^q(x) \quad q \in \mathcal{Q}_2 \\ \text{s.t.} & \\ & x \in G(\mathcal{V}, \mathcal{E}) \end{array} \tag{4}$$

In order to find the shortest distances between $x$ and all the nodes, we need the **distance matrix** $D$ of shortest distances between all pairs of nodes. Note that $D_{ij} = d(v_i, v_j)$. This matrix can be calculated in $O(n^3)$ running time using Floyd's algorithm or by applying Dijkstra's algorithm to all $n$ nodes. For details on these graph procedures, see Thulasiraman and Swamy [13]. For an undirected network the distance matrix $D$ is symmetric.

This model is a combination of two well-known models. The minisum and the maxisum models. The solution procedures for these two models are similar, but we will explain the most important details here. For the maxisum problem, some interesting theory is found in Church and Garfinkel [3]. They introduce the concept of bottleneck points, and refer to nodes with degree one as **dangling nodes** (often called pendant nodes). The minisum problem has been well studied, and we refer to Daskin [4] for details.

We will now outline the concept of bottleneck-points as it is presented in Church and Garfinkel [3]. There are two types of bottleneck-points. The edge-bottleneck-points are defined as follows, for each edge $(v_i, v_j) \in \mathcal{E}$: Let $x$ be on the edge $(v_i, v_j)$. If there exists a node $v_k \neq v_i, v_j$ such that

$$D_{ki} + d(x, v_i) = D_{kj} + d(x, v_j)$$

then $x$ is an **edge-bottleneck-point**. It is easily seen, that edge $(v_i, v_j)$ contains an edge-bottleneck-point with respect to node $v_k$ if and only if

$$|D_{ki} - D_{kj}| < l((v_i, v_j))$$

This sets the upper bound for the number of edge-bottleneck-points on an edge to $n - 2$. Now we define the node-bottleneck-points. Assume there exists distinct nodes $v_i$,$v_h$ and $v_k$. If there exists a node $v_j \neq v_i, v_h, v_k$ such that

$$D_{ik} + D_{kj} = D_{ih} + D_{hj}$$

then node $v_j$ is a **node-bottleneck-point** with respect to node $v_i$ (and $v_i$ to $v_j$). Considering the whole edge $(v_i, v_j)$ including the nodes, it contains at most $n$ bottleneck-points. Since there are $m$ edges in $G$, the total number of bottleneck-points is bounded by $mn$.
It is important to note that the bottleneck-points are independent of the weights. They only depend on the network structure including the edge-lengths. We will denote the **edge-bottleneck-point matrix** of shortest distances from all edge-bottleneck-points to all nodes by $B$. So $B_{ij}$ is the shortest distance from edge-bottleneck-point $B_i$ to node $v_j$. This matrix is needed for easy calculation of the objective-values in the bottleneck-points.

When we know the shortest distance matrix $D$, the bottleneck-points can be calculated in $O(mn)$ running time, because for each edge we have to evaluate all nodes. This can be improved to an algorithm that takes $O(n\ log\ n)$ time, see Hansen *et al.* [7].

In Church and Garfinkel [3] it is shown that there exists a point $x$, that is either a bottleneck-point or a dangling node that solves the maxisum problem. This is true because the weighted-sum objective is a piecewise linear, **concave** function on the edges, with break-points only in the edge-bottleneck-points. This corresponds to minimizing the weighted sum where all weights are negative. The objective function is then a piecewise linear, **convex** function with break-points only in the edge-bottleneck-points, see $f^1$ in Figure 1. Note that the optimum need not be unique, it can be a subedge between two (or more) bottleneck-points, or the optimum value may also be obtained on a different edge. It is well-known that the optimum for the minisum problem is found in a node ($f^2$ in Figure 1). The standard way of solving this problem is to sum the rows of the distance matrix $D$ multiplied by the weights. The row with the smallest weighted sum corresponds to the minisum optimum node. For further details see Daskin [4].

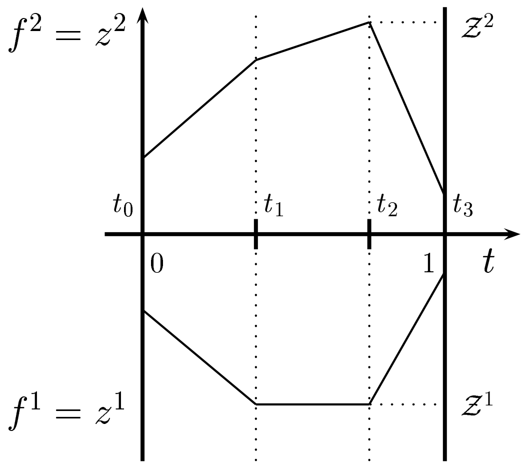

Figure 1: Illustration of the objective functions on an edge.

We denote the set of optimal solutions to a single-objective problem by $\mathcal{X}^q$. The corresponding objective values are denoted by $\mathcal{Z}^q$. Note that these sets of objective-values only contain one value, namely the optimal value, but the notation generalizes to the nondominated set $\mathcal{Z}_{Par}$ defined below.

Solving the $Q$-criteria semi-obnoxious network location problem means finding the set of efficient points. For an introduction to multiple criteria analysis see Steuer [12].
The definition of efficiency is as follows.

**Definition 1** *A solution $x \in G(\mathcal{V},\mathcal{E})$ to (4) is* ***efficient*** *(Pareto optimal) iff there does not exist another solution $\bar{x} \in G(\mathcal{V},\mathcal{E})$ to (4) such that $f^q(\bar{x}) \leq f^q(x)$ $\forall q \in \mathcal{Q}$ and $\exists q \in \mathcal{Q}$ s.t. $f^q(\bar{x}) < f^q(x)$. Otherwise $x$ is* ***inefficient****.*

The set of all efficient/Pareto optimal solutions are denoted by $\mathcal{X}_{Par}$. Efficiency is defined in the decision space. There is a natural counterpart in the criterion space. The criterion space is denoted by $\mathcal{Z}$ and is given by $\mathcal{Z} = \{f(x) \in R^Q | x \in G(\mathcal{V},\mathcal{E})\}$.

**Definition 2** *$f(x) \in \mathcal{Z}$ is a* ***nondominated*** *criterion vector iff $x$ is an efficient solution to (4). Otherwise $f(x)$ is a* ***dominated*** *criterion vector.*

The set of all nondominated criterion vectors are denoted by $\mathcal{Z}_{Par}$ where $\mathcal{Z}_{Par} = f(\mathcal{X}_{Par})$. We use the Pareto optimality notation for both decision and criterion space.
Let $S$ be a subset of $G(\mathcal{V},\mathcal{E})$. We will define the set of **locally efficient** solutions, denoted $\mathcal{X}_{Par}(S)$, to be the solutions that are efficient with respect to all other solutions in the subset $S$. Similarly, $\mathcal{Z}_{Par}(S)$ denotes the set of criterion vectors from $f(S)$ that are **locally nondominated** by any other criterion vector in $f(S)$.

## 2.1 Example

Now we present two small examples to illustrate the structure of the directed and the undirected problem, see Figure 2 and 3. Let the distance matrix $D_{directed}$ be given by

$$D_{directed} = \begin{bmatrix} 0 & 1 & 5 & 4 & 3 & 6 \\ 7 & 0 & 6 & 3 & 10 & 5 \\ 1 & 2 & 0 & 5 & 4 & 7 \\ 4 & 3 & 3 & 0 & 7 & 2 \\ 3 & 4 & 2 & 7 & 0 & 3 \\ 8 & 1 & 7 & 4 & 11 & 0 \end{bmatrix}$$

for the directed network of Figure 2. Let the weights be $w^1 = (-1,-2,-1,-1,-2,-2)$ and $w^2 = (2,1,2,2,2,1)$.
The solution procedure for the directed network in Figure 2 is explained in Section 3.2, and the criterion values are presented in Table 3.
Let the distance matrix $D$ be given by

$$D = \begin{bmatrix} 0 & 1 & 1 & 4 & 3 & 2 \\ 1 & 0 & 2 & 3 & 4 & 1 \\ 1 & 2 & 0 & 3 & 2 & 3 \\ 4 & 3 & 3 & 0 & 5 & 2 \\ 3 & 4 & 2 & 5 & 0 & 3 \\ 2 & 1 & 3 & 2 & 3 & 0 \end{bmatrix}$$

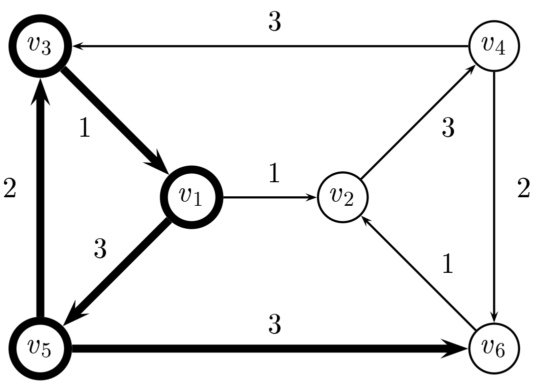

Figure 2: The directed network of Example 2.1. The bold parts constitute the set of efficient points.

for the undirected network of Figure 3. $B$ can be calculated as

$$B = \begin{bmatrix} 2 & 3 & 3 & 6 & 1 & 4 \\ 3 & 2 & 4 & 1 & 6 & 3 \\ 2 & 3 & 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 & 4 & 3 \\ 2 & 3 & 1 & 4 & 1 & 4 \\ 3 & 2 & 4 & 1 & 4 & 1 \\ 4 & 3 & 3 & 4 & 1 & 2 \\ 3 & 2 & 4 & 3 & 2 & 1 \end{bmatrix}.$$

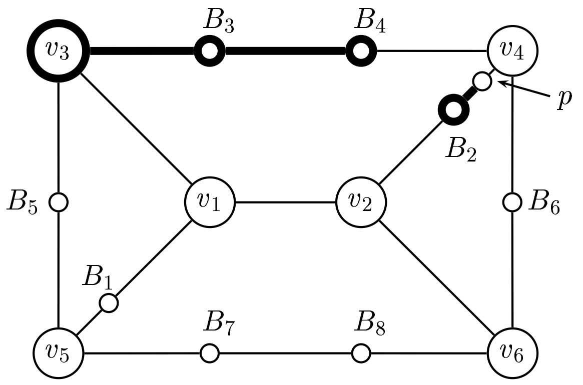

Figure 3: The undirected network of Example 2.1. The bold parts constitute the set of efficient points.

To clarify the solution to the undirected network in Figure 3 we present some function values in Table 2. The solution method for this bicriterion model is described in Section 4. Please note the values of $p$ and $B_4$. This proves that a subedge, not having endpoint at a node or a bottleneck-point, can be efficient. We will refer to this example in Section 3 and 4.

| Point $x$ | $f(x) = (f^1(x), f^2(x))$ |
|---|---|
| $v_1$ | $(-17, 19)$ |
| $v_2$ | $(-16, 21)$ |
| $v_3$ | $(-18, 17)$ |
| $v_4$ | $(-27, 29)$ |
| $v_5$ | $(-24, 27)$ |
| $v_6$ | $(-15, 21)$ |
| $B_1$ | $(-27, 31)$ |
| $B_2$ | $(-30, 33)$ |
| $B_3$ | $(-25, 23)$ |
| $B_4$ | $(-28, 27)$ |
| $B_5$ | $(-23, 29)$ |
| $B_6$ | $(-20, 27)$ |
| $B_7$ | $(-25, 25)$ |
| $B_8$ | $(-23, 27)$ |
| $p$ | $(-28, 30\frac{1}{3})$ |

Table 2: Criterion values for all nodes, all bottleneck-points and point $p$.

From Table 2 we note that bottleneck-point $B_2$ is optimal for the maxisum criterion $(f^1)$ and node $v_3$ is optimal for the minisum criterion $(f^2)$.

## 3 General solution method for the $Q$ criteria case

First, we solve two simple cases of the problem, namely the node problem and the directed case of the absolute location problem. Then we present the absolute location problem on an undirected network.

### 3.1 The easy case: $1/G, G_D/\bullet/d(\mathcal{V}, \mathcal{V})/(Q_1\text{-}\sum_{obnox}, Q_2\text{-}\sum)_{Par}$

In this case the new facility can be placed only at the nodes of the given network, and we can determine the efficient set $\mathcal{X}_{Par} = \mathcal{X}_{Par}(\mathcal{V})$ by the following approach in $O(Qn^2)$ time, given the distance matrix $D$. This approach is presented in [5].

Algorithm 3.1:

```
1. $\mathcal{X}_{Par}(\mathcal{V}) = \mathcal{V}$;
2. for $i = 1$ to $n$ do
        for $j = 1$ to $n$ do
            if $f(v_j)$ dominates $f(v_i)$ then $\mathcal{X}_{Par}(\mathcal{V}) = \mathcal{X}_{Par}(\mathcal{V}) \setminus \{v_i\}$;
```

3. Output $\mathcal{X}_{Par}(\mathcal{V})$;

## 3.2 The easy case: $1/G_D/\bullet/d(\mathcal{V},G)/(Q_1\text{-}\sum_{obnox}, Q_2\text{-}\sum)_{Par}$

For this problem we have to investigate the objective function (2) of the directed case. First, we observe that the objective functions are constant on the interior of the edges. This is true because each term in the sum in (2) consists of a shortest cycle multiplied by a weight.

**Theorem 1** *The directed objective function $f^q(x)$ defined in (2) is* ***constant*** *on $(e,(0,1))$ for all $e \in \mathcal{E}$ and for all $q \in \mathcal{Q}$.*

**Proof** :
Assume $e = (v_i, v_j) \in \mathcal{E}$. In the objective function

$$f^q(x) = \sum_{k=1}^{n} w_k^q \; (d(x,v_k) + d(v_k,x)) \quad q \in \mathcal{Q}$$

we observe that

$$\begin{aligned} d(x,v_k) &= d(x,v_j) + d(v_j,v_k) \quad \forall k \in \mathcal{V} \\ d(v_k,x) &= d(v_k,v_i) + d(v_i,x) \quad \forall k \in \mathcal{V} \end{aligned}$$

on the interior of $e$, and that

$$d(x,v_j) = (1-t)l(e) \quad \text{and} \quad d(v_i,x) = tl(e)$$

for some $t \in (0,1)$. After substituting the distance terms we get

$$f^q(x) = \sum_{k=1}^{n} w_k^q \; (d(v_j,v_k) + d(v_k,v_i) + l(e)) \tag{5}$$

which is independent of $t$, and thus of $x$, on the interior of $e$. ■

Next we use the triangular inequality to prove that the obnoxious objective functions, $q \in \mathcal{Q}_1$, have a higher value at the endnodes of $e$, and that the desirable objective functions, $q \in \mathcal{Q}_2$, have a lower value at the endnodes of $e$. To see this we analyze the objective function (2) once again.

**Theorem 2** *Let $e = (v_i, v_j) \in \mathcal{E}$ be given. The obnoxious objective function values $f^q(v_i)$ and $f^q(v_j)$ are higher than $f^q(x)$, where $x$ is an interior point on $e$ for all $q \in \mathcal{Q}_1$.*

**Proof** :
WLOG we prove that $f^q(x) - f^q(v_i) < 0$. Remember that $w_i^q < 0, \forall i = 1, 2, \ldots, n$ and $q \in \mathcal{Q}_1$. Let us examine the two sums in

$$f^q(x) - f^q(v_i) = \sum_{k=1}^{n} w_k^q \ (d(x, v_k) - d(v_i, v_k)) + \sum_{k=1}^{n} w_k^q \ (d(v_k, x) - d(v_k, v_i)) \tag{6}$$

Starting at the second sum of (6) we use that $d(v_k, x) = d(v_k, v_i) + d(v_i, x)$ to get

$$\sum_{k=1}^{n} w_k^q \ (d(v_k, x) - d(v_k, v_i)) = \sum_{k=1}^{n} w_k^q \ d(v_i, x) = \sum_{k=1}^{n} w_k^q \ tl(e)$$

In the first sum of (6) we use the triangular inequality $d(v_i, v_k) \leq d(v_i, v_j) + d(v_j, v_k)$ and that $d(x, v_k) = d(x, v_j) + d(v_j, v_k)$. Remembering $w_i^q < 0$, we get

$$\begin{aligned}
\sum_{k=1}^{n} w_k^q \ (d(x, v_k) - d(v_i, v_k)) &= \sum_{\substack{k=1 \\ k \neq i}}^{n} w_k^q \ (d(x, v_k) - d(v_i, v_k)) + w_i^q \ (d(x, v_j) + d(v_j, v_i)) \\
&\leq \sum_{\substack{k=1 \\ k \neq i}}^{n} w_k^q \ (d(x, v_j) - d(v_i, v_j)) + w_i^q \ ((1-t)l(e) + d(v_j, v_i)) \\
&= \sum_{\substack{k=1 \\ k \neq i}}^{n} -w_k^q \ tl(e) + w_i^q \ ((1-t)l(e) + d(v_j, v_i)) \\
&= \sum_{k=1}^{n} -w_k^q \ tl(e) + w_i^q \ (l(e) + d(v_j, v_i)).
\end{aligned}$$

Hence,

$$f^q(x) - f^q(v_i) \leq w_i^q \ (l(e) + d(v_j, v_i)) < 0$$

because $w_i^q < 0$. The proof that $f^q(x) - f^q(v_j) < 0$ is similar, apart from the triangular inequality being used in the second sum of (6). ■

**Theorem 3** *Let $e = (v_i, v_j) \in \mathcal{E}$ be given. The desirable objective function values $f^q(v_i)$ and $f^q(v_j)$ are lower than $f^q(x)$, where $x$ is an interior point on $e$ for all $q \in \mathcal{Q}_2$.*

**Proof** :
Similar to the proof of Theorem 2, except $w_i^q > 0, \forall i = 1, 2, \ldots, n$ and $q \in \mathcal{Q}_2$. ■

Using Theorem 3, we observe that the function values on $int(e)$ cannot dominate the function values at the nodes $v_i$ and $v_j$, because the desirable function values at the nodes are lower. Similarly, the function values at the nodes cannot dominate the function value on the interior of $e$, because the obnoxious function value is lower on $int(e)$ by Theorem

2. This observation cannot, however, be used to conclude that nodes and edges cannot dominate each other. The objective function values on edge $e_{12}$ in the directed network in Figure 2 are illustrated in Figure 4.

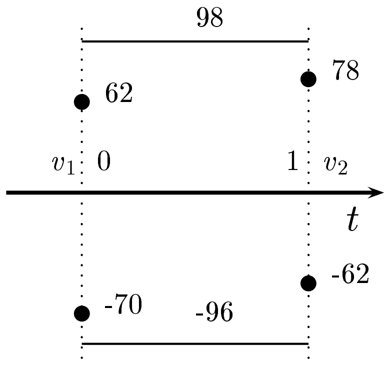

Figure 4: $f((v_1, v_2))$. Notice that $f(v_1)$ dominates $f(v_2)$.

In Algorithm 3.2 we have to compare all nodes and edges, but we only need one vector of function values on each edge, calculated easily by (5).
To present a compact form of the algorithm, we define the $n+m$ points $a_i$ on $G(\mathcal{V}, \mathcal{E})$ as the $n$ nodes and the midpoints on the $m$ edges:

$$\begin{aligned} a_i &= v_i \quad \forall\, i = 1, 2, \ldots, n \\ a_{n+i} &= x_i = (e_i, \frac{1}{2}) \quad \forall\, i = 1, 2, \ldots, m \end{aligned}$$

Algorithm 3.2:

```
1. $\mathcal{X}_{Par} = G(\mathcal{V}, \mathcal{E})$;

2. for $i = 1$ to $n+m$ do

       for $j = 1$ to $n+m$ do

           if $f(a_j)$ dominates $f(a_i)$ then
               if $i \leq n$ then $\mathcal{X}_{Par} = \mathcal{X}_{Par} \setminus \{v_i\}$;
               if $i > n$ then $\mathcal{X}_{Par} = \mathcal{X}_{Par} \setminus (e_{i-n}, (0, 1))$;

3. Output $\mathcal{X}_{Par}$;
```

When we make the pairwise comparison on the $n+m$ points, each taking $O(Q)$ time, we get a complexity bound of $O(Q(n+m)^2)$ time.

For the directed example in Figure 2, using (2) and (5), we get the criterion values of Table 3. The optimal value for the obnoxious function is $-126$ attained on $(v_5, v_6)$ and the optimal desirable function value is 62 attained at $v_1$ and $v_3$. After running Algorithm 3.2 we have determined the efficient nodes and edges as indicated in the table and the figure.

| Point $x$ | $f(x) = (f^1(x), f^2(x))$ | |
|---|---|---|
| $v_1$ | $(-70, 62)$ | Efficient |
| $v_2$ | $(-62, 78)$ | |
| $v_3$ | $(-70, 62)$ | Efficient |
| $v_4$ | $(-68, 72)$ | |
| $v_5$ | $(-82, 80)$ | Efficient |
| $v_6$ | $(-74, 102)$ | |
| $(v_1, v_2)$ | $(-96, 98)$ | |
| $(v_1, v_5)$ | $(-94, 92)$ | Efficient |
| $(v_2, v_4)$ | $(-74, 84)$ | |
| $(v_3, v_1)$ | $(-76, 74)$ | Efficient |
| $(v_4, v_3)$ | $(-96, 98)$ | |
| $(v_4, v_6)$ | $(-98, 120)$ | |
| $(v_5, v_3)$ | $(-106, 98)$ | Efficient |
| $(v_5, v_6)$ | $(-126, 140)$ | Efficient |
| $(v_6, v_2)$ | $(-86, 108)$ | |

Table 3: Criterion values for all nodes and all edges.

## 3.3 Solving $1/G/\bullet/d(\mathcal{V}, G)/(Q_1\text{-}\sum_{obnox}, Q_2\text{-}\sum)_{Par}$

The general solution method consists of pairwise comparison of subedges. The objective functions are all piecewise linear, and the idea is to partition the network into subedges, where the objective functions are linear. The points where the piecewise linear functions change in slope are in fact the bottleneck-points. We then make a pairwise comparison of all these subedges and delete the inefficient parts. The result is the complete set of efficient solutions $\mathcal{X}_{Par}$.

It is important to note that part of a subedge may be efficient, starting at a point that is not a node or an edge-bottleneck-point (see Example 2.1 at point $p$).

For each comparison of two subedges we will construct a linear program to detect inefficient points (segments), that can be solved in linear time by methods found in Megiddo [9].

Let $z^q(t) = f^q(x_t)$, $x_t = (e, t)$. These $Q$ functions are all piecewise linear with the same set of possible breakpoints corresponding to the bottleneck-points. Assume there are $P+1$

breakpoints including the two nodes. We then have $P$ subedges. Let these breakpoints on $(e,t)$ be denoted by $t_j,\ j=0,1,\ldots,P$, $(1 \leq P \leq n-1)$, with $t_0 = v_i$, $t_P = v_j$ and $t_{j-1} < t_j \ \forall\, j = 1,2,\ldots,P$. For $t \in [t_{j-1}, t_j]$, the $z^q(t)$'s are linear functions of the form

$$z^q(t) = m_j^q t + b_j^q \quad \forall\, q = 1,2,\ldots,Q \quad \text{with}$$

$$\begin{array}{lll} m_1^q \leq m_2^q \leq \ldots \leq m_P^q\ , & b_1^q \geq b_2^q \geq \ldots \geq b_P^q & \quad q \in \mathcal{Q}_1 \\ m_1^q \geq m_2^q \geq \ldots \geq m_P^q\ , & b_1^q \leq b_2^q \leq \ldots \leq b_P^q & \quad q \in \mathcal{Q}_2 \end{array}$$

This is illustrated in Figure 1. Let us now compare the subedge A on edge $e_A$, $(e_A, [t_{j-1}, t_j])$ with subedge B on edge $e_B$, $(e_B, [s_{p-1}, s_p])$. A point $(e_A, t) \in (e_A, [t_{j-1}, t_j])$ is dominated by some point $(e_B, s) \in (e_B, [s_{p-1}, s_p])$ if and only if

$$m_p^q s + b_p^q \leq m_j^q t + b_j^q \quad \forall\, q = 1,2,\ldots,Q$$

where at least one inequality is strict. This comparison is illustrated in Figure 5 for two subedges from Example 2.1. Subedge $(B_7, B_8)$ is compared with subedge $(v_5, B_7)$.

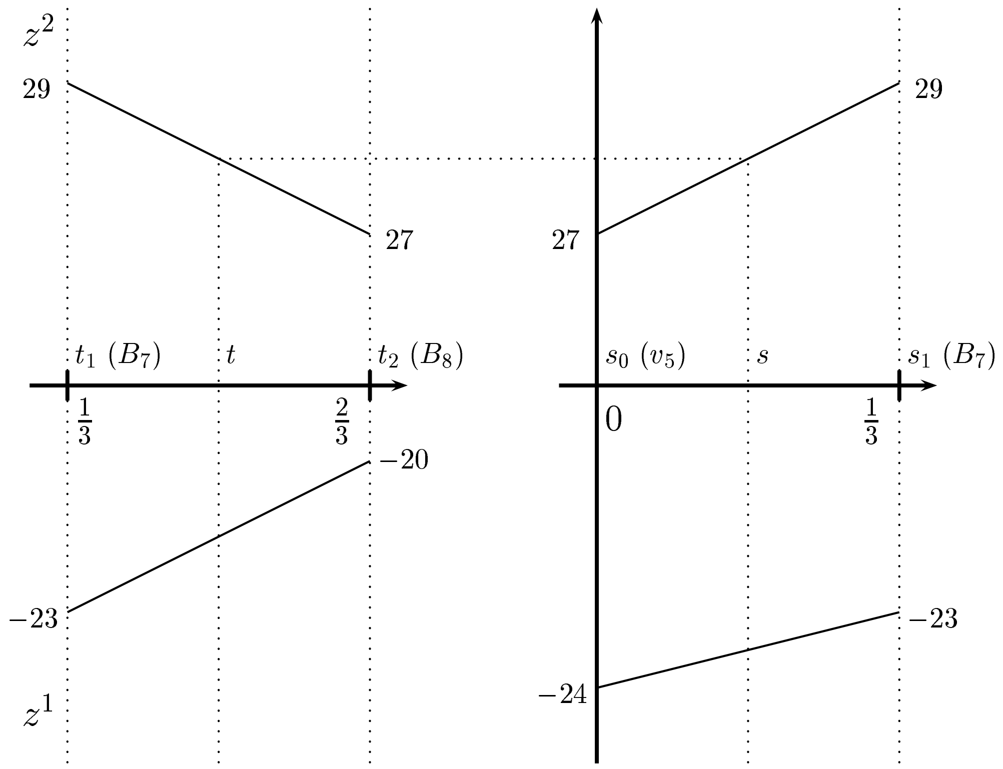

Figure 5: Comparing subedge $(B_7, B_8)$ with subedge $(v_5, B_7)$.

Let us define the set $\mathcal{T}$ where the inequalities hold (for these particular subedges) by

$$\mathcal{T} = \{(s,t) |\ m_j^q t - m_p^q s \geq b_p^q - b_j^q,\ \forall\, q \in \mathcal{Q}\} \cap ([s_{p-1}, s_p] \times [t_{j-1}, t_j])$$

If $\mathcal{T} = \emptyset$, $(e_B, [s_{p-1}, s_p])$ does not contain a point dominating any point in $(e_A, [t_{j-1}, t_j])$. Otherwise $\mathcal{T} \neq \emptyset$ is taken as a feasible solution set of the two 2-variable linear programs:

$$LB = \min\{\ t \mid (s,t) \in \mathcal{T}\} \quad \text{and} \quad UB = \max\{\ t \mid (s,t) \in \mathcal{T}\}$$

Using methods described by Megiddo [9], $LB$ and $UB$ can be calculated in $O(Q)$ time. We now check if we have only **weak dominance**. This means that none of the inequalities need to be strict as required by Definition 1. Note that points with weak dominated objective function values may be efficient. Let $s_{LB}$ and $s_{UB}$ be optimal values of $s$ corresponding to $LB$ and $UB$. These $s$-values are not necessarily unique as illustrated in Figure 6, where $s_{LB}$ can be any point in $[0, \frac{1}{3}]$. In the case where $s_{LB}$ (and/or $s_{UB}$) is not unique ($s_{LB} \in [s_a, s_b]$), we choose $s_{LB} = \frac{1}{2}(s_a + s_b)$ to avoid problems with weak dominance in the subedge endnodes. To check for weak dominance, we examine the subedge endnodes. If $m_p^q s_{LB} + b_p^q = m_j^q LB + b_j^q\ \forall\, q \in \mathcal{Q}$, then $LB$ is only weakly dominated and can therefore still be efficient. Similarly, if $m_p^q s_{UB} + b_p^q = m_j^q UB + b_j^q\ \forall\, q \in \mathcal{Q}$, then $UB$ is only weakly dominated. If both $LB$ and $UB$ are only weakly dominated, the entire subedge $(e_A, [t_{j-1}, t_j])$ is only weakly dominated by $(e_B, [s_{p-1}, s_p])$. This means that all the inequalities in $\mathcal{T}$ are in fact equalities. Otherwise the inefficient part of the subedge is deleted. If both $LB$ and $UB$ are dominated, then

$$(e_A, [t_{j-1}, t_j]) = (e_A, [t_{j-1}, t_j]) \setminus (e_A, [LB, UB])$$

and if, say $LB$ is only weakly dominated, then

$$(e_A, [t_{j-1}, t_j]) = (e_A, [t_{j-1}, t_j]) \setminus (e_A, (LB, UB])$$

This comparison can also be done in linear time. The approach is simplified if one or both subedges consists of a single point $(e_A, t')$ (or $(e_B, s'')$). If $(e_A, [t_{j-1}, t_j]) = (e_A, t') = x$, then $LB = UB = t'$ and

$$\mathcal{T}' = \{s|\ -m_p^q s \geq b_p^q - f^q(x),\ \forall\, q \in \mathcal{Q}\} \cap [s_{p-1}, s_p]$$

If $(e_B, [s_{p-1}, s_p]) = (e_B, s'') = y$, then

$$\mathcal{T}'' = \{t|\ m_j^q t \geq f^q(y) - b_j^q,\ \forall\, q \in \mathcal{Q}\} \cap [t_{j-1}, t_j]$$

and

$$LB = \min\{\ t \mid t \in \mathcal{T}''\} \quad \text{and} \quad UB = \max\{\ t \mid t \in \mathcal{T}''\}$$

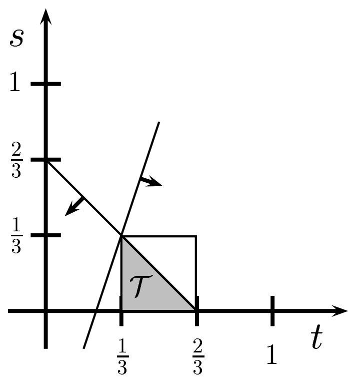

Figure 6: The linear programming constraints for comparing $(B_7, B_8) = (e, [\frac{1}{3}, \frac{2}{3}])$ with $(v_5, B_7) = (e, [0, \frac{1}{3}])$ on edge $(v_5, v_6)$ in Example 2.1. $\mathcal{T}$ is indicated by the shaded area.

This subedge comparison is illustrated in Figure 6, where the subedge $(B_7, B_8) = (e, [\frac{1}{3}, \frac{2}{3}])$ from Example 2.1 is compared with $(v_5, B_7) = (e, [0, \frac{1}{3}])$. Both subedges are on the same edge. Since $\mathcal{T}$ is non-empty, we solve the two programs and find $LB = \frac{1}{3}$ and $UB = \frac{2}{3}$. Both $LB$ and $UB$ are dominated, so the subedge $(B_7, B_8)$ is completely deleted.

Since we are removing a connected piece of $(e_A, [t_{j-1}, t_j])$, three things can happen. First, $(e_A, [t_{j-1}, t_j])$ can be completely deleted if $t_{j-1} = LB$ and $t_j = UB$ are both dominated. Second, a piece of $(e_A, [t_{j-1}, t_j])$ that includes one of the endpoints $t_{j-1}$ or $t_j$ can be deleted, in which case one connected subedge remains, say $(e_A, [t_{j-1}, LB))$ or $(e_A, [t_{j-1}, LB])$. The third case is when an interior part of $(e_A, [t_{j-1}, t_j])$ is deleted, so we end up with the two subedges $(e_A, [t_{j-1}, LB))$ and $(e_A, (UB, t_j])$, possibly including one of the points $LB$ or $UB$. The third case is illustrated in Figure 7 where $UB$ is not deleted, because $z(UB) = z(t_2)$.

In order to complete the comparison, we simply make an ordered subedge comparison. First, we compare $(e_1, [t_0, t_1])$ with all the other subedges, possibly dividing $(e_1, [t_0, t_1])$ into new subedges. Then we compare the second subedge $(e_1, [t_1, t_2])$ with all the remaining subedges. If $(e_1, [t_0, t_1])$ is not completely dominated, we also compare with this subedge. This comparison continues until we have compared the last subedge $(e_m, [s_{P-1}, s_P])$ with all the remaining subedges.

Notice that we can still use the entire subedge $(e_A, [t_{j-1}, t_j])$ to compare with the other subedges, even though a part of it is inefficient. It is only for the set of efficient points $\mathcal{X}_{Par}$, that we have to remember what part of $(e_A, [t_{j-1}, t_j])$ is efficient. But if the whole subedge $(e_A, [t_{j-1}, t_j])$ is inefficient, we should delete it from further consideration, also in the comparison process.

Assume that edge $e_i \in \mathcal{E}$ is divided into $P_i$ bottleneck-point subedges.

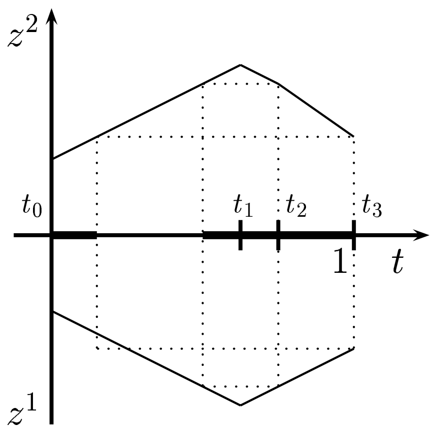

Figure 7: There are 4 breakpoints ($P = 3$) and 4 efficient subedges. Locally Pareto optimal subedges are indicated in bold on the $t$ axes. Note that $(e, [t_2, t_3])$ dominates an interior part of $(e, [t_0, t_1])$.

Algorithm 3.3:

```
1. $\mathcal{X}_{Par} = G(\mathcal{V}, \mathcal{E})$;

2. for $i = 1$ to $m$ do

   for $x = 1$ to $P_i$ do

        for $j = 1$ to $m$ do

        for $y = 1$ to $P_j$ do

            compare $(e_i, [t_{x-1}, t_x])$ with $(e_j, [t_{y-1}, t_y])$
                $\mathcal{X}_{Par}$ unchanged if no points are dominated
                $\mathcal{X}_{Par} = \mathcal{X}_{Par} \setminus (e_i, [LB, UB])$ if $LB$ and $UB$ are dominated;
                $\mathcal{X}_{Par} = \mathcal{X}_{Par} \setminus (e_i, (LB, UB])$ if only $UB$ is dominated;
                $\mathcal{X}_{Par} = \mathcal{X}_{Par} \setminus (e_i, [LB, UB))$ if only $LB$ is dominated;

3. Output $\mathcal{X}_{Par}$;
```

This general algorithm has been implemented, and computational results are reported in Section 6. Each of the $m$ edges may consist of up to $n - 1$ bottleneck-point subedges, giving at most $O(mn)$ subedges. If we make the global pairwise comparison on the $O(mn)$ bottleneck-point subedges, each taking $O(Q)$ time, we get a complexity bound of $O(Qm^2n^2)$ time. This is also the bound for the case where $\mathcal{Q} = \mathcal{Q}_2$ found in Hamacher *et al.* [5].

## 4 Bicriteria case

In the case where we only have two criteria, we may use the image of the network mapped into criterion space $\mathcal{Z}$ to solve the problem faster. This is done by calculating the lower envelope, see Hershberger [8]. This can be done in $O(p \ log \ p)$ time, where $p$ is the number of line-segments. There are three different situations. $\mathcal{Q}_1 = \emptyset$ denoted min-min $(1/G/\bullet/d(\mathcal{V},G)/2\text{-}(\sum)_{Par})$, $|\mathcal{Q}_1| = |\mathcal{Q}_2| = 1$ denoted max-min $(1/G/\bullet/d(\mathcal{V},G)/(\sum_{obnox},\sum)_{Par})$ and $\mathcal{Q}_2 = \emptyset$ denoted max-max $(1/G/\bullet/d(\mathcal{V},G)/2\text{-}(\sum_{obnox})_{Par})$. All three cases are solved by the same method.

### 4.1 Direct mapping of the network into criterion space

This procedure is best described by an example, so we present the undirected network of Example 2.1 in criterion space.

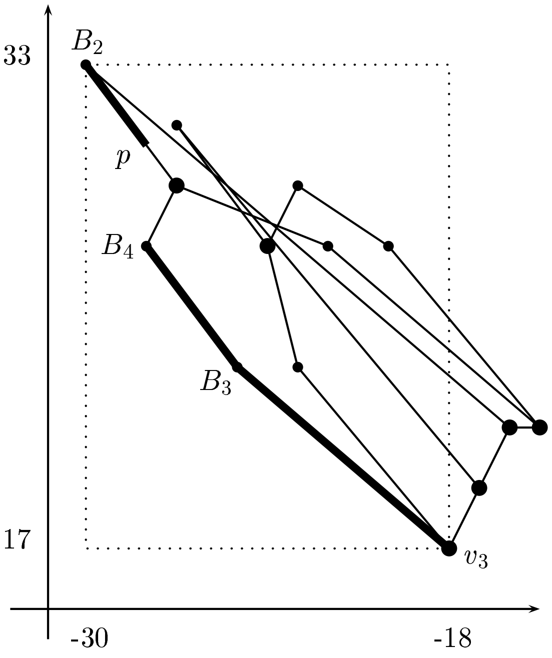

Figure 8: Mapping of the undirected network from Example 2.1 into criterion space. The bold parts constitute the set of nondominated points.

Since we want to find the set of efficient solutions $\mathcal{X}_{Par}$, we are only interested in values between the two extreme optimal solutions, namely $\mathcal{Z}^1$ and $\mathcal{Z}^2$. We therefore investigate the region $[f^1_{\mathcal{Z}^1}, f^1_{\mathcal{Z}^2}] \times [f^2_{\mathcal{Z}^2}, f^2_{\mathcal{Z}^1}]$, denoted $S$.

We have to make sure that the slope of the envelope is decreasing, when the $f^1$-values increase, to ensure that there are no dominated points on the envelope. This can be done by adding horizontal lines to all nodes and bottleneck-points in $S$, with the horizontal lines ending at $f^1_{\mathcal{Z}^2}$. This will at worst double the number of line-segments in the region $S$. Alternatively we could add the horizontal line to bottleneck-points that does not have a subedge with negative slope leaving the point. In the example of Figure 8 none of the points in $S$ would need the horizontal line added. After the lower envelope is determined, we delete the horizontal parts (if any), because the points on a horizontal line are dominated by the left endpoint. The result is $\mathcal{Z}_{Par}$. The set of efficient solutions are then given by $\mathcal{X}_{Par} = f^{-1}(\mathcal{Z}_{Par})$. The efficient set corresponding to the nondominated set of Figure 8 is indicated in Figure 3.

We have the same complexity bound on the lower envelope calculation, as in Hamacher *et al.* [5], namely $O(mn\ log(mn))$. This bound can be rewritten by examining the *log* term and using the fact that $m$ is at most $n^2$ for dense graphs. We therefore get the bound of $O(mn\ log\ n)$ time for the envelope calculation.

## 5 Center objectives - $1/G/\bullet/d(\mathcal{V},G)/(Q_3\text{-max}_{obnox}, Q_4\text{-max})_{Par}$

We now investigate the maximin and minimax objectives. These criterion functions are often referred to as the **weighted anti-center** and **center** of a network. The problem is formulated as follows:

$$\begin{array}{lll} \max & f^q(x) = \min_i\ w_i^q \cdot d(x,v_i) & q \in \mathcal{Q}_3 \\ \min & f^q(x) = \max_i\ w_i^q \cdot d(x,v_i) & q \in \mathcal{Q}_4 \\ \text{s.t.} & & \\ & x \in G(\mathcal{V},\mathcal{E}) & \end{array} \tag{7}$$

$\mathcal{Q}_3$ is the set of obnoxious objective functions, and $\mathcal{Q}_4$ is the set of attraction objective functions. At most one of the sets are allowed to be empty.

For simplicity we again multiply all objective functions in $\mathcal{Q}_3$ by $-1$ in order to minimize in stead of maximize. This gives the following formulation:

$$\begin{array}{lll} \min & f^q(x) = \max_i\ -w_i^q \cdot d(x,v_i) & q \in \mathcal{Q}_3 \\ \min & f^q(x) = \max_i\ w_i^q \cdot d(x,v_i) & q \in \mathcal{Q}_4 \\ \text{s.t.} & & \\ & x \in G(\mathcal{V},\mathcal{E}) & \end{array} \tag{8}$$

We notice that the objective functions are again piecewise linear, but the breakpoints are now weight dependent, see Figure 9. If we find these breakpoints, we can apply the same solution approach as in Section 3.3 for the multicriteria case, and the envelope method of Section 4 for the bicriteria case. When we only have center objective functions, the new breakpoints are the only ones needed. If we combine these objectives with the sum objectives, we may get a lot more breakpoints, because the bottleneck-point breakpoints are also needed.

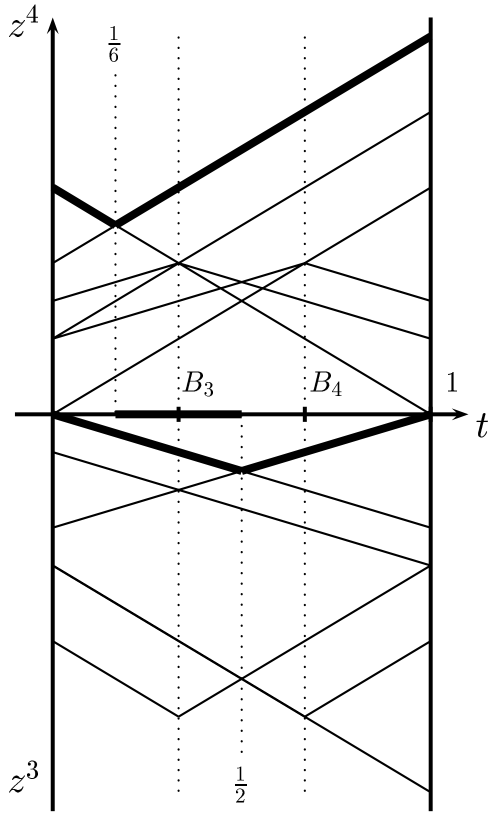

Figure 9: $f((v_3, v_4))$. There are two edge-bottleneck-points on this edge, and we find two new breakpoints. $f^3$ and $f^4$ are indicated with a bold lines.

In the following we expand Example 2.1 to illustrate what the center objectives look like. In Figure 9 we illustrate the locally efficient points on $(v_3, v_4)$, where $w^3 = w^1$ and $w^4 = w^2$, as $\mathcal{X}_{Par}((v_3, v_4)) = ((v_3, v_4), [\frac{1}{6}, \frac{1}{2}])$.

In this example both objective functions turn out to be convex, but this is not the general case. The center objective is known to be neither convex nor concave. But the anti-center (maximin) objective is a **concave** function (so in problem (8) it is convex). This is true, because it is the minimum of piecewise linear concave functions. When we convert the problem to a minimax with negative weights, we get a piecewise linear **convex** function. This fact leaves little hope for finding an improved approach for this general case where we combine both sum and center objectives. After having investigated the different problems in turn, we can conclude that the method described in Section 3.3 works for any piecewise linear objective functions.

## 6 Computational results

In this section we present computational results from an implementation of Algorithm 3.3. We have not used the methods of Megiddo [9] in this implementation to solve the small LP's. Instead, we have used CPLEX 6.6. The code is programmed in C++ and the tests are run on a 700 MHz Linux PC.
We have used random networks of varying size generated using NETMAKER. A description of NETMAKER can be found in Skriver and Andersen [11]. All the random networks have a fixed number of nodes and a random number of edges with mean 4 times the number of nodes, i.e. a 50 node network has approximately 200 edges. Each network contains a random Hamiltonian cycle, and for each node three random edges are generated. The weights are generated negatively correlated. If one weight is in the integer interval from 1 to 33, the other is in the integer interval of 67 to 100. The same holds for the negative weights for the obnoxious objective functions (except for the sign). In each group we have used 10 random networks, and the mean is reported in the following tables.

First, we examine some semi-obnoxious bicriterion networks, having one push objective and one pull objective. The results are presented in Table 4. It appears that the number of subedges grows a little less than squared the number of nodes. The number of subedges is important, because in worst case we have to make a pairwise comparison of all these subedges, (# Subedges)$^2$. The number of actual comparisons made is presented in the table, and the percentage of actual comparisons to the worst case is also presented. It is important to note that this percentage decreases as the networks increase in size.

| # Nodes | 50 | 100 | 150 | 200 | 250 |
|---|---|---|---|---|---|
| CPU-time | 40.96 | 229.54 | 774.64 | 1505.42 | 3326.37 |
| # Subedges | 3033.6 | 9411.5 | 18525.2 | 28368.1 | 39540.2 |
| # Subedge comparisons (in millions) | 0.358 | 1.770 | 5.138 | 8.655 | 16.531 |
| # Efficient subedges | 96.2 | 155.3 | 175.7 | 222.5 | 264.5 |
| % Efficient subedges | 3 | 1.6 | 0.95 | 0.78 | 0.67 |
| % Comparisons | 4.00 | 2.02 | 1.50 | 1.08 | 1.05 |
| # Comparisons per sec | 8733 | 7709 | 6633 | 5749 | 4970 |

Table 4: Semi-obnoxious bicriterion results, 1 push - 1 pull objective.

The number of efficient subedges is also presented in Table 4, and this number seems to grow linearly with the number of nodes. This number is in fact higher than the number of actual efficient subedges, because more subedges may contain the same efficient point, when this point is a node. If a node is efficient, all the subedges connected to this node contain some efficient points (perhaps only the node which is the endpoint of the subedge). The last row in Table 4 are the numbers of comparisons made per CPU-second. Assuming that CPLEX performs independently of the number of problems it has to solve, this decrease indicates that the large problems require a lot more storage of data, and accessing this data takes an increasing amount of time.

Next we examine the effect of having more objectives. These results are all computed on networks with 50 nodes. We reuse the results of the bicriterion (1-1) networks of Table 4, examine two types of three objective problems and one type of four objective problems. The three objective networks are generated with both 1 obnoxious and 2 desirable objectives (1-2), and 2 obnoxious and 1 desirable objectives (2-1). The four objective networks are all with 2 obnoxious and 2 desirable objective functions (2-2). The results are presented in Table 5.

As expected both the number of subedges containing efficient points and the CPU-time increase rapidly when more negatively correlated objective functions are added. With four objectives more than 75 % of the subedges contain efficient points. It is seen that the CPU-time for these instances is almost proportional to the number of subedge comparisons, since the data size of the instances is approximately the same (last line in Table 5).

| # Objectives | 1-1 | 1-2 | 2-1 | 2-2 |
|---|---|---|---|---|
| CPU-time | 40.96 | 123.05 | 105.49 | 870.57 |
| # Subedges | 3033.6 | 3293.1 | 3158.8 | 2853.6 |
| # Subedge comparisons (in millions) | 0.358 | 1.019 | 0.914 | 6.128 |
| # Efficient subedges | 96.2 | 359.1 | 357.9 | 2237.7 |
| % Efficient subedges | 3 | 11 | 11 | 78 |
| % Comparisons | 4.00 | 9.47 | 9.53 | 75.46 |
| # Comparisons per sec | 8733 | 8349 | 8720 | 7077 |

Table 5: The effect of having more objectives. All networks have 50 nodes.

Finally, we conclude that the computational results are constructive in the sence that rather large problems can be solved within a reasonable amount of time. Since location problems are usually not of the type you have to resolve often, a longer CPU-time is acceptable.

The most encouraging result being that for bicriterion networks with objective functions in almost opposite directions, a very small proportion of the networks is efficient. This indicates that this model is in fact an aid for the decision-maker, since a large part of the network can be omitted from further consideration. On the efficient parts of the network, the trade-off between the two objectives can then be revealed.

As a final comment, we note that with negatively correlated objectives, at most three objective functions should be considered. Otherwise the results are inconclusive, since a large proportion of the network will be efficient.

## 7 Concluding remarks

In this paper we have set up a multicriterion network location model for locating a (semi) obnoxious facility. We have proposed an efficient solution algorithm based on ideas from the multicriterion median network location problem presented in Hamacher *et al.* [5].
In the bicriterion case we have found an improved method, but this method has not been implemented. The general method presented in this paper works for all piecewise linear objective functions, and has been implemented in C++ using CPLEX as a solver. The computational results show that networks of realistic size can be solved in a reasonable amount of time. We thus conclude that this model is a good tool for general network location decisions.

## References

[1] J. Brimberg and H. Juel. A bicriteria model for locating a semi-desirable facility in the plane. *European Journal of Operational Research*, 106:144–151, 1998.

[2] E. Carrizosa, E. Conde, and D. Romero-Morales. Location of a semiobnoxious facility. A biobjective approach. In 1996 Torremolinos, editor, *Advances in multiple objective and goal programming*, pages 338–346. Springer-Verlag, Berlin-Heidelberg, 1997.

[3] R.L. Church and R.S. Garfinkel. Locating an obnoxious facility on a network. *Transportation Science*, 12:107–118, 1978.

[4] M.S. Daskin. *Network and Discrete Location.* Wiley, New York, 1995.

[5] H.W. Hamacher, M. Labbe, and S. Nickel. Multicriteria network location problems with sum objectives. *Networks*, 33:79–92, 1999.

[6] H.W. Hamacher and S. Nickel. Multicriteria planar location problems. *European Journal of Operational Research*, 94:66–86, 1996.

[7] P. Hansen, M. Labbè, and J.F. Thisse. From the median to the generalized center. *RAIRO Rech. Opér.*, 25:73–86, 1991.

[8] J. Hershberger. Finding the upper envelope of $n$ line segments in $O(n\ log\ n)$ time. *Info Process Lett*, 33:169–174, 1989.

[9] N. Megiddo. Linear-time algorithms for linear programming in $R^3$ and related problems. *SIAM J. Comput*, 12:759–776, 1983.

[10] A.J.V. Skriver and K.A. Andersen. A bicriterion semi-obnoxious facility location model solved by an $\epsilon$-approximation. Technical Report 2000-1, Department of Operations Research, University of Aarhus, Denmark, 2000.

[11] A.J.V. Skriver and K.A. Andersen. A label correcting approach for solving bicriterion shortest path problems. *Computers and Operations Research*, 27:507–524, 2000.

[12] R.E. Steuer. *Multiple criteria optimization: Theory, Computation, and Application.* Wiley, New York, 1986.

[13] K. Thulasiraman and M.N.S. Swamy. *Graphs: Theory and Algorithms.* Wiley, New York, 1992.