A Comparison of Solution Strategies for Biobjective Shortest Path Problems

Andrea Raith
Department of Engineering Science
The University of Auckland, New Zealand
email: a.raith@auckland.ac.nz

Matthias Ehrgott
Department of Engineering Science
The University of Auckland, New Zealand
email: m.ehrgott@auckland.ac.nz
and
Laboratoire d'Informatique de Nantes Atlantique
Université de Nantes, France
email: matthias.ehrgott@univ-nantes.fr

February 16, 2007

Abstract

We consider the biobjective shortest path (BSP) problem as the natural extension of the single objective shortest path problem. BSP problems arise in various applications where networks usually consist of large numbers of nodes and arcs. Since obtaining the set of efficient solutions to a BSP problem is more difficult (i.e. $\mathcal{NP}$-hard and intractable) than solving the corresponding single objective problem there is a need for fast solution techniques. Our aim is to compare different strategies for solving the BSP problem. We consider a standard label correcting method, a purely enumerative near shortest path approach, and the two phase method, investigating different approaches to solving problems arising in phase 1 and phase 2. In particular, we propose to combine the two phase method with ranking in phase 2. In order to compare the different approaches, we investigate their performance on three different types of networks. We employ grid networks and random networks, as is generally done in the literature. Furthermore, road networks are utilized to compare performance on networks with a structure that is more likely to actually arise in applications.

Keywords: Biobjective shortest path problem, two phase method, label correcting algorithm, near shortest path algorithm.

1 Introduction

Shortest path problems have been studied intensively in the literature (e.g. Gallo and Pallotino 1988). The single objective shortest path problem is most widely studied. However, it is often not sufficient to restrict oneself to one objective. Applications often indicate the necessity of taking two or more objectives into account, resulting in biobjective or multiple objective shortest path problems. Examples include transportation problems (Pallottino and Scutellà 1998), routing in railway networks (Müller-Hannemann and Weihe 2006), and problems in satellite scheduling (Gabrel and Vanderpooten 2002).

We consider the biobjective shortest path (BSP) problem as the natural extension of the single objective case. BSP belongs to the class of multiple objective combinatorial optimization (MOCO) problems. In BSP the aim is to find efficient solutions. BSP is an $\mathcal{NP}$-hard problem (Serafini 1986) and it also is intractable, i.e. the number of efficient solutions may be exponential in the number of nodes (Hansen 1980). Despite this fact, Müller-Hannemann and Weihe (2006) suggest that in practical applications with certain characteristics we can expect to find a reasonably small number of efficient solutions.

Only a subset of the image of the set of efficient solutions of a BSP problem is situated on the boundary of the convex hull of the feasible set Z in objective space, the so-called supported efficient solutions. The images of the non-supported efficient solutions are located in the interior of $conv(Z)$. Supported efficient solutions of the BSP problem can be obtained by solving problems with a weighted sum, i.e. single, objective. There is no known characterization of non-supported efficient solutions that leads to a polynomial time algorithm for their computation.

There are two main approaches to solving BSP problems. On the one hand there are enumerative approaches such as label correcting (Skriver and Andersen 2000; Brumbaugh-Smith and Shier 1989) and label setting (Martins 1984; Tung and Chew 1988, 1992) or ranking methods (Clímaco and Martins 1982).

Label correcting methods work similarly to their single objective (e.g. Bertsekas 1998) counterparts. In BSP problems a node can have several labels, which do not dominate one another. The set of efficient solutions of the BSP problem corresponds to all labels at the target node after a labeling algorithm finishes. In label correcting and label setting methods, either one label at a certain node is extended by all arcs out of that node (label-selection) or all labels at a node are extended simultaneously (node-selection).

Ranking methods are single objective k-shortest path methods. Starting with the optimal value for one objective, the second-best solution, the third-best solution etc. is obtained until the k-best solution is reached. For BSP, the process continues until it is guaranteed that all non-dominated points have been found. K-shortest path methods have been found not to be competitive with label correcting methods (Huarng et al. 1996; Skriver 2000). Instead, we investigate the application of a near-shortest path method by Carlyle and Wood (2005), which the authors successfully apply to the k-shortest path problem.

Another approach, the two phase method, is taking advantage of the problem structure (Mote et al. 1991; Ulungu and Teghem 1995). In the first phase,

the extreme supported efficient solutions (efficient solutions which define extreme points of $conv(Z)$) are computed. In the second phase the remaining efficient solutions are computed with one of the enumerative approaches mentioned before. The enumerative methods can be employed in a very effective way as enumeration can be restricted to small areas of the objective space.

We present well known strategies to solve the BSP problem and introduce the two phase method with near shortest path ranking by an adaptation of a near shortest path approach in phase 2. Our aim is to compare the performance of the different solution approaches. We investigate performance on two different artificial network structures and also on road networks, to include some real world network structures into our considerations as is done by Zahn and Noon (1998) for single criterion shortest path problems. This comparison is in contrast to earlier studies, where a single type of networks has been used.

In Section 2 basic concepts of BSP problems are introduced. Recent literature is discussed in Section 3. In Section 4 we present the different algorithms we use to solve BSP, that is label correcting, near shortest path and the two phase method. Finally, numerical results are presented in Section 5.

2 Biobjective Shortest Path Problems

In this section, terminology and basic theory of biobjective shortest path problems is introduced following the notation of Przybylski et al. (2007).

Let $G = (N, A)$ be a *directed network* with a set of nodes $N = \{1, \ldots, n\}$ and a set of arcs $A = \{(i_1, j_1), \ldots, (i_m, j_m)\} \subseteq N \times N$. Two positive costs $c_{ij} = (c^1_{ij}, c^2_{ij}) \in \mathbb{N} \times \mathbb{N}$ are associated with each arc $(i, j) \in A$. In a road network, for example, the costs c^1_{ij} and c^2_{ij} could represent time and distance for traversing arc (i, j), respectively.

A *path* in G from node $i_0 \in N$ to node $i_l \in N$ is a sequence $\{(i_0, i_1), (i_1, i_2), \ldots, (i_{l-1}, i_l)\}$ of arcs in A. The *biobjective shortest path problem* (BSP) with *source node* $s \in N$ and *target node* $t \in N$ can be formulated as a network flow problem:

$$\min \qquad z(x) = \begin{cases} z_1(x) = \sum_{(i,j)\in A} c^1_{ij} x_{ij} \\ z_2(x) = \sum_{(i,j)\in A} c^2_{ij} x_{ij} \end{cases} \tag{1}$$

$$\text{s.t.} \qquad \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \neq s, t \\ -1 & \text{if } i = t \end{cases} \tag{2}$$

$$x_{ij} \in \{0, 1\}, \text{ for all } (i, j) \in A. \tag{3}$$

Here x is a vector of flows on the arcs and the constraints (2) represent flow balance at the different nodes. A balance of 1, -1, and 0 indicates that there exists a surplus of one unit of flow, a demand of one unit of flow, or neither of the two, respectively. The model ensures that one unit of flow is transported through the network from s to t. The arcs with flow value 1 form a path from s to t. The *feasible set* X is described by constraints (2) and (3) and its image under the objective function is $Z := z(X)$.

In the remainder of this paper we use the following orders on $\mathbb{R}^2$:

$$\begin{aligned} y^1 \leqq y^2 &\Leftrightarrow y_k^1 \leqq y_k^2 \; k = 1,2 \\ y^1 \leq y^2 &\Leftrightarrow y_k^1 \leqq y_k^2 \; k = 1,2; \; y^1 \neq y^2 \\ y^1 < y^2 &\Leftrightarrow y_k^1 < y_k^2 \; k = 1,2. \end{aligned}$$

We are seeking those feasible solutions that do not allow to improve one component of the objective vector $z(x)$ without deteriorating the other one.

Definition 1 *A feasible solution $\hat{x} \in X$ is called* efficient *if there does not exist any $x' \in X$ with $(z_1(x'), z_2(x')) \leq (z_1(\hat{x}), z_2(\hat{x}))$. The image $z(\hat{x}) = (z_1(\hat{x}), z_2(\hat{x}))$ of $\hat{x}$ is called* non-dominated. *Let X_E denote the* set of all efficient solutions *and let Z_N denote the* set of all non-dominated points. *We distinguish two different types of efficient solutions.*

- supported *efficient solutions are those efficient solutions that can be obtained as optimal solutions to a (single objective) weighted sum problem*

$$\min_{x \in X} \lambda^1 z_1(x) + \lambda^2 z_2(x) \tag{4}$$

  for some $\lambda^1 > 0, \lambda^2 > 0$. The set of all supported efficient solutions is denoted by X_{SE}, its non-dominated image Z_{SN}. The supported non-dominated points lie on the convex hull $conv(Z)$ of the feasible set in objective space.

- *The remaining efficient solutions in $X_{NE} := X_E \backslash X_{SE}$ are called* non-supported *efficient solutions. They cannot be obtained as solutions of a weighted sum problem as their image lies in the interior of $conv(Z)$. The set of non-supported non-dominated points is denoted by Z_{NN}.*

The two objective functions z_1 and z_2 do generally not attain their individual optima for the same values of $\hat{x}$. We will assume in the following that there exists no $\hat{x}$ such that $\hat{x} \in \text{argmin}\{z_1\}$ and $\hat{x} \in \text{argmin}\{z_2\}$ for a problem of the form (1) - (3).

Definition 2 *Two feasible solutions x and x' are called* equivalent *if $z(x) = z(x')$. A* complete set *X_E is a set of efficient solutions such that all $x \in X \backslash X_E$ are either dominated or equivalent to at least one $x \in X_E$.*

We will only consider solution approaches that compute a complete set X_E.

Another notion of optimality that is used in the context of biobjective optimization is *lexicographic minimization*. Here, we choose among all optimal feasible solutions for the *preferred* component k of the objective vector one that is optimal for the other component l.

Definition 3 *Let $k \in \{1,2\}$ and $l \in \{1,2\} \backslash \{k\}$. Then $z(\hat{x}) <_{lex(k,l)} z(x')$ if either $z_k(\hat{x}) < z_k(x')$ or both $z_k(\hat{x}) = z_k(x')$ and $z_l(\hat{x}) < z_l(x')$. We call $\hat{x}$ a $lex(k,l)$-best solution if $z(\hat{x}) \leqq_{lex(k,l)} z(x)$ for all $x \in X$. Let $x_{lex(k,l)}$ denote a $lex(k,l)$-best solution.*

When solving a single objective version of the BSP problem (by either dropping one objective or by using a weighted sum objective) with the network simplex algorithm (e.g. Helgason and Kennington 1995), the fomulation (1) - (3) is not favourable. Problems arise as the network simplex method performs many basis exchanges without an actual flow change because the flow on all basic arcs that are not part of the actual path from s to t is zero. If a basis exchange involves only those arcs, there is no flow change at all. To avoid this situation we use another formulation, the biobjective shortest path tree (BSPT) problem. This formulation is also used by Mote et al. (1991):

$$\min \qquad z(x) = \begin{cases} z_1(x) = \sum_{(i,j)\in A} c_{ij}^1 x_{ij} \\ z_2(x) = \sum_{(i,j)\in A} c_{ij}^2 x_{ij} \end{cases} \tag{5}$$

$$\text{s.t.} \qquad \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} n-1 & \text{if } i = s \\ -1 & \text{if } i \neq s \end{cases} \tag{6}$$

$$x_{ij} \geq 0 \text{ and integer for all } (i,j) \in A. \tag{7}$$

By modifying the constraint set of (BSP), we now state the problem of finding the shortest path from source node s to all other nodes, resulting in nonzero flow on all basic arcs. Although not every basis exchange leads to a change of the shortest s-t path, it does lead to some change in the shortest path tree rooted at s. This approach ensures a flow change whenever the basis changes.

3 Literature

The most recent survey on BSP problems is by Skriver (2000). The surveys on MOCO problems by Ehrgott and Gandibleux (2000) and Ehrgott and Gandibleux (2002) both include a section about shortest path problems. In the following we mention literature that was to our knowledge not yet covered in a survey. We focus on exact methods.

Martins and Dos Santos (2000) discuss the labeling algorithm for the multiobjective shortest path problem with loops (MSPL). They prove boundedness and finiteness results for the MSPL problem and also correctness of the label setting and label correcting approach. They present a generic labeling algorithm with label selection. They also propose a label setting algorithm based on node selection for acyclic networks, taking advantage of the fact that acyclic networks can be put in topological order. Martins and Dos Santos introduce a new approach to complexity analysis, as multi-objective shortest path problems are intractable in general. They consider the number of *dominated* paths that are generated by an algorithm in the worst case.

Guerriero and Musmanno (2001) investigate label correcting and label setting methods for the multicriteria shortest path tree problem. They propose several strategies for label-selection and node-selection. Computational results are presented for two different classes of test problems. There are problem

instances where label-selection is superior and others where node-selection is superior.

Gabrel and Vanderpooten (2002) describe the application of a multiple objective shortest path label setting procedure for the daily scheduling of earth observing satellites that take photos of the surface of the earth. They formulate the problem as a shortest path problem with three objectives (taking as many photos as possible, fulfilling as many priority requests as possible and minimizing equipment use). The nodes in the network are in topological order, which facilitates label setting. The authors also present an interactive procedure to select one of the enumerated paths.

Sastry et al. (2003) propose several algorithms for multi-objective shortest path problems with positive and negative arc costs. They detect negative cycles by a repeated application (at most once for every objective) of some single objective shortest path algorithm that can detect negative cycles. For networks without negative cycle, they propose a label correcting multiobjective shortest path algorithm with node-selection similar to the one presented by Brumbaugh-Smith and Shier (1989). Sastry et al. also propose two other label correcting approaches. They are both variations to the approach by Corley and Moon (1985). In each iteration of the algorithm, the labels at each node are updated from all predecessor nodes. The algorithm stops when either none of the label sets is changed in an iteration or when after n iterations the existence of a negative cycle is asserted. In each iteration nodes are chosen randomly by Sastry et al. whereas Corley and Moon choose nodes in order of their indices $1, 2, \ldots, n$. The other variation by Sastry et al. is to change the manner in which label sets are updated, the approach is similar to Yen (1970). Each iteration is split into two phases now. In the first phase, nodes are updated by labels at nodes with smaller index than the current node only, in the second phase nodes are updated by labels at nodes with bigger index. They mention that their first algorithm performs best in practical tests.

Müller-Hannemann and Weihe (2006) investigate the cardinality of the set of efficient solutions that arises in practical applications. They examine the characteristics of shortest path problems in train networks with two and three objectives. They relate network and problem characteristics to the actual number of efficient solutions. They find that this number is very low despite the fact that biobjective shortest path problems are in general intractable.

There are recent heuristic approaches to solving the MSP problem, for instance the following two: Sastry et al. (2005) present an approach using a k shortest path method and a weighted sum objective function to compute some efficient solutions. Sonnier et al. (2006) obtain a subset of the efficient set of an MSP problem with m criteria by solving p problems with only $p-1$ criteria and then merging the efficient solutions.

A summary of approaches found in the literature is given in Table 1.

Table 1: Literature on the exact solution of BSP/MSP problems.

Reference	Problem	Solution approach
Hansen (1980)	BSP	Label setting
Clímaco and Martins (1982)	BSP	Ranking
Martins (1984)	MSP	Label setting
Corley and Moon (1985)	MSP	Label correcting
Hartley (1985)	MSP	Label correcting (Dyn. progr.)
Henig (1985)	BSP	Label correcting (Dyn. progr.)
Brumbaugh-Smith and Shier (1989)	BSP	Label correcting, node-selection
Mote et al. (1991)	BSP	Two phase method
Tung and Chew (1988)	BSP	Label setting, label-selection
Tung and Chew (1992)	MSP	Label setting, label-selection
Huarng et al. (1996)	BSP	computational comparison
Skriver and Andersen (2000)	BSP	Label correcting, node-selection
Martins and Dos Santos (2000)	MSP	Label setting and correcting node- and label-selection
Guerriero and Musmanno (2001)	MSP	Label setting and correcting node- and label-selection
Sastry et al. (2003)	MSP	Label correcting, node-selection

4 Solution Methods

We investigate different methods to solve BSP. Three main approaches are identified. One is a biobjective label correcting algorithm with node-selection, which is identified as the most successful approach to solve BSP problems by Skriver and Andersen (2000). Another one is the adaptation of a near shortest path procedure by Carlyle and Wood (2005) to BSP. We also investigate the two phase method for BSP by Mote et al. (1991). A formulation of the two phase method for general MOCO problems can be found in Ulungu and Teghem (1995). We compare different strategies that can be used for initialization and in phases 1 and 2.

4.1 Label Correcting

A biobjective label correcting method is a straightforward extension of the single objective version. The main difference for two or more objectives is that there may be several labels at a node, each corresponding to one path. The labels at one node do not dominate one another.

Approaches to label correcting differ in whether they employ label-selection or node-selection. Label-selection means that all labels are treated separately. A label l at some node i is extended by all arcs (i, j) with tail node i. The extended label $l + c_{ij}$ is inserted into the label set at node j if it is not dominated. The new label may dominate other labels at node j which are deleted. Also, a non-dominated extended label $l + c_{ij}$ at j has to be reconsidered in a later iteration. Node selection means that a node i is selected and *all* its labels are extended via all outgoing arcs. We explain node-selection together with Algorithm 1.

Despite the results of Guerriero and Musmanno (2001), we opt for node-selection, the approach also chosen by Skriver and Andersen (2000) (see also

Brumbaugh-Smith and Shier 1989), which will be described below, refer to Algorithm 1.

Initially, the only labeled node is the source node s with its label set $Labels(s) = \{(0,0)\}$. All labels at a particular node i are extended along all outgoing arcs (i,j). Dominated labels are eliminated from the extended labels from node i and the labels already present at the end node j. The remaining labels form the new label set at node j. Whenever the label set of a node changes, the node has to be marked for reconsideration. At reconsideration, the mark of the node is deleted. When no nodes are marked for reconsideration any more, the algorithm terminates. When traversing an outgoing arc from a node with multiple labels, every label has to be extended along this arc and tested for dominance with the labels of the end node of the arc, this operation is called *merging*. Merging is the most expensive component of a biobjective label correcting algorithm. The label sets are ordered so that the first component is increasing to reduce computational effort of the merge operation, which in our case is $\mathcal{O}(|L| + |M|)$ when the sets L and M are merged (Brumbaugh-Smith and Shier 1989). We also implement Skriver and Andersen's condition to detect dominance of the whole label set.

Algorithm 1 Biobjective Label Correcting

1: $modNodes = \{s\}$ {list of nodes with modified labels that have not yet been reconsidered, treated in FIFO order}
2: $Labels(s) = \{(0,0)\}$ and $Labels(i) = \varnothing, i = \{1,\ldots,n\}\backslash\{s\}$ {$Labels(i)$ is the list of labels at a particular node i}
3: **while** $modNodes$ is nonempty **do**
4: remove first node i from $modNodes$ {FIFO}
5: **for all** outgoing arcs (i,j) **do**
6: $merge(Labels(i) + c_{ij}, Labels(j))$ {extend all labels at i by c_{ij} and merge with labels at j, eliminating all dominated labels}
7: **if** the label set of j has changed and $j \notin modNodes$ **then**
8: append j to modNodes {FIFO}
9: **end if**
10: **end for**
11: **end while**

Once the label correcting algorithm terminates, the set $Labels(t)$ contains all non-dominated path costs at the target node t. The corresponding efficient solutions (the paths) can be obtained by backtracking the appropriate labels.

4.2 Ranking – Near Shortest Path

Methods such as the k-shortest path method generate one path after the other, in order of increasing length. According to the literature, k-shortest path approaches could not be successfully applied to BSP problems as the cost of finding paths in order of their lengths is quite high (Huarng et al. 1996; Skriver 2000). Instead of a k-shortest path procedure, we use the near shortest path method by Carlyle and Wood (2005), which aims at finding all paths the length of which

is within a certain deviation ϵ from the optimal path length ω, thus having a maximal path length of $\delta = \omega + \epsilon$. We use their implementation of the method `ANSPRO`, which the authors identify as best approach, and carry out some slight modifications. On the basis of computational tests, Carlyle and Wood conclude that their near shortest path routine solves the k-shortest path problem faster than other algorithms dedicated to solving the k-shortest path problem.

In order to use the near shortest path (NSP) procedure, a weighted sum problem (4) corresponding to BSP is considered. Thus weighting factors $\lambda^1 > 0$ and $\lambda^2 > 0$ are defined:

$$\lambda^1 = z_2(x_{lex(1,2)}) - z_2(x_{lex(2,1)}) \text{ and } \lambda^2 = z_1(x_{lex(2,1)}) - z_1(x_{lex(1,2)}). \quad (8)$$

The $lex(1,2)$- and $lex(2,1)$-best solutions are determined in an initialization phase. We investigate the usage of different algorithms in initialization, see Section 4.3.1.

Upper bounds originating from the two lexicographically best solutions $x_{lex(1,2)}$ and $x_{lex(2,1)}$ can be used to restrict enumeration. For every candidate solution $\hat{x}$ with $z(\hat{x}) = (z_1(\hat{x}), z_2(\hat{x}))$ we get:

$$z_1(\hat{x}) \leq z_2(x_{lex(2,1)}) \text{ and } z_2(\hat{x}) \leq z_1(x_{lex(1,2)}). \quad (9)$$

$z^N = (z_1(x_{lex(2,1)}), z_2(x_{lex(1,2)}))$ is called the *nadir point* of the BSP problem, the situation is indicated in Figure 1.

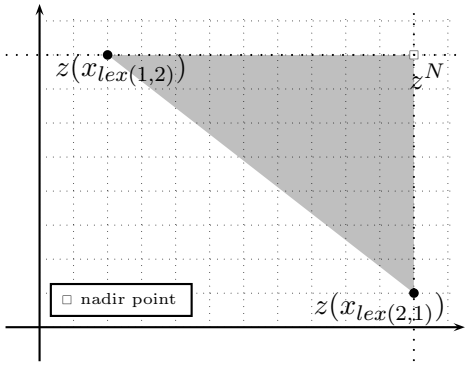

Figure 1: Bounds on z_1 and z_2.

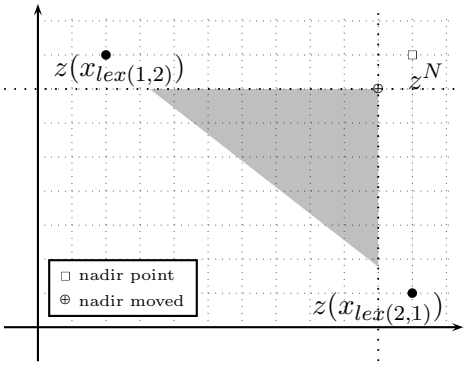

Figure 2: Improved bounds on z_1 and z_2.

The bounds (9) can be further improved by the fact that we are dealing with integer problems. Efficient solutions, which are not equivalent to solutions obtained previously, can only be situated one unit below and one unit to the left of the nadir point z^N as indicated in Figure (2). We get the following improved bounds:

$$z_1(\hat{x}) \leq z_1(x_{lex(2,1)}) - 1 \text{ and } z_2(\hat{x}) \leq z_2(x_{lex(1,2)}) - 1. \quad (10)$$

Algorithm 2 gives a description of the NSP algorithm for a directed graph $G = (N, A)$ with source node s and target node t. A cost $c^\lambda_{ij} > 0$ is associated with each arc (i, j), where $c^\lambda_{ij} = \lambda^1 c^1_{ij} + \lambda^2 c^2_{ij}$. The maximum path length is

$\delta = \lambda^1(z_1(x_{lex(2,1)}) - 1) + \lambda_2(z^2(x_{lex(1,2)}) - 1)$, the weighted sum value of the improved nadir point. We modify NSP slightly to integrate bounds (10) on the respective objectives. We simply add two label sets d^1 and d^2 that keep track of the current value of the two objectives and thus allow for comparison with the respective upper bounds. See Algorithm 2 which incorporates our changes to the original NSP.

Algorithm 2 NSP

$L(i)$: the weighted sum path length at i
$d^1(i), d^2(i)$: length of the path at i for the first and second objective, resp.
for all $i \in N$ **do**
  $d(i)$ = weighted shortest path distance from i to t
end for
$stack \leftarrow s$
$L(s) = 0$ and $d^k(s) = 0; k = 1, 2$
while the stack is not empty **do**
  $i \leftarrow$ top node of *stack*
  if $nextArcOutOf(i) \neq \varnothing$ **then**
    $(i, j) \leftarrow$ next arc out of i
    if $L(i) + c^{\lambda}_{ij} + d(j) \leq \delta$ AND $d^1(i) + c^1_{ij} \leq z^1(x_{i+1}) - 1$ AND $d^2(i) + c^2_{ij} \leq z^2(x_i) - 1$ **then**
      $L(j) = L(i) + c^{\lambda}_{ij}$ and $d^k(j) = d^k(i) + c^k_{ij}; k = 1, 2$
      if j is target node t **then**
        save current candidate solutions {possibly eliminating previous candidate solutions that are now dominated}
        pop j from *stack*
      else
        put j on top of *stack*
      end if
    end if
  else
    pop i from *stack* {no more outgoing arcs}
  end if
end while

The NSP algorithm repeatedly computes candidate solutions within the bounds (10) and with length $< \delta$. Only after the algorithm terminates, we know that the remaining candidate solutions are indeed efficient. It is, however, possible to exploit candidate solutions in order to improve the upper bound δ. We take advantage of the fact that every computed candidate excludes a certain area of the objective space by domination.

First, define the *local nadir point* of two points $z^k = (z^k_1, z^k_2)$ and $z^l = (z^l_1, z^l_2)$ with $z^k_1 < z^l_1$ and $z^k_2 > z^l_2$ to be $z^{LN} = (z^l_1, z^k_2)$.

We consider straight lines parallel to the line connecting $z(x_{lex(1,2)})$ and $z(x_{lex(2,1)})$ through the local nadir point of any two consecutive candidate points and $z(x_{lex(1,2)})$ and $z(x_{lex(2,1)})$ as indicated in Figure 3. The upper bound corresponds to the line through the point that has maximal distance from the

straight line connecting $z(x_{lex(1,2)})$ and $z(x_{lex(2,1)})$. Let $z_c^j = (z_1(x_c^j), z_2(x_c^j))$ with $j \in \{1, \ldots, p\}$ be the candidate solutions ordered by increasing z_1. This yields the upper bound Δ:

$$\begin{aligned} \gamma &= \max\{\lambda^1 z_1(x_c^1) + \lambda^2 z_2(x_{lex(1,2)}), \lambda^1 z_1(x_{lex(2,1)}) + \lambda^2 z_2(x_c^p)\} \\ \Delta &= \max\{\gamma, \max\{\lambda^1 z_1(x_c^{j+1}) + \lambda^2 z_2(x_c^j j); j = 1, \ldots, p-1\}\}. \end{aligned}$$

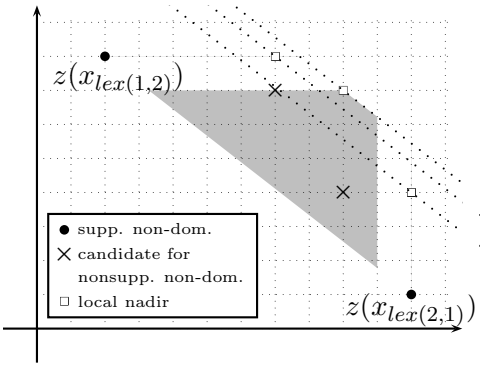

Figure 3: Weighted sum bounds (two candidate points).

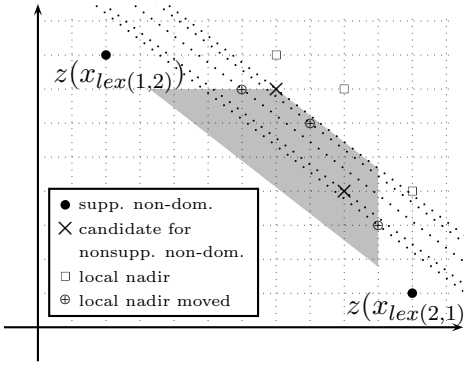

Figure 4: Improved weighted sum bounds (two candidate points).

Again, the upper bound can be improved by considering the point one unit below and one unit to the left of the local nadir point between each pair of consecutive points. But we also have to account for the candidate points themselves as illustrated in Figure 4. This yields the improved upper bound Δ':

$$\begin{aligned} \gamma_1 &= \max\{\lambda^1(z_1(x_c^1) - 1) + \lambda^2(z_2(x_{lex(1,2)}) - 1), \\ &\qquad \lambda^1(z_1(x_{lex(2,1)}) - 1) + \lambda^2(z_2(x_c^p) - 1)\} \\ \gamma_2 &= \max\{\lambda^1 z_1(x_c^j) + \lambda^2 z_2(x_c^j), j = 1, \ldots, p\} \\ \gamma_3 &= \max\{\lambda^1(z_1(x_c^{j+1}) - 1) + \lambda^2(z_2(x_c^j) - 1), j = 1, \ldots, p-1\} \\ \Delta' &= \max\{\gamma_1, \gamma_2, \gamma_3\}. \end{aligned} \tag{11}$$

We refer to Przybylski et al. (2007) for a more detailed presentation of the upper bounds (10) and (11).

In Algorithm 2, we can insert an additional step: δ can be updated by $\Delta' \leq \delta$ whenever a new candidate solution is computed. We insert the following step between steps 15 and 16:

Compute Δ' and update $\delta = \Delta'$.

4.3 Two Phase Method

The two phase method (Mote et al. 1991; Ulungu and Teghem 1995) is based on computing supported and non-supported non-dominated points separately. In phase 1 only the supported efficient solutions are computed, possibly taking

advantage of their property of being obtainable as solutions to the weighted sum problem (4), for an illustration see Figure 5. In phase 2 the non-supported efficient solutions are computed with an enumerative approach, as there is no theoretical characterization for their efficient calculation. It is expected that the search space for the enumerative approach in phase 2 is highly restricted due to information obtained in phase 1 so that the associated problems can be solved a lot quicker than by solving BSP with a purely enumerative approach only. The search space in phase 2 can be restricted to triangles given by two consecutive supported non-dominated points as indicated in Figure 6. An initialization phase is needed in the two phase method that computes one or two initial solutions. We investigate the usage of different solution methods in initialization, phase 1, and phase 2.

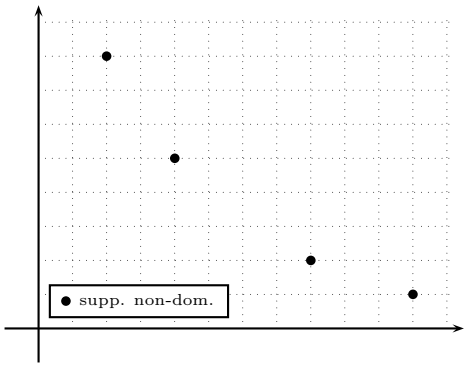

Figure 5: Supported non-dominated points.

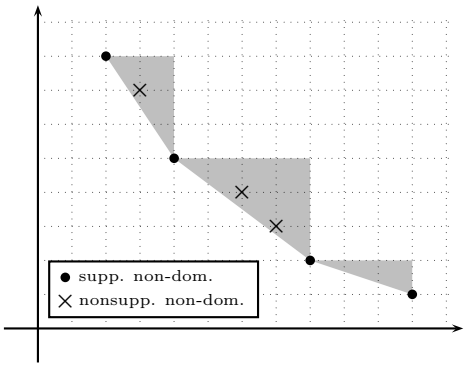

Figure 6: All non-dominated points.

For initialization and in phase 1 we pursue two main approaches. On the one hand we use single objective label setting and correcting shortest path methods. On the other hand we use a network simplex implementation (e.g. Helgason and Kennington 1995) to solve (BSPT) with weighted sum objective. We modify an implementation called MCF (Löbel 2003) for our purposes. The network simplex implementation takes advantage of strongly feasible trees (Cunningham 1976) to prevent cycling.

4.3.1 Initialization

In the initialization phase we need to compute a $lex(1,2)$-best or $lex(2,1)$-best solution or both, depending on the approach chosen in phase 1. Here, single objective shortest path problems are solved with appropriate objective functions, the relations $<$ and $>$ in the following algorithms are adapted to $lex(1,2)$ and $lex(2,1)$ respectively. We investigate the following options:

- Single objective label correcting algorithm (LC). Refer to Bertsekas (1998) for label correcting shortest path algorithms. We modify an implementation of LC by Carlyle and Wood (2005).
- Single objective label setting algorithm: Dijkstra's algorithm (D). For label setting shortest path algorithms refer to Bertsekas (1998).

- Network simplex (S). To run the network simplex, an initial artificial solution for (BSPT) is constructed by adding an artificial root node and artificial arcs connecting that root with all other nodes and equipping them with adequate flow values and cost vectors.

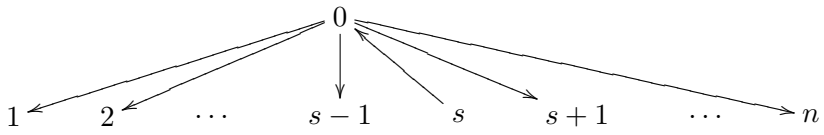

Figure 7: An initial artificial basis with artificial root node 0 with balance 0.

There is an artificial arc from node s to the artificial root 0 with cost vector $(M, 0)$ and flow $n-1$. All other artificial basic arcs from the root 0 to all nodes except s are equipped with cost vectors $(M, 0)$ and a flow of 1 to satisfy the demand of one unit of flow at each node except s. Note that this is the initial solution for obtaining a $lex(1,2)$-best solution.

Multiple partial pricing is used to speed up the selection of basic entering arcs. A disadvantage of the network simplex method is that, when computing the $lex(1,2)$-best solution, in order to eliminate all artificial arcs from the basis, at least as many iterations as there are nodes have to be performed as there is one artificial arc per node. If the $lex(2,1)$-best solution is also required, we can start off from the $lex(1,2)$-best solution.

4.3.2 Phase 1

Phase 1 is dedicated to the computation of supported efficient solutions. This can be done by solving several single objective problems in weighted sum formulation (4), which happens in the two dichotomic approaches SDIC and LDIC described below. The network simplex algorithm gives rise to a prametric approach. Basic entering arcs are chosen in such a manner that all extreme supported efficient solutions are generated. This approach is introduced below as SPAR.

In a dichotomic approach, weights are chosen to obtain a supported nondominated point that has the maximal distance to the straight line connecting the two initial points $z(x_{lex(1,2)})$ and $z(x_{lex(2,1)})$ as illustrated in Figure 8 for the same example as in Figures 5 and 6. The efficient solution $\hat{x}$ thus obtained leads to two new weighted sum problems: one between $z(x_{lex(1,2)})$ and $z(\hat{x})$ (yielding no new solution), and one between $z(\hat{x})$ and $z(x_{lex(2,1)})$ (yielding one more solution), see Figure 9. If the image of the efficient solution of such a problem is distinct from the images of the two supported solutions defining it, two new subproblems can be formulated. Otherwise, there is nothing else to do. The dichotomic method iterates until all weighted sum problems and arising subproblems have been solved and a complete set of the extreme supported efficient solutions is obtained.

The dichotomic method might not allow us to find all nondominated points on $conv(Z)$ in case there are more than two solutions on the same facet. How-

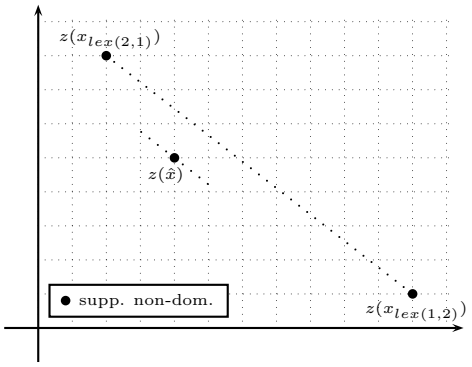

Figure 8: Dichotomic method, first iteration.

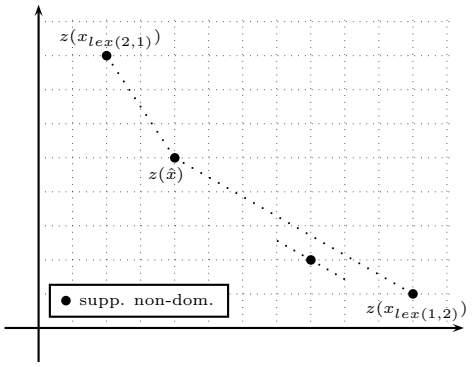

Figure 9: Dichotomic method, second iteration.

ever, all extreme points will be computed. Missing supported solutions are computed in phase 2.

We employ two different solution strategies:

- Network simplex dichotomic (SDIC). We use the network simplex to solve the single objective weighted sum problems that arise from the dichotomic approach.
- Label correcting dichotomic (LDIC). We use the label correcting method described for initialization in Section 4.3.1 to solve the single objective weighted sum problems that arise from the dichotomic approach.

It remains to explain the parametric network simplex approach. Starting off with a $lex(1,2)$-best solution the supported efficient set X_{SE} is explored from the upper left to the lower right. The goal is to reach a $lex(2,1)$-best solution of the problem, and basic entering arcs are chosen so that every supported efficient solution is computed on the way to the $lex(2,1)$-best solution. In each simplex iteration, the basic entering arc is chosen to be the one with the least ratio between improvement of z_2 and the deterioration of z_1, both expressed through reduced costs. Whenever the shortest path from s to t changes, another efficient solution is found. The disadvantage of SPAR is, that all non basic arcs have to be considered when choosing a basic entering arc, as an arc with minimal ratio has to be chosen. Thus partial pricing can not be used to speed up the simplex algorithm. For further detail please refer to Mote et al. (1991). We call this approach:

- Network simplex parametric (SPAR).

4.3.3 Phase 2

In phase 2 it is possible to benefit from the work already done in phase 1 to significantly reduce computation time of the enumerative methods used. Let $z^1, \ldots, z^k$, where $z^i = (z_1(x^i), z_2(x^i))$ and z^i are sorted by increasing z_1, be (at least) the extreme points of a complete supported efficient set obtained in phase

1. It was mentioned before that non-supported non-dominated points can only be situated in the area defined by two consecutive supported non-dominated points, as indicated in Figure 6. For every pair of consecutive supported non-dominated points $z(x^i)$ and $z(x^{i+1})$ with $i \in \{1, \ldots, k-1\}$ an enumerative shortest path method is used to obtain non-supported solutions (if there are any).

We investigate two different enumerative solution procedures. We again employ bounds on each objective and on their weighted sum, in the same manner that was presented in the context of NSP in Section 4.2. For every pair of consecutive supported non-dominated points, z^k and z^l are substituted by $z^i = z(x^i)$ and $z^{i+1} = z(x^{i+1})$, respectively, in (10) and (11). We investigate a label correcting method (LCOR) and a ranking method (NSP).

- Near shortest path (NSP) as described in Section 4.2, is executed for every pair of consecutive supported non-dominated points. Paths are only expanded if they do not violate any bounds. Due to the lower bounds considered in NSP, paths can often be discarded early on during computations.

- Biobjective label correcting (LCOR) as described in Section 4.1. LCOR can also be run for every pair of consecutive supported non-dominated points. Labels are discarded as soon as they violate any bounds. We found that a lot of effort is put in the enumeration of paths that are discarded at a very late stage of the algorithm. In particular, lots of paths are enumerated for every pair of consecutive solutions that do not end up within the bounds for any of them.

  Therefore, in phase 2 we run LCOR just once (instead of once for every triangle), and discard labels if they are not in any of the areas defined by two conscutive supported non-dominated points or can not be extended to end up within any of them.

5 Numerical Results

We investigate the performance of the different solution methods on three different kinds of networks. We introduce the types of networks considered and then present computational results.

5.1 Test Sets

We investigate three different network types: grid networks, random `NetMaker` networks and road networks. We did also experiment with networks generated by `NETGEN` (Klingman et al. 1974), which we modified to incorporate two costs for each arc. The networks thus generated had very few efficient paths, often only between one and three. Therefore we decided not to include `NETGEN` networks in our considerations.

Table 2: Grid network test problems.

Name	h×w	Nodes	Arcs	$\lvert Z_N \rvert$	Name	h×w	Nodes	Arcs	$\lvert Z_N \rvert$
G1	30×40	1202	4720	37	G15	2450×2	4902	19596	6
G2	20×80	1602	6240	80	G16	1225×4	4902	19592	6
G3	50×90	4502	17820	124	G17	612×8	4898	19586	10
G4	90×50	4502	17900	46	G18	288×17	4898	19550	15
G5	50×200	10002	39600	290	G19	196×25	4902	19550	18
G6	200×50	10002	39900	12	G20	140×35	4902	19530	32
G7	100×150	15002	59700	149	G21	111×44	4886	19448	54
G8	150×100	15002	59800	122	G22	92×53	4878	19398	53
G9	100×200	20002	79600	247	G23	79×62	4900	19468	77
G10	200×100	20002	79800	132	G24	70×70	4902	19460	93
G11	200×150	30002	79800	204	G25	62×79	4900	19343	95
G12	50×50	10002	39600	52	G26	53×92	4878	19320	93
G13	100×100	10002	39800	113	G27	44×111	4886	19314	137
G14	200×200	40002	159600	309	G28	35×140	4902	19320	209
					G29	25×196	4902	19208	244
					G30	17×288	4898	19008	371
					G31	8×612	4898	18360	819
					G32	4×1225	4902	17150	1383
					G33	2×2450	4902	19596	1594

5.1.1 Grid Networks

Nodes are arranged in a rectangular grid with given height and width. Every node has at most four outgoing arcs (up, down, left and right), to its immediate neighbours. Only nodes on the boundary of the grid have less outgoing arcs. There are two distinct nodes beyond the grid structure: A source node s and a target node t. There is an arc from s to every node on the left margin of the grid and an arc from every node of the right margin of the grid to the target node t. The costs (c^1_{ij}, c^2_{ij}) for arc (i, j) are chosen randomly from a discrete uniform distribution with $c^k_{ij} \in \{1, 2, \ldots, 10\}, k = 1, 2$. Carlyle and Wood (2005) use grid networks for numerical tests on NSP algorithms. Refer to Table 2 for a listing of problem instances. Instances G15-G33 are grid networks with approximately the same number of nodes, but varying in width and height.

5.1.2 `NetMaker`

Skriver and Andersen (2000) propose an alternative, `NetMaker`, to using a pure random network generator such as `NETGEN`. They state that `NETGEN` generates networks containing very few efficient paths, an observation we agree with. Here, nodes are numbered from 1 to n, where node 1 is the source node, node n is the target node. We use a random number generator that generates discrete uniformly distributed random numbers. `NetMaker` networks are constructed by first generating a random Hamiltonian cycle. Then a random number of arcs out of every node is generated, in between a minimum and maximum number of outgoing arcs. An arc out of node i can only reach nodes j with $j \in [i - \lceil \frac{I_{node}}{2} \rceil, i + \lceil \frac{I_{node}}{2} \rceil]$, where I_{node} denotes the *node interval*, the maximum allowed range for an arc. Arc costs are determined randomly. It is randomly chosen whether $c^1_{ij} \in \{1, 2, \ldots, 33\}$ or $c^1_{ij} \in \{67, 68, \ldots, 100\}$ and a number in the chosen interval is randomly allocated as cost. The cost c^2_{ij} is then randomly chosen from the other interval. We investigate three modifications to

Table 3: `NetMaker` network test problems.

			Outgoing arcs		Var a)		Var b)		Var c)	
Name	Nodes	I_{node}	min	max	Arcs	$\|Z_N\|$	Arcs	$\|Z_N\|$	Arcs	$\|Z_N\|$
NM1	3000	20	5	15	31559	6	31502	1	31646	3
NM2	3000	20	1	20	33224	8	33122	1	33229	4
NM3	3000	50	5	15	31345	9	31548	2	31775	2
NM4	3000	50	1	20	33536	15	32641	3	32963	4
NM5	3000	50	10	40	76095	6	76924	3	77388	3
NM6	7000	20	5	15	73524	6	73940	1	73575	2
NM7	7000	20	1	20	77024	5	76775	3	76547	3
NM8	7000	50	5	15	73676	3	73282	2	73369	3
NM9	7000	50	1	20	76821	7	77518	1	76658	3
NM10	7000	50	10	40	178476	6	178292	6	180611	4
NM11	14000	20	5	15	146598	6	147388	2	146979	2
NM12	14000	20	1	20	154159	6	154115	4	154252	1
NM13	14000	50	5	15	146919	2	146900	2	147187	1
NM14	14000	50	1	20	153742	17	154213	2	153068	4
NM15	14000	50	10	40	357866	7	358264	3	356367	3
NM16	21000	20	5	15	220313	5	220685	3	220794	3
NM17	21000	20	1	20	231402	4	230403	1	230432	1
NM18	21000	50	5	15	220687	7	219606	3	219931	1
NM19	21000	50	1	20	230497	4	231876	2	232465	1
NM20	21000	50	10	40	534288	5	536151	3	533980	3

the structure of `NetMaker`:

a) Penalize the cycle: Arc weights c_{ij}^k, $k = 1, 2$ as above but for all arcs in the Hamiltonian cycle, choose c_{ij}^k, $k = 1, 2$ randomly in $\{1, 2, \ldots, 10000\}$.

b) To make `NetMaker` networks more comparable to grid networks, we enforce that roughly half of the arcs out of a node go to nodes with higher node numbers and half of them to nodes with lower numbers. Arc weights are chosen like in a) for all arcs of the Hamiltonian cycle, for all other arcs set $c_{ij}^k \in \{1, 2, \ldots, 10\}, k = 1, 2$.

c) More penalty on cycle: For all arcs that are part of the Hamiltonian cycle $c_{ij}^k = 10000$, $k = 1, 2$. Everything else is the same as in b).

For problem instances of `NetMaker` refer to Table 3.

5.1.3 Road Networks

The road networks of the states of the US were extracted by Schultes (2005) from US Census (2000). We use road networks to test our methods on real world data. The original data come as undirected networks, we convert them into directed networks by duplicating arcs. We also add a Hamiltonian cycle with high arc costs to ensure connectedness of the networks. In the original data, there is not always a path from a node to every other node. This happens for example for the Rhode Island data, as there are a few islands that are not connected to the mainland via roads. Each arc (i, j) is equipped with arc costs where c_{ij}^1 is the time needed to travel the arc and c_{ij}^2 is the travel distance in meters. Travel time is determined by multiplying the travel distance of an arc by one of four different road quality factors. Source and target node are chosen randomly from a discrete uniform distribution.

Table 4: Road network test problems.

Name	State	Nodes	Arcs	$\lvert Z_N \rvert$: Average	Min	Max
DC1-DC9	Washington DC	9559	39377	3.33	1	7
RI1-RI9	Rhode Island	53658	192084	9.44	2	22
NJ1-NJ9	New Jersey	330386	1202458	10.44	2	21

We run tests with three different kinds of road networks. We use the networks of the states Washington DC, Rhode Island and New Jersey, the network sizes are listed in Table 4. For each road network we test nine instances with different source and target nodes.

5.2 Results

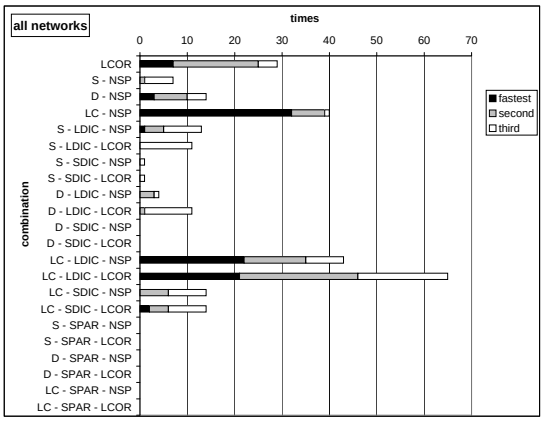

Figure 10: All networks – fastest three algorithms.

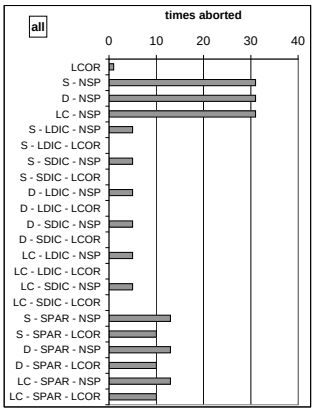

Figure 11: Tests exceeding timeout of 30 minutes.

We identify the best methods for the different phases in the two phase method. We also compare total computation times of the two phase method and the enumerative approaches LCOR and NSP.

The different solution methods presented in Section 4 lead to a total of 22 combinations. For the enumerative solution methods we have NSP with three different initializations and LCOR. There are 18 different combinations for the two phase method:

- LCOR
- NSP and initialization with L/D/S
- Two phase method:
  - initialization with L/D/S
  - phase 1 with SDIC/LDIC/SPAR
  - phase 2 with LCOR/NSP

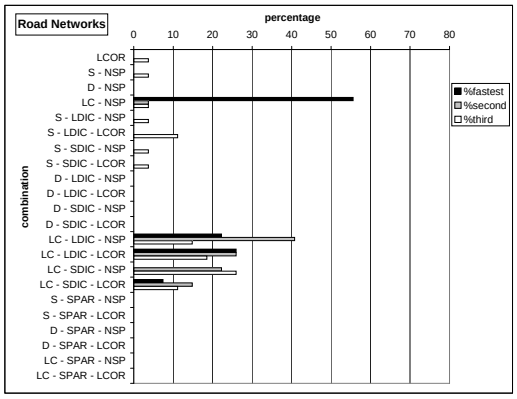

Figure 12: Road Networks – fastest three algorithms.

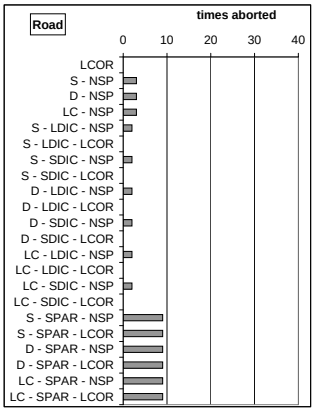

Figure 13: Tests exceeding timeout of 30 minutes.

All numerical tests are performed on a Linux (Fedora Core 4, kernel 2.6.14-1.1656) computer with 3GHz Pentium 4 processor and 1GB RAM. We use the gcc compiler (version 4.0.2) with compile option -O3. The methods are implemented in C, we adapt program code from Carlyle and Wood (2005) for NSP and LC. The network simplex is a modified version of MCF (by Löbel 2003). When measuring runtime, we disregard the time it takes to read the problem from a problem file. Runtime does include the generation of all non-dominated path labels together with the actual paths. In LCOR the paths can be obtained by backtracking the labels at each node. We do not include the time for the backtracking process in the runtime. Whenever the runtime exceeded 30 minutes, the computation was aborted.

Note that we omit all problem instances that have only one efficient solution, which means that there is a path that is optimal in both objectives. The according run times would falsify our results, as the $lex(1,2)$-best and $lex(2,1)$-best solutions will have the same path costs, which can already be detected after initialization in any approach that requires two initial solutions. This occurs for the two phase method with dichotomic phase 1 (SDIC or LDIC) and the NSP algorithm in which case the question is only which initialization method is the fastest one. Computational results show that the fastest initialization method in this case is LC, followed by S.

We do not list the actual run times in the following, as there are just too many results – 22 computations are performed for each problem instance and we have 80 instances. Instead we focus on the fastest three combinations for each problem instance. We list how often each combination is the fastest, the second fastest or the third fastest one, see Figures 10, 12, 14, and 16. There may be several combinations that achieve the fastest, second fastest or third fastest runtime. We also display how often a computation was aborted after running for 30 minutes in Figures 11, 13, 15, and 17. To give the reader an idea of run times, we include Tables 5 and 6 in which the fastest runtime for each problem is listed. The fastest approaches for *all* test problem are displayed in Figure 10 and 11. Considering this graph, the most successful combinations are LC -

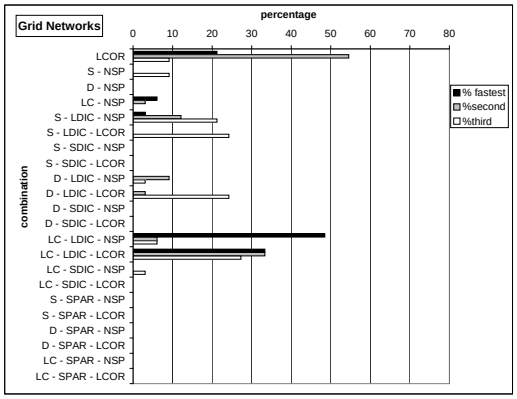

Figure 14: Grid Networks – fastest three algorithms.

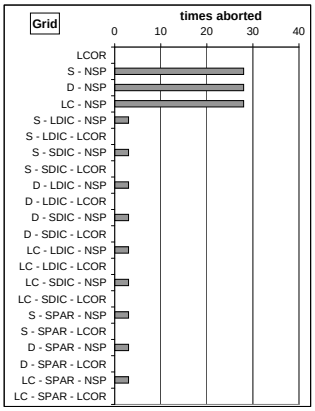

Figure 15: Tests exceeding timeout of 30 minutes.

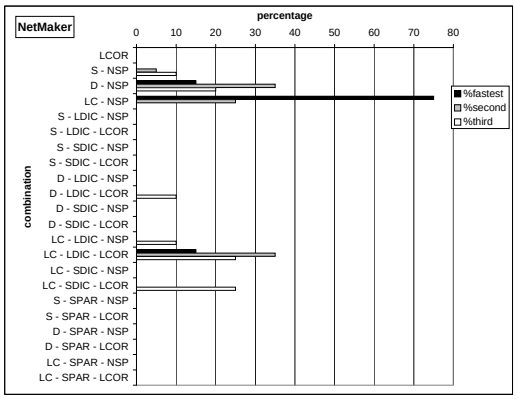

Figure 16: `NetMaker` Var a) – fastest three algorithms.

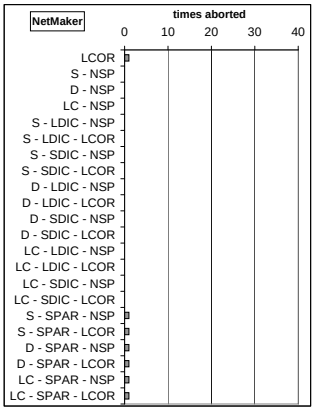

Figure 17: Tests exceeding timeout of 30 minutes.

LDIC - NSP/LCOR, followed by LC - NSP and LCOR. We can also discard some approaches: Initialization with S or D is not very successful. Furthermore, SPAR in phase 1 of the two phase method does not perform well. This behavior does not come surprisingly – as was mentioned in Section 4.3.2, in every simplex iteration all non-basic arcs have to be considered, which is a huge computational effort.

It is, however, worthwhile considering each problem class individually, see Figures 12, 14, and 16. In these figures, we display in how many percent of all computational tests of a problem class a combination is fastest (or second/third fastest). The figures show that in the different problem classes, different approaches are more or less effective. Consider road networks first (Figure 12). We identify the NSP algorithm with LC initialization as quickest approach. Also, the two phase method with LC - LDIC - NSP/LCOR is rather successful, employing SDIC in phase 1 is slightly worse.

Performance on grid networks is very interesting (Figure 14). The NSP algorithm performs really badly here, it is aborted 28 times out of 33. LCOR

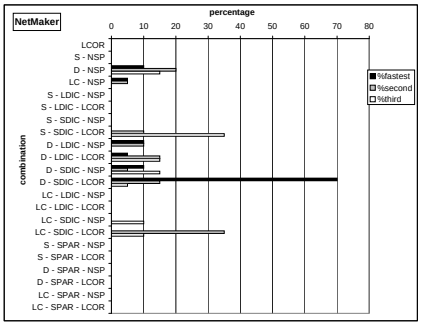

Figure 18: `NetMaker` Var b) – fastest three algorithms.

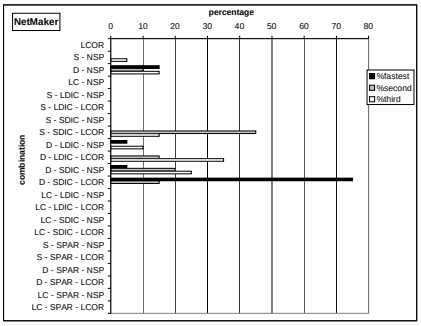

Figure 19: `NetMaker` Var c) – fastest three algorithms.

in contrast performs quite well on grid networks. However, the most successful approach for grid networks is LC - LDIC - *NSP*. This demonstrates well the benefit of the two phase method. Runtime is significantly improved for NSP in phase 2.

Var a) of the `NetMaker` networks shows behavior that is very different to grid networks. Here, the combination LC - NSP is very successful, whereas LCOR does not even appear among the fastest three. The fastest two phase approach is the combination LC - LDIC - *LCOR*, again illustrating the benefit of the two phase method.

Behavior on the last two problem classes is very contrary. On grid networks, LCOR alone performs very well, but in the two phase method the combination with NSP as phase 2 is best. In `NetMaker` networks on the other hand, LC - NSP alone is the most successful approach but the two phase method succeeds with LCOR in phase 2.

It is worth noting that variations in arc costs in the test networks may lead to drastically different results. We investigate three different variations of `NetMaker`. The networks have the same basic parameters, yet computational results are very different. In Figures 16, 18 and 19 the different results for Variations a), b) and c) are illustrated. Variations b) and c) are the only problem instances investigated where the usage of a single objective label setting method (D) was successful. In variations b) and c) the two phase method is a lot more successful than in variation a).

We conclude this Section with observations about the best approach to be chosen for each phase, again distinguishing the different problem classes, see Figure 20. We again distinguish the three different types of networks, results for each network type are displayed in different color. For each phase of the two phase method we indicate which approach was most successful by displaying in how many percent of all problem instances an approach was the fastest one. We distinguish between initialization that computes one or two initial solutions. We also compare the two enumerative approaches LCOR and LC - NSP (which has performed best). Note that we only consider Var a) of `NetMaker` here.

The best approach for initialization is LC throughout the different network types. The results for phase 1 of the two phase method clearly indicate that

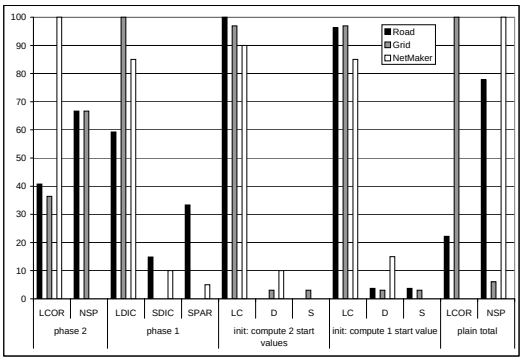

Figure 20: Fastest approaches in different phases.

LDIC is the best approach to employ. The preferred approach in phase 2 depends on the network type, LCOR is better suited for grid networks, NSP for road and `NetMaker` networks. For purely enumerative approaches behavior is again just the opposite.

In Tables 5 and 6 the best run times for all problem instances are listed together with the best approach. All instances could be solved in less than one minute, the nine instances that required more than ten seconds all have more than 200 efficient solutions or more than 50,000 nodes.

6 Conclusion

We were able to show that the two phase method is competitive with other commonly applied approaches to solve the BSP problem. The two phase method works well with both a ranking approach and a label correcting approach in phase 2. The purely enumerative NSP approach was a very successful approach to solve some problem instances, however it often had to be aborted on others.

We illustrate all this on various test instances. It also becomes clear that it depends a lot on the network type which approach performs best, even small variations on the network may have a high impact on performance.

Table 5: Fastest runtime for grid networks (top) and road networks (bottom).

Instance	h	w	$\|Z_N\|$	Time (sec)	Best approach
G1	30	40	37	0.01	LC-LDIC-NSP
G2	20	80	80	0.05	LC-LDIC-NSP
G3	50	90	124	0.22	LC-LDIC-NSP
G4	90	50	46	0.04	LC-LDIC-NSP
G5	50	200	290	6.21	LC-LDIC-LCOR
G6	200	50	44	0.08	LC-LDIC-NSP
G7	100	150	149	4.57	LC-LDIC-LCOR
G8	150	100	122	0.66	LC-LDIC-NSP
G9	100	200	247	13.27	LC-LDIC-LCOR
G10	200	100	132	0.82	LC-LDIC-NSP
G11	200	150	204	5.87	LC-LDIC-NSP
G12	50	50	52	0.04	LC-LDIC-NSP
G13	100	100	113	1.02	LC-LDIC-LCOR
G14	200	200	309	28.49	LC-LDIC-LCOR
G15	2450	2	6	0.01	LCOR
G16	1225	4	6	0.01	LC-NSP / LCOR
G17	612	8	10	0.01	LCOR / LC-NSP / LC-LDIC-NSP
G18	288	17	15	0.02	LC-LDIC-NSP
G19	196	25	18	0.01	LC-LDIC-NSP
G20	140	35	32	0.02	LC-LDIC-NSP
G21	111	44	54	0.05	LC-LDIC-NSP
G22	92	53	53	0.08	LC-LDIC-NSP
G23	79	62	77	0.09	LC-LDIC-NSP
G24	70	70	93	0.30	LC-LDIC-LCOR
G25	62	79	95	0.31	LC-LDIC-LCOR
G26	53	92	93	0.41	LC-LDIC-LCOR
G27	44	111	137	0.63	LCOR / LC-LDIC-LCOR
G28	35	140	209	1.23	LC-LDIC-LCOR
G29	25	196	244	2.11	LC-LDIC-LCOR
G30	17	288	371	4.27	S-LDIC-NSP
G31	8	612	819	16.53	LCOR
G32	4	1225	1383	33.30	LCOR
G33	2	2450	1594	39.06	LCOR

Instance	Nodes	Arcs	$\|Z_N\|$	Time (sec)	Best approach
DC1	9559	39377	2	0.14	LC-NSP
DC2	9559	39377	6	0.24	LC-SDIC-LCOR
DC3	9559	39377	3	0.07	LC-LDIC-LCOR
DC4	9559	39377	2	0.05	LC-NSP
DC5	9559	39377	1	0.07	LC-NSP / 2phase with init = LC
DC6	9559	39377	7	0.29	LC-NSP
DC7	9559	39377	2	0.10	LC-NSP
DC8	9559	39377	1	0.08	LC-NSP / 2phase with init = LC
DC9	9559	39377	6	0.19	LC-LDIC-LCOR
RI1	53658	192084	3	0.40	LC-NSP
RI2	53658	192084	15	6.60	LC-LDIC-NSP
RI3	53658	192084	2	0.40	LC-NSP
RI4	53658	192084	17	3.23	LC-LDIC-NSP
RI5	53658	192084	16	2.97	LC-LDIC-LCOR
RI6	53658	192084	3	11.56	LC-LDIC-NSP
RI7	53658	192084	3	0.55	LC-NSP
RI8	53658	192084	4	0.46	LC-NSP
RI9	53658	192084	22	2.06	LC-LDIC-LCOR
NJ1	330386	1202458	2	2.45	LC-NSP
NJ2	330386	1202458	6	4.09	LC-NSP
NJ3	330386	1202458	21	7.47	LC-NSP
NJ4	330386	1202458	5	2.25	LC-NSP
NJ5	330386	1202458	7	13.75	LC-LDIC-LCOR
NJ6	330386	1202458	12	14.68	LC-LDIC-NSP
NJ7	330386	1202458	6	7.16	LC-LDIC-LCOR
NJ8	330386	1202458	13	3.81	LC-NSP
NJ9	330386	1202458	2	53.57	LC-NSP

Table 6: Fastest runtime for `NetMaker` networks.

					Var a)			Var b)			Var c)		
Instance	Nodes	Int.	Min	Max	$\|Z_N\|$	Time	Best approach	$\|Z_N\|$	Time	Best approach	$\|Z_N\|$	Time	Best approach
NM1	3000	20	5	15	6	0.10	D-NSP / LC-NSP	1	0.05	D-NSP / 2phase with init = D	3	0.12	D-SDIC-LCOR
NM2	3000	20	1	20	8	0.09	LC-NSP	1	0.05	D-NSP / 2phase with init = D	4	0.15	D-SDIC-LCOR
NM3	3000	50	5	15	9	0.06	LC-NSP	2	0.06	D-NSP	2	0.06	D-NSP
NM4	3000	50	1	20	15	0.07	LC-NSP	3	0.16	D-SDIC-LCOR	4	0.14	D-NSP
NM5	3000	50	10	40	6	0.19	D-NSP	3	0.31	D-SDIC-NSP	3	0.23	D-SDIC-LCOR
NM6	7000	20	5	15	6	0.33	LC-NSP	1	0.26	D-NSP / 2phase with init = D	2	0.38	D-SDIC-LCOR
NM7	7000	20	1	20	5	0.24	LC-NSP	3	0.72	D-SDIC-LCOR	3	0.46	D-SDIC-LCOR
NM8	7000	50	5	15	3	0.16	LC-LDIC-LCOR	2	0.51	D-SDIC-LCOR	3	0.40	D-SDIC-LCOR
NM9	7000	50	1	20	7	0.15	LC-NSP	1	0.25	D-NSP / 2phase with init = D	3	0.49	D-SDIC-LCOR
NM10	7000	50	10	40	6	0.51	D-NSP	6	1.82	D-SDIC-LCOR	4	0.73	D-SDIC-LCOR
NM11	14000	20	5	15	6	0.88	LC-NSP	2	2.22	D-SDIC-LCOR	2	1.56	D-SDIC-LCOR
NM12	14000	20	1	20	6	0.79	LC-NSP	4	2.10	D-SDIC-LCOR	1	0.98	D-NSP / 2phase with init = D
NM13	14000	50	5	15	2	0.45	LC-LDIC-LCOR	2	2.03	D-SDIC-LCOR	1	0.94	D-NSP / 2phase with init = D
NM14	14000	50	1	20	17	0.53	LC-NSP	2	1.86	D-SDIC-LCOR	4	1.91	D-SDIC-LCOR
NM15	14000	50	10	40	7	2.03	LC-NSP	3	5.08	D-SDIC-NSP	3	2.31	D-SDIC-LCOR
NM16	21000	20	5	15	5	1.54	LC-NSP	3	4.34	D-SDIC-LCOR	3	4.40	D-SDIC-NSP
NM17	21000	20	1	20	4	1.48	LC-NSP	1	2.41	DD-NSP 2phase with init = D	1	2.11	D-NSP / 2phase with init = D
NM18	21000	50	5	15	7	0.97	LC-NSP	3	3.92	D-SDIC-LCOR	1	2.08	D-NSP / 2phase with init = D
NM19	21000	50	1	20	4	0.88	LC-LDIC-LCOR	2	2.19	LC-NSP	1	2.13	D-NSP / 2phase with init = D
NM20	21000	50	10	40	5	3.30	LC-NSP	3	4.34	D-SDIC-LCOR	3	3.11	D-SDIC-LCOR

References

D.P. Bertsekas. *Network Optimization Continuous and Discrete Models*. Athena Scientific, Belmont, Massachusetts, 1998.

J. Brumbaugh-Smith and D. Shier. An empirical investigation of some shortest path algorithms. *European Journal of Operational Research*, 43(2):216–224, 1989.

W.M. Carlyle and R.K. Wood. Near-shortest and k-shortest paths. *Networks*, 46(2):98–109, 2005.

J.C.N. Clímaco and E.Q.V. Martins. A bicriterion shortest path problem. *European Journal of Operational Research*, 11:399–404, 1982.

H.W. Corley and I.D. Moon. Shortest paths in networks with vector weights. *Journal of Optimization Theory and Applications*, 46(1):79–86, 1985.

W.H. Cunningham. A network simplex method. *Mathematical Programming*, 11:105–116, 1976.

M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22:425–460, 2000.

M. Ehrgott and X. Gandibleux. Multiobjective combinatorial optimization – theory, methodology, and applications. In M. Ehrgott and X. Gandibleux, editors, *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, International Series in Operationas Research & Management Science, pages 369–444. Kluwer, 2002.

V. Gabrel and D. Vanderpooten. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite. *European Journal of Operational Research*, 139(2):533–542, 2002.

G. Gallo and S. Pallotino. Shortest path algorithms. *Annals of Operations Research*, 13:3–79, 1988.

F. Guerriero and R. Musmanno. Label correcting methods to solve multicriteria shortest path problems. *Journal of Optimization Theory and Applications*, 111(3):589–613, 2001.

P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making, Theory and Application. Proceedings of the 3rd International Conference, Hagen/Königswinter 1979*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 109–127. Springer Verlag, Berlin, 1980.

R. Hartley. Vector optimal routing by dynamic programming. In P. Serafini, editor, *Mathematics of Multiobjective Optimization*, CISM International Centre for Mechanical Sciences – Courses and Lectures, pages 215–224. Springer Verlag, Wien, 1985.

R.V. Helgason and J.L. Kennington. Primal simplex algorithms for minimum cost network flows. In M. O. Ball, C.L. Magnanti, T.L.and Monma, and G.L. Nemhauser, editors, *Network Models*, Handbooks in Operations Research and Management Science, pages 85–133. Elsevier, Amsterdam, 1995.

M. Henig. The shortest path problem with two objective functions. *European Journal of Operational Research*, 25:281–291, 1985.

F. Huarng, S. Pulat, and L.-H. Shih. A computational comparison of some bicriterion shortest path algorithms. *The Chinese Institute of Industrial Engineers. Journal*, 13(2):121–125, 1996.

D. Klingman, A. Napier, and J. Stutz. NETGEN: A program for generating large scale assignment, transportation, and minimum cost flow problems. *Management Science*, 20:814–821, 1974.

A. Löbel. MCF, version 1.3. `http://www.zib.de/Optimization/Software/Mcf/`, 2003.

E.Q.V. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16:236–245, 1984.

E.Q.V Martins and J.L.E. Dos Santos. The labelling algorithm for the multiobjective shortest path problem. Technical report, Universidade de Coimbra, Portugal, Departamento de Matématica, 2000.

J. Mote, I. Murthy, and D. L. Olson. A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53: 81–92, 1991.

M. Müller-Hannemann and K. Weihe. On the cardinality of the Pareto set in bicriteria shortest path problems. *Annals of Operations Research*, 147: 269–286, 2006.

S. Pallottino and M.G. Scutellà. Shortest path algorithms in transportation models: classical and innovative aspects. In P. Marcotte and S. Nguyen, editors, *Equilibrium and Advanced Transportation Modelling*, pages 245–281. Kluwer Academic Publishers, 1998.

A. Przybylski, X. Gandibleux, and M. Ehrgott. Two-phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, To appear, 2007.

V.N. Sastry, T.N. Janakiraman, and S.I. Mohideen. New algorithms for multi objective shortest path problem. *Opsearch*, 40(4):278–298, 2003.

V.N. Sastry, T.N. Janakiraman, and S. I. Mohideen. New polynomial time algorithms to compute a set of Pareto optimal paths for multi-objective shortest path problems. *International Journal of Computer Mathematics*, 82(3):289–300, 2005.

D. Schultes. Tiger Road Networks for 9th DIMACS Implementation Challenge – Shortest Path. `http://www.dis.uniroma1.it/~challenge9/data/tiger/`, 2005.

P. Serafini. Some considerations about computational complexity for multi objective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent advances and historical development of vector optimization*, volume 294 of *Lecture Notes in Economics and Mathematical Systems*, pages 222–232. Springer Verlag, Berlin, 1986.

A.J.V. Skriver. A classification of bicriterion shortest path (BSP) algorithms. *Asia-Pacific Journal of Operational Research*, 17:199–212, 2000.

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

D. L. Sonnier, Y. Chan, and M. Bradley. A fast parallel algorithm for the multicriteria shortest path problem. 2006.

C.T. Tung and K.L. Chew. A bicriterion Pareto-optimal path algorithm. *Asia-Pacific Journal of Operational Research*, 5:166–172, 1988.

C.T. Tung and K.L. Chew. A multicriteria Pareto-optimal path algorithm. *European Journal of Operational Research*, 62:203–209, 1992.

E. L. Ulungu and J. Teghem. The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20(2):149–165, 1995.

US Census. US Census 2000 TIGER/Line Files. `http://www.census.gov/geo/www/tiger/tigerua/ua_tgr2k.html`, 2000.

J.Y. Yen. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, 27 (4):526–530, 1970.

F.B. Zahn and C. E. Noon. Shortest path algorithms: An evaluation using real road networks. *Transportation Science*, 32(1):65–73, 1998.