Intl. Trans. in Op. Res. 8 (2001) 15–29

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Algorithms for the weight constrained shortest path problem

Irina Dumitrescu and Natashia Boland

Department of Mathematics and Statistics, University of Melbourne, Parkville VIC 3052, Australia

Received 7 July 1999; received in revised form 20 April 1999; accepted 2 May 2000

Abstract

Given a directed graph whose arcs have an associated cost, and associated weight, the weight constrained shortest path problem (WCSPP) consists of finding a least-cost path between two specified nodes, such that the total weight along the path is less than a specified value. We will consider the case of the WCSPP defined on a graph without cycles. Even in this case, the problem is NP-hard, unless all weights are equal or all costs are equal, however pseudopolynomial time algorithms are known. The WCSPP applies to a number of real-world problems. Traditionally, dynamic programming approaches were most commonly used, but in recent times other methods have been developed, including exact approaches based on Lagrangean relaxation, and fully polynomial approximation schemes. We will review the area and present a new exact algorithm, based on scaling and rounding of weights.

Keywords: combinatorial optimization, networks, routing

1. Introduction

Given a directed graph which has a cost, and a weight associated with each arc, the weight constrained shortest path problem (WCSPP) consists of finding the least-cost route between two specified nodes, such that the total weight is less than a specified value. We note that the WCSPP can also be defined with weight on nodes rather than on arcs: it is not hard to show that the two problems are equivalent. The WCSPP is an NP-hard problem, even if we consider the case when the graph is acyclic and all weights and costs are positive (Garey and Johnson, 1979, p. 214).

Throughout this paper $G = (V, A)$ will be a directed acyclic graph, where $V = \{1, \ldots, n\}$ is the set of nodes, $|V| = n$, and A is the set of arcs. Each arc $(i, j) \in A$ has an associated weight w_{ij} which is a non-negative integer, and an associated cost c_{ij} which is a non-negative integer. A path in the graph is a sequence of nodes $(i_0, i_1, \ldots, i_p)$ such that $(i_{k-1}, i_k) \in A$ for all k from 1 to p. Let W, a positive integer, be the weight limit. We say that the path is weight feasible if and only if the total weight accumulated along the path is at most W, ie, $\sum_{k=1}^{p} w_{i_{k-1}i_k} \leq W$. Let s be a given source node, and t a given destination node. The WCSPP consists of finding a minimum cost weight feasible path in G from s to t.

An integer formulation of the WCSPP is given as follows:

$$\min \sum_{a \in A} c_a x_a$$

$$\text{s.t.} \sum_{a \in \delta^+(i)} x_a - \sum_{a \in \delta^-(i)} x_a = \begin{cases} 1, & \text{if } i = s \\ -1, & \text{if } i = t \\ 0, & \text{if } i \in V \backslash \{s, t\} \end{cases} \qquad \forall i \in V \tag{1}$$

$$\sum_{a \in A} w_a x_a \leqslant W \tag{2}$$

$$x_a \in \{0, 1\}, \qquad \forall a \in A \tag{3}$$

where $\delta^+(i)$ denotes the set of arcs leaving node i, ie, $\delta^+(i) = \{(i, j) \in A\}$, and $\delta^-(i)$ denotes the set of arcs entering node i, ie, $\delta^-(i) = \{(j, i) \in A\}$, for each $i \in V$.

We give an example of WCSPP: consider the graph in Figure 1, where $s = 1$, $t = 5$ and $W = 6$, and the values associated with each arc represent the cost and the weight respectively. The paths from node 1 to node 5 are (1, 2, 3, 5), (1, 2, 5), (1, 2, 4, 5) and (1, 4, 5). The optimal solution of WCSPP is the path (1, 2, 3, 5), of cost 8, and total weight 4. Although the paths (1, 2, 4, 5), and (1, 4, 5) are 'cheaper', they are not weight feasible, so they are not solutions of WCSPP.

The WCSPP is closely related to other problems that have appeared in the literature. For example, it is a special case of the shortest path problem with time windows (SPPTW), and also of the resource constrained shortest path problem (RCSPP), which uses a vector of weights, or resources, rather than a scalar. The WCSPP is also closely related to the multi-objective shortest path problem (MOSPP), which seeks all Pareto optimal solutions. When there are only two objectives, the MOSPP is known as the bi-criteria shortest path problem. Clearly for any WCSPP there exists an optimal solution which is Pareto optimal for the bi-criteria shortest problem with two criteria, minimizing cost and weight, so any method which generates all Pareto optimal solutions to this bi-criteria problem also generates an optimal solution to the WCSPP. Although this approach is unlikely to be as efficient as solving the WCSPP directly, efficient specializations of methods first developed for multi-objective problems have proved useful.

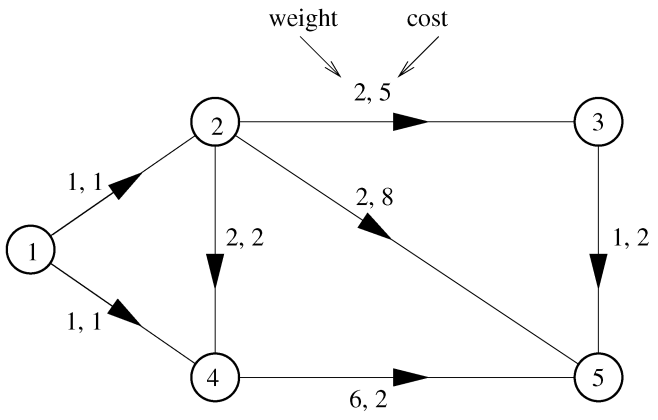

Fig. 1. An example of a WCSPP, $s = 1$, $t = 5$, $W = 6$.

The WCSPP, and closely related problems such as the SPPTW, RCSPP and MOSPP, apply to a number of real-world situations, (for example Halpern and Priess (1974) describe an application to railroad management), and also arise in practice as subproblems of some well-known problems, for example in the case of column generation approaches to vehicle routing problems with time windows (Desrochers, Desrosiers and Solomon, 1992), or to long-haul aircraft routing problems (Barnhart et al., 1998). In the latter case, the column generation subproblem involves a graph in which nodes represent flights. Paths must start with a flight which can follow aircraft maintenance, and end with a flight which can be followed by maintenance. Weights represent flying hours in each flight or indicate date lines crossed, and the maximum weight models a regulated maximum on flying hours or days between maintenance. For long-haul flying, the graphs will not naturally have weight feasible cycles.

Our purpose in this paper is partly expository: we give a review of the area, highlighting common features and differences between approaches to the WCSPP and closely related problems seen in the literature. We then propose a new exact algorithm which we believe overcomes some of the shortfalls of existing approaches. We begin with a brief overview of the methods for solving the WCSPP. Then we focus on three particular methods, which may be viewed as representative of the different approaches which have appeared in the literature. We first extend the pre-processing methods existing in literature. We describe in some detail a dynamic programming approach, in the form of a label setting algorithm, an approach using Lagrangean relaxation, and a fully polynomial approximation scheme, based on cost scaling. In the next section we present a new exact algorithm, using weight scaling. Advantages of our algorithm are that it can be applied to problems having costs of large magnitude, and that it yields a sequence of tighter lower bounds, whilst still enjoying a worst-case complexity guarantee. Finally, we summarize and suggest directions for further research.

2. An overview of methods in the literature

Methods appearing in the literature that apply to the WCSPP can be divided into those based on k-shortest paths, node labelling methods derived from dynamic programming equations, Lagrangean relaxation, and approximation algorithms, although some work combines these approaches. Pre-processing can also be important.

It was observed early that k-shortest path methods could solve the WCSPP: paths are generated in increasing order of cost, and the method stopped as soon as a weight feasible path is found. Whilst many k-shortest path methods have been developed, their complexity when applied to solving the WCSPP is exponential. Little computational experience is available: Handler and Zang (1980) implement the k-shortest path method of Yen (1971) and compare it with their Lagrangean relaxation approach. They report the latter to be substantially faster, in some cases by orders of magnitude. (Note that Handler and Zang's method in fact closes the duality gap by using Yen's method; this an example of work which combines approaches.)

A number of early papers gave dynamic programming formulations, for example Joksch (1966), and Lawler (1976, Chapter 3). A variety of algorithms based on these dynamic programming formulations, all using some kind of node labelling approach, have been developed since. Examples are the methods of Aneja et al. (1983), Desrosiers, Pelletier and Soumis (1983), Desrochers and Soumis (1988a), and, more recently, Jaumard, Semet and Vovor (1996). A good exposition of some of these approaches can also be found in Desrosiers et al. (1995). The label setting algorithm of Desrochers and Soumis

(1988a) appears to have been widely regarded as the most effective for the WCSPP. (See for example the comments in Jaumard et al., 1996.) With appropriate data structures, when the weights are positive, its worst-case complexity is reported in Desrochers and Soumis (1988a) to be $\mathcal{O}(|A|W)$, with results given showing problems with up to 2500 nodes and 250 000 arcs solved in a matter of seconds. Because of its pseudopolynomial time complexity, the label setting algorithm is also very useful as a subprocedure in approximation algorithms. We describe the label setting algorithm in some detail, and discuss its use in a fully polynomial approximation scheme. In the context of column generation, Desrochers and Soumis (1988b) develop a re-optimization algorithm based on label setting, designed especially to re-solve problems after a subset of the nodes has been removed. For re-solution, they report a 2–10 times speed-up over label setting.

Lagrangean relaxation of the weight constraint is another approach that has appeared widely in the literature. Handler and Zang (1980) gave such an algorithm. They solve the Lagrangean dual problem using a specialization of Kelley's cutting plane method (Kelley, 1960), and close the duality gap, if there is one, by applying the kth shortest path algorithm of Yen (1971), with lengths set to the reduced costs arising from the optimal Lagrangean dual solution. On randomly generated problems with up to approximately 200 nodes and 1100 arcs, they report solution times of a few seconds, in many cases two orders of magnitude faster than the method of Yen applied to the original costs. Beasley and Christofides (1989) solve the same Lagrangean dual problem, generalized to the RCSPP, using subgradient optimization, and apply a branch and bound procedure to close the duality gap (Lagrangean dual values are used as lower bounds at each node of the tree). With 10 resources and on problems with up to 500 nodes and 5000 arcs, they report solution times of within 30 seconds. In other work making use of this Lagrangean relaxation, Ribeiro and Minoux (1985) propose a heuristic algorithm for a slight generalization of the WCSPP. They solve the Lagrangean dual problem, but only on a 'partial' graph, with a parametric approach. Interestingly, it can be seen from their results that the Lagrangean dual can be solved with worst-case complexity $\mathcal{O}(n|A|W)$ using the parametric approach. They test their method on 10 transformed (hard) knapsack problems, having up to 50 000 nodes, and 100 000 arcs. For one version of the method they are able to obtain optimal solutions for all 10 instances; however they recommend a different version, which achieved a 9 out of 10 rate, but took less time. They do not report actual running times.

A third approach to solving the WCSPP is to use approximation. This approach was first developed in the context of bi-criteria problems, and multi-objective problems, and later specialized to WCSPP. In early work on bi-criteria problems, Hansen (1979) took advantage of the pseudopolynomial time algorithms available for WCSPP, combining with scaling and rounding to produce a fully polynomial approximation scheme with complexity $\mathcal{O}(|A|^2 n^2(\log(n^2/\epsilon))(1/\epsilon))$. Later, Warburton (1987) developed a similar idea for multi-objective problems. Warburton also described a simple application to the WCSPP, in which costs are scaled and rounded, and sub-intervals of the cost range systematically searched, to yield a fully polynomial approximation algorithm with complexity $\mathcal{O}((\log U)n^3(\log n)1/\epsilon)$, where U is an upper bound on the cost of the optimal path and ϵ is the required accuracy. The first direct approximation approach to WCSPP was developed by Hassin (1992): two fully polynomial approximation schemes for WCSPP based on cost scaling and rounding are given. The first uses a kind of geometric bisection search whilst the second iteratively extends paths which are in some sense approximately efficient. The first has a reported complexity of $\mathcal{O}(\log\log U(|A|n(1/\epsilon) + \log\log U))$, while the second has a complexity of $\mathcal{O}(|A|n^2(\log(n/\epsilon))1/\epsilon)$. (Note this is a clear improvement over the method of Hansen (1979).) The advantage of the latter is that it is independent of cost data (the bound depends on cost data); however, if U is relatively modest, the former will have a better complexity. In this paper we present the former method in some detail. As far as we are aware, no computational experience has been reported for any approximation algorithm.

Finally, we note that good pre-processing techniques can reduce the size of the problem, and also detect infeasibility. In some cases a problem may even be solved in pre-processing. However only a few authors discuss it. Aneja et al. (1983) give a thorough discussion, which we repeat here, but do not report the effects of these techniques on computation. However, Beasley and Christofides (1989) use similar techniques and report typical reductions in graph size of 10–20%. They also give a specialized form of reduced cost variable fixing: they show, using reduced costs based on their optimal Lagrangean dual values, that some arcs can be eliminated from consideration, since any path using them must have cost greater than the cost of a known feasible path.

3. A review of representative methods

In this section we give a more detailed review of methods that appear to be representative of the literature. We begin by describing pre-processing ideas, including some put forward by Aneja et al. (1983). We then describe the label setting method of Desrochers and Soumis (1988a), the Lagrangean relaxation approach taken by Handler and Zang (1980), and conclude with the fully polynomial approximation scheme of Hassin (1992).

3.1. Pre-processing and problem simplification

A given WCSPP may be simplified in a number of ways. One of the few papers in the literature to discuss problem simplification and pre-processing techniques is that of Aneja et al. (1983). They present their techniques in the context of a RCSPP; here we will discuss their procedure in the special case that there is only one resource, ie, the problem is a WCSPP. We also suggest a slight extension to their procedure.

These pre-processing ideas exploit the fact that finding the shortest path from s to every node $i \in V$ requires no more computational effort than finding the shortest path from s to t, and that finding the shortest path from every node $i \in V$ to t likewise requires no more computation than finding the shortest path from s to t. So with only four shortest path calculations we can obtain for each node $i \in V$:

$\sigma_i^w =$ length of shortest path from s to i with arc lengths given by weights w,
$\tau_i^w =$ length of shortest path from i to t with arc lengths given by weights w,
$\sigma_i^c =$ length of shortest path from s to i with arc lengths given by costs c, and
$\tau_i^c =$ length of shortest path from i to t with arc lengths given by costs c.

In addition, we ask that the shortest path procedure record a shortest path tree in each case.

Clearly if there is no path from s to t in the graph, or if $\sigma_t^w > W$ then the WCSPP must be infeasible. Also it is clear that if the path from s to t in the shortest path tree which yields σ_t^c is weight feasible then it must be an optimal solution to the WCSPP. In either case the WCSPP is solved. Otherwise we proceed with the pre-processing steps outlined below.

For each $i \in V \backslash \{s, t\}$ Aneja et al. observe that if $\sigma_i^w + \tau_i^w > W$ then i cannot appear in any weight feasible path from s to t, so i and all arcs incident to it may be removed from the graph. Furthermore, for each arc $(i, j) \in A$ they observe that if $\sigma_i^w + w_{ij} + \tau_j^w > W$, then arc (i, j) cannot appear in any weight feasible path from s to t, so it may be removed from the graph. We extend this idea to costs. We note that if U denotes the cost of the path that yielded σ_t^w, and if $\sigma_i^c + c_{ij} + \tau_j^c \geqslant U$ for some arc (i, j), then using arc (i, j) cannot yield a better solution than a known feasible solution, so (i, j) can be deleted from the graph.

If at least one node or arc has been removed from the graph, the shortest paths may have changed, so the entire procedure is repeated. Obviously this pre-processing procedure cannot be repeated more than $|A|$ times.

3.2. The label setting algorithm

Here we describe the label setting method of Desrochers and Soumis (1988a). The algorithm uses a set of *labels* for each node. Each label on a node corresponds to a different path from node s to that node, and consists of a pair of numbers representing the cost and the weight of the corresponding path. No labels having the same cost are stored, and for each label on a node, any other label on that node with lower cost must have a greater weight. We formalize these ideas below, where I_i is the index set of labels on node i and for each $k \in I_i$ there is a corresponding path P_i^k from s to i having weight W_i^k and cost C_i^k. We note that path weights and costs can easily be calculated for path $P_i^k = (s = j_0, j_1, \ldots, j_{n_i^k=1})$ by $W_i^k = w(P_i^k) = \sum_{m=1}^{n_i^k} w_{j_{m-1}j_m}$ and $C_i^k = c(P_i^k) = \sum_{m=1}^{n_i^k} c_{j_{m-1}j_m}$. We commonly refer to (W_i^k, C_i^k) as the *label* and to P_i^k as the *corresponding path*.

Definition 1. Let (W_i^k, C_i^k) and (W_i^l, C_i^l) be two labels at node i, corresponding to two different paths P_i^k, and P_i^l. Then we say that (W_i^k, C_i^k) *dominates* (W_i^l, C_i^l) if and only if $W_i^k \leqslant W_i^l$, $C_i^k \leqslant C_i^l$, and the labels are not equal.

Definition 2. A label (W_i^k, C_i^k) is said to be *efficient* if it is not dominated by any other label at node i, ie, if $(w(P), c(P))$ does not dominate (W_i^k, C_i^k) for any path P from s to i. A path is said to be efficient if the label it corresponds to is efficient.

The label setting algorithm finds all efficient labels on every node. Starting with no labels on any node, except for the label (0, 0) on node s, the algorithm extends the set of all labels by *treating* an existing label on a node, that is, by extending the corresponding path along all outgoing arcs. More formally, the *treatment* of a label (W_i^k, C_i^k) considers each arc $(i, j) \in \delta^+(i)$ such that $W_i^k + w_{ij} \leqslant W$: if $(W_i^k + w_{ij}, C_i^k + c_{ij})$ is not dominated by any label on node j, then it is added to the set of labels on node j, with corresponding path P_i^k extended by arc (i, j). The label setting algorithm, as it applies to the WCSPP, is given in Algorithm 1. We use L_i to denote the set of labels on node i and $T_i \subseteq I_i$ to index the labels on node i which have been treated.

Once all efficient labels have been generated, and the algorithm stops, the solution to the WCSPP is given by P_t^k for $k \in I_t$ having minimal cost C_t^k.

We note that this algorithm could be improved somewhat by integrating some pre-processing techniques. For example, in Step 1, instead of the test $W_i^k + w_{ij} \leqslant W$, the test $W_i^k + w_{ij} + \tau_j^w \leqslant W$ could be used instead. We also note that with the use of appropriate data structures, the complexity of this algorithm can be bounded by $\mathscr{O}(|A|W)$ (Desrochers and Soumis, 1988a). If there *are* zero weights, then the complexity is actually $\mathcal{O}(n^2 W)$ (one approach in this case is to treat labels with identical weight in increasing order of cost).

Algorithm 1. *The label setting algorithm*

```
Step 0: Initialization
        Set $L_s = \{(0, 0)\}$ and $L_i = \emptyset$ for all $i \in V \backslash \{s\}$.
        Initialize $I_i$ accordingly for each $i \in V$.
        Set $T_i = \emptyset$ for each $i \in V$.
Step 1: Selection of the label to be treated
        if $\bigcup_{i \in V}(I_i \backslash T_i) = \emptyset$ then STOP; all efficient labels have been generated
        else choose $i \in V$ and $k \in I_i \backslash T_i$ so that $W_i^k$ is minimal.
Step 2: Treatment of label $(W_i^k, C_i^k)$
        for all $(i, j) \in \delta^+(i)$ with $W_i^k + w_{ij} \leqslant W$ do
           if $(W_i^k + w_{ij}, C_i^k + c_{ij})$ is not dominated
              by $(W_j^l, C_j^l)$ for any $l \in I_j$
           then
              Set $L_j = L_j \cup \{(W_i^k + w_{ij}, C_i^k + c_{ij})\}$.
              Update $I_j$ accordingly.
        Set $T_i := T_i \cup \{k\}$.
        go to Step 1.
```

3.3. A Lagrangean relaxation approach

In this section we consider a Lagrangean relaxation approach to the WCSPP in which the weight constraint (2) is relaxed. Handler and Zang (1980) and Beasley and Christofides (1989) both give algorithms based on this relaxation, although the latter was given in the context of the RCSPP. In Handler (1980), Kelley's cutting plane method (1960) is specialized to solve the Lagrangean dual problem, while in Beasley (1989), subgradient optimization is used. As we will see, for the WCSPP, the Lagrangean dual can be solved to optimality very efficiently, so there would be little point in using subgradient optimization in the case of only one resource; here we focus on the cutting plane method used by Handler and Zang. (From results in Ribeiro (1985) we observe that a parametric solution of this Lagrangean dual takes $\mathcal{O}(n|A|W)$ time, but it is not known which approach performs best in practice.)

We define $\mathcal{P}$ to be the set of all $x \in \{0, 1\}^{|A|}$ that induce paths in G from s to t, ie, the set of all binary vectors satisfying (1). The Lagrangean relaxation we consider, denoted by $LR(\lambda)$, is $z_{LR}(\lambda) = \min_{x \in \mathcal{P}} \sum_{a \in A}(c_a + \lambda w_a)x_a - W\lambda$. The Lagrangean dual, LD, is $z_{LD} = \max_{\lambda \geqslant 0} z_{LR}(\lambda)$. Kelley's cutting plane method applied to this Lagrangean dual would begin with an initial set of paths, $\mathcal{Q} \subseteq \mathcal{P}$, chosen so that the problem $KCP(\mathcal{Q})$ defined by

$$z^{\mathcal{Q}} = \max_{\lambda \geqslant 0} \left(\min_{x \in \mathcal{Q}} \sum_{a \in A} (c_a + \lambda w_a)x_a - W\lambda \right)$$

was bounded above. Let $\lambda^{\mathcal{Q}}$ denote the optimal solution to $KCP(\mathcal{Q})$. Then an optimal solution $x^{\mathcal{Q}}$ for $LR(\lambda^{\mathcal{Q}})$ would be chosen. If $z_{LR}(\lambda^{\mathcal{Q}}) < z^{\mathcal{Q}}$ then $x^{\mathcal{Q}}$ would be added to $\mathcal{Q}$, $KCP(\mathcal{Q})$ re-solved, and the whole procedure repeated; otherwise the method would stop with the observation that $z^{\mathcal{Q}} = z^{\mathcal{P}} = z_{LD}$, so the Lagrangean dual has been solved as required. In most applications of Kelley's cutting plane method, $KCP(\mathcal{Q})$ is solved by solving a linear program (note that $\mathcal{Q}$ is a finite set). However Handler and Zang (1980) show how Kelley's cutting plane method can be efficiently specialized to this WCSPP Lagrangean dual, so that no linear programs need to be solved. We discuss this further below. Here we note that the only other computational requirement is the solution of $LR(\lambda^{\mathcal{Q}})$: this is simply the problem of finding a shortest path from s to t in G with arc lengths given by $c + \lambda^{\mathcal{Q}} w$ and so can be solved very efficiently.

Handler and Zang specialize Kelley's cutting plane method as follows. They begin by finding a shortest path from s to t with arc lengths given by c, and a shortest path from s to t with arc lengths given by w; we let x^1 denote the indicator vector of former path and x^2 denote the indicator vector of latter path. Note that x^1 also solves $z_{LR}(0)$. As was observed in our discussion of pre-processing, if $wx^1 \leqslant W$ then x^1 must be an optimal solution of the WCSPP while if $wx^2 > W$ the WCSPP must be infeasible; in either case the problem is solved. Note also that if $wx^2 \leqslant W$ and $cx^2 = cx^1$ then x^2 must be an optimal solution. So we assume otherwise, and have $wx^1 > W$, $wx^2 \leqslant W$ and $cx^1 < cx^2$. Writing $z^q(\lambda) = (wx^q - W)\lambda + cx^q$ for $q = 1, 2$, and initializing $\mathcal{Q}$ to $\{x^1, x^2\}$, we see that $KCP(\mathcal{Q})$ is the problem of maximizing the minimum of two linear functions of λ, $z^1(\lambda)$ and $z^2(\lambda)$, where one passes through $(0, cx^1)$ and has positive slope, and the other passes through $(0, cx^2)$ and has negative slope. Clearly an optimal solution of $KCP(\mathcal{Q})$ is given by the intersection of the two lines, at $\lambda^{\mathcal{Q}} = (cx^2 - cx^1)/(wx^1 - wx^2) > 0$. Now assuming we have not yet solved LD, so $z_{LD}(\lambda^{\mathcal{Q}}) < z^{\mathcal{Q}}$, then if $wx^{\mathcal{Q}} > W$, we can see that for any $\mathcal{Q}'$ containing $\{x^1, x^2, x^{\mathcal{Q}}\}$, the optimal solution to $KCP(\mathcal{Q}')$ will never have $z^1(\lambda)$ active at the solution. Thus we may simply *replace* x^1 in $\mathcal{Q}$ with $x^{\mathcal{Q}}$. Similarly if $wx^{\mathcal{Q}} \leqslant W$, $x^{\mathcal{Q}}$ may simply replace x^2 in $\mathcal{Q}$. So Kelley's cutting plane method becomes a simple matter of iteratively finding the intersections of pairs of linear functions of a single variable.

At the end of the Kelley's cutting plane method, the final value of $z^{\mathcal{Q}}$ is precisely z_{LD} and so provides a lower bound on the value of the WCSPP. Furthermore, x^2 is feasible, and so provides an upper bound; each time a new feasible solution is found in the course of the cutting plane method, this upper bound may be updated. Let λ^* denote the value of the optimal solution to LD, L denote the value of the lower bound and U denote the value of the upper bound at the end of the cutting plane method. Clearly if $U = L$ then we are done. Otherwise, there is a duality gap.

Handler and Zang (1980) close the duality gap by generating kth-shortest paths from s to t with arc lengths given by $c + \lambda^* w$. They observe that the two paths in $\mathcal{Q}$ at the end of the cutting plane method are shortest paths, and that a third may have been obtained in the last solution of $LR(\lambda)$. For larger k, they use the method of Yen (1971). Consider the sequence of paths generated in this way: $x^1, x^2, x^3, \ldots$. Now for each x^k in the sequence which is not weight feasible, $(c + \lambda^* w)x^k - W\lambda^*$ provides a lower bound on the value of cx^j for all $j > k$ with x^j weight feasible. This is obvious: $j > k$ so x^j is 'longer' than x^k, ie;

$$\begin{aligned}
(c + \lambda^* w)x^k &\leqslant (c + \lambda^* w)x^j \\
\Rightarrow (c + \lambda^* w)x^k - W\lambda^* &\leqslant (c + \lambda^* w)x^j - W\lambda^* \\
&= cx^j + \lambda^*(wx^j - W) \\
&\leqslant cx^j
\end{aligned}$$

since $\lambda^* \geqslant 0$ and x^j is weight feasible. So for each k in turn, if x^k is not weight feasible, we can update $L = (c + \lambda^* w)x^k - W\lambda^*$ and get an increasing sequence of lower bounds. Clearly whenever x^k is weight feasible and $cx^k < U$ we can update $U = cx^k$. The procedure finishes when either $L \geqslant U$, or all paths have been exhausted.

3.4. An approximation scheme based on cost scaling

In this section we consider the fully polynomial approximation scheme of Hassin (1992). We first introduce the notion of approximation algorithms and approximation schemes.

Let P be a minimization problem. Given $\epsilon > 0$, algorithm A is called an *ϵ-approximation algorithm* for P if it finds a feasible solution to any instance I of P with value

$$OPT(I) \leqslant A(I) \leqslant (1 + \epsilon)OPT(I),$$

where $OPT(I)$ is the optimal cost of instance I. An *approximation scheme* for P is a sequence of algorithms A_ϵ such that for all $\epsilon > 0$, A_ϵ is an ϵ-approximation algorithm for P. We say that A_ϵ is a *fully polynomial approximation scheme* (FPAS) if its complexity is a polynomial function of the length of the instance data and $1/\epsilon$.

Fully polynomial approximation schemes have been developed for many NP-hard problems which are solvable in pseudopolynomial time. Scaling and rounding techniques are used in most of these cases. For example, if an exact algorithm is available which is polynomial in the costs, but not the length of the cost data, and polynomial in the length of all other data, then the costs may be scaled and rounded down, so that they become ‘small’, ie, are polynomial in the length of the other data, and in $1/\epsilon$. The exact algorithm applied to these new costs will thus run in time polynomial in the length of the problem data. However the solution obtained will not necessarily be a solution to the original problem, it will be an ‘approximate’ one, with an error which can be bounded.

The exact algorithm used by Hassin in his FPAS is a dynamic programming algorithm, which takes as input an upper limit C on the cost of the path and calculates the minimum weight path from s to t with cost at most C. In fact the dynamic program given by Hassin can be seen to be equivalent to the label setting algorithm given in Algorithm 1, with costs and weights exchanged. Note that Hassin's algorithm is restricted to WCSPP's having positive costs.

Hassin's method begins by determining simple upper and lower bounds, U and L, on the value of the optimal solution. It then carries out a kind of geometric bisection search on the value of the optimal solution: taking $M = \sqrt{UL}$ it seeks to determine whether or not there is a weight feasible path with cost no greater than M. This is done approximately: costs are scaled and rounded according to $\bar{c}_a = \lfloor c_a(n-1)/(\epsilon M) \rfloor$ for all $a \in A$, and a minimum weight path having cost no more than scaled upper bound, $C = \lfloor (n-1)/\epsilon \rfloor$, is sought, using the label setting algorithm with costs and weights exchanged. Because of the scaling, this will run in time polynomial in $|A|C \leqslant |A|n1/\epsilon$ if there are no zero scaled costs, or $n^2C \leqslant n^3 1/\epsilon$ otherwise, which is polynomial in the length of the original problem data and $1/\epsilon$. Also because of the scaling, the result will be approximate: it is not difficult to show that if a weight feasible path with scaled cost no more than the (scaled) upper bound is found, then this path has original cost no more than $M(1+\epsilon)$, so the upper bound may be updated, $U = M(1+\epsilon)$. Furthermore, if there is no weight feasible path with scaled cost no more than the (scaled) upper bound, then there is no feasible path with original cost less than M, so the lower bound may be updated, $L = M$. Hassin suggests stopping when $U/L \leqslant 2$. At this point, costs are scaled and rounded according to $\bar{c}_a = \lfloor c_a(n-1)/(\epsilon L) \rfloor$ for all $a \in A$, and the label setting algorithm is run with weights, and costs exchanged, and weight limit $C = \lfloor 2(n-1)/\epsilon \rfloor$. From all the efficient labels at the destination node output by the label setting algorithm the one of minimum scaled cost and weight at most W is chosen. This label will correspond to an ϵ-optimal solution.

Algorithm 2 gives a slightly simplified form of Hassin's algorithm. Note that when the label setting algorithm is used in Step 1, with costs and weights exchanged, it can be stopped early, in fact it can be stopped as soon as a label with 'cost' no more than W has been generated on node t, since its purpose is only to determine if there *exists* a weight feasible path with cost no more than C. Also note that U and L may be initialized in a variety of ways: we suggest starting with U and L obtained in pre-processing. (Note we assume that pre-processing steps have determined that the problem is feasible, ie, that $\sigma_t^w \leqslant W$.)

It is not hard to determine that Steps 1 and 2 will be repeated no more than $\log\log(U/L)$ times. The dominant operations in Steps 1 and 2 are calculating $\sqrt{UL}$ and running the label setting algorithm. Hassin observes that the former can be approximated in $\mathcal{O}(\log\log U/L)$ steps, without affecting the complexity of the outer loop, while the latter requires $\mathcal{O}(|A|n1/\epsilon)$ steps to be performed, if there are no zero scaled costs, or $\mathcal{O}(n^3 1/\epsilon)$, otherwise. Thus Algorithm 2 has time complexity $\mathcal{O}(\log\log(U/L)(|A|n(1/\epsilon) + \log\log(U/L)))$, if there are no zero scaled costs, or $\mathcal{O}(\log\log(U/L)(n^3(1/\epsilon) + \log\log(U/L)))$, otherwise.

Algorithm 2. *A fully polynomial approximation scheme based on cost scaling*

```
Step 0: Initialize a lower bound $L$ and an upper bound $U$ on the optimum cost
        Set $C = \lfloor n - 1/\epsilon \rfloor$.
Step 1: **if** $(U/L) \leqslant 2$ **then go to** Step 3.
        Set $M = \sqrt{UL}$.
        **for** each $a \in A$ **do** set $\bar{c}_a = \lfloor c_a(n-1)/(\epsilon M) \rfloor$.
        Run the label setting algorithm with weights $\bar{c}$, costs $w$ and weight limit $C$.
Step 2: **if** there exists a feasible path with cost $\leqslant C$ **then** set $U = M(1+\epsilon)$
        **else** set $L = M$.
        **go to** Step 1.
Step 3: **for** each $a \in A$ **do** set $\bar{c}_a = \lfloor c_a(n-1)/(\epsilon L) \rfloor$.
        Set $C = \lfloor 2(n-1)/\epsilon \rfloor$.
        Run the label setting algorithm with weights $\bar{c}$, costs $w$ and weight limit $C$.
        Choose $k \in I_t$ so that $\bar{c}(P_t^k)$ is minim and $w(P_t^k) \leqslant W$.
        $P_t^k$ is an $\epsilon$-approximate solution.
```

4. An exact algorithm using weight scaling

In the previous sections we have seen that Hassin's approximation algorithm suffers from a complexity dependent on the magnitude of the costs, the label setting algorithm, while coping with costs of large magnitude without affecting its complexity, does not provide lower bounds, and the Lagrangean relaxation approach of Handler and Zang, while it does provide consistently improving lower bounds, does not have any complexity guarantees and may have to resort to an exponential algorithm to close the duality gap. An effective method would be one which could handle costs which may have large magnitude, and provided good lower bounds in modest time.

Here we develop an exact algorithm which consistently improves the lower bound, and can handle costs of large magnitude without effect on its complexity. The core of our algorithm is weight scaling: we scale the weights and then apply the label setting algorithm. The paths generated may not be weight feasible, but any weight infeasible path found provides a lower bound, and any feasible path found provides an upper bound. Unfortunately, it is not possible to guarantee that any weight feasible path will be found, nor is it possible to bound the deviation of the lower bound from the optimal cost, unless the scaling factor is sufficiently small. Thus we propose to systematically reduce the scaling factor, and repeat the procedure, until either our bounds are tight, or the scaling factor has reached the critical size. In the latter case we are guaranteed to find the optimal solution, since we have basically reverted to solving the original problem by label setting. However we hope that in practice, and on average, good lower bounds and even optimal solutions can be found well before this point is reached.

We specify our method in Algorithm 3. Here we use the notation $d(P)$, where $d \in \mathbb{R}^{|A|}$ is any function on the arcs and $P = (s = i_0, i_1, \ldots, i_m = t)$ is any path from s to t, to give the sum of d values on the arcs of the path, ie, $d(P) = \sum_{l=1}^{m} d_{i_{l-1}i_l}$. We will make particular use of the weight of a path, $w(P)$, and the cost of a path, $c(P)$. We assume that pre-processing steps have been performed, so that WCSPP is known to be feasible.

Before demonstrating correctness of our algorithm and discussing its complexity, we illustrate, with a small example, why it is necessary to consider ϵ as small as $1/W$, and why the deviation of the lower bound from the optimal cost cannot be meaningfully bounded otherwise. Consider the graph with $n = 3$, $A = \{(1, 2), (1, 3), (2, 3)\}$, $s = 1$, $t = 3$, W odd, $W \geqslant 3$, $w = ((W + 1)/2, W, (W + 1)/2)$ and $c = (0, M, 0)$ for $M > 0$ large. For any $\epsilon > (1/W)$, $\overline{w}_{12} = \overline{w}_{23} \leqslant \frac{1}{2}\overline{W}$, so the path (1, 2, 3) is feasible for the scaled problem. Furthermore $\overline{w}_{13} = \overline{W}$, so the label $(\overline{w}_{12} + \overline{w}_{23}, 0)$ for path (1, 2, 3) dominates the label $(\overline{W}, M)$ for the path (1, 3): the label setting algorithm cannot return a weight feasible path, and cannot return a lower bound of value any closer to the optimal value M, than 0.

Algorithm 3. *An exact algorithm using weight scaling*

```
Initialize a lower bound $L$ and an upper bound $U$ on the optimal cost
Set $r = \lfloor \log W \rfloor - 1$.
while $U > L$ and $r \geqslant 0$ do
    Set $\epsilon = (2^r/W)$.
    for each $a \in A$ do set $\overline{w}_a = \lfloor w_a/(\epsilon W) \rfloor$.
    Set $\overline{W} = \lfloor 1/\epsilon \rfloor$.
    Run the label setting algorithm with weights $\overline{w}$, costs $c$ and weight limit $\overline{W}$.
    Let $I_t$ index the final set of labels on node $t$ generated by the label setting algorithm.
    if $w(P_t^k) \leqslant W$ for all $k \in I_t$ then
        Output the path $P_t^k$ with least cost $C_t^k$ of any $k \in I_t$ and STOP.
    Set $L' = \min\{C_t^k | k \in I_t\}$.
    Update $L = \max(L, L'\}$.
    if $w(P_t^k) \leqslant W$ for some $k \in I_t$ then
        Set $U' = \min\{C_t^k | k \in I_t$ and $w(P_t^k) \leqslant W\}$.
        Update $U = \min\{U, U'\}$.
    Set $r = r - 1$.
  **if** $U = L$ **then** output the path that gave rise to $U$ and STOP.
```

We now prove the correctness of our algorithm. First we show that if at some iteration, we find $w(P_t^k) \leqslant W$ for all $k \in I_t$ then the path P_t^k with least cost C_t^k of any $k \in I_t$ is an optimal path for the WCSPP.

We use the fact that the scaling and rounding operation preserves weight feasibility.

Lemma 1. *If P is a path from s to t in G with $w(P) \leqslant W$ then $\overline{w}(P) \leqslant \overline{W}$, where $\overline{w}$ and $\overline{W}$ are as defined in Algorithm 3.*

Proof. Let $A(P)$ denote the arcs in path P, so if $P = (s = i_0, i_1, \ldots, i_m = t)$ then $A(P) = \{(i_{l-1}, i_l) | l = 1, \ldots, m\}$. Then for P with $w(P) \leqslant W$,

$$\overline{w}(P) = \sum_{a \in A(P)} \left\lfloor \frac{w_a}{\epsilon W} \right\rfloor \leqslant \left\lfloor \frac{1}{\epsilon W} \sum_{a \in A(P)} w_a \right\rfloor = \left\lfloor \frac{1}{\epsilon W} w(P) \right\rfloor \leqslant \left\lfloor \frac{1}{\epsilon} \right\rfloor = \overline{W}$$

and the result is proved. ■

We also use the following observation which follows from the previous lemma.

Lemma 2. *Let I_t index the set of labels on node t generated by the label setting algorithm for a WCSPP. Then for P any path from s to t in G with $w(P) \leqslant W$, either $(W_t^k, C_t^k) = (w(P), c(P))$ for some $k \in I_t$, or (W_t^k, C_t^k) dominates $(w(P), c(P))$ for some $k \in I_t$.*

We can now prove that if Algorithm 3 terminates in the ‘while’ loop then the path output must be an optimal path.

Proposition 1. *If $w(P_t^k) \leqslant W$ for all $k \in I_t$, where I_t indexes the final set of labels on node t generated by the label setting algorithm applied with weights $\overline{w}$ and weight limit $\overline{W}$, then $P_t^{k^*}$ is an optimal path for the WCSPP, for some $k^* \in \arg\min_{k \in I_t} C_t^k$.*

Proof. Let P^* be an optimal path for the WCSPP. Then for all $k \in I_t$, since P_t^k is weight feasible, it must be that $c(P_t^k) \geqslant c(P^*)$. Now by Lemma 1, $\overline{w}(P^*) \leqslant \overline{W}$, so by Lemma 2, either $(\overline{W}_t^k, C_t^k) = (\overline{w}(P^*), c(P^*))$ for some $k \in I_t$, or $(\overline{W}_t^k, C_t^k)$ dominates $(\overline{w}(P^*), c(P^*))$ for some $k \in I_t$. In either case there exists $k \in I_t$ with $C_t^k \leqslant c(P^*)$, so there must exist $k^* \in \arg\min_{k \in I_t} C_t^k$ with $C^{k^*} = c(P^*)$, ie, with $P_t^{k^*}$ optimal for the WCSPP. ■

We now prove the validity of our lower bound. We use the following property of the label setting algorithm.

Proposition 2. *If I_t indexes the final set of labels on node t generated by the label setting algorithm applied with weights $\overline{w}$ and weight limit $\overline{W}$, and $k \in I_t$ is the index of the path of minimum cost corresponding to a label in I_t, then C_t^k is a lower bound on the optimal value of the WCSPP.*

Proof. Let P^* be an optimal path for the WCSPP. From Lemma 2 it follows that either $(W_t^l, C_t^l) = (\overline{w}(P^*), c(P^*))$ for some $k \in I_t$, or (W_t^l, C_t^l) dominates $(\overline{w}(P^*), c(P^*))$ for some $l \in I_t$. In either case, there is $l \in I_t$ such that $C_t^l \leqslant c(P^*)$. Hence $C_t^k \leqslant C_t^l \leqslant c(P^*)$.

Therefore C_t^k is a lower bound on the optimal solution. ■

From Proposition 2 it follows immediately that the lower bound defined in the weight scaling algorithm is valid: it is the maximum of two valid lower bounds.

It is obvious that our upper bound is valid: it is the cost of some weight feasible path. So to prove correctness of our method, it remains only to prove that when $r = 0$, ie, $\epsilon = (1/W)$, that $w(P_t^k) \leqslant W$ for all $k \in I_t$ and so by Proposition 1, the algorithm must terminate with the optimal path. But this is obvious, as w and W are integer, so $\overline{w}_a = \lfloor w_a/(\epsilon W) \rfloor = \lfloor w_a \rfloor = w_a$ for all $a \in A$, and $\overline{W} = \lfloor 1/\epsilon \rfloor = \lfloor W \rfloor = W$, so the label setting algorithm is run on the original problem data, and must generate the optimal path. We have thus proved the following theorem.

Theorem 1. *Algorithm 3 terminates with the optimal solution to the WCSPP.*

The worst-case complexity of Algorithm 3 is not hard to determine. The dominant operation is running the label setting algorithm at each iteration of the 'while' loop: this requires $\mathcal{O}(|A|\overline{W}) = \mathcal{O}(|A|W/2^r)$ calculations, if there are no zero scaled weights, or $\mathcal{O}(n^2\overline{W}) = \mathcal{O}(n^2(W/2^r))$ calculations, otherwise. In the worst case, we perform all $R + 1$ iterations, where $R = \lfloor \log W \rfloor - 1$, giving a complexity of order

$$|A|\frac{W}{2^R} + \cdots + |A|\frac{W}{4} + |A|\frac{W}{2} + |A|W = |A|W\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^R}\right) \leqslant 2|A|W,$$

if there are no zero scaled weights, or

$$n^2\frac{W}{2^R} + \cdots + n^2\frac{W}{4} + n^2\frac{W}{2} + n^2W = n^2W\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^R}\right) \leqslant 2n^2W,$$

otherwise.

Thus the worst-case complexity is $\mathcal{O}(|A|W)$, if there are no zero scaled weights, or $\mathcal{O}(n^2W)$, otherwise. We note that in fact other choices of R are possible. It is our intention in future research to experiment with this algorithm numerically; finding effective choices for R will be one subject of our investigation. We also note that it is possible to take $\epsilon = (\beta^r/W)$ for any $\beta > 1$, (in this case we would take $R = \lfloor(\log W/\log\beta)\rfloor$), although as β decreases from 2, the complexity will worsen. We will also investigate 'jumping over' small values of r: we will include a threshold T so that if r falls below T, r is immediately set to 0, ie, we will replace the statement decrementing r with

if $r = T$ **then** set $r = 0$ **else** set $r = r - 1$

immediately before the end of the 'while' loop. Effective choices of T, as well as β, will also be a subject of future investigation.

5. Conclusions and further research

We have presented a review of methods for solving the WCSPP, including methods for solving closely related problems that can be applied to the WCSPP. The three most prevalent classes of method appearing in the literature are labelling-based, or dynamic programming, methods; methods based on Lagrangean relaxation of the weight constraint; and polynomial approximation algorithms based on cost scaling and rounding. The first and third classes enjoy worst-case complexity guarantees (of course these are pseudopolynomial for the former). The last two classes generate successively better lower bounds as the algorithm proceeds. Experimental results have been reported for the first two classes, for a variety of different generalizations of WCSPP, which makes comparison difficult. However it is clear that labelling methods, in particular, are effective for very large problems.

We have discussed in detail three representative approaches, as well as some pre-processing techniques, and have indicated some of the common threads underlying the different approaches.

We note that none of the exact methods we are aware of in the literature have pseudopolynomial worst-case complexity guarantees *and* generate steadily improving lower bounds.

In future work, we would like to see a comprehensive numerical comparison of the different approaches conducted, on the same class of base problems, in particular on the WCSPP. We also intend to experiment numerically with the algorithm we propose here, to determine effective parameter settings. Finally, we observe that both Hassin's approximation scheme given in Section 3.4, and our exact algorithm, employ Algorithm 1 many times in succession, each time with different weights. Thus efficient re-solution could substantially improve the running times of both methods, in practice. We intend to investigate this in future research.

Aknowledgements

We thank Refael Hassin and Danny Raz for their helpful discussions and insights into the cost scaling algorithm.

References

Aneja, Y.P., Aggarwal, V., Nair, K.P.K., 1983. Shortest Chain Subject to Side Constraints. *Networks* 13, 295–302.

Barnhart, C., Boland, N., Clarke, L., Johnson, E.L., Nemhauser, G.L., Shenoi, R.G., 1998. Flight String Models for Aircraft Fleeting and Routing. *Transportation Science* 32, 208–220.

Beasley, J.E., Christofides, N., 1989. An Algorithm for the Resource Constrained Shortest Path Problem. *Networks* 19, 379–394.

Boland, N., Clarke, L.W., Nemhauser, G.L. (forthcoming). The Asymmetric Traveling Salesman Problem with Replenishment Arcs. In: R. Burkard, M. Labbe, T. Rammos, J. Sicilia (Eds.), *European Journal of Operational Research: Special Issue on Combinatorial Optimization*.

Desrochers, M., Soumis, F., 1988a. A Generalized Permanent Labeling Algorithm for the Shortest Path Problem with Time Windows. *INFOR* 26, 191–212.

Desrochers, M., Soumis, F., 1988b. A Reoptimization Algorithm for the Shortest Path Problem with Time Windows. *European Journal of Operational Research* 35, 242–254.

Desrochers, M., Desrosiers, J., Solomon, M., 1992. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research* 40, 342–354.

Desrosiers, J., Pelletier, P., Soumis, F., 1983. Plus Court Chemin avec Constraints d'Horaires. *RAIRO* 17, 357–377.

Desrosiers, J., Dumas, Y., Solomon, M., Soumis F., 1995. Time Constrained Routing and Scheduling. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (Eds.), *Handbook in Operations Research and Management Science 8: Network Routing*, North-Holland, Amsterdam, 35–139.

Garey, M.R., Johnson, D.S., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

Hansen, P., 1979. Bicriterion Path Problems. In: Fandel, G., Gal, T., (Eds.), *Multiple Criteria Decision Making Theory and Application: Lecture Notes in Economics and Mathematical Systems 177*. Springer-Verlag, Berlin.

Halpern, J., Priess, J., 1974. Shortest Paths with Time Constraints on Moving and Parking. *Networks* 4, 241–253.

Handler, G.Y., Zang, I., 1980. A Dual Algorithm for the Constrained Shortest Path Problem. *Networks* 10, 293–309.

Hassin, R., 1992. Approximation Schemes for the Restricted Shortest Path Problem. *Mathematics of Operations Research* 17, 36–42.

Jaumard, B., Semet, F., Vovor, T., 1996. A Two-Phase Resource Constrained Shortest Path Algorithm for Acyclic Graphs. *Les Cahier du GERAD*, G-96–48.

Joksch, H.C., 1966. The Shortest Route Problem with Constraints. *Journal of Mathematical Analysis and Applications* 14, 191–197.

Kelley, J.E., 1960. The Cutting Plane Method for Solving Convex Programs. *Journal of the SIAM* 8, 703–712.

Lawler, E.L., 1976. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart & Winston, New York.

Ribeiro, C., Minoux, M., 1985. A Heuristic Approach to Hard Constrained Shortest Path Problems. *Discrete Applied Mathematics* 10, 125–137.

Warburton, A., 1987. Approximation of Pareto Optima in Multiple-Objective, Shortest-Path Problems. *Operations Research* 35, 70–79.

Yen, J.Y., 1971. Finding the k-Shortest, Loopless Paths in a Network. *Management Science* 17, 711–715.