Please cite this paper as:

A.N. Letchford & S.D. Nasiri (2014) Pricing routines for vehicle routing with time windows on road networks. LUMS Working Paper 2014:2, Lancaster University Management School, Lancaster, United Kingdom.

Lancaster University Management School
Working Paper 2014:2

Pricing Routines for Vehicle Routing with Time Windows on Road Networks

Adam N. Letchford & Saeideh D. Nasiri

The Department of Management Science
Lancaster University Management School
Lancaster LA1 4YX
UK

Pricing Routines for Vehicle Routing with Time Windows on Road Networks

Saeideh D. Nasiri* Adam N. Letchford*

January 2014

Abstract

Several very effective exact algorithms have been developed for vehicle routing problems with time windows. Unfortunately, most of these algorithms cannot be applied to instances that are defined on road networks, because they implicitly assume that the cheapest path between two customers is equal to the quickest path. Garaix and co-authors proposed to tackle this issue by first storing alternative paths in an auxiliary multi-graph, and then using that multi-graph within a branch-and-price algorithm. We show that, if one works with the original road network rather than the multi-graph, then one can solve the pricing subproblem more quickly, in both theory and practice.

Keywords: Vehicle routing, combinatorial optimization, bi-criteria shortest paths.

1 Introduction

Vehicle Routing Problems (VRPs) are a much-studied class of combinatorial optimization problems, and several books have been written about them (e.g., [3, 25, 26, 39]). Many VRPs arising in practical applications involve restrictions on the time at which service begins at the customers. Well-known examples include the *Traveling Salesman Problem with Time Windows* or TSPTW [4], the *Multiple Traveling Salesman Problem with Time Windows* or m-TSPTW [14, 37], and the *Vehicle Routing Problem with Time Windows* or VRPTW [15, 38]. Good surveys on such problems include [10, 17].

As mentioned in the above-mentioned books and surveys, there are several effective exact algorithms available to solve such time-constrained VRPs, some of which are capable of routinely solving instances with up to around 100 customers to proven optimality. There are also several effective heuristics that are able to provide good solutions for even larger instances.

*Department of Management Science, Lancaster University Management School, Lancaster LA1 4YX, United Kingdom. E-mail: {S.D.Nasiri,A.N.Letchford}@lancaster.ac.uk

Unfortunately, and surprisingly, most of these algorithms are based on a key assumption that is not guaranteed to hold in the real world. This assumption is that, for all ordered pairs (i, j) of nodes (that represent either depots or customers), one is provided with two numbers: c_{ij}, the cost of traveling from i to j, and t_{ij}, the time taken to travel from i to j. In reality, however, many VRPs are concerned with the routing of vehicles on *road networks*. In a real-life road network, the cheapest path between two points is unlikely to be the same as the quickest path. Therefore, to model and solve time-constrained VRPs on road networks correctly, one should take into account the trade-off between travel costs and travel times.

This issue was explained in detail in a recent paper by Garaix *et al.* [23], who proposed to remedy the situation as follows. First, in a pre-processing stage, they solve a series of *bicriteria shortest path* problems in the road network. This yields, for each pair of customer and/or depot nodes in the road network, a set of paths that completely represent the cost-time trade-off. These paths are stored in an auxiliary *multi-graph*. Then, they use a traditional branch-and-price algorithm, but solve the pricing subproblem on the multi-graph rather than the original road network. Finally, dynamic programming is used to convert the optimal solution into a collection of feasible routes in the road network.

Although the approach of Garaix *et al.* [23] is elegant, it does require the use of specialised techniques for constructing the multi-graph, solving the pricing problem, and converting the solution. Moreover, as we explain in Section 3, constructing and storing the multi-graph can take exponential time and space in the worst case. These considerations led us to develop more 'natural' algorithms for the pricing subproblem, that work directly on the original road network. It turns out that these natural algorithms, as well as being simpler, are faster in both theory and practice.

The paper is organized as follows. In Section 2, we review the relevant literature. In Section 3, we present a result about the size of the multi-graph. In Section 4, we present our first pricing routine, which is designed for the case of 'elementary' routes. We also show that we obtain, as a by-product, an exact algorithm for the 'road network' version of the TSPTW. In Section 5, we present our second pricing routine, for the case in which 'non-elementary' routes are permitted, and show that it is faster (in the worst case) than the one of Garaix *et al.* [23]. In Section 6, we show that it is faster also in practical computations. Finally, some concluding remarks appear in Section 7.

2 Literature Review

We now review the relevant literature. Subsection 2.1 deals with the TSPTW, m-TSPTW and VRPTW, Subsection 2.2 covers VRPs on road networks,

and Subsection 2.3 focuses on VRPs with time windows on road networks.

2.1 Standard VRPs with time windows

The TSPTW is defined as follows [4]. Let G be a complete directed graph with vertex set $V = \{0, 1, \ldots, n\}$ and arc set A. Vertex 0 represents the depot and the other vertices represent customers. For each $(i, j) \in A$ we are given a cost c_{ij} and a traversing time t_{ij}. Each customer i has a time window $[e_i, \ell_i]$ and a service time s_i. Service at customer i must start no earlier than e_i and no later than ℓ_i. If the vehicle arrives at customer i before e_i, it has to wait. The vehicle departs from the depot at time 0 and must return to the depot by time T. The objective is to find a minimum cost route that services each customer once and satisfies the time window requirements. It is usually assumed that all costs and times are positive integers.

The m-TSPTW is identical to the TSPTW, except that there are several vehicles and each customer must be visited by exactly one vehicle [14, 37]. The VRPTW is similar, except that each customer i has a positive integral demand q_i and the total load of each vehicle must not exceed some positive integral capacity Q [15, 16].

Exact approaches to the TSPTW include, e.g., dynamic programming (DP) [20], hybrid DP / branch-and-bound [6], constraint programming [35] and branch-and-cut [2, 12]. Exact approaches to the multi-vehicle problems include, e.g., Lagrangian relaxation [29], branch-and-cut [33], branch-and-price [8, 14, 16, 21], branch-cut-and-price [13, 28, 30] and, very recently, hybrid DP / dual ascent / branch-and-bound [5].

All of the exact approaches to the multi-vehicle problems are based, either explicitly or implicitly, on *set covering* or *set partitioning* formulations. The set covering formulation takes the form

$$\begin{aligned} \min \quad & \textstyle\sum_{r\in\Omega} c_r\lambda_r && && (1)\\ \text{s.t.} \quad & \textstyle\sum_{r\in\Omega} a_{ir}\lambda_r \geq 1 && (\forall i \in V \setminus \{0\}) && (2)\\ & \lambda_r \in \{0,1\} && (\forall r \in \Omega), && (3) \end{aligned}$$

where Ω denotes the set of all feasible routes for a single vehicle, c_r denotes the cost of route r, λ_r is a binary variable taking the value 1 if and only if a vehicle uses route r, and a_{ir} is a binary constant, taking the value 1 if and only if customer i is serviced by route r. The set partitioning formulation is identical, except that the inequalities (2) are changed to equations. (If the costs and times obey the triangle inequality, which is usually the case in practice, this change affects neither the optimal solution, nor the lower bound from the LP relaxation.)

Since $|\Omega|$ can be exponentially large, if one wishes to solve the LP relaxation of (1)–(3) exactly, it must be solved via *column generation*, i.e, the variant of the simplex method in which columns of negative reduced

cost are generated on-the-fly via pricing routines. The pricing subproblem is strongly $\mathcal{NP}$-hard [18], but can be solved in pseudo-polynomial time if one permits *non-elementary* routes, i.e., routes that visit customers more than once [14, 16]. The resulting enlargement of the column set Ω does not change the validity of the set covering / partitioning formulations, but it weakens the lower bound obtained when one solves the LP relaxation. As a compromise, one can forbid some non-elementary routes but not others (e.g., [5, 13]).

2.2 Routing on road networks

All of the approaches mentioned in the previous subsection assume that the instance is defined on a complete directed graph. Most VRPs arising in practice, however, take place on road networks. It is usually possible to transform a VRP on a road network into a VRP on a complete graph, via a series of shortest-path computations (e.g., [1, 11, 22]). Nevertheless, it can be preferable to work with the original road network, in an attempt to exploit any properties, such as sparsity or planarity, that it may have (e.g., [11, 22, 31, 32]).

Much of the literature on VRPs on road networks has been concerned with so-called *arc routing* problems, in which the customers are located along the edges or arcs of the network, rather than at nodes. For brevity, we do not review the arc routing literature here, and refer the reader to the book [19]. We mention however that the paper [32], concerned with the so-called *Capacitated Arc Routing Problem* (CARP), can be viewed as a companion paper to the present one. It shows that pricing for the CARP can be performed more quickly if one works on the original road network rather than on a complete graph. The pricing routines presented in [32] are however quite different from the ones presented here. In particular, the routine for elementary routes in [32] is based on integer programming, whereas the one we present in Section 4 is based on dynamic programming. Moreover, the routine for non-elementary routes in [32] is slower and more complex than the one we present in Section 5, due to the fact that the vehicle load does not change when an edge is traversed.

We now return to node routing. The following 'road network' version of the TSP was defined in [11, 22, 34]. We are given:

- an undirected road network $\tilde{G} = (\tilde{V}, \tilde{E})$,
- a specified depot node, say node 0,
- a set of customer nodes $C \subset \tilde{V} \setminus \{0\}$,
- a cost $\tilde{c}_e$ for each $e \in E$.

The task is to find a minimum-cost tour, starting and ending at the depot, that passes through each customer node at least once. Nodes may be visited more than once, and edges may be traversed more than once, if desired. (Note that nodes in $\tilde{V} \setminus (C \cup \{0\})$ represent road junctions.)

Following [11, 31], we call the above variant of the TSP the *Steiner* TSP. In [9, 22], the Steiner TSP is formulated as an integer program with $\mathcal{O}(|\tilde{E}|)$ variables and an exponential number of constraints, and solved with cutting planes and branch-and-bound. In [31], it is formulated as an integer program with only $\mathcal{O}(|\tilde{E}|)$ variables and constraints, and solved via plain branch-and-bound.

Several generalisations of the Steiner TSP were also presented in [31]. Of relevance to us is the Steiner version of the TSPTW. This is like the Steiner TSP, but each customer $i \in C$ now has a time window $[e_i, \ell_i]$ and a service time s_i, and the vehicle must leave the depot at time 0 and return by time T. One can easily define Steiner versions of the m-TSPTW and VRPTW in a similar way. (In [31], the graph $\tilde{G}$ was assumed to be undirected, but one can also allow it to be directed, or mixed.)

2.3 Routing on road networks with time windows

Unfortunately, VRPs on road networks become significantly harder to model and solve when time windows are present. Indeed, in [31], we were able to formulate all problems considered as integer programs with only $\mathcal{O}(|\tilde{E}|)$ variables and constraints, *except* the Steiner TSPTW. The reason for this phenomenon is as follows: for the other problems considered in [31], it is never optimal for a vehicle to traverse an edge more than once in any given direction. When time windows are present, this is no longer true.

As mentioned in the introduction, another key paper in this regard is Garaix *et al.* [23]. They point out that, when time windows are present, one cannot always transform a 'road network' VRP into a standard VRP. This is because in a real-life road network the cheapest path between two points is unlikely to be the same as the quickest path.

To see this, consider the simple road network displayed in Figure 1. Suppose the depot is located at node 0,and the customers are located at nodes 2 and 4. Also suppose that we have (undirected) edges rather than (directed) arcs, for simplicity. For each of the seven edges, the corresponding traversing cost and time are displayed in parentheses. Between the depot and node 4, the cheapest path costs 2 and takes 4 time units, and the quickest path costs 4 and takes 2 time units. So there is no unique way to define c_{02} and t_{02}. A similar consideration applies to the paths between the two customers.

Garaix *et al.* proposed to remedy the situation with a three-phase procedure. In the first phase, they solve a series of *bicriteria shortest path* problems in $\tilde{G}$, in order to compute, for each ordered pair of nodes in $\{0\} \cup C$, a

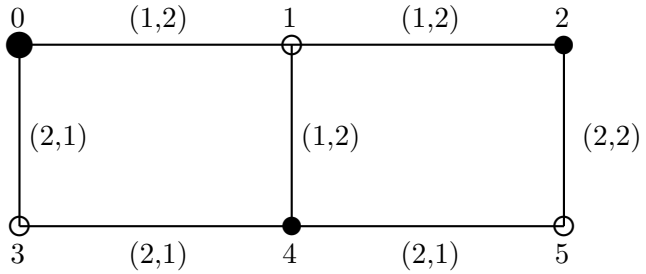

Figure 1: Example of a possible road network $\tilde{G}$.

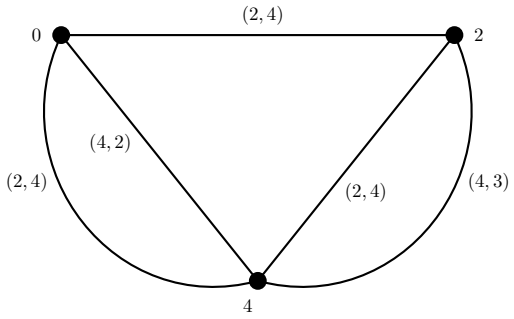

Figure 2: Corresponding multi-graph.

complete and non-redundant set of *efficient* (or *non-dominated*, or *pareto-optimal*) paths. (A variety of algorithms exist for generating such complete sets; see [36].) A (loopless and directed) multi-graph is then constructed, which has node set $\{0\} \cup C$ and, for each ordered pair of nodes, a set of parallel arcs, representing the corresponding complete set. (Figure 2 is the multi-graph corresponding to the road network in Figure 1, but with edges instead of arcs, for clarity.)

In the second phase, Garaix *et al.* formulate the problem as a set covering problem with a variable for each feasible route in the multi-graph, and solve it by branch-and-price. In the third and final phase, the optimal set covering solution is converted into an optimal solution for the original problem, via dynamic programming.

3 Drawbacks of the Multi-Graph Approach

Although the branch-and-price approach of Garaix *et al.* [23] performs reasonably well in practice, it seems both simpler and more natural to work with the original road network $\tilde{G}$ rather than the multi-graph. Moreover, the time and space requirements of the multi-graph approach can be very high in the worst case. Specifically:

- Computing complete sets from a source node to all other nodes takes $\mathcal{O}(|\tilde{A}|\, T)$ time (see [36]). Therefore, computing complete sets for all ordered pairs (i, j) with i and j in $C \cup \{0\}$ will take $\mathcal{O}(|C|\,|\tilde{A}|\, T)$ time.

- For a given ordered pair (i, j), the cardinality of a complete set can be as large as T [27]. So the number of arcs in the multi-graph can be exponential in the encoding length of T.

- Converting the solution for the transformed problem into a solution for the original problem takes $\mathcal{O}(m\,|\tilde{A}|\,T)$ time, where m is the number of vehicles used.

In fact, the situation is even worse than this, as shown by the following theorem.

Theorem 1 *The multi-graph can contain $\Theta(|C|^2\, T)$ arcs, even if the original graph $\tilde{G}$ contains only $\mathcal{O}(|C|\, \log T)$ arcs.*

Proof. Suppose initially that $|C| = 1$, i.e., there is just a single customer. Let k be any positive integer. Construct a graph $\tilde{G}^k$ as follows. The vertex set is $\{0, \ldots, 2k\}$, where 0 is the depot and $2k$ is the customer. For $i = 1, \ldots, k$, the arc set $\tilde{A}$ contains the arcs $(i-1, i)$, $(i-1, i+k)$ and $(i, i+k)$, together with arcs in the opposite direction. The arcs $(i-1, i)$ have cost $2 + 2^{i-1}$ and time 1. The arcs $(i-1, i+k)$ have cost 1 and time 1. The arcs $(i, i+k)$ have cost 1 and time 2^k. For each of these arcs, the arc going in the opposite direction has the same cost and time. (Figure 3 shows the graph $\tilde{G}^3$, but with pairs of opposite arcs shown as edges, to aid clarity.)

By construction, for any integer t between 0 and $2^k - 1$, there is a path from 0 to $2k$ in $\tilde{G}^k$ that has time $k+t$ and cost $2k+2^k-1-t$. Each of these paths is non-dominated. If we set $T = 2^{k+1}$, then each of these paths is also feasible, and therefore corresponds to an arc from 0 to $2k$ in the multi-graph. The multi-graph also contains analogous arcs from $2k$ to 0. Thus, the multi-graph contains $2^{k+1} = T$ arcs, yet $\tilde{G}^k$ contains only $3k = \mathcal{O}(\log T)$ arcs. This proves the result for $|C| = 1$.

To prove the result for $|C| > 1$, create $|C|$ copies of the graph $\tilde{G}^k$, one for each customer, and merge them into a single graph, by identifying the depot nodes. Now, between any pair of customers, for any integer t between 0 and $2^{k+1} - 2$, there is a non-dominated path between the customers that has time $2k + t$. (It suffices to concatenate a non-dominated path from the first customer to the depot, of time $k + \lfloor t/2 \rfloor$, with a non-dominated path from the depot to the other customer, of time $k + \lceil t/2 \rceil$.) Moreover, if we set $T = 2^{k+2}$, then each of these paths is feasible, and therefore corresponds to an arc in the multi-graph. Thus, the multi-graph contains $\Theta(|C|^2 T)$ arcs, yet the original graph contains only $3k|C| = \mathcal{O}(|C|\, \log T)$ arcs. □

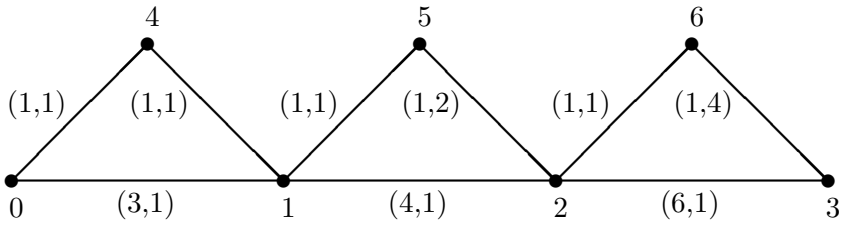

Figure 3: The graph $\tilde{G}^3$, with costs and times.

The above considerations suggest that it may be better to work directly with the original road network $\tilde{G}$ rather than with the multi-graph. So, suppose we have an instance of the 'Steiner' m-TSPTW or VRPTW, and consider once again the set covering formulation (1)–(3). In the Garaix *et al.* approach, Ω is assumed to contain all feasible routes in the multi-graph. We propose instead to assume that Ω contains all feasible routes in $\tilde{G}$. It is important to note that this change in the definition of Ω has no effect on the cost of the optimal solution to the set covering problem, nor on the lower bound from its LP relaxation. Indeed, given any feasible route in the multi-graph, there is a corresponding feasible route in $\tilde{G}$ of no greater cost that services the same set of customers, and vice-versa.

We remark that it is always valid to use set covering rather than set partitioning when working on $\tilde{G}$. Indeed, if an optimal solution to the set covering problem corresponds to a set of routes that collectively service a customer more than once, we can always take one of the routes that services that customer, and modify it so that the vehicle passes through the given customer node without actually servicing the customer. The modified route remains feasible and has the same cost as the original route. Repeating this procedure, if necessary, we obtain an optimal set of routes that collectively service each customer exactly once. (Incidentally, the same argument shows the validity of using set covering in the multi-graph approach.)

As in the case of the standard m-TSPTW and VRPTW [14, 16], one can if desired enlarge the column set Ω so that non-elementary routes are permitted. In the 'Steiner' context, this corresponds to permitting routes in $\tilde{G}$ that *service* some customers more than once. (Recall that even elementary routes in $\tilde{G}$ are permitted to *pass through* nodes more than once, regardless of whether or not they are customer nodes.) In the following two sections, we present pricing routines for the Steiner m-TSPTW and Steiner VRPTW for the cases of elementary and non-elementary routes, respectively.

4 Pricing Elementary Routes

In this section, we present an exact pricing routine for the Steiner m-TSPTW, for the case in which only elementary routes are permitted. We

also show how to adapt it to the Steiner VRPTW. We will see that these routines are much faster than the analogous routines based on the multigraph. Throughout, we assume for simplicity of notation that the graph $\tilde{G}$ is directed, with node set $\tilde{V}$ and arc set $\tilde{A}$. The routines can be easily modified for the case in which $\tilde{G}$ is undirected or mixed.

For a given $\Omega' \subset \Omega$, suppose that the LP relaxation of the formulation (1)–(3) has been solved, but only using columns in Ω'. For each $i \in C$, let $\pi_i \geq 0$ be the dual price of the corresponding constraint (2) in the solution to the relaxation. The reduced cost of the variable λ_r for a route $r \in \Omega \setminus \Omega'$ is equal to its true cost c_r, minus $\sum_{i \in C} a_{ir} \pi_i$. We now show that, in the case of the Steiner m-TSPTW, one can find routes of negative reduced cost in $\mathcal{O}\left(2^{|C|}\,|\tilde{A}|\,T\right)$ time.

For all $S \subseteq C$, all $i \in \tilde{V}$ and all $0 \leq t \leq T$, let $f(S,i,t)$ denote the reduced cost of the cheapest feasible path in $\tilde{G}$ (if any) that starts at the depot, services every customer in S exactly once, possibly visiting other nodes in $\tilde{V}$ along the way, and ends at node i at time t. (If no such feasible path exists, we view $f(S,i,t)$ as being infinite.) Also, for any given node $i \in \tilde{V}$, let $n^+(i)$ denote the set of *forward neighbors* of i in $\tilde{G}$, by which we mean the set of nodes j for which the arc (i,j) exists in $\tilde{A}$. One can then compute $f(S,i,t)$ for all feasible triples (S,i,t) as follows:

Set $f(\emptyset,0,0) = 0$.

For $t = 0,\ldots,T$ do:

  For all $i \in \tilde{V}$ do:

    For all $S \subseteq C$ do:

      For all $j \in n^+(i)$ such that $t + \tilde{t}_{ij} < T$ do:

        If $f(S,i,t) + \tilde{c}_{ij} < f(S,j,t+\tilde{t}_{ij})$,

          Set $f(S,j,t+\tilde{t}_{ij})$ to $f(S,i,t) + \tilde{c}_{ij}$.

      For all $j \in n^+(i) \cap (C \setminus S)$ such that $t + \tilde{t}_{ij} < e_j$ do:

        If $f(S,i,t) + \tilde{c}_{ij} - \pi_j < f(S \cup \{j\}, j, e_j + s_j)$,

          Set $f(S \cup \{j\}, j, e_j + s_j)$ to $f(S,i,t) + \tilde{c}_{ij} - \pi_j$.

      For all $j \in n^+(i) \cap (C \setminus S)$ such that $e_j \leq t + \tilde{t}_{ij} \leq \ell_j$ do:

        If $f(S,i,t) + \tilde{c}_{ij} - \pi_j < f(S \cup \{j\}, j, t + \tilde{t}_{ij} + s_j)$,

          Set $f(S \cup \{j\}, j, t + \tilde{t}_{ij} + s_j)$ to $f(S,i,t) + \tilde{c}_{ij} - \pi_j$.

In this DP algorithm, the three inner loops involving j correspond, respectively, to (i) travelling from i to j without servicing j, (ii) travelling from i to j, waiting, and then servicing j, and (iii) travelling from i to j and servicing j immediately. When the DP terminates, any pair (S,t) with $f(S,0,t) < 0$ corresponds to a column with negative reduced cost. To con-

struct such a column one simply traces the path backward from the end state $(S, 0, t)$ to the initial state $(\emptyset, 0, 0)$.

Note that, for a given t and S, each arc in $\tilde{A}$ is examined no more than three times. Since there are at most $T + 1$ choices for t and $2^{|C|}$ choices for S, the running time is $\mathcal{O}\left(2^{|C|}\,|\tilde{A}|\,T\right)$, as claimed. Although this running time is exponential in $|C|$, it compares very favourably with the running time that would be obtained if one used the multi-graph. Indeed, in that case, to compute $f(S, i, t)$ for a given S, i and t, one would have to examine every arc in the multi-graph whose head is i and whose tail is in $S \setminus \{i\}$. From the proof of Theorem 1, this would take $\mathcal{O}(|C|\,T)$ time. Since there are $\mathcal{O}\left(2^{|C|}\right)$ choices for S, $\mathcal{O}(|C|)$ choices for i, and T choices for t, the DP would take $\mathcal{O}\left(2^{|C|}\,|C|^2\,T^2\right)$ time.

As usual with DP algorithms (e.g., [8, 13, 20, 21]), some simple tests can be used to eliminate states from consideration. For example, suppose that we pre-compute, for all $i \in \tilde{V} \setminus \{0\}$, the time taken to travel from node i to the depot, via a quickest path. Then one can eliminate the state (S, i, t) if the time taken to travel from i to the depot exceeds $T - t$. Even with such additional tests, however, the algorithm can be expected to be viable for large values of $|C|$ only if the time windows are very narrow. This is because pricing typically has to be performed many times in a branch-and-price algorithm.

We remark that to adapt the above pricing routine to the Steiner VRPTW it suffices to introduce an additional state variable q, representing the cumulative load of the partial path, and add another 'for' loop in which q ranges from 0 to Q. The running time does however then increase by a factor of Q.

To close this section, we observe that the above pricing routine can easily be converted into an exact algorithm for the Steiner TSPTW. Indeed, if we set π_j to zero for all $j \in C$ and run the routine, then the cost of an optimal Steiner TSPTW solution is given by:

$$\min_{0<t\leq T} \{f(C, 0, t)\},$$

and one can construct such an optimal solution by tracing the path backward from the optimal final state to the initial state $(\emptyset, 0, 0)$. The running time remains unchanged at $\mathcal{O}\left(2^{|C|}\,|\tilde{A}|\,T\right)$. We imagine that, with the addition of some standard state-elimination and dominance tests (as in [20]), this could lead to a viable solution algorithm for the Steiner TSPTW, provided the time windows are reasonably narrow. We are not aware of any other exact algorithm for the Steiner TSPTW.

5 Pricing Non-Elementary Routes

In this section, we present exact pricing routines for the Steiner m-TSPTW and VRPTW, for the case in which non-elementary routes are permitted. Once again, we will see that the routines are much faster than the analogous routines based on the multi-graph.

The new pricing routine for the Steiner m-TSPTW is similar to the one described in the previous section, but we drop the index S from the state, since we no longer need to keep a record of which customers have already been serviced. Accordingly, for all $i \in \tilde{V}$ and all $0 \leq t \leq T$, let $f(i,t)$ denote the reduced cost of the cheapest feasible path (if any) that starts at the depot, services any customers any number of times, possibly visiting other nodes in $\tilde{V}$ along the way, and ends at node i at time t. One can compute $f(i,t)$ for all i and t as follows:

Set $f(0,0) = 0$.

For $t = 0, \ldots, T$ do:

  For all $i \in \tilde{V}$ do:

    For all $j \in n^+(i)$ such that $t + \tilde{t}_{ij} < T$ do:
      If $f(i,t) + \tilde{c}_{ij} < f(j, t + \tilde{t}_{ij})$,
        Set $f(j, t + \tilde{t}_{ij})$ to $f(i,t) + \tilde{c}_{ij}$.

    For all $j \in n^+(i) \cap C$ such that $t + \tilde{t}_{ij} < e_j$ do:
      If $f(i,t) + \tilde{c}_{ij} - \pi_j < f(j, e_j + s_j)$,
        Set $f(j, e_j + s_j)$ to $f(i,t) + \tilde{c}_{ij} - \pi_j$.

    For all $j \in n^+(i) \cap C$ such that $e_j \leq t + \tilde{t}_{ij} \leq \ell_j$ do:
      If $f(i,t) + \tilde{c}_{ij} - \pi_j < f(j, t + \tilde{t}_{ij} + s_j)$,
        Set $f(j, t + \tilde{t}_{ij} + s_j)$ to $f(i,t) + \tilde{c}_{ij} - \pi_j$.

The explanation of this DP routine is similar to that given in the previous section. When it terminates, any t with $f(0,t) < 0$ corresponds to a column with negative reduced cost.

This DP routine is easily seen to run in $\mathcal{O}(|\tilde{A}|\ T)$ time. As we will show in the next section, this is fast enough to be practically useful. It also compares very favourably, again, with that which would be obtained with the multi-graph approach, which can easily be shown to be $\mathcal{O}\left(|C|^2\, T^2\right)$.

As in the previous section, state-elimination and dominance tests could improve the practical performance of the routine, and one can adapt the routine to the Steiner VRPTW, at the cost of increasing the running time by a factor of Q.

6 Computational Experiments

In this section, we report on some computational experiments. Subsection 6.1 explains how we created a collection of sparse graphs for use in the experiments. Subsection 6.2 is concerned with the construction of the multi-graph and Subsection 6.3 is concerned with the pricing routines.

6.1 Construction of sparse graphs

To begin with, we constructed ten sparse, undirected graphs, each with its own edge costs. This was done using a similar procedure to the one used in our earlier paper [31], which was specifically designed to produce graphs that resemble real life road networks. In detail, the following was done for $n \in \{50, 100, 150, 200, 250\}$ and then for $n = \{100, 200, 300, 400, 500\}$:

1. Set $\tilde{V} = \{1, \ldots, n\}$ and $\tilde{E} = \emptyset$.
2. Place the n nodes at random in a circle of radius $10\sqrt{n}$.
3. Define the set of *potential edges* $P = \{\{i, j\} : 1 \leq i < j \leq n\}$.
4. For each $\{i, j\} \in P$, let the cost c_{ij} be the Euclidean distance between the end-nodes, rounded up to the nearest integer.
5. Sort the potential edges in non-decreasing order of cost.
6. Examine each edge in P in turn. Insert it into $\tilde{E}$ if both of the following conditions are satisfied:
   (a) It does not cross one of the edges already inserted into $\tilde{E}$.
   (b) It does not form an angle of less than 60° with an edge that has already been inserted into $\tilde{E}$ and with which it shares an end-node.
7. If there are no isolated nodes in the resulting graph, output the graph $(\tilde{V}, \tilde{E})$. Otherwise, repeat the procedure.

The only difference between this procedure and the one used in [31] is that, in step 2, the circle is of radius $10\sqrt{n}$, rather than 100. This change was made to ensure that the average edge cost is roughly the same across all instances, which is closer to what happens in real life.

For each of the ten graphs, one node was selected at random to be the depot. For the first five graphs, with $n \in \{50, \ldots, 250\}$, each non-depot node was then given a probability of $2/3$ of being required. For the other five graphs, with $n = \{100, \ldots, 500\}$, the probability was set to $1/3$.

Figure 4 shows the graph that was obtained for $n = 300$ and a probability of $1/3$. The required and non-required nodes are represented by small black and white circles, respectively, and the depot is represented by the small black square.

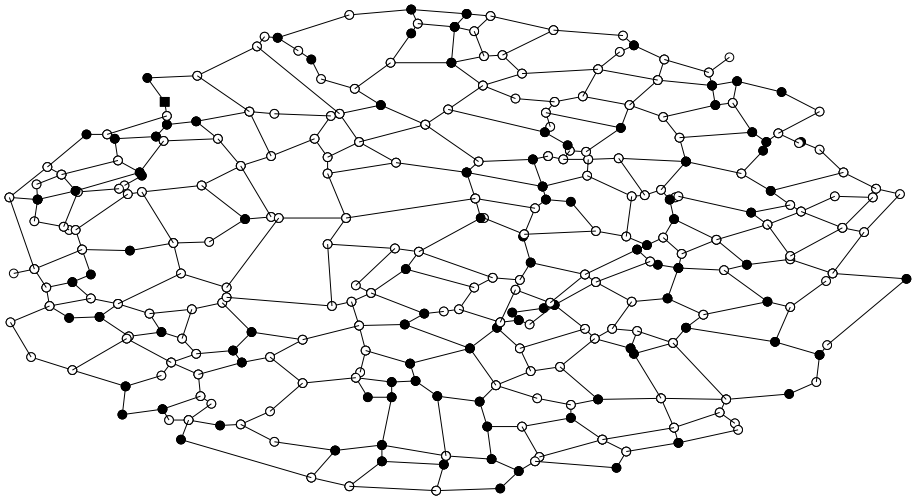

Figure 4: Graph with $n = 300$ and $|C| = 100$.

6.2 Experiments with multi-graph construction

In our first set of experiments, we examined the complexity of multi-graph computation in practice. Along the way, we also explored the effect of *correlations* between the costs and travel times. Intuitively, one would expect the number of edges in the multi-graph to increase as the correlation between costs and times decreases. Accordingly, for each of the ten graphs we created three different sets of random travel times, with differing levels of correlation with the costs: strong correlation, weak correlation and no correlation. This was done as follows, for each instance:

- Let $\bar{c}$ be the maximum edge cost.
- For each edge $e \in \tilde{E}$, let r_e be a random number uniformly distributed between 0 and 1. Set:
  - $t_e = \lceil 0.9c_e + 0.1r\bar{c} \rceil$ for strong correlation,
  - $t_e = \lceil 0.5c_e + 0.5r\bar{c} \rceil$ for weak correlation,
  - $t_e = \lceil r\bar{c} \rceil$ for no correlation.

The formulae used here were intended to make the travel times of the same order of magnitude as the costs.

For each of the resulting 30 cases, we computed the corresponding multi-graph using an algorithm of our own. This consisted of calling a single-source bi-criterion shortest-path subroutine $|C|$ times, with each customer in turn taking the role of the source node. The subroutine was a straightforward

dynamic programming algorithm. Although our algorithm is fairly rudimentary, its running time is $\mathcal{O}(|C|\,|\tilde{E}|\,T)$, which matches that of the best algorithms (see Section 3).

Table 1 shows the results obtained. The first three columns show the number of nodes, edges and customers in each of the ten sparse graphs. The next column shows the degree of correlation (strong, weak, none). The following two columns show the number of edges in the multi-graph, and the time taken to compute the multi-graph using our algorithm, in seconds.

We see that the number of edges in the multi-graph is rather high in all cases. As expected, it increases as the correlation decreases, and increases as $|\tilde{V}|$, $|\tilde{E}|$ and $|C|$ increase. From the way the multi-graph is constructed, one would expect the number of edges in the multi-graph to grow quadratically in $|C|$ and exponentially in $|\tilde{V}|$ and $|\tilde{E}|$. This is consistent with the results in the table.

The running times, on the other hand, are quite reasonable for these instances. This suggests that the time taken to compute the multi-graph is not an issue after all, although it could be, if the travel times were measured to a higher precision.

6.3 Comparison of pricing routines

Next, we wished to compare the time taken to perform pricing (with non-elementary routes permitted), when using either our algorithm (described in Section 5), or the corresponding algorithm based on the multi-graph. Since we expected the pricing time to depend on the width of the time windows, for each of the thirty combinations mentioned in the previous subsection, we created two Steiner m-TSPTW instances, with two different widths for the time windows. Each customer node $i \in C$ was assigned an integer service time $s_i \in [1, 2]$ at random. A set of routes was then constructed in a greedy way, so that each customer was serviced by exactly one vehicle. Finally, time windows $[e_i, \ell_i]$ were assigned to each customer such that the given routes were feasible and such that either $e_i + 3 \leq \ell_i \leq e_i + 4$ (for narrow time windows) or $e_i + 10 \leq \ell_i \leq e_i + 15$ (for wide time windows). The time horizon T was set to 3000 in all cases.

Table 2 shows the results obtained for the resulting 60 instances. The first four columns have the same meaning as before. The next four columns show the time taken, in seconds, to solve the LP relaxation of the set covering problem under various settings. The subscripts 'n' and 'w' indicate narrow and wide time windows, respectively, and the subscripts 'm' and 'o' indicate the multi-graph approach and our approach, respectively. All times include the time taken to re-optimise the LPs after adding columns, but, for these instances, the bulk of the time was spent pricing.

A first observation is that the pricing times are rather high in all cases. This is because we only added one column after each pricing call, namely,

$\lvert\tilde{V}\rvert$	$\lvert\tilde{E}\rvert$	$\lvert C\rvert$	Corr	$\lvert E\rvert$	Time (s)
50	69	33	S	674	0.5
			W	903	0.5
			N	1324	0.5
100	139	66	S	2850	2.2
			W	5009	2.1
			N	6892	2.2
150	215	100	S	5852	5.1
			W	12765	5.1
			N	18337	5.0
200	291	133	S	11641	9.0
			W	31770	9.0
			N	35617	9.1
250	367	166	S	18454	13.3
			W	37585	13.9
			N	56042	13.4
100	139	33	S	776	1.0
			W	1241	1.0
			N	1646	1.1
200	291	66	S	2857	4.3
			W	6096	4.2
			N	9708	4.3
300	442	100	S	7224	10.0
			W	16030	9.9
			N	23480	9.9
400	597	133	S	12946	18.1
			W	32087	18.9
			N	47359	18.7
500	744	166	S	23650	29.5
			W	60824	29.5
			N	79321	30.3

Table 1: Results on the construction of the multi-graph.

$\lvert\tilde{V}\rvert$	$\lvert\tilde{E}\rvert$	$\lvert C\rvert$	Corr	T_{nm}	T_{no}	T_{wm}	T_{wo}
50	69	33	S	8	2	10	3
			W	13	3	10	3
			N	32	6	32	7
100	139	66	S	103	23	216	29
			W	301	49	294	35
			N	504	59	518	75
150	215	100	S	974	122	760	126
			W	1405	139	2344	198
			N	7369	219	3713	208
200	291	133	S	17377	692	4854	569
			W	7777	479	11666	446
			N	14151	588	12187	620
250	367	166	S	5613	457	4753	300
			W	8415	720	10095	411
			N	4596	381	14264	538
100	139	33	S	17	10	17	9
			W	19	8	24	10
			N	28	13	37	14
200	291	66	S	348	97	398	100
			W	1043	223	697	161
			N	999	129	796	120
300	442	100	S	1008	210	695	365
			W	1875	259	1714	248
			N	3692	340	2362	272
400	597	133	S	1457	300	2462	423
			W	2055	194	4171	408
			N	11011	380	7428	436
500	744	166	S	4554	795	7803	942
			W	6643	620	12197	984
			N	23220	1775	47820	1551

Table 2: Time taken by pricing routines on Steiner m-TSPTW instances.

the one with the most negative reduced cost. One could probably reduce the number of pricing iterations, and therefore the pricing time, if one added many columns after each call. A second observation is that the pricing times tend to be higher when time windows are wide. This is probably because the cardinality of the column set Ω is larger when the windows are wider.

More importantly, for our purposes, it is clear that our approach is significantly faster than the multi-graph approach. Moreover, the difference seems to grow as the size of the original graph grows. In our view, this demonstrates clearly that working on the original graph is preferable in terms of running time, as well as in terms of ease of implementation.

7 Conclusion

Despite the vast research effort that has been put into VRPs with time windows, most of the existing exact and heuristic solution algorithms cannot be relied upon to give meaningful solutions when the original instance is defined on a road network. The approach of Garaix *et al.* [23], based on the computation of an auxiliary multi-graph, provides one way around this difficulty. In this paper, we have shown that it is possible instead to work with the original graph throughout, and that this can lead to significant savings in computing time.

We leave to future research the development of a full branch-and-price or branch-cut-and-price algorithm for the Steiner versions of the m-TSPTW or VRPTW. We remark that in order to develop such algorithms one would need to develop specialised branching rules that work on the original graph rather than the multi-graph. For a discussion of this issue in the context of the Capacitated Arc Routing Problem, see Bode & Irnich [7].

References

[1] M.P. de Aragão, H. Longo & E. Uchoa (2006) Solving capacitated arc routing problem using a transformation to the CVRP. *Comput. & Oper. Res.*, 33, 1823-1837.

[2] N. Ascheuer, M. Fischetti & M. Grötschel (2001) Solving the asymmetric traveling salesman problem with time windows by branch-and-cut. *Math. Program.*, 90, 475–506.

[3] M.O. Ball, T.L. Magnanti, C.L. Monma & G.L. Nemhauser (eds.) (1995) *Network Routing.* Handbooks in Operations Research and Management Science, vol. 8. Elsevier.

[4] E. Baker (1983) An exact algorithm for the time-constrained traveling salesman problem. *Oper. Res.*, 31, 938–945.

[5] R. Baldacci, A. Mingozzi & R. Roberti (2011) New route relaxation and pricing strategies for the vehicle routing problem. *Oper. Res.*, 59, 1269–1283.

[6] R. Baldacci, A. Mingozzi & R. Roberti (2012) New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS J. Comput.*, 24, 356–371.

[7] C. Bode & S. Irnich (2012) Cut-first branch-and-price-second for the capacitated arc-routing problem. *Oper. Res.*, 60, 1167–1182.

[8] A. Chabrier (2006) Vehicle routing problem with elementary shortest path based column generation. *Comput. & Oper. Res.*, 33, 2972-2990.

[9] A. Corberán, A.N. Letchford & J.M. Sanchis (2001) A cutting plane algorithm for the general routing problem. *Math. Program.*, 90, 291–316.

[10] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon & F. Soumis (2002) VRP with time windows. In P. Toth & D. Vigo (eds.) *Op. Cit.*

[11] G. Cornuéjols, J. Fonlupt & D. Naddef (1985) The traveling salesman on a graph and some related integer polyhedra. *Math. Program.*, 33, 1–27.

[12] S. Dash, O. Günlk, A. Lodi & A. Tramontani (2012) A time bucket formulation for the traveling salesman problem with time windows. *INFORMS J. Comput.*, 24, 132-147.

[13] G. Desaulniers, F. Lessard & A. Hadjar (2008) Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle with time windows. *Transp. Sci.*, 42, 387–404.

[14] J. Desrosiers, F. Soumis & M. Desrochers (1984) Routing with time windows by column generation. *Networks*, 14, 545–565.

[15] M. Desrochers (1988) Vehicle routing with time windows: optimization and approximation. In B.L. Golden & A.A. Assad (1988) *Vehicle Routing: Methods and Studies*. North Holland: Elsevier.

[16] M. Desrochers, J. Desrosiers & M.M. Solomon (1992) A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.*, 40, 342–354.

[17] J. Desrosiers, Y. Dumas, M.M. Solomon, F. Soumis (1995) Time-constrained routing and scheduling. In M.O. Ball *et al.* (eds.) *Op. Cit.*

[18] M. Dror (1994) Note on the complexity of the shortest path models for column generation in VRPTW. *Oper. Res.*, 42, 977–979.

[19] M. Dror (ed.) (2000) *Arc Routing: Theory, Solutions and Applications*. Kluwer Academic Publishers.

[20] Y. Dumas, J. Desrosiers, É. Gelinas & M.M. Solomon (1995) An optimal algorithm for the traveling salesman problem with time windows. *Oper. Res.*, 43, 367–371.

[21] D. Feillet, P. Dejax, M. Gendreau, C. Gueguen (2004) An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Networks*, 44, 216–229.

[22] B. Fleischmann (1985) A cutting plane procedure for the traveling salesman problem on a road network. *Eur. J. Opl Res.*, 21, 307–317.

[23] T. Garaix, C. Artigues, D. Feillet & D. Josselin (2010) Vehicle routing problems with alternative paths: An application to on-demand transportation. *Eur. J. Oper. Res.*, 204, 62-75.

[24] G. Ghiani & G. Laporte (2000) A branch-and-cut algorithm for the undirected rural postman problem. *Math. Program.*, 87, 467–481.

[25] B.L. Golden & A. Assad (eds.) (1988) *Vehicle Routing: Methods and Studies*. Amsterdam: North-Holland.

[26] B.L. Golden, S. Raghavan & E.A. Wasil (eds.) (2008) *The Vehicle Routing Problem: Latest Advances and New Challenges*. Series in Operations Research/Computer Science Interfaces, vol. 43. Berlin: Springer.

[27] P. Hansen (1980) Bicriterion path problems. In: G. Fandel & T. Gal (eds.) *Multiple Criteria Decision Making: Theory and Applications*, pp. 109-127. Lectures Notes in Economics and Mathematical Systems, vol. 177. Heidelberg: Springer.

[28] M. Jepsen, B. Petersen, S. Spoorendonk & D. Pisinger (2008) Subset-row inequalities applied to the vehicle-routing problem with time windows. *Oper. Res.*, 56, 497–511.

[29] B. Kallehauge, J. Larsen & O.B.G. Madsen (2006) Lagrangian duality applied to the vehicle routing problem with time windows. *Comp. & Oper. Res.*, 33, 1464–1487.

[30] N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon & F. Soumis (1999) 2-Path cuts for the vehicle routing problem with time windows. *Transp. Sci.*, 33, 101-116.

[31] A.N. Letchford, S.D. Nasiri & D.O. Theis (2013) Compact formulations of the Steiner traveling salesman problem and related problems. *Eur. J. Oper. Res.*, 228, 83–92.

[32] A.N. Letchford & A. Oukil (2009) Exploiting sparsity in pricing routines for the capacitated arc routing problem. *Comp. & Oper. Res.*, 36, 2320–2327.

[33] J. Lysgaard (2006) Reachability cuts for the vehicle routing problem with time windows. *Eur. J. Oper. Res.*,175, 210–223.

[34] C.S. Orloff (1974) A fundamental problem in vehicle routing. *Networks*, 4, 35–64.

[35] G. Pesant, M. Gendreau, J.-Y. Potvin, J.-M. Rousseau (1999) An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transp. Sci.*, 32, 12–29.

[36] A. Raith & M. Ehrgott (2009) A comparison of solution strategies for biobjective shortest path problems. *Comput. & Oper. Res.*, 36, 1299–1331.

[37] M.W.P. Savelsbergh (1985) Local search in routing problems with time windows. *Ann. Oper. Res.*, 4, 285–305.

[38] M.M. Solomon (1987) Algorithms for vehicle routing and scheduling problems with time window constraints. *Oper. Res.*, 35, 254–265.

[39] P. Toth & D. Vigo (eds.) (2002) *The Vehicle Routing Problem*. Philadelphia: SIAM Publications.