Computers & Operations Research 51 (2014) 331–337

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Pricing routines for vehicle routing with time windows on road networks

Adam N. Letchford [a,*], Saeideh D. Nasiri [a], Amar Oukil [b]

[a] Department of Management Science, Lancaster University Management School, Lancaster LA1 4YX, United Kingdom
[b] Department of Operations Management and Business Statistics, College of Economics and Political Science, Sultan Qaboos University, Muscat, Oman

ARTICLE INFO

Available online 2 July 2014

Keywords:
Vehicle routing
Combinatorial optimization
Bi-criteria shortest paths

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.

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., [3,24,25,39]). 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.

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 sub-problem 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

* Corresponding author.
E-mail addresses: A.N.Letchford@lancaster.ac.uk (A.N. Letchford), S.D.Nasiri@lancaster.ac.uk (S.D. Nasiri), aoukil@squ.edu.com (A. Oukil).

http://dx.doi.org/10.1016/j.cor.2014.06.022
0305-0548/

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. Section 2.1 deals with the TSPTW, *m*-TSPTW and VRPTW, Section 2.2 covers VRPs on road networks, and Section 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

$$\min \quad \sum_{r\in\Omega} c_r\lambda_r \tag{1}$$

$$\text{s.t.} \quad \sum_{r\in\Omega} a_{ir}\lambda_r \geq 1 \quad (\forall i\in V\backslash\{0\}) \tag{2}$$

$$\lambda_r\in\{0,1\} \quad (\forall r\in\Omega), \tag{3}$$

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., [5,13,27]).

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, in the case of the CARP, the vehicle load does not change when an edge is deadheaded.

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 $C\subset\tilde{V}\backslash\{0\}$,
- 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 $\tilde{V}\backslash(C\cup\{0\})$ 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,