Computers & Operations Research 39 (2012) 902–909

Contents lists available at ScienceDirect

Computers & Operations Research

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

Restricted dynamic programming: A flexible framework for solving realistic VRPs

J. Gromicho [a,b], J.J. van Hoorn [a,b], A.L. Kok [b], J.M.J. Schutten [c,*]

[a] Vrije Universiteit, Amsterdam, The Netherlands
[b] ORTEC, Gouda, The Netherlands
[c] University of Twente, Enschede, The Netherlands

ARTICLE INFO

Available online 18 July 2011

Keywords:
Restricted DP
Giant-tour representation
VRP
Real-life restrictions

ABSTRACT

Most successful solution methods for solving large vehicle routing and scheduling problems are based on local search. These approaches are designed and optimized for specific types of vehicle routing problems (VRPs). VRPs appearing in practice typically accommodate restrictions that are not accommodated in classical VRP models, such as time-dependent travel times and driving hours regulations. We present a new construction framework for solving VRPs that can handle a wide range of different types of VRPs. In addition, this framework accommodates various restrictions that are not considered in classical vehicle routing models, but that regularly appear in practice. Within this framework, restricted dynamic programming is applied to the VRP through the giant-tour representation. This algorithm is a construction heuristic which for many types of restrictions and objective functions leads to an optimal algorithm when applied in an unrestricted way. We demonstrate the flexibility of the framework for various restrictions appearing in practice. The computational experiments demonstrate that the framework competes with state of the art local search methods when more realistic constraints are considered than in classical VRPs. Therefore, this new framework for solving VRPs is a promising approach for practical applications.

1. Introduction

Metaheuristics have proved to be very successful in solving large vehicle routing and scheduling problems. The survey by Gendreau et al. [1] lists six types for the capacitated VRP alone, and from the surveys of Parragh et al. [2,3] we can only estimate the number to be much larger when allowing different types of VRP. While metaheuristics rely on the careful definition (and redefinition) of neighborhoods for each type of VRP, Pisinger and Ropke [4] propose a framework for construction, destruction and amendment of solutions, in which they assume construction to be simple and focus on identifying parts of the solution suspected of compromising quality and amending them. Alongside with the extensive work on metaheuristics for the vehicle routing problem and the work of Pisinger and Ropke, we propose a general construction only framework for very general vehicle routing problems based on dynamic programming.

We believe that there is still room for a general construction method that is able to deal with realistic and general types of vehicle routing problems. Solutions found may certainly be improved by either local search or other amendment algorithms, but our approach already allows for a trade-off between performance and quality. Moreover, the computational experiments indicate that our approach is generic enough to allow finding reasonable solutions for many types of vehicle routing problems without substantial redesign and coding effort.

The approach we propose is a restricted dynamic programming heuristic for very general vehicle routing problems, along the same lines as followed by Malandraki and Dial [5] for the time-dependent Traveling Salesman Problem. Furthermore, our parameter H has the same interpretation as theirs homonym parameter: $H=1$ renders our approach to a nearest neighbor approach, while higher values of H generalize this simple approach by allowing at most H solutions to be expanded further in each stage of the state space. By also including a new restriction, now on the number E of expansions per state, we reason as Toth and Vigo [6] in their Granular Tabu Search approach: long arcs are not likely to be part of optimal solutions. Therefore, ignoring those long arcs may substantially reduce the required computation times, without significant quality loss. Our approach can also be seen as a beam search approach, following the terminology introduced by Raj Reddy on his Artificial Intelligence courses at the Carnegie Mellon University in 1976. The first

* Corresponding author. Tel.: +31 53 489 4676; fax: +31 53 489 2159.
E-mail addresses: joaquim.gromicho@ortec.com (J. Gromicho), j.j.van.hoorn@vu.nl (J.J. van Hoorn), leendert.kok@ortec.com (A.L. Kok), m.schutten@utwente.nl (J.M.J. Schutten).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.07.002

published use of the term seems to be Lowerre [7]. Novel from our perspective is the concept being used on the state space of a dynamic equation and not on the solution space, as traditional enumeration algorithms do.

In practice, extensions of various types of VRPs need to be solved, such as the capacitated VRP (CVRP), the VRP with time windows (VRPTW), and the pickup and delivery problem (PDP). Toth and Vigo [8] give an extensive overview of different types of VRPs and proposed solution methods. In addition, companies such as logistic service providers and distribution firms have their own set of restrictions of which a certain part may be general to all companies, but most companies also have some unique restrictions. As a consequence, each company requires a unique solution method to solve their routing problems.

The framework we propose covers a wide range of different types of VRPs. Moreover, it accommodates various restrictions that appear in practice, but that have been generally ignored in VRP literature, such as time-dependent travel times and driving hours regulations. We apply restricted dynamic programming to the VRP through the giant-tour representation, which was introduced by Funke et al. [9]. The giant-tour representation allows us to handle single tour and multiple tour problems in a similar way.

The contributions of this paper are the following. First, we propose an exact algorithm for the VRP. In the framework of this algorithm, various real-life restrictions and VRP variants can be accommodated. Second, we demonstrate this flexibility by showing how the algorithm can be applied to known VRP variants, and by showing how various additional real-life restrictions can be accommodated. Third, in addition to the way Malandraki and Dial [5] propose to restrict the state space to reduce computation times, we propose an additional way to restrict the state space which reduces computation times even further while maintaining or even improving solution quality. As a result of restricting the state space, the algorithm runs in practical (polynomial) computation times and produces high quality solutions for realistic types of VRPs. Fourth, we show the quality of our framework by solving the classical Solomon [10] benchmark instances for the VRPTW. Moreover, we apply it to benchmarks of more restrictive vehicle routing problems: the PDP with time windows (PDPTW) and the VRPTW with the European Community (EC) social legislation on driving and working hours (VRPTW-EC). The results illustrate the power of this construction framework when applied to more realistic vehicle routing problems: when more restrictions are added to the problem, the performance of the framework improves, making it even competitive with state of the art improvement methods for the VRPTW-EC.

Our paper is organized as follows. Section 2 first describes dynamic programming for the TSP, then describes the giant-tour representation of VRP solutions, and finally describes our framework for solving VRPs. Section 3 describes a way to reduce the state space of this dynamic programming formulation to obtain solutions within practical computation times. Section 4 demonstrates the flexibility of our framework by showing how known types of VRPs can be solved within this framework, and how various additional real-life restrictions can be accommodated. Section 5 presents the results of computational experiments. Section 6 summarizes the main findings in this paper.

2. DP applied to the VRP

Our framework for solving VRPs is based on the restricted dynamic programming (DP) heuristic for the TSP proposed by Malandraki and Dial [5]. We apply the (DP) heuristic to the VRP through the giant-tour representation (GTR) of vehicle routing solutions introduced by Funke et al. [9]. The term giant tour was also introduced by Beasley [11]. However, this giant tour is actually a TSP solution with only one single copy of the depot. Therefore, it is not a representation of a feasible VRP solution, but it has to be turned into a feasible VRP solution by somehow cutting the giant tour into feasible VRP routes. We first describe the DP for the TSP and the GTR of vehicle routing solutions.

2.1. Dynamic programming for the TSP

The restricted dynamic programming heuristic for the TSP is based on the exact dynamic programming algorithm for the TSP of Held and Karp [12] and Bellman [13]. The exact dynamic programming algorithm for the TSP can be described as follows.

The TSP considers the problem of visiting a set $V=\{0,1,\ldots,n-1\}$ of n cities exactly once, starting and ending at city 0, and minimizing the total travel distance. The travel distance between each pair of cities $i,j \in V$ is given by c_{ij}.

A state $(S,j), j \in S, S \subseteq V \backslash 0$ in the DP algorithm represents a path starting at city 0, visiting all cities in S exactly once, and ending in city j. The cost $C(S,j)$ of a state is given by the length of the smallest of such paths. In the first stage, the costs of the states are determined by $C(\{j\},j)=c_{0j}, \forall j \in V\backslash 0$. Next, in each successive stage the costs of the states are calculated with the recurrence relation $C(S,j)=\min_{i \in S \backslash j}\{C(S\backslash j,i)+c_{ij}\}$. Finally, the length of the optimal TSP tour is given by $\min_{j \in V\backslash 0}\{C(V\backslash 0,j)+c_{j0}\}$.

Since there are $\sum_{|S|=1}^{n-1}\binom{n-1}{|S|} \approx 2^n$ subsets S and each subset S contains $|S| \leq n-1$ possible end nodes, the total number of states is $\mathcal{O}(n2^n)$. Next each state is calculated by comparing at most $n-2$ additions, resulting in an algorithm with a running time complexity of $\mathcal{O}(n^2 2^n)$. The optimal TSP tour can be backtracked by saving for each state (S,j) the city $i \in S\backslash j$ that minimizes $C(S\backslash j,i)+c_{ij}$.

Since this approach constructs only one route, it cannot be applied directly to the VRP. We propose to apply it to the VRP through the GTR of vehicle routing solutions.

2.2. Giant-tour representation

Funke et al. [9] introduce the GTR of vehicle routing solutions, because it allows to handle single and multiple route problems in a similar way. Besides, it is a 'natural' representation of vehicle routing solutions. We use the GTR for the development of our general framework for solving VRPs. The GTR can be described as follows.

The basis of any routing problem is a directed graph $G=(V,A)$, in which the node set V consists of request nodes $R \subset V$, origin nodes $O \subset V$, and destination nodes $D \subset V$, and the arc set A represents feasible travels between these nodes. For a VRP, the request nodes R correspond to all customer requests. Furthermore, for each vehicle there is one origin and one destination node, which all may represent the same location. Therefore, if m is the number of vehicles available, we get $|O|=|D|=m$. If we order the vehicle routes $r^v, v=1,\ldots,m$ in a routing solution, then the GTR of this solution is a cycle in the graph G in which each route end node d^v is connected to the route start node of the next vehicle route o^{v+1}. Finally, the cycle is closed by connecting d^m with o^1.

To use this representation, the number of available vehicles must be known beforehand. If the number of available vehicles is not given, we can use an upperbound on the required number of vehicles, e.g., by setting it equal to the number of customers. If it turns out that some of the vehicles are not required (typically when one of the objectives is to minimize the number of vehicles), this is represented by directly connecting their origin and destination nodes.

In Fig. 1, we present an example of a vehicle routing solution with three vehicles, two depots (A and B) and nine customers. Vehicle 1 starts at depot A and ends at depot B, vehicle 2 starts and ends at depot B, and vehicle 3 starts and ends at depot A. Fig. 2 presents the same solution with its corresponding GTR.