European Journal of Operational Research 212 (2011) 482–496

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing and Logistics

The tree representation for the pickup and delivery traveling salesman problem with LIFO loading

Yongquan Li [a], Andrew Lim [b], Wee-Chong Oon [b], Hu Qin [a,b,*], Dejian Tu [c]

[a] School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China
[b] Department of Management Sciences, City University of Hong Kong, Tat Chee Ave, Kowloon Tong, Hong Kong
[c] Department of Computer Science, School of Information Science and Technology, Sun Yat-Sen University, Guangzhou, Guangdong, China

ARTICLE INFO

Article history:
Received 6 June 2010
Accepted 3 February 2011
Available online 10 March 2011

Keywords:
Traveling salesman
Pickup and delivery
Last-in-first-out loading
Tree data structure
Variable neighborhood search

ABSTRACT

The feasible solutions of the traveling salesman problem with pickup and delivery (TSPPD) are commonly represented by vertex lists. However, when the TSPPD is required to follow a policy that loading and unloading operations must be performed in a last-in-first-out (LIFO) manner, we show that its feasible solutions can be represented by trees. Consequently, we develop a novel variable neighborhood search (VNS) heuristic for the TSPPD with last-in-first-out loading (TSPPDL) involving several search operators based on the tree data structure. Extensive experiments suggest that our VNS heuristic is superior to the current best heuristics for the TSPPDL in terms of solution quality, while requiring no more computing time as the size of the problem increases.

1. Introduction

The traveling salesman problem with pickup and delivery (TSPPD) has been extensively studied by a large number of researchers (e.g., Kalantari et al., 1985; Healy and Moll, 1995; Renaud et al., 2000; Renaud et al., 2002; Dumitrescu et al., 2009). In the TSPPD, there is a set of n requests, denoted by $R = \{1,\ldots,n\}$, each of which is composed of a pickup vertex and a delivery vertex. Let $P = \{1^+,\ldots,n^+\}$ be the set of pickup vertices and $D = \{1^-,\ldots,n^-\}$ be the set of delivery vertices. Vertices 0^+ and 0^- represent the exit from and the entrance to the depot, respectively. The TSPPD is defined on a complete and undirected graph $G = (V,E,d)$, where $V = P \cup D \cup \{0^+,0^-\}$ is the vertex set; $E = \{(x,y):x,y \in V, x \neq y\}$ is the edge set; and $d(x,y)$ denotes the non-negative distance between vertices x and y. The objective of the TSPPD is to find a shortest Hamiltonian tour on G, starting from vertex 0^+ and ending at vertex 0^-, for a vehicle with unlimited capacity, subject to the precedence constraints that each pickup vertex is visited before its associated delivery vertex. In most of the existing literature on the TSPPD, feasible solutions are represented by lists (or sequences) of vertices complying with the precedence constraints.

This study addresses a variant of the TSPPD in which loading and unloading operations must be performed in a last-in-first-out (LIFO) manner; this problem is referred to as the TSPPD with LIFO loading (TSPPDL). It models the fact that the storage units of most transport vehicles have only a single door located at the rear, and therefore the goods that are first picked up must be stored towards the front of the vehicle (away from the door). As a consequence, the last goods picked up should be the first ones unloaded. The LIFO constraints are especially applicable when transporting bulky items, or when the amount of handling must be minimized during the transport of hazardous materials.

The TSPPDL has been considered a more complex problem than the TSPPD because in order to judge whether a solution that is represented as a vertex list is feasible, we have to check not only the precedence constraints, but also the LIFO constraints. In this paper, we show that there is a one-to-one correspondence between feasible solutions of the TSPPDL and tree representations of such solutions. This tree data structure embodies the nature of the TSPPDL solutions and greatly simplifies this seemingly complicated problem. As a result, we were able to build upon the Variable Neighbourhood Search (VNS) heuristic proposed by Carrabs et al. (2007b), which is the best existing approach for the TSPPDL at the time of this writing, by both reproducing five of the original operators using the tree representation as well as introducing three new operators. Extensive experiments show that our new heuristic outperforms the original one in terms of solution quality, and it also requires

* Corresponding author at: School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China. Tel.: +852 67964841; fax: +852 34420189.
E-mail addresses: liyongquan@mail.hust.edu.cn (Y. Li), lim.andrew@cityu.edu.hk (A. Lim), weecoong@cityu.edu.hk (W.-C. Oon), tigerqin@cityu.edu.hk (H. Qin), tudejian@gmail.com (D. Tu).

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.02.008

comparatively less computation time as the number of requests in the problem increases.

2. Existing research

The TSPPDL is a difficult combinatorial optimization problem that combines a routing problem with loading constraints (Iori and Martello, 2010). It was first mentioned by Ladany and Mehrez (1984). However, they neither formulate it into a mathematical model nor propose solution procedures except for enumeration, which can be viewed as the simplest exact algorithm.

Carrabs et al. (2007a) introduced a branch-and-bound algorithm that applies an additive lower bound technique proposed by Fischetti and Toth (1989) to generate lower bounds. To further improve the quality of the lower bounds, at each node of the branch and bound search tree a set of elimination rules is utilized to remove from the graph edges that are impossible to be part of any feasible solution due to the precedence relationships. Experimental results showed that this branch-and-bound algorithm is capable of solving all instances with 15 requests and several instances with 21 requests.

Currently, the best exact algorithm for the TSPPDL is the branch-and-cut algorithm described by Cordeau et al. (2010). This algorithm was constructed based on the fundamental component from the commercial integer programming solver ILOG CPLEX and employs several families of valid inequalities. Computational results showed that it is able to handle most instances with up to 17 requests in less than 10 minutes, and several instances with 25 requests within 1 hour.

For the generation of near-optimal solutions for the large instances widely encountered in practice, the best approaches so far have made use of efficient heuristics. An early approach by Pacheco (1997) employed a simulated annealing algorithm for this problem. Cassani and Righini (2004) developed a greedy heuristic and a variable neighborhood descent (VND) heuristic which combines four types of exchange operators, known as *couple-exchange*, *block-exchange*, *relocate-block* and *relocate-couple*. Carrabs et al. (2007b) built on this work by devising a variable neighborhood search (VNS) heuristic which includes the four exchange operators along with four new operators: *multi-relocate*, *2-opt-L*, *double-bridge* and *shake*. They compared the VND and VNS heuristics by solving instances with up to 375 requests; the computational results showed that at the expense of more computing time, the VNS heuristic produces significantly better solutions than the VND heuristic. All of these implementations represented feasible solutions of the TSPPDL by vertex lists.

3. The tree representation of feasible tours

In this section, we describe the tree representation of feasible tours for the TSPPDL, which is the primary contribution of this study. In particular, a feasible tour for the TSPPDL can be represented as an ordered tree (i.e., there is an order for the children of each tree node) with $|R| + 1$ nodes, where the root node is labeled 0 and the remaining nodes are labeled $\{1,\ldots,|R|\}$ in some permutation; we call a tree of this type a *TSPPDL tree*. We show that by representing solutions using this tree, the feasibility of the solution is automatically guaranteed. We also show that there is a one-to-one correspondence between the set of all feasible solutions to the TSPPDL and the set of all such trees.

Fig. 1(a) shows an example of a TSPDDL tree. The dashed arrows in Fig. 1(b) pictorially shows how this ordered tree can be converted into a tour that automatically respects the precedence and LIFO constraints of the TSPPDL, corresponding to the vertex list given in Fig. 2; this is similar to a preorder traversal of the tree, where the pickup vertex is instantiated when its node is first encountered, and the delivery vertex is instantiated when the node is last encountered. The conversion procedure is provided in Algorithm 1, which runs in $O(n)$ time. Note that any TSPPDL subtree has the property of a "no-crossing" (Volchenkov, 1982) or a "nested palindrome" (Cordeau et al., 2010).

Algorithm 1. Procedure for converting a TSPPDL tree into a feasible tour

1: INPUT: An ordered tree T;
2: Initialize the current node $x \leftarrow$ node 0;
3: Initialize the current tour $S \leftarrow \emptyset$;
4: Execute recursive procedure $DFS(T,S,x)$, defined as:
5: $DFS(T,S,x)$
6: {Append x^+ to the tail of S;
7: **while** node x has unvisited children
8: $y \leftarrow$ the leftmost unvisited child of node x;
9: Invoke $DFS(T,S,y)$;
10: **end while**
11: Append x^- to the tail of S; }
12: Return S;

We now define some terminology and notations that we will employ in this paper. In standard graph terminology, the terms *node* and *vertex* have the same meaning and are used interchangeably. However, in this study we distinguish between the two terms: we specify that *node* refers to a TSPPDL tree node corresponding to a request in R, and *vertex* refers to the pickup or delivery vertex in V. We also define a *tour* as the vertex list representing a solution of the TSPPDL. Furthermore, we use the notation $T_{S_x},\ x \in R$ to denote the subtree in T rooted at the node corresponding to request x. Finally, for ease of discussion we will refer to TSPPDL trees simply as *trees*.

Theorem 1. *Let T be a TSPPDL tree. The tour generated from T using Algorithm 1 is a feasible solution to the TSPPDL.*

Proof. This is proved by dividing the set of feasible TSPPDL tours into four cases and showing that any generated tour must fulfill one of these cases. See Appendix A. □

Theorem 2. *Algorithm 1 provides a one-to-one correspondence between TSPPDL trees and feasible solutions of the TSPPDL.*

Proof. We first show that the number of TSPPDL trees with n nodes and the number of feasible TSPPDL solutions with n requests are the same, and then show that Algorithm 1 implements an injective function from the set of TSPPDL trees to the set of feasible TSPPDL tours. See Appendix B. □

To convert a feasible tour into a tree, a reverse procedure is presented in Algorithm 2, which is the inverse of Algorithm 1 and also runs in $O(n)$ time. These algorithms are linear time procedures that convert a TSPPDL tree into a feasible tour and vice versa, so there is at most a linear time difference between two approaches that use either representation.

Algorithm 2. Procedure for converting a feasible tour into a TSPPDL tree

1: INPUT: A feasible tour S;
2: Let $node(v)$ be the node associated with vertex $v \in S$;