Computers & Operations Research 39 (2012) 1074–1086

Contents lists available at ScienceDirect

Computers & Operations Research

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

# Metaheuristics for the traveling salesman problem with pickups, deliveries and handling costs

Güneş Erdoğan [a,*], Maria Battarra [b], Gilbert Laporte [c], Daniele Vigo [d]

[a] Industrial Engineering Department, Özyeğin University, Kuşbakışı Cad. 2, 34662 Altunizade, İstanbul, Turkey
[b] School of Mathematics, University of Southampton, Southampton SO17 1BJ, UK
[c] Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montreal, Canada H3T 2A7
[d] DEIS, University of Bologna, via Venezia 52, 47521 Cesena, Italy

ARTICLE INFO

Available online 22 July 2011

*Keywords:*
Traveling salesman problem
Handling cost
Pickup and delivery
Metaheuristics

ABSTRACT

This paper studies the Traveling Salesman Problem with Pickups, Deliveries, and Handling Costs. The subproblem of minimizing the handling cost for a fixed route is analyzed in detail. It is solved by means of an exact dynamic programming algorithm with quadratic complexity and by an approximate linear time algorithm. Three metaheuristics integrating these solution methods are developed. These are based on tabu search, iterated local search and iterated tabu search. The three heuristics are tested and compared on instances adapted from the related literature. The results show that the combination of tabu search and exact dynamic programming performs the best, but using the approximate linear time algorithm considerably decreases the CPU time at the cost of slightly worse solutions.



## 1. Introduction

The *Traveling Salesman Problem with Pickups, Deliveries and Handling Costs* (TSPPD-H), recently introduced by Battarra et al. [1], is a generalization of the *Single Vehicle Pickup and Delivery Problem with combined demands* (SVPDP–P&D) (see [2]). As in the SVPDP–P&D, the aim is to design an optimal tour through the depot and a set of customers, each of which requires a pickup service, a delivery service, or both. The vehicle leaves the depot carrying all the deliveries, visits each customer once, and returns to the depot carrying all pickups, without ever exceeding its capacity.

In the TSPPD-H the vehicle is represented as a single stack in which the commodities can be loaded and unloaded only from the rear. This vehicle configuration is commonly encountered in real world applications. The vehicle is a stack whenever the goods have one dimension comparable to the truck width. In Battarra et al. [1], the collection and delivery of damaged and undamaged bicycles in public bicycle sharing systems is presented as an example of TSPPD-H; we also mention the problem of delivering calves to farms and collecting mature cows [3]. Note that under the assumption of a single stack, pickup commodities may obstruct the delivery operations.

It is common practice to either disregard the handling operations and leave the driver free to decide about them, or to impose specific tour shapes [4,5] to handle the obstruction issue. For example, by forcing some or all delivery operations to take place at the beginning of the tour in order to create some empty space on board. However, non-Hamiltonian tours can increase routing costs substantially. An example of LIFO application is the auto-carrier transportation problem presented in Dell'Amico et al. [6], in which it is shown that a significant improvement in routing cost could be obtained by dropping the LIFO constraint and allowing handling operations.

The TSPPD-H consists of determining an Hamiltonian tour in which the commodities on board are possibly rearranged at the customer locations. The objective is to minimize the sum of handling and routing costs, the handling cost being the time spent in rearranging the load on board in the vehicle, and the routing cost the time required to actually perform the Hamiltonian tour. The problem becomes particularly interesting whenever the handling cost is comparable to the routing cost and a tradeoff between handling and routing can be envisaged. In fact, if the routing cost is significantly higher than the handling cost, the problem can be solved efficiently as a SVPDP–P&D and then by optimizing the handling decisions based on the routing decisions. On the other hand, if the handling cost is significantly higher than the routing cost, the problem can be solved as a *Double-Path Single Vehicle Pick and Delivery Problem* [4], or possibly by using two vehicles, one for the pickup and one for the delivery service.

There is a growing interest in SVPDP–P&Ds involving handling decisions. Variants of the Traveling Salesman Problem with Pickups, Deliveries with LIFO loading [7,8], FIFO loading [9] and multiple stacks [10,11] have been recently studied, as well as Vehicle Routing Problems including loading issues [12]. We refer the reader to the surveys by Iori and Martello [13] and Laporte [14]. All applications pertaining to the SVPDP–P&Ds with rear-loading are valid for the

* Corresponding author.
*E-mail address:* gunes.erdogan@ozyegin.edu.tr (G. Erdoğan).

0305-0548/$ - see front matter © 2011 Published by Elsevier Ltd.
doi:10.1016/j.cor.2011.07.013

TSPPD-H if the vehicle load can be rearranged, and hence there exists a potential for cost reduction in these problems.

Battarra et al. [1] have shown that the optimal loading decisions for a given tour can be determined in polynomial time. However, these decisions cannot easily be implemented in practice since they may require an excessive amount of handling. Hence, three elementary loading policies were proposed in order to simplify the handling operations. Branch-and-cut algorithms were developed for each policy. The results highlighted the relative importance of handling costs with respect to routing costs. Policies 1 and 2 (in which the pickup commodities are located either in the rear or in the front of the vehicle at each customer, respectively) are very easy to implement, but compromise the overall solution quality. On the other hand, Policy 3 achieves high quality results. This policy consists of deciding either to place, at each customer location, the pickup commodities in the rear or in the front of the vehicle. Policy 3 is simpler to implement than the optimal policy for the driver because the pickup and delivery commodities on board exhibit a contiguous pattern. Moreover, it is more efficient than the first two policies, because it avoids long queues of pickup commodities at the rear, as well as too frequent handling of delivery commodities.

The TSPPD-H, being a generalization of the SVPDP–P&D, is an *NP-hard* problem which combines the complexity of routing and scheduling decisions. Although promising results were reported under Policy 3, exact algorithms were only able to solve small-size instances (with up to 25 customers). In this paper, we therefore develop and compare powerful metaheuristics for the TSPPD-H. Our goal is to determine high quality solutions for large size instances within limited computing time. To this end, we first study the handling cost subproblem in greater depth, we next analyze possible neighborhoods and their evaluation, and we finally describe the three metaheuristics.

The remainder of the paper is organized as follows. Section 2 introduces the notation and describes the algorithms developed for the handling cost subproblem. Local search operators are analyzed in Section 3. Section 4 describes the proposed metaheuristic algorithms. Computational results are presented in Section 5, followed by conclusions in Section 6.

## 2. The handling cost subproblem

Let $G=(V, A)$ be a complete directed graph where $V=\{0,\ldots,n\}$ is the vertex set and $A$ is the arc set. Vertex 0 is the depot, whereas $V_c=V\backslash\{0\}$ are the customers. Each arc $(i,j)\in A$ is associated with a travel time $c_{ij}$, and each customer $i\in V_c$ are associated with $\alpha_i$ delivery commodity units and $\beta_i$ pickup commodity units. There are $\alpha_0=\sum_{i\in V_c}\alpha_i$ delivery commodity units originating from the depot, and $\beta_0=\sum i\in V_c\beta_i$ pickup commodity units destined to the depot. We assume both commodities have the same unit dimensions (for example boxes, pallets, or containers of the same goods). We denote the delivery and pickup commodities as type $a$ and $b$, respectively. The vehicle capacity is $Q$ units.

As previously stated, the vehicle follows a Last-In-First-Out (LIFO) loading policy [15]. When visiting a customer, the type $b$ commodities obstructing the delivery commodities have to be unloaded. The time spent in unloading and reloading a type $b$ commodity on board is equal to $h_b$. In addition, type $a$ commodities may be unloaded and reloaded in order to place pickup commodities in a more suitable position. The time spent to unload and reload a type $a$ commodity is equal to $h_a$. We denote these unloading and reloading activities as *additional operations*. The handling operations strictly required to perform delivery and pickup are constant and unavoidable, and are therefore not taken into account in the handling cost evaluation. We refer to Battarra et al. [1] for an exact mixed integer linear programming formulation of the problem.

The optimal solution for the TSPPD-H is a Hamiltonian tour on $G$ which minimizes the sum of routing costs and additional operation costs. As a consequence, in a local search algorithm, moves in the neighborhood of a solution have to evaluated considering these two aspects. The extra travel time cost can be simply computed as in the routing literature: the cost of the newly introduced arcs is added to the previous routing cost, whereas the cost of the deleted arcs is subtracted. This extra travel time computation is a constant time complexity operation, so it does not interfere with the algorithm speed. On the other hand, computing the handling cost variation is more difficult. For example, every time a customer is relocated in the tour, the overall handling cost has to be computed from scratch. The handling cost computation can be a time consuming task: any improvement to speed up this task should then result in an immediate reduction of the overall computing time.

In the remainder of this section we propose two ways of solving the handling cost subproblem. The first one is an exact dynamic programming algorithm with quadratic time and space complexity, and the second one is a linear time heuristic based on the solution of a special case of the handling cost subproblem.

### 2.1. Dynamic programming algorithm

Given a tour, in order to compute an exact optimal solution to the handling cost subproblem under Policy 3, we need to decide between placing the pickup commodities at the rear (Policy 1) or at the front (Policy 2) of the vehicle at every customer location. The cost of each decision is based on the current configuration of the commodities on board. This naturally leads to the idea of using the number of pickup and delivery boxes as state variables. However, this approach inevitably leads to a pseudopolynomial complexity algorithm. We now provide a genuinely polynomial dynamic programming algorithm to solve the handling cost subproblem.

For the sake of brevity and clarity, assume that we have renumbered the customers according to their order in the given tour. Let $f(i)$ denote the optimal cost of handling customers $i+1$ to $n$, given that Policy 2 has been applied at customer $i$. Furthermore, let $p_{ij}$ denote the cost of applying Policy 1 at customers $i+1$ to $j-1$, and Policy 2 at customer $j$:

$$\text{(DP)}\quad f(i)=\min_{j\in\{i+1,n\}}\{p_{ij}+f(j)\},\quad \forall i\in\{0,\ldots,n-1\}, \tag{1}$$

$$f(n)=0. \tag{2}$$

In DP, there are $n$ states, each of which requires $O(n)$ computing time, which yields a complexity of $O(n^2)$. However, the application of DP requires the values of $p_{ij},\forall i,j\in\{1,\ldots,n\}$ to be computed beforehand. This is achieved through Algorithm 2.1, which also has an $O(n^2)$ complexity. In the pseudo-code, $\alpha'$ and $\beta'$ denote the number of delivery and pickup commodities on board, respectively, and $\theta$ is the cumulative cost of applying Policy 1.

**Algorithm 2.1.** P $(\boldsymbol{\alpha},\boldsymbol{\beta},h_a,h_b)$

**for** $i=0$ **to** $(n-1)$
  $\beta'=0;\quad \theta=0$; // Policy 2 has been applied
  // compute the remaining number of delivery items on board
  $\alpha'=\alpha_0$;
  **for** $j=1$ **to** $i$
    $\alpha'=\alpha'-\alpha_j$;
  **for** $j=i+1$ **to** $n$
    $\alpha'=\alpha'-\alpha_j$; // deliver, no matter Policy 1 or 2
    // cost of Policy 1 up to this point plus the cost of Policy 2
    **if** $(\beta'+\beta_j>0)\quad p_{ij}=\theta+h_a\alpha'+h_b\beta'$
    **else** $\quad p_{ij}=\theta$