Computers & Operations Research 36 (2009) 1639 – 1645

Contents lists available at ScienceDirect

Computers & Operations Research

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

A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem

Hipólito Hernández-Pérez, Inmaculada Rodŕguez-Martín, Juan José Salazar-González*

DEIOC, Facultad de Matemáticas, Universidad de La Laguna, 38271 La Laguna, Tenerife, Spain

ARTICLE INFO

Available online 29 March 2008

Keywords:
Pickup and delivery
Traveling salesman problem
Hybrid heuristic
GRASP
VND

ABSTRACT

We address in this paper the one-commodity pickup-and-delivery traveling salesman problem, which is characterized by a set of customers, each of them supplying (pickup customer) or demanding (delivery customer) a given amount of a single product. The objective is to design a minimum cost Hamiltonian route for a capacitated vehicle in order to transport the product from the pickup to the delivery customers. The vehicle starts the route from a depot, and its initial load also has to be determined. We propose a hybrid algorithm that combines the GRASP and VND metaheuristics. Our heuristic is compared with other approximate algorithms described in Hernández-Pérez and Salazar-González [Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. Transportation Science 2004;38:245–55]. Computational experiments on benchmark instances reveal that our hybrid method yields better results than the previously proposed approaches.

1. Introduction

The *one-commodity pickup-and-delivery traveling salesman problem* (1-PDTSP) is a routing problem that generalizes the classical *traveling salesman problem* (TSP). We are given a set of locations and the travel distances among them. One specific location is considered to be a vehicle *depot*, while all the others are identified with *customers*. There is a unique commodity or product that has to be transported from some customers to others. To this end, each customer is visited once by the vehicle. Customers are divided into two groups, depending on whether they supply a given amount of product (*pickup customers*) or they demand a given amount of it (*delivery customers*). The product collected at pickup customers can be supplied to delivery customers. Moreover, the vehicle has a known capacity, and it must start and end its route at the depot. The 1-PDTSP consists of finding a minimum length Hamiltonian route for the vehicle that satisfies all the customer requirements. It is not assumed that the vehicle leaves empty or full loaded from the depot. On the contrary, the initial load of the vehicle also has to be determined (see [1]). We will assume that the travel distances among locations are symmetric.

The 1-PDTSP has several practical applications in routing a single commodity. One of them is described by Anily and Bramel [2] in the context of inventory repositioning. Consider a set retailers owned by the same firm and located at different sites in a region. At a given moment, due to the random nature of demands, some retailers may have an excess of inventory while others are in need of additional stock. Then, the firm may decide to transfer inventory from the first group of retailers to the second one. Determining the cheapest Hamiltonian route to do so with a capacitated vehicle is exactly the 1-PDTSP. Anily and Bramel [2] propose heuristic algorithms for the special case of the 1-PDTSP where the delivery and pickup quantities are all equal to one unit. This problem is called *capacitated TSP with pickups and deliveries*. Chalasani and Motwani [3] consider the same problem with the name *Q-delivery TSP*. The problem with unitary pickups and deliveries on a path or tree network has been studied by Wang et al. [4].

Hernández-Pérez [5] is the first to introduce the 1-PDTSP. He makes a theoretical study of the problem and presents solution methods. Hernández-Pérez and Salazar-González [6] describe an exact branch-and-cut algorithm able to solve instances with up to 60 customers. The same authors propose in [1] two heuristic approaches to deal with larger instances. The first heuristic approach is a simple local search procedure developed to provide initial upper bounds for their branch-and-cut algorithm. The second approach is a more elaborated algorithm based on "incomplete optimization". That is, the branch-and-cut algorithm described in [6] is applied to a restricted search space obtained by considering only a subset of model variables, associated to promising edges of a graph. Moreover, the branch-and-cut execution is truncated by imposing a limit to the number of levels in the decision tree exploration (see [1] for details). A primal heuristic is also embedded in the branch-and-cut to build feasible integer solutions from the information given by the

* Corresponding author. Tel.: +34 922 318184; fax: +34 922 318170.
E-mail address: jjsalaza@ull.es (J.J. Salazar-González).

0305-0548/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2008.03.008

fractional solutions. This procedure is periodically applied during the search process.

There are many other pickup-and-delivery routing problems described in the literature. For recent surveys, we refer the reader to Savelsbergh and Sol [7], Parragh et al. [8,9], and Berbeglia et al. [10]. However, as observed in [9], little attention has been paid to the 1-PDTSP. As far as we know, the only heuristic approaches are those in [1], none of these is a metaheuristic.

The 1-PDTSP is $\mathcal{NP}$-hard since it coincides with the TSP when the vehicle capacity is large enough. Even more, the problem of checking the existence of a feasible solution is $\mathcal{NP}$-complete in the strong sense (see [5]). This is a fundamental difference with respect to the TSP, as even just finding a feasible tour may be a very complex task. In this article we present a hybrid heuristic method that combines a greedy randomized adaptive search procedure (GRASP) with variable neighborhood descent (VND). The proposed algorithm is compared with the heuristic methods described in [1]. The outcomes of the computational tests show that the new heuristic yields better results than the previous ones, managing to improve the best known solution for most large instances.

We introduce now the notation used throughout this article. The depot is denoted by 1 and each customer by i ($i = 2, \ldots, n$). The set $V := \{1, 2, \ldots, n\}$ is the vertex set and E is the edge set. For each pair of locations $\{i, j\}$, the travel distance (or cost) c_{ij} of traveling between i and j is given. A non-zero demand q_i associated with each customer i is also given, being $q_i < 0$ if i is a delivery customer and $q_i > 0$ if i is a pickup customer. The capacity of the vehicle is represented by Q and is assumed to be a positive number. Note that typically $Q \leqslant \max\{\sum_{i\in V:q_i>0} q_i, -\sum_{i\in V:q_i<0} q_i\}$ on a 1-PDTSP instance. The depot can be considered a customer by defining $q_1 := -\sum_{i=2}^{n} q_i$, i.e., a customer absorbing or providing the necessary amount of product to ensure product conservation.

The remainder of this article is organized as follows. Section 2 describes our algorithm and its constituent parts. The computational results in Section 3 show the effectiveness of our method, that improves the performance of the heuristics presented in [1]. Final remarks are made in Section 4.

2. The algorithm

Search based heuristics for combinatorial optimization problems usually require some kind of diversification to overcome local optimality. Multi-start methods seek diversification by re-starting a local search procedure from multiple randomly generated initial solutions. The GRASP metaheuristic, proposed by Feo and Resende [11], is a multi-start procedure. Therefore it consists basically of a loop embedding a construction phase and a local search phase. The best overall solution is kept as the final result. The construction phase builds up a solution iteratively, randomly selecting each time an element from a *restricted candidate list* (RCL). The elements in the list are sorted according to a greedy function previously defined. This function measures the benefit of selecting each element. The procedure is adaptive since the benefits associated to every element are updated at each iteration of the construction phase, reflecting the changes brought on by the selection of the previous elements. The probabilistic component of a GRASP is characterized by a random choice of the element from the list, that is not necessarily the top candidate of the RCL. This choice technique allows for different solutions to be generated at each GRASP iteration. The whole strategy has been successfully applied to solve several difficult optimization problems (see Festa and Resende [12] for a review, and Resende and Ribeiro [13]).

On the other hand, VNS (variable neighborhood search) is based on the systematic change of neighborhood within the search (see Mladenović and Hansen [14]). The key idea is to change the local search operator, or neighborhood, once a local optimum is attained. To rapidly expose the main steps of VNS, let us denote by N_k ($k = 1, \ldots, k_{\max}$) a set of pre-selected neighborhood structures, by x a given solution, and by $N_k(x)$ the set of neighbor solutions of x in the k-th neighborhood. The algorithm performs a series of iterations until a stopping condition is satisfied. At each iteration, and starting with $k = 1$, a neighbor solution $x' \in N_k(x)$ is randomly generated. Then, a local search is applied to x' producing a local optimum. If the local optimum improves the current solution, then x is updated and the process is repeated. Otherwise, the algorithm resumes from x using a higher order neighborhood, if there is any. The VND method is a variant of VNS (see [14]) where the change of neighborhood is performed in a deterministic way. More precisely, the local minimum found when performing local search within a neighborhood is the starting point of the local search within the next neighborhood. The basic scheme of VND is stated in Algorithm 1.

Algorithm 1. VND(x) procedure
for $k \leftarrow 1$ to *max* **do**
  $x' \leftarrow$ LocalSearch($x, N_k(x)$)
  if x' is better than x **then**
    $x \leftarrow x'$
  end if
end for
return x

The algorithm we propose for solving the 1-PDTSP is a hybrid algorithm that combines the GRASP and the VND paradigms. The first part consists of a GRASP where the local search has been replaced by a VND procedure. That is, at each iteration of the GRASP loop, the solution given by the greedy randomized algorithm is taken as the starting point of a first VND, referred to as VND_1. This procedure is composed of two edge-exchange neighborhood structures. The GRASP loop is iterated until a certain stopping condition is met. Then, it follows the second part of the heuristic, a post-optimization phase consisting of a second VND, called VND_2, that starts from the best solution found so far. The procedure VND_2 uses two neighborhood structures based on vertex-exchange movements. The whole scheme of the hybrid heuristic is outlined in Algorithm 2.

Algorithm 2. Hybrid heuristic for the 1-PDTSP
while stopping criterion is not satisfied **do**
 $x \leftarrow$ GreedyRandomizedInitSol() {`construction phase`}
 {`improvement phase`}
 $x \leftarrow$ VND_1(x) {`edge-exchange neighborhoods`}
 if x is feasible and improves the best solution x' **then**
  $x' \leftarrow x$
 end if
end while
{`post-optimization`}
$x' \leftarrow$ VND_2(x') {`vertex-exchange neighborhoods`}
return x'

Next we describe with more detail each of the hybrid heuristic components.

2.1. Construction phase

To generate an initial solution we proceed in a greedy and adaptive way, starting from a randomly selected customer and iteratively adding a new one each time until all customers are in the solution. Recall that a partial solution corresponds to a path for the vehicle from the first to the last customer in the solution, and that, as explained in [1], it is feasible only if the difference between the maximum and the minimum load of the vehicle along the path does not exceed the capacity. More precisely, let $\vec{P}$ be a path through the