CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

The Pickup and Delivery Traveling Salesman Problem with Handling Costs

Marjolein Veenstra
Kees Jan Roodbergen
Iris F.A. Vis
Leandro C. Coelho

September 2015

CIRRELT-2015-44

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2015-10.

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

The Pickup and Delivery Traveling Salesman Problem with Handling Costs

Marjolein Veenstra[1,*], Kees Jan Roodbergen[1], Iris F.A. Vis[1], Leandro C. Coelho[1,2]

[1] Department of Operations, University of Groningen, P.O. Box 72, 9700 AB Groningen, The Netherlands

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325 de la Terrasse, Université Laval, Québec, Canada G1V 0A6

Abstract. This paper introduces the pickup and delivery traveling salesman problem with handling costs (PDTSPH). In the PDTSPH, a single vehicle has to transport loads from origins to destinations. Loading and unloading of the vehicle is operated in a last-in-first-out (LIFO) fashion. However, if a load must be unloaded that was not loaded last, additional handling operations are allowed to unload and reload other loads that block access. Since the additional handling operations take time and effort, penalty costs are associated with them. The aim of the PDTSPH is to find a feasible route such that the total costs, consisting of travel costs and penalty costs, are minimized. We show that the PDTSPH is a generalization of the pickup and delivery traveling salesman problem (PDTSP) and the pickup and delivery traveling salesman problem with LIFO loading (PDTSPL). We propose a large neighborhood search (LNS) heuristic to solve the problem. We compare our LNS heuristic against best known solutions on 163 benchmark instances for the PDTSP and 42 benchmark instances for the PDTSPL. We provide new best known solutions on 52 instances for the PDTSP and on 14 instances for the PDTSPL, besides finding the optimal or best known solution on 100 instances for the PDTSP and on 21 instances for the PDTSPL. The LNS finds optimal or near-optimal solutions on instances for the PDTSPH. Results show that PDTSPH solutions provide large reductions in handling compared to PDTSP solutions, while increasing the travel distance by only a small percentage.

Keywords. Routing, pickup and delivery, packing, handling costs, metaheuristics.

Acknowledgements: This project was funded by the Dutch Institute for Advanced Logistics (Dinalog) and partly funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant 2014-05764. This support is greatly acknowledged.

---

* Corresponding author: marjolein.veenstra@rug.nl

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2015

1 Introduction

In this paper, we introduce, model and solve the pickup and delivery traveling salesman problem with handling costs (PDTSPH). In the PDTSPH, a single vehicle based at a central depot must fulfill a set of requests. Each request determines the transportation of items from a specific pickup location, where the items are loaded into the vehicle, to a specific delivery location, where the items are unloaded from the vehicle. We consider a rear-loaded vehicle with a single (horizontal) stack that is operated in a last-in-first-out (LIFO) fashion. At a pickup location, items are placed on top of the stack. At a delivery location, if an item is not on top of the stack, there are items on top blocking the access, and handling operations are required to unload and reload the items that blocked the access. Since the additional handling operations of unloading and reloading items take time and effort, penalty costs are associated with them. Figure 1 shows two feasible routes, in which route (1a) requires no additional handling, whereas in route (1b) an additional handling operation is needed to move item 3 upon delivery of item 2. The aim of the PDTSPH is to find a feasible route such that the total costs, consisting of travel costs and penalty costs, are minimized.

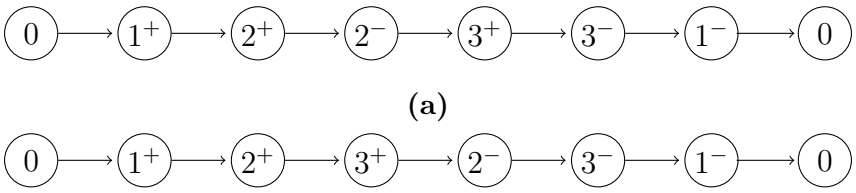


Figure 1: Two feasible routes, in which route (a) does not require any additional handling operations, and route (b) requires an additional handling operation. In the figure, i^+ and i^- correspond to the pickup and delivery location of request i, respectively.

The PDTSPH as presented here arises in the transportation of less-than-truckload items, and is especially faced by freight transportation companies responsible for transporting large items that are easy to load and unload, such as cars. A typical challenge for these

companies is to define a route along all customers by finding a trade-off between travel costs and additional handling operations. Minimizing travel costs can result in routes with a large number of additional handling operations, whereas minimizing the additional handling operations may result in sub-optimal routes with respect to the travel distance. Therefore, it is relevant for these companies to simultaneously take both aspects into account when generating a vehicle route.

To our knowledge, the PDTSPH has not yet been studied in literature. Related is the research of Battarra et al. [2] and Erdoğan et al. [10]. They propose exact and heuristic methods for a problem that considers requests that either originate from or destinate to the depot. This implies that all loads destined to customers are already in the vehicle at the start of the route, and all loads originating from customers are in the vehicle at the end of the route.

The PDTSPH is a generalization of two problems, as we will prove in Section 3, namely the pickup and delivery traveling salesman problem (PDTSP), and the pickup and delivery traveling salesman problem with LIFO loading (PDTSPL). The aim of the PDTSP is to find a vehicle route that fulfills all requests and minimizes transportation costs. Additional handling operations are not considered in the PDTSP. Exact and heuristic methods have been proposed for the PDTSP. Savelsbergh [19] describes different local search algorithms, Healy and Moll [11] propose an extension of traditional improvement algorithms, and Renaud et al. [16] develop a two-stage heuristic consisting of a construction phase and a deletion and reinsertion phase. Different perturbation heuristics are proposed by Renaud et al. [17] and Dumitrescu et al. [8] present a branch-and-cut algorithm. The aim of the PDTSPL is to find a vehicle route that completes all requests and minimizes transportation costs, while prohibiting additional handling operations. This implies that a vehicle can only visit a delivery location if the corresponding item is on top of the stack. Exact and heuristic methods have been proposed for the PDTSPL. Cassani and Righini [5] propose a variable neighborhood descent heuristic, Carrabs et al. [4] introduce a variable neighborhood search (VNS) heuristic, Cordeau et al. [6] develop a branch-and-cut

algorithm, Li et al. [13] present a VNS heuristic based on a tree representation, Côté et al. [7] describe a large neighborhood search heuristic, and Wei et al. [21] propose a VNS with a different perturbation operator.

The contribution of our paper is fourfold: (1) we formally describe and formulate the PDTSPH, (2) we prove that the problem is a generalization of the PDTSP and of the PDTSPL, and (3) we derive a heuristic solution method to efficiently solve the problem and its special cases. Namely, we propose a large neighborhood search (LNS) metaheuristic, that includes new removal operators, and is shown to provide good quality solutions. (4) As a part of extensive computational results on benchmark instances for the PDTSP, the PDTSPL, and for the newly defined PDTSP, we provide new best known solutions on 52 instances for the PDTSP and on 14 instances for the PDTSPL.

The remainder of this paper is structured as follows. In Section 2, we develop a binary integer programming formulation for the PDTSPH. In Section 3, we prove that the PDTSPH is a generalization of the PDTSP and the PDTSPL. Section 4 describes the proposed LNS heuristic. Section 5 reports the experimental setting and the results of extensive computational experiments performed on the three classes of problems, followed by conclusions in Section 6.

2 Mathematical Formulation

The PDTSPH is defined on a directed graph $G = (V, A)$, where V is the set of nodes and A is the set of arcs. Let n be the number of requests. The set of nodes is given by $V = \{0, 1, \ldots, 2n\}$, where 0 corresponds to the depot, $P = \{1, \ldots, n\}$ is the set of pickup nodes, and $D = \{n + 1, \ldots, 2n\}$ is the set of delivery nodes. Let $V' = V \backslash \{0\}$ be the set of nodes excluding the depot, and let A' be the subset of arcs having both endpoints in V'. Each request i is associated to a pickup node $i \in P$ and a delivery node $(n + i) \in D$. For convenience, we also refer to the set P as the set of requests. Each request corresponds to the transportation of one item. The travel cost of arc $(i, j) \in A$

corresponds to the travel distance and is given by c_{ij}. We assume that c_{ij} satisfies the triangle inequality. An additional handling operation consists of unloading and reloading an item at a location. We only allow an item to be unloaded and reloaded when it is blocking the delivery operation, i.e., if an item is on top of the item to be delivered. We assume that the reloading sequence is the inverse of the unloading sequence, i.e., the relative positions of the items remain the same. The penalty cost associated to an additional handling operation is fixed and given by h. The number of handling operations corresponding to loading the items at their pickup locations and unloading them at their delivery locations is constant and cannot be avoided. Therefore, without loss of generality, in our formulation no penalty costs are associated to them.

The flow based formulation of the binary integer program for the PDTSPH is based on the model of Erdoğan et al. [9] for the PDTSP. Let x_{ij} be a binary variable equal to one if and only if arc $(i,j) \in A$ is traveled by the vehicle. Let y^1_{ijk}, y^2_{ijk} and y^3_{ijk} be three binary flow variables. Variable y^1_{ijk} is equal to one if and only if arc $(i,j) \in A$ is on the partial path from node 0 to node k; variable y^2_{ijk} is equal to one if and only if arc $(i,j) \in A$ is on the partial path from node k to node $n+k$; variable y^3_{ijk} is equal to one if and only if arc $(i,j) \in A$ is on the partial path from node $n+k$ to node 0. We introduce binary variables r_{kl} $\forall k \in P$, $l \in D$, equal to one if and only if an additional handling operation is associated to item k at delivery node l. Then, the PDTSPH is formulated as:

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij} + h\sum_{k\in P}\sum_{l\in D} r_{kl} \tag{1}$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in A} x_{ij} = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{i:(i,j)\in A} x_{ij} = 1 \qquad \forall j \in V \tag{3}$$

$$\sum_{j:(i,j)\in A} y^1_{ijk} - \sum_{j:(j,i)\in A} y^1_{jik} = \begin{cases} 1 & \text{if } i = 0 \\ -1 & \text{if } i = k \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in V,\ \forall k \in P \tag{4}$$

$$\sum_{j:(i,j)\in A} y^2_{ijk} - \sum_{j:(j,i)\in A} y^2_{jik} = \begin{cases} 1 & \text{if } i = k \\ -1 & \text{if } i = n + k \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in V,\ \forall k \in P \tag{5}$$

$$\sum_{j:(i,j)\in A} y^3_{ijk} - \sum_{j:(j,i)\in A} y^3_{jik} = \begin{cases} 1 & \text{if } i = n + k \\ -1 & \text{if } i = 0 \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in V',\ \forall k \in P \tag{6}$$

$$y^1_{ijk} + y^3_{ijk} = x_{ij} \qquad \forall (i,j) \in A \backslash A',\ k \in P \tag{7}$$

$$y^1_{ijk} + y^2_{ijk} + y^3_{ijk} = x_{ij} \qquad \forall (i,j) \in A',\ k \in P \tag{8}$$

$$r_{kl} \geq \sum_{i:(i,l-n)\in A'} y^2_{i,l-n,k} - \sum_{i:(l,i)\in A'} y^2_{lik} \qquad \forall k \in P,\ l \in D \tag{9}$$

$$r_{kl} \in \{0,1\} \qquad \forall k \in P,\ l \in D \tag{10}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{11}$$

$$y^1_{ijk},\ y^3_{ijk} \in \{0,1\} \qquad \forall (i,j) \in A,\ k \in P \tag{12}$$

$$y^2_{ijk} \in \{0,1\} \qquad \forall (i,j) \in A',\ k \in P \tag{13}$$

The objective function (1) minimizes the total costs, consisting of the travel costs and penalty costs associated to the additional handling operations. Constraints (2) and (3) are standard degree constraints. Constraints (4) ensure that there is a path from the depot to each pickup node. Due to constraints (5), the pickup node of a request is visited before its corresponding delivery node. Constraints (6) ensure that there is a path from each delivery node to the depot. Moreover, constraints (4)–(6) eliminate subtours. Constraints (7) and (8) couple the overall routing variables with the flow variables. Constraints (9) are introduced for this specific problem and ensure that variables r_{kl} are equal to one if

item $k \in P$ is on top of item $l-n$ when arriving at delivery node $l \in D$. When minimizing the objective function (1), variable r_{kl} will be equal to one if and only if an additional handling operation is associated to item k at delivery node l. Constraints (10)–(13) define the domain and nature of the variables.

3 Special Cases of the PDTSPH

In this section, we prove that the PDTSPH is a generalization of both the PDTSP, where handling operations are not considered, and the PDTSPL, where handling operations are not allowed. First, we prove that the PDTSP and the PDTSPH with penalty cost $h = 0$ are equivalent. Then, we show that the PDTSPL and the PDTSPH with $h > 2 \max_{k \in P} c_{k,n+k}$ are equivalent. At the end, we compare the size of the feasible solution space of the PDTSPH, the PDTSP and the PDTSPL.

Theorem 1. *The PDTSP is equivalent to the PDTSPH with $h = 0$.*

Proof. Let a problem instance be given. Then, if we set $h = 0$ for the PDTSPH, the objective of the PDTSPH and the objective of the PDTSP are to minimize travel costs and constraints (9) become redundant. Hence, it can easily be seen that the PDTSP is equivalent to the PDTSPH with $h = 0$. □

Theorem 2. *The PDTSPL is equivalent to the PDTSPH with $h > 2 \max_{k \in P} c_{k,n+k}$.*

Proof. Let a problem instance be given. By contradiction, assume there exists an optimal solution to the PDTSPH with $h > 2 \max_{k \in P} c_{k,n+k}$ and $k' \in P$, $l \in D$ such that $r_{k'l} > 0$. Let this solution be represented by $s^* = (0, \ldots, k', \ldots, n+k', \ldots, 2n+1)$. Let s' be the route derived from route s^*, where node $n+k'$ is visited directly after node k', that is $s' = (0, \ldots, k', n+k', \ldots, 2n+1)$. Since the triangle inequality holds, the objective value f of solution s' can be expressed by $f(s') \leq f(s^*) + 2c_{k',n+k'} - h < f(s^*) + 2c_{k',n+k'} - 2 \max_{k \in P} c_{k,n+k} \leq f(s^*)$. This contradicts the assumption that s^* is optimal. Hence, there

does not exist $k' \in P$, $l \in D$ such that $r_{k'l} > 0$. Hence, an optimal solution to the PDTSPH with $h > 2 \max_{k \in P} c_{k,n+k}$ does not contain any additional handling operations. Therefore, it can easily be seen that the PDTSPL is equivalent to the PDTSPH with $h > 2 \max_{k \in P} c_{k,n+k}$. □

The feasible solution space of the PDPTSP has the same size as the PDTSPH, whereas the feasible solution space of the PDTSP is only a small subset of the solution space of the PDTSPH. The number of feasible solutions with n requests for the three problem classes are given in Table 1, which is based on Table 1 given in Cordeau et al. [6].

Table 1: The number of feasible solutions for n requests.

n	PDTSPH	PDTSP	PDTSPL
1	1	1	1
2	6	6	4
3	90	90	30
4	2,520	2,520	336
5	113,400	113,400	5,040
6	7,484,400	7,484,400	95,040
7	681,080,400	681,080,400	2,162,160
8	81,729,648,000	81,729,648,000	57,657,600

4 Large Neighborhood Search Heuristic

In this section, we present our large neighborhood search (LNS) heuristic for the PDTSPH. The concept of LNS was introduced by Shaw [20] for the vehicle routing problem with time windows. It has been extended and successfully applied to many different routing problems, see, e.g., Azi et al. [1], Côté et al. [7], and Masson et al. [14]. The general structure of the LNS heuristic is to iteratively destroy and repair a solution in order to improve it, as depicted in Algorithm 1. The procedure starts with an initial solution (line 1). Several operators are used to iteratively destroy and repair the current solution (lines

5–6). The acceptance of a solution is determined by a simulated annealing-based acceptance criteria (line 9) Kirkpatrick et al. [12]. The procedure continues until a stopping criteria is met (line 3).

```
1  Construct initial solution S;
2  S_BEST = S;
3  while stopping criteria is not met do
4  |   S' = S;
5  |   Remove q requests from S';
6  |   Reinsert removed requests in S';
7  |   if f(S') < f(S_BEST) then
8  |   |   S_BEST = S';
   |   end
9  |   if Accept(S', S) then
10 |   |   S = S';
   |   end
   end
11 Result: S_BEST;
```

Algorithm 1: Structure of the LNS heuristic

Details about the initial solution are provided in Section 4.1, the list of removal operators is presented in Section 4.2, the reinsertion procedure is described in Section 4.3, and the acceptance and stopping criteria are detailed in Section 4.4.

4.1 Initial Solution

The route corresponding to the initial solution is constructed incrementally, starting from a route consisting of only the depot. For each request, the costs for inserting the pickup and delivery node at their best positions are calculated. The pickup and delivery nodes corresponding to the request with the smallest cost increase are inserted in their best

positions. The procedure continues until all pickup and delivery nodes are inserted in the route. This procedure corresponds to the *basic greedy heuristic* proposed by Ropke and Pisinger [18].

4.2 Removal Operators

At each iteration, the current solution is modified by removing and later reinserting the pickup and delivery nodes corresponding to a set of requests. The number of requests to be removed is set equal to q, which is a random number dependent on the instance size. One of several removal operators is randomly selected and it determines the specific set of requests to be removed from the solution. We propose five different operators. Two of them, the *random removal* operator and the *worst removal* operator, are described by Ropke and Pisinger [18]. The three other operators are newly introduced: the *worst distance removal* and the *worst handling removal* operators, which are based on the *worst removal* operator mentioned before; and the *block removal* operator, which is inspired by the *relocate-block* operator introduced by Cassani and Righini [5]. These five operators are now described.

4.2.1 Random Removal

The *random removal* operator randomly selects q requests and removes the pickup and delivery nodes corresponding to them.

4.2.2 Worst Removal

In the *worst removal* operator, introduced by Ropke and Pisinger [18], the cost of a request $k \in P$ in route s is given by $c_k(s) = f(s) - f_{-k}(s)$, i.e., the objective value of the route minus the objective value of the route after removing nodes k and $n + k$. The selection of q requests is done randomly, and the probability of selecting a request increases with the costs. This is done as follows. Let L be the array of all requests sorted by descending

cost. Let y be randomly drawn from the interval $[0, 1)$. Request $r = L[\lceil y^p |L| \rceil]$ is selected, where p is a parameter defined in the experimental setting. Request r is removed from from array L. The procedure continues until q requests are selected. After the selection of q requests, the pickup and delivery nodes corresponding to these requests are removed from the route.

4.2.3 Worst Distance Removal

The *worst distance removal* operator is based on the *worst removal* operator. The cost of a request $k \in P$ is given by $\tilde{c}_k(s) = f^d(s) - f^d_{-k}(s)$, where $f^d(s)$ corresponds to the distance costs of route s. The selection of q requests is done at random, and the probability of selecting a request increases with the costs, as previously explained. The pickup and delivery nodes corresponding to the selected requests are removed from the solution.

4.2.4 Worst Handling Removal

The *worst handling removal* operator works similar to the worst distance removal operator, but with the costs based on penalty costs for additional handling operations instead of routing distances.

4.2.5 Block Removal

The *block removal* operator randomly selects one request $k \in P$. The requests for which the pickup and/or delivery nodes are between nodes k and $n + k$ are also selected. The pickup and delivery nodes corresponding to the selected requests are removed from the solution. If the number of involved requests exceeds q, we limit the number of removed requests to the first q requests, since we do not want to destroy too much of the route.

4.3 Insertion Operator

The insertion operator reinserts requests in the route that were previously removed by the removal operators. The insertion of pickup and delivery nodes is computationally intensive, because it not only requires the computation of the difference in routing costs, but also the calculation of the difference in additional handling operations. In our LNS heuristic, we only apply the *greedy insertion* operator. Preliminary experiments have shown that incorporating other operators such as the *basic greedy heuristic* and the *regret heuristic* proposed in Ropke and Pisinger [18] do not lead to significant improvements, while increasing the computation time significantly.

The *greedy insertion* operator randomly determines the sequence of the requests to be inserted. Then, based on this sequence, the pickup and delivery nodes corresponding to the current request are each inserted at their best position.

4.4 Acceptance and Stopping Criteria

In a given iteration, a new solution is accepted if it is better than the current solution. If the new solution is worse than the current solution, the acceptance of the new solution is determined by a simulated annealing criterion Kirkpatrick et al. [12]. Given the current solution s, a new solution s' is accepted with probability $e^{-(f(s')-f(s))/T}$, where $f(s)$ denotes the objective value of solution s, and $T > 0$ is the temperature at the given iteration. At the start of the solution procedure the temperature is set to T_{start}, after which the temperature decreases each iteration by $T = c \cdot T$, where $0 < c < 1$ is the cooling rate. The stopping criteria is based on the number of iterations performed. The maximum number of iterations, and the values for the parameters T_{start} and c, are determined by the experiments reported in Section 5.

5 Computational Experiments

The LNS heuristic was coded in Java and the experiments were performed on computers equipped with ADM Opteron 2435 processors and 24GB of RAM memory. Our code uses a single thread and no more than 0.5 GB of memory even on large instances. We describe the parameter setting for the LNS heuristic in Section 5.1. In Section 5.2 we compare the results obtained by the LNS heuristic with the results reported in the literature on special cases of the problem, namely the PDTSP and the PDTSPL. Thereafter, in Section 5.3 we compare the results obtained by the LNS heuristic with the results of a branch-and-bound algorithm applied to the binary integer program from Section 2 on small instances of the general PDTSPH. Finally, an evaluation on the trade-off between the routing costs and penalty costs corresponding to the additional handling operations is presented in Section 5.4.

5.1 LNS Parameter Settings

For the purpose of tuning the parameters of our heuristic, we created a test set of PDTSPH problems. This test set contains 96 instances with up to 101 nodes, which are derived from 24 instances of the set of Carrabs et al. [4]. These were in turn derived from the six instances $fnl4461$, $brd14051$, $d15112$, $d18512$, $nrw1379$, and $pr1002$ of TSPLIB [15]. For each of the 24 instances, we have created four new instances by setting the penalty cost of an additional handling operation to 0, 10, 100 and 100000. This makes sure that we have instances for the PDTSP, if the penalty cost is set to 0, for the PDTSPL, if the penalty cost is set to 100000, and something in between. We have modified the instances by interchanging the delivery locations corresponding to the requests as follows. The delivery location corresponding to item $i \in P$, $i \neq n$, is matched with the pickup location of item $i + 1$, and the delivery location corresponding to item $i = n$ is matched with the pickup location of item $i = 1$.

We start our parameter tuning by setting the parameters based on settings from literature.

Based on Ropke and Pisinger [18], the start temperature is set such that the probability of accepting a solution that is 5% worse than the current solution is equal to 0.5 and the cooling rate c at 0.999875716 such that the temperature at the last iteration is 0.2% of the start temperature. The parameter p is set to 3. The number of removed requests in one iteration q is a random value in the interval $[\min\{30, 0.20n\}, \min\{50, 0.55n\}]$, which is based on the experiments in Côté et al. [7]. Based on preliminary analysis, we set the number of iterations to 50000, which results in a good balance between solution quality and computation time.

We have validated these parameter setting by changing each of the parameter values individually to smaller and larger values, while keeping the remaining parameters fixed. We did not find significant improvements for one of the alternative parameter settings. Therefore, we have decided to keep the values as specified above. The ALNS heuristic described by Ropke and Pisinger [18] incorporates an adaptive mechanism that updates the probabilities of the operators based on their performance. We observed that incorporating the adaptive mechanism in our heuristic did not yield significant improvements.

5.2 Assessment of the Heuristic on Related Problems

We now investigate the results obtained by our LNS heuristic for the PDTSPH on benchmark instances for the PDTSP and the PDTSPL. We compare the results for four different instance sets.

5.2.1 Performance on the PDTSP

The first instance set is proposed by Dumitrescu et al. [8]. The data set contains 35 randomly generated instances with up to 71 nodes. The authors solved the problem by branch-and-cut, which yielded optimal solutions for 28 instances. For the other instances, the authors propose an upper bound, which is the best solution over a large number of runs of an LNS heuristic. Consistent with literature (e.g., [4], [13]), we have run our LNS

heuristic ten times. We have compared the average results and the best results with those obtained by Dumitrescu et al. [8]. Comparing the average results, we see that our LNS heuristic finds identical results for 29 instances, and on average performs 0.04% worse. Comparing the best results over the ten runs, we find the same results as Dumitrescu et al. [8] for all but one instance, and for the remaining instance we improve the best known solution by 0.11%. These results are presented in Table 2.

Table 2: Results for the PDTSP instances of Dumitrescu et al. [8]

			LNS: Average results		LNS: Best results		
Instance	Nodes	Best cost [8]	Cost	Gap (%)	Cost	Gap (%)	Time (s)
prob5a	11	3585 *	**3585**	0.00	**3585**	0.00	7.3
prob5b	11	2565 *	**2565**	0.00	**2565**	0.00	2.4
prob5c	11	3787 *	**3787**	0.00	**3787**	0.00	1.3
prob5d	11	3128 *	**3128**	0.00	**3128**	0.00	1.1
prob5e	11	3123 *	**3123**	0.00	**3123**	0.00	0.9
prob10a	21	4896 *	**4896**	0.00	**4896**	0.00	2.0
prob10b	21	4490 *	**4490**	0.00	**4490**	0.00	1.9
prob10c	21	4070 *	**4070**	0.00	**4070**	0.00	2.0
prob10d	21	4551 *	**4551**	0.00	**4551**	0.00	1.7
prob10e	21	4874 *	**4874**	0.00	**4874**	0.00	1.8
prob15a	31	5150 *	**5150**	0.00	**5150**	0.00	4.4
prob15b	31	5391 *	5414.2	0.43	**5391**	0.00	4.4
prob15c	31	5008 *	**5008**	0.00	**5008**	0.00	4.3
prob15d	31	5566 *	**5566**	0.00	**5566**	0.00	4.4
prob15e	31	5229 *	**5229**	0.00	**5229**	0.00	4.4
prob20a	41	5698 *	**5698**	0.00	**5698**	0.00	8.8
prob20b	41	6213 *	**6213**	0.00	**6213**	0.00	8.6
prob20c	41	6200 *	**6200**	0.00	**6200**	0.00	8.6
prob20d	41	6106 *	**6106**	0.00	**6106**	0.00	8.7
prob20e	41	6465 *	**6465**	0.00	**6465**	0.00	8.9
prob25a	51	7332	**7332**	0.00	**7332**	0.00	15.1
prob25b	51	6665 *	6665.8	0.01	**6665**	0.00	15.6
prob25c	51	7095 *	7099.3	0.06	**7095**	0.00	15.8
prob25d	51	7069 *	7122.6	0.76	**7069**	0.00	15.2
prob25e	51	6754 *	**6754**	0.00	**6754**	0.00	15.5
prob30a	61	7309	**7309**	0.00	**7309**	0.00	31.7
prob30b	61	6857 *	**6857**	0.00	**6857**	0.00	25.0
prob30c	61	7723 *	**7723**	0.00	**7723**	0.00	25.5
prob30d	61	7310 *	**7310**	0.00	**7310**	0.00	25.5

Continued on next page

Table 2 – *Continued from previous page*

			LNS: Average results		LNS: Best results		
Instance	Nodes	Best cost [8]	Cost	Gap (%)	Cost	Gap (%)	Time (s)
prob30e	61	7213	**7213**	0.00	**7213**	0.00	25.9
prob35a	71	7746 *	**7746**	0.00	**7746**	0.00	45.6
prob35b	71	7904	**7904**	0.00	**7904**	0.00	36.8
prob35c	71	7949	7957.3	0.10	**7949**	0.00	38.6
prob35d	71	7905	**7905**	0.00	**7905**	0.00	38.7
prob35e	71	8530	8540.4	0.12	**8521**	-0.11	37.7
Average		5927.3	5930.2	0.04	5927.1	0.00	14.2

* indicates proven optimal

The second data set is proposed by Renaud et al. [16] and consists of 108 instances containing between 51 and 493 nodes, and is derived from TSP instances from the TSPLIB [15]. The data set is used to report results for the heuristics proposed in Renaud et al. [17], and Renaud et al. [16]. Moreover, Dumitrescu et al. [8] report the results, of either their branch-and-cut algorithm or a heuristic upper bound, on a subset of 33 of these instances. We have run our LNS heuristic ten times and have compared the best results of our LNS heuristic with the best results obtained by Renaud et al. [17], Renaud et al. [16], and Dumitrescu et al. [8]. We have received the results of Renaud et al. [17] and Renaud et al. [16] from the authors upon request. In Table 3 we report our new best known solutions for 51 instances. On average, the results obtained by our LNS heuristic are 0.75% better than the best known solutions. The largest improvement is 4.71%, the worst solution obtained by our LNS heuristic is 0.56% worse than the best known solution.

Table 3: Results for the PDTSP instances of Renaud et al. [16]

Instance	Nodes	Best cost [8],[17],[16]	Best cost LNS	Gap (%)	Time (s)
EIL51A	51	464	**464**	0.00	17.3
EIL51B	51	469	**469**	0.00	14.6
EIL51C	51	488	**488**	0.00	15.3
ST69A	69	764	**764**	0.00	29.8
ST69B	69	771	**771**	0.00	28.7
ST69C	69	793	**793**	0.00	35.8

Continued on next page

Table 3 – *Continued from previous page*

Instance	Nodes	Best cost [8],[17],[16]	Best cost LNS	Gap (%)	Time (s)
EIL75A	75	583	584	0.17	36.9
EIL75B	75	601	**601**	0.00	41.0
EIL75C	75	590	**590**	0.00	46.1
PR75A	75	130531	**130531**	0.00	38.5
PR75B	75	128397	**128397**	0.00	39.4
PR75C	75	124509	**124509**	0.00	45.8
KROA99A	99	24980	**24980**	0.00	80.2
KROA99B	99	26552	**26552**	0.00	90.6
KROA99C	99	25769	**25769**	0.00	99.5
KROB99A	99	25631	**25631**	0.00	78.2
KROB99B	99	25384	**25384**	0.00	87.7
KROB99C	99	25795	**25795**	0.00	99.3
KROC99A	99	26113	**26113**	0.00	81.4
KROC99B	99	25602	**25602**	0.00	85.4
KROC99C	99	26065	**26065**	0.00	98.0
KROD99A	99	25392	**25427**	0.14	76.3
KROD99B	99	26179	26195	0.06	85.1
KROD99C	99	26021	26041	0.08	98.0
KROE99A	99	25879	**25879**	0.00	77.9
KROE99B	99	26584	26591	0.03	84.9
KROE99C	99	26021	**26021**	0.00	100.9
RAT99A	99	1401	**1401**	0.00	76.5
RAT99B	99	1460	1464	0.27	85.6
RAT99C	99	1370	**1370**	0.00	101.7
RD99A	99	9522	**9506**	-0.17	77.1
RD99B	99	9464	**9464**	0.00	85.9
RD99C	99	9185	**9185**	0.00	101.0
EIL101A	101	695	**695**	0.00	87.8
EIL101B	101	705	**705**	0.00	94.2
EIL101C	101	690	**690**	0.00	107.9
LIN105A	105	17791	**17710**	-0.46	94.0
LIN105B	105	17482	**17482**	0.00	104.7
LIN105C	105	17813	**17813**	0.00	122.8
PR107A	107	51537	**51537**	0.00	94.4
PR107B	107	51675	51686	0.02	108.7
PR107C	107	52657	**52657**	0.00	128.2
PR123A	123	75542	75552	0.01	142.5
PR123B	123	75389	**75389**	0.00	158.6
PR123C	123	82341	**82341**	0.00	194.6

Continued on next page

Table 3 – *Continued from previous page*

Instance	Nodes	Best cost [8],[17],[16]	Best cost LNS	Gap (%)	Time (s)
BIER127A	127	133653	**131974**	-1.26	162.2
BIER127B	127	134670	**133378**	-0.96	183.8
BIER127C	127	132972	**132199**	-0.58	206.0
PR135A	135	111475	**111120**	-0.32	187.9
PR135B	135	110763	**110763**	0.00	207.2
PR135C	135	114232	**114232**	0.00	257.8
PR143A	143	80274	**80274**	0.00	206.8
PR143B	143	90484	**89472**	-1.12	221.3
PR143C	143	91979	**91979**	0.00	259.5
KROA149A	149	31467	**30833**	-2.01	235.9
KROA149B	149	31733	**31733**	0.00	252.6
KROA149C	149	32351	**32235**	-0.36	309.7
KROB149A	149	31360	**31045**	-1.00	251.6
KROB149B	149	31995	**31696**	-0.93	262.9
KROB149C	149	31771	**31735**	-0.11	306.3
PR151A	151	90494	**90494**	0.00	254.2
PR151B	151	94937	**94937**	0.00	265.6
PR151C	151	97288	**97288**	0.00	299.4
U159A	159	51710	**51710**	0.00	288.4
U159B	159	50494	**50494**	0.00	304.8
U159C	159	53189	53242	0.10	359.1
D197A	197	16032	**15886**	-0.91	568.4
D197B	197	16435	**16060**	-2.28	595.7
D197C	197	16724	**16673**	-0.30	713.8
KROA199A	199	34484	**34400**	-0.24	597.4
KROA199B	199	35259	**34899**	-1.02	629.1
KROA199C	199	36251	**35927**	-0.89	739.6
KROB199A	199	34774	34891	0.34	607.2
KROB199B	199	34806	**34212**	-1.71	636.2
KROB199C	199	36784	**35761**	-2.78	743.3
PR225A	225	105321	**103652**	-1.58	823.3
PR225B	225	109502	**109384**	-0.11	1250.4
PR225C	225	113116	**112240**	-0.77	1517.5
TS225A	225	156646	157524	0.56	1249.4
TS225B	225	161353	**159163**	-1.36	966.7
TS225C	225	165073	**161278**	-2.30	1127.1
GIL261A	261	2808	**2773**	-1.25	2335.1
GIL261B	261	2887	**2837**	-1.73	2470.2
GIL261C	261	2885	**2812**	-2.53	1951.5

Continued on next page

Table 3 – *Continued from previous page*

Instance	Nodes	Best cost [8],[17],[16]	Best cost LNS	Gap (%)	Time (s)
PR263A	263	61805	**60936**	-1.41	1832.3
PR263B	263	60489	**60445**	-0.07	2090.3
PR263C	263	65514	**64161**	-2.07	2333.0
PR299A	299	57532	**57279**	-0.44	5450.0
PR299B	299	59342	**57629**	-2.89	5428.2
PR299C	299	59436	**58860**	-0.97	6642.3
LIN317A	317	51303	**49729**	-3.07	6244.9
LIN317B	317	51444	**50573**	-1.69	6610.8
LIN317C	317	51257	**50657**	-1.17	7670.7
RD399A	399	18101	**17356**	-4.12	11192.5
RD399B	399	18451	**18143**	-1.67	11841.4
RD399C	399	18839	**18172**	-3.54	13711.7
FL417A	417	13874	**13792**	-0.59	12268.8
FL417B	417	14089	**13815**	-1.94	12503.2
FL417C	417	15618	**15618**	0.00	15155.4
PR439A	439	109424	**105887**	-3.23	14250.7
PR439B	439	115580	**110139**	-4.71	15252.2
PR439C	439	113514	**112121**	-1.23	17358.7
PCB441A	441	61289	**60225**	-1.74	14654.4
PCB441B	441	61691	**59615**	-3.37	15128.5
PCB441C	441	61984	**60569**	-2.28	17665.0
D493A	493	37725	**36007**	-4.55	19236.5
D493B	493	37806	**36888**	-2.43	19515.9
D493C	493	38794	**37986**	-2.08	23067.4
Average		48323.9	47902.4	-0.75	2829.5

The third data set is proposed by Renaud et al. [17] and contains 20 instances that are generated from an optimal TSP tour. Ten instances contain 101 nodes, the other ten contain 201 nodes. The branch-and-cut algorithm of Dumitrescu et al. [8] finds optimal solutions for all these instances. As shown in Table 4, when running our heuristic ten times, our heuristic always finds the optimal solution, whereas the heuristic of Renaud et al. [17] does not find an optimal solution for all instances.

Table 4: Results for the PDTSP instances of Renaud et al. [17]

			LNS: Average results		LNS: Best results		
Instance	Nodes	Best cost [8]	Cost	Gap (%)	Cost	Gap (%)	Time (s)
N101P1	101	799 *	**799.0**	0.00	**799**	0.00	110.0
N101P2	101	729 *	**729.0**	0.00	**729**	0.00	103.4
N101P3	101	748 *	**748.0**	0.00	**748**	0.00	104.5
N101P4	101	807 *	**807.0**	0.00	**807**	0.00	104.5
N101P5	101	783 *	**783.0**	0.00	**783**	0.00	102.5
N101P6	101	755 *	**755.0**	0.00	**755**	0.00	102.3
N101P7	101	767 *	**767.0**	0.00	**767**	0.00	104.0
N101P8	101	762 *	**762.0**	0.00	**762**	0.00	104.8
N101P9	101	766 *	**766.0**	0.00	**766**	0.00	104.8
N101P10	101	754 *	**754.0**	0.00	**754**	0.00	105.9
N201P1	201	1039 *	**1039.0**	0.00	**1039**	0.00	1162.6
N201P2	201	1086 *	**1086.0**	0.00	**1086**	0.00	765.2
N201P3	201	1070 *	1071.1	0.10	**1070**	0.00	764.4
N201P4	201	1050 *	1050.1	0.01	**1050**	0.00	905.8
N201P5	201	1052 *	**1052.0**	0.00	**1052**	0.00	765.8
N201P6	201	1059 *	**1059.0**	0.00	**1059**	0.00	768.2
N201P7	201	1036 *	**1036.0**	0.00	**1036**	0.00	1043.7
N201P8	201	1079 *	**1079.0**	0.00	**1079**	0.00	757.8
N201P9	201	1050 *	**1050.0**	0.00	**1050**	0.00	756.6
N201P10	201	1085 *	**1085.0**	0.00	**1085**	0.00	767.2
Average		913.8	913.9	0.01	913.8	0.00	475.2

* indicates proven optimal

Table 5: Results for the PDTSPL instances of Carrabs et al. [4]; Average results over 10 runs

Instance	Nodes	Cost [4]	Cost [13]	Cost [7]	Cost [21]	Cost LNS	Gap with [4] (%)	Gap with [13] (%)	Gap with [7] (%)	Gap with [21] (%)	Time (s)
brd14051	25	4682.2	4672.0	4672.0	4672.0	**4672**	-0.22	0.00	0.00	0.00	10.1
	51	7763.2	7740.0	7740.0	7740.0	**7740**	-0.30	0.00	0.00	0.00	13.4
	75	7309.1	7232.4	7232.0	7232.4	**7232**	-1.05	-0.01	0.00	-0.01	36.7
	101	10005.2	9735.0	9731.8	9733.2	9734.6	-2.70	0.00	0.03	0.01	90.3
	251	24119.3	22566.0	22650.8	22729.0	22616.8	-6.23	0.23	-0.15	-0.49	2441.5
	501	52806.8	50369.9	50865.8	50076.3	50364	-4.63	-0.01	-0.99	0.57	21105.9
	751	86230.1	82983.1	83500.3	82223.6	82453.1	-4.38	-0.64	-1.25	0.28	61861.4
pr1002	25	16221.0	16221.0	16221.0	16221.0	**16221**	0.00	0.00	0.00	0.00	2.8
	51	31186.7	30936.0	30936.0	30936.0	**30936**	-0.80	0.00	0.00	0.00	12.7
	75	46911.0	46673.0	46701.4	46673.0	46701.3	-0.45	0.06	0.00	0.06	34.9
	101	63611.1	61433.0	61611.1	61495.0	61482.6	-3.35	0.08	-0.21	-0.02	83.2
	251	200028.5	190665.4	192502.0	191413.4	190978.1	-4.52	0.16	-0.79	-0.23	2399.1
	501	485042.3	470294.5	474161.0	465868.0	468099.1	-3.49	-0.47	-1.28	0.48	20670.0
	751	819197.7	788885.5	800790.1	790395.2	**787200.6**	-3.91	-0.21	-1.70	-0.40	60967.9
fnl4461	25	2168.0	2168.0	2168.0	2168.0	**2168**	0.00	0.00	0.00	0.00	2.7
	51	4020.0	4020.0	4020.0	4020.0	**4020**	0.00	0.00	0.00	0.00	12.7
	75	5865.0	5739.0	5739.0	5739.0	**5739**	-2.15	0.00	0.00	0.00	34.6
	101	8852.8	8562.0	8557.3	8563.1	8558.6	-3.32	-0.04	0.02	-0.05	84.7
	251	29330.6	28797.9	28802.7	28561.0	28668.2	-2.26	-0.45	-0.47	0.38	2380.8
	501	71208.5	68876.6	69384.5	68911.0	**68873.8**	-3.28	0.00	-0.74	-0.05	20901.2
	751	118383.1	114030.0	114993.8	112902.0	114019.2	-3.69	-0.01	-0.85	0.99	61729.6
d18512	25	4683.4	4672.0	4672.0	4672.0	**4672**	-0.24	0.00	0.00	0.00	2.6
	51	7565.6	7502.0	7502.0	7502.0	**7502**	-0.84	0.00	0.00	0.00	13.0
	75	8781.5	8629.0	8629.0	8629.0	**8629**	-1.74	0.00	0.00	0.00	37.9
	101	10332.4	10256.4	10242.0	10280.6	10249.7	-0.80	-0.07	0.08	-0.30	86.7
	251	24855.4	23466.8	23472.6	23435.2	23466.1	-5.59	0.00	-0.03	0.13	2443.3
	501	52295.6	49544.7	49849.1	48377.9	49262.5	-5.80	-0.57	-1.18	1.83	21347.5
	751	83763.3	80734.1	81568.2	80756.9	**80588.5**	-3.79	-0.18	-1.20	-0.21	61883.0
d15112	25	93981.0	93981.0	93981.0	93981.0	**93981**	0.00	0.00	0.00	0.00	3.2
	51	143575.2	142113.0	142113.0	142113.0	**142113**	-1.02	0.00	0.00	0.00	12.9
	75	201385.4	199047.8	199001.0	199047.8	**199001**	-1.18	-0.02	0.00	-0.02	34.0
	101	276876.8	266925.3	266135.3	265894.5	266807	-3.64	-0.04	0.25	0.34	84.6
	251	589066.9	564182.2	567356.1	564356.4	**563877.5**	-4.28	-0.05	-0.61	-0.08	2364.0
	501	953764.5	926331.2	935452.4	920286.1	926598.2	-2.85	0.03	-0.95	0.69	20812.9
	751	1352866.6	1311002.1	1334500.2	1310555.0	1312268.6	-3.00	0.10	-1.67	0.13	62406.7
nrw1379	25	3194.8	3192.0	3192.0	3192.0	**3192**	-0.09	0.00	0.00	0.00	3.2
	51	5095.0	5055.0	5055.0	5055.0	**5055**	-0.79	0.00	0.00	0.00	12.5
	75	6865.1	6831.0	6831.0	6831.0	**6831**	-0.50	0.00	0.00	0.00	34.2
	101	10197.5	9889.4	9850.4	9829.1	**9823.8**	-3.66	-0.66	-0.27	-0.05	86.2
	251	27936.2	26735.6	26705.9	26521.5	**26513.8**	-5.09	-0.83	-0.72	-0.03	2443.9
	501	60584.5	58441.8	59181.9	58493.7	**58429.5**	-3.56	-0.02	-1.27	-0.11	20737.3
	751	105136.1	101737.7	103606.2	101499.8	102443.4	-2.56	0.69	-1.12	0.93	62070.6
Average		145660.6	141020.7	142425.6	141020.7	140947.2	-2.42	-0.07	-0.41	0.11	12185.6

5.2.2 Performance on the PDTSPL

The fourth data set used to evaluate our heuristic is proposed by Carrabs et al. [4] for the PDTSPL. The set is created from instances of the TSPLIB [15] and contains 42 instances with up to 751 nodes. Results for this set of benchmark instances are provided in Carrabs et al. [4], Li et al. [13], Côté et al. [7] and Wei et al. [21]. For comparison, we provide average results over ten runs as in Carrabs et al. [4], Li et al. [13], Côté et al. [7] and Wei et al. [21]. Our results are presented in Table 5. On average, the gap between the results obtained by our LNS heuristic and the results of Carrabs et al. [4], Li et al. [13], Côté et al. [7] and Wei et al. [21] are -2.42%, -0.07% -0.41% and 0.11%, respectively. Comparing the best results obtained by our LNS heuristic with the best known results in literature, we improve the best known solutions for 14 instances and obtain the same best known solutions for 21 other instances. These results are presented in Table 6.

Table 6: Results for the PDTSPL instances of Carrabs et al. [4]; Best results over 10 runs compared with best known solutions

		Best known results	LNS: Best results	
Instance	Nodes	Cost	Cost	Gap (%)
brd14051	25	4672	**4672**	0.00
	51	7740	**7740**	0.00
	75	7232	**7232**	0.00
	101	9731	**9731**	0.00
	251	22243	22453	0.94
	501	50027	**49894**	-0.27
	751	82223.6	**81522**	-0.85
pr1002	25	16221	**16221**	0.00
	51	30936	**30936**	0.00
	75	46600	46700	0.21
	101	61433	**61433**	0.00
	251	188960	189017	0.03
	501	465868	**464273**	-0.34
	751	788885.5	**783943**	-0.63
fnl4461	25	2168	**2168**	0.00
	51	4020	**4020**	0.00
	75	5739	**5739**	0.00
	101	8530	**8530**	0.00

Continued on next page

Table 6 – *Continued from previous page*

		Best known results	LNS: Best results	
Instance	Nodes	Cost	Cost	Gap (%)
	251	28561	**28525**	-0.13
	501	68502	**68452**	-0.07
	751	112902	113009	0.09
d18512	25	4672	**4672**	0.00
	51	7502	**7502**	0.00
	75	8629	**8629**	0.00
	101	10242	**10242**	0.00
	251	23243	**23190**	-0.23
	501	48377.9	49023	1.33
	751	80734.1	**80262**	-0.58
d15112	25	93981	**93981**	0.00
	51	142113	**142113**	0.00
	75	199001	**199001**	0.00
	101	265191	266388	0.45
	251	562072	**559165**	-0.52
	501	920286.1	**916423**	-0.42
	751	1310555	**1302774**	-0.59
nrw1379	25	3192	**3192**	0.00
	51	5055	**5055**	0.00
	75	6831	**6831**	0.00
	101	9817	**9803**	-0.14
	251	26470	**26437**	-0.12
	501	58441.8	**58064**	-0.65
	751	101499.8	101778	0.27
Average		140769.6	140017.5	-0.05

5.3 Evaluation on Small Instances of the PDTSPH

We now compare the results obtained by the LNS heuristic with optimal solutions obtained by branch-and-bound applied to the binary integer program described in Section 2. We compare our results on a set of 80 small instances derived from a subset of instances proposed by Carrabs et al. [3] for the PDTSPL. We derived our instance set from the instances $att532$, $brd14051$, $d15112$, $d18512$, $fnl4461$, $nrw1379$, $pr1002$, and $ts225$ by

considering 19 and 23 nodes. For each of these 16 instances, we have created five new instances by setting the penalty cost of an additional handling operation equal to 0, 1, 10, 100 and 500. For 78 out of 80 instances we have found an optimal solution. For the other two instances, the gap to the optimal solution is not bigger than 0.36%. These results are presented in Table 7. We also observe that the average running time for the branch-and-bound algorithm was 792.8 seconds, while that of our heuristic was 1.6 seconds.

Table 7: Results for the small PDTSPH instances

			ILP:		LNS: Average results		LNS: Best results		
Instance	Nodes	h	Cost	Time (s)	Cost	Gap (%)	Cost	Gap (%)	Time(s)
att532	19	0	3795	8.3	3795.0	0.00	3795	0.00	7.9
	19	1	3809	8.6	3809.0	0.00	3809	0.00	2.3
	19	10	3911	78.0	3911.0	0.00	3911	0.00	1.6
	19	100	4250	1981.0	4250.0	0.00	4250	0.00	1.5
	19	500	4250	1579.5	4250.0	0.00	4250	0.00	1.3
brd14051	19	0	4347	64.1	4347.0	0.00	4347	0.00	1.4
	19	1	4352	104.2	4352.0	0.00	4352	0.00	1.3
	19	10	4389	6.3	4389.0	0.00	4389	0.00	1.1
	19	100	4555	408.1	4560.0	0.11	4555	0.00	1.1
	19	500	4555	217.2	4571.3	0.36	4555	0.00	1.1
d15112	19	0	69150	6.6	69150.0	0.00	69150	0.00	1.3
	19	1	69164	10.0	69164.0	0.00	69164	0.00	1.3
	19	10	69290	5.8	69290.0	0.00	69290	0.00	1.2
	19	100	70542	7.6	70542.0	0.00	70542	0.00	1.4
	19	500	74452	120.6	74452.0	0.00	74452	0.00	1.2
d18512	19	0	4239	4.2	4239.0	0.00	4239	0.00	1.4
	19	1	4245	7.1	4245.0	0.00	4245	0.00	1.2
	19	10	4288	44.5	4288.0	0.00	4288	0.00	1.3
	19	100	4446	251.3	4446.0	0.00	4446	0.00	1.1
	19	500	4446	529.4	4446.0	0.00	4446	0.00	1.1
fnl4461	19	0	1797	1.4	1797.0	0.00	1797	0.00	1.1
	19	1	1804	2.4	1804.0	0.00	1804	0.00	1.2
	19	10	1847	2.5	1847.0	0.00	1847	0.00	1.1
	19	100	1866	2.5	1866.0	0.00	1866	0.00	1.2
	19	500	1866	2.8	1866.0	0.00	1866	0.00	1.1
nrw1379	19	0	2531	15.8	2531.0	0.00	2531	0.00	1.1
	19	1	2534	18.9	2534.0	0.00	2534	0.00	1.1
	19	10	2553	31.9	2553.0	0.00	2553	0.00	1.1
	19	100	2672	122.9	2672.0	0.00	2672	0.00	1.1
	19	500	2691	334.1	2691.0	0.00	2691	0.00	1.0
pr1002	19	0	12342	1.4	12342.0	0.00	12342	0.00	1.2
	19	1	12351	2.3	12351.0	0.00	12351	0.00	1.1
	19	10	12432	2.4	12432.0	0.00	12432	0.00	1.1
	19	100	12947	4.0	12947.0	0.00	12947	0.00	1.1

Continued on next page

Table 7 – *Continued from previous page*

			ILP:		LNS: Average results		LNS: Best results		
Instance	Nodes	h	Cost	Time (s)	Cost	Gap (%)	Cost	Gap (%)	Time(s)
	19	500	12947	2.5	12947.0	0.00	12947	0.00	1.0
ts225	19	0	18000	3.3	18000.0	0.00	18000	0.00	1.1
	19	1	18002	2.5	18002.0	0.00	18002	0.00	1.0
	19	10	18020	2.7	18020.0	0.00	18020	0.00	1.0
	19	100	18200	3.2	18200.0	0.00	18200	0.00	1.0
	19	500	19000	3.8	19000.0	0.00	19000	0.00	1.0
att532	23	0	4177	4.8	4177.0	0.00	4177	0.00	1.8
	23	1	4186	7.2	4186.0	0.00	4186	0.00	1.8
	23	10	4267	18.8	4267.0	0.00	4267	0.00	1.8
	23	100	4748	773.6	4748.0	0.00	4748	0.00	1.7
	23	500	5038	10242.8	5038.0	0.00	5038	0.00	1.7
brd14051	23	0	4385	357.2	4385.0	0.00	4385	0.00	1.9
	23	1	4396	623.5	4396.0	0.00	4396	0.00	1.9
	23	10	4467	791.5	4467.0	0.00	4467	0.00	1.8
	23	100	4655	6978.4	4655.0	0.00	4655	0.00	1.7
	23	500	4655	2409.7	4655.0	0.00	4655	0.00	1.7
d15112	23	0	72783	5.1	72783.0	0.00	72783	0.00	2.0
	23	1	72804	7.9	72804.0	0.00	72804	0.00	2.0
	23	10	72993	8.3	72993.0	0.00	72993	0.00	2.0
	23	100	74883	12.5	74883.0	0.00	74883	0.00	2.0
	23	500	81587	3999.1	81587.0	0.00	81587	0.00	1.9
d18512	23	0	4277	14.5	4277.0	0.00	4277	0.00	1.9
	23	1	4286	12.3	4286.0	0.00	4286	0.00	1.9
	23	10	4329	198.6	4329.0	0.00	4329	0.00	1.8
	23	100	4588	8293.0	4588.0	0.00	4588	0.00	1.8
	23	500	4658	4243.4	4658.0	0.00	4658	0.00	1.7
fnl4461	23	0	1880	4.2	1880.0	0.00	1880	0.00	1.9
	23	1	1887	6.9	1887.0	0.00	1887	0.00	1.9
	23	10	1926	7.2	1926.0	0.00	1926	0.00	1.8
	23	100	2067	439.3	2067.0	0.00	2067	0.00	1.7
	23	500	2067	908.3	2067.0	0.00	2067	0.00	1.7
nrw1379	23	0	2690	79.3	2690.0	0.00	2690	0.00	1.9
	23	1	2698	106.6	2698.0	0.00	2698	0.00	1.9
	23	10	2755	870.4	2755.0	0.00	2755	0.00	1.9
	23	100	2919	6686.1	2919.0	0.00	2919	0.00	1.8
	23	500	2919	9217.6	2919.0	0.00	2919	0.00	1.8
pr1002	23	0	13267	3.7	13267.0	0.00	13267	0.00	1.9
	23	1	13276	5.9	13276.0	0.00	13276	0.00	1.9
	23	10	13357	5.0	13357.0	0.00	13357	0.00	1.9
	23	100	13872	8.3	13872.0	0.00	13872	0.00	1.7
	23	500	13872	6.7	13872.0	0.00	13872	0.00	1.8
ts225	23	0	22000	15.2	22000.0	0.00	22000	0.00	1.8
	23	1	22002	10.2	22002.0	0.00	22002	0.00	1.7
	23	10	22020	11.0	22020.0	0.00	22020	0.00	1.7
	23	100	22200	13.9	22200.0	0.00	22200	0.00	1.7
	23	500	23000	13.3	23000.0	0.00	23000	0.00	1.7
Average			15471.4	792.8	15471.7	0.01	15471.4	0.00	1.6

5.4 Evaluation of the Handling Costs

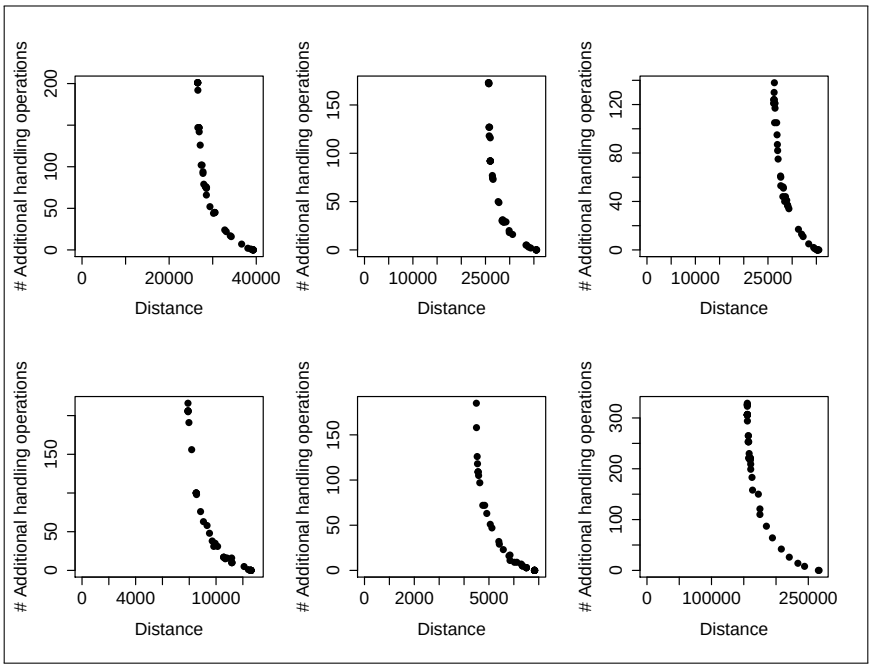
We have generated a data set which is a subset of the instances from the benchmark instance sets of the PDTSP and the PDTSPL mentioned before. The instance set consist of 36 instances each one containing between 71 and 101 nodes. For each of the instances, we set 44 different penalty values. The data set is available upon request.

For each instance and for each penalty value, we have run the LNS heuristic ten times and we have used the results with the smallest objective for the evaluation. On average the number of additional handling operations can be decreased by 52.0% by only increasing the travel distance by 6.0%, compared to the solutions obtained by setting the penalty costs equal to zero. On average, a decrease in additional handling operations of 76.9% and 84.5% results in an increase in travel distance of 16.9% and 24.7%, respectively. Eliminating all additional handling operations does come at a large increase in distance, namely of 57.7%. For six instances, the generated results are depicted in Figure 2, where the number of additional handling operations is plotted against the travel distance for each of the penalty values. The graph shows the trade-off between the distance and the number of additional handling operations. We can see, both from the average results and from the graphs in Figure 2, that a large number of additional handling operations can be eliminated without significantly increasing the travel distance, compared to the solutions obtained by only considering the travel distance. Eliminating all additional handling operations requires a significant increase in travel distance.

6 Conclusions

In this paper we have introduced the pickup and delivery traveling salesman problem with handling costs. We have defined the problem and showed that it is a generalization of the

Figure 2: Illustrative examples showing the trade-off between the distance costs and the number of handling operations.



pickup and delivery problem and the pickup and delivery problem with LIFO loading. A large neighborhood search heuristic is proposed for the problem, for which existing and new removal operators are introduced. The heuristic is evaluated on benchmark instances for the PDTSP and the PDTSPL, which are special cases of the problem. The results obtained with the LNS heuristic are often better than the results obtained in literature having improved 52 best known solutions for the PDTSP and 14 best known solutions for the PDTSPL. Moreover, we have shown that for small instances of the PDTSPH, the heuristic yields optimal or near-optimal solutions for all instances. For the instances we have studied, incorporating the penalty costs for the additional handling operations leads to routes that have a large reduction in the number of additional handling operations while increasing the travel distance by only a small percentage compared to the routes obtained by only taking into account the travel distance.

References

[1] Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers & Operations Research*, 41:167–173, 2014. doi: 10.1016/j.cor.2013.08.016.

[2] Maria Battarra, Guneş Erdoğan, Gilbert Laporte, and Daniele Vigo. The traveling salesman problem with pickups, deliveries, and handling costs. *Transportation Science*, 44(3):383–399, 2010.

[3] Francesco Carrabs, Raffaele Cerulli, and Jean-François Cordeau. An additive branch-and-bound algorithm for the pickup and delivery traveling salesman problem with LIFO or FIFO loading. *INFOR: Information Systems and Operational Research*, 45 (4):223–238, 2007.

[4] Francesco Carrabs, Jean-François Cordeau, and Gilbert Laporte. Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. *INFORMS Journal on Computing*, 19(4):618–632, 2007.

[5] L Cassani and G Righini. Heuristic algorithms for the TSP with rear-loading. In *35th Annual Conference of the Italian Operations Research Society (AIRO XXXV), Lecce, Italy*, 2004.

[6] Jean-François Cordeau, Manuel Iori, Gilbert Laporte, and Juan José Salazar González. A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. *Networks*, 55(1):46–59, 2010.

[7] Jean-François Côté, Michel Gendreau, and Jean-Yves Potvin. Large neighborhood search for the pickup and delivery traveling salesman problem with multiple stacks. *Networks*, 60(1):19–30, 2012.

[8] Irina Dumitrescu, Stefan Ropke, Jean-François Cordeau, and Gilbert Laporte. The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm. *Mathematical Programming*, 121(2):269–305, 2010.

[9] Güneş Erdoğan, Jean-François Cordeau, and Gilbert Laporte. The pickup and delivery traveling salesman problem with first-in-first-out loading. *Computers & Operations Research*, 36(6):1800–1808, 2009.

[10] Güneş Erdoğan, Maria Battarra, Gilbert Laporte, and Daniele Vigo. Metaheuristics for the traveling salesman problem with pickups, deliveries and handling costs. *Computers & Operations Research*, 39(5):1074–1086, 2012.

[11] Patrick Healy and Robert Moll. A new extension of local search applied to the dial-a-ride problem. *European Journal of Operational Research*, 83(1):83–104, 1995.

[12] Scott Kirkpatrick, C D Gelatt Jr., and M P Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[13] Yongquan Li, Andrew Lim, Wee-Chong Oon, Hu Qin, and Dejian Tu. The tree representation for the pickup and delivery traveling salesman problem with LIFO loading. *European Journal of Operational Research*, 212(3):482 – 496, 2011.

[14] Renaud Masson, Fabien Lehuédé, and Olivier Péton. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3):344–355, 2013.

[15] Gerhard Reinelt. TSPLIB – A traveling salesman problem library, 1991.

[16] Jacques Renaud, Fayez F. Boctor, and Jamal Ouenniche. A heuristic for the pickup and delivery traveling salesman problem. *Computers & Operations Research*, 27(9): 905–916, 2000.

[17] Jacques Renaud, Fayez F. Boctor, and Gilbert Laporte. Perturbation heuristics for the pickup and delivery traveling salesman problem. *Computers & Operations Research*, 29(9):1129–1141, 2002.

[18] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic

for the pickup and delivery problem with time windows. *Transportation Science*, 40 (4):455–472, 2006.

[19] M W P Savelsbergh. An efficient implementation of local search algorithms for constrained routing problems. *European Journal of Operational Research*, 47(1):75–85, 1990.

[20] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming – CP98*, pages 417–431. Springer, 1998.

[21] Lijun Wei, Hu Qin, Wenbin Zhu, and Long Wan. A study of perturbation operators for the pickup and delivery traveling salesman problem with LIFO or FIFO loading. *Journal of Heuristics,*, forthcoming, 2015.