Heuristics for the Traveling Repairman Problem with Profits

Thijs Dewilde[1], Dirk Cattrysse[1], Sofie Coene[2], Frits C.R. Spieksma[2], and Pieter Vansteenwegen[1]

1 **Centre for Industrial Management/Traffic & Infrastructure (CIB)**
Katholieke Universiteit Leuven, Belgium
Thijs.Dewilde@cib.kuleuven.be

2 **Research group Operations Research and Business Statistics (ORSTAT)**
Katholieke Universiteit Leuven, Belgium
Sofie.Coene@econ.kuleuven.be

Abstract

In the traveling repairman problem with profits, a repairman (also known as the server) visits a subset of nodes in order to collect time-dependent profits. The objective consists of maximizing the total collected revenue. We restrict our study to the case of a single server with nodes located in the Euclidean plane. We investigate properties of this problem, and we derive a mathematical model assuming that the number of visited nodes is known in advance. We describe a tabu search algorithm with multiple neighborhoods, and we test its performance by running it on instances based on TSPLIB. We conclude that the tabu search algorithm finds good-quality solutions fast, even for large instances.

1998 ACM Subject Classification I.2.8 Heuristic Methods

Keywords and phrases Traveling Repairman, Profits, Latency, Tabu Search, Relief Efforts

Digital Object Identifier 10.4230/OASIcs.ATMOS.2010.34

1 Introduction

Imagine a single server, traveling at unit speed. There are n locations given, each with a profit $p_i, 1 \leq i \leq n$. At $t = 0$, the server starts traveling and collects revenue $p_i - t_i$ at each visited location, where t_i denotes the server's arrival time at location i. Not all locations need to be visited. The problem is to find a travel plan for the server that maximizes total revenue. This problem is known as the traveling repairman problem with profits (TRPP) and forms the subject of this paper. In particular, we perform a computational study of the TRPP in the Euclidean plane.

Motivation

The TRPP occurs as a routing problem in relief efforts. For example, consider the following situation. In the aftermath of a disaster like an earthquake, there are a number of villages that experience an urgent need for medicine. The sooner the medicine gets to a village, the more people can be rescued. Since the cost of transport is negligible compared to the value of a human life, rescue teams are only concerned with the total number of people that can be saved. Assume that at location i there are p_i people in need of the medicine, and that every instance of time, there is one of them dying. Suppose also that we have one truck available. With t_i denoting the arrival time of the truck at location i, the number of people that will survive equals $p_i - t_i$. Thus, the goal of the rescue team is to maximize $\sum_i (p_i - t_i)$,


© Thijs Dewilde, Dirk Cattrysse, Sofie Coene, Frits C.R. Spieksma and Pieter Vansteenwegen;
licensed under Creative Commons License NC-ND
10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10).
Editors: Thomas Erlebach, Marco Lübbecke; pp. 34–44

OpenAccess Series in Informatics
Schloss Dagstuhl Publishing, Germany

where the sum runs over all the visited locations. This situation is described in [6] and is equivalent to the TRPP.
Another, more theoretical, motivation concerns the k-traveling repairman problem (k-TRP). The k-TRP is the problem with multiple servers that need to visit all clients such that the latency, i.e., the average arrival time, is minimized. Observe that no profits are considered in this problem. One potential way of solving such a problem is a set-partitioning approach where an integer programming model is built, using a variable for each set of clients [7]. Next, a branch-and-price approach can be applied to the resulting integer program. Without going into further details, we observe here that the so-called pricing problem in such a branch-and-price approach is exactly the TRPP, where the dual variables play the role of profits.
Other applications of routing problems with time-dependent revenues are described in [7, 9, 13] and [14], which deals with a problem occurring in "multi-robot routing".

Literature
Several problems are closely related to the traveling repairman problem with profits (TRPP). The TRPP has similarities with the traveling salesman problem (TSP) [3]. However, contrary to the TSP, in the TRPP not all the nodes need to be visited. Further, an optimal TRPP-solution is a path which course is influenced by the depot location and may contain intersections. Notice that the latter is always sub-optimal for the Euclidean TSP.
Also the TSP with profits (TSPP) [10] and the orienteering problem (OP) [20], have some similarities with the TRPP. In the OP a subset of nodes should be selected in order to maximize the profit under a time-constraint. As for the TRPP, a solution for the TSPP may leave some nodes unvisited. Both the total profit and the distance traveled are inserted in the objective function of both problems; only in the TRPP, however, the revenues are time-dependent.
Problems with time-dependent revenues are relevant in many cases. See [13] for the time-dependent traveling salesman problem (TDTSP). In TDTSP, the travel time between two vertices depends on the arrival time of the server. In the objective function of the TDTSP only the used travel time is included. This is different from the TRPP where the travel time between vertices is constant. A related problem that uses latency in the objective function is the traveling repairman problem (TRP) [5], also known as the minimum latency problem or the delivery man problem. Here, a single server needs to visit all nodes such that total latency is minimized. In a classical paper [2], it is shown that the TRP on the line can be solved in polynomial time by dynamic programming. This result was generalized to the TRPP on the line by [7]. Since the TRP is NP-hard for more general metric spaces (see the argument given in [5]), and since the TRPP is a generalization of the TRP, we conclude that the TRPP for these general metric spaces, among which the Euclidian plane, is NP-hard.
As far as we know, no computational studies have been performed for the TRPP. So far, the TRPP is only tackled in one paper. In [7] the TRPP on the line is being solved in polynomial time by a dynamic programming algorithm. No other results are known.
Exact algorithms and approximation algorithms for the TRP have been described in [4, 12, 18, 21]; metaheuristics for the TRP are described in recent contributions [15] and [17]. As far as we are aware, these are the only studies that present metaheuristics for the TRP. For a review of the metaheuristics for other related problems we refer to [10, 20] and the references contained therein. A general description of some metaheuristics, including the ones that are used in this paper is given in [11, 19].

This paper is structured as follows. In the next section, the TRPP is described in detail, and a mathematical model is given. A tabu search algorithm is presented in Section 3. The data sets are introduced in Section 4 and the computational results are discussed in Section 5. The conclusions of this paper are summarized in Section 6.

2 Mathematical model

Given is a complete undirected graph $G = (V, E)$, where $V = \{0, 1, \ldots, n\}$ is the node set, and E is the set of edges. Each node $i \neq 0$ has an associated profit p_i. There is a single server located at node 0, the depot. The time it takes the server to travel from node i to node j is defined by $d_{i,j}$. We assume that the time to serve a node is negligible. If the server arrives at node i at time t_i, a revenue of $p_i - t_i$ is collected. As a consequence, an optimal tour will not contain a node i with $p_i \leq t_i$. The goal of the TRPP is to select an ordered subset of nodes such that visiting them one by one maximizes the sum of all the revenues. This should be achieved under the conditions that each node can only be visited once, and that at the end, the server does not need to return to the depot.
We now derive a mathematical model for this problem in which the number of visited nodes is assumed to be given. Define k as this number, i.e., k is the number of nodes whose revenue is collected. For the ease of notation, we write the set of integers $\{1, 2, \ldots, k\}$ as K.

For each $i \in V, j \in V_0 = V \setminus \{0\}$, and $\ell \in K$, we define the variable y as follows,

$$y_{i,j,\ell} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is used as } \ell^{\text{th}} \text{ edge,} \\ 0 & \text{else.} \end{cases}$$

This definition says that if $y_{i,j,\ell} = 1$ then (i, j) is the ℓ^{th} edge of the path. Hence i is the $(\ell - 1)^{\text{th}}$ and j is the ℓ^{th} node that is visited. The depot is node 0 of the solution. Observe that if $y_{i,j,\ell} = 1$, $d_{i,j}$ is counted $k + 1 - \ell$ times in the total latency. Hence

$$\sum_{i:\text{visited}} t_i = \sum_{\{(i,j,\ell)\,|\,y_{i,j,\ell}=1\}} (k + 1 - \ell)\, d_{i,j}.$$

Now the mathematical model can be constructed.

Given the number of visited nodes, k, the mathematical model is the following

$$\max \sum_{i \in V} \sum_{j \in V_0} \sum_{\ell \in K} (p_j - (k + 1 - \ell)\, d_{i,j})\, y_{i,j,\ell} \tag{1}$$

subject to

$$\sum_{i \in V} \sum_{\ell \in K} y_{i,j,\ell} \leq 1 \qquad \forall j \in V_0, \tag{2}$$

$$\sum_{i \in V} \sum_{j \in V_0} y_{i,j,\ell} = 1 \qquad \forall \ell \in K, \tag{3}$$

$$\sum_{i \in V} y_{i,j,\ell} - \sum_{i \in V_0} y_{j,i,\ell+1} = 0 \qquad \forall j \in V_0, \forall \ell \in K \setminus \{k\}, \tag{4}$$

$$\sum_{j \in V_0} y_{0,j,1} = 1, \tag{5}$$

$$y_{i,j,\ell} \in \{0, 1\} \qquad \forall i \in V, \forall j \in V_0, \forall \ell \in K. \tag{6}$$

The objective function (1) sums the difference between the profit of a node and the number of times the edge preceding that node is counted in the total latency. The first set of restrictions makes sure that each node can only be visited once (2). The second set dictates that k nodes different from the depot must be visited (3); for each $\ell = 1, 2, \ldots, k$ the server has to travel from a node $i \in V$ to a node $j \in V_0$. Constraints (4) ensure the connectivity, and the departure from the depot is arranged by (5). Finally, all $y_{i,j,\ell}$ must be binary (6). This model is used in Section 5 for obtaining the optimal solution or the LP-relaxation of the considered instances using CPLEX.

Notice that in this model we assume that the value of k and hence the set of integers K is given. However, in the TRPP, k is a decision variable and should be determined by the model itself. It is not difficult to introduce k as a variable in the model, see [8]. However, preliminary results in [8] showed that this leads to a much weaker LP-relaxation and hence to a much worse computational performance compared to solving the LP-relaxation of (1)-(6). On the other hand, it will be shown next that it is not easy to determine the optimal value of k apart from solving the above model for each value of $k \leq n$.

Before doing so, let us first introduce some notation. Define k^* as the optimal number of visited nodes and $f^* = f(k^*)$ as the global optimal objective value. Define $f(k)$ as the optimal objective value for which the solution visits exactly k nodes, hence $f^* = f(k^*)$.
As mentioned above, we will now show that the mathematical model needs to be solved for each value of $k \leq n$ in order to find k^* and hence the global optimum. Therefore we will demonstrate that (1) $f(k)$ in function of k is not unimodal and (2) an increase in the number of nodes may result in a decreasing value for k^*. Let us first go into (1). It holds that when the server is forced to visit one node extra than the k^* nodes which lead to f^*, this results in an inferior solution. Intuitively one may think that the further k lies from the optimal number of visited nodes, k^*, the worse the objective value will be. In other words,

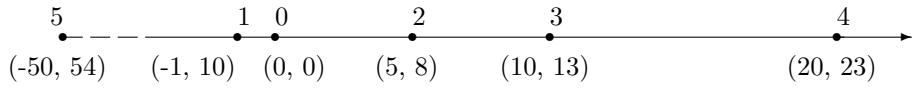

Figure 1 Network with 6 collinear points

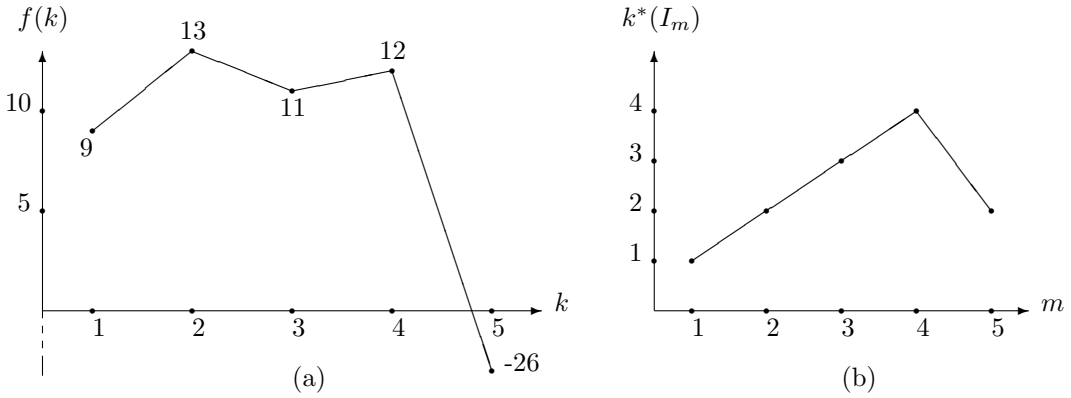

Figure 2 Solution results for the network with 6 collinear points

our intuition may tell us that for any $k \geq k^*$ we have that $f(k^*) \geq f(k) \geq f(k+1)$, and analogue for any $k \leq k^*$. However, this is not always true. To show this, consider the network with 6 collinear points given in Figure 1. The numbers between brackets are respectively the location along the axis and the profit. So the leftmost node, node 5, has as coordinate -50 and its profit equals 54.
When we solve this instance to optimality for $k = 1, \ldots, 5$, i.e., when we force the solution to visit exactly k nodes, we find the results depicted in Figure 2(a). It can be seen that when solving the mathematical model for $k = 2$, the resulting path is $\langle 0, 1, 5 \rangle$ with total revenue 13, for $k = 3$ the optimal path has revenue $f(k) = 11$, whereas forcing k to be 4, the solution is $\langle 0, 1, 2, 3, 4 \rangle$ with objective value 12. You can see that $k^* = 2$. More importantly, the non-unimodality of this graph shows that $f(k)$ can have multiple local optima, which suggests that, in order to find k^* for a particular instance, model (1)-(6) has to be solved for each $k = 1, \ldots, n$.
The second property (2) that can be conducted from this example deals with adding a node to an instance. If an extra node is added to a data set, our intuition may tell us that the optimal number of visited nodes will be the same or larger than before adding that node. However, this is not always true. Clearly, by adding a new node to an instance, the optimal value cannot decrease. But nothing can be said about the optimal number of visited nodes of this new instance as witnessed by the given example. Define $I_m : m \leq n$, as the instance consisting of the first m nodes of the network. The number of nodes in the optimal solution for instance I_m is $k^*(I_m)$. The results for the value of $k^*(I_m)$ for the network of Figure 1 are given in Figure 2(b). This example indicates that knowing $k^*(I_m)$ for a certain value of m does not give any information about $k^*(I_{m'})$ with $m' > m$. Again, we can only conclude that, to find k^*, the model (1)-(6) has to be solved for each $k = 1, \ldots, n$. Notice that the observations above already hold in the case of a line metric.

3 Metaheuristic methods

In this section a metaheuristic for the TRPP is presented. First, we discuss a way to build a non-trivial solution which will then be systematically improved by a tabu search algorithm. We define the *trivial solution* as the path $\langle 0 \rangle$, i.e., the situation in which the server does not leave the depot. By starting from the trivial solution and adding a node in each step we can obtain a new solution. This process is called the construction phase and is the subject of the next section. In Section 3.2 some local search methods are discussed. These methods are integrated in the second step of the solution procedure, a tabu search metaheuristic.

3.1 Construction phase

Consider a partial path P, and define the set $\bar{V}$ as the set of all non-visited nodes, $\bar{V} \subseteq V_0$. In order to improve the partial path P, a node from $\bar{V}$ should be added. This process requires two decisions: which node to insert and where to place it in the path. Naturally two factors influence these choices: the profit of the nodes and the extra latency incurred by inserting that node.
We use the following ratio to determine which node to add to our partial path. Let $d_{i,j}$ and p_j be as before. Then, for each $i \in V \setminus \bar{V}$ and $j \in \bar{V}$ we define $ratio^m_{i,j}$ as follows:

$$ratio^m_{i,j} = \begin{cases} \frac{1}{d_{i,j}} & \text{if } m = 0, \\ p_j \cdot \left(\frac{1}{d_{i,j}}\right)^m & \text{if } m = 1, \ldots, 10. \end{cases} \tag{7}$$

In this way, the parameter m determines the impact of $d_{i,j}$ on the ratio.
The construction method that is used in this paper is based on insertion. In each step the node $j^* = \arg\max_{j \in \bar{V}} ratio^m_{i,j}$, for an i and m, is selected to insert. The place of insertion is then determined based on the improvement in score by adding this node. For a more detailed description and a pseudo-code, we refer to [8].
In preliminary tests, the insertion based method is compared with other construction methods such as a greedy method, and the use of (7) to select nodes is evaluated [8]. Regarding objective function value and computation time, the insertion method using (7) turned out to perform the best on average.

3.2 Improvement phase

This section describes a tabu search metaheuristic for the TRPP. The insertion based algorithm from the previous section is used as input. A tabu search metaheuristic starts from a given solution. By searching neighboring solutions, it tries to improve the current solution. Our algorithm works with multiple neighborhoods. We next define the moves and corresponding neighborhoods. Then, the tabu search procedure is explained in section 3.2.2.

3.2.1 Neighborhoods

The objective of the TRPP is to maximize total collected revenue, which is based on profits that decrease over time. Hence, improving a solution can be done by altering the collection of visited nodes, or by decreasing the total latency by changing the visiting sequence. The moves that alter the subset of selected nodes are straightforward: deletion, insertion, and replacement. The other set of moves consists of seven moves, among which the well-known swap(-adjacent), 2-opt, and or-opt [1]. The choice for or-opt is justified by the fact that the visiting order of the nodes is not reversed, while the change is large enough to circumvent local optima where other moves might end up. The last three moves are explained next.
Move-up (down) consists of shifting a node up (down) the path. A special type of a move-up is the remove-insert. In this move the node with the largest between-distance of a given node is removed and back inserted at the end of the path.
Although swap-adjacent and remove-insert are special cases of swap and move-up, respectively, they are used separately. This is because they have linear complexity, while move-up (down) and swap have a neighborhood of size $\mathcal{O}(n^2)$. Hence separating these moves can speed up the algorithm. Note that the same can be said about move-up (down) and or-opt which has a neighborhood of cubic size.
The hierarchy in which these ten moves are used is shown in Figure 3. First the neighborhoods that alter the sequence of the path are considered, then those that alter the set of nodes, and finally or-opt is used to perturb the solution to escape from a local optimum. The choice for this sequence is justified by the fact that altering the set of selected nodes without re-optimizing the sequence is useless.

In each iteration of the tabu search algorithm (see below), a move will be selected according the principles of a variable neighborhood descend heuristic (VND) [11, 19]. This means that the neighborhoods will be searched through one by one, in the sequence of Figure 3. Whenever an improving move is detected, the best solution from the corresponding neighborhood is chosen as next solution. In the case that there is no better solution in a neighborhood, the next move will be investigated.

3.2.2 Tabu search

The metaheuristic used to improve the construction phase solution is tabu search (TS). The basic idea of tabu search is to avoid repetition of solutions and to use steepest ascend combined with mildest descend to escape from local optima. Next to the standard extensions as aspiration and intensification followed by a diversification phase, see [11, 19] for more details, some specific features are added. For example, the use of multiple neighborhoods requires several tabu lists, and restricted candidate lists are used for the largest neighborhoods.

In the remaining of this section the main components of the tabu search algorithm are explained. For a more detailed description, we refer to [8].
First of all, as explained at the end of the previous section, our tabu search uses the principles of variable neighborhood descend (VND). The neighborhoods are ordered as in Figure 3. In order to speed up the algorithm some restrictions are used to limit the size of the neighborhoods. For move-up (down), swap, 2-opt, and or-opt the maximum number of visited nodes in the path between the move-determining attributes is limited to $n/2$, with n the number of available nodes. If no improving solution is found in any of the restricted neighborhoods, the best possible neighbor over all the neighborhoods is chosen as next solution.
When a number of local optima have been reached, the intensification phase starts. It begins with updating the attribute matrix M. $[M]_{i,j}$ is the number of times that the edge (i, j) was part of the current local optimum. Next, the current solution is used as start solution for a full neighborhood VND without tabu moves.
After the intensification phase, the diversification phase starts. First, the tabu lists are cleared. Then the attribute matrix is used to penalize frequently used attributes; by subtracting from the score, a given penalty times $[M]_{i,j}$ for each edge (i, j) in the intensification phase solution, we favor non-used attributes. In order to find a new and diverse solution, we re-initialize the algorithm, including these penalties for 100 iterations. During this process, the tabu lists are built up again to prevent a quick return towards the previous solutions. The path that is returned from the diversification phase is used as input for the main part of the tabu search algorithm.
After some diversification phases, the current solution may lay in an area of the solution space far away from the first solutions. By allowing that the penalties become bonuses, frequently used attributes will be favored. Hence intermediate solutions or new promising solutions will be used as current path. This enforces the search since it results in paths that are combinations of very promising solutions. Without this extra feature, the penalties prevent this, and very good solutions can be missed.
The next component of the tabu search is the use and the composition of the tabu lists. Due to the use of different neighborhood structures, more than one tabu list is required. In general, each move is added to exactly one list, but moves of more than one type, for

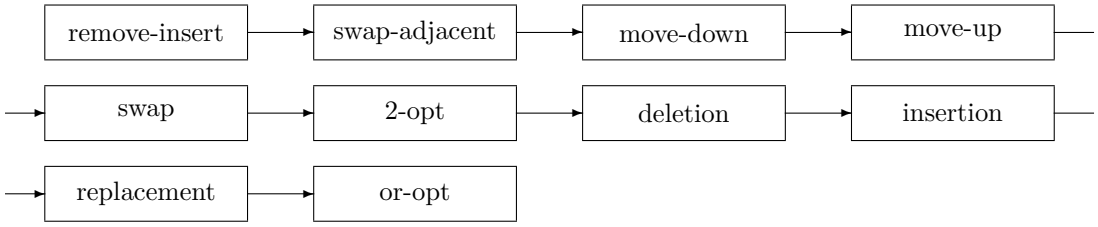

Figure 3 Sequence of the moves

example insertion and deletion, can belong to the same tabu list.
After each move only the corresponding tabu list is updated, since otherwise after a deletion and a short re-optimizing, the insertion of that node might already be allowed. By forbidding this, repetition in the long run is prevented.
The last aspect of the tabu search algorithm for the TRPP to discuss is the stop criterium. A balance must be made between computation time and efficiency. The number of consecutive non-improving steps and a maximum computation time determine the stop criterium.

3.3 Upper bound

Since the TRPP in the Euclidian plane is NP-hard, see Section 1, the mathematical model can only be solved for small instances. However, to assess the quality of the solutions found by tabu search, an upper bound is required. We informally sketch here a simple bound. Assume that the number of visited nodes, k, is known. In order to get a lower bound for the latency in case k nodes are visited, we need the k-minimal spanning tree (k-MST). Since solving a k-MST is again an NP-hard problem, we use the minimal k-forest to approximate this. The minimal k-forest of a graph is the subgraph containing the $k-1$ shortest edges that do not form a circuit. Each edge of the k-forest is assigned a multiplicity. The longest edge gets 1, the second longest 2, ..., until the shortest edge gets k. The sum of the distances weighted with the corresponding multiplicities is then a lower bound for an optimal solution to the k-MST.
Next, by summing the k largest profits, we get an upper bound for the collected profits. The difference of this upper bound and the lower bound for the k-MST gives an upper bound for the TRPP, under the assumption that k is known. Taking the maximum over $k = 1, \ldots, n$, leads us to an upper bound for the TRPP.

4 Instances

Two types of data sets are used. The first type is based on data sets obtained from TSPLIB [16]. To obtain a data set with exactly n nodes and a depot, we selected the first $n+1$ nodes of an instance containing enough nodes. The first node is chosen as depot and gets a profit of 0. The remaining ones are allocated a profit that is randomly selected according the uniform distribution in the interval $[L, U]$ with $L < U$. The values of L and U are chosen in such a way that in the construction phase solution an acceptable amount of nodes is visited, i.e., $k \geq 0.60 \cdot n$. This is done to obtain interesting data sets.
The data sets of the second type are randomly generated according the uniform distribution. The nodes are spread out over the Euclidean plane and have integer valued coordinates. As for the data sets of the first type, the profits are randomly generated and satisfy the following inequality: $d_{0,i} \leq L < p_i < U$ for each node $i \neq 0$.
For each data set, the number of nodes different from the depot (n) is 10, 20, 50, 75, 100, 150, 200, or 500, and for each value of n there are 10 random instances and 5 instances from TSPLIB. The data sets are available on the following website:
http://www.mech.kuleuven.be/en/cib/trpp.

5 Results

In this section we will discuss the computational results. First, a comparison between the exact results, the tabu search results and the upper bound from Section 3.3 is given for small datasets. Second, the latter two are used to measure the performance of tabu search on

larger instances.

The first results are presented in Table 1. The first column shows the computation time for the exact solutions, found by solving the mathematical model from Section 2 using Cplex. The other columns present the average gap with the exact solution and the required computation time of the LP-relaxation, the construction phase, the tabu search algorithm, and the upper bound, respectively. The gap is computed as follows:

$$\text{gap}_{\text{X}}(\%) = \left| \frac{\text{X} - (\text{exact solution})}{(\text{exact solution})} \right|.$$

In the case of $n = 50$, we were not able to compute exact results since solving the mathematical model for some $k \leq n$ requested too much computation time. In this case the gap is computed with respect to the LP-relaxation. When n gets larger, Cplex is not able to find a solution anymore due to memory restrictions.
From this table it is clear that TS was able to find the optimal solution for all instances when $n = 10$ or 20. When $n = 50$, we see that on average, the gap with the LP-relaxation is 13.99%. Next, we can see that TS needs much less time than Cplex[1]. Finally, the upper bound gives worse results than the LP-relaxation, but it needs much less time.

In Table 2, the improvement that tabu search makes compared to the solution of the construction phase is presented. This is then compared with the upper bound. In the first column the computation time for the construction phase is given. Columns 2 and 3 contain, respectively, the improvement of tabu search compared to the construction phase and the computation time of the tabu search algorithm. The results for the upper bound are summarized in the last two columns. First the average gap between the tabu search solution and the upper bound is given, and second, the time, in seconds, needed for computing the upper bound is presented. We define improv and gap as

$$\begin{aligned} \text{improv}(\%) &= \frac{(\text{TS-solution}) - (\text{construction solution})}{(\text{construction solution})}, \\ \text{gap}(\%) &= \frac{(\text{upper bound}) - (\text{TS-solution})}{\text{upper bound}}. \end{aligned}$$

We see that tabu search improves the solution from the construction phase considerable. Also, the gap with the upper bound is only slowly increasing.

6 Conclusion

We have studied the traveling repairman problem with profits (TRPP). In this problem a server has to visit a subset of nodes in order to maximize the total collected revenues which are declining in time. After motivating this problem and reviewing the related literature, we develop a mathematical model in which we make the assumption that the number of visited nodes is known in advance. Using an example, we find that it is not straightforward to determine this number optimally. As our main contribution, we propose a tabu search algorithm with multiple neighborhoods. We have implemented this method, and we tested

[1] The tabu search algorithm is programmed in C++, the exact solutions are found with Cplex 10.1. Both were run on a DELL Optiplex 760, Intel(R) Core(TM) 2 Duo 3.00GHz, 4.00GB RAM, 64-bit Operating System.

	model (Cplex)	LP-relaxation		construction phase		tabu search		upper bound	
	time	gap	time	gap	time	gap	time	gap	time
n	(s)	(%)	(s)	(%)	(s)	(%)	(s)	(%)	(s)
10	1	4.91	0	1.26	0	0	2	21.91	0
20	89	6.42	1	2.1	0	0	2	17.15	0
50			154	16.7	0	13.99	14	7.18	0

Table 1 Comparison of the results of the mathematical model (1)-(6)

	construction	tabu search		upper bound	
	time	improv	time	gap	time
n	(s)	(%)	(s)	(%)	(s)
10	0	1.31	2	16.05	0
20	0	2.21	2	14.20	0
50	0	3.29	14	19.39	0
75	0	4.71	45	19.02	0
100	0	4.33	208	14.09	0
150	0	7.83	545	18.89	0
200	0	6.59	580	20.44	0
500	2	16.02	500	24.81	0

Table 2 Comparison of the results of the metaheuristic

the performance of this metaheuristic, comparing it to a quite crude upper bound. The computational results show that the tabu search algorithm is able to find optimal solutions for small instances in a reasonable amount of time. For larger instances the optimal solution is not known, but the metaheuristic obtains a considerable improvement compared to the initial solution; even up to an average of 16% compared to the insertion-based construction phase solution.

Acknowledgements Dr. P. Vansteenwegen is a post-doctoral research fellow of the "Fonds Wetenschappelijk Onderzoek - Vlaanderen (FWO)".

References

1 E. Aarts, J.K. Lenstra (eds), Local search in combinatorial optimization, *Wiley-interscience series in discrete mathematics and optimization, 1997.*

2 F. Afrati, S. Cosmadakis, C.H. Papadimitriou, G. Papageorgiou, N. Papakostantinou, The complexity of the travelling repairman problem, *Informatique Théorique et Applications 20 (1986) pp. 79-87.*

3 D.L. Applegate, R.E. Bixby, V. Chvátal, W.J. Cook, The traveling salesman problem: a computational study, *Princeton University Press, 2006.*

4 G. Ausiello, V. Bonifaci, S. Leonardi, A. Marchetti-Spaccamela, Prize-collecting traveling salesman and related problems, *in: T.F. Gonzalez (eds), Handbook of Approximation Algorithms and Metaheuristics, CRC Press (2007) pp. 40.1-40.13.*

5 A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, M. Sudan, The minimum latency problem, *in: Proceedings of the twenty-sixth annual ACM symposium on the theory of computing (1994) pp. 163-171.*

6 A.M. Campbell, D. Vandenbussche, W. Hermann, Routing for relief efforts, *Transportation Science 42 (1994) pp. 127-145.*

7 S. Coene, F.C.R. Spieksma, Profit-based latency problems on the line, *Operations Research Letters 36 (2008) pp. 333-337.*

8 T. Dewilde, Het profit-based latency probleem, *Master's thesis, Katholieke Universiteit Leuven, 2009 (in Dutch).*

9 E. Erkut, J. Zhang, The maximum collection problem with time-dependent rewards, *Naval Research Logistics 43 (1996) pp. 749-763.*

10 D. Feillet, P. Dejax, M. Gendreau, Traveling salesman problems with profits, *Transportation Science 39 (2005) pp. 188-205.*

11 F. Glover, G.A. Kochenberger (eds), Handbook of metaheuristics, *Kluwer Academic Publishers (2003).*

12 M. Goemans, J. Kleinberg, An improved approximation ratio for the minimum latency problem, *Mathematical Programming 82 (1998) pp. 111-124.*

13 A. Lucena, Time-dependent traveling salesman problem - The deliveryman case, *Networks 20 (1990) pp. 753-763.*

14 J. Melvin, P. Keskinocak, S. Koenig, C. Tovey and B.Y. Ozkaya, Multi-robot routing with rewards and disjoint time windows, *in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (2007) pp. 2332-2337.*

15 S.U. Ngueveu, C. Prins, R. Wolfler-Calvo, An effective memetic algorithm for the cumulative capacitated vehicle routing problem, *Computers & Operations Research 37 (2010) pp. 1877-1885.*

16 G. Reinelt, TSPLIB, *Institut für angewandte Mathematik, Universität Heidelberg (2001).* http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

17 A. Salehipour, K. Sörensen, P. Goos, O. Bräysy, An efficient GRASP+VND metaheuristic for the traveling repairman problem, *Working paper, University of Antwerp, Faculty of Applied Economics (2008).*

18 J.F.M Sarubbi, H.P.L. Luna, A new flow formulation for the minimum latency problem, *in: International Network Optimization Conference, Spa (2007).*

19 E.G. Talbi, MetaHeuristics: From design to implementation, *Wiley, 2009.*

20 P. Vansteenwegen, W. Souffriau, D. Van oudheusden, The orienteering problem: A survey, *European Journal of Operational Research (in press) doi:10.1016/j.ejor.2010.03.045.*

21 B.Y. Wu, Z.-N. Huang, F.-J. Zhan, Exact algorithms for the minimum latency problem, *Information Processing Letters 92 (2004) pp. 303-309.*