DEPARTMENT OF ENVIRONMENT,
TECHNOLOGY AND TECHNOLOGY MANAGEMENT

A simple GRASP+VND for the travelling salesperson problem with hotel selection

Marco Castro, Kenneth Sörensen, Pieter Vansteenwegen & Peter Goos

UNIVERSITY OF ANTWERP
Faculty of Applied Economics

Stadscampus
Prinsstraat 13, B.226
BE-2000 Antwerpen
Tel. +32 (0)3 265 40 32
Fax +32 (0)3 265 47 99
http://www.ua.ac.be/tew

FACULTY OF APPLIED ECONOMICS

DEPARTMENT OF ENVIRONMENT,
TECHNOLOGY AND TECHNOLOGY MANAGEMENT

A simple GRASP+VND for the travelling salesperson problem with hotel selection

Marco Castro, Kenneth Sörensen, Pieter Vansteenwegen & Peter Goos

RESEARCH PAPER 2012-024
OCTOBER 2012

University of Antwerp, City Campus, Prinsstraat 13, B-2000 Antwerp, Belgium
Research Administration – room B.226
phone: (32) 3 265 40 32
fax: (32) 3 265 47 99
e-mail: joeri.nys@ua.ac.be

The papers can be also found at our website:
www.ua.ac.be/tew (research > working papers) &
www.repec.org/ (Research papers in economics - REPEC)

D/2012/1169/024

A simple GRASP+VND for the travelling salesperson problem with hotel selection

M. Castro*[1], K. Sörensen[1], P. Vansteenwegen[2,3], and P. Goos[1,4]

[1]University of Antwerp, Belgium
[2]Ghent University, Belgium
[3]University of Leuven, Belgium
[4]Erasmus University Rotterdam, The Netherlands

October 23, 2012

In this paper, we present a new metaheuristic solution procedure for the travelling salesperson problem with hotel selection (TSPHS). We develop a simple but powerful metaheuristic for the TSPHS. On the existing benchmark instances for which an optimal solution is known, it obtains 27 out of 28 known optimal solutions. For the other instances, it obtains several new best known solutions. This heuristic clearly outperforms the only existing heuristic in the literature.

1 Introduction

In the *travelling salesperson problem* (TSP) (Applegate et al., 2007), a salesperson departs from home, visits a set of customers, and go back home at the end of the day. However, sometimes the workload cannot be completed in a single day of work. Moreover, it is possible that, in practical terms, it is better to stay at an intermediate hotel from which, the salesperson must continue with her/his journey. Therefore, a decision must be made at the end of each day: whether to go home, or stay at an intermediate hotel. This problem is tackled in the *travelling salesperson problem with hotel selection* (TSPHS) (Vansteenwegen et al., 2011).

In TSPHS lexicon, a "trip" corresponds to a single day of work, i.e., a sequence of visits to customers starting and ending at a hotel, while a "tour" is a set of connected trips that, together, visits all customers.

More formally, the TSPHS is characterised as follows: given a non-empty set H of hotels and a set C of customers, the TSPHS is defined on a complete undirected graph $G = (V, E)$ where $V = H \cup C$ and $E = \{\{i, j\} \mid i, j \in V; i < j\}$. Each customer $i \in C$ requires a service or visiting time t_i (with $t_i = 0, \forall i \in H$). The time c_{ij} needed to travel from location i to j is known for all pairs of locations. The goal is to first minimise the number of connected trips required to visit all customers, and then to minimise the total travel time of the tour. Every trip must (a) start and end in one of the available hotels, (b) the starting hotel of one trips must be the ending hotel of the previous trip, (c) not exceed a total travel length L. Moreover, the starting and ending hotel of the tour, i.e. the starting point of the first trip and the ending point of the last trip, have to be the same and given ($i = 0, i \in H$).

The remaining of this paper is organised as follows: in Section 2, the relevant literature is presented. In Section 3, a simple but powerful metaheuristic for the TSPHS is outlined, while in Section 4, a parametric analysis as well as the results are presented. Finally, in Section 5, our conclusions and research opportunities are drawn.

*Corresponding author: ✉ Universiteit Antwerpen, Stadscampus B.513, Prinsstraat 13, 2000 Antwerpen, Belgium, ☎ +32 3 265 40 61, 🖷 +32 3 265 49 01, 🖂 marco.castro@ua.ac.be

2 Literature review

Since the TSPHS is a generalisation of the classical TSP, it is also related to other node routing problems that arise in the literature. Among the related problems we can enumerate the following:

- the multiple travelling salesperson problem (mTSP) (Bektas, 2006)
- the vehicle routing problem (VRP) (Toth and Vigo, 2002)
- the multi-depot vehicle routing problem (MDVRP) (Cordeau et al., 1997; Polacek et al., 2004)

In these problems, a set of customers must be served by means of an available fleet of vehicles while minimising the travelled distance. In the mTSP and VRP, the fleet of vehicles is based at a central depot, while in the MDVRP multiple fleets are available at different depots.

Although similar, these problems differ from the TSPHS in two of its main features: (1) the routes have to start and end at the same depot, and (2) there are multiple vehicles available whereas in the TSPHS, there is only one salesperson.

The requirement of routes starting and ending at a common depot is relaxed in routing problems with *intermediate facilities* (IF). This class of problems allow a route to be split into trips which start and end at an IF. Some problems involving IFs can be found in the literature, some of which are:

- the periodic vehicle routing problem with intermediate facilities (PVRP-IF) (Angelelli and Grazia Speranza, 2002)
- the waste collection vehicle routing problem with time windows (WCVRPTW) (Kim et al., 2006; Benjamin and Beasley, 2010)
- the multiple-depot routing problem with inter-depot routes (MDVRPI) (Crevier et al., 2007)
- the capacitated arc routing problem with intermediate facilities (CARPIF) (Ghiani et al., 2001; Polacek et al., 2008)
- the arc routing problem with intermediate facilities under capacity and length restrictions (CLARPIF) Ghiani et al. (2004)

Similar to the TSPHS, these problems, which are variants of the VRP and CARP, also allow an IF to be visited on multiple occasions. However, there are several differences between these problems and the TSPHS, namely (1) a visit to an IF is not explicitly constrained by a length, like in the TSPHS, but for the vehicle capacity whereas in the TSPHS there is no capacity involved, and (2) in the TSPHS, the primary objective is the minimisation of the number of trips, rather than minimising the total travelled time.

Although in the MDVRPI and CLARPIF, there is an upper bound on the travel time, this bound is on the total travel time, and not on a single trip between two IFs like in the TSPHS. This makes the problems to be unrelated between them.

As noted above, the TSPHS is a generalisation of the TSP and, hence, it is also NP-hard. For this reason, we do not expect to solve large instances to optimality and, therefore, a metaheuristic solution procedure is developed in the next section.

3 A GRASP+VND metaheuristic for the TSPHS

In this section, a new metaheuristic for the TSPHS is introduced. This heuristic uses three different classes of operators:

(I) three well-known *intra-trip* operators for improving a sequence of customers with a starting and ending hotel: `2-opt` (Croes, 1958), `3-opt` (Lin, 1965), `Or-opt`, and (Or, 1976),

(II) two *inter-trip* operators for reallocating a set of customers between two trips, such as `Relocate` and `Exchange` (Laporte et al., 2000), and

(III) two *hotel selection* operators to either improve the choice of a hotel between two trips (`ChangeHotels`, see Figure 1) or to remove a hotel between two trips (`JoinTrips`, see Figure 2), which are explained in detail in the next paragraphs.

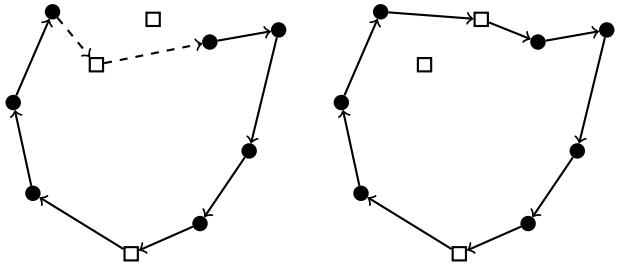


Figure 1: Example of a `ChangeHotels` move

Given a TSPHS solution, the aim of the `ChangeHotels` operator is to improve the choice of an intermediate hotel between two consecutive trips, while maintaining the feasibility of the solution. Every intermediate hotel in the solution is tested against the other available hotels. If one or more favourable hotel swaps are identified, the intermediate hotel is replaced with the hotel that leads to the largest improvement in total length. In Figure 1, the graph on the left represents a solution before applying the operator, while the graph on the right represents the solution after applying the operator. The white squares indicate the hotels, while the black circles indicate the customers. The hotel in the middle of the dotted arcs in the graph on the left, represents the hotel which is replaced.

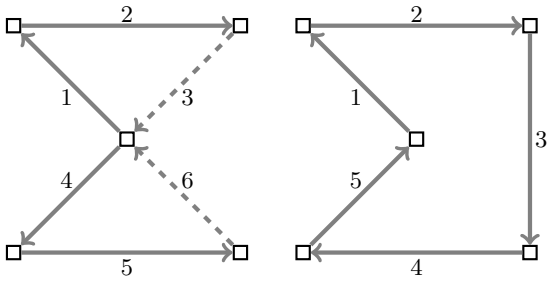


Figure 2: Example of a `JoinTrips` move

The `JoinTrips` operator attempts to decrease the number of trips by joining any pair of trips which have at least one hotel in common. The idea of this operator is to remove a common hotel and join the trips. The simplest situation where this operator is applicable is when two consecutive trips can be joined by removing the intermediate hotel. A more complicated case is when two non-consecutive trips are joined, as this may change the order of the trips, and it may also require changing the order in which the customers are visited in some of the trips, in order to maintain a feasible solution. If the length of the resulting trip does not exceed the time budget L, the move is marked as suitable. Since there may be more than one way to join a single pair of trips, the one which leads to a larger decrease (or smaller increase) in total length is preferred. Note that, whenever feasible, joining trips is always beneficial, even when the total length increases. This is due to the ordering of the TSPHS's two objectives.

In Figure 2, we see an example of a feasible `JoinTrips` move. The nodes represent the hotels and the arcs represent the sequence of customers visited in each trip. The labels associated with the arcs indicate the order in which the trips are visited. The dotted arcs in the graph on the left represent the trips which are going to be joined. In the example, the third and the sixth trip are joined. In order to build a feasible solution, the direction of the fourth and fifth trip has to be reversed, and the order of visiting the customers in these trips is reversed as well.

Our new metaheuristic approach for the TSPHS, is a two-phase multi-start procedure which combines (i) a simple GRASP (Resende and Ribeiro, 2003) heuristic to construct a feasible solution from scratch, and (ii) a *Variable Neighbourhood Descent* (VND) (Hansen and Mladenović, 2001) heuristic to improve that solution.

3.1 Construction phase

The purpose of the construction phase is to provide a good initial feasible TSPHS solution. To this end, a certain number (`ISOL`) of diverse solutions are generated, starting from an empty solution, and the

best one is subjected to the improvement phase. The steps performed by the construction phase are summarised in Algorithm 1.

Algorithm 1 Construction phase

```
t ← ∅
for i = 1 → ISOL do
  T ← GenerateTspTour()
  t_i ← Split(T)
  for all z such that z is a trip contained in t_i do
    z ← ImproveTrip(z)
  end for
  if t_i is better than t then
    t ← t_i
  end if
end for
return t
```

3.1.1 GenerateTspTour

At each iteration of the construction phase, a number `ISOL` of different TSP tours, visiting all customers and ignoring the budget constraint, are generated. The diversity of the generated solutions depends on the way these initial TSP tours are generated. Therefore, the TSP tour is not constructed by using a deterministic procedure such as the nearest neighbour heuristic. We have implemented a simple GRASP-based strategy: a TSP tour beginning at the starting hotel ($i = 0$) is created and customers are added one by one. Each time, the customer that is added to the TSP solution is randomly selected from the list of the `CLIST` nearest neighbours to the most recently added customer. Note that, the parameter `CLIST` can take any value from 1 to N. With a `CLIST` value of 1, the GRASP strategy reduces to a nearest neighbourhood heuristic, while, with a value of N it generates fully random TSP tours. As explained in Section 4.2 we will use a value of 3 for `CLIST`.

This GRASP approach performs much better than generating the initial TSP tours totally randomly. That would require a large number of moves to reach a local optimum and the resulting TSPHS solutions will be of low quality.

Each TSP tour generated by the GRASP approach is improved by using one or more of the intra-trip operators. Next, the resulting TSP tour, which is generally infeasible for the TSPHS, is split into a set of feasible trips.

3.1.2 Split

The first trip starts at the given starting hotel ($i = 0$) and the customers are added one by one in the order in which they appear in the TSP tour. A customer is added to the trip only if it is possible to reach at least one hotel from this customer without exceeding the time budget. When no more customers can be added to a trip, the trip is completed by including the hotel that is closest to the last customer.

The second trip starts at the ending hotel of the first trip and it is built using the same procedure as the first trip. The procedure is continued until all customers have been added to a trip. In theory, it is possible to end up in a scenario where, given an ending hotel of a certain trip, it is not possible to visit the next customer in the TSP sequence without exceeding the time budget. In such case, an empty trip is generated from that ending hotel directly to a more suitable one, i.e., a hotel closer to the next customer in the TSP sequence. Finally, the ending point of the last trip has to be the starting hotel ($i = 0$).

3.1.3 ImproveTrip

When a feasible TSPHS solution has been built, each trip contained in the solution is optimised. To this end, the intra-trip operators are used.

3.2 Improvement phase

The improvement phase starts with the best feasible TSPHS solution generated in the construction phase. The four neighbourhoods defined by the two inter-trip and two hotel selection operators `Relocate`, `Exchange`, `ChangeHotels` and `JoinTrips` are sequentially explored in a VND structure. The search over the neighbourhoods is performed in a best improvement fashion, i.e., the complete set of candidate solutions within a neighbourhood is explored and the best one is selected. Also, the search over each neighbourhood is repeated as long as a better solution can be found in that neighbourhood.

The VND algorithm starts by exploring the `Relocate` neighbourhood. When no more improving solutions are found in that neighbourhood, the algorithm starts exploring the `Exchange` neighbourhood. When the `Relocate` and `Exchange` neighbourhoods have been depleted, the algorithm continues with the hotel-selection related operators, first `ChangeHotels` and then `JoinTrips`.

Finally, after each of the four neighbourhoods has been explored, each modified trip is improved by means of intra-trip operators.

The improvement phase is repeated as long as better solutions are found. When it is no longer possible to find a better solution, the improvement phase terminates and the solution is reported.

Algorithm 2 presents in detail the steps performed in the improvement phase.

Algorithm 2 Improvement phase

Require: Initial feasible solution x / Neighbourhood structures $N_k, k = 1, \ldots, 4$
  repeat
    for $k = 1 \rightarrow 4$ **do**
      repeat
        Find the best neighbour x' of x in $N_k(x)$
        if x' is better than x **then**
          $x \leftarrow x'$
        end if
      until No improvement is possible for this k
    end for
    for all trips z contained in x **do**
      ImproveTrip(z)
    end for
  until No improvement was made in the current iteration

4 Computational experiments

In this section, the metaheuristic is tested on the existing benchmark instances. First, a brief description of the test instances is given, followed by a parametric analysis the goal of which is to set each of the parameters of the metaheuristic to its best possible level. Finally, the results obtained for all instances by our exact and heuristic approach are presented.

Throughout this section, the metaheuristic developed in Vansteenwegen et al. (2011) will be referred to as I2LS, whereas the metaheuristic developed in this paper will be referred to as VND.

All experiments were run on an Intel Core i7 850 processor with 2.93 GHz and 4 GiB of RAM.

4.1 Test instances

To test the heuristic developed in this paper, the four sets of instances developed in Vansteenwegen et al. (2011) were used. All of these instances are generated based on well-known VRP and TSP benchmarks and can be downloaded from `http://antor.ua.ac.be/tsphs`. In Table 1, the main characteristics of the benchmark instances are given.

Set	Number of customers	Number of hotels
SET 1	48–288	6
SET 2	10 15 30 40	2
SET 3	52–1002	3 5 10
SET 4	52–1002	10

Table 1: Parameters

4.2 Parametric analysis

In this section, a statistical experiment is run to determine the best values for the parameters of the algorithm developed in Section 3.

The construction phase of the algorithm is governed by 2 parameters: the size of the list of nearest neighbours (`CLIST`), and the number of generated solutions (`ISOL`). In addition, both the construction and the improvement phase of the algorithm are executed a certain user-defined number of times (`STARTS`). Therefore, the complete heuristic is controlled by five parameters: `STARTS`, `ISOL`, `CLIST`, `OPT1` and `OPT2`, where `OPT1` is the parameter indicating which intra-trip operators are used to improve the initial TSP tour and the TSPHS trips in the construction phase, while `OPT2` is the parameter indicating which operators are used to improve the trips in the improvement phase. The possible levels for `OPT1` and `OPT2` are: (1) `3-opt`, (2) `2-opt`, (3) `2-opt+Or-opt` (with $k = 1$), and (4) `2-opt+Or-opt` (with $k \leq 3$) where k is the number of consecutive customers that are to be relocated.

In order to determine the impact of each parameter on the algorithm's performance, an experiment was conducted with the aim to determine the most robust parameter configuration. In Table 2, an overview of the five parameters is provided, as well as the values which were tested for them. Note that the use of operator `3-opt` in the construction phase was ruled out because the required amount of CPU time is prohibitively large.

Parameter	Values	Number of levels
`STARTS`	5, 10, 20, 30	4
`ISOL`	5, 25, 50, 100	4
`CLIST`	3, 5	2
`OPT1`	2, 3, 4	3
`OPT2`	1, 2, 3, 4	4

Table 2: Parameters

A full factorial experiment was conducted on each of 30 different instances picked randomly from SET 1, SET 3 and SET 4, resulting in a total of $4^3 \times 2 \times 3 \times 30 = 11520$ observations. The instances were selected in such way that it was possible to test the algorithm on instances with a small, medium and large number of customers.

To quantify the impact of the five parameters on the algorithm's performance, two type III analysis of variance (ANOVA) models were estimated using the statistical software package JMP. The type III ANOVA models involve a random effect for each instance and fixed effects for the main effects and the interaction effects of the parameters.

The function of the random effect for the instances is to capture the correlation between $4^3 \times 2 \times 3$ test results for each instance. The assumption of random instance effects is justified because the instances

can be considered a random sample from the population of all instances.

The influence of the parameters on two performance measures was analysed: the objective function value and the total CPU time. For each of the two performance measures, a model was fitted involving all main effects and all two-parameter interactions. In Table 3, the results are shown for the two estimated ANOVA models. In Column 1, the parameter or interaction is shown, while, in Columns 2 and 3, the p-values for each of the estimated models are presented. Bold values indicate statistically significant main effects or interactions.

Parameter	p-values	
	Obj	Time
STARTS	**<0.0001**	**<0.0001**
ISOL	**<0.0001**	**<0.0001**
CLIST	**<0.0001**	**<0.0001**
OPT1	**<0.0001**	**<0.0001**
OPT2	0.3558	0.9476
STARTS×ISOL	**<0.0001**	0.8438
STARTS×CLIST	0.2121	0.8164
STARTS×OPT1	0.3723	1.0000
STARTS×OPT2	0.6127	1.0000
ISOL×CLIST	0.4411	0.8503
ISOL×OPT1	**<0.0001**	**<0.0001**
ISOL×OPT2	0.9065	1.0000
CLIST×OPT1	0.4031	0.7092
CLIST×OPT2	0.8223	0.9638
OPT1×OPT2	0.7658	1.0000

Table 3: Initial ANOVA model including all interactions

It can be seen that the parameters STARTS, ISOL, CLIST and OPT1, as well as the interaction ISOL×OPT1 are significant in both the model for the objective function value and the model for the CPU time. However, the interaction STARTS×ISOL is only significant for the quality of the final solution.

It is not surprising that the interaction ISOL×OPT1 is significant in both models. On the one hand, this interaction is directly related to the quality of the initial solution. On the other hand, since a TSP tour is generated ISOL times, and such tour is improved by means of the operator indicated by OPT1, this interaction determines to a large extent the required computing time. In other words, the CPU time and quality of the initial solution are strongly related to the quality of the final solution.

After the significant main effects and interactions have been identified, the initial models can be simplified by dropping the insignificant terms. In Tables 4 and 5, the resulting models are shown.

Source	Nparm	DF	DFDen	F Ratio	Prob > F
STARTS	3	3	11466	165.9244	**<0.0001**
ISOL	3	3	11466	96.0138	**<0.0001**
CLIST	1	1	11466	73.5436	**<0.0001**
OPT1	2	2	11466	56.8299	**<0.0001**
STARTS×ISOL	9	9	11466	4.7252	**<0.0001**
ISOL×OPT1	6	6	11466	4.9789	**<0.0001**

Table 4: Simplified ANOVA model for the objective function value

In Figures 3 and 4, it can be seen that, for increasing values of the parameter STARTS, the objective function value decreases while the computing time increases. Figure 3 also shows that there is a diminishing marginal return on increasing the number of starts and increasing the number of initial solutions. This behaviour is expected, and therefore it is necessary to define a number of starts which represent a good trade-off between the time requirements and the quality of the final solution.

Source	Nparm	DF	DFDen	F Ratio	Prob > F
STARTS	3	3	11475	6203.5100	**<0.0001**
ISOL	3	3	11475	9161.0690	**<0.0001**
CLIST	1	1	11475	532.4602	**<0.0001**
OPT1	2	2	11475	1229.2810	**<0.0001**
ISOL×OPT1	6	6	11475	14.8899	**<0.0001**

Table 5: Simplified ANOVA model for CPU time requirements

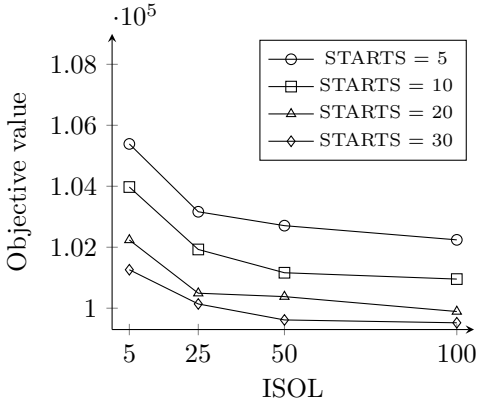


Figure 3: Influence of the interaction STARTS×ISOL on the solution quality

However, although the number of starts directly affects the quality and the time consumption of the algorithm, the interaction ISOL×OPT1 is strongly related with both performance measures as well. This can be seen in Figures 5a and 5b. From these figures, it can be seen that the value of parameter ISOL should be between 25 and 50. It certainly should not be greater than 50, because the increase in computing time is proportionally greater compared with the proportional gain in terms of quality. It can also be observed that the selection of a more complex OPT1 operator, leads to better solutions. However, the computing time increases faster with more complex operators. Therefore, the selection of the parameter OPT1 should be done keeping in mind the existing compromise between quality and time requirements as well.

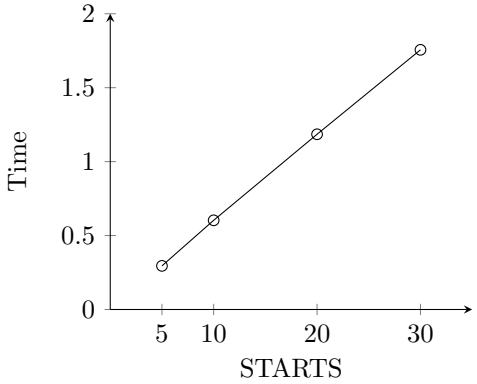
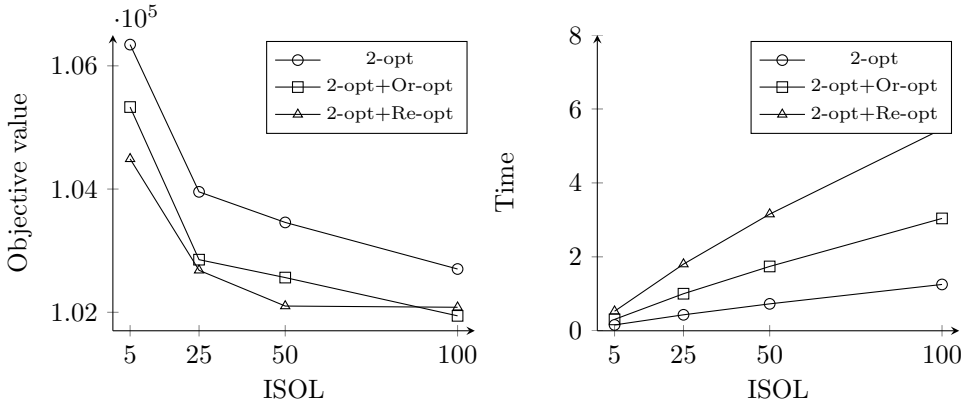


Figure 4: Influence of the parameter STARTS on the CPU time

Although statistically significant, the interactions mentioned in the previous paragraphs do not seem practically important, because the only clear conclusion that can be drawn is that the more time is spent solving this problem, the better the solution is obtained, which is, a typical behaviour of metaheuristics.

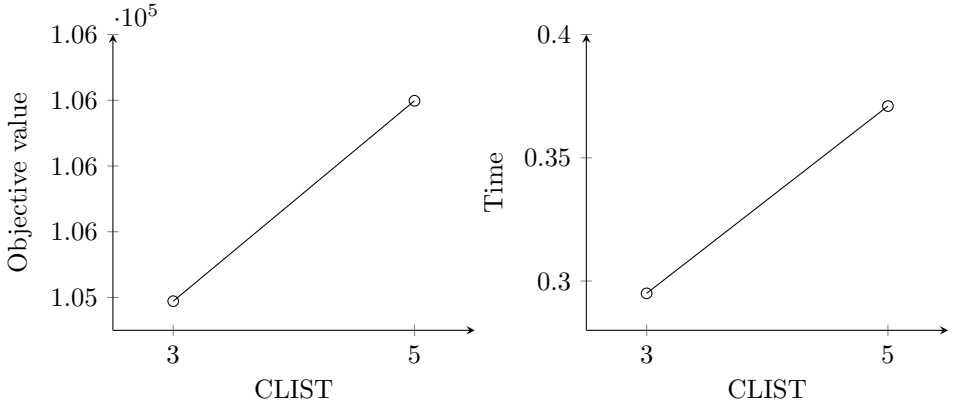
Finally, for the parameter CLIST, it can be observed in Figure 6 that the optimal choice should be 3, because both the total travel time and the time consumption decrease as this parameter increases.

The values we chose for each parameter are presented in Table 6, based on the results of the ANOVA. The



(a) ISOL/OPT1 vs Objective value (b) ISOL/OPT1 vs Time

Figure 5: Influence of the interaction `ISOL`×`OPT1` on the solution quality and the CPU time



(a) Objective value (b) CPU Time

Figure 6: Influence of the parameter `CLIST` on the solution quality and the CPU time

values were chosen in order to obtain the solutions in computing times similar to the ones reported for the I2LS heuristic. In order to compare the CPU times obtained by both metaheuristics, the procedure described in Gajpal and Abad (2009) was used to estimate the computing time of the I2LS heuristic in our machine. The resulting ratio was 0.8226 and it was used to estimate the CPU time spent by the I2LS heuristic on our machine.

Parameter	Value
`STARTS`	5
`ISOL`	50
`CLIST`	3
`OPT1`	2
`OPT2`	1

Table 6: Parameter settings

In Section 4.3, the results produced by our heuristic are presented using the parameter settings in Table 6. The computing times reported for the I2LS heuristic, when available, are also presented after applying the time ratio mentioned above. It is important to note, that our results can be improved in terms of quality by, for example, increasing the number of starts.

4.3 Results

The results for SET 1 are presented in Table 7. The first two columns contain the name and the number of customers of each instance. Columns 3 and 4 show the number of trips and the objective function value for the solution obtained by our approach, while Column 5 shows the required CPU time (in seconds). Columns 6 and 7 present the best results obtained by the I2LS approach, and Column 8 shows the computation time of that heuristic. Finally, Column 9 presents the gap percentage between both approaches. Positive values for the gap indicate that our VND approach outperforms the I2LS approach.

$$\text{Gap} = \frac{\text{Length}_{\text{I2LS}} - \text{Length}_{\text{VND}}}{\text{Length}_{\text{I2LS}}} \times 100\% \tag{1}$$

From Column 9, it can be seen that our approach was able to improve the results obtained by the I2LS approach for all benchmarks contained in SET 1. The improvement in solution quality comes at the expense of a larger CPU time for large instances, but not for small instances.

Instance	N	VND			I2LS			Gap (%)
		Trips	Length	Time	Trips	Length	Time	
c101	100	9	9622.7	0.4	9	9685.6	0.3	0.64
r101	100	9	1726.9	0.3	9	1801.3	0.4	4.13
rc101	100	8	1689.0	0.4	8	1724.1	0.3	2.03
c201	100	3	9564.1	0.4	3	9600.0	1.1	0.37
r201	100	2	1646.8	0.3	2	1678.0	1.5	1.85
rc201	100	2	1644.9	0.4	2	1670.0	1.3	1.50
pr01	48	2	1413.9	0.0	2	1446.0	0.2	2.21
pr02	96	3	2551.8	0.3	3	2569.3	0.9	0.68
pr03	144	4	3448.5	1.3	4	3584.1	1.3	3.78
pr04	192	5	4284.8	3.2	5	4366.3	2.3	1.86
pr05	240	6	5017.5	6.4	6	5122.1	4.7	2.04
pr06	288	7	6061.3	11.3	7	6137.3	6.0	1.23
pr07	72	3	2072.4	0.1	3	2090.9	0.2	0.88
pr08	144	4	3410.4	1.3	4	3504.7	1.1	2.69
pr09	216	5	4459.3	4.5	5	4617.6	3.3	3.42
pr10	288	7	6032.5	11.4	7	6097.5	6.0	1.06
Avg.				2.6			1.9	1.90

Table 7: Results for SET 1

The results for SET 2 are presented in Tables 8, 9, 10 and 11 (for 10, 15, 30 and 40 customers, respectively). The instances contained in this set were solved with an exact method developed in Vansteenwegen et al. (2011). For 28 of the 52 instances in SET 2, the exact method led to an optimal solution. In Column 1, the name of each instance is given. Columns 2 and 3 show the best solution obtained by the exact method, while Columns 4-9 show the number of trips, the total length, and the gap with respect to the solution found by the exact method for our VND approach and for the I2LS approach, respectively. The gaps are computed using the expression 1, but with respect to the best solution reported by the exact method. In this case, negative values indicate better solutions. No computing times are reported for this set since all of them are smaller than 0.2s for both metaheuristics.

Tables 8 and 9 contain the results for the instances with 10 and 15 customers. In each of these cases, our VND approach was able to obtain at least the best feasible solution found by the exact method, while the I2LS method was not, which resulted in gaps of up to 7.28% with respect to the optimal solution for the smallest instances and up to 6.31% for the larger instances. Moreover, for instance c101 in Table 9, our VND approach was able to find a better solution than the best feasible solution found by the exact method.

Name	Exact		VND			I2LS		
	Trips	Cost	Trips	Cost	Gap (%)	Trips	Cost	Gap (%)
c101	1	**955.1**	1	**955.1**	0.00	1	955.4	0.03
r101	2	**272.8**	2	**272.8**	0.00	2	286.2	4.91
rc101	1	237.5	1	237.5	0.00	1	237.5	0.00
pr01	1	**426.6**	1	**426.6**	0.00	1	**426.6**	0.00
pr02	1	**661.9**	1	**661.9**	0.00	1	**661.9**	0.00
pr03	1	**553.3**	1	**553.3**	0.00	1	559.1	1.04
pr04	1	**476.4**	1	**476.4**	0.00	1	511.1	7.28
pr05	1	**528.9**	1	**528.9**	0.00	1	560.9	6.05
pr06	1	**597.4**	1	**597.4**	0.00	1	604.1	1.12
pr07	1	**670.2**	1	**670.2**	0.00	1	708.1	5.65
pr08	1	**573.4**	1	**573.4**	0.00	1	573.4	0.00
pr09	1	**645.5**	1	**645.5**	0.00	1	647.5	0.30
pr10	1	**461.5**	1	**461.5**	0.00	1	461.5	0.00
Avg.					0.00			2.03

Note: Bold values indicate optimal solutions

Table 8: Results for SET 2 instances with 10 customers

Name	Exact		VND			I2LS		
	Trips	Cost	Trips	Cost	Gap (%)	Trips	Cost	Gap (%)
c101	2	1452.8	2	1452.2	-0.05	2	1456.7	0.26
r101	2	**379.8**	2	**379.8**	0.00	2	391.5	3.08
rc101	2	303.2	2	303.2	0.00	2	306.1	0.95
pr01	1	**590.4**	1	**590.4**	0.00	1	**590.4**	0.00
pr02	1	**745.6**	1	**745.6**	0.00	1	751.1	0.73
pr03	1	**632.9**	1	**632.9**	0.00	1	649.1	2.55
pr04	1	**683.4**	1	**683.4**	0.00	1	**683.4**	0.00
pr05	1	**621.2**	1	**621.2**	0.00	1	660.4	6.31
pr06	1	**685.2**	1	**685.2**	0.00	1	**685.2**	0.00
pr07	1	**795.3**	1	**795.3**	0.00	1	812.5	2.16
pr08	1	**707.2**	1	**707.2**	0.00	1	**707.2**	0.00
pr09	1	**771.7**	1	**771.7**	0.00	1	773.6	0.24
pr10	1	**611.9**	1	**611.9**	0.00	1	**611.9**	0.00
Avg.					0.00			1.26

Note: Bold values indicate optimal solutions.

Table 9: Results for SET 2 instances with 15 customers

Table 10 presents the results for the instances with 30 customers. As it can be seen from Columns 6 and 9, our method was able to either improve or at least reach the same solution reported by the exact method for 8 out of 13 instances, while the I2LS approach could only reach the same solution reported by the exact method for two instances.

The results for the instances with 40 customers are presented in Table 11. For this set, our VND approach

Name	Exact		VND			I2LS		
	Trips	Cost	Trips	Cost	Gap (%)	Trips	Cost	Gap (%)
c101	3	2876.3	3	2863.6	-0.45	3	2907.8	1.09
r101	3	676.2	3	655.7	-3.04	3	676.2	0.00
rc101	4	712.4	4	739.1	3.74	4	747.0	4.85
pr01	1	**964.8**	1	**964.8**	0.00	1	**964.8**	0.00
pr02	2	1078.3	2	1082.9	0.42	2	1140.6	5.77
pr03	1	**952.5**	1	**952.5**	0.00	1	957.1	0.48
pr04	2	1091.6	2	1091.6	0.00	2	1149.3	5.28
pr05	1	**924.7**	1	**924.7**	0.00	1	936.3	1.25
pr06	2	1065.3	2	1069.3	0.37	2	1114.4	4.60
pr07	2	1130.4	2	1130.4	0.00	2	1158.0	2.44
pr08	2	**968.4**	2	1006.2	3.90	2	1056.1	9.05
pr09	2	1091.4	2	1123.3	2.92	2	1133.1	3.82
pr10	1	**918.9**	1	**918.9**	0.00	1	927.1	0.89
Avg.					0.60			3.04

Note: Bold values indicate optimal solutions

Table 10: Results for SET 2 instances with 30 customers

was able to outperform, between 0.11% and up to 6.41% , the best feasible solution reported by the exact method for 7 of the 13 instances. Furthermore, for other 3 instances, the VND ended in the same solution reported by the exact method. On the other hand, the I2LS could reach the solution reported by the exact method for 6 instances, which in turn, were all outperformed by the VND approach. For the rest of the instances, the VND and I2LS approaches obtained gaps of up to 1.40% and 6.41%, respectively.

Name	Exact		VND			I2LS		
	Trips	Cost	Trips	Cost	Gap (%)	Trips	Cost	Gap (%)
c101	4	3950.0	4	3867.3	-2.09	4	3950.0	0.00
r101	4	895.5	4	862.8	-3.65	4	895.5	0.00
rc101	4	851.2	4	850.3	-0.11	4	851.2	0.00
pr01	2	1160.5	2	1176.8	1.40	2	1223.6	5.44
pr02	2	1336.9	2	1336.9	0.00	2	1418.2	6.08
pr03	2	1303.4	2	1316.1	0.97	2	1386.9	6.41
pr04	2	1259.5	2	1259.5	0.00	2	1292.9	2.65
pr05	2	1200.7	2	1200.7	0.00	2	1200.8	0.01
pr06	2	1271.5	2	1251.1	-1.60	2	1279.2	0.61
pr07	2	1426.5	2	1418.9	-0.53	2	1426.5	0.00
pr08	2	1305.9	2	1222.2	-6.41	2	1305.9	0.00
pr09	2	1284.4	2	1286.3	0.15	2	1287.0	0.20
pr10	2	1233.6	2	1200.4	-2.69	2	1233.6	0.00
Avg.					-1.12			1.26

Table 11: Results for SET 2 instances with 40 customers

The results for SET 3 are shown in Tables 12, 13 and 14. SET 3 was designed with the purpose of having a best known solution for each instance. In Column 1, the tables show the name and the size of the instance, while, in Column 2, the optimal length of the TSP solution for each instance is presented. This value is also the length of the best known solutions for this set. Columns 3-6 contain the solution found by our VND heuristic, the CPU time (in seconds) as well as the gap between the VND solution and the best length. In a similar way, Columns 7-10 show the results obtained with the I2LS heuristic.

From the results for SET 3, it is observed that our VND heuristic was able to find 2, 3 and 1 solutions with the minimum length for the instances with 3, 5 and 10 extra hotels, respectively, while the I2LS heuristic was also able to find one for an instance with 10 extra hotels. However, it is clear from Columns 6 and 10 in Tables 12, 13 and 14 that the VND heuristic consistently produced better solutions than the I2LS approach and that its CPU time is an order of magnitude lower.

We can also infer from the average gap obtained for instances with 3, 5 and 10 extra hotels that the difficulty of an instance increases with the number of hotels available. An incorrect selection of hotels has a detrimental impact on the quality of the final solution.

Name_N	TSP	VND				I2LS			
		Trips	Cost	Time	Gap (%)	Trips	Cost	Time	Gap (%)
eil_51	426	5	439	0.0	3.05	5	477	0.0	11.97
berlin_52	7542	4	**7542**	0.0	**0.00**	5	8150	0.0	8.06
st_70	675	5	705	0.1	4.44	5	754	0.0	11.70
eil_76	538	5	583	0.1	8.36	5	629	0.1	16.91
pr_76	108159	4	111081	0.2	2.70	4	111260	0.0	2.86
kroA_100	21282	4	21428	0.4	0.68	5	22806	0.0	7.16
kroC_100	20749	4	20812	0.4	0.30	5	23532	0.0	13.41
kroD_100	21294	5	22108	0.4	3.82	5	24870	0.1	16.79
rd_100	7910	5	8363	0.4	5.72	5	8869	0.1	12.12
eil_101	629	5	670	0.3	6.51	5	730	0.1	16.05
lin_105	14379	4	**14379**	0.5	**0.00**	5	16878	0.2	17.37
ch_150	6528	5	6971	1.5	6.78	5	7426	0.0	13.75
tsp_225	3916	5	4072	4.9	3.98	5	4555	6.0	16.31
a_280	2579	5	2787	9.0	8.06	5	3003	9.4	16.44
pcb_442	50778	5	53775	37.7	5.90	5	56058	96.2	10.39
pr_1002	259045	5	273764	563.7	5.68	5	291158	7250.3	12.39
Avg.				38.7	4.12			460.2	12.73

Note: Bold values indicate optimal solutions

Table 12: Results for SET 3 instances with 3 extra hotels

Name_N	TSP	VND				I2LS			
		Trips	Cost	Time	Gap (%)	Trips	Cost	Time	Gap (%)
eil_51	426	7	452	0.0	6.10	7	469	0.0	10.09
berlin_52	7542	6	**7542**	0.0	**0.00**	6	8200	0.0	8.72
st_70	675	6	694	0.1	2.81	7	777	0.0	15.11
eil_76	538	7	569	0.1	5.76	7	621	0.0	15.42
pr_76	108159	6	**108159**	0.2	**0.00**	7	123266	0.1	13.96
kroA_100	21282	7	22033	0.4	3.52	7	22644	0.0	6.39
kroC_100	20749	6	20885	0.4	0.65	8	23394	0.0	12.74
kroD_100	21294	6	22184	0.4	4.17	7	24287	0.0	14.05
rd_100	7910	7	8541	0.4	7.97	7	8638	0.0	9.20
eil_101	629	7	669	0.3	6.35	8	727	0.1	15.58
lin_105	14379	6	**14379**	0.5	**0.00**	7	16082	0.2	11.84
ch_150	6528	7	6945	1.4	6.38	8	7444	0.4	14.03
tsp_225	3916	7	4139	5.2	5.69	8	4792	5.1	22.36
a_280	2579	7	2827	9.1	9.61	8	3122	9.8	21.05
pcb_442	50778	7	53794	38.6	5.93	8	58494	97.5	15.19
pr_1002	259045	7	272517	588.5	5.20	8	298827	7245.9	15.35
Avg.				40.4	4.38			459.9	13.82

Note: Bold values indicate optimal solutions

Table 13: Results for SET 3 instances with 5 extra hotels

Name_N	TSP	VND				I2LS			
		Trips	Cost	Time	Gap (%)	Trips	Cost	Time	Gap (%)
eil_51	426	12	457	0.0	7.27	12	460	0.1	7.98
berlin_52	7542	8	7980	0.0	5.80	9	8272	0.1	9.67
st_70	675	11	726	0.1	7.55	10	**675**	0.3	**0.00**
eil_76	538	12	571	0.1	6.13	13	607	0.4	12.82
pr_76	108159	12	113609	0.2	5.03	14	121861	0.5	12.66
kroA_100	21282	11	**21282**	0.4	**0.00**	12	23379	0.0	9.85
kroC_100	20749	11	21366	0.4	2.97	13	23337	0.0	12.47
kroD_100	21294	12	21947	0.4	3.06	13	23529	1.3	10.49
rd_100	7910	11	8252	0.4	4.32	11	9158	0.9	15.77
eil_101	629	12	679	0.3	7.94	12	671	0.8	6.67
lin_105	14379	10	14457	0.5	0.54	11	16167	1.5	12.43
ch_150	6528	13	6983	1.4	6.96	12	7203	5.4	10.34
tsp_225	3916	13	4234	5.2	8.12	13	4377	24.1	11.77
a_280	2579	13	2808	9.1	8.87	14	3107	31.5	20.47
pcb_442	50778	13	53497	38.6	5.35	13	56912	511.0	12.08
pr_1002	259045	13	278895	588.5	7.66	13	289477	11831.4	11.74
Avg.				40.4	5.47			775.6	11.08

Note: Bold values indicate optimal solutions

Table 14: Results for SET 3 instances with 10 extra hotels

Finally, Table 15 contains the results for SET 4[1]. This set contains the most difficult instances, for which no optimal solutions are known. For each of these instances, our heuristic method was able to improve the best known solution. In Column 8, we can see the percentage of improvement achieved by our heuristic compared to the best known solution. For instance st_70, we can see that our heuristic gives a solution with a length greater than the I2LS heuristic. However, this is compensated by the fact that our heuristic was able to reduce the number of trips by one. For instances kroA_100, kroC_100, ch_150 and a_280, it was also possible to reduce the number of trips.

Name_N	VND			I2LS			Gap (%)
	Trips	Cost	Time	Trips	Cost	Time	
eil_51	6	430	0.0	6	479	0.0	10.22
berlin_52	7	8680	0.0	7	8823	0.0	1.62
st_70	6	763	0.1	7	745	0.0	-2.42
eil_76	6	564	0.1	6	595	0.0	5.21
pr_76	7	117924	0.2	7	129789	0.7	9.14
kroA_100	6	22205	0.4	7	22828	0.0	2.72
kroC_100	6	21694	0.4	7	23744	0.0	8.63
kroD_100	6	22041	0.4	6	24904	0.0	11.49
rd_100	6	8411	0.4	6	8769	0.0	4.08
eil_101	6	647	0.3	6	693	0.0	6.63
ch_150	6	6694	1.5	7	7679	0.1	12.82
tsp_225	7	4455	5.1	7	4819	4.3	7.55
a_280	6	2803	9.5	7	3123	7.5	10.24
pcb_442	7	57183	44.0	7	63822	72.3	10.4
pr_1002	7	304708	606.8	7	330282	4574.9	7.74
Avg.			44.6			310.7	7.07

Table 15: Results for SET 4

Regarding CPU times, it can be seen that, for instances with up to 150 customers, the time requirements for the VND and I2LS approaches are similar. However, for instances with a larger number of customers, the required computing time increases much faster for the I2LS heuristic than for the VND approach.

In Table 16, a summary of the results is provided. The table contains information related to the performance of both heuristic approaches (VND and I2LS) for each set of instances. The table contains two columns for each set. The first column corresponds to our VND heuristic, while the second column corresponds to the I2LS approach. The table's first row shows the number of optimal solutions found by

[1]In Vansteenwegen et al. (2011), an erroneous solution is given for the instance lin105. This instance contains one customer which cannot be included in any trip that satisfies the time budget. Such a customer can never be visited and, hence, the instance is infeasible. We did not use that instance in our experiments.

each approach. The second row presents the number of best known solutions found by each heuristic. The third row presents, for each heuristic, the average gap to the best known solution (for sets 1, 3 and 4) or to the optimal solution (for set 2). Finally, the fourth row presents the average time, in seconds, required for each approach. It should be clear that, on each of the four performance measures in Table 16, our approach outperforms the I2LS approach.

	SET 1		SET 2		SET 3		SET 4	
	VND	I2LS	VND	I2LS	VND	I2LS	VND	I2LS
# Optimal solutions	-	-	27	9	-	-	-	-
Best known solutions	16	0	-	-	6	1	15	0
Average gap (%)	0.00	1.90	0.13	1.99	4.65	12.54	0	7.07
Average time (%)	2.6	1.9	-	-	39.8	565.3	44.6	310.7

Table 16: Summary of results

5 Conclusions

The TSPHS is a difficult optimisation problem which arises in many practical situations. In this paper, a new heuristic solution method for the TSPHS is presented. The heuristic combines a construction method with a variable neighbourhood descent method. This approach clearly outperforms the only existing heuristic in the literature. The quality of the results is clearly improved on the benchmark instances while using similar CPU times.

As can be observed from the results for the benchmark instances in SET 2 and SET 3, where either optimal solutions or best solutions are known for most of the instances, our heuristic method was able to find such solutions for most smaller instances. It also led to small gaps with respect to the best solutions for larger instances.

As can be observed in Tables 12, 13 and 14, for a fixed number of customers, the difficulty of the problem increases with the number of hotels. This indicates that an erroneous selection of hotels has a negative impact on the final solution quality. Hence, the hotel selection is a keystone in solving the TSPHS.

We believe that more research on the TSPHS would be useful. First, studying extensions of the problem, including time windows, hotel costs, and vehicle capacity, would have added value. Also, it would be useful to seek alternative formulations of the TSPHS problem in order to be able to solve larger problems with a commercial solver.

Acknowledgement

The first, second and fourth author gratefully acknowledge financial support of the Fonds voor Wetenschappelijk Onderzoek – Vlaanderen (FWO).

References

E. Angelelli and M. Grazia Speranza. The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research*, 137(2):233–247, 2002.

D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The traveling salesman problem: a computational study*. Princeton University Press, 2007.

T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

A. M. Benjamin and J. E. Beasley. Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. *Computers & Operations Research*, 37(12):2270–2280, 2010.

J. F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

B. Crevier, J. F. Cordeau, and G. Laporte. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756–773, 2007.

G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

Y. Gajpal and P. L. Abad. Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. *European Journal of Operational Research*, 196(1):102–117, 2009.

G. Ghiani, G. Improta, and G. Laporte. The capacitated arc routing problem with intermediate facilities. *Networks*, 37(3):134–143, 2001.

G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Tabu search heuristics for the arc routing problem with intermediate facilities under capacity and length restrictions. *Journal of Mathematical Modelling and Algorithms*, 3(3):209–223, 2004.

P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

B. I. Kim, S. Kim, and S. Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, 2006.

G. Laporte, M. Gendreau, J. Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7(4-5):285–300, 2000.

S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10): 2245–2269, 1965.

I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Northwestern University, 1976.

M. Polacek, R. F. Hartl, K. Doerner, and M. Reimann. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics*, 10(6):613–627, 2004.

M. Polacek, K. F. Doerner, R.F. Hartl, and V. Maniezzo. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of Heuristics*, 14(5):405–423, 2008.

M. Resende and C. Ribeiro. Greedy randomized adaptive search procedures. *Handbook of metaheuristics*, pages 219–249, 2003.

P. Toth and D. Vigo, editors. *The vehicle routing problem*, volume 9 of *Discrete Mathematics and Applications*. Society for Industrial and Applied Mathematics, 2002.

P. Vansteenwegen, W. Souffriau, and K. Sörensen. The travelling salesperson problem with hotel selection. *Journal of the Operational Research Society*, 63(2):207–217, 2011.