Erasmus University Rotterdam
Erasmus School of Economics

Bachelor's thesis
Econometrics and Operations Research

Implementation of the iterative three-component heuristic for the team orienteering problem with time windows

Author: Y.K. Cheung (377511)

Supervisor: T.R. Visser, MSc
Co-reader: dr. R. Spliet

4 July 2016

Abstract

This thesis studies the Team Orienteering Problem with Time Windows (TOPTW). In this problem, a set of locations is given and each of these locations has a service time, a profit and a time window. The objective is to maximize the profit by visiting the locations, while taking into account the time window of the locations. The aim of this thesis is to reproduce the iterative three-component heuristic (I3CH) by Hu and Lim (2014) for TOPTW. This heuristic consists of a local search and a simulated annealing which give solutions that can be improved by a third component: routing recombination. The heuristic is applied on test instances which is also used in Hu and Lim (2014). Results of the Hu and Lim (2014) are verified and we were also able so find new best known solutions for the TOPTW. Furthermore, we adapted the I3CH for the Orienteering Problem with Hotel Selection and Time Windows. We tested the heuristic on a small set of benchmark instances which also give solutions near the best known solutions for these benchmark instances.

Contents

1 Introduction

This thesis studies the Team Orienteering Problem with Time Windows (TOPTW). The TOPTW is an optimization problem of which the objective is to maximize the profit by visiting a set of locations. Each location has a service time, a profit and a time window and a location can be visited at most once. The number of routes is fixed. A feasible route is defined as follows: it should start and end at the depot and the locations should visit within the time window.

A real-life application of the TOPTW is the generation of a personal tourist trip by electronic tourist guides. Tourists usually have little time when they visit a particular city. The electronic tourist guides provide a set of routes in which the most valuable points of interest (POI) are visited taking into account the opening hours of the POI and the available time spend at a POI. An application to the city of Ghent can be found in Souffriau et al. (2008).

Hu and Lim (2014) pose an iterative three-component heuristic (I3CH) to solve the TOPTW. The three components of the heuristic are local search (LS), simulated annealing (SA) and the route recombination (RR). The aim of this thesis is to reproduce the heuristic and verify the results of Hu and Lim (2014). Furthermore, the I3CH is adapted for the Orienteering Problem with Hotel Selection and Time Windows (OPHS-TW). In contrary to the TOPTW, a route contains a fixed starting hotel and ending hotel and should visit a given number of intermediate hotels.

This thesis is structured as follows. In Section 2, a literature review is given about the TOPTW and related problems. In Section 3 the TOPTW is described. The iterative three-component heuristic is described in Section 4 and its results are discussed in Section 5. The heuristic is adapted for the Orienteering Problem with Hotel Selection and Time Windows in Section 6. The thesis is concluded with a conclusion in Section 7.

2 Literature review

The TOPTW is one of the many variants of the classic Orienteering Problem (OP). The OP is derived from the outdoor sport *orienteering*. The participants start at a certain point, visit as much as possible checkpoints and then return to the starting position within a certain time limit. Each checkpoint has a certain score and the participants try to maximize their total score.

Golden et al. (1987) showed that the OP is $\mathcal{NP}$-hard, hence it is unlikely that an optimal solution can be found in polynomial time. Heuristics and meta-heuristics are the most common techniques to tackle large problem instances. Chao et al. (1996) pose a *two step fast and effective heuristic* for the OP. In the first step an ellipse around the starting point is created. The range of the ellipse is determined by the time limit. The second step consists of creating an optimal path within the ellipse.

The Team Orienteering Problem (TOP) is an extension of the OP. In this extended

problem, more than one route is possible.

The TOP with Time Windows (TOPTW) is an extension of the TOP. In the recent years, this extension gained more attention, because problem instances with time windows can not be solved efficiently using heuristics for problem without time windows (Vansteenwegen et al., 2011). Labadie et al. (2012) pose a granular Variable Neighborhood Search (VNS) procedure. The aim of the granular VNS is to reduce the size of the neighborhood by inspecting the dual problem of a relaxed integer programming of the TOPTW. Labadie et al. (2012) found that this increases the efficiency of the algorithm.

Vansteenwegen et al. (2009) pose iterated local search meta-heuristic to solve the TOPTW. The main contribution on this paper consists of an insert step combined with a shake step in order to escape from local optima. The performance of this heuristic is good on a large and diverse set of instances.

Montemanni et al. (2011) improved their ant colony system (ACS) heuristic (Montemanni and Gambardella, 2009) for the TOPTW. This type of algorithms for solving optimization problems is inspired by ant colonies. The parallel search for solutions is the main feature of the ACS. In this paper the enhanced ACS performs well on benchmark instances.

Lin and Yu (2012) pose a fast and a slow SA heuristic for the TOPTW. The fast SA has a low computation time, while the slow SA provides better solutions.

In Hu and Lim (2014), the I3CH is compared with among others the ACS by Montemanni and Gambardella (2009), the iterated local search algorithm of Vansteenwegen et al. (2009), the VNS by Tricoire et al. (2010), the slow SA by Lin and Yu (2012) and the granular VNS by Labadie et al. (2012). The average gap of the I3CH is smaller than other algorithms for most instances. However, the average computing time is only shorter for the ACS by Montemanni et al. (2011).

More recently, Gunawan et al. (2015) propose a hybrid algorithm. This algorithm is a combination of the iterative local search and the simulated annealing (SAILS). Instead of starting with a random solution for the SAILS, a greedy algorithm is used for creating an initial solution.

The Orienteering Problem with Hotel Selection and Time Windows (OPHS-TW) is an extension of the OP. This problem consists of one contiguous route that has a fixed starting and ending hotel with intermediate hotels and considering the time windows. There is little research conducted about this problem. Divsalar et al. (2013) developed a *Variable Neighborhood Search* to solve the Orienteering Problem with Hotel Selection. Divsalar et al. (2014) extended this to *Genetic Algorithm with a Variable Neighborhood Descent* to solve OPHS-TW.

3 Problem description

In the TOPTW a directed graph $G = (V, A)$ is given. This graph consists of a set of vertices $\mathcal{V}$ and a set of arcs $\mathcal{A}$. The vertices represent the locations and the set of arcs contains paths from $i \in \mathcal{V}$ to $j \in \mathcal{V}$ with $i \neq j$. This problem consists of $n + 1$ locations. Every location i is characterized by a profit p_i, a service time s_i, the x- and y-coordinate and a time window $[O_i, C_i]$. The depot is denoted by the location 0. The remaining locations represent the customers. These customers can be visited at most once. For each route $k \in \mathcal{P}$, k represents an ordered list of visiting customers starting and ending at location 0. Let $|\mathcal{P}| = m$, which is the number of routes set in advance. A visit to a customer should start within the customer's time window. If the vehicle arrives at a location before O_i, then the vehicle has to wait until O_i. The objective of the TOPTW is maximizing the profit by creating feasible set of routes with respect to the time windows.

This problem can also be formulated as a mixed integer programming. We use a formulation of the TOPTW that is formulated by Montemanni and Gambardella (2009). Let a binary variable x_{ij}^k be 1 if arc $(i, j) \in \mathcal{A}$ is in route k and 0 otherwise. Let z_i^k be an integer containing the time of visiting a customer i in route k. Let t_{ij} be the Euclidean distance between i and j and M be a large constant. Let location $n + 1$ be a dummy depot.

$$\max \sum_{k\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}} p_i x_{ij}^k \tag{1}$$

$$\text{s.t.} \sum_{k\in\mathcal{P}} \sum_{j\in\mathcal{V}} x_{ij}^k \leq 1 \qquad \forall i \in \mathcal{V}, \tag{2}$$

$$\sum_{j\in\mathcal{V}} x_{0j}^k = 1 \qquad \forall k \in \mathcal{P}, \tag{3}$$

$$\sum_{i\in\mathcal{V}} x_{ih}^k - \sum_{j\in\mathcal{V}} x_{hj}^k = 0 \qquad \forall h \in \mathcal{V} \setminus \{0, n+1\}, \forall k \in \mathcal{P}, \tag{4}$$

$$\sum_{i\in\mathcal{V}} x_{i(n+1)}^k = 1 \qquad \forall k \in \mathcal{P}, \tag{5}$$

$$z_i^k + t_{ij} + s_i - M(1 - x_{ij}^k) \leq z_j^k \qquad \forall i, j \in \mathcal{V}, \forall k \in \mathcal{P}, \tag{6}$$

$$O_i \leq z_i^k \leq C_i \qquad \forall i \in \mathcal{V}, \forall k \in \mathcal{P}, \tag{7}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall (i, j) \in \mathcal{A}, \forall k \in \mathcal{P}, \tag{8}$$

$$z_i^k \in \mathbb{N} \qquad \forall i \in \mathcal{V}, \forall k \in \mathcal{P}. \tag{9}$$

The objective function (1) maximizes the total profit of the customers. Constraints (2) ensure that each location is visited at most once. Constraints (3) impose that a route start at the depot. Constraints (4) ensures the connection of the arcs in a route. Constraints (5) impose that a route end at the depot. Inequalities (6) and (7) make sure that the time windows constraints are satisfied.

A solution for $m = 2$ is represented as follows: route $r_1 = \{0, 1, 2, 3, 4, 0\}$, $r_2 = \{0, 5, 6, 7, 8, 0\}$ and $u = \{9, 10\}$. In this representation route r_1 will start from the depot and visit customer 1, 2, 3 and 4, and then return back to the depot. Route r_2 visits customer 5, 6, 7 and 8. The remaining customers 9 and 10 are not visited. In figure 1, this solution is depicted.

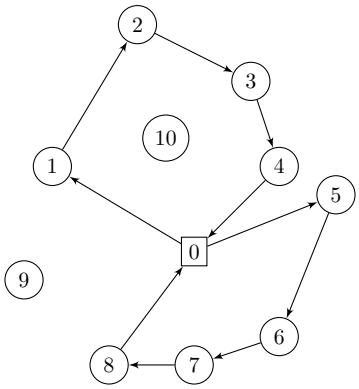


Figure 1: Example of a solution for $m = 2$

4 Methodology

The I3CH by Hu and Lim (2014) consists of three components. These components are the *local search* (LS), *simulated annealing* (SA) and the *route recombination* (RR). The routes found by the LS and SA are stored in a solution pool. The RR finds a new combination of routes with equal or higher profit found by one of the two first components. This procedure is repeated in the iterative framework until all customers are visited or the maximum number of iterations is reached.

4.1 The iterative framework

In the initialization the *Eliminator* operator generates $3 \cdot N$ solutions and select the best as the starting solution A. All routes of the initial solutions are stored in a route pool. As input for the *Eliminator* an empty solution is given with m empty routes and a randomly shuffled list of unvisited customers. Since no customers are visited, no customers can be removed. The *Eliminator* improves the solution by adding as much as possible customers in one of the routes. Thereafter, the *post-processing procedure* is executed to further improve the solution. The framework then subsequently runs RR, LS and SA until the stopping criterion is met. In algorithm 1 the framework is shown. In the following subsections, we first describe the *Eliminator* and then the individual components of the heuristic.

Algorithm 1 I3CH for the TOPTW

Require: maximum number of iterations I_{max}
Require: integer N repetitions
Require: route pool $POOL$
Use *eliminator* to create $3 \cdot N$ solutions and store routes in $POOL$ and obtain best starting solution A
$i \leftarrow 1$
while $i \leq I_{max}$ **do**
  $Z_{RR} \leftarrow$ solution from RR over $POOL$
  $X_{LS} \leftarrow$ best solution LS with N neighbors and store routes in $POOL$
  $Y_{SA} \leftarrow$ best solution SA with N steps and store routes in $POOL$
  $A \leftarrow \max\{A, X_{LS}, Y_{SA}, Z_{RR}\}$, breaking ties arbitrarily
  $i \leftarrow i + 1$
  if all customers are served in A **then**
    $i \leftarrow I_{max} + 1$
return Best solution A

4.2 *Eliminator*

The *Eliminator* is a neighborhood operator that searches for neighborhood solutions. The LS and SA invoke the *Eliminator* to obtain a neighborhood solution. Given a solution A, the *Eliminator* randomly removes customers from some routes with the following policy. A customer c is randomly selected from a route has the profit p_c. If the customer's profit is higher than the average profit of solution A, then this customer is removed with a probability P_h. If the customer's profit is lower than the average, then the customer is

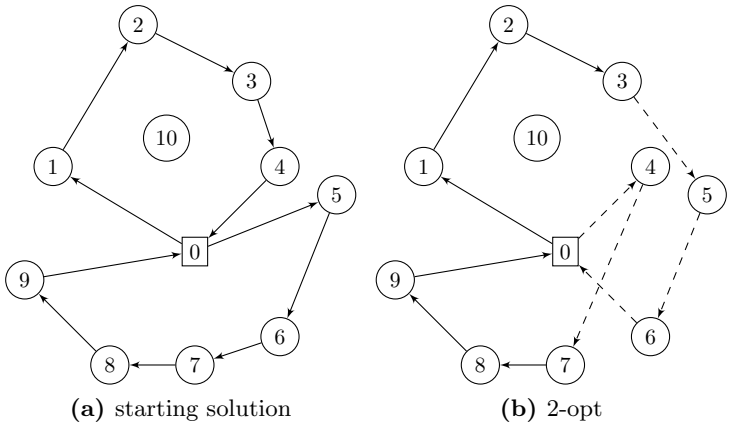
removed with a probability P_l. The probability P_h is lower than the probability P_l, to ensure that the more profitable is more likely to remain in the route. The probabilities P_h and P_l are set in advance. The removed customers are stored in a list of unvisited customers, denoted by u. This is solution A'.

After removing some customers, the routes of this solution provide space to insert customers. A' is called *partial*. We are now able to improve the solution by adding customers from u. After shuffling u, we pick the first customer c from the list and insert this customer into the first possible position in route r_i with $i = 1, ..., m$. If the customer c can not be inserted in r_i, the customer is moved to the end of the list. In case of a successful insertion, the customer is removed from u. This procedure is repeated until no further insertion could be made. The resulting solution B is called *complete*.

Procedure 2 Post-processing procedure

Require: solution B
impr $\leftarrow$ **true**
while impr **do**
  Apply 2-relocate, 2-opt and 2-exchange on B and obtain best solution B'
  Apply 1-relocate and 2-exchange on B' and obtain best solution B''
  Apply 0-relocate and 0-exchange on B'' and obtain best solution B'''
  if B''' is not an improvement of B **then**
    impr $\leftarrow$ **false**
  $B \Leftarrow B'''$
return Best solution B

Subsequently, the solution B will be further improved by the procedure *post-processing*. In this procedure, several neighborhood operator will be used. Each operator enumerates all feasible solutions. These neighborhood operators can be divided into three types: intra-routes, inter-routes and profit-increasing. In this order, the post-processing procedure first reduces the total travel distance. The total reduced travel distances provide more room the for profit-increasing operators.

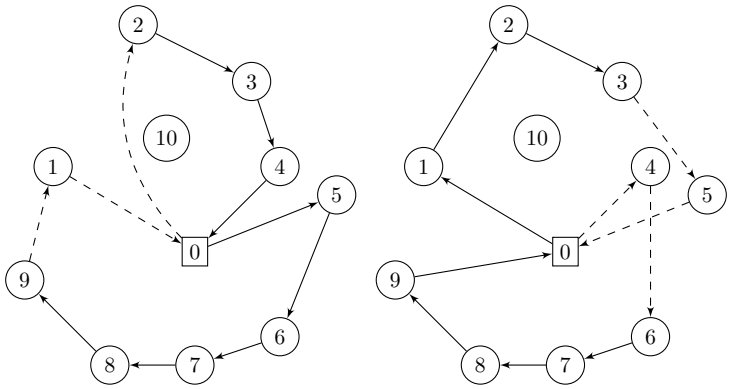


(a) starting solution **(b)** 2-opt

Figure 2: The starting solution and solution after a 2-opt operation.

The intra-route operators consist of 2-relocate, 2-opt and 2-exchange. The 2-opt selects

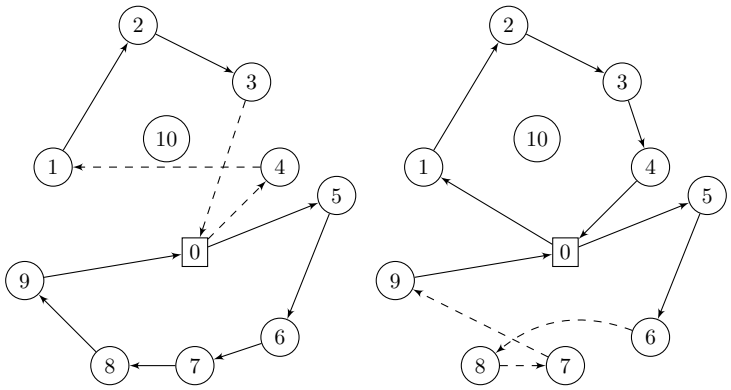
two routes r_i and r_j. Each route is divided into two parts. The first part of one route is connected with the last part of the other route. This operator prevents that routes are crossed over each other and will usually lead to a lower travel distance. In figure 2a, the starting situation is shown. Route r_1 is divided into $\{0, 1, 2, 3\}$ and $\{0, 4\}$. Route r_2 is split into $\{0, 5, 6\}$ and $\{7, 8, 9, 0\}$. The result of this 2-opt operation is shown in figure 2b. The new routes are: $\{0, 1, 2, 3, 5, 6, 0\}$ and $\{0, 4, 7, 8, 9, 0\}$. Note that the order of visiting locations of a split route can reverse. After a 2-opt operation, the route has to be feasible in terms of time windows, otherwise this solution is not used.



(a) 2-relocate **(b)** 2-exchange

Figure 3: Intra-route operators

A 2-relocate operation removes a customer c from route r_i and try to insert c in route j, provided that $i \neq j$. In figure 2a and 3a, customer 1 is moved from the route above the depot to the route below. The 2-exchange swaps customer c from r_i with customer d from r_j with $i \neq j$. In figure 3b customer 4 and 5 are swapped. The solution B' has the least travel time among the intra-route operators and is the starting point of the inter-route operators.

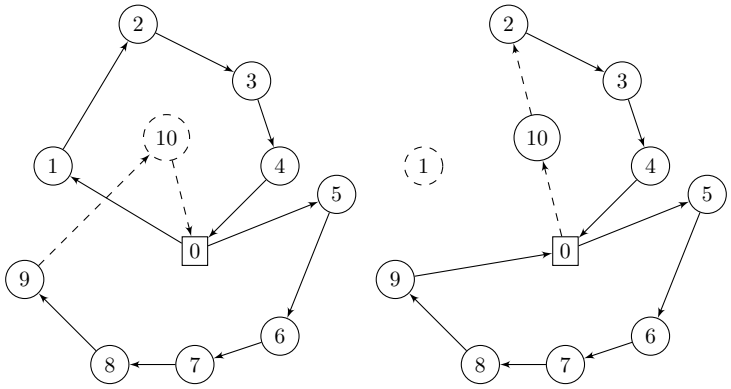


(a) 1-relocate **(b)** 1-exchange

Figure 4: Inter-route operators

Given solution B', the 1-relocate moves customer c to another position in the same

route, an example is shown in figures 2a and 4a. Customer 4 is routed between customer 3 and the depot, after the operation it is routed between the depot and customer 1. 1-exchange swap two customers from the same route with each other. In figure 2a and 4b a swap is shown with the customers 7 and 8. B'' represents the solution with the least travel time after the inter-route operations is used for the profit-increasing operators.



(a) 0-relocate **(b)** 0-exchange

Figure 5: Profit-increasing operators

The last part of the post-processing procedure contains the two profit-increasing operator. For each customer c in u, the 0-relocate operator attempts to insert c in any possible position of any route. An example of the 0-relocate is depicted in figures 2a and 5a. Customer 10, currently in the list of unvisited customers, is added into the route located below the depot. The result of this operation is shown in figure 5a. The 0-exchange swaps an unvisited customer with any customer in any route. Figure 5b shows the swap of a visited customer 1 to an unvisited customer 10. Only swaps, in which the unvisited customers has a higher profit than the customer to be replaced, are considered.

4.3 Local search

Given a starting solution A, the *local search* operator iteratively invokes the eliminator to obtain a neighborhood solution. In each iteration, N neighborhood solutions are explored. All routes of the N neighbors are stored in the route pool. X is updated with the neighborhood with the highest total profit if there is one, otherwise the last explored operator replaces X. Let X_{best} the best solution found so far, if the best solution has a higher total profit than X, there is an improvement. The *local search* explores until the maximum number of consecutive iterations with no improvement, denoted by $I_{\text{no impr}}$, is reached. Then, the current best solution of the heuristic A will replace X.

4.4 Simulated annealing

The *simulated annealing* (SA) produces each time one iteration and moves to the neighbor if it is an improvement, otherwise it will move with a certain probability. The main

Algorithm 3 Local search

Require: starting solution A
Require: maximum number of no improvements $I_{\text{no impr}}$
Require: integer N repetitions
if first local search **then**
  $X \leftarrow A$, $X_{best} \leftarrow X$
if $I_{\text{ls no impr}} > I_{\text{no impr}}$ **then**
  $X \leftarrow A$, $I_{\text{ls no impr}} \leftarrow 0$
for N neighbors **do**
  $X' \leftarrow$ neighbor of X from the *Eliminator*
  if X' is better than X_{best} **then**
    $X_{best} \leftarrow X'$, $I_{\text{ls no impr}} \leftarrow 0$
if no new X_{best} **then**
  $X \leftarrow$ last explored neighbor
  $I_{\text{ls no impr}} \leftarrow I_{\text{ls no impr}} + 1$
else
  $X \leftarrow X_{best}$
return X_{best}

advantage of this method is that the SA could escape from a local optimum. Let T_0, α and $I_{\text{no impr}}$ be respectively, the starting temperature, the cooling speed and the maximum consecutive iterations with no improvement. At each step, a neighbor Y' of starting solution Y is explored. If there is an improvement, the SA will move to Y'. If the neighbor Y' is better than best solution Y_{best}, then Y_{best} is replaced. In case of no improvement, SA will accept the neighbor with a certain probability. This probability is computed with Equation (10). The SA explores until the maximum number of consecutive iterations with no improvement, denoted by $I_{\text{no impr}}$, is reached. Then, the current best solution of the heuristic A will replace Y.

Algorithm 4 Simulated annealing

Require: starting solution A
Require: maximum number of no improvements $I_{\text{no impr}}$
Require: starting temperature T_0, cooling speed α
Require: integer N repetitions
if first simulated annealing **then**
  $Y \leftarrow A$, $Y_{best} \leftarrow Y$,$T \leftarrow T_0$
if $I_{\text{sa no impr}} > I_{\text{no impr}}$ **then**
  $Y \leftarrow A$, $T \leftarrow T_0$, $I_{\text{sa no impr}} \leftarrow 0$
for N neighbors **do**
  $Y' \leftarrow$ neighbor of Y from the *Eliminator*
  if Y' is better than Y **then**
    $Y \leftarrow Y'$
    if Y' is better than Y_{best} **then**
      $Y_{best} \leftarrow Y'$, $I_{\text{sa no impr}} \leftarrow 0$
  else if $rand(0,1) < P_{SA}$ **then**
    $Y \leftarrow Y'$
  $T \leftarrow \alpha \cdot T$
if no new X_{best} **then**
  $I_{\text{sa no impr}} \leftarrow I_{\text{sa no impr}} + 1$
return Y_{best}

$$P_{SA} = \exp\left(\frac{1}{T}\frac{Y' - Y}{Y_{best}}\right) \tag{10}$$

The probability of accepting increases as the SA is cooling down, that is, the temperature is decreasing. Furthermore, if the neighborhood solution is relatively far from the the starting solution, the P_{SA} goes to 1. After each step, the temperature is updated: $T \leftarrow \alpha T$ and the routes of the neighbor is stored in POOL. The values of Y, Y' and Y_{best} represent the total profit of the starting solution, neighbor and best solution respectively.

4.5 Route recombination

The route recombination (RR) creates a new solution from routes from POOL. The RR solves a set packing formulation. The aim is to cover as much as possible locations with the routes from POOL. The size of POOL is restricted by a number S_{pool}. Let $POOL = \{r_1, r_2...., r_{S_{pool}}\}$, where S_{pool} is the size of POOL. For all customers $c \in C$ and $k \in \{1, 2, ..., S_{pool}\}$, a_{ck} indicates whether a customer is in route r_k. The total profit a route is computed by the sum of the visited customers of a route and is denoted by p_k. The Integer Programming (IP) is defined as follows.

$$\max \sum_{k=1}^{S_{pool}} p_k x_k \tag{11}$$

$$\text{subject to} \sum_{k=1}^{S_{pool}} a_{ck} x_k \leq 1 \qquad \forall c \in C, \tag{12}$$

$$\sum_{k=1}^{S_{pool}} x_k \leq m, \tag{13}$$

$$x_k \in \{0, 1\} \qquad \forall k \in \{1, 2, ..., S_{pool}\}. \tag{14}$$

The objective function maximizes the profit over the routes. The first constraints restrict that a customer can be visited more than once. The second restriction sets the maximum number of routes used in the new solution.

After the new combination of routes, a list of unvisited customers u is created and sorted decreasing order of profit. Then a complete solution is created by adding as much as possible customers from u.

Each route in POOL has a value called *apr* denoting its appreciation. The value of new routes are assigned with an $apr = 0$. Routes that are included in the solution found by the IP are assigned an *apr* of 100. The *apr* of routes, which are not in the final solution, are decreased by one. Since a maximum number of routes in the POOL S_{pool} is set, it is more likely to keep the routes which are already used. When the number of routes exceeds the S_{pool}, we remove the routes with the lowest *apr* from POOL until the number of routes in POOL is below the maximum. Before we solve the IP, the upper bound of LP

relaxation can be used to determine whether it is necessary to solve the IP. Therefore, the IP is only solved, if the upper bound is better than the best solution found. This prevents the unnecessary use of the time consuming IP solver (Hu and Lim, 2014).

5 Computational results

The iterative heuristic is written in Java. The computations were run on a laptop equipped with an Intel Core i7-6500U CPU clocked at 2.50 GHz, 8 GB RAM and running Windows 10. The ILOG CPLEX 12.6.3 (Windows 64-bit version) is used to solve the MIP. To compare the running time with the original I3CH, we use *Super Pi* as an benchmark. *Super Pi* calculates the first million decimals of π. The seconds it takes to compute these decimals is the benchmark. The performance of our laptop is slightly better than Hu and Lim (2014), 13.2 and 14.7 respectively.

5.1 Test instances

The I3CH is tested on the benchmark instances originally created for the Vehicle Routing Problem with Time Windows (VRPTW) by Solomon (1987) and Cordeau et al. (1997). Each of the Solomon instances contains 100 locations and 1 depot. These locations are grouped into the distribution of the coordinates of the locations. The geographical locations of r100 and c100 instances are respectively randomly uniform distributed and clustered. In the rc100 instances, the locations are both randomly distributed and clustered. Furthermore, the locations of r100, c100 and rc100 have short time windows, we denote these three instance sets by Solomon 100. The Cordeau et al. (1997) instances pr01-pr10 each has a different number of locations. The geographical coordinates of these locations are randomly distributed. We denote these instances by Cordeau 1-10. Righini and Salani (2009) adapted the Solomon 100 and Cordeau 1-10 instances for the OPTW.

Montemanni and Gambardella (2009) adapted the Solomon (1987) and Cordeau et al. (1997) instances for long time windows and these modified instances are respectively denoted by Solomon 200, which contains r200, c200 and rc200, and Cordeau 11-20. These instances are the same as the ones used by Hu and Lim (2014).

The calculated Euclidean distances between locations are rounded down to one decimal for the Solomon instances and to two decimals for the Cordeau et al. instances.

5.2 Comparison with the I3CH

We initially run the aforementioned instances for $m = 1, ..., 4$ with a random $seed = 3$. Note that, if $m = 1$ the TOPTW is equivalent to the OPTW. Therefore, the RR is not used when $m = 1$, because it would only select the route with the highest profit (Hu and Lim, 2014). We run the I3CH with the same parameter settings as used by Hu and Lim (2014). The settings of the *Eliminator* parameters are $P_h = 0.1$ and $P_l = 0.3$. As starting temperature for the SA, we used $T_0 = 0.1$ and the cooling speed is $\alpha = 0.995$. The maximum number of routes in POOL is set on $S_{pool} = 1000$ and the maximum number of iterations $I_{max} = 3000$. The number of repetitions in LS, SA and *Eliminator* is set $N = 50$. Furthermore, the LS and SA will use a new starting solutions after 20 iterations of no improvement $I_{\text{no impr}}$.

In the table 1, a summary of the results of the Hu and Lim's I3CH and the reproduction shown. The column m represents the number of routes. The AG is the average gap between the best known solution BKS from 2014. The gap of an instance is calculated with Equation (15). The column AT shows the average computation time in seconds and is not adjusted to the computer's speed of the original I3CH. The columns $\#B$ and $\#W$ show the number of instances for which the I3CH reproduction has found better or worse solutions than the best known solutions (BKS) respectively. The BKSs are retrieved from Hu and Lim (2014) are used and could be improved by others in the meantime. In the appendix, the tables A6, A7, A8 and A9 show the solutions of each Solomon instance for $m = 1, 2, 3, 4$ respectively.

$$Gap = \frac{BKS - I3CH}{BKS} \times 100\% \tag{15}$$

Table 1: Comparison Solomon instances with $seed = 3$

m	Set	I3CH (original)		I3CH				m	Set	I3CH (original)		I3CH			
		AG (%)	AT (s)	AG (%)	AT (s)	#B	#W			AG (%)	AT (s)	AG (%)	AT (s)	#B	#W
1	c100	0.00	25.2	0.00	38.2	0	0	3	c100	0.00	190.2	-0.11	248.1	1	0
	r100	0.56	28.6	0.06	33.7	0	1		r100	0.21	118.3	0.00	204.5	1	1
	rc100	1.66	25.6	0.00	32.6	0	0		rc100	0.26	101.0	0.30	175.3	1	2
	c200	0.40	84.4	0.00	201.5	0	0		c200	0.00	12.3	0.00	42.5	0	0
	r200	1.04	176.2	0.33	629.3	3	5		r200	0.01	90.8	0.00	265.5	0	0
	rc200	2.68	119.3	0.73	337.6	2	4		rc200	-0.04	164.1	-0.03	511.3	1	1
2	c100	0.00	87.0	0.00	159.2	0	0	4	c100	0.01	261.8	0.01	388.9	1	1
	r100	0.54	63.0	0.09	142.2	1	3		r100	0.05	184.3	-0.16	301.5	8	0
	rc100	0.90	58.9	0.00	127.9	0	0		rc100	0.12	152.4	-0.23	276.4	4	0
	c200	0.68	401.2	-0.08	1112.3	1	0		c200	0.00	0.1	0.00	53.3	0	0
	r200	0.16	526.8	-0.38	1171.0	7	0		r200	0.00	0.2	0.00	82.4	0	0
	rc200	0.56	439.7	0.00	1360.3	5	2		rc200	0.00	0.2	0.00	46.3	0	0

The AG and AT denote the average gap in percentage and average computation time in seconds respectively. The #B and #W denote the number of instances for which the solution found is better and worse than BKS respectively.

Table 1 shows that our I3CH underperforms in terms of computation time. Especially, for instance sets modified by Solomon 200 for all m, the solving time is greater than the original I3CH. For $m = \{1, 2, 3\}$, this is possible due to the longer time windows, which lead to more possible exchange and relocate possibilities. However, the average gap is for the most instance sets lower than the original I3CH of Hu and Lim (2014) and in some cases even better than the BKS.

Table 2 shows the performance of the I3CH on the Cordeau instances. On average, the gaps of our I3CH is smaller than the average gap of the original heuristic. However, also in this case the the computation time is longer. This is possibly due to the difference in interpretation and the coding of the reproduction of the I3CH. In table 3, the results are shown for the Cordeau instances pr01-pr10, using the number of routes of which all customers could be visited. The table shows that the gap is lower than the original I3CH. However, we were not able to find the optimal solution which the original I3CH could not find.

Table 2: Comparison Cordeau instances with $seed = 3$

m	Cordeau	I3CH (original) AG (%)	I3CH (original) AT (s)	I3CH AG (%)	I3CH AT (s)	#B	#W
1	1-10	1.06	109.0	0.32	129.3	1	1
	11-20	3.79	130.2	1.43	170.1	2	6
2	1-10	0.94	247.1	0.08	429.2	2	3
	11-20	2.69	304.6	1.71	552.3	2	8
3	1-10	0.35	424.0	0.19	765.3	5	3
	11-20	1.00	497.0	0.34	989.2	4	4
4	1-10	0.06	566.5	-0.15	1157.6	4	4
	11-20	-0.64	728.6	-1.36	1564.3	7	2

Table 3: Performance using minimum required m to obtain the optimal solution for Coredeau 1-10

name	m	OPT	I3CH (original) profit	I3CH (original) gap (%)	I3CH (original) t (s)	I3CH profit	I3CH gap (%)	I3CH t (s)
pr01	3	657	619	5.78	146.7	622	5.33	210
pr02	6	1220	1207	1.07	669.7	1213	0.57	1288
pr03	9	1788	1781	0.39	1383.7	1785	0.17	3779
pr04	12	2477	2477	0	641.9	2477	0.00	174
pr05	15	3351	3351	0	19.3	3351	0.00	180
pr06	18	3671	3671	0	30	3671	0.00	325
pr07	5	948	943	0.53	299.7	945	0.32	543
pr08	10	2006	2006	0	55.9	2006	0.00	63
pr09	15	2736	2736	0	10.8	2736	0.00	142
pr10	20	3850	3850	0	9.1	3850	0.00	385

5.3 Parameter tuning

The heuristic has nine parameters. I_{max}, N and $I_{\text{no impr}}$ in the iterative framework, P_h and P_l in the *Eliminator*, T_0 and α in the simulated annealing, S_{pool} in the route recombination and a random seed. We use the instances c103 and c105 to tune the parameters with $m = 4$. These two instances perform the same as the original I3CH in terms of profit. The limited number of instances on which we tune the parameters is partly due to the time restriction of the research.

We first tune the parameters of the neighbor operator *Eliminator*. The parameters of the *Eliminator* which we test $P_h \in \{0.0, 0.1, 0.2, 0.3, 0.5\}$ and $P_l \in \{0.1, 0.2, 0.3, 0.5, 0.75, 1.0\}$. We repeated this four times with $seed \in \{3, 5, 7, 9\}$. Table 4 shows the results of the parameter tuning on the two instances. The parameter settings with $P_h = 0.5$ and for all $P_l = 0.2$ give on average the lowest gap. This is different than the parameter setting of Hu and Lim (2014), which was $P_h = 0.1$ and $P_l = 0.3$. However, our sample is small and therefore we can not state that these parameter setting is useful for all instances.

The RR has one parameter S_{pool}, the size of the route pool. A small route pool size probably leads to a worse solution quality. A large route pool size leads to a longer solving time when the RR also uses the IP. Again we tested this with $seed \in \{3, 5, 7, 9\}$. We used

Table 4: Parameter tuning on the *Eliminator*

P_h	$P_l = 0.1$	$P_l = 0.2$	$P_l = 0.3$	$P_l = 0.5$	$P_l = 0.75$	$P_l = 1.0$
0	1.73	1.73	1.73	1.73	1.73	1.73
0.1	1.73	1.74	1.63	1.63	1.63	1.63
0.2	1.63	1.63	1.63	1.63	1.63	1.63
0.3	1.63	1.63	1.63	1.74	1.63	1.63
0.5	1.63	1.51	1.81	1.73	1.73	1.89

the following parameter values for the $S_{pool} = \{100, 200, 300, 400, 500, 750, 1000, 1500, 2000\}$. The best average gap with 1.51 is at $S_{pool} = 750$.

Table 5: Parameter tuning on RR

S_{pool}	AG (%)	AT(%)
100	1.73	127.4
200	1.63	130.8
300	1.63	153.9
400	1.63	144.1
500	1.63	150.6
750	1.51	161.6
1000	1.63	171.0
1500	1.63	198.5
2000	1.63	205.3

5.4 Stability of best solutions

Some instances beat the BKS, therefore we are interested whether this solution is found fortuitously or this is structural. We rerun the several instances with a new best solution with different seeds: $seed = \{2, 4, 6, 8, 10\}$. In table 6, the results are shown.

Table 6: Stability of best solution

instance	m	Min gap	Max gap	AG(%)	AT(%)
r202	2	-0.30	-0.15	-0.22	1439.0
r209	2	-1.28	-0.21	-0.75	1964.4
c108	4	-0.88	0.00	-0.53	355.0
rc104	4	-0.57	-0.57	-0.57	354.2
pr03	4	-2.05	-1.23	-1.69	784.8
pr15	4	-0.87	0.58	0.01	2767.6

Table 6 shows that the results vary over the different seeds. That is, more optimal solution than the BKS is found most of the times, except for the instances c108 and pr15. For pr15, a solution was found which was worse than the BKS. However, on average the BKS was beaten.

5.5 Analysis RR, LS and SA

We use the computational results of Section 5.2 to analyze the individual components. The histogram of figure 6 shows the frequency of the different components which give the best final solution for the Solomon instances. Since the RR is not invoked for $m = 1$, only the results of $m = 2, 3, 4$ is showed in the graph. The RR provides in more than two third of the Solomon 100 instances the best final solution. This is a confirmation the effectiveness of the heuristic. For the Solomon 200 instances, the most optimal solution was usually found after the initialization. That is, a solution is found in which the all customers were visited.

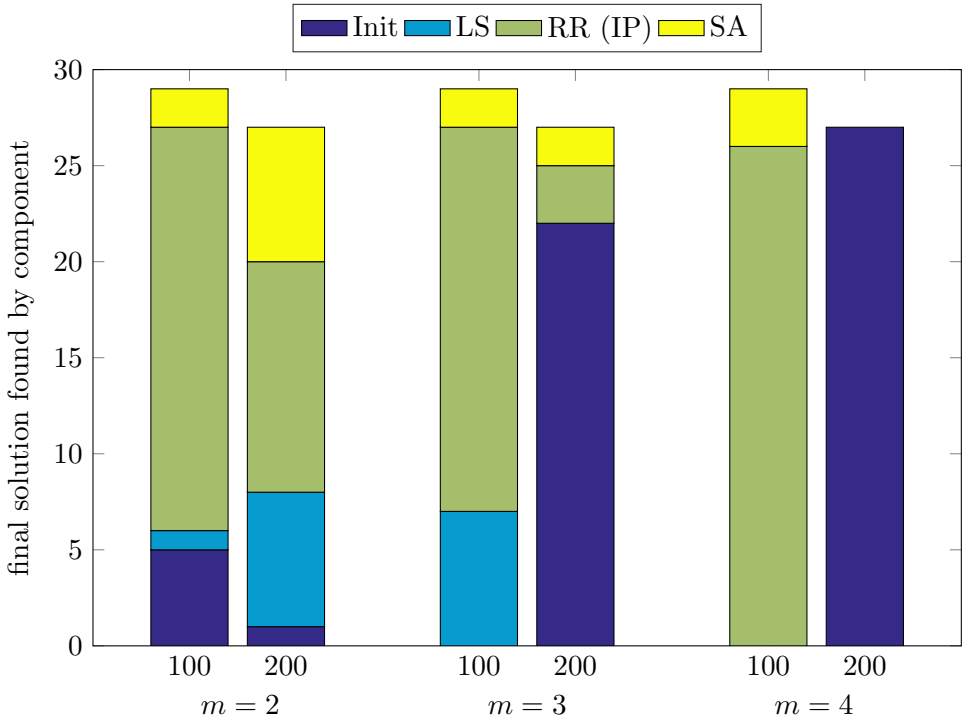


Figure 6: Best final solution for each component for the Solomon instances

Table 7: Computation time of the individual components of the I3CH

m	Instance set	AT (s) RR	LS	SA	m	Instance set	AT (s) RR	LS	SA
1	Solomon 100	0	17	17	3	Solomon 100	82	63	64
	Solomon 200	0	202	211		Solomon 200	19	102	104
	Cordeau 1-10	0	64	65		Cordeau 1-10	112	321	329
	Cordeau 11-20	0	84	86		Cordeau 11-20	126	421	438
2	Solomon 100	72	35	36	4	Solomon 100	95	111	115
	Solomon 200	125	499	568		Solomon 200	0	3	3
	Cordeau 1-10	103	160	164		Cordeau 1-10	118	508	526
	Cordeau 11-20	108	218	224		Cordeau 11-20	131	691	733

Table 7 shows the average computing time of the three components for the instance set Solomon 100, 200, Cordeau 1-10 and Cordeau 11-20. The computation times of the LS

and SA are almost equal, since the *Eliminator* is used N times for both components. The computation time of the RR component is usually lower than the other two combined. Sometimes the RR takes only 10% of the total computation time. For the instances on which the RR lead to the best solution, the RR usually takes a larger part of the total computation time. An explanation for this could be that RR then solves the integer programming.

Table 8 shows the average number of times the component obtain a new best solution during solving an instance. Noteworthy is that for $m = 2$ for all instance sets the RR often provides new best solution. That also holds for the Solomon 100 for $m = 3$ and $m = 4$.

Table 8: Average number of new best best solutions by component

m	Set	RR	LS	SA	m	Set	RR	LS	SA
1	c100	0.0	0.3	0.0	3	c100	1.2	0.4	0.3
	r100	0.0	0.7	0.8		r100	3.2	0.4	0.3
	rc100	0.0	0.8	0.3		rc100	3.4	0.4	0.1
	c200	0.0	1.1	0.8		c200	0.0	0.0	0.0
	r200	0.0	5.3	5.6		r200	0.1	0.2	0.3
	rc200	0.0	4.9	3.3		rc200	0.9	0.9	0.8
2	c100	0.6	0.4	0.1	4	c100	1.9	0.1	0.4
	r100	2.8	0.2	0.8		r100	5.7	0.0	0.3
	rc100	2.5	0.4	0.6		rc100	3.9	0.0	0.4
	c200	0.6	0.6	0.4		c200	0.0	0.0	0.0
	r200	2.7	4.4	2.7		r200	0.0	0.0	0.0
	rc200	4.6	2.5	2.6		rc200	0.0	0.0	0.0

6 The OP with Hotel Selection and Time Windows

6.1 Problem description

The OP with Hotel Selection and Time Windows (OPHS-TW) is another variant of the Orienteering Problem. In the OPHS-TW, a directed graph $G = (V, A)$ is given. This problem consists of at least $n+2$ locations: n customers and a begin and ending hotel. Also a set of intermediate hotels is given and also the number of routes. Not all intermediate hotels need to be visited. Let a route be a ordered set of locations which start and end at a hotel. Let a tour be a ordered set of routes. The tour should start and end with the given hotels. Each customer has a profit p_i, a service time s_i and time window $[O_i, C_i]$.

Since there are multiple possible orders to visit hotels in between the starting hotel end ending hotel, we fix the the order of hotels on beforehand. Note that, this could lead to a sub-optimal solution since the order of hotels is fixed.

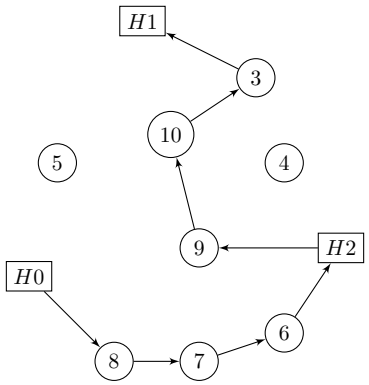


Figure 7: Example of a solution for $m = 2$ with one additional hotel

6.2 Adjustments to I3CH

To solve the OPHS-TW, we need to determine the order of the intermediate hotels before the I3CH can be executed. As a optimal order of hotels is challenging (Divsalar et al., 2013), we simplify the hotel selection with the following algorithm. From the starting hotel, we choose the next intermediate hotel of which the difference between the distance from the starting hotel to the next hotel and the trip length is the smallest. We repeat this until the number of intermediate hotels are visited. Then we add the ending hotel to the list of visiting hotels. The disadvantage of this algorithm is that we do not necessarily obtain the optimal ordering of visiting hotels. Moreover, there is a possibility that the distance to the ending hotel is larger than the trip length.

Furthermore, we change the operator 2-opt. The original 2-opt splits a route into two parts. By splitting two routes and reconnecting each other, there exists a possibility that the the order of visiting hotels could be changed. Therefore, we split the the route, without considering the hotels. If we apply this adapted 2-opt operation to the solution

in figure 7, we obtain the solution in figure 8. The remaining operators do not need to be changed, since these operators do not interact with the hotels.

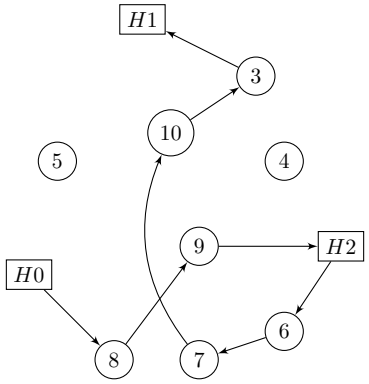


Figure 8: Example of a solution after a 2-opt operation

The route $r_1 = \{H0, 8, 7, 6, H2\}$ is split into $\{8\}$ and $\{7, 6\}$ and the route $r_2 = \{H2, 9, 10, 3, H1\}$ is split into $\{9\}$ and $\{10, 9\}$. The new routes are $\{H0, 8, 9, H_2\}$ and $\{H2, 6, 7, 10, 9, H1\}$.

Since the routes do not start and end at one hotel, the route recombination has to be changed. We need to find a sequence of routes with the right order of visiting hotels. Let define $t \in T$ as a trip type between starting hotel h_i and h_j. A solution is feasible if it contains all trip types. Equation (14) of the RR is changed into:

$$\sum_{k=1}^{S_{pool}} b_{tk} x_k = 1 \qquad \forall t \in T \tag{16}$$

Where b_{tk} is an indicator variable which equals to 1 if the trip k is of trip type t and 0 otherwise. The LS and SA component do not need to be changed, since these components do not alter the hotels in a particular route.

6.3 Results

The performance of the I3CH on the OPHS-TW is tested on several instances retrieved from http://www.mech.kuleuven.be/en/cib/op. We only applied the I3CH on a small number of routes and intermediate hotels. These instances have different number of locations, total trip length, number of routes and number of intermediate hotels. The number of intermediate hotels we use is $m - 1$, that means we visit all given intermediate hotels. We use the same parameters as used by Hu and Lim (2014) for the I3CH. Not all instances are were solvable, since our hotel selection algorithm was not always able to return a feasible hotel selection. In table 9 a summary of our results are shown. An optimal solution for each instance is known, therefore the gap is the percentage difference between the optimal solution value and the solution value found by the heuristic.

The results of table 9 show that for $m = 2$, the average gap is smaller than Divsalar et al. (2014). The computation time of the I3CH is substantially higher for all m. The

Table 9: Summary of results I3CH for OPHS-TW

		I3CH		Divsalar et al. (2014)	
m	#instances	AG (%)	AT (s)	AG (%)	AT (s)
2	20	0.98	203.20	3.80	0.98
3	17	10.90	161.65	1.55	0.68
4	10	17.06	141.00	1.59	0.56

average gap increases if m increases. The main cause could be the hotel selection algorithm. More detailed results for each instance are shown in table A10.

7 Conclusion

In this thesis, we have reproduced the iterative three-component heuristic by Hu and Lim (2014) for the TOPTW. The computational results show that our reproduction performs on average as good as the original I3CH or sometimes even better in terms of solution quality with the same parameter settings as Hu and Lim (2014). For the most instance sets, the average gap between the best known solutions and the solutions of our I3CH was 0.00%. For several instance sets the average gap was even lower, which means we also found some new best known solutions. However, our reproduced heuristic did not perform better in computation time. Since both the original I3CH and the reproduction is written in Java, coding or difference in interpretation could be the cause of the different computation times. Nonetheless, we do reach the computation time of the Hu and Lim (2014).

Partly due to the longer computation time, running a parameter tuning takes a lot of computation time. Therefore, we had to limit the number of instances on which we tune the parameters, and the number of parameters tuned. However, our results with the parameter settings of Hu and Lim (2014) show that it already performs relatively good. This is a confirmation that the I3CH of Hu and Lim (2014) is effective in terms of solution quality.

In addition to the the reproduction of the heuristic, we also applied the reproduction on the OPHS-TW, one of the variants of the original OP. We were able to solve some instances adapted for the OPHS-TW. However, not every instance could be solved by our adapted I3CH. The order of visiting intermediate hotels is a difficult problem (Divsalar et al., 2013). Therefore, when it is desirable to use the I3CH on the OPHS-TW, a better hotel selection algorithm is preferable. A possibility for further research could be a combination of the I3CH and hotel selection algorithm.

References

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1996). A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475 – 489.

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

Divsalar, A., Vansteenwegen, P., and Cattrysse, D. (2013). A variable neighborhood search method for the orienteering problem with hotel selection. *International Journal of Production Economics*, 145(1):150 – 160.

Divsalar, A., Vansteenwegen, P., Chitsaz, M., Sörensen, K., and Cattrysse, D. (2014). Personalized multi-day trips to touristic regions: A hybrid ga-vnd approac.

Golden, B. L., Levy, L., and Vohra, R. (1987). The orienteering problem. *Naval research logistics*, 34(3):307–318.

Gunawan, A., Lau, H. C., and Lu, K. (2015). Sails: Hybrid algorithm for the team orienteering problem with time windows. In *Proceedings of the 7th Multidisciplinary International Scheduling Conference (MISTA 2015)*.

Hu, Q. and Lim, A. (2014). An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 232(2):276 – 286.

Labadie, N., Mansini, R., Melechovský, J., and Calvo, R. W. (2012). The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, 220(1):15 – 27.

Lin, S.-W. and Yu, V. F. (2012). A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 217(1):94 – 107.

Montemanni, R. and Gambardella, L. (2009). Ant colony system for team orienteering problems with time windows. *Foundations of Computing and Decision Sciences*, 34(4):287–306.

Montemanni, R., Weyland, D., and Gambardella, L. M. (2011). An enhanced ant colony system for the team orienteering problem with time windows. In *Computer Science and Society (ISCCS), 2011 International Symposium on*, pages 381–384.

Righini, G. and Salani, M. (2009). Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, 36(4):1191 – 1203.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265.

Souffriau, W., Vansteenwegen, P., Vertommen, J., Vanden Berghe, G., and Van Oudheusden, D. (2008). A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence*, 22(10):964–985.

Tricoire, F., Romauch, M., Doerner, K. F., and Hartl, R. F. (2010). Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37(2):351 – 367.

Vansteenwegen, P., Souffriau, W., and Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1 – 10.

Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., and Oudheusden, D. V. (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281 – 3290. New developments on hub location.

Appendices

A Parameter settings CPLEX

Table A1: Parameter settings CPLEX MIP solver

parameter	value
`IloCplex.Param.Simplex.Tolerances.Feasibility`	1E-09
`IloCplex.Param.Simplex.Tolerances.Optimality`	1E-09
`IloCplex.Param.Simplex.Tolerances.Markowitz`	0.99999
`IloCplex.DoubleParam.EpGap`	1E-09
`IloCplex.IntParam.ParallelMode`	1
`IloCplex.IntParam.Threads`	1

B Comparision with I3CH

The following pages in this section contain the solution values of the I3CH for the Solomon 100, Solomon 200, Cordeau 1-10 sn Cordeau 11-20 instances. The parameter settings are as follows: $P_h = 0.1$, $P_l = 0.3$, $T_0 = 0.1$, $\alpha = 0.995$, $S_{pool} = 1000$, $I_{max} = 3000$, $N = 50$ and $I_{\text{no impr}} = 20$. The tables shows the profit, the gap and the computation time of the original I3CH by Hu and Lim (2014) and the reproduced I3CH. In addition, the computation time of each the three components, RR, LS and SA, are given. The best known solutions (BKS) are from Hu and Lim (2014) and thus some solutions could be outdated. The solution values in bold are solutions that are better than the BKS and Hu and Lim (2014). G denotes the gap between the BKS and the solution.

Table A2: Comparision I3CH for $m = 1$ and $seed = 3$

name	BKS	I3CH (original)			I3CH					
		profit	gap (%)	t (s)	profit	gap (%)	t (s)	RR	LS	SA
pr01	308	305	0.97	20.8	308	0.00	40	0	19	20
pr02	404	394	2.48	47.9	404	0.00	77	0	38	39
pr03	394	394	0.00	72.9	394	0.00	90	0	44	46
pr04	489	489	0.00	109.3	489	0.00	126	0	62	64
pr05	595	594	0.17	185.4	595	0.00	227	0	111	115
pr06	590	590	0.00	189.9	**591**	-0.17	223	0	109	113
pr07	298	298	0.00	26.5	298	0.00	29	0	14	15
pr08	463	454	1.94	77.4	463	0.00	81	0	40	41
pr09	493	490	0.61	137.8	493	0.00	161	0	80	80
pr10	594	568	4.38	222.2	574	3.37	239	0	119	120
pr11	351	353	-0.57	30.8	353	-0.57	31	0	15	15
pr12	442	433	2.04	59.8	438	0.90	73	0	37	36
pr13	461	466	-1.08	89.5	457	0.87	111	0	54	57
pr14	567	521	8.11	144.4	555	2.12	181	0	91	90
pr15	685	707	-3.21	248.2	**708**	-3.36	333	0	163	169
pr16	674	619	8.16	228.6	631	6.38	298	0	148	149
pr17	362	360	0.55	34.7	362	0.00	39	0	19	20
pr18	539	497	7.79	99.0	539	0.00	103	0	50	53
pr19	562	538	4.27	164.6	560	0.36	217	0	106	111
pr20	667	588	11.84	202.7	616	7.65	315	0	158	156

Table A3: Comparision I3CH for $m = 2$ and $seed = 3$

name	BKS	I3CH (original)			I3CH					
		profit	gap (%)	t (s)	profit	gap (%)	t (s)	RR	LS	SA
pr01	502	502	0.00	51.8	502	0.00	147	82	32	31
pr02	714	714	0.00	127.7	714	0.00	274	111	82	79
pr03	742	731	1.48	175.6	742	0.00	326	115	105	106
pr04	924	917	0.76	270.1	920	0.43	445	106	163	175
pr05	1090	1101	-1.01	410.0	1094	-0.37	642	99	261	280
pr06	1076	1040	3.35	427.6	1049	2.51	700	117	290	290
pr07	566	566	0.00	71.8	566	0.00	178	90	44	44
pr08	834	824	1.20	184.1	834	0.00	319	89	114	114
pr09	905	878	2.98	304.9	904	0.11	495	103	191	200
pr10	1124	1117	0.62	447.0	**1145**	-1.87	766	120	322	322
pr11	566	559	1.24	71.3	559	1.24	199	83	59	56
pr12	774	768	0.78	143.6	768	0.78	284	95	91	96
pr13	831	832	-0.12	238.6	832	-0.12	376	88	143	143
pr14	1017	978	3.83	337.3	992	2.46	614	110	247	254
pr15	1219	1205	1.15	479.1	1183	2.95	928	129	401	394
pr16	1231	1124	8.69	500.5	1156	6.09	913	129	384	396
pr17	652	639	1.99	117.0	639	1.99	204	87	57	59
pr18	938	937	0.11	231.0	**944**	-0.64	425	106	155	163
pr19	1034	1003	3.00	386.1	1029	0.48	634	125	249	257
pr20	1232	1155	6.25	541.6	1209	1.87	946	129	394	420

Table A4: Comparision I3CH for $m = 3$ and $seed = 3$

name	BKS	I3CH (original)			I3CH					
		profit	gap (%)	t (s)	profit	gap (%)	t (s)	RR	LS	SA
pr01	622	622	0	132.1	622	0.00	210	80	62	66
pr02	942	936	0.64	260.6	**943**	-0.11	405	85	159	160
pr03	1010	1010	0	301.5	996	1.39	524	100	206	216
pr04	1294	1286	0.62	442.7	**1295**	-0.08	791	113	331	344
pr05	1482	1481	0.07	650.3	**1499**	-1.15	1210	155	514	535
pr06	1514	1501	0.86	651.2	1487	1.78	1218	141	533	538
pr07	744	738	0.81	260.4	744	0.00	277	89	90	97
pr08	1138	1139	-0.09	307.0	**1141**	-0.26	550	105	218	225
pr09	1275	1272	0.24	503.1	1270	0.39	960	104	424	427
pr10	1573	1567	0.38	731.1	**1574**	-0.06	1508	149	676	677
pr11	654	654	0	151.7	654	0.00	256	79	84	92
pr12	1002	997	0.5	294.3	981	2.10	453	106	166	180
pr13	1139	1145	-0.53	378.9	**1156**	-1.49	691	125	276	287
pr14	1372	1315	4.15	533.7	1335	2.70	928	120	392	411
pr15	1650	1654	-0.24	708.1	**1673**	-1.39	1682	151	750	772
pr16	1668	1609	3.54	818.1	1629	2.34	1747	154	782	803
pr17	838	841	-0.36	184.3	840	-0.24	335	110	112	112
pr18	1281	1276	0.39	386.6	1281	0.00	689	122	274	291
pr19	1417	1403	0.99	604.1	1413	0.28	1238	140	534	557
pr20	1684	1658	1.54	909.7	1699	-0.89	1873	149	837	879

Table A5: Comparision I3CH for $m = 4$ and $seed = 3$

name	BKS	I3CH (original)			I3CH					
		profit	gap (%)	t (s)	profit	gap (%)	t (s)	RR	LS	SA
pr01	657	657	0.00	0.1	657	0.00	1	0	0	0
pr02	1079	1073	0.56	380.6	1078	0.09	586	102	231	251
pr03	1222	1232	-0.82	436.6	**1247**	-2.05	744	103	306	332
pr04	1557	1585	-1.80	603.6	1573	-1.03	1228	123	539	559
pr05	1833	1838	-0.27	902.9	1814	1.04	2060	161	934	954
pr06	1860	1835	1.34	939.6	**1881**	-1.13	2198	133	1005	1049
pr07	876	872	0.46	228.9	876	0.00	389	158	112	117
pr08	1382	1377	0.36	429.9	1377	0.36	783	118	323	338
pr09	1619	1604	0.93	698.5	**1622**	-0.19	1389	130	617	635
pr10	1939	1943	-0.21	1044.4	1911	1.44	2198	149	1009	1029
pr11	657	657	0.00	0.1	657	0.00	2	0	0	0
pr12	1132	1120	1.06	477.1	1125	0.62	703	119	271	309
pr13	1364	1386	-1.61	672	**1387**	-1.69	1143	118	485	528
pr14	1670	1651	1.14	783.2	**1683**	-0.78	1614	151	706	747
pr15	1958	2065	-5.46	1161.7	**2067**	-5.57	2743	173	1266	1291
pr16	2065	2017	2.32	1183.8	2050	0.73	2923	180	1328	1399
pr17	933	934	-0.11	332.8	934	-0.11	509	100	192	215
pr18	1525	1539	-0.92	559.5	**1549**	-1.57	1101	128	468	500
pr19	1723	1750	-1.57	919.4	**1764**	-2.38	2019	169	881	958
pr20	2037	2062	-1.23	1196.6	**2095**	-2.85	2886	173	1314	1383

Table A6: Comparision I3CH for $m = 1$ and $seed = 3$

instance	BKS	I3CH (original)			I3CH						instance	BKS	I3CH (original)			I3CH					
		profit	G (%)	t (s)	profit	G (%)	t (s)	RR	LS	SA			profit	G (%)	t (s)	profit	G (%)	t (s)	RR	LS	SA
c101	320	320	0.00	21.8	320	0.00	29	0	14	14	c201	870	870	0.00	70.1	870	0.00	125	0	62	62
c102	360	360	0.00	28.1	360	0.00	42	0	21	21	c202	930	930	0.00	87.6	930	0.00	194	0	93	99
c103	400	400	0.00	27.1	400	0.00	44	0	22	22	c203	960	960	0.00	92.3	960	0.00	268	0	132	134
c104	420	420	0.00	27.1	420	0.00	51	0	25	25	c204	980	970	1.02	117.4	980	0.00	361	0	170	187
c105	340	340	0.00	23.4	340	0.00	37	0	19	19	c205	910	900	1.10	70.7	910	0.00	137	0	68	68
c106	340	340	0.00	23.6	340	0.00	32	0	16	16	c206	930	920	1.08	75.7	930	0.00	184	0	90	93
c107	370	370	0.00	24.7	370	0.00	34	0	17	17	c207	930	930	0.00	77.4	930	0.00	161	0	76	83
c108	370	370	0.00	24.8	370	0.00	32	0	16	17	c208	950	950	0.00	84.0	950	0.00	182	0	88	93
c109	380	380	0.00	26.3	380	0.00	43	0	21	21											
r101	198	198	0.00	20.4	198	0.00	23	0	11	11	r201	797	789	1.00	101.8	797	0.00	210	0	103	106
r102	286	286	0.00	29.3	286	0.00	32	0	16	16	r202	929	930	-0.11	175.6	922	0.75	377	0	177	196
r103	293	293	0.00	28.8	293	0.00	33	0	17	17	r203	1021	1020	0.10	221.4	1015	0.59	658	0	315	338
r104	303	298	1.65	27.3	303	0.00	39	0	20	20	r204	1086	1073	1.20	236.9	1086	0.00	1239	0	602	624
r105	247	247	0.00	26.0	247	0.00	27	0	14	14	r205	953	946	0.73	129.3	953	0.00	332	0	163	167
r106	293	293	0.00	29.4	293	0.00	31	0	15	16	r206	1029	1021	0.78	169.3	994	3.40	681	0	335	342
r107	299	297	0.67	27.8	297	0.67	33	0	17	17	r207	1072	1050	2.05	192.8	**1078**	-0.56	775	0	372	396
r108	308	306	0.65	29.7	308	0.00	43	0	22	22	r208	1112	1092	1.80	230.0	1111	0.09	1169	0	574	580
r109	277	277	0.00	31.1	277	0.00	35	0	18	18	r209	950	948	0.21	136.5	938	1.26	373	0	179	192
r110	284	284	0.00	33.9	284	0.00	37	0	18	19	r210	987	982	0.51	176.9	**1002**	-1.52	470	0	232	234
r111	297	295	0.67	27.7	297	0.00	34	0	17	17	r211	1046	1013	3.15	167.4	**1050**	-0.38	638	0	314	319
r112	298	289	3.02	32.0	298	0.00	37	0	18	19											
rc101	219	219	0.00	21.8	219	0.00	27	0	14	14	rc201	795	795	0.00	80.9	795	0.00	156	0	76	80
rc102	266	266	0.00	25.5	266	0.00	26	0	13	13	rc202	936	924	1.28	129.3	934	0.21	266	0	130	135
rc103	266	266	0.00	27.1	266	0.00	28	0	14	14	rc203	1003	966	3.69	134.3	1003	0.00	395	0	192	200
rc104	301	301	0.00	27.2	301	0.00	36	0	18	18	rc204	1140	1093	4.12	167.5	**1143**	-0.26	853	0	411	438
rc105	244	244	0.00	26.4	244	0.00	38	0	19	19	rc205	859	847	1.40	99.2	853	0.70	199	0	97	102
rc106	252	250	0.79	25.0	252	0.00	33	0	17	16	rc206	895	863	3.58	98.4	**899**	-0.45	197	0	96	100
rc107	277	274	1.08	26.3	277	0.00	34	0	17	17	rc207	983	957	2.64	122.0	947	3.66	266	0	130	135
rc108	298	264	11.41	25.1	298	0.00	30	0	15	15	rc208	1053	1003	4.75	123.0	1032	1.99	369	0	181	185

Table A7: Comparision I3CH for $m = 2$ and $seed = 3$

instance	BKS	I3CH (original)			I3CH						instance	BKS	I3CH (original)			I3CH					
		profit	G (%)	t (s)	profit	G (%)	t (s)	RR	LS	SA			profit	G (%)	t (s)	profit	G (%)	t (s)	RR	LS	SA
c101	590	590	0.00	53.4	590	0.00	123	64	30	29	c201	1460	1450	0.68	321.2	1460	0.00	566	124	216	225
c102	660	660	0.00	78.4	660	0.00	145	62	41	42	c202	1470	1470	0.00	405.9	1470	0.00	885	105	384	392
c103	720	720	0.00	116.1	720	0.00	289	180	54	54	c203	1480	1470	0.68	458.2	1480	0.00	1359	135	586	631
c104	760	760	0.00	94.4	760	0.00	173	53	58	62	c204	1480	1480	0.00	498.5	1490	-0.68	2102	149	910	1029
c105	640	640	0.00	70.6	640	0.00	119	60	29	29	c205	1470	1450	1.36	322.2	1470	0.00	744	125	303	313
c106	620	620	0.00	149.2	620	0.00	179	111	33	34	c206	1480	1480	0.00	355.9	1480	0.00	974	137	407	427
c107	670	670	0.00	65.4	670	0.00	129	64	32	33	c207	1490	1470	1.34	423.5	1490	0.00	1073	158	435	475
c108	680	680	0.00	85.9	680	0.00	129	57	36	36	c208	1490	1470	1.34	424.3	1490	0.00	1195	142	494	554
c109	720	720	0.00	69.4	720	0.00	147	53	48	46											
r101	349	349	0.00	42.1	349	0.00	97	59	19	19	r201	1250	1254	-0.32	333.6	**1259**	-0.72	681	136	256	285
r102	508	508	0.00	62.4	508	0.00	135	75	29	30	r202	1347	1344	0.22	588.5	**1351**	-0.30	1203	153	452	587
r103	522	519	0.57	68.3	521	0.19	133	60	36	36	r203	1414	1416	-0.14	815.9	**1431**	-1.20	2628	193	1127	1280
r104	552	549	0.54	75.3	550	0.36	156	75	40	40	r204	1458	1458	0.00	33.5	1458	0.00	95	0	9	7
r105	453	447	1.32	56.2	453	0.00	118	62	28	28	r205	1379	1380	-0.07	606.1	**1387**	-0.58	1271	161	520	582
r106	529	529	0.00	63.5	529	0.00	144	80	32	32	r206	1440	1427	0.90	739.9	**1450**	-0.69	3169	211	1377	1560
r107	535	533	0.37	63.2	**538**	-0.56	165	90	38	37	r207	1458	1458	0.00	453.8	1458	0.00	47	0	2	2
r108	558	550	1.43	65.7	552	1.08	163	73	45	44	r208	1458	1458	0.00	4.3	1458	0.00	113	0	5	4
r109	506	506	0.00	60.5	506	0.00	132	62	34	35	r209	1405	1404	0.07	600.3	**1410**	-0.36	1766	173	714	865
r110	525	525	0.00	68.9	525	0.00	160	79	40	40	r210	1423	1415	0.56	728.5	**1428**	-0.35	1858	179	758	906
r111	544	542	0.37	67.0	544	0.00	150	78	36	36	r211	1458	1450	0.55	890.9	1458	0.00	50	3	10	12
r112	544	534	1.84	63.0	544	0.00	153	79	37	37											
rc101	427	427	0.00	52.3	427	0.00	120	70	25	25	rc201	1377	1384	-0.51	267.4	1377	0.00	493	115	182	194
rc102	505	505	0.00	59.8	505	0.00	125	63	31	31	rc202	1509	1500	0.60	386.7	1511	-0.13	752	127	286	333
rc103	524	519	0.95	60.3	524	0.00	131	64	34	33	rc203	1632	1627	0.31	482.8	1639	-0.43	1272	149	517	591
rc104	575	556	3.30	59.9	575	0.00	144	64	40	41	rc204	1716	1704	0.70	760.8	1718	-0.12	3840	178	1650	1970
rc105	480	480	0.00	58.6	480	0.00	112	58	27	27	rc205	1458	1452	0.41	311.0	1460	-0.14	626	124	239	259
rc106	483	481	0.41	56.0	483	0.00	127	72	28	28	rc206	1546	1525	1.36	335.9	1540	0.39	812	119	336	352
rc107	534	529	0.94	62.9	534	0.00	122	62	30	30	rc207	1587	1582	0.32	408.6	1579	0.50	906	126	368	404
rc108	556	547	1.62	61.4	556	0.00	142	69	36	36	rc208	1691	1669	1.30	564.4	1692	-0.06	2181	144	937	1085

Table A8: Comparision I3CH for $m = 3$ and $seed = 3$

instance	BKS	I3CH (original)			I3CH						instance	BKS	I3CH (original)			I3CH					
		profit	G (%)	t (s)	profit	G (%)	t (s)	RR	LS	SA			profit	G (%)	t (s)	profit	G (%)	t (s)	RR	LS	SA
c101	810	810	0.00	303.1	810	0.00	176	73	51	50	c201	1810	1810	0.00	3.8	1810	0.00	9	0	0	0
c102	920	920	0.00	237.5	920	0.00	344	181	81	81	c202	1810	1810	0.00	7.2	1810	0.00	30	0	1	1
c103	980	990	-1.02	156.5	990	-1.02	252	80	83	88	c203	1810	1810	0.00	7.9	1810	0.00	48	0	1	1
c104	1030	1030	0.00	155.9	1030	0.00	286	78	104	102	c204	1810	1810	0.00	11.2	1810	0.00	89	0	3	2
c105	870	870	0.00	110.2	870	0.00	187	67	58	61	c205	1810	1810	0.00	12.6	1810	0.00	23	0	0	0
c106	870	870	0.00	227.4	870	0.00	210	75	67	67	c206	1810	1810	0.00	18.0	1810	0.00	38	0	1	1
c107	910	910	0.00	151.8	910	0.00	334	197	68	69	c207	1810	1810	0.00	16.9	1810	0.00	46	0	1	1
c108	920	920	0.00	212.6	920	0.00	209	73	66	70	c208	1810	1810	0.00	20.6	1810	0.00	57	0	2	1
c109	970	960	1.03	157.2	970	0.00	235	80	74	79											
r101	484	481	0.62	67.7	484	0.00	125	57	34	34	r201	1441	1439	0.14	981.2	1441	0.00	2162	161	979	1011
r102	694	691	0.43	111.4	694	0.00	178	60	58	59	r202	1458	1458	0.00	15.3	1458	0.00	34	1	3	3
r103	747	740	0.94	125.6	747	0.00	226	84	69	72	r203	1458	1458	0.00	0.2	1458	0.00	65	0	2	3
r104	777	777	0.00	128.3	**778**	-0.13	257	87	83	86	r204	1458	1458	0.00	0.2	1458	0.00	166	0	8	8
r105	620	619	0.16	165.8	620	0.00	146	61	43	43	r205	1458	1458	0.00	0.3	1458	0.00	25	0	1	1
r106	729	729	0.00	122.7	729	0.00	191	74	57	59	r206	1458	1458	0.00	0.3	1458	0.00	56	0	2	3
r107	760	759	0.13	120.5	760	0.00	223	75	73	75	r207	1458	1458	0.00	0.3	1458	0.00	102	0	5	4
r108	797	797	0.00	135.7	797	0.00	251	88	80	82	r208	1458	1458	0.00	0.2	1458	0.00	177	0	8	10
r109	710	710	0.00	103.6	710	0.00	181	69	56	56	r209	1458	1458	0.00	0.3	1458	0.00	34	0	1	1
r110	737	736	0.14	109.9	736	0.14	213	87	62	63	r210	1458	1458	0.00	0.2	1458	0.00	43	0	2	2
r111	774	773	0.13	118.1	774	0.00	228	78	74	76	r211	1458	1458	0.00	0.4	1458	0.00	56	0	3	3
r112	776	776	0.00	110.5	776	0.00	235	85	73	76											
rc101	621	621	0.00	87.6	621	0.00	136	56	41	39	rc201	1698	1693	0.29	575.2	1692	0.35	1327	131	586	604
rc102	714	714	0.00	105.4	711	0.42	164	66	49	49	rc202	1724	1724	0.00	1.1	1724	0.00	47	3	14	13
rc103	764	764	0.00	105.7	747	2.23	199	80	59	60	rc203	1724	1724	0.00	0.3	1724	0.00	43	0	2	2
rc104	833	834	-0.12	124.4	**835**	-0.24	213	63	74	75	rc204	1724	1724	0.00	0.3	1724	0.00	102	0	5	3
rc105	682	682	0.00	90.4	682	0.00	152	68	43	41	rc205	1709	1719	-0.59	734.4	1719	-0.59	2490	229	1112	1138
rc106	706	706	0.00	88.1	706	0.00	142	61	40	40	rc206	1724	1724	0.00	0.5	1724	0.00	16	0	1	1
rc107	773	762	1.42	98.7	773	0.00	177	74	50	52	rc207	1724	1724	0.00	0.3	1724	0.00	24	0	1	1
rc108	795	789	0.75	107.4	795	0.00	219	99	59	61	rc208	1724	1724	0.00	0.4	1724	0.00	41	0	1	1

Table A9: Comparision I3CH for $m = 4$ and $seed = 3$

instance	BKS	I3CH (original)			I3CH						instance	BKS	I3CH (original)			I3CH					
		profit	G (%)	t (s)	profit	G (%)	t (s)	RR	LS	SA			profit	G (%)	t (s)	profit	G (%)	t (s)	RR	LS	SA
c101	1020	1020	0.00	176.4	1020	0.00	258	78	89	89	c201	1810	1810	0.00	0.1	1810	0.00	14	0	0	0
c102	1150	1150	0.00	274.1	1150	0.00	360	77	137	144	c202	1810	1810	0.00	0.1	1810	0.00	38	0	1	1
c103	1200	1210	-0.83	301.1	1210	-0.83	447	87	175	183	c203	1810	1810	0.00	0.2	1810	0.00	67	0	2	1
c104	1260	1260	0.00	281.8	1260	0.00	514	81	205	226	c204	1810	1810	0.00	0.1	1810	0.00	116	0	4	2
c105	1070	1060	0.93	333.2	1060	0.93	382	172	103	106	c205	1810	1810	0.00	0.1	1810	0.00	28	0	1	1
c106	1080	1080	0.00	179.4	1080	0.00	285	77	100	106	c206	1810	1810	0.00	0.1	1810	0.00	48	0	1	1
c107	1120	1120	0.00	209.3	1120	0.00	432	200	114	117	c207	1810	1810	0.00	0.1	1810	0.00	52	0	1	2
c108	1130	1130	0.00	373.1	1130	0.00	444	202	116	125	c208	1810	1810	0.00	0.1	1810	0.00	63	0	2	2
c109	1190	1190	0.00	227.6	1190	0.00	378	81	145	150											
r101	611	608	0.49	93.5	611	0.00	158	72	42	44	r201	1458	1458	0.00	0.2	1458	0.00	16	0	1	1
r102	843	837	0.71	162.6	843	0.00	264	77	93	93	r202	1458	1458	0.00	0.2	1458	0.00	43	0	2	2
r103	926	928	-0.22	206.9	928	-0.22	323	97	109	114	r203	1458	1458	0.00	0.2	1458	0.00	86	0	5	5
r104	972	969	0.31	209.1	**975**	-0.31	383	102	137	142	r204	1458	1458	0.00	0.2	1458	0.00	187	0	9	9
r105	778	778	0.00	142.4	778	0.00	219	85	66	68	r205	1458	1458	0.00	0.2	1458	0.00	37	0	1	2
r106	905	906	-0.11	185.9	906	-0.11	297	83	106	108	r206	1458	1458	0.00	0.1	1458	0.00	65	0	3	3
r107	945	950	-0.53	221.4	950	-0.53	367	90	136	139	r207	1458	1458	0.00	0.2	1458	0.00	112	0	6	8
r108	994	994	0.00	220.0	**995**	-0.10	369	81	140	146	r208	1458	1458	0.00	0.2	1458	0.00	192	0	14	14
r109	885	885	0.00	145.8	885	0.00	252	84	83	84	r209	1458	1458	0.00	0.2	1458	0.00	46	0	2	2
r110	914	915	-0.11	168.6	**915**	-0.11	296	85	104	106	r210	1458	1458	0.00	0.2	1458	0.00	56	0	3	2
r111	949	952	-0.32	203.5	951	-0.21	331	84	121	125	r211	1458	1458	0.00	0.2	1458	0.00	66	0	4	4
r112	971	967	0.41	251.8	**974**	-0.31	359	89	134	135											
rc101	811	808	0.37	119.7	811	0.00	188	75	57	56	rc201	1724	1724	0.00	0.2	1724	0.00	14	0	1	1
rc102	908	899	0.99	136.7	**909**	-0.11	253	73	90	89	rc202	1724	1724	0.00	0.2	1724	0.00	32	0	1	2
rc103	970	974	-0.41	170.2	974	-0.41	302	89	104	108	rc203	1724	1724	0.00	0.2	1724	0.00	58	0	2	3
rc104	1059	1064	-0.47	194.1	**1065**	-0.57	379	85	144	148	rc204	1724	1724	0.00	0.2	1724	0.00	127	0	4	6
rc105	875	875	0.00	127.0	875	0.00	224	80	71	71	rc205	1724	1724	0.00	0.2	1724	0.00	20	0	1	1
rc106	909	909	0.00	143.0	909	0.00	254	87	82	84	rc206	1724	1724	0.00	0.2	1724	0.00	27	0	1	1
rc107	980	980	0.00	155.4	**987**	-0.71	280	86	94	100	rc207	1724	1724	0.00	0.2	1724	0.00	39	0	2	2
rc108	1025	1020	0.49	173.2	1025	0.00	331	89	118	123	rc208	1724	1724	0.00	0.2	1724	0.00	53	0	3	3

C Comparison I3CH on OPHS-TW

The names of the instance sets are structured as follows: number of locations - total trip length - additional hotels - number of routes.

Table A10: Comparion I3CH and Divsalar et al. (2014)

			I3CH			Divsalar et al. (2014)		
m	instance	OPT	profit	gap (%)	t (s)	profit	gap (%)	t (s)
2	64-45-1-2	816	816	0.00	146	816	0.00	0.6
	64-50-1-2	900	876	2.67	174	876	2.67	1.0
	64-55-1-2	984	960	2.44	201	960	2.44	1.3
	64-60-1-2	1062	1062	0.00	247	1062	0.00	2.1
	64-65-1-2	1116	1116	0.00	276	936	16.13	1.7
	64-70-1-2	1188	1170	1.52	345	1152	3.03	1.9
	64-75-1-2	1236	1224	0.97	389	1218	1.46	1.6
	64-80-1-2	1284	1284	0.00	403	1272	0.93	1.7
	66-40-1-2	575	570	0.87	104	570	0.87	0.5
	66-45-1-2	650	645	0.77	119	600	7.69	0.7
	66-50-1-2	730	715	2.05	121	705	3.42	0.9
	66-55-1-2	825	825	0.00	138	825	0.00	0.9
	66-60-1-2	915	910	0.55	137	910	0.55	1.1
	66-130-1-2	1680	1675	0.30	671	1655	1.49	2.2
	100-30-1-2	173	160	7.51	82	160	7.51	0.1
	100-35-1-2	241	241	0.00	97	241	0.00	0.2
	100-40-1-2	299	299	0.00	105	216	27.76	0.4
	100-45-1-2	367	367	0.00	133	367	0.00	0.4
	102-50-1-2	181	181	0.00	78	181	0.00	0.1
	102-60-1-2	243	243	0.00	98	243	0.00	0.1
3	64-45-2-3	816	816	0.00	148	816	0.00	0.5
	64-50-2-3	900	870	3.33	152	870	3.33	0.6
	64-55-2-3	984	960	2.44	162	936	4.88	0.7
	64-60-2-3	1062	864	18.64	169	1062	0.00	1.1
	64-65-2-3	1116	1116	0.00	263	1116	0.00	1.5
	64-75-2-3	1236	1218	1.46	328	1218	1.46	1.2
	64-80-2-3	1284	1284	0.00	338	1128	12.15	1.5
	66-40-2-3	575	320	44.35	85	570	0.87	0.2
	66-45-2-3	650	645	0.77	108	645	0.77	0.2
	66-50-2-3	730	330	54.79	92	715	2.05	0.4
	66-55-2-3	825	410	50.30	96	825	0.00	0.6
	66-60-2-3	915	910	0.55	127	910	0.55	0.7
	66-125-2-3	1670	1525	8.68	338	1665	0.30	1.9
	100-30-2-3	173	173	0.00	75	173	0.00	0.1
	100-35-2-3	241	241	0.00	93	241	0.00	0.1
	102-50-2-3	181	181	0.00	74	181	0.00	0.1
	102-60-2-3	243	243	0.00	100	243	0.00	0.1
4	64-50-3-4	900	666	26.00	113	858	4.67	0.5
	64-55-3-4	984	774	21.34	144	954	3.05	0.6
	64-60-3-4	1062	852	19.77	165	1062	0.00	0.7
	64-65-3-4	1116	996	10.75	193	1116	0.00	0.8
	64-75-3-4	1236	1086	12.14	222	1194	3.40	1.0
	66-40-3-4	575	570	0.87	94	570	0.87	0.2
	66-45-3-4	650	645	0.77	104	645	0.77	0.3
	66-55-3-4	825	455	44.85	93	825	0.00	0.3
	100-30-3-4	173	160	7.51	71	160	7.51	0.1
	100-35-3-4	241	241	0.00	88	241	0.00	0.1