Implementation of the Iterative Three-Component Heuristic for the Team Orienteering Problem With Time Windows

Author
Natasja Sluijk

Supervisor
Thomas R. Visser, MSc.

Second accessor
Remy Spliet, Dr.

Bachelor Thesis
Econometrics and Operations Research

Erasmus School of Economics
Erasmus University Rotterdam

July 1, 2016

Abstract

The purpose of this thesis is to replicate the Iterative Three-Component Heuristic posed by Hu and Lim [13]. This heuristic is applied to the Team Orienteering Problem with Time Windows in which the aim is to maximize the total profit collected by servicing a set of customers with a limited number of vehicles. The first two components in the heuristic are local search and simulated annealing. They explore neighbourhood solutions and discover new sets of routes. The last component recombines the routes that have been found and tries to find the best solution, namely the one yielding the highest profit. Our computation results show that the computation time is significantly larger than the computation time given in [13]. Moreover, we obtained different and often worse results than reported in [13]. One reason for our replication performing worse is the decrease of the maximum number of iterations (140 instead of 3000) in order to keep the computation time to a limit. We did find new best solutions for *c108* (1140) and *rc102* (909) for $m = 4$ vehicles. We extended the original heuristic to include multi-threading which performed 1.6 times faster than the original heuristic. Furthermore, we investigated increased route pool sizes, which lead to better results at the cost of only a small increase in the computation time.

Contents

1 Introduction **3**

2 Literature Review **4**

3 Problem Definition **5**

4 Methodology **6**
4.1 Neighbourhood Operators 6
4.2 Route Pool 10
4.3 Local Search 10
4.4 Simulated Annealing 10
4.5 Route Recombination 11
4.6 Iterative Framework 11
4.7 Extensions 12
4.7.1 Multi-Threading 12
4.7.2 Size of POOL 13

5 Data Description **14**

6 Results **15**
6.1 Extensions 18
6.1.1 Multi-Thread 18
6.1.2 Size of POOL 20

7 New Best Known Solutions **21**

8 Conclusion **22**

9 Appendix **23**
9.1 Algorithms 23
9.2 Detailed results of our implementation of I3CH 25
9.3 Detailed results of using Multi-Threading (Extension) 33
9.4 Detailed results of changing the POOL size (Extension) 35
9.5 Routes of the new Best Known Solutions 44

10 References **45**

1 Introduction

This thesis studies the Team Orienteering Problem with Time Windows (TOPTW). The set up of this problem can be explained with the help of a game called *orienteering* that is often played in areas with rough landscapes. Given the location of check points, players will try to visit as many of them as possible. Each check point has its own score and the aim is to maximize the total score. Due to a limitation on the time, players are not always able to visit all check points and therefore have to select a subset of locations that will result in the highest total score under the restriction of time. Not arriving at the finish point on time results in disqualification [25].

Several extensions on the Orienteering Problem (OP) have been posed and studied in literature. One extension is to consider the game to be played by teams instead of individuals (Team Orienteering Problem, TOP). Another well studied extension is the Orienteering Problem with Time Windows (OPTW). In this setting, the team consists of one member and each location can only be visited within a specified time window.

The Team Orienteering Problem with Time Windows (TOPTW) combines the TOP and OPTW. In the TOPTW we typically use a different terminology. Instead of visiting checkpoints and collecting scores, customers (also referred to as locations) will be visited and profit will be collected. The aim of TOPTW is to maximize the total profit collected by visiting a set of locations. In this thesis, it is assumed that each location has a profit, a service time and a time window. The TOPTW can be modelled as a multi-level optimization problem. First, a subset of customers to visit has to be chosen. Then, the shortest, feasible path over this subset has to be found. Note that these two levels are dependent on each other. On one hand, it could be that the subset of customers is quite small and that there is space left in a tour for more customers. This situation can be solved by trying to add more customers to the final tour. On the other hand, it could be impossible to find a feasible tour due to the large size of the subset. In order to solve this situation, customers have to be removed from the subset.

The TOPTW can be applied to many planning problems. One example is the personalized electronic tourist guide that is used by tourists to assist them in the planning of their trips [26]. A second example is the salesman that sells his products from door-to-door. Suppose he has works from Monday till Friday and he has one week to visit the customers in a city. He will then construct five routes (one for each day) such that he maximizes profit. Because of the time restriction, he is not able to visit all customers and therefore prefers to visit the customers with high profit. Besides taking into account his own time restriction, he also has to take into account the time windows of his customers. It could be that some of his customers are only home on Monday morning, while others can only be reached on Friday. This example can be seen as an TOPTW in which the salesman is his own "team member" and the team consists of five members. His time window contains all weekdays, but it split up in five smaller parts, one for each day and thus for each "team member".

The purpose of this thesis is to replicate the results of the Iterative Three-Component Heuristic developed by Hu and Lim [13]. The first two components explore the solution space with help of local search and simulated annealing, and discover a set of routes. The third component recombines the routes and finds the best combination with the help of an Integer Programming problem. Besides replicating the heuristic, two extensions will be considered. The first extension is the use of multi-threading in the code. Local search and simulated annealing can be seen as two independent procedures and could therefore run in parallel. This will reduce the computation time and thus make the heuristic faster. The second extension is the investigation of the trade-off between computation time and solution quality with respect to the size of the POOL. In this extension, multi-threading

is used in order to obtain results within a reasonable amount of time.

The remainder of this paper is organized as follows. In the next section, a literature overview on TOPTW is given. In Section 3, the problem description is given, and in Section 4 the methodology is explained. In Section 5, a description of the data is given and in Section 6 the results are presented. We found new best solution and represent those in Section 7. This thesis is rounded off with a conclusion in Section 8.

2 Literature Review

Among researchers that have studied the OP are Chao et al. [1], Schilde et al. [22], and Fischetti et al. [7]. Both Golden et al. [10] and Laporte and Martello [17] proved that the OP is a NP-hard problem. This implies that only for small instances an exact solution can be found within a reasonable amount of time. As has been shown in Section 1, many extensions exist and they are considered as NP-hard problems as well. A complete overview of all currently studied extensions can be found in the paper of Gavalas et al. [9].

One possible extension is the Team Orienteering Problem (TOP) where multiple tours are considered instead of one [2]. In this problem, rather than looking at only one subset of locations, the subset of locations of each vehicle has to be taken into account. Once one of the vehicles has visited a location, another vehicle cannot receive a profit from that location anymore. Vansteenwegen et al. [27] argue that including the time windows in the OP, as is done in the OPTW, significantly affects the nature of OP and its algorithmic approaches. For instance, reordering locations within a route does not guarantee feasibility due to those time window restrictions. Other papers that discuss the OPTW include Kantor and Rosenwein [14] and Righini and Salani [21].

Vansteenwegen et al. [26] introduced the TOP with Time Windows (TOPTW). Since TOPTW is considered to be NP-hard, most TOPTW literature is focused on developing heuristics that obtain a good solution quality within a reasonable amount of time. One heuristic is the ant colony system (ACO) that resulted in, at that time, the best results on the OPTW instances [19]. The ACO algorithm is a metaheuristic that is based on real ants. Ants leave their nest to find food sources and while walking they spread a pheromone so that they can find their way back. This pheromone is also important for routes of other ants, since they will choose their path with a certain probability and this probability depends on the amount of pheromone. The more pheromone on a path, the higher the probability of that path will be. In the algorithm, artificial ants are created that try to find shortest path. Each individual ant tries to find the shortest path, and meanwhile shares his information with other ants. Hence, they corporate. Further development of this algorithm by these authors lead even to new best solutions [8, 20].

Vansteenwegen et al. [26] posed an Iterated Local Search (ILS) heuristic that is currently the fastest known algorithm posed to produce solutions of reasonable quality for TOPTW [9]. They obtained neighbourhood solutions of the starting solution by either inserting new customers to the route or by deleting customers from the route. Tricoire et al. [24] developed an algorithm that deals with a more complex version of the TOPTW, namely the Multi-Period Orienteering Problem with Multiple Time Windows. Their heuristic contains a local search algorithm which is based on the Variable Neighbourhood Search (VNS). The idea of VNS is a systematic change of neighbourhoods within a local search procedure. It explores neighbourhood solutions of increasing sizes. The initial solution is determined by finding a solution in the set of neighbourhood structures. From this incumbent solution, random points are generated on which neighbourhoods searches as shaking

and iterative improvement are applied.

Another heuristic posed for the TOPTW is the Greedy Randomized Adaptive Search Procedure with Evolutionary Local Search (GRASP-ELS) [15]. GRASP-ELS gives considerably better quality solutions than ILS, but at the expense of increased computation time. Whereas ILS starts with an initial solution s obtained by a heuristic, generates many neighbourhood by changing s and finally improve these solutions by local search, ELS generates multiple copies of the starting solution s and then applies ILS on each of these copies. The authors also posed a heuristic called Granular Variable Neighbourhood Search (GVNS) which lead to new results [16]. The difference between GVNS and VNS is that the granular variant aims at reducing the size of the analysed neighbourhoods by excluding non promising arcs during the node sequence construction in the local search procedure.

Lin and Vincent [18] developed two versions of a simulated annealing algorithm. The fast version (FSA) computes a solution within only several seconds, while the Slow Simulated Annealing (SSA) requires more computation time, but gives better solutions. These calculations are performed on the benchmark instances that are also used in this thesis and will be defined later. Some of the current best-known solutions on these benchmark instances are obtained by the SSA. Hu and Lim [13] posed an Iterative Three-Component Heuristic (I3CH) for solving the TOPTW. Their heuristic found 35 new best solutions on benchmark instances for which the optimal solution is unknown and achieved 55 optimal solutions on the instances for which the optimal solution is known, which is sixteen more than the previous best approach. This occurs, however, at the expense of longer computation times. Cura [5] posed a relatively new technique called artificial bee colony (ABC) approach to solve the TOPTW. This heuristic simulates the foraging behaviour of learning, memorizing and information sharing of honey-bee swarms.

El-Hajj et al. [6] used a column generation algorithm to solve the TOPTW exactly. The authors were able to prove the optimality of several instances by finding their integer solutions at the root node while solving the linear relaxation of the model. Finally, Gunawan et al. [12] recently posed a hybridization of Simulated Annealing and Iterated Local Search (SAILS). The main difference between SA and SAILS is that an additional strategy of intensification is included. If there is no improvement of the solution after a certain number of iterations, then the search is focussed once again starting from the best solution obtained so far. In the usual SA, if the upper limit on the number of consecutive iterations without improvement is reached, the SA calculations is done. SAILS differs from SA in that it does not stop when the upper limit on iterations is reached, but continues by taking the best solution, again applying SA to it, and it will only stop when the time limit is met.

3 Problem Definition

The TOPTW is formulated on the network $G = (V, A)$, where there are $n + 1$ different vertices denoted by the set $V = \{0, 1, 2, ..., n\}$ and a set of arcs that connects these locations $A = \{(i, j) : i \neq j \in V\}$. Location 0 is the depot and each of the remaining vertices correspond to one customer. The travel duration t_{ij} from location i to j is equal to the Euclidean distance between these two locations. Only a limited amount of vehicles m is available. Each vehicle must begin and end its route at the depot within the time window of the depot $[O_0, C_0]$. For each customer c_i, where $i = 1, ..., n$, the profit p_i, service time T_i, and time window $[O_i, C_i]$ is known. Customers can be visited at most once. A service is successfully delivered to a customer if it begins within his or her time window. If there is an early arrival, the vehicle has to wait until the opening of the time

window. Profit is collected from the successful services. Due to the limited amount of vehicles, some customers may not be serviced in the feasible solution. The objective of this thesis is to find a feasible combination of routes that yields maximum profit. A MIP formulation for this problem is given in [26].

4 Methodology

The Iterative Three-Component Heuristic (I3CH) consists of the following components: Local Search (LS), Simulated Annealing (SA), and Route Recombination (RR). In general, both LS and SA will make use of neighbourhood search. The solutions of LS and SA are used for RR and by iteratively solving, the best solution of RR, LS, and SA is then used as inputs for the LS and SA.

4.1 Neighbourhood Operators

The neighbourhoods are created with the help of the neighbourhood operator *eliminator* which removes some customers from the routes and replaces them by unvisited customers that are stored in a list called u. If those added customers are more profitable than the ones that were placed in the route before, the solution quality is improved. The elimination of the customers occurs randomly. Let $\bar{\pi}$ be the average profit over all customers on m routes that are given as a starting solution to the *eliminator*. Customer c_j is then eliminated with probability P_h if $\pi_j \geq \bar{\pi}$ and with probability P_l if $\pi_j < \bar{\pi}$. Assuming that there is a preference of keeping customers with high profit, $P_h < P_l$. Both P_h and P_l are set in advance and remain fixed throughout the heuristic. After the *eliminator* has removed customers from the routes, it will improve the solution by adding customers from the head of u to any route. If a customer cannot be added to a route, he or she will be placed at the end of u. Due to the stochastic element in this procedure, many neighbours can be created with the same starting solution.

Next, the *post-processing* (PP) procedure is applied to improve the solution that the *eliminator* returns. The PP consists of seven different operators: the *relocate* operators (3), the *exchange* operators (3), and a *2-opt* operator. Relocating a customer is done by the *relocate* operators and implies replacing customers from the list of unvisited customers to a feasible insertion position into a route (*0-relocate*), replacing a customer to a different position of the route he was currently in (*1-relocate*), and replacing a customer from one route to another route (*2-relocate*). The *exchange* operators that are used are *0-exchange*, *1-exchange*, and *2-exchange*. *0-exchange* replaces a customer on a route with a customer from u, *1-exchange* interchanges two customers on the same route, whereas *2-exchange* interchanges two customers from different routes. The *2-opt* operator deletes two edges on two different routes (one on each) resulting in four separated routes. Those routes are then recombined, if possible and profitable, in a different way such that two new feasible routes are created and the total travel distance is reduced. Note that only *0-relocate* and *0-exchange* can lead to a improvement of the solution quality with respect to profit, while the other five operators cannot. However, they can improve the solution quality with respect to distance. The seven operators are illustrated in Figure 1. A green circle indicates a customer that was initially in no route, but is included in a route in the new situation. A red circle indicates a customer who got removed from a route. Finally, a blue circle indicates a customer who has changed from position within or between routes. The order in which the operators are invoked is given in Algorithm 1.

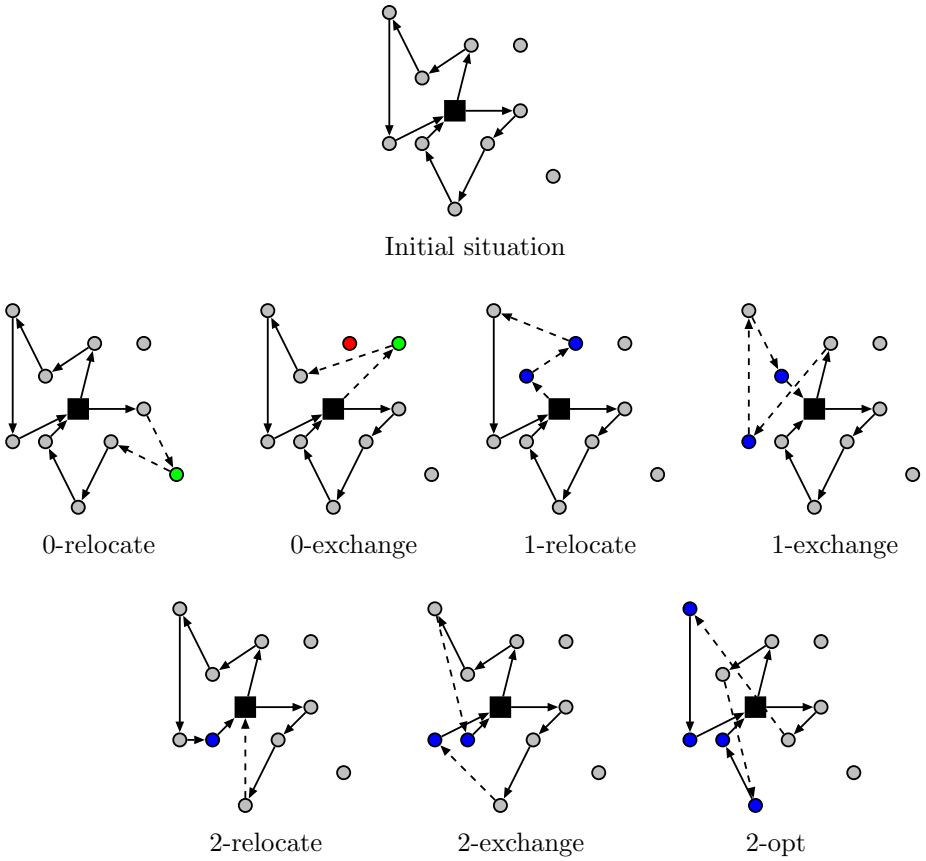

Figure 1: Illustration of the seven operators in the Post-Processing procedure

Algorithm 1 Post-Processing procedure

Input: starting solution S
Set impr $\leftarrow$ `true`
while impr is `true` **do**
    Call *2-relocate*, *2-exchange*, and *2-opt* on S and obtain S'
    Call *1-relocate* and *1-exchange* on S' and obtain S''
    Call *0-relocate* and *0-exchange* on S'' and obtain S'''
    if $S''' > S$ **then**
        impr $\leftarrow$ `true`
        $S \leftarrow S'''$
    else
        impr $\leftarrow$ `false`
    end if
end while
Return: S

One of the main requirements is that the heuristic should be fast. A reason for this is that it might be used by a company who wants to be able to determine the route for each day given

the time windows of the customers of those days. The company is certainly not willing to wait hours for a solution and rather prefers to obtain a solution within seconds since the operation of the company depends on it. One way to achieve this is to provide possibilities for quick evaluations of a possible move of an operator. Checking each visit on their feasibility would require much time. This can be avoided by keeping track of a few important variables. For this, we used the method posed by Vansteenwegen et al. [26]. $Wait_i$ keeps track of the waiting time for location i if the arrival at location i with arrival time $Arrival_i$ takes place before the time window of that location ($Opening_i$) opens. If the arrival occurs during the time window, $Wait_i$ will be equal to zero (Equation (1)).

$$Wait_i = \max[0, Opening_i - Arrival_i] \tag{1}$$

Another important variable is $MaxDelay_i$ that is defined as the maximum amount of time the start of a service can be delayed, without making any visit infeasible. It is equal to the sum of the waiting time and the maximum delay of the next location, unless it is limited by the difference between the closing of its window ($Closing_i$) and the start of service of customer ($Start_i$) (Equations (2) and (3)).

$$\begin{aligned} MaxDelay_i &= \min[Closing_i - Start_i, Wait_{i+1} + MaxDelay_{i+1}] & (2)\\ Start_i &= \max[Opening_i, Arrival_i] & (3) \end{aligned}$$

By recording the $MaxDelay_i$, the time it takes to evaluate a possible move is reduced to constant instead of linear time. When a customer is considered to be inserted in the route, its corresponding $Shift_j$ is calculated (Equation (4)). This variable indicates the time it would take to include this customer in the route.

$$Shift_j = t_{ij} + Wait_i + Duration_j + t_{jk} - t_{ik} \tag{4}$$

In the above equation, t_{ij} indicates the travel time from customer i to customer j which is equal to the Euclidean distance between two locations. $Duration_j$ equals the service duration of customer j.

There are two conditions that need to be satisfied for an insertion of customer j between customers i and k to be feasible. First of all, the time consumption of inserting customer j should be smaller than or equal to the waiting time and maximum delay of customer k (Equation (5)). The second condition is that the start of service of customer j should be before his time window closes (Equation (6)).

$$\begin{aligned} Shift_j &\leq Wait_k + MaxDelay_k & (5)\\ Start_j &\leq Closing_j & (6) \end{aligned}$$

When looking for a feasible insertion point, the first feasible insertion position is chosen since considering all feasible insertion positions and taking the best one would be time consuming. Inserting customer j into a route means that all variables of the customers after customer j and the maximum delay of the customers before j have to be updated. The variables of the visits after location j are updated with the formulas given in Equations (7) - (11).

$$
\begin{aligned}
Arrival_{k^*} &= Arrival_k + Shift_j && (7)\\
Start_{k^*} &= Start_k + Shift_j && (8)\\
Wait_{k^*} &= \max[0, Wait_k - Shift_j] && (9)\\
Shift_{k*} &= \max[0, Shift_j - Wait_k] && (10)\\
MaxDelay_{k^*} &= MaxDelay_k - Shift_k && (11)
\end{aligned}
$$

A pseudocode for updating these variables is given in Algorithm 2. Note that we leave the for-loop once we have found a possible insertion position for a customer since we will choose the first feasible insertion position.

Algorithm 2 Insertion of a customer

Input: Tour for which it will be checked whether a customer from u can be added, u
for each customer j in u **do**
    Insertion $\leftarrow$ `false`
    for each possible insertion position **do**
        if Insertion equals `true` **then**
            break
        end if
        Determine $Shift_j$
        if $Shift_j \leq Wait_k + maxDelay_k$ and $Start_j \leq Closing_j$ **then** ▷ Insertion is possible
            Insertion $\leftarrow$ `true`
            Calculate $Arrival$, $Start$ and $Wait$ of location j
            for each visit after j until $Shift == 0$ **do**
                Update $Arrival$, $Start$, $Wait$, $Shift$, and $MaxDelay$;
            end for
            For location j: Update $MaxDelay$
            for each visit before j **do**
                Update $MaxDelay$
            end for
        end if
    end for
end for

Next to inserting customers, customers will also be deleted. When a customer is deleted, the values of the variables of the other customers in the route have to be updated, which is done according to Algorithm 3.

Algorithm 3 Removal of a customer

1: **Input:** Tour from which the customer will be removed, removal position
2: Calculate shift (the amount of extra time)
3: **for** each visit after the removal position **do**
4: Update *Arrival*, *Start*, *Wait*, *Shift*, and *MaxDelay*
5: **end for**
6: **for** each visit before the removal position **do**
7: Update *MaxDelay*
8: **end for**

4.2 Route Pool

Throughout the heuristic, many routes are created and saved into a list called $POOL$. RR will use this list. Since the set containing all possible routes has exponential elements, the size of $POOL$ will become large. Having a large set of routes will affect the computation time of RR negatively. Hu and Lim [13] therefore decided to put an upper bound of 1,000 on the size of $POOL$. This implies that routes can be added without any problem if the size of $POOL$ is smaller than 1,000. When $POOL$ is full and new routes have to be added, old routes should be removed to make space for the newly discovered routes. On one hand, not having all possible routes in $POOL$ implies that RR might not obtain the optimal combination of routes. On the other hand, the computation time should be limited as well. Hence, a trade off has been made. Each route has an *apr* value that indicates how often and how recent the route has been used in a solution of RR. When a route is added to $POOL$, its *apr* value is initialized with 0. If a route is contained in the solution of RR, its *apr* value will increase to 100, while the *apr* values of all other routes that are in $POOL$ but not incorporated in the solution will decrease by one. When $POOL$ is full and new routes have to be added, the concept of Least Recently Used (LRU) is applied and the routes with the lowest *apr* values will be removed. If multiple routes have the same *apr*, the one with the lowest profit will be removed. If a tie occurs again, then the route with the largest distance will be removed. If there is still a tie, one of the considered routes is deleted arbitrarily.

4.3 Local Search

Local Search iteratively generates N neighbourhoods of the starting solution S with the help of the *eliminator* and PP. S will be updated with the best neighbourhood solution Y if this solution is strictly better than S. To avoid ending up in an infinite loop, the variable $I_{LS\_no\_impr}$ keeps track of the number of consecutive iterations without improvement. If an improvement has been made, $I_{LS\_no\_impr}$ is set equal to zero. In case of no improvement, $I_{LS\_no\_impr}$ is increased by one. The corresponding algorithm can be found in Appendix 9.1.

4.4 Simulated Annealing

Simulated Annealing is an alternative of local search with as main difference that it sometimes accept worse solutions in order to escape local optima. SA requires an initial temperature T_0 and a cooling speed α as parameters and $I_{SA\_no\_impr}$ as a variable. Similar to $I_{LS\_no\_impr}$, $I_{SA\_no\_impr}$ keeps track of the number of consecutive iterations without improvement. In each step, one neighbourhood solution Y' of the neighbourhood of Y is generated, where Y is initialized with the routes

from the starting solution S. If Y' is better than Y, Y is set to Y'. If not, SA will accept this neighbourhood with probability P_{SA} as given in Equation 12.

$$P_{SA} = e^{\frac{1}{T}\frac{Y'-Y}{Y_{SA}}} \tag{12}$$

In this equation, Y_{SA} equals the best neighbourhood solution found over all iterations. Moving from Y to Y' is called a step. When a step has been made, the temperature is updated to $T \leftarrow \alpha T$. After N steps have been made, thus, N neighbourhoods have been visited, Y is compared to Y_{SA}. When Y is better than Y_{SA}, Y_{SA} is set equal to Y and $I_{SA\_no\_impr}$ is set equal to zero. Otherwise, $I_{SA\_no\_impr}$ is increased by one. The corresponding algorithm can be found in Appendix 9.1.

4.5 Route Recombination

Route Recombination is the third component of the heuristic and is used to find the best combination of routes in $POOL$ such that profit is maximized and the constraints are not violated. The first set of constraints (14) makes sure that each customer is included at most once in the combination of routes. The second set of constraints (15) gives an upper bound on the number of routes that can be chosen. As can be observed from the formulation, in the last set of constraints (15) an inequality sign is used instead of an equality sign, since it could be that the optimal profit is already obtained with less vehicles than m, the number of vehicles available. In this formulation S_{pool} indicates the size of $POOL$. If customer j is included in route k, a_{jk} Equals one. The total profit of route p_k is calculated as the sum of profits of all customers in route k. Let variable x_k be equal to one if route k is selected and zero otherwise. The formulation is classified as a set packing formulation since not all possible routes are considered. Instead, only the routes that are saved in $POOL$ are considered.

$$\max \sum_{k=1}^{S_{pool}} q_k x_k \tag{13}$$

$$\text{s.t.} \sum_{k=1}^{S_{pool}} a_{jk} x_k \leq 1 \qquad \forall j \in V \setminus \{0\}, \tag{14}$$

$$\sum_{k=1}^{S_{pool}} x_k \leq m, \tag{15}$$

$$x_k \in \{0,1\} \qquad \forall k \in \{1,2,...,S_{pool}\}. \tag{16}$$

4.6 Iterative Framework

LS, SA, and RR are placed into an iterative framework (I3CH) to ensure better cooperation between them. First, $3N$ solutions are generated with the help of the *eliminator* by initializing m empty routes. All customers are stored in a random order into the list of unvisited customers u. Since there are no customers to be removed from the routes, the *eliminator* will only add customers from u to the routes until all customers have been considered. In order to improve the solutions found by the *eliminator*, the PP procedure is applied. From the $3N$ solutions, the best solution A is chosen as a starting solution for LS and SA.

Several parameters have been discussed already. Table 1 presents them once more with their corresponding values and interpretations. The Iterative Three-Component Heuristic is described in Algorithm 4.

Table 1: Parameters

Parameter	Value	Interpretation
P_h	0.1	Probability of removing a customer when its profit is higher than the average profit
P_l	0.3	Probability of removing a customer when its profit is higher than the average profit
T_0	0.1	Initial temperature
α	0.995	Cooling speed
S_{pool}	1000	Size of $POOL$
I_{max}	3000	Maximum number of iterations in the iterative framework
N	50	Number of neighbourhoods to consider
I_{no_impr}	20	Maximum number of iterations without improvement

Algorithm 4 Iterative Three-Component Heuristic

Input: P_h, P_l, T_0, α, S_{pool}, I_{max}, N, and I_{no_impr}
Generate $3N$ solutions and set the best one as initial solution A
Save the routes from A in $POOL$
Set $iteration \leftarrow 1$
while $iteration \leq I_{max}$ **do**
    Invoke Route Recombination over $POOL$ to obtain X_{RR}
    Apply Local Search to explore N neighbourhoods and obtain best solution X_{LS}. Save the newly discovered routes into $POOL$
    Apply Simulated Annealing with N steps to obtain best solution X_{SA} and also save the routes into $POOL$
    Select the best solution B from $\{A, X_{RR}, X_{LS}, X_{SA}\}$. In case of a tie, break it arbitrarily
    $A \leftarrow B$, $iteration \leftarrow iteration + 1$
    if all customers are served in A **then**
        $iteration \leftarrow I_{max} + 1$
    end if
end while
solution of A;

4.7 Extensions

4.7.1 Multi-Threading

One way to decrease the computation time of a computer program is to run computations parallel. In this extension, the use of Multi-Threads in Java is explored and included. One requirement for

components to run parallel is that they are independent of each other. In the I3CH, there are two components that can be constructed in such a way this requirement is met. These two components are Local Search and Simulated Annealing. They both make use of the starting solution A, which is given as an input and then explore neighbourhood solutions. In the original program, both components do, however, not run completely independent, but the heuristic can be changed such that they will be independent.

The first adaptation needed is to change the saving procedures of routes into $POOL$. In the original program, routes are saved into $POOL$ while executing the component (LS/SA). In order to avoid concurrency problems here, the routes that should be added to $POOL$ will be saved in a list and added to the $POOL$ when both components have finished their calculations. Note that it is not certain whether it would be an issue to add them to the $POOL$ directly, but that this step is taken in order to avoid problems. After the calculation for both components has been done, both lists of routes generated by LS and SA will be added to $POOL$. In the original heuristic, LS is executed before SA and thus the routes found by LS will be added to the $POOL$ first. Due to the already existing checking algorithm, no route will be added twice.

The second adaptation needed is the change from one random generator to three random generators. In the original heuristic, the random generator is used for the initialization, LS and SA, since they all make use of the *Eliminator* that requires a random generator as input. However, when using multi-threading, it might be difficult to replicate results if only one random generator will be used since it could happen that one time LS is just a bit faster than SA, while the next time it is the other way around. This would lead to the drawing of different random numbers within a component and would result in different outcomes. Another reason for using different random generators is that if both threads would use the same random generator, concurrency could result in dread locking issues. If LS is using the random generator, but SA also wants to use it, SA has to wait until the random generator is not used by LS anymore and this affects the computation times. Instead, three random generators will be used, one for each component. Each of the random generators will have the same seed. Hence, for each thread a local random generator will be used.

The effects of using threads will be investigated on a subset of instances that will be chosen based on the results of the replication with $m = 4$. In choosing this subset, two criteria are considered. First, all instances for which the solution is obtained in one iteration (due to optimality) will be eliminated. Second, instances that require more than 10 minutes computation time will be eliminated in order to keep the computation time to a limit. Since selection of the instances for the subset does depend on their corresponding computation times and solution procedures, the instances that are selected for the subset will be further specified in Section 6.1.1.

In order to make a fair comparison of the possible speed-up of using threads, the original program has to be executed again due to the change in the use of random generators. Then the five runs for each instance will be executed using the program that incorporates the multi-threading.

4.7.2 Size of POOL

Here, the trade-off between the computation time and solution quality with respect to the size of $POOL$ is investigated. In this extension, the $POOL$ sizes will be investigated for the Multi-Thread heuristic in order to obtain results within reasonable time. Another way to ensure obtaining results within reasonable time is to consider fewer runs. Instead of taking an average over five runs, an average of three runs will be taken. In the paper of Hu and Lim [13] investigation on the size of $POOL$ has been done for sizes of the $POOL$ up to 2000. However, in this extension, we investigate

this trade-off more thoroughly by considering sizes from 0 to 5000. From the subset of instances that has been specified in Section 4.7.1 again a subset will be constructed such that the size of this subset is equal to 25. The four instances with the largest computation times, when using multi-threading, will be removed.

5 Data Description

The data that is used in this thesis is collected from two different sources. First, a group of instances are selected from the instances for the standard sets of routing problems as given in Christofides et al. [3]. This selection is chosen by Solomon [23]. Righini and Salani [21] used this selection to create OPTW instances, which then Montemanni and Gambardella [19] used to construct the TOPTW instances by increasing the number of vehicles: $m = 2, 3, 4$. The geographical data are randomly generated by a random uniform distribution (problem sets R1 and R2), clustered (problems sets C1 and C2), and semi-clustered (problem sets RC1 and RC2). A semi-clustered problem set contains a mix of clusters and randomly generated data. Problem sets R1, C1, and RC1 have a short scheduling horizon. In the setting of the TOPTW, the combination of customers that could be serviced by one vehicle is restricted by time windows and in this case allows only a few customers to be serviced by the same vehicle. In contrast, the sets R2, C2, and RC2 have a long scheduling horizon; this characteristic, coupled with large vehicle capacities, permits many customers to be serviced by the same vehicle.

Given the design method, problem sets C1 and C2 are composed of structured problems in the sense that the customers appear in clusters and the time windows are positioned around the arrival times at customers. This approach permits the identification of a very good, possibly optimal, cluster-by-cluster solution, which, in turn, provides an additional means of evaluating heuristic performance. The coordinates of the locations in the R*, C* and RC* sets are the same in both the 1-sets as the 2-set. The only difference between these two sets is the time window, since, as mentioned before, the 2-set has larger time windows.

Beside these instances, also instances of Cordeau et al. [4] are selected (PR). The number of vertices in these instances range from 48 to 288.

Each instance contains information on the X and Y coordinate of each vertex, and its corresponding service duration, profit, and time window.

In total, 76 instances are included in the data set and an overview of how they are divided over the sets can be found in Table 2.

Table 2: Instances

Instance Type	Name	# Instances	# vertices
C	c101 - c109	9	100
C	c201 - c208	8	100
R	r101 - r112	12	100
R	r201 - r211	11	100
RC	rc101 - rc108	8	100
RC	rc201 - rc208	8	100
PR	pr01 - pr10	10	different for each instance
PR	pr11 - pr20	10	different for each instance

6 Results

All computations were carried out on a Lenovo Ideapad 500S-14ISk laptop equiped with a Intel Core i5-6200U, 2.30GHz, and 8 GB RAM using Eclipse with Java edition 1.8.0. While executing the heuristic, we observed that our computation times were of larger order than those mentioned in [13]. This could be due to the use of a different machine and possible misinterpretation of some parts in the paper of Hu and Lim [13]. Another reason could be our memory usage. Unfortunately, we have little insight in how they used their memory and we are thus not able to compare our implementation with theirs. Steps have been taken to reduce the computation times.

One way to decrease the computation time is to decrease the maximum number of iterations. Hence, instead of allowing 3000 iterations, fewer iterations will be considered. Parameter analysis for the maximum number of iterations has been done to come up with an appropriate value. For this analysis, the same ten instances are considered as the ones that Hu and Lim [13] used for testing. They reasoned that these instances performed relatively worse in preliminary tests in which the parameters were set to preliminary values. The instances of interest are *c203*, *c207*, *pr02*, *pr07*, *pr12*, *pr16*, *r102*, *r105*, *rc107*, and *rc204*. In the parameter analysis, the others parameters do not change in value and only solutions containing four routes are considered.

Table 3: Parameter tuning on I_{max}

I_{max}	Average Gap (%)	Average Time
20	3.37	129.04
40	2.92	237.60
60	2.91	419.50
80	2.81	445.86
100	2.73	542.44
120	2.52	637.04
140	2.47	742.95
160	2.47	845.49
180	2.47	944.93
200	2.47	1050.25

The maximum number of iterations ranges from 20 to 200 with step size 20. For each number of iterations, the average gap with respect to the best known solution and the average computation time are given in Table 3. Random seeds (1,2,3,4, and 5) are used to avoid that a solution after 40 iterations is better than a solution after 60 iterations. Note that our solutions are compared to the best known solutions as reported in [13].

The second column shows that choosing $I_{max} = 140$ would be an appropriate choice since the average gap does not decline with more iterations. The average computation time is high, but this is mainly due to one instance (*pr16*) that is responsible for almost half of the computation time.

The final values for the parameters are given in Table 4.

Table 4: Values of the parameters used in the replication

Parameter	Value
P_h	0.1
P_l	0.3
T_0	0.1
α	0.995
S_{pool}	1000
I_{max}	140
N	50
I_{no_impr}	20

.

Table 5 summarizes the the results for the 76 instances and compares the results obtained by Hu and Lim [13] with the results that we obtained with this heuristic. Here, similar to the method of Hu and Lim [13], one run ($seed = 3$) is used to obtain a good approximation. The first two columns provide general information on the set to which the instances belong. The *num* column gives the number of instances in a set. The next two columns contain the results obtained by Hu and Lim [13]. The last columns show our results. First, the average gap of our solutions (SVR) with respect to the best known solution (BKS) is given. Next, the average gap that we obtained with respect to the solutions of Hu and Lim [13] (SVHL) are presented. These gaps are computed by the Equations (17) and (18).

$$Gap_{BKS} = \frac{BKS - SVR}{BKS} \times 100\% \tag{17}$$

$$Gap_{HL} = \frac{SVHL - SVR}{SVHL} \times 100\% \tag{18}$$

Finally, the number of identical results and the number of improvements of our solutions with respect to the solutions of Hu and Lim [13] are given in the last two columns .

Table 5: Overall comparison to I3CH

Set	Subset	num	Hu & Lim		Sluijk				
			AG_{BKS} (%)	AT (s)	AG_{BKS} (%)	AG_{HuLim} (%)	AT (s)	#Same SVHL	#Impr SVHL
m = 1									
Solomon 100	C	9	0.00	25.2	0.53	0.53	104.2	8	0
	R	12	0.56	28.6	0.59	0.02	100.8	6	2
	RC	8	1.66	25.5	0.46	-1.33	89.3	5	3
Solomon 200	C	8	0.40	84.4	3.48	3.09	775.2	0	0
	R	11	1.04	176.2	12.33	11.42	1802.5	0	0
	RC	8	2.68	119.3	8.93	6.43	2111.0	0	0
Cordeau	PR	10	1.05	109.0	5.68	4.66	442.7	1	2
Cordeau	PR	10	3.79	130.2	8.96	3.79	1401.0	2	1
m = 2									
Solomon 100	C	9	0.00	87.0	0.96	0.96	215.6	4	0
	R	12	0.54	63.0	1.36	0.82	190.8	5	1
	RC	8	0.90	58.9	1.65	0.75	182.7	2	1
Solomon 200	C	8	0.68	401.2	4.55	3.91	1470.9	0	0
	R	11	0.16	526.8	8.36	8.21	2506.6	0	0
	RC	8	0.56	439.7	8.53	8.02	3178.1	0	0
Cordeau 1-10	PR	10	0.94	247.1	5.43	4.53	835.2	2	0
Cordeau 11-20	PR	10	2.69	304.6	8.68	6.16	3443.8	0	0
m = 3									
Solomon 100	C	9	0.00	190.2	2.11	2.11	345.3	1	0
	R	12	0.21	118.3	2.27	2.07	296.6	4	1
	RC	8	0.26	101.0	1.08	0.83	282.0	2	1
Solomon 200	C	8	0.00	12.3	3.04	3.04	1653.2	3	0
	R	11	0.01	90.8	0.04	0.03	277.9	10	0
	RC	8	-0.04	164.1	0.80	0.84	1587.4	3	0
Cordeau 1-10	PR	10	0.35	424.0	5.79	5.46	1324.1	1	0
Cordeau 11-20	PR	10	1.00	497.0	6.15	5.22	1731.6	1	0
m = 4									
Solomon 100	C	9	0.01	261.8	1.87	1.86	497.5	3	0
	R	12	0.05	184.3	2.55	2.49	438.0	1	0
	RC	8	0.12	152.4	0.81	0.68	425.3	2	1
Solomon 200	C	8	0.00	0.1	0.00	0.00	44.3	8	0
	R	11	0.00	0.2	0.00	0.00	39.0	11	0
	RC	8	0.00	0.2	0.00	0.00	32.0	6	0
Cordeau 1-10	PR	10	0.05	566.5	6.61	6.58	1834.1	1	0
Cordeau 11-20	PR	10	-0.64	728.6	5.73	6.30	2069.1	1	0
Grand total		304	0.59	200.9	3.80	3.18	984.0	93	13

We obtained better results than given in the paper of Hu and Lim [13] on eight instances for $m = 1$, on two instances for both $m = 2$ and $m = 3$, and on one instance for $m = 4$. Hu and Lim [13] reported 35 new best solutions. While replicating the heuristic, only one new best solution has been found, namely for *pr11* when $m = 1$. This new best solution is equal to the solution that Hu and Lim [13] obtained for this instance (353, $BKS = 351$).

The last row of Table 5 gives the total number of instances and the average performances over all instances. It can be concluded that the heuristic required more time than given in [13] and that the overall results are worse. One possible reason for these worse results is that we allowed a lower maximum number of iterations ($I_{max} = 140$ instead of 3000). Another reason could be the way we managed the memory of our program. However, for some instances we obtained better results, but this is most likely because of the stochasticity of the heuristic.

As has been stated before, our computation times are larger than the ones reported in [13]. During the calculations, we kept track of the computation time that was required for each component. Those computation times are included in Table 6. The last four columns of this table show how many times there was a change in the solution and how many times each component contributed to this change. We only considered the changes that did not involve a tie. Hence, those who were strictly better than the previous solution and the solutions of the other components. Detailed results can be found in Appendix 9.2.

Table 6: Detailed results on time and counters

Set	Subset	AT (s)	AT RR (s)	AT LS (s)	AT SA (s)	RR (#)	LS (#)	SA (#)	Total (#)
m = 1									
Solomon 100	C	104.2	4.9	46.4	52.8	0.78	1.56	0.33	2.67
	R	100.8	4.9	45.3	50.6	1.58	1.58	0.42	3.58
	RC	89.3	5.1	40.2	44.0	0.63	1.13	0.88	2.63
Solomon 200	C	775.2	7.5	396.6	370.8	0.25	12.50	0.00	12.75
	R	1802.5	10.5	1087.3	704.3	0.64	10.36	0.00	11.00
	RC	2111.0	12.9	1263.7	834.0	0.25	9.75	0.00	10.00
Cordeau	PR	442.7	9.7	251.3	181.6	0.60	2.80	0.10	3.50
Cordeau	PR	1401.0	18.7	818.5	563.4	0.70	4.00	0.10	4.80
m = 2									
Solomon 100	C	215.6	12.4	101.4	101.6	1.89	4.33	0.00	6.22
	R	190.8	11.3	94.4	85.1	3.67	3.33	0.08	7.08
	RC	182.7	10.1	85.7	86.9	3.25	2.75	0.00	6.00
Solomon 200	C	1470.9	18.5	764.9	686.9	10.63	4.38	0.00	15.00
	R	2506.6	25.3	1435.8	1044.8	14.36	2.45	0.00	16.82
	RC	3178.1	36.4	1867.3	1273.5	16.38	1.63	0.00	18.00
Cordeau	PR	835.2	15.3	473.1	346.6	7.60	4.30	0.00	11.90
Cordeau	PR	3443.8	38.8	2038.6	1365.8	16.80	3.10	0.00	19.90
m = 3									
Solomon 100	C	345.3	9.5	172.1	163.5	3.33	9.67	0.00	13.00
	R	296.6	10.7	152.4	133.5	6.50	4.58	0.00	11.08
	RC	282.0	8.7	142.3	130.9	6.25	4.00	0.13	10.38
Solomon 200	C	1653.2	3.9	887.1	761.2	0.00	7.13	0.13	7.25
	R	277.9	0.5	163.3	113.1	0.00	3.09	0.00	3.09
	RC	1587.4	4.1	957.0	625.6	0.00	8.63	0.00	8.63
Cordeau	PR	1324.1	13.7	750.8	559.2	16.40	4.50	0.00	20.90
Cordeau	PR	1731.6	22.1	1005.7	703.4	30.20	1.50	0.00	31.70
m = 4									
Solomon 100	C	497.5	9.1	250.8	237.3	11.89	8.11	0.00	20.00
	R	438.0	11.0	226.3	200.6	9.58	4.67	0.00	14.25
	RC	425.3	8.6	219.0	197.6	9.75	2.25	0.13	12.13
Solomon 200	C	44.3	0.0	25.0	18.0	0.00	0.88	0.13	1.00
	R	39.0	0.1	18.6	19.0	0.00	1.00	0.00	1.00
	RC	32.0	0.3	19.3	11.5	0.00	1.00	0.00	1.00
Cordeau	PR	1834.1	13.3	1047.4	772.8	14.50	7.60	0.00	22.10
Cordeau	PR	2069.1	16.2	1211.9	840.5	27.30	4.80	0.00	32.10
Grand Average		984.0	11.8	560.6	411.1	6.91	4.42	0.07	11.40
Grand Total						2095	1329	19	3443

From this table can be concluded that the RR does not require much time (11.8 seconds), while LS and SA components require most of the time, on average 560.6 seconds and 411.1 seconds respectively. LS requires even more time than SA. The last four columns indicate that, on average, RR contributes the most to the improvements. LS is accounted for the second largest contribution. Especially the low number of times that SA contributes to an improvement is remarkably, namely only 19 times out of 3443 times. We also observe that out of the 140 iterations, on average only 11.4 lead to an improvement of the solution. It differs per instance and even per random seed in which iterations these differences occur. Some already obtain the reported solution in the first few iterations, while other instances require more iterations to obtain the reported solution. Finally, for $m = 4$ and the set of *Solomon 200* we observe that on average only one iteration is required. The *Solomon 200* instances are created in such a way that all customers fit in four routes. Hence, for these instances the optimal solution is already found in the first iteration.

6.1 Extensions

6.1.1 Multi-Thread

For this extension, as mentioned in Section 4.7.1, we only consider instances being solve with $m = 4$ vehicles. This leaves 76 possible instances that could be incorporated in evaluating this extension. However, due to large computation times, fewer instances will be considered. First, we eliminated all instances that have as final solution that all customers are visited, since they require only one iteration. The instances that satisfy this criterion are *c201-208* , *r201-r211*, *rc201-rc208*, *pr01*, and *pr11*. Hence, all the instances in the 2-set of the Solomon instances and two instances from

the Cordeau set. Next, instances that require more than 10 minutes computation time are left out of consideration. The computation times were taken from Tables 17 and 18 in Appendix 9.2 that contains the computation times of the considered instances for one run. The instances that satisfy this criterion are *c104*, *pr02-pr10* (excluding *pr07*), and *pr12-pr20*. The subset now consists of 29 instances, namely *c101-c109* (excluding *c104*), *r101-r112*, *rc101-rc108*, and *pr07*.

After changing the use of random generators in the original program, the solution of each instance in the subset is calculated. The original heuristic had to be executed again to make a fair comparison between the heuristic with and without multi-threading due to the change in use of random generators. The average computation times of the original heuristic and the Multi-Threading heuristic are given in Table 7.

Table 7: Results of using threads in the program

Name	AT_{no_thread} (s)	AT_{thread} (s)	Speed up factor	Name	AT_{no_thread} (s)	AT_{thread} (s)	Speed up factor
c101	464.5	288.4	1.6	r108	476.1	305.0	1.6
c102	544.1	291.8	1.9	r109	377.8	242.5	1.6
c103	600.3	327.9	1.8	r110	421.9	270.9	1.6
c105	481.4	252.8	1.9	r111	422.0	272.6	1.5
c106	435.0	262.2	1.7	r112	465.1	328.5	1.4
c107	444.5	273.2	1.6	rc101	333.9	254.7	1.3
c108	460.2	284.0	1.6	rc102	376.3	296.6	1.3
c109	497.1	304.2	1.6	rc103	435.3	278.5	1.6
r101	267.8	165.8	1.6	rc104	477.7	313.7	1.5
r102	416.4	237.7	1.8	rc105	386.8	241.3	1.6
r103	433.2	270.8	1.6	rc106	404.9	251.5	1.6
r104	497.2	307.1	1.6	rc107	432.0	275.3	1.6
r105	389.1	199.0	2.0	rc108	445.4	296.8	1.5
r106	478.8	270.9	1.8	pr07	602.8	426.9	1.4
r107	448.7	283.9	1.6				
Average	445.4	278.4	1.6				

Original instances were selected on the 10 minutes restriction, but using multiple runs with different seeds resulted for two instances in an average computation time that is slightly larger than 10 minutes in the original heuristic. These instances are *c103* and *pr07*.

We can also conclude that the average computation time without multi-threading is equal to 445.4 seconds, while the average computation times including multi-threading is equal to 278.4 seconds. This implies that the Multi-Threading heuristic is 1.6 times faster than the original heuristic. However, we expected the heuristic to be (nearly) two times faster. A possible explanation for Multi-Threading heuristic to not be two times faster is that creating threads also requires times, which result in larger computation times for the LS and SA component. Hence, it is certainly a good idea to make use of multi-threading. Using five runs instead of one also leads to different results. Detailed result of this extension can be found in Appendix 9.3.

Our focus here is on the computation time, because the solution quality remains the same across the two cases. Nevertheless, it is important to emphasize some results. Taking an average of 5 runs ($seed$ = 1,2,3,4,5) resulted in different solutions compared to the situation in which we considered

only one run. For two instances (*rc102* and *rc104* we obtained better solutions than the best known solutions reported in [13]. For *rc102* we obtained 909 instead of 908, and for *rc104* we obtained 1063 instead of 1059. Hu and Lim [13] obtained 902 for *rc102* and 1064 for *rc104*. Hence, they obtained a better solution for *rc104*, but their solution for *rc102* is not as good as ours.

6.1.2 Size of POOL

For this investigation, we reduced the subset to 25. This implies that four more instances are removed from the subset. The instances that are excluded are *c103*, *r112*, *rc104*, and *pr07* since they require the largest computation times. Note that the size of $POOL$ influences the computation time on two positions in the heuristic. The first position is in the RR component since the time that is required to solve the set packaging formulation is affected. The second position is in both the LS and SA component since the time that is needed to add and remove routes from $POOL$ is affected. Using a larger $POOL$ size implies that more searching time is required in order to find the correct position to insert and remove routes from the list.

Note also that the Multi-Threading heuristic is used which implies that the computation times for changes in the $POOL$ are larger than in the original heuristic. In the Multi-Threading heuristic, routes are not added to the $POOL$ directly, but first to a list (one for LS and one for SA) and once the threads in the iteration are finished, each route in each list is added to $POOL$. This procedure requires more time than adding the routes to $POOL$ directly.

Table 8 reports the average performance for each setting of S_{pool} for the subset. The first column provides information on the size of $POOL$. The second and the third column provide information on the average gap (AG) of the average profit (AP) and the best profit (BP) found with respect to the best known solution. Column four contains the corresponding average (A) computation time. Column five and six show how much percent of the computation time is spent on solving the set packaging formulation (RR) and on changing the content of $POOL$. The final two columns give information on the percentage of RR being chosen with respect to the total number of improvements. Detailed results can be found in Appendix 9.4.

From the table can be observed that the average gap with respect to average profit decreases until $S_{pool} = 4000$, the average gap with respect to best profit decreases until $S_{pool} = 1500$, and the average percentage of times that RR got chosen as the new solution A is increasing until $S_{pool} = 3000$. The average computation time does increase with S_{pool}.

In Figure 2 the computation time for each component is given. In this graph, one can clearly observe that the time required for the RR component and the addition and removal of routes in $POOL$ increases as the allowable size of $POOL$ increases. The time required for changes in $POOL$ increases when the S_{pool} increases due to the fact that more routes have to be added to $POOL$ each time and more time is required for checking whether a route is already included in $POOL$ since more comparisons have to be made.

We also observe from the table that for small sizes of $POOL$, the percentage of times that RR gets chosen increases rapidly, but once it has reached the size of 1500, it does not change much any more. This implies that from a size of 1500 onwards, the size of $POOL$ does not have much influence on the number of times that RR is chosen.

In the original heuristic, S_{pool} was set equal to 1000. We would suggest to increase this size to 1500 in the Multi-Thread heuristic for three reasons. Firstly, setting S_{pool} larger than 1500 will lead to larger average gaps with respect to best profit, unless a size of 4000 is chosen. Secondly, the average computation time for this size is just below five minutes, which is a reasonable boundary.

Finally, the percentage of time that RR gets chosen does not change much for larger *POOL*-sizes.

Table 8: Parameter tuning on pool size S_{pool}

S_{pool}	AG (%) AP	AG (%) BP	AT (s) A	RR Time (%) A	*POOL* time (%) A	RR chosen (%) A	# changes A
0	5.43	3.90	231.38	0.00	0.00	0.00	7.15
500	3.03	1.71	235.71	1.04	4.41	28.96	8.21
1000	1.87	0.86	255.20	3.37	10.25	58.94	8.32
1500	1.41	*0.59*	*277.56*	5.88	15.17	76.51	8.33
2000	1.32	0.80	302.09	7.43	19.12	77.44	8.29
3000	1.10	0.61	346.61	10.75	25.60	**82.56**	8.33
4000	**0.92**	**0.55**	394.39	12.68	31.16	82.33	8.44
5000	1.00	0.64	474.02	14.54	34.42	82.79	8.52

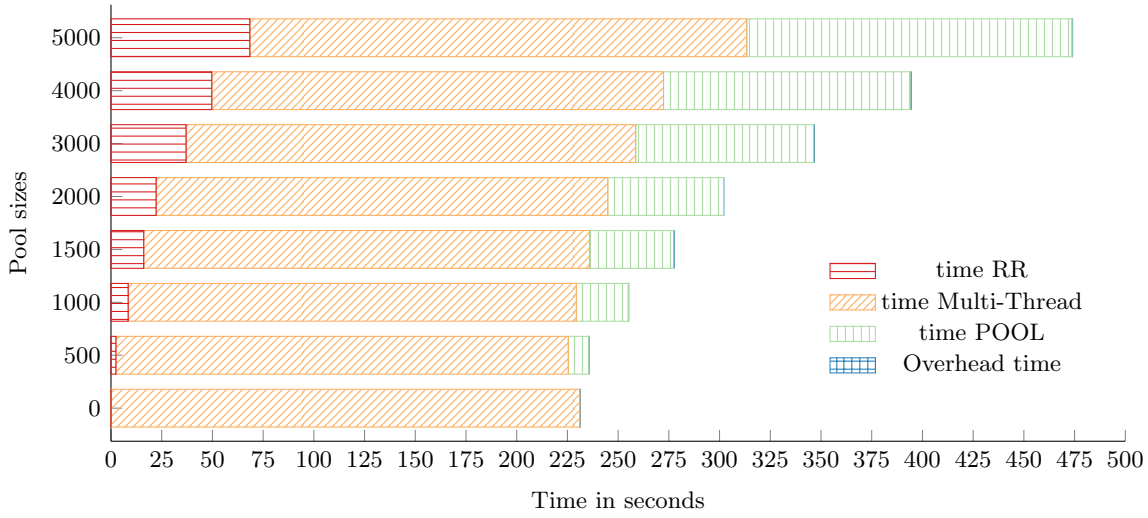

Figure 2: Analysis of computation time on the sizes of *POOL*

While investigating this extension, we found two solutions that were better than the best known solutions given in [13]. For *c108* we found a value of 1140 ($BKS = 1130$) when the size of the *POOL* was equal to 4000. For *r107* we found a value of 950 ($BKS = 945$) when the size of the *POOL* was equal to 5000. A final note is that the solution 909 for *rc102* that has been reported before as an improvement, was obtained multiple times for different *POOL*-sizes.

7 New Best Known Solutions

Throughout Section 6 for some instances we obtained better results than the best known solutions (BKS) reported in [13]. In Table 9 the corresponding instances are given. For each instance, the number of vehicles, the BKS given in [13], the latest BKS published by [11], the solution found by

Hu and Lim [13] and our solution is given. As can be concluded from the table, we found new best solutions for *c108*and *rc104*. Their corresponding routes are given in Appendix 9.5.

Table 9: New best known solutions

Instance	m	BKS [13]	Latest BKS [11]	Hu & Lim	Sluijk
pr11	1	351	353	353	353
c108	4	1130	1130	1130	**1140**
r107	4	945	950	950	950
rc102	4	908	908	902	**909**
rc104	4	1059	1064	1064	1063

8 Conclusion

The aim of this thesis was to replicate the results of the Iterative Three-Component Heuristic. This heuristic is applied to the Team Orienteering Problem with Time Windows (TOPTW). It uses a local search procedure and a simulated annealing procedure as the first two components to explore the neighbourhood solutions. During this search, new routes are discovered and stored in a route *POOL* which is used by the third component to obtain a high quality solution by route recombination. Together they form a three-component heuristic that is applied iteratively in order to improve the solution quality.

When comparing the overall performances on each set of instances, it can be concluded that overall we obtained worse results than Hu and Lim [13]. However, we did also find solutions that were better than the solutions that Hu and Lim [13] found.

The average computation time reported in [13] is equal to 200.9 seconds, while the average computation time we obtained with a lower value for the maximum number of iterations is equal to 982.0. Hence, our program is significantly slower than theirs. One reason for this is that our memory usage could be different than theirs. Another reason could be the use of different machine. A final reason could be that we misunderstood some parameters or procedures which could also cause larger computation times.

Two extensions were considered in the paper. In the first extension, multi-threading was added to the heuristic in order to decrease the computation time. LS and SA were made to run independently and in parallel. The Multi-Threading heuristic is on average 1.6 times faster than the original heuristic.

In the second extension, sensitivity analysis was performed on the values for the parameter S_{pool} for the Multi-Thread heuristic. From the results we concluded that instead of using $S_{pool} = 1000$, a pool size of 1500 is more beneficial since it increases both the average gap with respect to average profit as the average gap with respect to best profit. Moreover, it only leads to a small increase in computation time.

Throughout the replication and extensions we have found two new best known solutions, namely for *c108* (1140) and *rc102* (909).

Further research could be focused on applying the extension of changing pool size to all instances. Another interesting topic would be the contribution of Simulated Annealing. From our results, it can be concluded that it does not contribute much, but it is interesting to investigate why this contribution is low and how this can be improved.

9 Appendix

9.1 Algorithms

Algorithm 5 Local Search

Input: combination of routes R and a list of unvisited customers u
Initialize: $I_{no\_impr\_LS} \leftarrow 0$
while $I_{no\_impr\_LS} \leq I_{no\_impr}$ **do**
    Generate N neighbourhoods with *eliminator* and PP. Save each generated route in $POOL$
    Save the best neighbourhood solution Y' of these N neighbourhood solutions
    if Y' is better than the previous found best solution Y **then**
        $Y \leftarrow Y'$
        $I_{no\_impr\_LS} \leftarrow 0$
    else
        $I_{no\_impr\_LS} = I_{no\_impr\_LS} + 1$
    end if
end while
Return Y

Algorithm 6 Simulated Annealing

Input: combination of routes R and a list of unvisited customers u
Initialize: $I_{no\_impr} \leftarrow 0$
while $I_{no\_impr\_SA} \leq I_{no\_impr}$ **do**
    $T \leftarrow T_0$
    Y $\leftarrow$ R
    for Fifty times **do**
        change $\leftarrow$ `false`
        Generate a neighbourhood solution Y' of Y with *eliminator* and PP
        Save the routes of Y' in $POOL$
        if Y' is better than the Y **then**
            change $\leftarrow$ `true`
        end if
        if *change* equals `false` **then**
            Calculate the acceptance probability P_{SA}
            Generate a random number r
            if $r \leq P_{SA}$ **then**
                Accept worse neighbourhood
                change $\leftarrow$ `true`
            end if
        end if
        if *change* equals `true` **then**
            $Y \leftarrow Y'$
            $T = \alpha T$
        end if
    end for
    if Y is better than the solution found by SA (Y_{SA}) so far **then**
        $Y_{SA} \leftarrow Y$
        $I_{no\_impr\_SA} \leftarrow 0$
    else
        $I_{no\_impr\_SA} = I_{no\_impr\_SA} + 1$
    end if
end while
Return Y_{SA}

9.2 Detailed results of our implementation of I3CH

In this part of the appendix, the detailed solutions obtained by the iterative three-component heuristic (I3CH) on all TOPTW instances are given. The values of the parameters used can be found in Table 10.

Table 10: Final parameter values

Parameter	Value
P_h	0.1
P_l	0.3
T_0	0.1
α	0.995
S_{pool}	1000
I_{max}	140
N	50
I_{no_impr}	20
$seed$	3

The solutions are given in Tables 11 up to 18. The eight tables contain the solutions for the TOPTW instances $m = 1, 2, 3, 4$ and instance sets Solomon 100 & Cordeau 1-10 and Solomon 200 & Cordeau 11-20 respectively. The first columns contain information about the name of the instance and its best known solution. The columns under the heading *Hu & Lim* show the results of Hu and Lim [13]. The column *Profit* gives the total profit in a solution and the column *Time(s)* the computation time in seconds. The content of the final column, *Gap (%)*, is calculated with the help of Equation 19. Note that the original maximum number of iterations is used here ($I_{max} = 3000$).

$$AG_{BKS\_SVHL} = \frac{BKS - SVHL}{BKS} \times 100\% \tag{19}$$

The results that we obtained are given under the heading *Sluijk*. First, the obtained profit is given. The columns *Gap* with respect to BKS and SVHL are computed by Equations 20 and 21.

$$Gap_{BKS\_SVR} = \frac{BKS - SVR}{BKS} \times 100\% \tag{20}$$

$$Gap_{HL} = \frac{SVHL - SVR}{SVHL} \times 100\% \tag{21}$$

If the solution we found is better than the solution reported by Hu and Lim [13] or the best known solution given in [13], our solution is given in **boldface**, and the solution it is better over (BKS/SVHL) is given in *italic*.

The last eight columns contain information on the calculation time and the number of times a change occurred and which component caused this change. The first four of the eight columns give the total time and the time the RR, LS, and SA component required. The fifth column shows how many changes of the solution occurred while iterating and the last three columns show how many times each component contributed to a change. We only considered the changes that did not involve a tie. Hence, those who were strictly better than the previous solution and the solutions of the other components.

Table 11: Solomon 100 and Cordeau 1-10 with $m = 1$

		Hu & Lim			Sluijk										
Name	BKS	Profit	Gap (%) BKS	Time (s)	Profit	Gap (%) BKS	Gap (%) SVHL	Time (s)	RR (s)	LS (s)	SA (s)	Total (#)	R (#)	LS (#)	SA (#)
c101	320	320	0.00	21.8	320	0.00	0.00	88.9	5.2	37.8	45.4	2	1	1	0
c102	360	360	0.00	28.1	360	0.00	0.00	107.8	4.8	49.1	53.8	1	1	0	0
c103	400	400	0.00	27.1	400	0.00	0.00	113.6	5.1	52.3	56.1	2	1	0	1
c104	420	420	0.00	27.1	400	4.76	4.76	129.9	4.7	57.0	68.2	3	2	0	1
c105	340	340	0.00	23.4	340	0.00	0.00	91.1	4.8	39.9	46.3	2	1	1	0
c106	340	340	0.00	23.6	340	0.00	0.00	92.0	4.9	40.6	46.5	2	1	1	0
c107	370	370	0.00	24.7	370	0.00	0.00	98.4	4.8	43.5	50.1	5	3	0	2
c108	370	370	0.00	24.8	370	0.00	0.00	102.8	5.0	46.3	51.4	6	3	0	3
c109	380	380	0.00	26.3	380	0.00	0.00	113.4	4.9	51.3	57.2	1	1	0	0
r101	198	198	0.00	20.4	198	0.00	0.00	162.0	4.5	31.2	126.3	0	0	0	0
r102	286	286	0.00	29.3	286	0.00	0.00	88.3	5.0	42.5	40.7	3	1	1	1
r103	293	293	0.00	28.8	290	1.02	1.02	92.6	4.8	45.4	42.4	4	1	1	2
r104	303	*298*	1.65	27.3	**303**	0.00	-1.68	105.8	4.9	53.5	47.3	6	2	0	4
r105	247	247	0.00	26.0	247	0.00	0.00	85.1	4.9	39.8	40.4	3	2	0	1
r106	293	293	0.00	29.4	289	1.37	1.37	84.8	4.9	39.8	40.0	3	2	0	1
r107	299	297	0.67	27.8	297	0.67	0.00	92.3	4.8	44.4	43.1	5	2	1	2
r108	308	306	0.65	29.7	301	2.27	1.63	104.6	5.0	52.4	47.1	4	2	0	2
r109	277	277	0.00	31.1	277	0.00	0.00	91.2	4.9	44.6	41.7	6	3	1	2
r110	284	284	0.00	33.9	281	1.06	1.06	96.8	4.9	46.9	45.1	3	1	1	1
r111	297	295	0.67	27.7	295	0.67	0.00	99.4	4.9	49.4	45.1	1	1	0	0
r112	298	*289*	3.02	32.0	**298**	0.00	-3.11	107.0	4.9	53.6	48.5	5	2	0	3
rc101	219	219	0.00	21.8	219	0.00	0.00	79.8	4.6	35.3	39.9	1	0	1	0
rc102	266	266	0.00	25.5	266	0.00	0.00	82.2	5.1	36.1	40.9	3	1	1	1
rc103	266	266	0.00	27.1	266	0.00	0.00	87.6	4.9	39.6	43.1	3	1	1	1
rc104	301	301	0.00	27.2	301	0.00	0.00	96.3	5.0	44.4	46.9	2	1	1	0
rc105	244	244	0.00	26.4	244	0.00	0.00	92.8	5.1	43.8	43.9	3	2	1	0
rc106	252	*250*	0.79	25.0	**252**	0.00	-0.80	86.7	5.4	40.1	41.2	1	1	0	0
rc107	277	*274*	1.08	26.3	**277**	0.00	-1.09	92.0	5.3	39.2	47.5	4	1	1	2
rc108	298	*264*	11.41	25.1	**287**	3.69	-8.71	96.9	5.0	43.6	48.3	4	2	1	1
pr01	308	*305*	0.97	20.8	**308**	0.00	-0.98	185.2	4.0	100.2	80.9	3	1	1	1
pr02	404	394	2.48	47.9	393	2.72	0.25	309.7	6.6	171.6	131.5	2	2	0	0
pr03	394	394	0.00	72.9	359	8.88	8.88	314.0	7.8	164.4	141.7	3	2	0	1
pr04	489	489	0.00	109.3	451	7.77	7.77	394.1	9.9	210.3	173.8	1	1	0	0
pr05	595	594	0.17	185.4	540	9.24	9.09	682.0	12.7	406.8	262.3	6	6	0	0
pr06	590	590	0.00	189.9	568	3.73	3.73	786.6	16.3	465.0	305.1	1	1	0	0
pr07	298	298	0.00	26.5	298	0.00	0.00	149.6	4.7	77.1	67.7	4	2	0	2
pr08	463	*454*	1.94	77.4	**457**	1.30	-0.66	354.0	8.2	199.5	146.3	5	4	0	1
pr09	493	490	0.61	137.8	438	11.16	10.61	502.5	11.0	285.1	206.3	7	6	0	1
pr10	594	568	4.38	222.2	523	11.95	7.92	749.1	15.6	433.0	300.4	3	3	0	0

Table 12: Solomon 200 and Cordeau 11-20 with $m = 1$

Name	BKS	Hu & Lim			Sluijk										
		Profit	Gap (%) BKS	Time (s)	Profit	Gap (%) BKS	Gap (%) SVHL	Time (s)	RR (s)	LS (s)	SA (s)	Total (#)	R (#)	LS (#)	SA (#)
c201	870	870	0.00	70.1	850	2.30	2.30	706.8	7.6	357.1	341.9	5	5	0	0
c202	930	930	0.00	87.6	890	4.30	4.30	785.1	7.5	401.8	375.5	14	13	0	1
c203	960	960	0.00	92.3	930	3.13	3.13	865.3	7.8	452.6	404.4	18	18	0	0
c204	980	970	1.02	117.4	960	2.04	1.03	924.7	7.4	469.1	447.9	12	11	0	1
c205	910	900	1.10	70.7	890	2.20	1.11	686.8	7.2	349.8	329.6	13	13	0	0
c206	930	920	1.08	75.7	880	5.38	4.35	707.5	7.1	362.0	338.1	6	6	0	0
c207	930	930	0.00	77.4	900	3.23	3.23	775.2	7.8	391.9	375.3	14	14	0	0
c208	950	950	0.00	84.0	900	5.26	5.26	750.3	7.4	388.7	353.8	20	20	0	0
r201	797	789	1.00	101.8	780	2.13	1.14	840.5	8.0	464.2	368.0	9	9	0	0
r202	929	930	-0.11	175.6	846	8.93	9.03	1060.6	8.0	630.7	421.6	11	11	0	0
r203	1021	1020	0.10	221.4	892	12.63	12.55	1521.9	8.6	923.5	589.6	12	12	0	0
r204	1086	1073	1.20	236.9	888	18.23	17.24	1797.4	10.1	1074.8	712.2	13	12	0	1
r205	953	946	0.73	129.3	851	10.70	10.04	1718.3	10.6	1031.2	676.2	7	7	0	0
r206	1029	1021	0.78	169.3	865	15.94	15.28	1858.8	10.4	1117.1	731.0	12	11	0	1
r207	1072	1050	2.05	192.8	930	13.25	11.43	2118.0	10.6	1325.4	781.4	11	7	0	4
r208	1112	1092	1.80	230.0	890	19.96	18.50	2252.7	11.0	1376.8	864.2	8	8	0	0
r209	950	948	0.21	136.5	851	10.42	10.23	2021.6	11.6	1212.1	797.3	14	14	0	0
r210	987	982	0.51	176.9	887	10.13	9.67	2086.7	12.1	1257.6	816.2	11	10	0	1
r211	1046	1013	3.15	167.4	907	13.29	10.46	2551.2	14.0	1547.3	989.3	13	13	0	0
rc201	795	795	0.00	80.9	775	2.52	2.52	1291.6	10.1	708.8	572.5	5	5	0	0
rc202	936	924	1.28	129.3	861	8.01	6.82	1667.6	12.3	966.4	688.7	7	7	0	0
rc203	1003	966	3.69	134.3	906	9.67	6.21	2192.9	13.0	1363.6	815.8	15	15	0	0
rc204	1140	1093	4.12	167.5	951	16.58	12.99	2254.2	12.7	1363.2	877.9	11	10	0	1
rc205	859	847	1.40	99.2	764	11.06	9.80	1921.5	9.8	1146.4	764.9	9	9	0	0
rc206	895	863	3.58	98.4	854	4.58	1.04	2023.4	13.4	1193.1	816.5	13	13	0	0
rc207	983	957	2.64	122.0	899	8.55	6.06	2539.9	13.9	1530.5	995.0	10	10	0	0
rc208	1053	1003	4.75	123.0	943	10.45	5.98	2996.5	18.2	1837.3	1140.5	10	9	0	1
pr11	*351*	353	-0.57	30.8	**353**	-0.57	0.00	625.3	6.0	356.1	263.1	6	3	0	3
pr12	442	433	2.04	59.8	411	7.01	5.08	1100.4	17.4	607.0	475.9	5	4	0	1
pr13	461	466	-1.08	89.5	431	6.51	7.51	1745.9	23.4	973.0	749.5	5	4	1	0
pr14	567	521	8.11	144.4	462	18.52	11.32	2348.4	31.5	1291.1	1025.4	6	6	0	0
pr15	685	707	-3.21	248.2	620	9.49	12.31	5353.8	54.3	3271.8	2026.9	8	8	0	0
pr16	674	619	8.16	228.6	579	14.09	6.46	858.6	16.0	541.0	300.8	5	5	0	0
pr17	362	360	0.55	34.7	360	0.55	0.00	173.1	4.6	94.3	74.1	4	2	0	2
pr18	539	497	7.79	99.0	471	12.62	5.23	384.4	8.0	211.5	164.9	3	2	0	1
pr19	562	538	4.27	164.6	505	10.14	6.13	528.7	10.8	309.1	208.6	1	1	0	0
pr20	667	*588*	11.84	202.7	**592**	11.24	-0.68	891.0	15.4	530.1	345.4	5	5	0	0

Table 13: Solomon 100 and Cordeau 1-10 with $m = 2$

Name	BKS	Hu & Lim			Sluijk										
		Profit	Gap (%) BKS	Time (s)	Profit	Gap (%) BKS	Gap (%) SVHL	Time (s)	RR (s)	LS (s)	SA (s)	Total (#)	R (#)	LS (#)	SA (#)
c101	590	590	0.00	53.4	590	0.00	0.00	194.0	14.0	85.2	94.2	4	2	2	0
c102	660	660	0.00	78.4	650	1.52	1.52	206.4	2.2	102.2	101.8	2	0	2	0
c103	720	720	0.00	116.1	710	1.39	1.39	244.1	14.6	117.0	112.4	9	2	7	0
c104	760	760	0.00	94.4	750	1.32	1.32	276.0	13.3	134.0	128.6	4	1	3	0
c105	640	640	0.00	70.6	640	0.00	0.00	190.0	12.8	87.6	89.6	4	1	3	0
c106	620	620	0.00	149.2	610	1.61	1.61	195.9	14.6	90.1	91.1	6	3	3	0
c107	670	670	0.00	65.4	670	0.00	0.00	201.7	13.3	94.4	94.0	9	2	7	0
c108	680	680	0.00	85.9	680	0.00	0.00	210.2	13.9	97.3	98.9	10	4	6	0
c109	720	720	0.00	69.4	700	2.78	2.78	221.8	12.7	105.0	104.0	8	2	6	0
r101	349	349	0.00	42.1	349	0.00	0.00	131.6	10.8	56.7	64.0	4	2	1	1
r102	508	508	0.00	62.4	508	0.00	0.00	179.3	14.7	83.1	81.5	4	2	2	0
r103	522	519	0.57	68.3	517	0.96	0.39	179.7	2.2	90.9	86.6	5	0	5	0
r104	552	549	0.54	75.3	545	1.27	0.73	221.6	2.2	123.7	95.6	9	0	9	0
r105	453	447	1.32	56.2	447	1.32	0.00	170.5	13.9	77.8	78.8	9	5	4	0
r106	529	529	0.00	63.5	503	4.91	4.91	186.3	14.4	89.3	82.5	2	1	1	0
r107	535	533	0.37	63.2	522	2.43	2.06	208.5	13.8	105.7	89.0	11	7	4	0
r108	558	550	1.43	65.7	537	3.76	2.36	214.3	15.1	107.1	92.1	11	8	3	0
r109	506	506	0.00	60.5	506	0.00	0.00	184.4	13.6	89.6	81.1	7	4	3	0
r110	525	525	0.00	68.9	525	0.00	0.00	206.2	11.4	105.2	89.5	9	5	4	0
r111	544	542	0.37	67.0	541	0.55	0.18	195.2	10.0	97.3	87.9	6	4	2	0
r112	544	*534*	1.84	63.0	**538**	1.10	-0.75	212.3	13.5	106.1	92.6	8	6	2	0
rc101	427	427	0.00	52.3	419	1.87	1.87	158.6	9.7	71.4	77.4	2	1	1	0
rc102	505	505	0.00	59.8	504	0.20	0.20	177.3	10.3	84.2	82.8	11	5	6	0
rc103	524	519	0.95	60.3	507	3.24	2.31	185.0	11.8	85.1	88.1	6	3	3	0
rc104	575	556	3.30	59.9	554	3.65	0.36	191.7	10.8	90.4	90.5	12	8	4	0
rc105	480	480	0.00	58.6	480	0.00	0.00	176.7	8.3	84.2	84.1	4	2	2	0
rc106	483	*481*	0.41	56.0	**483**	0.00	-0.42	169.2	9.3	79.7	80.2	2	1	1	0
rc107	534	529	0.94	62.9	520	2.62	1.70	197.7	9.6	92.8	95.3	2	1	1	0
rc108	556	547	1.62	61.4	547	1.62	0.00	205.5	10.9	97.8	96.8	9	5	4	0
pr01	502	502	0.00	51.8	502	0.00	0.00	274.0	10.8	137.7	125.3	5	0	5	0
pr02	714	714	0.00	127.7	700	1.96	1.96	506.5	16.6	273.2	216.6	14	11	3	0
pr03	742	731	1.48	175.6	695	6.33	4.92	610.7	15.5	329.9	265.1	12	11	1	0
pr04	924	917	0.76	270.1	889	3.79	3.05	900.8	14.0	521.1	365.5	12	11	1	0
pr05	1090	1101	-1.01	410.0	946	13.21	14.08	1222.1	20.4	682.4	519.1	19	17	2	0
pr06	1076	1040	3.35	427.6	953	11.43	8.37	1340.4	7.0	796.3	536.8	10	0	10	0
pr07	566	566	0.00	71.8	566	0.00	0.00	299.8	14.3	154.7	130.6	9	4	5	0
pr08	834	824	1.20	184.1	810	2.88	1.70	653.2	18.4	366.7	268.0	7	3	4	0
pr09	905	878	2.98	304.9	851	5.97	3.08	1090.0	5.6	645.7	438.4	11	0	11	0
pr10	1124	1117	0.62	447.0	1026	8.72	8.15	1454.1	30.0	822.9	600.8	20	19	1	0

Table 14: Solomon 200 and Cordeau 11-20 with $m = 2$

Name	BKS	Hu & Lim			Sluijk										
		Profit	Gap (%) BKS	Time (s)	Profit	Gap (%) BKS	Gap (%) SVHL	Time (s)	RR (s)	LS (s)	SA (s)	Total (#)	R (#)	LS (#)	SA (#)
c201	1460	1450	0.68	321.2	1440	1.37	0.69	1409.7	21.3	740.8	647.0	20	16	4	0
c202	1470	1470	0.00	405.9	1430	2.72	2.72	1523.8	22.9	797.6	702.7	19	15	4	0
c203	1480	1470	0.68	458.2	1400	5.41	4.76	1545.1	24.3	788.4	731.8	12	10	2	0
c204	1480	1480	0.00	498.5	1420	4.05	4.05	1674.5	25.0	851.8	796.9	19	16	3	0
c205	1470	1450	1.36	322.2	1430	2.72	1.38	1402.8	21.7	735.7	644.7	16	14	2	0
c206	1480	1480	0.00	355.9	1420	4.05	4.05	1394.3	23.8	718.4	651.5	16	14	2	0
c207	1490	1470	1.34	423.5	1380	7.38	6.12	1414.3	4.6	747.5	661.6	10	0	10	0
c208	1490	1470	1.34	424.3	1360	8.72	7.48	1403.0	4.0	739.2	659.1	8	0	8	0
r201	1250	1254	-0.32	333.6	1199	4.08	4.39	1607.5	29.3	871.7	705.9	24	23	1	0
r202	1347	1344	0.22	588.5	1297	3.71	3.50	2081.8	24.4	1185.1	871.7	13	10	3	0
r203	1414	1416	-0.14	815.9	1273	9.97	10.10	2099.2	8.7	1220.0	869.7	14	8	6	0
r204	1458	1458	0.00	33.5	1282	12.07	12.07	2011.7	24.3	1119.4	867.3	15	13	2	0
r205	1379	1380	-0.07	606.1	1227	11.02	11.09	1718.8	23.9	942.4	752.0	8	6	2	0
r206	1440	1427	0.90	739.9	1292	10.28	9.46	2034.2	24.0	1156.8	852.8	16	14	2	0
r207	1458	1458	0.00	453.8	1331	8.71	8.71	2126.0	24.5	1233.8	867.1	11	10	1	0
r208	1458	1458	0.00	4.3	1310	10.15	10.15	2910.0	28.0	1665.1	1216.3	22	20	2	0
r209	1405	1404	0.07	600.3	1337	4.84	4.77	3605.3	34.3	2104.9	1465.0	24	22	2	0
r210	1423	1415	0.56	728.5	1327	6.75	6.22	3686.4	27.2	2160.4	1497.7	17	12	5	0
r211	1458	1450	0.55	890.9	1307	10.36	9.86	3691.9	29.3	2134.1	1527.2	21	20	1	0
rc201	1377	1384	-0.51	267.4	1308	5.01	5.49	2314.4	41.2	1291.6	980.9	14	13	1	0
rc202	1509	1500	0.60	386.7	1442	4.44	3.87	2902.0	35.2	1658.0	1208.1	23	22	1	0
rc203	1632	1627	0.31	482.8	1448	11.27	11.00	3031.8	33.3	1766.5	1231.3	11	10	1	0
rc204	1716	1704	0.70	760.8	1443	15.91	15.32	3279.4	30.5	1956.5	1291.4	18	16	2	0
rc205	1458	1452	0.41	311.0	1396	4.25	3.86	2858.2	34.7	1623.2	1199.7	16	14	2	0
rc206	1546	1525	1.36	335.9	1451	6.14	4.85	3364.5	38.7	2011.7	1313.1	30	28	2	0
rc207	1587	1582	0.32	408.6	1462	7.88	7.59	3791.5	38.8	2251.2	1500.3	17	16	1	0
rc208	1691	1669	1.30	564.4	1466	13.31	12.16	3883.0	39.2	2380.0	1463.1	15	12	3	0
pr11	566	559	1.24	71.3	540	4.59	3.40	707.8	3.1	385.7	319.0	1	0	1	0
pr12	774	768	0.78	143.6	728	5.94	5.21	1335.7	37.3	744.8	553.5	14	13	1	0
pr13	831	832	-0.12	238.6	773	6.98	7.09	2220.5	26.4	1293.6	900.0	11	9	2	0
pr14	1017	978	3.83	337.3	907	10.82	7.26	2966.9	30.9	1759.7	1175.7	21	20	1	0
pr15	1219	1205	1.15	479.1	1047	14.11	13.11	4278.2	49.4	2519.6	1708.5	39	38	1	0
pr16	1231	1124	8.69	500.5	1028	16.49	8.54	4592.0	64.2	2704.9	1822.0	28	27	1	0
pr17	652	639	1.99	117.0	612	6.13	4.23	1258.2	34.0	691.3	532.8	5	4	1	0
pr18	938	937	0.11	231.0	916	2.35	2.24	3232.0	9.0	1974.8	1247.7	19	0	19	0
pr19	1034	1003	3.00	386.1	915	11.51	8.77	5231.6	54.0	3170.4	2006.1	23	21	2	0
pr20	1232	1155	6.25	541.6	1135	7.87	1.73	8615.5	79.7	5141.2	3392.9	38	36	2	0

Table 15: Solomon 100 and Cordeau 1-10 with $m = 3$

Name	BKS	Hu & Lim			Sluijk										
		Profit	Gap (%) BKS	Time (s)	Profit	Gap (%) BKS	Gap (%) SVHL	Time (s)	RR (s)	LS (s)	SA (s)	Total (#)	R (#)	LS (#)	SA (#)
c101	810	810	0.00	303.1	810	0.00	0.00	290.8	11.8	140.7	137.6	10	3	7	0
c102	920	920	0.00	237.5	910	1.09	1.09	348.3	13.1	170.6	164.4	18	9	9	0
c103	980	990	-1.02	156.5	940	4.08	5.05	388.4	2.3	199.1	186.9	16	0	16	0
c104	1030	1030	0.00	155.9	1000	2.91	2.91	464.2	3.1	242.8	218.1	14	3	11	0
c105	870	870	0.00	110.2	850	2.30	2.30	302.4	12.8	144.9	144.6	3	0	3	0
c106	870	870	0.00	227.4	850	2.30	2.30	300.7	11.4	143.2	146.0	8	4	4	0
c107	910	910	0.00	151.8	900	1.10	1.10	329.4	13.5	163.8	152.0	14	4	10	0
c108	920	920	0.00	212.6	910	1.09	1.09	333.9	15.2	164.2	154.4	16	7	9	0
c109	970	960	1.03	157.2	930	4.12	3.13	349.9	2.5	179.7	167.5	18	0	18	0
r101	484	481	0.62	67.7	481	0.62	0.00	189.0	12.4	81.3	95.3	3	2	1	0
r102	694	691	0.43	111.4	691	0.43	0.00	287.0	15.0	141.6	130.4	9	7	2	0
r103	747	740	0.94	125.6	711	4.82	3.92	313.9	2.6	167.5	143.7	11	1	10	0
r104	777	777	0.00	128.3	745	4.12	4.12	339.4	14.7	174.5	150.1	10	8	2	0
r105	620	*619*	0.16	165.8	**620**	0.00	-0.16	243.3	16.1	115.2	111.8	8	4	4	0
r106	729	729	0.00	122.7	716	1.78	1.78	298.0	13.7	152.6	131.7	12	8	4	0
r107	760	759	0.13	120.5	745	1.97	1.84	307.9	12.2	159.5	136.1	16	13	3	0
r108	797	797	0.00	135.7	759	4.77	4.77	346.4	12.1	187.7	146.5	19	17	2	0
r109	710	710	0.00	103.6	710	0.00	0.00	281.6	11.7	142.0	127.8	12	9	3	0
r110	737	736	0.14	109.9	736	0.14	0.00	316.9	13.1	164.0	139.7	11	9	2	0
r111	774	773	0.13	118.1	742	4.13	4.01	303.2	2.2	160.7	140.2	11	0	11	0
r112	776	776	0.00	110.5	741	4.51	4.51	332.4	2.2	181.9	148.2	11	0	11	0
rc101	621	621	0.00	87.6	621	0.00	0.00	237.8	6.6	113.3	117.9	3	1	1	1
rc102	714	714	0.00	105.4	710	0.56	0.56	261.8	9.6	130.5	121.7	15	9	6	0
rc103	764	764	0.00	105.7	755	1.18	1.18	296.4	8.6	149.3	138.4	16	11	5	0
rc104	833	834	-0.12	124.4	807	3.12	3.24	321.9	8.6	166.7	146.5	11	7	4	0
rc105	682	682	0.00	90.4	680	0.29	0.29	276.6	8.5	140.4	127.7	7	3	4	0
rc106	706	706	0.00	88.1	706	0.00	0.00	252.1	8.3	121.9	121.9	17	11	6	0
rc107	773	762	1.42	98.7	746	3.49	2.10	294.0	9.4	156.4	128.1	6	4	2	0
rc108	795	*789*	0.75	107.4	**795**	0.00	-0.76	315.1	9.8	160.2	144.9	8	4	4	0
pr01	622	622	0.00	132.1	622	0.00	0.00	333.9	2.0	168.8	163.0	3	0	3	0
pr02	942	936	0.64	260.6	851	9.66	9.08	685.0	2.9	370.2	311.7	6	0	6	0
pr03	1010	1010	0.00	301.5	977	3.27	3.27	988.9	20.0	542.5	426.3	16	15	1	0
pr04	1294	1286	0.62	442.7	1263	2.40	1.79	1448.3	20.6	826.0	601.4	46	45	1	0
pr05	1482	1481	0.07	650.3	1377	7.09	7.02	1997.7	30.6	1126.8	839.9	36	35	1	0
pr06	1514	1501	0.86	651.2	1397	7.73	6.93	2108.8	25.6	1206.0	876.7	30	29	1	0
pr07	744	738	0.81	260.4	736	1.08	0.27	470.9	2.4	258.0	210.4	8	0	8	0
pr08	1138	1139	-0.09	307.0	1061	6.77	6.85	943.5	3.7	525.2	414.4	9	0	9	0
pr09	1275	1272	0.24	503.1	1108	13.10	12.89	1711.8	5.2	999.5	706.8	13	0	13	0
pr10	1573	1567	0.38	731.1	1465	6.87	6.51	2551.6	24.5	1484.9	1041.7	42	40	2	0

Table 16: Solomon 200 and Cordeau 11-20 with $m = 3$

Name	BKS	Hu & Lim			Sluijk										
		Profit	Gap (%) BKS	Time (s)	Profit	Gap (%) BKS	Gap (%) SVHL	Time (s)	RR (s)	LS (s)	SA (s)	Total (#)	R (#)	LS (#)	SA (#)
c201	1810	1810	0.00	3.8	1810	0.0	0.0	37.1	0.2	25.2	10.7	1	0	1	0
c202	1810	1810	0.00	7.2	1810	0.0	0.0	571.2	1.8	329.8	238.5	6	0	6	0
c203	1810	1810	0.00	7.9	1700	6.1	6.1	2451.9	4.5	1278.4	1167.8	6	0	6	0
c204	1810	1810	0.00	11.2	1670	7.7	7.7	2598.1	5.3	1325.3	1266.2	6	0	5	1
c205	1810	1810	0.00	12.6	1810	0.0	0.0	477.0	1.0	276.4	198.6	7	0	7	0
c206	1810	1810	0.00	18.0	1750	3.3	3.3	2338.6	6.1	1271.0	1060.6	9	0	9	0
c207	1810	1810	0.00	16.9	1780	1.7	1.7	2466.3	5.9	1390.9	1068.6	16	0	16	0
c208	1810	1810	0.00	20.6	1710	5.5	5.5	2285.4	6.3	1199.5	1078.5	7	0	7	0
r201	1441	1439	0.14	981.2	1434	0.5	0.3	2420.6	3.7	1382.8	1033.2	20	0	20	0
r202	1458	1458	0.00	15.3	1458	0.0	0.0	103.1	0.2	72.4	29.5	2	0	2	0
r203	1458	1458	0.00	0.2	1458	0.0	0.0	58.6	0.0	44.0	13.5	1	0	1	0
r204	1458	1458	0.00	0.2	1458	0.0	0.0	69.3	0.2	45.9	22.2	2	0	2	0
r205	1458	1458	0.00	0.3	1458	0.0	0.0	64.2	0.2	44.3	18.8	2	0	2	0
r206	1458	1458	0.00	0.3	1458	0.0	0.0	122.9	0.4	76.1	45.3	2	0	2	0
r207	1458	1458	0.00	0.3	1458	0.0	0.0	36.5	0.2	21.1	14.1	1	0	1	0
r208	1458	1458	0.00	0.2	1458	0.0	0.0	39.1	0.2	20.8	17.1	1	0	1	0
r209	1458	1458	0.00	0.3	1458	0.0	0.0	34.6	0.2	18.8	14.7	1	0	1	0
r210	1458	1458	0.00	0.2	1458	0.0	0.0	47.3	0.2	29.0	17.1	1	0	1	0
r211	1458	1458	0.00	0.4	1458	0.0	0.0	60.5	0.2	40.8	18.5	1	0	1	0
rc201	1698	1693	0.29	575.2	1668	1.8	1.5	2029.6	3.3	1201.1	824.7	9	0	9	0
rc202	1724	1724	0.00	1.1	1719	0.3	0.3	2920.5	6.3	1809.5	1104.0	14	0	14	0
rc203	1724	1724	0.00	0.3	1724	0.0	0.0	136.0	2.0	95.2	38.1	3	0	3	0
rc204	1724	1724	0.00	0.3	1664	3.5	3.5	2322.4	7.8	1304.1	1009.9	2	0	2	0
rc205	1709	1719	-0.59	734.4	1704	0.3	0.9	2401.2	4.7	1467.3	928.6	18	0	18	0
rc206	1724	1724	0.00	0.5	1714	0.6	0.6	2369.6	7.0	1435.6	926.5	16	0	16	0
rc207	1724	1724	0.00	0.3	1724	0.0	0.0	148.7	0.5	105.9	41.7	2	0	2	0
rc208	1724	1724	0.00	0.4	1724	0.0	0.0	370.9	1.2	237.8	131.2	5	0	5	0
pr11	654	654	0.00	151.7	654	0.0	0.0	443.0	2.8	239.5	200.6	2	0	2	0
pr12	1002	997	0.50	294.3	958	4.4	3.9	841.7	17.7	457.9	365.8	28	27	1	0
pr13	1139	1145	-0.53	378.9	1045	8.3	8.7	1283.0	19.5	732.9	530.3	18	16	2	0
pr14	1372	1315	4.15	533.7	1218	11.2	7.4	1756.9	27.2	986.6	742.7	42	41	1	0
pr15	1650	1654	-0.24	708.1	1587	3.8	4.1	2878.6	27.9	1721.8	1128.3	38	37	1	0
pr16	1668	1609	3.54	818.1	1495	10.4	7.1	2666.8	29.9	1556.4	1080.0	67	66	1	0
pr17	838	841	-0.36	184.3	835	0.4	0.7	642.5	20.4	348.7	273.3	14	12	2	0
pr18	1281	1276	0.39	386.6	1228	4.1	3.8	1531.6	23.1	895.3	612.9	26	25	1	0
pr19	1417	1403	0.99	604.1	1277	9.9	9.0	2096.4	19.8	1231.1	845.0	37	36	1	0
pr20	1684	1658	1.54	909.7	1532	9.0	7.6	3175.0	32.5	1886.5	1255.4	45	42	3	0

Table 17: Solomon 100 and Cordeau 1-10 with $m = 4$

Name	BKS	Hu & Lim			Sluijk										
		Profit	Gap (%) BKS	Time (s)	Profit	Gap (%) BKS	Gap (%) SVHL	Time (s)	RR (s)	LS (s)	SA (s)	Total (#)	R (#)	LS (#)	SA (#)
c101	1020	1020	0.00	176.4	980	3.9	3.9	427.8	3.6	215.5	207.5	10	0	10	0
c102	1150	1150	0.00	274.1	1150	0.0	0.0	519.8	9.2	261.8	248.5	26	20	6	0
c103	1200	1210	-0.83	301.1	1200	0.0	0.8	585.3	13.6	292.7	278.7	30	24	6	0
c104	1260	1260	0.00	281.8	1240	1.6	1.6	662.7	14.8	331.9	315.8	25	23	2	0
c105	1070	1060	0.93	333.2	1060	0.9	0.0	432.8	10.3	216.1	206.2	21	10	11	0
c106	1080	1080	0.00	179.4	1070	0.9	0.9	440.8	13.0	218.3	209.4	22	14	8	0
c107	1120	1120	0.00	209.3	1120	0.0	0.0	455.3	11.7	227.5	215.9	21	16	5	0
c108	1130	1130	0.00	373.1	1080	4.4	4.4	460.4	3.0	239.1	218.1	15	0	15	0
c109	1190	1190	0.00	227.6	1130	5.0	5.0	492.7	2.7	254.5	235.4	10	0	10	0
r101	611	608	0.49	93.5	608	0.5	0.0	258.6	12.0	118.9	127.6	9	4	5	0
r102	843	837	0.71	162.6	810	3.9	3.2	361.4	2.5	179.5	179.4	5	0	5	0
r103	926	928	-0.22	206.9	876	5.4	5.6	455.4	2.8	242.0	210.3	10	0	10	0
r104	972	969	0.31	209.1	955	1.7	1.4	474.7	13.9	249.8	210.9	25	24	1	0
r105	778	778	0.00	142.4	772	0.8	0.8	327.7	16.2	157.2	154.2	5	4	1	0
r106	905	906	-0.11	185.9	890	1.7	1.8	474.5	15.4	240.1	218.8	13	11	2	0
r107	945	950	-0.53	221.4	901	4.7	5.2	495.2	3.1	263.4	228.5	14	0	14	0
r108	994	994	0.00	220.0	956	3.8	3.8	559.0	15.1	300.4	243.3	24	23	1	0
r109	885	885	0.00	145.8	859	2.9	2.9	456.2	19.6	226.1	210.4	17	16	1	0
r110	914	915	-0.11	168.6	910	0.4	0.5	454.4	13.9	238.3	202.1	17	15	2	0
r111	949	952	-0.32	203.5	916	3.5	3.8	450.6	2.5	241.3	206.7	13	0	13	0
r112	971	967	0.41	251.8	959	1.2	0.8	488.1	14.6	257.9	215.4	19	18	1	0
rc101	811	808	0.37	119.7	808	0.4	0.0	370.1	7.5	177.3	185.1	4	3	0	1
rc102	908	*899*	0.99	136.7	**902**	0.7	-0.3	386.7	10.9	198.8	176.9	10	6	4	0
rc103	970	974	-0.41	170.2	967	0.3	0.7	448.9	9.6	230.7	208.4	16	15	1	0
rc104	1059	1064	-0.47	194.1	1031	2.6	3.1	492.0	8.5	265.1	218.3	11	10	1	0
rc105	875	875	0.00	127.0	875	0.0	0.0	386.6	7.5	190.8	188.1	6	4	2	0
rc106	909	909	0.00	143.0	901	0.9	0.9	402.1	8.4	205.5	188.2	14	9	5	0
rc107	980	980	0.00	155.4	974	0.6	0.6	420.9	5.4	225.5	189.9	9	7	2	0
rc108	1025	1020	0.49	173.2	1015	1.0	0.5	494.6	10.7	257.9	225.9	27	24	3	0
pr01	657	657	0.00	0.1	657	0.0	0.0	9.2	0.4	6.6	2.0	1	0	1	0
pr02	1079	1073	0.56	380.6	1013	6.1	5.6	1018.6	4.4	558.4	455.4	10	0	10	0
pr03	1222	1232	-0.82	436.6	1118	8.5	9.3	1386.5	4.7	774.2	607.2	14	0	14	0
pr04	1557	1585	-1.80	603.6	1532	1.6	3.3	2473.3	26.7	1406.3	1039.8	48	47	1	0
pr05	1833	1838	-0.27	902.9	1707	6.9	7.1	2825.8	29.7	1629.7	1165.5	36	34	2	0
pr06	1860	1835	1.34	939.6	1501	19.3	18.2	2849.1	6.8	1635.5	1206.1	16	0	16	0
pr07	876	872	0.46	228.9	867	1.0	0.6	581.3	2.7	313.0	265.4	14	0	14	0
pr08	1382	1377	0.36	429.9	1346	2.6	2.3	1287.9	22.6	705.1	559.9	24	23	1	0
pr09	1619	1604	0.93	698.5	1364	15.8	15.0	2276.4	5.5	1308.2	962.1	16	0	16	0
pr10	1939	1943	-0.21	1044.4	1855	4.3	4.5	3632.7	29.6	2137.3	1464.9	42	41	1	0

Table 18: Solomon 200 and Cordeau 11-20 with $m = 4$

Name	BKS	Hu & Lim			Sluijk										
		Profit	Gap (%) BKS	Time (s)	Profit	Gap (%) BKS	Gap (%) SVHL	Time (s)	RR (s)	LS (s)	SA (s)	Total (#)	R (#)	LS (#)	SA (#)
c201	1810	1810	0.00	0.1	1810	0.0	0.0	43.5	0.0	10.8	31.4	1	0	0	1
c202	1810	1810	0.00	0.1	1810	0.0	0.0	39.1	0.0	27.6	10.1	1	0	1	0
c203	1810	1810	0.00	0.2	1810	0.0	0.0	57.0	0.0	28.6	27.0	1	0	1	0
c204	1810	1810	0.00	0.1	1810	0.0	0.0	47.0	0.0	33.2	12.2	1	0	1	0
c205	1810	1810	0.00	0.1	1810	0.0	0.0	35.0	0.0	15.7	18.1	1	0	1	0
c206	1810	1810	0.00	0.1	1810	0.0	0.0	30.2	0.1	16.1	12.8	1	0	1	0
c207	1810	1810	0.00	0.1	1810	0.0	0.0	60.1	0.0	41.8	16.9	1	0	1	0
c208	1810	1810	0.00	0.1	1810	0.0	0.0	42.4	0.1	25.8	15.3	1	0	1	0
r201	1458	1458	0.00	0.2	1458	0.0	0.0	56.8	0.0	30.6	25.1	1	0	1	0
r202	1458	1458	0.00	0.2	1458	0.0	0.0	51.1	0.0	24.2	25.7	1	0	1	0
r203	1458	1458	0.00	0.2	1458	0.0	0.0	29.0	0.0	17.8	9.9	1	0	1	0
r204	1458	1458	0.00	0.2	1458	0.0	0.0	29.9	0.1	14.1	14.3	1	0	1	0
r205	1458	1458	0.00	0.2	1458	0.0	0.0	31.2	0.0	16.1	13.9	1	0	1	0
r206	1458	1458	0.00	0.1	1458	0.0	0.0	41.1	0.1	20.1	19.7	1	0	1	0
r207	1458	1458	0.00	0.2	1458	0.0	0.0	41.7	0.1	16.9	23.4	1	0	1	0
r208	1458	1458	0.00	0.2	1458	0.0	0.0	54.7	0.0	22.4	30.8	1	0	1	0
r209	1458	1458	0.00	0.2	1458	0.0	0.0	35.9	0.1	12.8	21.9	1	0	1	0
r210	1458	1458	0.00	0.2	1458	0.0	0.0	31.1	0.1	15.5	14.0	1	0	1	0
r211	1458	1458	0.00	0.2	1458	0.0	0.0	26.0	0.1	13.9	10.7	1	0	1	0
rc201	1724	1724	0.00	0.2	1724	0.0	0.0	29.1	0.3	20.4	7.6	1	0	1	0
rc202	1724	1724	0.00	0.2	1724	0.0	0.0	33.6	0.5	21.6	10.6	1	0	1	0
rc203	1724	1724	0.00	0.2	1724	0.0	0.0	33.4	0.3	17.0	15.1	1	0	1	0
rc204	1724	1724	0.00	0.2	1724	0.0	0.0	33.8	0.2	22.0	10.7	1	0	1	0
rc205	1724	1724	0.00	0.2	1724	0.0	0.0	36.1	0.5	22.4	12.4	1	0	1	0
rc206	1724	1724	0.00	0.2	1724	0.0	0.0	21.1	0.3	12.1	7.9	1	0	1	0
rc207	1724	1724	0.00	0.2	1724	0.0	0.0	40.9	0.3	28.6	11.1	1	0	1	0
rc208	1724	1724	0.00	0.2	1724	0.0	0.0	27.7	0.3	10.0	16.6	1	0	1	0
pr11	657	657	0.00	0.1	657	0.0	0.0	5.7	0.3	3.0	2.2	1	0	1	0
pr12	1132	1120	1.06	477.1	1098	3.0	2.0	997.3	12.6	541.5	442.9	29	28	1	0
pr13	1364	1386	-1.61	672.0	1332	2.3	3.9	1585.5	18.8	904.5	661.7	32	31	1	0
pr14	1670	1651	1.14	783.2	1536	8.0	7.0	2289.7	25.1	1328.9	935.1	29	28	1	0
pr15	1958	2065	-5.46	1161.7	1886	3.7	8.7	3305.8	26.1	1930.6	1347.9	47	46	1	0
pr16	2065	2017	2.32	1183.8	1874	9.2	7.1	3712.9	26.4	2191.0	1494.6	53	51	2	0
pr17	933	934	-0.11	332.8	917	1.7	1.8	687.0	2.4	394.5	290.0	25	0	25	0
pr18	1525	1539	-0.92	559.5	1348	11.6	12.4	1598.8	4.9	945.9	647.6	12	0	12	0
pr19	1723	1750	-1.57	919.4	1518	11.9	13.3	2606.8	13.0	1517.4	1075.8	32	30	2	0
pr20	2037	2062	-1.23	1196.6	1919	5.8	6.9	3902.0	32.3	2361.8	1507.0	61	59	2	0

9.3 Detailed results of using Multi-Threading (Extension)

In this appendix, the detailed results when using threads and multiple runs are given. The seeds that have been used for the different runs are 1, 2, 3, 4, and 5. In the second column, the best profit of these five runs is given. The third column gives the gap of this profit with respect to the profit found while executing one run. The results of only using one run can be found in Table 17. The fourth and the fifth column provides information on the gap of best profit with respect to the best known solution (BKS) and the solutions reported by Hu and Lim [13].

For eighteen instances, the solution quality increased, for nine instances the solution quality remained equal and for two instances the solution quality decreased. Moreover, for two instances (*rc102* and *rc104*) we obtained a better solution than the best known solution. A reason for an increase in the solution quality is that we are now using different random seeds which result in different, and thus possibly better, solutions.

Table 19: Detailed results of using threads and five runs

Name	BKS	BP	Gap (%) One Run	Gap (%) BKS	Gap (%) SVHL
c101	1020	1020	-4.08	0.00	0.00
c102	1150	1150	0.00	0.00	0.00
c103	1200	1200	0.00	0.00	0.83
c105	1070	1060	0.00	0.93	0.00
c106	1080	1070	0.00	0.93	0.93
c107	1120	1120	0.00	0.00	0.00
c108	1130	1120	-3.70	0.88	0.88
c109	1190	1140	-0.88	4.20	4.20
r101	611	608	0.00	0.49	0.00
r102	843	837	-3.33	0.71	0.00
r103	926	913	-4.22	1.40	1.62
r104	972	964	-0.94	0.82	0.52
r105	778	772	0.00	0.77	0.77
r106	905	904	-1.57	0.11	0.22
r107	945	940	-4.33	0.53	1.05
r108	994	986	-3.14	0.80	0.80
r109	885	880	-2.44	0.56	0.56
r110	914	893	1.87	2.30	2.40
r111	949	944	-3.06	0.53	0.84
r112	971	966	-0.73	0.51	0.10
rc101	811	808	0.00	0.37	0.00
rc102	908	909	-0.78	-0.11	-1.11
rc103	970	970	-0.31	0.00	0.41
rc104	1059	**1063**	-3.10	-0.38	0.09
rc105	875	875	0.00	0.00	0.00
rc106	909	**909**	-0.89	0.00	0.00
rc107	980	979	-0.51	0.10	0.10
rc108	1025	1020	-0.49	0.49	0.00
pr07	876	866	0.12	1.14	0.69

9.4 Detailed results of changing the POOL size (Extension)

In this appendix, the detailed results for using different sizes of *POOL* are given. The seeds that have been used for the different runs are 1,2, and 3. In the second column, the average profit for the three runs is given, with its average gap to the solutions of Hu and Lim [13] (SVHL) in the next column. Column four and five contain the best profit found and its gap with respect to SVHL. Column six provides information on the average computation times, which is split up in column seven and eight to the average time used for the RR-component and the average time required for the changes in *POOL*, respectively. Column nine contains information on the percentage of computation time that is devoted to the changes in *POOL* and the RR-component. The last two columns tells us how many changes were needed from the starting solution to derive at the final solution.

Table 20: Detailed results of using $S_{pool} = 0$

Name	AP	Gap (%)	BP	Gap (%)	AT (s)	AT RR (s)	AT Pool (s)	Time (%)	Changes (#)	RR chosen
c101	983.33	3.59	990.00	2.94	206.37	0.00	0.00	0.00	4.67	0.00
c102	1073.33	6.67	1120.00	2.61	303.39	0.00	0.00	0.00	7.00	0.00
c105	1023.33	4.36	1030.00	3.74	230.27	0.00	0.00	0.00	3.67	0.00
c106	1020.00	5.56	1020.00	5.56	236.88	0.00	0.00	0.00	4.33	0.00
c107	1090.00	2.68	1100.00	1.79	254.65	0.00	0.00	0.00	7.00	0.00
c108	1070.00	5.31	1080.00	4.42	266.98	0.00	0.00	0.00	7.00	0.00
c109	1110.00	6.72	1120.00	5.88	296.82	0.00	0.00	0.00	8.67	0.00
r101	591.00	3.27	598.00	2.13	118.76	0.00	0.00	0.00	1.67	0.00
r102	790.00	6.29	816.00	3.20	198.23	0.00	0.00	0.00	3.33	0.00
r103	872.33	5.80	886.00	4.32	247.97	0.00	0.00	0.00	8.67	0.00
r104	878.33	9.64	888.00	8.64	281.98	0.00	0.00	0.00	14.33	0.00
r105	755.00	2.96	771.00	0.90	175.33	0.00	0.00	0.00	4.67	0.00
r106	856.67	5.34	868.00	4.09	218.53	0.00	0.00	0.00	8.33	0.00
r107	864.67	8.50	878.00	7.09	234.47	0.00	0.00	0.00	10.00	0.00
r108	885.33	10.93	911.00	8.35	266.16	0.00	0.00	0.00	10.67	0.00
r109	812.33	8.21	837.00	5.42	201.14	0.00	0.00	0.00	9.33	0.00
r110	861.67	5.73	879.00	3.83	229.67	0.00	0.00	0.00	12.00	0.00
r111	885.67	6.67	900.00	5.16	256.04	0.00	0.00	0.00	10.67	0.00
rc101	798.33	1.56	808.00	0.37	173.89	0.00	0.00	0.00	3.67	0.00
rc102	864.67	4.77	882.00	2.86	228.52	0.00	0.00	0.00	4.00	0.00
rc103	925.00	4.64	943.00	2.78	269.06	0.00	0.00	0.00	8.67	0.00
rc105	855.00	2.29	864.00	1.26	201.42	0.00	0.00	0.00	6.33	0.00
rc106	862.33	5.13	885.00	2.64	199.46	0.00	0.00	0.00	6.33	0.00
rc107	956.33	2.41	965.00	1.53	233.55	0.00	0.00	0.00	7.33	0.00
rc108	957.33	6.60	963.00	6.05	255.06	0.00	0.00	0.00	6.33	0.00

Table 21: Detailed results of using $S_{pool} = 500$

Name	AP	Gap (%)	BP	Gap (%)	AT (s)	AT RR (s)	AT Pool (s)	Time (%)	Changes (#)	RR chosen
c101	996.67	2.29	1010.00	0.98	211.32	1.42	9.69	5.25	3.67	0.00
c102	1120.00	2.61	1150.00	0.00	281.82	4.15	10.69	5.27	8.67	5.33
c105	1046.67	2.18	1060.00	0.93	233.02	1.50	9.88	4.88	5.67	0.00
c106	1043.33	3.40	1050.00	2.78	235.64	1.43	10.25	4.96	5.67	0.00
c107	1090.00	2.68	1100.00	1.79	259.23	1.47	10.28	4.53	7.00	0.00
c108	1090.00	3.54	1100.00	2.65	259.54	1.54	9.87	4.40	7.00	0.00
c109	1130.00	5.04	1140.00	4.20	285.73	1.43	9.63	3.87	9.00	0.00
r101	600.67	1.69	604.00	1.15	127.00	1.25	9.54	8.50	2.33	0.33
r102	808.33	4.11	823.00	2.37	215.33	1.30	9.20	4.87	6.00	0.00
r103	880.67	4.90	891.00	3.78	249.61	1.29	9.70	4.40	10.00	0.00
r104	935.00	3.81	957.00	1.54	275.18	4.79	10.11	5.41	18.33	11.33
r105	763.33	1.89	772.00	0.77	190.15	1.40	9.63	5.80	5.00	0.00
r106	872.33	3.61	891.00	1.55	230.40	1.27	8.96	4.44	12.33	0.00
r107	875.33	7.37	890.00	5.82	246.18	1.33	9.86	4.54	12.67	0.00
r108	910.33	8.42	950.00	4.43	278.02	2.80	10.29	4.71	13.00	5.33
r109	860.00	2.82	876.00	1.02	219.40	2.28	9.76	5.49	9.33	3.33
r110	863.33	5.54	867.00	5.14	242.61	1.45	10.21	4.81	12.33	0.67
r111	908.67	4.25	949.00	0.00	293.46	3.01	12.32	5.23	12.67	4.33
rc101	808.00	0.37	808.00	0.37	180.65	3.59	11.01	8.08	2.33	1.33
rc102	894.33	1.51	908.00	0.00	223.74	5.06	10.80	7.09	6.33	4.00
rc103	967.67	0.24	968.00	0.21	242.36	4.50	10.07	6.01	8.67	7.33
rc105	865.67	1.07	867.00	0.91	204.59	1.32	9.95	5.51	6.67	2.33
rc106	906.33	0.29	909.00	0.00	212.18	3.74	10.66	6.78	5.33	4.33
rc107	966.67	1.36	978.00	0.20	238.44	3.20	10.13	5.59	8.00	5.00
rc108	1017.67	0.72	1023.00	0.20	257.03	4.47	10.35	5.76	7.33	6.33

Table 22: Detailed results of using $S_{pool} = 1000$

Name	AP	Gap (%)	BP	Gap (%)	AT (s)	AT RR (s)	AT Pool (s)	Time (%)	Changes (#)	RR chosen
c101	1020.00	0.00	1020.00	0.00	238.88	10.02	26.36	15.23	6.67	5.00
c102	1150.00	0.00	1150.00	0.00	294.78	11.26	26.70	12.88	9.67	8.00
c105	1046.67	2.18	1060.00	0.93	251.29	5.81	26.11	12.70	5.00	1.33
c106	1056.67	2.16	1070.00	0.93	262.16	12.11	27.63	15.16	6.67	5.33
c107	1110.00	0.89	1120.00	0.00	275.21	11.67	27.28	14.15	7.33	5.67
c108	1103.33	2.36	1120.00	0.88	283.34	9.15	25.39	12.19	7.67	3.33
c109	1133.33	4.76	1140.00	4.20	301.98	3.10	24.46	9.13	8.67	0.00
r101	608.00	0.49	608.00	0.49	160.74	12.20	25.66	23.55	3.33	2.33
r102	818.33	2.93	823.00	2.37	232.08	2.65	23.32	11.19	5.33	0.33
r103	888.00	4.10	913.00	1.40	268.41	6.22	24.61	11.49	10.00	3.33
r104	949.33	2.33	964.00	0.82	310.94	14.16	25.85	12.87	15.33	14.00
r105	770.67	0.94	772.00	0.77	201.37	2.61	24.01	13.22	5.33	1.33
r106	890.67	1.58	904.00	0.11	248.99	7.25	23.80	12.47	13.00	4.00
r107	894.67	5.33	939.00	0.63	269.13	5.80	25.23	11.53	11.67	4.67
r108	954.67	3.96	979.00	1.51	304.50	12.47	26.66	12.85	14.00	12.33
r109	873.33	1.32	880.00	0.56	234.76	8.65	24.33	14.05	13.33	8.67
r110	881.67	3.54	893.00	2.30	261.10	6.35	25.66	12.26	10.67	6.00
r111	906.33	4.50	926.00	2.42	286.43	7.14	28.52	12.45	10.67	3.67
rc101	808.00	0.37	808.00	0.37	202.07	7.08	26.19	16.46	2.33	1.33
rc102	903.67	0.48	909.00	-0.11	239.07	11.99	25.32	15.61	6.33	4.67
rc103	962.67	0.76	970.00	0.00	255.40	9.66	24.65	13.43	9.00	8.00
rc105	875.00	0.00	875.00	0.00	230.55	7.42	26.54	14.73	4.00	3.00
rc106	903.33	0.62	908.00	0.11	234.00	8.01	25.95	14.51	5.33	4.33
rc107	975.33	0.48	979.00	0.10	253.63	9.18	25.60	13.71	7.33	6.33
rc108	1017.33	0.75	1019.00	0.59	279.29	10.11	25.40	12.71	9.33	8.00

Table 23: Detailed results of using $S_{pool} = 1500$

Name	AP	Gap (%)	BP	Gap (%)	AT (s)	AT RR (s)	AT Pool (s)	Time (%)	Changes (#)	RR chosen
c101	1020.00	0.00	1020.00	0.00	252.02	10.89	42.05	21.01	6.00	4.67
c102	1146.67	0.29	1150.00	0.00	308.39	12.37	43.20	18.02	8.00	6.00
c105	1060.00	0.93	1060.00	0.93	281.95	18.40	44.57	22.33	6.00	4.33
c106	1063.33	1.54	1080.00	0.00	280.20	15.91	45.05	21.76	6.33	5.00
c107	1106.67	1.19	1110.00	0.89	302.61	19.05	44.31	20.94	7.00	4.67
c108	1120.00	0.88	1120.00	0.88	294.07	12.62	41.54	18.42	7.33	6.33
c109	1173.33	1.40	1190.00	0.00	334.40	17.70	41.97	17.84	12.33	10.67
r101	608.00	0.49	608.00	0.49	178.30	14.95	39.29	30.42	3.33	2.33
r102	821.00	2.61	828.00	1.78	250.05	9.33	38.82	19.26	5.33	2.67
r103	898.00	3.02	917.00	0.97	294.57	15.01	40.41	18.81	11.00	7.67
r104	943.00	2.98	970.00	0.21	330.31	21.67	41.27	19.05	15.00	14.00
r105	775.33	0.34	778.00	0.00	233.26	17.98	41.47	25.49	5.67	4.67
r106	894.33	1.18	898.00	0.77	272.37	22.02	37.42	21.83	8.67	7.00
r107	906.33	4.09	939.00	0.63	294.45	15.07	41.79	19.31	12.33	8.33
r108	955.33	3.89	969.00	2.52	322.07	19.24	42.08	19.04	19.67	18.67
r109	874.33	1.21	885.00	0.00	259.97	19.64	39.33	22.68	10.67	9.67
r110	881.67	3.54	892.00	2.41	287.05	12.97	43.07	19.52	13.00	9.00
r111	922.33	2.81	937.00	1.26	293.13	15.74	43.61	20.25	8.00	4.00
rc101	808.00	0.37	808.00	0.37	240.46	14.28	42.85	23.76	2.67	1.67
rc102	904.33	0.40	909.00	-0.11	260.87	21.44	39.73	23.45	5.67	4.33
rc103	961.33	0.89	967.00	0.31	276.02	15.70	39.37	19.95	7.33	6.33
rc105	875.00	0.00	875.00	0.00	250.20	14.60	41.43	22.39	4.00	2.67
rc106	909.00	0.00	909.00	0.00	258.77	14.44	41.68	21.69	5.67	4.67
rc107	972.67	0.75	978.00	0.20	282.21	16.46	41.29	20.46	7.33	6.33
rc108	1020.00	0.49	1022.00	0.29	301.20	16.43	40.07	18.76	10.00	9.00

Table 24: Detailed results of using $S_{pool} = 2000$

Name	AP	Gap (%)	BP	Gap (%)	AT (s)	AT RR (s)	AT Pool (s)	Time (%)	Changes (#)	RR chosen
c101	1020.00	0.00	1020.00	0.00	274.85	14.60	58.38	26.55	5.67	4.00
c102	1146.67	0.29	1150.00	0.00	330.31	18.40	59.46	23.57	8.67	7.67
c105	1056.67	1.25	1060.00	0.93	300.62	18.57	62.67	27.03	6.33	4.33
c106	1060.00	1.85	1060.00	1.85	310.44	19.05	64.95	27.06	6.33	5.33
c107	1103.33	1.49	1110.00	0.89	322.18	23.19	57.50	25.04	8.00	6.33
c108	1120.00	0.88	1120.00	0.88	356.97	27.29	60.24	24.52	8.00	6.67
c109	1163.33	2.24	1170.00	1.68	349.59	21.25	57.37	22.49	10.33	8.67
r101	608.00	0.49	608.00	0.49	194.92	18.45	52.22	36.26	3.33	2.33
r102	824.00	2.25	837.00	0.71	276.13	12.69	54.23	24.24	5.00	2.00
r103	891.67	3.71	906.00	2.16	314.64	18.93	55.62	23.69	6.67	3.00
r104	952.00	2.06	955.00	1.75	350.93	26.84	55.51	23.47	16.33	15.33
r105	773.33	0.60	778.00	0.00	280.92	23.90	62.58	30.79	5.67	4.33
r106	899.33	0.63	904.00	0.11	295.06	28.35	52.37	27.36	12.33	10.67
r107	933.00	1.27	945.00	0.00	321.41	25.89	58.51	26.26	13.00	12.00
r108	948.67	4.56	961.00	3.32	341.00	25.47	57.21	24.25	16.67	15.67
r109	877.00	0.90	885.00	0.00	282.31	28.91	53.13	29.06	9.00	8.00
r110	880.00	3.72	887.00	2.95	303.16	18.71	57.91	25.27	11.33	7.33
r111	936.67	1.30	939.00	1.05	320.90	26.59	57.67	26.26	11.33	9.67
rc101	808.00	0.37	808.00	0.37	245.53	22.50	54.52	31.37	2.33	1.33
rc102	906.00	0.22	908.00	0.00	281.76	27.46	54.19	28.98	7.00	5.33
rc103	960.00	1.03	970.00	0.00	294.17	21.53	52.60	25.20	8.33	7.33
rc105	875.00	0.00	875.00	0.00	272.52	19.98	56.24	27.97	3.67	2.33
rc106	903.67	0.59	909.00	0.00	303.32	23.08	59.71	27.30	5.00	4.00
rc107	976.00	0.41	979.00	0.10	301.30	22.70	56.11	26.16	7.33	6.33
rc108	1016.33	0.85	1018.00	0.68	327.35	22.20	54.52	23.44	9.67	8.67

Table 25: Detailed results of using $S_{pool} = 3000$

Name	AP	Gap (%)	BP	Gap (%)	AT (s)	AT RR (s)	AT Pool (s)	Time (%)	Changes (#)	RR chosen
c101	1020.00	0.00	1020.00	0.00	324.14	23.13	93.78	36.07	6.33	4.67
c102	1146.67	0.29	1150.00	0.00	369.91	27.55	91.54	32.19	10.00	9.00
c105	1060.00	0.93	1060.00	0.93	352.00	35.92	96.21	37.54	6.00	4.33
c106	1060.00	1.85	1070.00	0.93	350.33	30.04	99.75	37.05	6.00	5.00
c107	1110.00	0.89	1120.00	0.00	393.46	48.02	92.11	35.62	7.33	6.33
c108	1116.67	1.18	1120.00	0.88	385.46	37.23	92.18	33.57	8.00	6.67
c109	1166.67	1.96	1170.00	1.68	402.57	39.81	87.32	31.58	9.67	8.33
r101	608.00	0.49	608.00	0.49	229.31	26.68	78.57	45.90	3.00	2.33
r102	828.67	1.70	837.00	0.71	335.36	31.22	86.84	35.20	6.67	5.00
r103	911.00	1.62	914.00	1.30	372.21	37.86	90.21	34.41	10.67	9.00
r104	953.67	1.89	963.00	0.93	400.02	43.00	86.37	32.34	14.33	13.33
r105	776.00	0.26	778.00	0.00	305.23	41.46	87.64	42.29	5.67	4.67
r106	894.33	1.18	898.00	0.77	333.17	40.77	79.03	35.96	9.33	8.33
r107	932.00	1.38	942.00	0.32	368.89	39.16	90.24	35.08	11.67	10.67
r108	953.67	4.06	967.00	2.72	387.60	36.03	90.07	32.53	17.67	16.33
r109	883.00	0.23	885.00	0.00	326.64	43.43	82.54	38.56	11.67	10.33
r110	889.00	2.74	894.00	2.19	358.18	38.81	91.80	36.47	11.67	10.33
r111	940.67	0.88	946.00	0.32	367.15	42.94	90.58	36.37	10.67	9.33
rc101	808.00	0.37	808.00	0.37	286.42	39.00	82.82	42.53	2.33	1.33
rc102	906.67	0.15	909.00	-0.11	333.11	49.46	83.00	39.77	7.00	5.67
rc103	958.00	1.24	968.00	0.21	334.12	34.82	82.41	35.09	7.67	6.67
rc105	875.00	0.00	875.00	0.00	314.81	32.47	85.08	37.34	4.33	3.00
rc106	903.67	0.59	909.00	0.00	319.62	34.27	84.05	37.02	4.67	3.67
rc107	972.67	0.75	978.00	0.20	343.79	35.56	86.68	35.56	6.67	5.33
rc108	1016.00	0.88	1020.00	0.49	371.77	37.02	84.65	32.73	9.33	8.00

Table 26: Detailed results of using $S_{pool} = 4000$

Name	AP	Gap (%)	BP	Gap (%)	AT (s)	AT RR (s)	AT Pool (s)	Time (%)	Changes (#)	RR chosen
c101	1020.00	0.00	1020.00	0.00	365.98	29.75	131.26	43.99	5.33	3.67
c102	1146.67	0.29	1150.00	0.00	421.24	38.55	128.69	39.70	9.33	8.33
c105	1060.00	0.93	1060.00	0.93	400.34	45.26	135.15	45.06	6.33	5.00
c106	1060.00	1.85	1060.00	1.85	409.89	53.43	134.28	45.80	6.33	5.33
c107	1113.33	0.60	1120.00	0.00	415.83	42.19	126.01	40.45	7.00	6.00
c108	1126.67	0.29	1140.00	-0.88	439.83	52.51	131.21	41.77	8.67	7.67
c109	1173.33	1.40	1180.00	0.84	494.74	55.16	138.49	39.14	10.00	9.00
r101	608.00	0.49	608.00	0.49	268.78	35.50	108.12	53.44	3.33	2.33
r102	835.33	0.91	837.00	0.71	370.41	53.03	114.58	45.25	7.33	6.00
r103	915.67	1.12	922.00	0.43	417.78	54.61	120.11	41.82	9.67	8.67
r104	963.33	0.89	972.00	0.00	448.15	56.26	121.78	39.73	18.00	17.00
r105	773.33	0.60	778.00	0.00	366.35	51.58	128.81	49.24	4.67	3.67
r106	898.67	0.70	905.00	0.00	375.74	53.40	109.41	43.33	12.00	11.00
r107	937.67	0.78	939.00	0.63	422.30	53.25	124.64	42.12	12.67	11.00
r108	950.33	4.39	956.00	3.82	440.23	53.00	122.62	39.89	15.33	14.33
r109	884.67	0.04	885.00	0.00	367.18	54.92	112.26	45.53	12.00	10.67
r110	891.00	2.52	893.00	2.30	403.11	52.43	125.82	44.22	11.33	10.33
r111	939.67	0.98	943.00	0.63	415.91	53.55	125.43	43.04	11.00	10.00
rc101	808.00	0.37	808.00	0.37	333.77	53.55	115.20	50.56	2.33	1.33
rc102	906.00	0.22	908.00	0.00	377.71	62.30	114.64	46.85	6.33	5.00
rc103	958.00	1.24	968.00	0.21	373.91	46.38	110.97	42.08	6.67	5.33
rc105	875.00	0.00	875.00	0.00	359.89	44.87	116.40	44.81	4.00	2.00
rc106	904.00	0.55	909.00	0.00	360.25	48.49	114.66	45.29	5.67	4.33
rc107	971.67	0.85	975.00	0.51	391.59	48.98	119.80	43.10	6.00	5.00
rc108	1016.00	0.88	1017.00	0.78	418.96	49.48	116.67	39.66	9.67	8.67

Table 27: Detailed results of using $S_{pool} = 5000$

Name	AP	Gap (%)	BP	Gap (%)	AT (s)	AT RR (s)	AT Pool (s)	Time (%)	Changes (#)	RR chosen
c101	1020.00	0.00	1020.00	0.00	502.27	57.66	211.99	53.69	6.00	4.67
c102	1146.67	0.29	1150.00	0.00	470.45	46.98	170.52	46.23	9.00	8.00
c105	1056.67	1.25	1060.00	0.93	460.93	66.98	172.97	52.06	5.67	4.00
c106	1060.00	1.85	1060.00	1.85	449.34	52.93	177.06	51.18	6.00	5.00
c107	1106.67	1.19	1110.00	0.89	499.53	63.74	175.90	47.97	6.33	4.33
c108	1126.67	0.29	1130.00	0.00	470.75	67.49	161.02	48.54	9.00	7.67
c109	1176.67	1.12	1180.00	0.84	509.29	68.12	163.82	45.54	10.00	8.67
r101	608.00	0.49	608.00	0.49	315.41	48.57	142.09	60.45	3.33	2.33
r102	836.33	0.79	837.00	0.71	426.93	67.88	153.77	51.92	8.67	7.00
r103	913.33	1.37	922.00	0.43	465.77	68.41	157.63	48.53	11.00	9.67
r104	951.67	2.09	957.00	1.54	506.06	77.23	156.39	46.16	12.33	11.33
r105	776.00	0.26	778.00	0.00	378.59	64.46	148.75	56.32	5.33	4.33
r106	896.00	0.99	899.00	0.66	433.04	72.75	145.42	50.38	10.67	9.33
r107	941.00	0.42	950.00	-0.53	477.31	69.10	163.98	48.83	15.67	14.33
r108	950.00	4.43	957.00	3.72	497.86	68.75	162.23	46.39	15.67	14.67
r109	882.67	0.26	885.00	0.00	510.67	79.91	142.80	43.61	12.67	11.67
r110	884.33	3.25	886.00	3.06	477.80	70.92	173.00	51.05	9.00	8.00
r111	940.67	0.88	946.00	0.32	484.00	72.80	169.39	50.04	13.00	12.00
rc101	808.00	0.37	808.00	0.37	439.17	83.77	175.95	59.14	2.33	1.33
rc102	906.33	0.18	909.00	-0.11	624.18	99.99	146.01	39.41	7.67	6.33
rc103	954.67	1.58	967.00	0.31	423.96	63.08	144.90	49.06	7.33	6.33
rc105	875.00	0.00	875.00	0.00	539.26	69.01	144.61	39.61	4.00	2.67
rc106	908.67	0.04	909.00	0.00	410.35	69.46	147.76	52.93	6.67	5.67
rc107	975.00	0.51	980.00	0.00	445.77	65.79	155.83	49.72	6.33	5.33
rc108	1012.67	1.20	1020.00	0.49	631.70	75.51	147.34	35.28	9.33	8.33

9.5 Routes of the new Best Known Solutions

For two instances we obtained new best known solutions. There corresponding solutions are given below.

c108
Objective value of 1140
$r_1 = \{0, 12, 14, 11, 15, 16, 9, 10, 60, 7, 2, 0\}$
$r_2 = \{0, 65, 82, 69, 88, 79, 6, 4, 5, 3, 1, 70, 0\}$
$r_3 = \{0, 42, 39, 36, 40, 38, 41, 43, 35, 37, 96, 0\}$
$r_4 = \{0, 48, 21, 23, 19, 22, 57, 83, 66, 91, 94, 80, 0\}$

rc102
Objective value of 909
$r_1 = \{0, 98, 95, 81, 70, 82, 84, 85, 88, 89, 91, 47, 0\}$
$r_2 = \{0, 13, 18, 17, 15, 16, 93, 97, 100, 99, 2, 1, 75, 0\}$
$r_3 = \{0, 33, 32, 25, 46, 37, 38, 39, 36, 34, 50, 52, 0\}$
$r_4 = \{0, 57, 54, 63, 74, 56, 58, 60, 6, 22, 21, 49, 0\}$

10 References

[1] I-Ming Chao, Bruce L Golden, and Edward A Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475–489, 1996.

[2] I-Ming Chao, Bruce L Golden, and Edward A Wasil. The team orienteering problem. *European journal of operational research*, 88(3):464–474, 1996.

[3] N Christofides, A Mingozzi, P Toth, and C Sandi. Combinatorial optimization. 1979.

[4] Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997.

[5] Tunchan Cura. An artificial bee colony algorithm approach for the team orienteering problem with time windows. *Computers & Industrial Engineering*, 74:270–290, 2014.

[6] Racha El-Hajj, Aziz Moukrim, B Chebaro, and M Kobeissi. A column generation algorithm for the team orienteering problem with time windows. In *The 45th International Conference on Computers & Industrial Engineering (CIE45)*, 2015.

[7] Matteo Fischetti, Juan Jose Salazar Gonzalez, and Paolo Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–148, 1998.

[8] Luca Maria Gambardella, Roberto Montemanni, and Dennis Weyland. Coupling ant colony systems with strong local searches. *European Journal of Operational Research*, 220(3):831–843, 2012.

[9] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, 20(3):291–328, 2014.

[10] Bruce L Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval research logistics*, 34(3):307–318, 1987.

[11] Aldy GUNAWAN, LAU Hoong Chuin, and LU Kun. The latest best known solutions for the team orienteering problem with time windows (toptw) benchmark instances. 2015.

[12] Aldy Gunawan, Hoong Chuin Lau, and Kun Lu. Sails: Hybrid algorithm for the team orienteering problem with time windows. In *Proceedings of the 7th Multidisciplinary International Scheduling Conference (MISTA 2015)*, 2015.

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

[14] Marisa G Kantor and Moshe B Rosenwein. The orienteering problem with time windows. *Journal of the Operational Research Society*, pages 629–635, 1992.

[15] Nacima Labadie, Jan Melechovskỳ, and Roberto Wolfler Calvo. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *Journal of Heuristics*, 17(6):729–753, 2011.

[16] Nacima Labadie, Renata Mansini, Jan Melechovskỳ, and Roberto Wolfler Calvo. The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, 220(1):15–27, 2012.

[17] Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Discrete applied mathematics*, 26(2-3):193–207, 1990.

[18] Shih-Wei Lin and F Yu Vincent. A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 217(1):94–107, 2012.

[19] R Montemanni and LM Gambardella. An ant colony system for team orienteering problems with time windows. *Foundations of computing and Decision Sciences*, 34:287–306, 2009.

[20] R Montemanni, D Weyland, and LM Gambardella. 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. IEEE, 2011.

[21] Giovanni Righini and Matteo Salani. 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, 2009.

[22] Michael Schilde, Karl F Doerner, Richard F Hartl, and Guenter Kiechle. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3):179–201, 2009.

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

[24] Fabien Tricoire, Martin Romauch, Karl F Doerner, and Richard F Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37(2):351–367, 2010.

[25] Theodore Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, pages 797–809, 1984.

[26] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290, 2009.

[27] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.