# Accepted Manuscript

## Asia-Pacific Journal of Operational Research

| | |
|---|---|
| Article Title: | Efficient Cluster-based Heuristics for the Team Orienteering Problem with Time Windows |
| Author(s): | Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou |
| DOI: | 10.1142/S0217595919500015 |
| Accepted: | 17 October 2018 |
| To be cited as: | Damianos Gavalas *et al.*, Efficient Cluster-based Heuristics for the Team Orienteering Problem with Time Windows, *Asia-Pacific Journal of Operational Research*, doi: 10.1142/S0217595919500015 |
| Link to final version: | https://doi.org/10.1142/S0217595919500015 |

Asia-Pacific Journal of Operational Research


# Efficient Cluster-based Heuristics for the Team Orienteering Problem with Time Windows

Damianos Gavalas

*Department of Product & Systems Design Engineering, University of the Aegean, Syros, 84100, Greece and Computer Technology Institute and Press 'Diophantus' (CTI), Patras, Greece*
*dgavalas@aegean.gr*

Charalampos Konstantopoulos*

*Department of Informatics, University of Piraeus, Karaoli & Dimitriou 80 Piraeus, 18534, Greece and Computer Technology Institute and Press 'Diophantus' (CTI), Patras, Greece*
*konstant@unipi.gr*

Konstantinos Mastakas

*School of Applied Mathematical and Physical Sciences, National Technical University of Athens, Heroon Polytechniou 9 Zografou, 15780, Greece and Computer Technology Institute and Press 'Diophantus' (CTI), Patras, Greece*
*kmast@math.ntua.gr*

Grammati Pantziou

*Department of Informatics and Computer Engineering, University of West Attica, Agiou Spyridonos Aigaleo, 12243, Greece and Computer Technology Institute and Press 'Diophantus' (CTI), Patras, Greece*
*pantziou@uniwa.gr*

Received (Day Month Year)
Revised (Day Month Year)

In the Team Orienteering Problem with Time Windows (TOPTW), a variant of the Vehicle Routing Problem with Profits, a set of locations is given each associated with a profit, a visiting time and a time window. The aim is to maximize the overall profit collected by a number of routes, while the duration of each route must not exceed a given time budget. TOPTW is NP-hard and is typically used to model the Tourist Trip Design Problem. The latter deals with deriving near optimal multiple-day tours for tourists visiting a destination with several points of interest (POIs). The most efficient known heuristic approach to TOPTW which yields the best solution quality versus execution time, is based on Iterated Local Search (ILS). However, the ILS algorithm treats each node separately, hence it tends to overlook highly profitable areas of nodes situated far from the current solution, considering them too time-expensive to visit. We propose two cluster-based extensions to ILS addressing the aforementioned weakness by grouping

*Corresponding author

nodes on separate clusters (based on geographical criteria), thereby making visits to such nodes more attractive. Our approaches improve ILS with respect to solutions quality and execution time as evidenced by experimental tests exercised on both existing and new TTDP-oriented benchmark instances.

*Keywords*: Team Orienteering Problem with Time Windows; Iterated Local Search; Clustering; Tourist Trip Design Problem; Point of Interest.

## 1. Introduction

The Team Orienteering Problem (TOP) (Chao *et al.*, 1996) is a combinatorial routing problem defined as follows: given (i) a set of nodes (locations) each associated with a profit and a visiting time, (ii) a travel time between each pair of nodes, (iii) a time budget $B$, and (iv) an integer $k$, the objective is to find $k$ disjoint paths (or routes) starting from a node $s$ and ending at a node $t$, each with total duration bounded by $B$, that maximize the overall profit collected by the visited nodes. The TOP, originally known in the literature as Multiple Tour Maximum Collection Problem (MTMCP) (Butt and Cavalier, 1994), is a variant of the Vehicle Routing Problem with Profits (VRPP) (Archetti *et al.*, 2007; Gavalas *et al.*, 2014). In VRPP, a variant of the classical VRP, visiting the whole set of nodes is not compulsory; profit is collected when visiting a node, while the collection of profit is distributed over several vehicles. The TOP with Time Windows (TOPTW) introduced by Vansteenwegen (2008) extends TOP allowing the visit at nodes to take place only during specified time windows. TOPTW is NP-hard and APX-hard (e.g. see Golden *et al.* (1987), Laporte and Martello (1990)). Exact solutions for TOPTW can only be obtained for instances with very restricted number of locations (e.g. see the work by Li and Hu (2011), which is used on networks of up to 30 nodes). The literature relevant to TOPTW involves mainly heuristic approaches based on simulated annealing (Lin and Yu, 2012), iterated local search (Vansteenwegen *et al.*, 2009) and ant colony systems (ACS) (Montemanni and Gambardella, 2009).

The most common application of the TOPTW is to model the Tourist Trip Design Problem (TTDP) (Vansteenwegen and Van Oudheusden, 2007), a route-planning problem which deals with deriving near optimal multiple-day tours for tourists visiting a destination with several points of interest (POIs). Different versions of TTDP have been studied in the literature. TOPTW has been extensively used to model a version of TTDP that considers the following input data:

- A set of candidate POIs, each associated with the following attributes: (i) a location (i.e. geographical coordinates), (ii) time windows (i.e. opening hours for each day of the week), (iii) a “profit” value, calculated as a weighted function of the objective and subjective importance of the POI (subjectivity refers to the users’ individual preferences and interests on specific POI categories) and (iv) a visiting time (i.e. the anticipated duration of visit of a user at the POI, derived from the average anticipated duration and the user’s potential interest for that particular POI).

- The travel time among POIs, based on the topological distance between a pair of POIs.
- The number $k$ of routes that must be generated, based upon the period of stay (number of days) of the user at the tourist destination.
- The daily time budget $B$ that a tourist wishes to spend on visiting sights; the overall daily route length should be kept below $B$.

By solving the TTDP we expect to derive $k$ routes (typically starting and ending at the tourist's accommodation location) each of length at most $B$, that maximize the overall collected profit. Therefore, a TTDP solution should feature POI recommendations that match tourist preferences and near-optimal feasible route scheduling. The TTDP is typically dealt with online web and mobile applications with strict execution time restrictions (Souffriau and Vansteenwegen, 2010; Vansteenwegen *et al.*, 2011b). Hence, only highly efficient TOPTW heuristic approaches are eligible for TTDP solvers. The most efficient known heuristic for TOPTW is based on Iterated Local Search (ILS) (Vansteenwegen *et al.*, 2009, 2011a). ILS represents a fair compromise with respect to computational speed versus solutions quality, thereby being suitable for real-time TTDP applications. However, ILS treats each node separately, thereby commonly overlooking highly profitable areas of nodes situated far from current location, considering them too time-expensive to visit. ILS is also often trapped in areas with isolated high-profit nodes, possibly leaving considerable amount of the overall time budget unused.

Herein, we introduce RCRatio and CSCRoutes, two cluster-based algorithmic approaches for the TOPTW, which address the shortcomings of ILS. The main incentive behind our approaches is to motivate visits to topology areas featuring high density of 'good' candidate nodes (such areas are identified by a geographical clustering method performed offline); the aim is to improve the quality of derived solutions while not sacrificing time efficiency. Furthermore, both our algorithms favor solutions with reduced number of overly long transfers among nodes, which typically require public transportation rides (such transfers are costly and usually less attractive to tourists than short walking transfers). A preliminary version of this work has appeared in (Gavalas *et al.*, 2013).

The remainder of this article is organized as follows: Section 2 overviews algorithmic approaches relevant to the TOPTW. Section 3 presents our novel cluster-based heuristics, while Section 4 discusses the experimental results compiled from executing ILS as well as our algorithms on several test instances. Finally, Section 5 concludes the paper and provides directions for future work.

## 2. Related work

Vansteenwegen *et al.* (2009) proposed the Iterated Local Search (ILS) heuristic for solving the TOPTW problem. ILS defines an "insertion" and a "shake" step. The insertion step adds, one by one, new nodes in the routes of the current solution, ensuring that all following nodes in the routes remain feasible to visit, i.e. their time window constraints are satisfied and the time budget is not violated. The shake step is used to escape from local optima. During this step, one or more nodes are removed in each tour looking for non-included nodes that may either decrease the tour time length or increase the overall collected profit. The ILS heuristic is the fastest known algorithm proposed for TOPTW (Vansteenwegen *et al.*, 2011a).

Montemanni and Gambardella (2009) proposed an ant colony system (ACS) algorithm to derive solutions for a hierarchical generalization of TOPTW, wherein more than the $k$ required routes are constructed. At the expense of the additional overhead, those additional fragments are used to perform exchanges/insertions so as to improve the quality of the $k$ routes. The algorithm comprises two phases: (i) The construction phase where ants are sent out sequentially; when at node $i$, an ant chooses probabilistically the next node $j$ to visit (i.e. to include into the tour) based on the pheromone trail, the profit of $j$ as well as its distance from $i$. (ii) The local search phase performed upon the solutions derived from construction phase, aiming at taking them down to a local optimum. ACS has been shown to obtain high quality results (that is, low average gap to the best known solution) at the expense of prolonged execution time, practically prohibitive for online applications. In (Gambardella *et al.*, 2012) a modified ACS framework (Enhanced ACS) is presented and implemented for the TOPTW to improve the results of ACS.

Labadi et al. in (Labadi *et al.*, 2010) and (Labadi *et al.*, 2011) proposed a method that combines the greedy randomized adaptive search procedure (GRASP) with the evolutionary local search (ELS). GRASP generates independent solutions using a randomized heuristic, which are further improved by a local search procedure. ELS generates multiple copies of a starting solution using a random mutation (perturbation) and then applies a local search on each copy to yield an improved solution. GRASP-ELS derives solutions of comparable quality and requires significantly less computational effort than ACS. Compared to ILS, GRASP-ELS gives better quality solutions at the expense of increased computational complexity.

Tricoire *et al.* (2010) deal with the Multi-Period Orienteering Problem with Multiple Time Windows (MuPOPTW), a generalization of TOPTW, wherein each node may be assigned more than one time window on a given day, while time windows may differ on different days. Both mandatory and optional visits are considered. Two heuristics were developed: a deterministic constructive heuristic which provides a starting solution, and a stochastic local search algorithm, the Variable Neighbourhood Search (VNS) which considers random exchanges between chains of nodes.

Labadi *et al.* (2012) proposed a local search heuristic algorithm for TOPTW by introducing a granular variant to a VNS algorithm (GVNS). In the local search routine, the algorithm tries to replace a segment of a route by higher profit nodes. For that, an assignment problem related to the TOPTW is solved and based on this solution, the algorithm decides which arcs to select.

Lin and Yu (2012) proposed a heuristic algorithm based on simulated annealing (SA) for the TOPTW. At each iteration a neighbouring solution of the current solution is obtained by applying one of the swap, insertion or inversion moves, with equal probability. The new solution is adopted in the case that it is more profitable than the so far best found solution, otherwise it is accepted with a certain probability which is decreasing with increasing profit loss. After applying the above procedure for a certain number of iterations the best solution found so far is further improved by applying local search.

Guibadj and Moukrim (2013) proposed a memetic algorithm (MA) for the TOPTW based on the application of a split procedure to evaluate an individual. Hu and Lim (2014) proposed an iterative three-component heuristic (I3CH) for TOPTW, which iteratively applies a local search procedure, a simulated annealing procedure and a route recombination step. Each route of a solution obtained from the local search and simulated annealing procedure, is inserted into a pool of routes. Then, in the route recombination step, $k$ disjoint routes from the pool with the highest total profit are picked, hence deriving a high quality solution.

So far, performance comparison results among existing TOPTW approaches have been reported in (Guibadj and Moukrim, 2013) and (Hu and Lim, 2014). Guibadj and Moukrim (2013) compared their MA approach against ACS, ILS, VNS, GVNS, GRASP-ELS and the SA algorithm, while (Hu and Lim, 2014) compared IC3H against ILS, GRASP-ELS, ACS. In both cases, the implemented algorithms are compared with respect to quality of derived solutions (i.e. average profit gap from the optimal or best known solution) and execution time for different number of tours $k$ ($k = 1, \cdots, 4$). The performance results have been compiled upon a number of benchmark sets designed by Solomon (1987) and Cordeau *et al.* (1997). A combined interpretation of reported results indicates that the MA approach yields the highest quality solutions closely followed by I3CH, VNS and GRASP-ELS and then by GVNS, SA, ACS and ILS. In all cases, the performance gap achieved by ILS is $\sim 4.5\%$. Nevertheless, the solutions quality gain of MA is achieved at the expense of prolonged execution time, especially for larger datasets. ILS clearly outperforms all alternative approaches requiring, on average, $\sim 3$ sec. The next most efficient approach is GRASP-ELS requiring 2.5 to 7 times longer execution time, which is prohibitive for real time applications, especially for large number of tours ($k > 2$). Then, GVNS, SA, VNS, MA, I3CH and ACS follow, with ACS requiring three orders of magnitude longer time than ILS. Overall, both performance comparison tests agree on that ILS algorithm represents a fair compromise with respect to execution efficiency versus solutions quality and is the only existing TOPTW approach appropriate for real-time applications.

## 3. Cluster-based heuristics

In TOPTW we are given a directed graph $G = (V, A)$ where $V = \{1, ..., N\}$ is the set of nodes and $A$ is the set of links, an integer $k$, and a time budget $B$. The main attributes of each node are: the visiting time (visit duration), $\text{visit}_i$, the profit gained by visiting $i$, $\text{profit}_i$, and each day's time window $[\text{open}_{im}, \text{close}_{im}], m = 1, 2, \ldots, k$

(a POI may have different time windows per day). Every link $(i, j) \in A$ denotes the transportation link from $i$ to $j$ and is assigned a travel cost $\text{travel}_{ij}$. The objective is to find $k$ routes starting from 1 and ending at $N$ with no other common node, each with overall duration limited by the time budget $B$, that maximize the overall profit collected by the visited nodes.

Herein, we propose two cluster-based heuristic algorithms, the Randomized Cluster Ratio (RCRatio) and the Cluster Search Cluster Routes (CSCRoutes), which address the weaknesses of the ILS algorithm proposed by Vansteenwegen *et al.* (2009). The proposed heuristics employ clustering to organize nodes into groups based on topological distance criteria. The CSCRoutes algorithm reduces the number of transfers among clusters by strictly enforcing the completion of the visit in a local area (cluster) before moving to the next one. Thus, it favors walking transfers between nodes while long travel distances are minimized. The RCRatio algorithm does not enforce but only favours the sequential visit of nodes belonging to the same cluster, allowing return to the same cluster after visiting a different one. The CSCRoutes algorithm is likely to provide solutions of lower quality (i.e. decreased overall profit) than the RCRatio algorithm, especially, in instances featuring tight time windows. At the same time, CSCRoutes is expected to perform better than RCRatio with respect to execution time.

For the sake of clarity of presentation of the proposed heuristics, we shall first briefly present the ILS algorithm, discuss its weaknesses and then explain how we address them. As mentioned in Section 2, ILS defines an "insertion" and a "shake" step. At each insertion step (**ILS_Insert**) a node is inserted in a route, ensuring that all following nodes along the route remain feasible to visit. ILS modeling involves the following variables for each node $i$: (a) $\text{arrive}_i$ denoting the arrival time at node $i$, (b) $\text{start}_i$ denoting the time the visit at node $i$ starts, (c) $\text{wait}_i$ defined as the waiting time in case the arrival at $i$ takes place before $i$'s opening time, and (d) $\text{maxShift}_i$ defined as the maximum amount of time the start of the visit of $i$ can be delayed without making any visit of a node following in the route infeasible due to this delay, that is, without shifting the visit at a node after the closing time of the node or beyond the prescribed time budget. If a node $p$ is inserted in a route $t$ between $i$ and $j$, let $\text{shift}_p = \text{travel}_{ip} + \text{wait}_p + \text{visit}_p + \text{travel}_{pj} - \text{travel}_{ij}$ denote the time cost added to the overall route time due to the insertion of $p$. The node $p$ can be inserted in a route $t$ between $i$ and $j$ if and only if $\text{start}_i + \text{visit}_i + \text{travel}_{ip} \leq \text{close}_{pt}$ (i.e. the time window of $p$ is not violated) and at the same time $\text{shift}_p \leq \text{wait}_j + \text{maxShift}_j$ (i.e. the remainder of the route following $p$ remains feasible).

For each node $p$ not included in a route, the best route to include that node as well as the best insertion position in that route is determined by computing the lowest insertion time cost (shift). For each of these best possible insertions, one per each node $p$ not yet in any route, the heuristic calculates the ratio

$$\text{ratio}_p = \frac{\text{profit}_p^2}{\text{shift}_p}$$

which represents a measure of how profitable it is to visit $p$ versus the time delay this visit incurs. Among all candidate nodes, the heuristic selects for insertion the one with the highest ratio.

At the shake step (**Shake**) the algorithm tries to escape from local optimum by removing a number of nodes in each route of the current solution, in search of non-included nodes that may either decrease the route's time length or increase the overall collected profit. The shake step takes as input two integers: (a) the removeNumber that determines the number of the consecutive visits to be removed from each route and (b) the startNumber that indicates where to start removing nodes on each route of the current solution. If, throughout the process, the end location is reached, then the removal continues with the nodes following the start location. The ILS algorithm initializes the parameters startNumber and removeNumber of the shake step to 1 and loops up to a specified number of times (150) as long as the profit of the best solution is not improved. Inside the loop, the insertion step is applied until a local optimum is reached. If the current solution's profit is larger than the profit of the best solution, the current solution is kept as the best solution and parameter removeNumber is reset to one. In the sequel, the shake step is applied. After the application of the shake step, the values of its parameters are adapted as follows: the value of startNumber is increased by the value of removeNumber and the value of removeNumber is increased by one. If startNumber is greater than or equal to the size of the smallest route in the current solution, then startNumber is decreased by this size. If removeNumber equals to $\frac{N}{3k}$ then it is reset to one. The pseudo code of ILS algorithm is listed below (Algorithm 3.1).

To the best of our knowledge, ILS is the fastest known algorithm for solving the TOPTW offering a fair compromise in terms of speed versus deriving routes of reasonable quality. However, it presents the following weaknesses:

- During the insertion step, ILS may rule out candidate nodes with high profit value because they are relatively time-expensive to reach (from nodes already included in routes). This is also the case even when whole groups of high profit nodes are located within a restricted area of the plane but far from the current route instance. For instance, in Figure 1(a), ILS inserts consecutively $i$, $j$, $k$ and $l$ due to their small shift values. Then, no node from $C$ can be inserted due to the route's time budget. If however a node from $C$ was initially inserted then a higher profit route would have been produced. (Note that in Figure 1 the size of each circle denotes the value of the corresponding node's profit.)
- In the insertion step, ILS may be attracted and include into the solution some high-score nodes isolated from high-density topology areas. This may trap ILS and make it infeasible to visit far located areas with "good" candidate nodes due to prohibitively large traveling time (possibly leaving considerable amount of the overall time budget unused). For instance, in Figure 1(b), the route was trapped after inserting the high profit node

**Algorithm 3.1 ILS** (Vansteenwegen *et al.* (2009))

```
maxIterations ← 150
maxNumberToRemove ← N/3k
startNumber ← 1; removeNumber ← 1; notImproved ← 0
while notImproved < maxIterations do
    while not local optimum do
        ILS_Insert
    end while
    if currentSolution.profit > bestSolution.profit then
        bestSolution ← currentSolution
        removeNumber ← 1
        notImproved ← 0
    else increase notImproved by 1
    end if
    Shake(removeNumber,startNumber)
    increase startNumber by removeNumber
    increase removeNumber by 1
    if startNumber ≥ currentSolution.sizeOfSmallestRoute then
        decrease startNumber by currentSolution.sizeOfSmallestRoute
    end if
    if removeNumber = maxNumberToRemove then
        removeNumber ← 1
    end if
end while
return bestSolution
```

$j$.

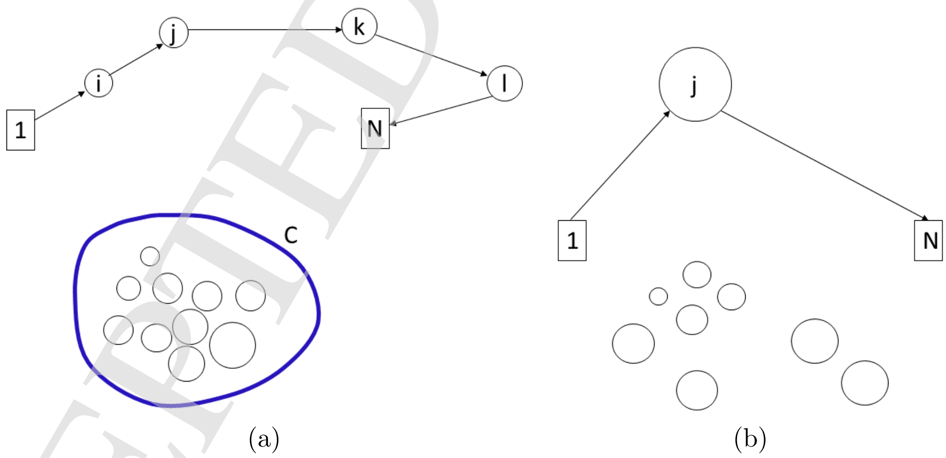


Fig. 1. Weaknesses of ILS

The proposed algorithms, Randomized Cluster Ratio (RCRatio) and Cluster Search Cluster Routes (CSCRoutes), address the aforementioned weaknesses of the ILS algorithm. Both algorithms employ clustering to organize nodes into groups (clusters) based on topological distance criteria. Nodes grouped within the same cluster are geographically close e.g., they are within walking distance or they belong to the same area of the city. Having visited a high-profit node that belongs to a certain cluster, our algorithms encourage visits to other nodes of the same cluster because such visits reduce (a) the duration of the routes and (b) the number of transfers among clusters. Note that a tourist apart from maximizing the total profit, may also prefer to minimize inter-cluster tranfers as those are typically long and require usage of public transportation; this may incur a considerable budget cost, while walking is usually a preferred option than using the public transportation.

Both RCRatio and CSCRoutes employ the global $k$-means algorithm (Likas *et al.*, 2003; Bagirov, 2008) to organize the set of nodes into an appropriate (based on the network topology) number of clusters (we term this parameter as numberOf-Clusters). Global $k$-means is an effective global clustering approach, which minimizes the clustering error and obtains a near-optimal solution for the clustering problem through applying a series of local searches using the $k$-means algorithm.

Once the clusters of nodes have been formed during a clustering phase, a route initialization phase **RouteInitPhase** starts. During this phase a single node is inserted into each of the $k$ initially empty routes. Each of the $k$ inserted nodes comes from a different cluster, i.e. no two inserted nodes belong to the same cluster, unless numberOfClusters$< k$. Since the number of clusters is usually larger than $k$ we need to decide which $k$ clusters will be chosen in the route initialization phase. Different approaches may be followed such as choosing the $k$ clusters with the highest average profit or the highest $h-$index[a], or trying different sets of $k$ high-profit clusters. Following the second approach, we consider a listOfClusterSets list containing a specific number of different sets of $k$ clusters. The list may contain all $k$-combinations of the elements of a small set $S$ with the most profitable clusters. **RouteInitPhase** takes as argument a set of $k$ clusters from listOfClusterSets and proceeds as follows: for each cluster $C$ in the set, it finds the node $p \in C$ with the highest ratio$_p$ and inserts it into one of the empty routes (Figure 2). By initializing each one of the $k$ routes of the TOPTW solution with a node from different cluster, the algorithms encourage searching different topology areas and avoid getting trapped at specific high-scored nodes. Then the algorithms combine an insertion step and a shake step to escape from local optima as described in the following subsections.

### 3.1. *Randomized Cluster Ratio Algorithm*

The Randomized Cluster Ratio algorithm (RCRatio) introduces a randomized insertion step **RCRatio_Insert** which takes into account the clustering of the nodes by using a parameter clusterParameter $\geq 1$ and a number randomly generated within

[a]Similarly to the $h$-index used to measure the impact of the published work of a scholar, a cluster has index $h$ if $h$ of its $N_p$ nodes have profit values equal or greater than $h$, while the other $(N_p - h)$ nodes have profit values less than $h$. Intuitively, the $h$-index should be a more accurate measure of a cluster's attractiveness since the average is a metric affected to a large extent by outlier values.

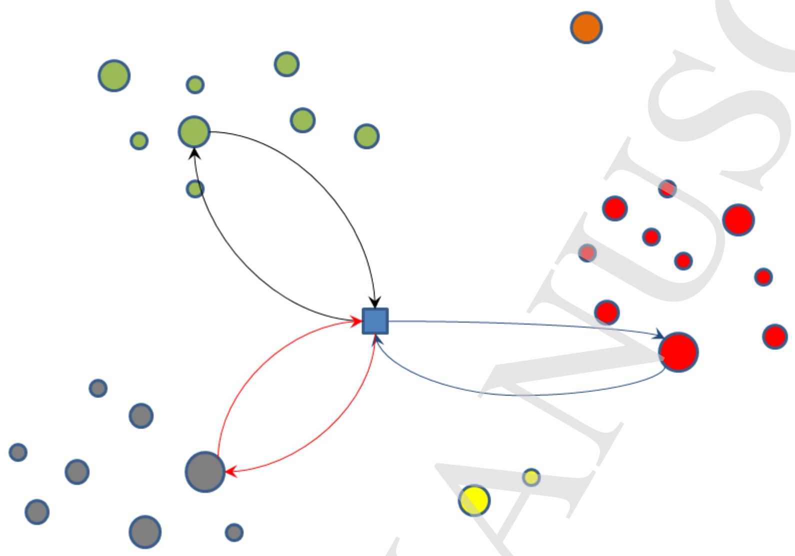

Fig. 2. Illustration of the RouteInitPhase

an interval $[1, \text{factorOfRandomness}]$ in order to add diversification. The higher the value of clusterParameter, the more the algorithm favors the insertion of a node $p$ before or after a node that belongs to the same cluster with $p$. Specifically, the parameter clusterParameter is used to increase the probability of inserting $p$ between $i$ and $j$ if $p$ belongs to the same cluster as either $i$ or $j$. For that, RCRatio considers the variable $\text{shiftCluster}_p$ defined as

$$\text{shiftCluster}_p = \frac{\text{shift}_p}{\text{clusterParameter} \cdot \text{rand}}$$

in the case that $\text{cluster}(p)$ coincides with $\text{cluster}(i)$ or $\text{cluster}(j)$, where $\text{cluster}(l)$ denotes the cluster where a node $l$ belongs to, and rand is a random number between 1 and factorOfRandomness. Otherwise, $\text{shiftCluster}_p = \frac{\text{shift}_p}{\text{rand}}$ (it is noted that $\text{shift}_p$ is defined as in the ILS algorithm). For each candidate node $p$, the position with the lowest $\text{shiftCluster}_p$ is determined and the ratio $\text{ratio}_p = \frac{\text{profit}_p^2}{\text{shiftCluster}_p}$ is computed. Then, the node $p$ with the highest ratio is inserted in the position with the lowest $\text{shiftCluster}_p$.

RCRatio initializes the clusterParameter with a large value (in this work we have chosen the value of 1.3 based on our experiments) in order to initially encourage visits to be within the same clusters and decreases the value of clusterParameter by a step at a chosen number of iterations ( we have chosen 0.1 every quarter of maxIterations, i.e. the maximum number of iterations without improvement). In this way, routes with many successive nodes belonging to the same cluster are initially favored; as the number of iterations without yielding improved solution increases, insertions of nodes belonging to the same cluster are less favored. The random factor of the insert step gives diversification, not allowing the algorithm to perform the same action (i.e. insert a node previously removed by the shake step) over and over again.

Apart from the insert and shake steps, RCRatio applies an additional step in between for reaching a better solution, the "replace" step (**Replace**). In the replace step, a node $v$ inserted in a route, is replaced by a non-inserted node $u$ such that $u$ is feasible to be inserted in the position of $v$ and the difference of their profits (i.e. $\text{profit}_u$-$\text{profit}_v$) is the maximum possible. A pseudocode for the replace step follows (Algorithm 3.2).

**Algorithm 3.2 Replace**

```
bestDiff ← 0
for each non-insereted node u do
    for each node v in a route do
        if u can replace v and profit_u-profit_v > bestDiff then
            bestInsert ← u, bestReplace ← v, maxDiff ← profit_u-profit_v
        end if
    end for
end for
Replace bestReplace by bestInsert
Update arrival times and maxShifts
```

RCRatio loops for a number of times equal to the size of the listOfClusterSets. Within the loop, firstly all nodes included into the current solution's routes are removed and the route initialization phase is executed receiving as input argument a set of clusters taken (pop operation) from the listOfClusterSets list. Secondly, the algorithm initializes the parameters startNumber and removeNumber of **Shake** to 1 and the parameter clusterParameter of **RCRatio_Insert** as discussed above, and executes an inner loop until there is no improvement of the best solution for maxIterations successive iterations. The insertion step is iteratively applied within this loop followed by a replace step with a specified probability, probabilityOfReplace, until a local optimum is reached. When a local optimum is reached, the solution is further improved applying replace steps, as long as an improvement is feasible. Lastly, the shake step is applied. The shake step takes as input the parameter removeNumber that determines the number of the consecutive visits to be removed from each route and the startNumber that indicates where to start removing nodes on each route of the current solution. If throughout the process, the end location is reached, then the removal continues with the nodes following the start location. The value of the parameter removeNumber used in the shake step, is allowed to be at most equal to the minimum of half of the size of the longest route in the current solution and $\frac{N}{3k}$ (note that in ILS the maximum allowed removeNumber equals $\frac{N}{3k}$). In this way, execution time is saved, since a local optimum is reached in short time, given that a small portion of the solution has been removed . Taking advantage of this time saving, the number of iterations can be increased without increasing the overall algorithm's execution time. Therefore, RCRatio may exercise a larger maxIterations value in comparison to ILS.

In order to reduce the search space (therefore, the execution time) of **RCRatio_Insert**, in the case that a non-inserted node $p$ is infeasible to be inserted in any route, it is removed from the list of candidate node and it is added back, only after **Shake** has been applied. The pseudo code of RCRatio algorithm is listed below (Algorithm 3.3). Figure 3 illustrates an example solution obtained by RCRatio (notice that the algorithm allows a route to include node of the same cluster, yet, not successively visited).

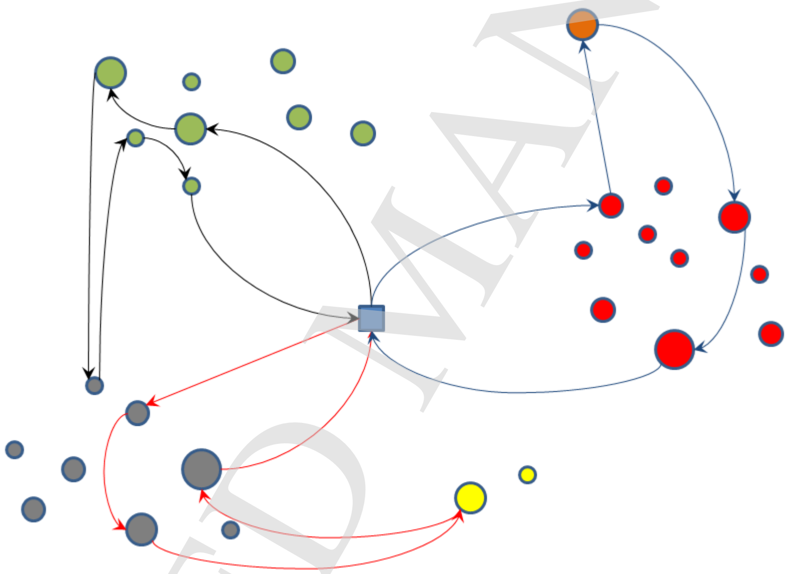

Fig. 3. Example of a RCRatio solution

### 3.2. *Cluster Search Cluster Routes Algorithm*

Given a route $t$ of a TOPTW solution, any maximal sub-route in $t$ comprising a sequence of nodes within the same cluster $C$ is defined as a *Cluster Route (CR) of $t$ associated with cluster $C$* and denoted as $CR_C^t$. Note that a route $t$ of a TOPTW solution constructed by the ILS or RCRatio algorithm may include more than one cluster route $CR_C^t$ for the same cluster $C$, i.e., a tour $t$ may visit and leave cluster $C$ more than once. CSCRoutes algorithm is designed to construct routes that visit each cluster at most once, i.e. if a cluster $C$ has been visited in a route $t$ it cannot be revisited in the same route and therefore, for each cluster $C$ there is only one cluster route in any route $t$ associated with $C$. The only exception allowed is when the start and the end node of a route $t$ belong to the same cluster $C'$. In this case,

**Algorithm 3.3 RCRatio**(numberOfClusters, maxIterations, factorOfRandomness, probabilityOfReplace)

```
run the global k-means algorithm with k = numberOfClusters
construct the list listOfClusterSets
it1 ← maxIterations/4; it2 ← 2·maxIterations/4; it3 ← 3·maxIterations/4
while listOfClusterSets is not empty do
    remove all nodes visited in the currentSolution
    theClusterSetIdToInsert ← listOfClusterSets.pop
    RouteInitPhase(theClusterSetIdToInsert, factorOfRandomness)
    startNumber ← 1; removeNumber ← 1; notImproved ← 0
    while notImproved < maxIterations do
        if notImproved < it2 then
            if notImproved < it1 then clusterParameter ← 1.3
            else clusterParameter ← 1.2
            end if
        else
            if notImproved < it3 then clusterParameter ← 1.1
            else clusterParameter ← 1.0
            end if
        end if
        while the insert step succeeds do
            RCRatio_Insert(clusterParameter, factorOfRandomness)
            if p < probabilityOfReplace then
                Replace
            end if
        end while
        while the replace step succeeds do
            Replace
        end while
        if currentSolution.profit > bestSolution.profit then
            bestSolution ← currentSolution ; removeNumber ← 1; notImproved ← 0
        else increase notImproved by 1
        end if
        if removeNumber > min{currentSolution.sizeOfLargestTour/2, N/(3k)} then
            removeNumber ← 1
        end if
        Shake(removeNumber,startNumber)
        increase startNumber by removeNumber
        increase removeNumber by 1
        if startNumber ≥ currentSolution.sizeOfSmallestTour then
            decrease startNumber by currentSolution.sizeOfSmallestTour
        end if
    end while
end while
return bestSolution
```

a route $t$ may start and end with nodes of cluster $C'$, i.e. $C'$ may be visited twice in the route $t$ and therefore, for a route $t$ there might be two cluster routes $CR^t_{C'}$.

The insertion step **CSCRoutes_Insert** of the CSCRoutes algorithm does not allow the insertion of a node $p$ in a route $t$, if this insertion creates more than one cluster routes $CR^t_C$ for some cluster $C$. In the following, the description of insertion step **CSCRoutes_Insert** is given, based on the following assumptions. Consider w.l.o.g. that the start and end nodes in the TOTPW coincide (*depot*). If a route $t$ contains two CR associated with the cluster of the *depot*, then let $CR_f^t$ be the first cluster route (starts at the *depot*) in $t$, and $CR_l^t$ be the last cluster route (ends at the *depot*) in $t$. Also, assume that for each node $p$ ratio$_p$ is calculated as in ILS algorithm. Finally, consider for each route $t$, the list clustersIn($t$) containing any cluster $C$ for which there is a nonempty $CR_C^t$.

Given a candidate for insertion node $p$ and a route $t$, **CSCRoutes_Insert** distinguishes among the following cases:

- cluster($p$)=cluster(*depot*) and clustersIn($t$) contains only the cluster(*depot*). Then $p$ can be inserted anywhere in the route, since the insertion would not violate the CR constraints.
- cluster($p$)=cluster(*depot*) and clustersIn(t) contains more than one cluster. Then $p$ can be inserted anywhere in $CR_f^t$ and in $CR_l^t$.
- cluster($p$)≠cluster(*depot*) and clustersIn($t$) contains only cluster(*depot*), then the insertion is feasible anywhere in $t$. If the insertion occurs, then a new CR will be created with $p$ as its only node.
- cluster($p$)≠cluster(*depot*) and clustersIn($t$) contains two or more clusters but not cluster($p$). Then $p$ can be inserted after the end of every CR in $t$, except for $CR_l^t$. If the insertion occurs, then a new CR will be created with $p$ as its only node.
- cluster($p$)≠cluster(*depot*) and clustersIn($t$) contains two or more clusters and also includes cluster($p$). Then $p$ can be inserted anywhere in $CR_{\text{cluster}(p)}^t$.

The pseudo code of **CSCRoutes_Insert** (Algorithm 3.4) follows.

Note that similarly to the RCRatio algorithm when a non-included node $p$ is infeasible to insert in any route, then $p$ is removed from the list of candidates and re-examined, only after **Shake** has been applied.

Like RCRatio algorithm, CSCRoutes executes a loop for a number of times equal to the size of the listOfClusterSets. Within the loop, firstly, all nodes in the current solution's routes are removed and the route initialization phase is executed. Secondly, the algorithm initializes the parameters startNumber and removeNumber of **Shake** to 1 and executes an inner loop up to a specific number of times (maxIterations) while the profit of the best solution is not improved. Within this loop, the insertion step **CSCRoutes_Insert** is applied until a local optimum is reached. At the end, the shake step is applied. The pseudo code of CSCRoutes Algorithm 3.5 is given below.

The CSCRoutes algorithm is likely to create solutions of lower quality (i.e. decreased overall profit), especially in instances featuring tight time windows. However, it significantly reduces the number of transfers among clusters and therefore it favors routes that include nodes of the same cluster. In this way, walking transfers are preferred while overly long travel distances are minimized. At the same

**Algorithm 3.4 CSCRoutes_Insert**

```
for each candidate node p do
    clusterID←cluster(p)
    for each route t do
        if clusterID=cluster(depot) then
            if clustersIn(t) contains only cluster(depot) then
                Search all possible insert positions in t for the least shift_p
            else
                Search all possible insert positions in CR^t_f and CR^t_l for the least shift_p
            end if
        else                                        ▷ clusterID≠cluster(depot)
            if clustersIn(t) contains only cluster(depot) then
                Search all possible insert positions in t for the least shift_p
            else
                if clustersIn(t) doesn't contain clusterID then
                    Search all possible positions in t that are the end of a CR, for the least shift_p
                else Search all possible insert positions in CR^t_clusterID for the least shift_p
                end if
            end if
        end if
    end for
end for
Insert the node q with the highest ratio.
Update times, maxShifts and lists clustersIn.
```

time, the CSCRoutes is expected to perform better than ILS and RCRatio with respect to execution time, since **CSCRoutes_Insert** is faster than **ILS_Insert** and **RCRatio_Insert** (this is because the number of possible insertion positions for any candidate node is much lower). Figure 4 illustrates an example solution obtained by CSCRoutes.

## 4. Experimental Results

### 4.1. *Test instances*

Montemanni and Gambardella (2009) designed TOPTW instances based on previous OPTW instances of Solomon (1987) (data sets for vehicle routing problems with time windows: c10*, r10* and rc10*) and Cordeau *et al.* (1997) (10 multi-depot vehicle routing problems: pr1-pr10). They also added 27 extra instances based on Solomon (c20*, r20* and rc20*) and 10 instances based on Cordeau et al. (pr11-pr20). Cordeau et al. instances have up to 288 customers and much wider time windows than in Solomon's problems. All the aforementioned instances involve one, two, three and four routes. Optimal solutions are available for some of those test instances. Herein, we compare the performance of our heuristics against the best-known algorithm suitable for real-time TTDP applications, ILS (Vansteenwegen *et al.*, 2009).

The aforementioned instances allow a fair comparison of our proposed heuristics against published results, yet, they do not represent suitable examples of real-life

**Algorithm 3.5 CSCRoutes**(numberOfClusters,maxIterations)

```
run the global k-means algorithm with k=numberOfClusters
construct the list listOfClusterSets
while listOfClusterSets is not empty do
    remove all nodes visited in the currentSolution
    theClusterSetIdToInsert ← listOfClusterSets.pop
    RouteInitPhase(theClusterSetIdToInsert)
    startNumber ← 1; removeNumber ← 1; notImproved ← 0
    while notImproved < maxIterations do
        while not local optimum do
            CSCRoutes_Insert
        end while
        if currentSolution.profit > bestSolution.profif then
            bestSolution ← currentSolution ; removeNumber ← 1; notImproved ← 0
        else increase notImproved by 1
        end if
        if removeNumber > min{currentSolution.sizeOfLargestTour/2, N/(3k)} then
            removeNumber ← 1
        end if
        Shake(removeNumber,startNumber)
        increase startNumber by removeNumber
        increase removeNumber by 1
        if startNumber ≥ currentSolution.sizeOfSmallestTour then
            decrease startNumber by currentSolution.sizeOfSmallestTour
        end if
    end while
end while
return bestSolution
```

TTDP problems. In such problems (a) POIs are typically associated with much wider, overlapping, multiple time windows (e.g. Monday closed, Tuesday-Friday 08:30-16:00, Saturday-Sunday 09:00-18:00); (b) POIs' locations are statistically dependent, i.e. typical tourist destination topologies feature dense concentration of POIs at certain areas, while isolated POIs are rare; (c) visiting time at a POI is typically correlated with its profit value (e.g. POIs associated with high profit value are expected to take long to visit); (d) the time available for sightseeing (daily time budget) is typically in the order of a few hours per day (in contrast, Cordeau et al. and Solomon r2*/rc2* instances allow time budget up to 16.5 hours, while Solomon c2* instances up to 56.5 hours, which is certainly unrealistic).

Along this line, we have created 100 new TOPTW instances (t*) with the following characteristics: the number of routes is 1-3; the number of nodes is 100-200, which is considered a fair estimation of available POIs on medium-to-large scale urban tourist destinations; 80% of the nodes are located around 1-10 virtual centers (the distances of nodes from their randomly assigned center follow a Gaussian distribution); a 20% of the nodes is set at a random location on the plane; the profit associated with nodes is 1-100, while visiting time at any node is 1-120 min (visiting time is proportional to the profit); regarding time windows, we assume that 50% of the nodes are open in 24h basis (e.g. squares, parks and landmarks

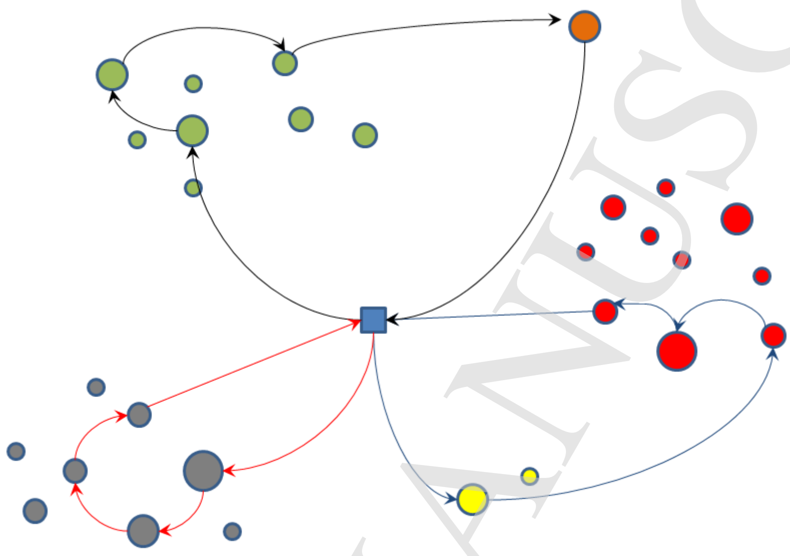

Fig. 4. Example of a CSCRoutes solution

open to visitors), while the remaining are closed either on weekends (15%) or one day per week, either Monday (15%), Tuesday (10%) or Wednesday (10%) (during their opening days, the non-24h nodes are open 08:30-17:00); the daily time budget is set to 10h (510-1110 min) in t1* and 5h (840-1140 min) in t2* instances.

Table 4.1 overviews the available TOPTW test instances. For every set of instances, the corresponding reference is given, along with the name of the original instances the set is based on. The number of instances, nodes ($N$) and routes ($k$) as well as the daily time budget ($B$) are also presented.

Table 1. TOPTW Instances

| Reference | Based on | # of instances | $N$ | $B$ | Average TW | $k$ |
|---|---|---|---|---|---|---|
| (Montemanni and Gambardella, 2009) | Solomon (c1*, r1* and rc1*) | 29 | 100 | c1*:1236 r1*:230 rc1*:240 | c1*:321 r1*:87 rc1*:85 | 1,2,3,4 |
| | Cordeau et al.(pr01-10) | 10 | 48-288 | 1000 | 135 | 1,2,3,4 |
| | Solomon (c2*, r2* and rc2*) | 27 | 100 | c2*:3390 r2*:1000 rc2*:960 | c2*:921 r2*:454 rc2*:370 | 1,2,3,4 |
| | Cordeau et al.(pr11-20) | 10 | 48-288 | 1000 | 269 | 1,2,3,4 |
| This article | t1* | 50 | 100-200 | 600 | 1000 | 1,2,3 |
| | t2* | 50 | 100-200 | 300 | 997 | 1,2,3 |

The benchmark instances of Montemanni and Gambardella are available at `http://www.mech.kuleuven.be/en/cib/op/`, while the t* instances are available at `http://dgavalas.ct.aegean.gr/public/op_instances/`.

### 4.2. *Results*

All computations were carried out on a personal computer Intel Core i3 with 2.30 GHz processor and 4 GB RAM. Our tests aim at comparing our proposed algorithms against the best known real-time TOPTW approach (ILS), which yields high quality solutions, while being suitable for real-time TTDP applications. Reported results compare ILS against RCRatio and CSCRoutes with respect to the following aspects: (a) overall collected profit and (b) execution (CPU) time required to derive a solution. In addition to our proposed algorithms, ILS has been also implemented to ensure fair comparison with respect to execution (CPU) time required to derive solutions; the overall collected profit values corresponding to ILS are those published in (Vansteenwegen *et al.*, 2009). Clearly, mostly preferred solutions are those associated with high profit values (higher profit values denote higher quality solutions) and reduced execution time (as this denotes improved suitability for real-time TTDP applications). All three algorithms have been coded in C++.

With respect to parameter setting in the algorithms RCRatio and CSCRoutes, a number of different alternative values have been tested; reasonably, we selected the ones that achieved the best performance results with respect to both the quality of solutions and the computational time needed to derive solutions. The number of clusters derived by the global $k$-means algorithm (numberOfClusters) is set to a fraction of the number of POIs ($N/10$). Note that the appropriate number of clusters depends on the graph topology and the distribution of the POIs in the area. The listOfClusterSets is implemented by adding $\left\lceil \frac{\text{numberOfClusters}}{k} \right\rceil$ disjoint sets of $k$ clusters, such that the first set contains the $k$ clusters with the highest average profit, the second set contains the next $k$ highest average profit clusters, etc. This allows each cluster to be visited at least once while keeping the number of elements of listOfClusterSets as short as possible. The value of maxIterations is set equal to $\frac{400}{|\text{listOfClusterSets}|} \cdot \frac{k+1}{2 \cdot k}$. In this way, the number of iterations is decreased when the number of routes increases; note that large number of routes implies large total time budget (over all routes) and therefore, optimal solutions can be reached within a smaller number of iterations. Finally, all combinations of factorOfRandomness and probabilityOfReplace with factorOfRandomness $= 1, 1.1, 1.2$ and probabilityOfReplace $= 0, 0.05, 0.1, 0.2$ have been examined with the best performance results obtained for factorOfRandomness and probabilityOfReplace set to 1.1 and 0.05, respectively.

#### 4.2.1. *Overview of results*

In Appendix Appendix A, we provide the analytical results of ILS, RCRatio and CSCRoutes based on the benchmark instances of Solomon (Tables 12, 14, 16 and 18) and Cordeau et al. (Tables 13, 15, 17 and 19). We provide results for one (Tables 12 and 13), two (Tables 14 and 15), three (Tables 16 and 17) and four routes (Tables 18 and 19) over the same sets of instances. Furthermore, Tables

20 and 21 list the results compiled from the new t1* and t2* instances, respectively. The first two columns show the instance's name and the number of clusters derived from the global $k$-means clustering algorithm (the latter is proportional to the instances' number of nodes, i.e. $\frac{N}{10}$), respectively. The next three sets of column dyads correspond to the results obtained from ILS, RCRatio and CSCRoutes, respectively. Total collected profit and execution time are reported for each algorithm. The results obtained by RCRatio are the average of five runs of the algorithm in each instance.

Table 4.2.1 offers a high-level comparative view of the compiled results, in effect, averages of the absolute values catalogued in Tables 12 – 21, grouped by instance 'families'. The performance gaps of RCRatio and CSCRoutes over ILS are shown in parenthesis, in percentages (clearly, positive gaps indicate performance gain for our algorithms).

RCRatio prevails over ILS in terms of profit in 22 cases among 30 average values, while in 4 cases it achieves identical performance (both algorithms reach optimality). In fact RCRatio achieves its higher performance gap over ILS on the t2* instances, which account for the TTDP-tailored topologies with 5 hours time budget. RCRatio's clear prevalence in profit is mainly due to being more effective in escaping local optima, as a result of the insertion step which incorporates randomization as well as the diversification obtained by the multiple values clusterParameter gets during the process. Furthermore, the Replace step following the insertion steps improves more the solution. Last, initializing the routes with nodes from different clusters results in searching the solution space more thoroughly without getting trapped in a region of the network. With respect to execution time, RCRatio executes slower on instances wherein a local optimum is reached fast, i.e. those not requiring thorough exploration of the search space. On those scenarios, ILS derives a solution earlier (typically in less than 0.5 sec) due to the smaller number of iterations (maxIterations=150) compared to RCRatio (maxIterations=400). Nevertheless, this time lag is expected to remain unnoticeable by TTDP application users as it is typically below 0.5 sec. On the other hand, RCRatio executes faster in 8 cases among the 12 wherein ILS takes longer than a second to execute, i.e. in the cases wherein the performance of ILS is marginally acceptable for real-time applications (the 4 cases where ILS performs better are the r2* instances for one tour, rc2* instances for one and two tours and c2* instances for three tours). Notably, in the cases where ILS takes more than 5 sec to execute (pr* for 3 and 4 tours), RCRatio achieves time savings of more than 2 sec.

CSCRoutes attains higher profit values than ILS only on 4 (among 30) instance families, executing however faster on 25 out of the 30 cases. As regards the TTDP-tailored t* instances, it yields similar profit values while clearly outperforming ILS with respect to execution time. In fact, on some instances wherein ILS hardly meets the requirements of real-time applications (e.g. $\sim 7.3$ sec on pr* for 4 tours), CSCRoutes may execute almost three times as fast ($\sim 2.7$ sec) while not compromising much of the solutions quality (−5.5%). Notably, CSCRoutes prevails with regards to all performance parameters on several scenarios (see rc1* instances for 1 tour, c2* for 2 and 3 tours, t2*). The above described performance profile of CSCRoutes is mainly attributed to its focal design objective to prioritize the insertion of successive nodes which belong to the same cluster. This drastically reduces the search space, hence the execution time. This gain comes at the expense of solutions quality as the CSCRoutes is not free to consider candidate nodes irrespective of their cluster association.

Table 2. Averages of performance parameters grouped by instance 'families'

| | | ILS | | RCRatio | | | | CSCRoutes | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $k$ | Name | Profit | Time(ms) | Profit | Gap | Time(ms) | Gap | Profit | Gap | Time(ms) | Gap |
| 1 | c1* | 362.2 | 121.1 | 365.3 | 0.9 | 194.4 | -60.5 | 356.7 | -1.5 | 146.8 | -21.2 |
| | c2* | 911.3 | 655.3 | 923 | 1.3 | 732.4 | -11.8 | 906.3 | -0.5 | 215.9 | 67.1 |
| | r1* | 276.9 | 135.3 | 280.1 | 1.2 | 188 | -39 | 273.8 | -1.1 | 130.9 | 3.3 |
| | r2* | 969.3 | 1216.5 | 965.6 | -0.4 | 1337.2 | -9.9 | 817.9 | -15.6 | 256.6 | 78.9 |
| | rc1* | 257.8 | 123.8 | 264.6 | 2.6 | 143.5 | -15.9 | 259.4 | 0.6 | 108.8 | 12.1 |
| | rc2* | 925.3 | 901.9 | 923 | -0.2 | 892 | 1.1 | 848.9 | -8.3 | 231.4 | 74.3 |
| | pr* | 458.6 | 1064 | 467.7 | 2 | 776.8 | 27 | 421.3 | -8.1 | 307.7 | 71.1 |
| | **Average** | **594.5** | **602.6** | **598.5** | **0.7** | **609.2** | **-1.1** | **554.9** | **-6.7** | **199.7** | **66.9** |
| 2 | c1* | 666.7 | 395.2 | 669.1 | 0.4 | 400.3 | -1.3 | 641.1 | -3.8 | 286.4 | 27.5 |
| | c2* | 1440 | 1284.4 | 1451 | 0.8 | 1540.8 | -20 | 1446.3 | 0.4 | 518.5 | 59.6 |
| | r1* | 499.2 | 428.8 | 505.1 | 1.2 | 387.9 | 9.5 | 493 | -1.2 | 268 | 37.5 |
| | r2* | 1370.5 | 876 | 1371.9 | 0.1 | 1531.2 | -74.8 | 1229.9 | -10.3 | 480.9 | 45.1 |
| | rc1* | 497.9 | 466.4 | 503.1 | 1 | 323.9 | 30.6 | 487.8 | -2 | 241.5 | 48.2 |
| | rc2* | 1501 | 1117.9 | 1492 | -0.6 | 1554.2 | -39 | 1345.1 | -10.4 | 436.5 | 61 |
| | pr* | 831.7 | 2696.1 | 839.4 | 0.9 | 1876.8 | 30.4 | 759.6 | -8.7 | 839.6 | 68.9 |
| | **Average** | **972.4** | **1037.8** | **975.9** | **0.4** | **1087.9** | **-4.8** | **914.7** | **-5.9** | **438.8** | **57.7** |
| 3 | c1* | 897.8 | 1050.3 | 903.3 | 0.6 | 723.7 | 31.1 | 884.4 | -1.5 | 523.9 | 50.1 |
| | c2* | 1775 | 907.8 | 1773.3 | -0.1 | 1224.3 | -34.9 | 1778.8 | 0.2 | 494.6 | 45.5 |
| | r1* | 704 | 1017.8 | 704.7 | 0.1 | 732.6 | 28 | 686.3 | -2.5 | 501.4 | 50.7 |
| | r2* | 1452.1 | 479.1 | 1452.5 | 0 | 1017.8 | -112.4 | 1404.6 | -3.3 | 489.1 | -2.1 |
| | rc1* | 713.1 | 831.6 | 722.2 | 1.3 | 604.5 | 27.3 | 708 | -0.7 | 434.5 | 47.8 |
| | rc2* | 1695.4 | 637.3 | 1698.2 | 0.2 | 1209.7 | -89.8 | 1567.1 | -7.6 | 501 | 21.4 |
| | pr* | 1112.5 | 5523.2 | 1130.2 | 1.6 | 3485.2 | 36.9 | 1051.9 | -5.4 | 1728.8 | 68.7 |
| | **Average** | **1192.8** | **1492.4** | **1197.8** | **0.4** | **1285.4** | **13.9** | **1154.4** | **-3.2** | **667.6** | **55.3** |
| 4 | c1* | 1101.1 | 1404.7 | 1107.1 | 0.5 | 1115.5 | 20.6 | 1085.6 | -1.4 | 813.4 | 42.1 |
| | c2* | 1810 | 339.9 | 1810 | 0 | 794.1 | -133.6 | 1810 | 0 | 406.6 | -19.6 |
| | r1* | 862.6 | 1410.1 | 866.1 | 0.4 | 1121.6 | 20.5 | 847.5 | -1.8 | 811.1 | 42.5 |
| | r2* | 1458 | 287.3 | 1458 | 0 | 720.9 | -150.9 | 1440.3 | -1.2 | 414.6 | -44.3 |
| | rc1* | 913.1 | 1463 | 919.3 | 0.7 | 922 | 37 | 899.9 | -1.4 | 639.5 | 56.3 |
| | rc2* | 1724 | 368.5 | 1724 | 0 | 844.5 | -129.2 | 1674.8 | -2.9 | 527.5 | -43.1 |
| | pr* | 1341.1 | 7360.6 | 1359.3 | 1.4 | 5195.1 | 29.4 | 1267.8 | -5.5 | 2742.7 | 62.7 |
| | **Average** | **1315.7** | **1804.9** | **1320.5** | **0.4** | **1530.5** | **15** | **1289.4** | **-2** | **907.9** | **49.7** |
| | t1* | 788.6 | 1448.4 | 790.4 | 0.2 | 1458.3 | -0.7 | 781.5 | -0.9 | 1000.3 | 30.9 |
| | t2* | 349.9 | 582.9 | 358.5 | 2.5 | 470.2 | 19.3 | 353.6 | 1.1 | 406.2 | 30.3 |

### 4.2.2. *Detailed analysis of results*

The comparison results between ILS and RCRatio for Solomon and Cordeau et al. instances are summarized in Tables 3 and 4, respectively, for different number of routes. Positive gaps denote predominance of our algorithm against ILS. The opposite (i.e. prevalence of ILS solution) is signified by negative gap values. RCRatio yields significantly higher profit values, especially for instances with tight $B$ and small number of routes (e.g. 1.15 in r1* and 2.78 in rc1*, for one tour, in Table 3). This is because ILS is commonly trapped in isolated areas with few high profit nodes, failing to explore remote areas with considerable numbers of fairly profited candidate nodes, due to prohibitively large travelling time and the limited time budget (see relevant discussion in Section 3). The null (0) values mostly appearing in c2*, r2* and rc2* instances for 3 or 4 routes indicate that both approaches derive the optimum solution since $k$ and $B$ are large enough to accommodate all nodes into the solution. ILS and RCRatio attain similar execution times in most cases, however the former clearly executes faster when examining instances with both large $B$ and $k$ values (in those cases ILS exercises less iterations than RCSRatio and usually reaches a local optimum that can be no longer improved while RCRatio iterates further and by reinitializing the solution with the RouteInitPhase, escapes easier from a local optimum.) Even in those cases though, the absolute execution time for RCRatio is well below 1 sec, which is certainly satisfactory for real-time TTDP applications.

It should be noted that the average values per instance family shown in Tables 3 – 11 are indeed close to the averages of Table 4.2.1 with the exception of execution time values (in fact the figures shown in Table 4.2.1 are a lot more positive for our algorithms than those in Tables 3 – 11). Notably, the tabular values of Table 4.2.1 denote the gaps of the average absolute values while those in Tables 3 – 11 denote the average gaps. However, the latter do not accurately represent the practical performance gap among the compared algorithms. To illustrate this, the execution time of ILS for pr01 and pr20 instances, $k = 4$ tours, is 79ms and 16212ms, respectively (see Table 19). The execution time of RCRatio for the same instances and number of tours is 229ms and 12377ms, respectively. Hence, the shorter execution of ILS by 150ms on pr01 yields a 190% gap in favor of ILS, while ILS lengthier execution by 3835ms on pr20 yields only a 24% gap in favor of RCRatio. Due to the outlier values, the average gap over pr* instances becomes 18.46% in favor of ILS (see Table 4) while the gap of the average absolute times is 29.42% in favor of RCRatio (see Table 4.2.1). In practical terms though, a performance gain of $\sim$ 4sec is far more desirable than a 150ms gain; hence the time gap values of Table 4.2.1 are considered far more indicative. The same observation also holds when comparing the execution times of ILS and CSCRoutes.

The comparison results between ILS and CSCRoutes for Solomon and Cordeau et al. instances are summarized in Tables 5 and 6, respectively. The results indicate a trade-off between profit and execution time. In particular, ILS yields better results with respect to profit as it inserts best candidate nodes freely, irrespective of their cluster assignment. This is especially true when considering instances which combine large $B$ with tight time windows (e.g. r2*), whereby CSCRoutes fails to use the time budget effectively, as it might get trapped within clusters, spending considerable amount of time waiting for the nodes' opening time, while not allowed to escape by visiting neighbour cluster nodes. This disadvantage is mitigated when $k$ increases, as high-profit nodes are then more likely to be selected. On the other hand, CSCRoutes attains shorter execution times (excluding the c1* instances for $k = 1$ , r2* for $k = 3$ and c2*, r2*, rc2* for $k = 4$), as it significantly reduces the search space in its insertion phase.

The comparison results between RCRatio and CSCRoutes for Solomon and Cordeau et al. instances are summarized in Tables 7 and 8, respectively, where negative values indicate prominence of RCRatio. As expected, RCRatio obtains better results in terms of profit as it enables broader exploration of the search space on its insertion phase (mainly due to the randomization and the more relaxed conformance to the cluster association of candidate nodes) as well as the Replace step. On the other hand, RCRatio performs worse with respect to the execution time. The execution time gap increases in favor of CSCRoutes on instances with large $B$ values (e.g. c2*, r2* and rc2*) as their respective solutions accommodate higher numbers of nodes, hence, the insertion iterations (which are more time consuming in RCRatio) increase accordingly.

Table 9 compares ILS against RCRatio on our new benchmark instances (i.e. t1* and t2*). The latter achieves considerably higher profit gaps compared to the previously examined instances, especially when considering instances featuring tight time budgets (2.98% gap for the t2* instances). This agrees with the results compiled for c1*, r1* and rc1* Solomon instances, which possess similar characteristics. This improvement is mainly attributed to the RouteInitPhase incorporated into both our proposed algorithms, which increases the likelihood of initially inserting high-profit nodes located on far-reached clusters (such nodes are typically overlooked by ILS itineraries due to the high travel time, hence low ratio). The profit gap is also due to the randomization inherent to RCRatio's insertion step which enables better exploration of the solution space and allows escaping local optima. As far as the execution time is concerned, the two algorithms present comparable results.

Table 10 compares ILS against CSCRoutes on the new instances. ILS yields higher profit values in t1* instances, however, the performance gap is decreased compared to the results reported on previous instances. This is due to the wider and overlapping time windows chosen in t* instances, which diminishes the waiting time (until opening) and allows more effective use of the time budget by CSCRoutes. CSCRoutes performs much better with respect to the execution time. Interestingly, CSCRoutes prevails over ILS with respect to both performance indicators on t2* instances, especially on the overall profit.

Last, Table 11 compares RCRatio against CSCRoutes (negative values indicate prominence of RCRatio) on the t* instances. The general picture is that RCRatio prevails with respect to solutions quality, while CSCRoutes yields improved results with regard to the execution time.

Table 3. Comparison between ILS and RCRatio for Solomon instances

| | **1 tour Gap(%)** | | **2 tours Gap(%)** | | **3 tours Gap(%)** | | **4 tours Gap(%)** | |
|---|---|---|---|---|---|---|---|---|
| Name | Profit | Time | Profit | Time | Profit | Time | Profit | Time |
| c101 | -0.63 | -2.17 | 0 | 18.88 | 1.52 | 42.75 | -0.6 | 33.19 |
| c102 | 0 | -57.06 | 0 | 24.47 | 0.67 | 20.21 | 2.94 | 32.57 |
| c103 | 0.51 | -120 | 0 | -43.37 | 0.21 | 8.02 | 1.04 | -44.29 |
| c104 | 5 | -105.52 | 0.27 | -3.82 | -0.4 | -3.71 | -0.16 | 27.32 |
| c105 | 0 | -31.53 | 0 | 13.13 | 1.43 | 19.94 | 1.75 | 49.64 |
| c106 | 0 | -45.36 | 0 | 6.07 | 1.19 | 30.31 | 0.58 | 17.31 |
| c107 | 2.22 | -61.38 | 0 | -1.31 | 0.44 | 50.1 | 0.36 | -13.42 |
| c108 | 0 | -35.18 | 1.49 | -13.37 | 0.44 | 67.76 | 0.55 | 32.84 |
| c109 | 0 | -78.88 | 1.41 | -13.67 | 0.42 | -16.52 | -1.36 | 7.4 |
| Average | 0.79 | -59.68 | 0.35 | -1.44 | 0.66 | 24.32 | 0.57 | 15.84 |
| c201 | 3.57 | 23.49 | 2 | 15.4 | -0.23 | 17.19 | 0 | -109.34 |
| c202 | 0.44 | 34.73 | 0.84 | -28.88 | 0.8 | 3.19 | 0 | -124.81 |
| c203 | 0.85 | -1.61 | 0.98 | -61.13 | 0 | -76.47 | 0 | -133.86 |
| c204 | 1.68 | -77.19 | -0.68 | -64.69 | -0.45 | -110.2 | 0 | -153.14 |
| c205 | 0 | -23.15 | -0.28 | 36.2 | 0.11 | -65.64 | 0 | -125.31 |
| c206 | 1.1 | -37.28 | 1.53 | -47.36 | 1.02 | -110.4 | 0 | -160.73 |
| c207 | 1.76 | -2.63 | 1.1 | -17.56 | -1.22 | -10.88 | 0 | -130.97 |
| c208 | 1.08 | -49.81 | 0.68 | -39.28 | -0.77 | -30.31 | 0 | -136 |
| Average | 1.31 | -16.68 | 0.77 | -25.91 | -0.09 | -47.94 | 0 | -134.27 |
| r101 | 2.09 | -22.57 | 4.06 | 27.03 | -0.25 | 17.32 | 1.13 | 39.84 |
| r102 | 0 | -28.62 | -0.39 | 3.19 | -0.35 | 1.88 | 1.09 | 24.34 |
| r103 | 2.24 | -88.97 | 0 | -12.48 | 0.42 | 26.78 | 1.66 | -8.91 |
| r104 | 2.02 | -80.89 | 0.82 | -27.84 | 0.16 | 26.05 | 0.91 | 2.4 |
| r105 | 0 | 15.03 | 5.35 | 8.34 | 0.1 | 51.16 | 2.72 | 3.02 |
| r106 | 0 | -42.42 | 0 | 34.72 | -0.19 | 6.27 | 1.24 | -9.95 |
| r107 | 2.71 | -88.17 | 0.83 | -17.63 | -0.35 | 17.75 | -1.17 | -7.26 |
| r108 | 3.7 | -20.72 | 1.64 | -47.82 | -0.53 | 44.73 | -2.38 | -16.7 |
| r109 | 0.36 | -14.69 | 0.72 | 22.2 | 0.37 | 49.4 | 0.58 | 16.7 |
| r110 | 0 | -8.12 | -1.01 | 24.28 | 1.27 | 10.97 | 1.24 | -5.8 |
| r111 | 0.14 | -40.41 | 0.11 | 42.39 | -0.37 | 21.36 | -1.33 | 63.25 |
| r112 | 0.54 | -78.15 | 3.88 | 19 | 0.9 | 18.17 | 0.32 | 46.25 |
| Average | 1.15 | -41.56 | 1.33 | 6.28 | 0.1 | 24.32 | 0.5 | 12.27 |
| r201 | -0.58 | 20.51 | -1.27 | -36.37 | -0.04 | -31.98 | 0 | -80.72 |
| r202 | 1.2 | 14.54 | 1.84 | -15.72 | 0.33 | -62.63 | 0 | -111.47 |
| r203 | 0.06 | 35.77 | -0.26 | -61.92 | 0 | -107.21 | 0 | -124.88 |
| r204 | -0.95 | -70 | -0.68 | -116.96 | 0 | -163.93 | 0 | -171.44 |
| r205 | -1.55 | 38.43 | -0.81 | -63.54 | 0 | -124.29 | 0 | -157.9 |
| r206 | -0.54 | -51.89 | -0.03 | -99.28 | 0 | -142.69 | 0 | -170 |
| r207 | -0.58 | -67.88 | 0.42 | -136.04 | 0 | -154.21 | 0 | -180.18 |
| r208 | 0.56 | -8.74 | -0.05 | -184.44 | 0 | -173.39 | 0 | -274.9 |
| r209 | -0.71 | -55.88 | 1.29 | -54.4 | 0 | -138.23 | 0 | -231.63 |
| r210 | -0.17 | 6.54 | 0.53 | -43.32 | 0 | -135.97 | 0 | -136.47 |
| r211 | -0.9 | -85.46 | 0.2 | -119.36 | 0 | -162.1 | 0 | -171.64 |
| Average | -0.38 | -20.37 | 0.11 | -84.67 | 0.03 | -126.97 | 0 | -164.66 |
| rc101 | 0 | 1.67 | -0.75 | 61.91 | 1.79 | 35.07 | 0.23 | 46.07 |
| rc102 | 2.7 | -3.85 | 0.65 | 36.81 | 0.32 | 62.26 | 0.75 | 53.79 |
| rc103 | -0.38 | -32.11 | 0.66 | -9.87 | 0.72 | 31.54 | 0.53 | 22.61 |
| rc104 | 1.35 | -76.81 | 0.35 | 14.17 | -0.71 | -1.56 | 1.57 | 33.77 |
| rc105 | 9.86 | 1.52 | 3.4 | 35.74 | 2.32 | 16.06 | 1.33 | 9.53 |
| rc106 | 4.35 | 5.07 | 2.97 | 39.12 | 1.74 | 16.33 | 0.85 | 26.1 |
| rc107 | 0.88 | -6.48 | 1.13 | 37.63 | 1.26 | 27.88 | 0.63 | 18.46 |
| rc108 | 3.47 | -37.67 | 0.22 | 10.99 | 3.09 | -18.69 | -0.46 | 54.89 |
| Average | 2.78 | -18.58 | 1.08 | 28.31 | 1.32 | 21.11 | 0.68 | 33.15 |
| rc201 | -0.15 | 21.4 | 1.58 | -15.59 | 0.27 | -45.88 | 0 | -109.98 |
| rc202 | 1.97 | 21.4 | -0.82 | -11.1 | 0.23 | -61.87 | 0 | -113.92 |
| rc203 | -0.4 | -11.57 | -2.28 | -55.06 | 0 | -103.73 | 0 | -140.91 |
| rc204 | -1.24 | -48.37 | -0.12 | -118.98 | 0 | -137.88 | 0 | -176.39 |
| rc205 | 0.19 | 4.09 | 0.25 | -7.7 | -0.07 | -96.8 | 0 | -126.17 |
| rc206 | 0.28 | -12.58 | -2.31 | -65.21 | 0.52 | -104.42 | 0 | -136.72 |
| rc207 | -0.67 | 11.75 | -1.44 | -69.53 | 0.4 | -80.52 | 0 | -139.05 |
| rc208 | -1.39 | 18.01 | 0.66 | -18.17 | 0 | -149.45 | 0 | -117.65 |
| Average | -0.18 | 0.52 | -0.56 | -45.17 | 0.17 | -97.57 | 0 | -132.6 |

Table 4. Comparison between ILS and RCRatio for Cordeau et al. instances

| | **1 tour Gap(%)** | | **2 tours Gap(%)** | | **3 tours Gap(%)** | | **4 tours Gap(%)** | |
|---|---|---|---|---|---|---|---|---|
| Name | Profit | Time | Profit | Time | Profit | Time | Profit | Time |
| pr01 | 0.86 | -0.66 | 3.23 | -55.41 | -0.2 | -103.51 | 1.4 | -189.62 |
| pr02 | 1.92 | 30.53 | 4.18 | 3.78 | -0.4 | -29.73 | 0.53 | -48.63 |
| pr03 | 1.41 | -2.17 | -0.36 | 19.22 | 2.37 | 23.77 | -1.34 | 24.43 |
| pr04 | 4.52 | 33.7 | 0.35 | 39.58 | 1.49 | 58.55 | 1.42 | 39.46 |
| pr05 | -1.84 | 20.59 | 2.24 | 20.64 | 2.58 | 25.23 | 2.13 | 63.41 |
| pr06 | 1.04 | 15.99 | -1.48 | 41.91 | 1.38 | 62.91 | 2.06 | 28.3 |
| pr07 | 1.44 | -27.41 | 0.36 | 17.14 | 1.15 | 31.68 | 0.26 | -52.26 |
| pr08 | -1.9 | 35.24 | -0.08 | 22.29 | -1.11 | 8.27 | 1.4 | -0.16 |
| pr09 | 1.21 | 11.2 | -1.41 | 40.57 | 3.01 | 51.37 | 2.07 | 30.93 |
| pr10 | 0.41 | 12.9 | 4.48 | 39.48 | -2.63 | 52.29 | 0.68 | 32.11 |
| pr11 | 2.18 | 0.25 | -0.92 | -65.64 | 0.13 | -163.64 | 0.18 | -209.55 |
| pr12 | 0.37 | 6.62 | -0.22 | -12.2 | 2.75 | -68.1 | 1.42 | -92.57 |
| pr13 | -0.36 | -8.15 | 3.12 | 19.76 | 2.87 | 1.39 | 1.62 | 21.31 |
| pr14 | 6.89 | -9.7 | 0.37 | 23.78 | 4.86 | 25.86 | 0.98 | 38.93 |
| pr15 | 0.03 | 8.38 | -1.94 | 43.33 | 2.3 | 14.25 | 2.26 | -4.62 |
| pr16 | 5.76 | 65.36 | -1.01 | 33.52 | 1.96 | 24.07 | -1.82 | 37.21 |
| pr17 | 3.93 | 14.6 | 0.16 | 48.21 | -1.53 | -90.86 | -0.27 | -127.39 |
| pr18 | 2.88 | 35.53 | 1.03 | -12.15 | 0.57 | -5.74 | 3.02 | -0.01 |
| pr19 | 3.57 | 21.95 | -1.34 | 26.4 | 6.83 | 12.66 | 2.67 | 15.83 |
| pr20 | 5.3 | 16.47 | 7.88 | 25.09 | 1.16 | 52.4 | 3.98 | 23.65 |
| Average | 1.98 | 14.06 | 0.93 | 15.97 | 1.48 | -0.84 | 1.23 | -18.46 |

Table 5. Comparison between ILS and CSCRoutes for Solomon instances

| | **1 tour Gap(%)** | | **2 tours Gap(%)** | | **3 tours Gap(%)** | | **4 tours Gap(%)** | |
|---|---|---|---|---|---|---|---|---|
| Name | Profit | Time | Profit | Time | Profit | Time | Profit | Time |
| c101 | -6.25 | 12.5 | -6.78 | 48.44 | -2.53 | 63.14 | -1 | 45.37 |
| c102 | 0 | -12.84 | 0 | 47.89 | 0 | 39.16 | 0 | 52.81 |
| c103 | -2.56 | -53.04 | 0 | -13.43 | -3.13 | 43.23 | -0.87 | -3.38 |
| c104 | 2.5 | -66.21 | -4 | 31.46 | -0.99 | 25.89 | -0.82 | 57.53 |
| c105 | -2.94 | 1.8 | -6.25 | 33.13 | -2.38 | 51.47 | -0.97 | 70.19 |
| c106 | -2.94 | -22.32 | -3.23 | 8.67 | -1.19 | 51.09 | -1.92 | 28 |
| c107 | 0 | -6.9 | -7.46 | 40.91 | -3.33 | 68.25 | -3.64 | 22.48 |
| c108 | -2.7 | -5.11 | -4.48 | 30.81 | -1.11 | 67.85 | -1.82 | 49.64 |
| c109 | 0 | -29.6 | -2.82 | 13.53 | 1.05 | 8.18 | -1.69 | 15.33 |
| Average | -1.66 | -20.19 | -3.89 | 26.82 | -1.51 | 46.47 | -1.41 | 37.55 |
| c201 | 0 | 72.33 | 1.43 | 62.04 | 1.14 | 64.47 | 0 | -3.3 |
| c202 | -2.2 | 79.45 | -0.7 | 61.83 | 1.14 | 57.01 | 0 | -40.4 |
| c203 | -3.19 | 76.33 | 0.7 | 45.3 | -0.57 | 40.15 | 0 | -16.19 |
| c204 | 2.11 | 60.94 | 0 | 50.52 | 0 | 26.39 | 0 | -31.96 |
| c205 | -1.11 | 54.47 | -0.69 | 76.55 | 0 | 19.54 | 0 | -12.98 |
| c206 | 0 | 52.78 | 1.39 | 53.35 | 1.13 | 21.62 | 0 | -31 |
| c207 | 1.1 | 66.9 | 0.69 | 61.05 | -0.55 | 49.14 | 0 | -12.09 |
| c208 | -1.08 | 56.55 | 0.68 | 53.12 | -0.55 | 52.14 | 0 | -11.04 |
| Average | -0.55 | 64.97 | 0.44 | 57.97 | 0.22 | 41.31 | 0 | -19.87 |
| r101 | -1.1 | -20 | -1.52 | 46.86 | -5.82 | 56.01 | -4.16 | 68.61 |
| r102 | -1.4 | 6.15 | -1.38 | 43.36 | -2.77 | 41.72 | 0.12 | 50.13 |
| r103 | 1.05 | -26.72 | -1.75 | 8.96 | -2.92 | 59.98 | -1.71 | 16.81 |
| r104 | 2.02 | -24.44 | -1.86 | 16.13 | -2.35 | 54.03 | -0.32 | 46.68 |
| r105 | -3.64 | 39.22 | -1.86 | 34.14 | -4.43 | 66.81 | -1.36 | 22.08 |
| r106 | -4.78 | -7.26 | -3.78 | 49.48 | -1.67 | 19.36 | -2.87 | 18.99 |
| r107 | 0.35 | -18.26 | -1.13 | 23.97 | -1.07 | 44.21 | -2.91 | 22.66 |
| r108 | 2.02 | 14.43 | -0.73 | 17.04 | -2.91 | 66.23 | -3.56 | 20 |
| r109 | -6.16 | 24.48 | -3.01 | 56.98 | -4.58 | 60.86 | -1.39 | 39.6 |
| r110 | 0 | 23.64 | -1.75 | 39.21 | -2.39 | 14.18 | 0.69 | 39.17 |
| r111 | 0.68 | 6.08 | 0.56 | 66 | -0.65 | 48.91 | -3.21 | 63.36 |
| r112 | -3.39 | -14.62 | 3.11 | 26.2 | -0.26 | 43.53 | -0.64 | 50.78 |
| Average | -1.2 | 0.22 | -1.26 | 35.69 | -2.65 | 47.99 | -1.78 | 38.24 |
| r201 | -39.59 | 84.08 | -25.35 | 68.02 | -17.4 | 30.07 | -9.95 | 4.85 |
| r202 | -16.14 | 82.92 | -12.2 | 62.89 | -5.06 | 24.11 | -2.67 | -54.36 |
| r203 | -6.73 | 86.93 | -12.42 | 20.89 | -4.46 | 1.3 | -0.75 | -48.49 |
| r204 | -2.33 | 67.67 | -5.28 | 29.17 | 0 | -11.48 | 0 | -45.41 |
| r205 | -30.83 | 90.22 | -20.63 | 62.42 | -1.92 | 0.42 | 0 | -49.18 |
| r206 | -14.76 | 71 | -11.06 | 34.7 | -1.92 | -23.32 | 0 | -68.75 |
| r207 | -11.66 | 67.6 | -2.87 | 17.34 | -0.14 | -65.15 | 0 | -67.43 |
| r208 | -0.75 | 80.35 | -2.74 | 16.98 | 0 | 6.06 | 0 | -44.59 |
| r209 | -23.43 | 71.15 | -9.59 | 60.16 | -2.4 | -5.28 | 0 | -55.78 |
| r210 | -19.31 | 82.79 | -9.08 | 53.92 | -3.22 | -44.78 | 0 | -51.1 |
| r211 | -15.15 | 58.67 | -4.71 | 28.09 | 0 | -4.03 | 0 | -43.18 |
| Average | -16.43 | 76.67 | -10.54 | 41.33 | -3.32 | -8.37 | -1.22 | -47.58 |
| rc101 | -1.37 | 15.74 | -1.87 | 72.4 | -0.99 | 61.74 | -0.76 | 62.4 |
| rc102 | 2.7 | 20.74 | -1.01 | 42.75 | -1 | 77.81 | -0.68 | 64.69 |
| rc103 | 0.38 | 1.75 | -0.19 | 18.24 | 1.34 | 50 | -1.9 | 42.07 |
| rc104 | 1.35 | -36.17 | -1.06 | 46.22 | -0.61 | 27.35 | -2.06 | 56.39 |
| rc105 | 9.05 | 21.97 | -6.32 | 48.68 | 0.31 | 39.36 | -2.5 | 36.42 |
| rc106 | 4.6 | 28.17 | 0 | 48.79 | -2.95 | 36.87 | 0.8 | 39.13 |
| rc107 | -4.74 | 20.69 | -4.08 | 45.23 | -1.61 | 33.29 | -1.16 | 52.24 |
| rc108 | -4.86 | 6.67 | -2.01 | 48.13 | -0.4 | 15.6 | -3.01 | 70.82 |
| Average | 0.89 | 9.95 | -2.07 | 46.31 | -0.74 | 42.75 | -1.41 | 53.02 |
| rc201 | -18.97 | 78.39 | -21.07 | 67.66 | -17.42 | 31.72 | -9.63 | -63.07 |
| rc202 | -2.04 | 76.95 | -20.81 | 71.11 | -13.29 | 33.12 | -6.44 | -51.87 |
| rc203 | -6.15 | 75.03 | -8.46 | 57.19 | -6.15 | 6.73 | -0.87 | -27.96 |
| rc204 | 0.36 | 66 | -1.69 | 46.88 | -0.81 | 0.24 | 0 | -55.79 |
| rc205 | -13.57 | 74.85 | -18.25 | 75.6 | -11.51 | 21.81 | -5.34 | -23.13 |
| rc206 | -12.91 | 69.1 | -9.36 | 51.09 | -6.97 | 28.14 | 0 | -53.45 |
| rc207 | -10.69 | 74.66 | -7.05 | 52.64 | -5.2 | 26.4 | -0.58 | -67.13 |
| rc208 | -6.36 | 78.03 | -0.44 | 57.13 | 0 | -2.86 | 0 | 0 |
| Average | -8.79 | 74.13 | -10.89 | 59.91 | -7.67 | 18.16 | -2.86 | -42.8 |

Table 6. Comparison between ILS and CSCRoutes for Cordeau et al. instances

| | **1 tour Gap(%)** | | **2 tours Gap(%)** | | **3 tours Gap(%)** | | **4 tours Gap(%)** | |
|---|---|---|---|---|---|---|---|---|
| Name | Profit | Time | Profit | Time | Profit | Time | Profit | Time |
| pr01 | -18.75 | 54.1 | -7.22 | -2.7 | -1.17 | 12.37 | 0.78 | -62.03 |
| pr02 | -2.86 | 73.32 | -4.39 | 66.13 | -6.79 | 40.43 | -6.02 | 24.92 |
| pr03 | -9.11 | 47.47 | -9.52 | 63.99 | -6.98 | 65.93 | -6.54 | 63.64 |
| pr04 | -4.92 | 73.05 | -1.97 | 69.75 | -8.7 | 78.28 | -4.34 | 64.08 |
| pr05 | -23.09 | 73.1 | -15.13 | 71.75 | -7.08 | 65.4 | -2.7 | 83.25 |
| pr06 | -12.27 | 61.72 | -14.34 | 71.67 | -13.15 | 80.79 | -8.37 | 65.58 |
| pr07 | 0 | 35.56 | 0.91 | 64.6 | -2.24 | 67.74 | -7.02 | 34.88 |
| pr08 | -12.53 | 74.27 | -12.56 | 69.89 | -9.61 | 60.05 | -5.76 | 49.2 |
| pr09 | -5.21 | 60.78 | -10.03 | 71.1 | 0.7 | 75.98 | -6.16 | 57.98 |
| pr10 | -8.72 | 61.61 | -7.17 | 68.82 | -13.78 | 74.66 | -10.94 | 63.86 |
| pr11 | -1.82 | 49.37 | -3.14 | 1.66 | -0.95 | -15.91 | 0 | -35.82 |
| pr12 | -5.34 | 59.85 | -4.26 | 56.17 | -1.11 | 23.24 | -1.73 | 19.87 |
| pr13 | -3.78 | 53.64 | 0.53 | 65.89 | -6.02 | 54.16 | -6.81 | 48.97 |
| pr14 | -6.02 | 57.49 | 0.22 | 63.16 | -1 | 63.05 | -6.48 | 66.94 |
| pr15 | -2.98 | 72.3 | -13.68 | 74.79 | -5.91 | 52.05 | -3.8 | 49.2 |
| pr16 | -5.01 | 86.42 | -17.48 | 68.92 | -1.01 | 62.17 | -14.61 | 69.96 |
| pr17 | -4.34 | 60.7 | -3.37 | 78.74 | -0.87 | 19.01 | -0.79 | -27.86 |
| pr18 | -20.04 | 77.04 | -14.94 | 62.02 | -8.67 | 45.58 | -0.89 | 48.04 |
| pr19 | -2.4 | 67.02 | -8.59 | 68.8 | -2.91 | 58.15 | -4.29 | 52.83 |
| pr20 | -8.42 | 70.18 | -11.36 | 65.63 | -2.91 | 77.41 | -1.84 | 53.14 |
| Average | -7.88 | 63.45 | -7.88 | 61.04 | -5.01 | 53.03 | -4.92 | 39.53 |

Table 7. Comparison between RCRatio and CSCRoutes for Solomon instances

| | **1 tour** Gap(%) | | **2 tours** Gap(%) | | **3 tours** Gap(%) | | **4 tours** Gap(%) | |
|---|---|---|---|---|---|---|---|---|
| Name | Profit | Time | Profit | Time | Profit | Time | Profit | Time |
| c101 | -5.66 | 14.36 | -6.78 | 36.44 | -3.99 | 35.62 | -0.4 | 18.23 |
| c102 | 0 | 28.15 | 0 | 31.01 | -0.67 | 23.76 | -2.85 | 30.02 |
| c103 | -3.06 | 30.43 | 0 | 20.88 | -3.33 | 38.28 | -1.89 | 28.35 |
| c104 | -2.38 | 19.13 | -4.26 | 33.98 | -0.6 | 28.53 | -0.66 | 41.56 |
| c105 | -2.94 | 25.34 | -6.25 | 23.02 | -3.76 | 39.38 | -2.67 | 40.81 |
| c106 | -2.94 | 15.85 | -3.23 | 2.77 | -2.35 | 29.82 | -2.49 | 12.93 |
| c107 | -2.17 | 33.76 | -7.46 | 41.67 | -3.76 | 36.37 | -3.99 | 31.65 |
| c108 | -2.7 | 22.25 | -5.88 | 38.97 | -1.55 | 0.26 | -2.35 | 25.01 |
| c109 | 0 | 27.55 | -4.17 | 23.93 | 0.63 | 21.2 | -0.34 | 8.56 |
| Average | -2.43 | 24.09 | -4.22 | 28.08 | -2.15 | 28.14 | -1.96 | 26.35 |
| c201 | -3.45 | 63.83 | -0.56 | 55.14 | 1.37 | 57.1 | 0 | 50.66 |
| c202 | -2.63 | 68.52 | -1.53 | 70.38 | 0.34 | 55.6 | 0 | 37.55 |
| c203 | -4.01 | 76.71 | -0.28 | 66.05 | -0.57 | 66.08 | 0 | 50.32 |
| c204 | 0.41 | 77.95 | 0.69 | 69.95 | 0.45 | 64.98 | 0 | 47.87 |
| c205 | -1.11 | 63.03 | -0.41 | 63.25 | -0.11 | 51.43 | 0 | 49.86 |
| c206 | -1.09 | 65.61 | -0.14 | 68.34 | 0.11 | 62.75 | 0 | 49.76 |
| c207 | -0.65 | 67.75 | -0.41 | 66.86 | 0.67 | 54.13 | 0 | 51.47 |
| c208 | -2.13 | 71 | 0 | 66.34 | 0.22 | 63.27 | 0 | 52.95 |
| Average | -1.83 | 69.3 | -0.33 | 65.79 | 0.31 | 59.42 | 0 | 48.8 |
| r101 | -3.12 | 2.1 | -5.36 | 27.18 | -5.59 | 46.79 | -5.23 | 47.82 |
| r102 | -1.4 | 27.03 | -0.99 | 41.5 | -2.43 | 40.6 | -0.96 | 34.08 |
| r103 | -1.16 | 32.94 | -1.75 | 19.06 | -3.32 | 45.34 | -3.32 | 23.62 |
| r104 | 0 | 31.2 | -2.65 | 34.39 | -2.51 | 37.83 | -1.22 | 45.36 |
| r105 | -3.64 | 28.46 | -6.84 | 28.14 | -4.53 | 32.04 | -3.97 | 19.65 |
| r106 | -4.78 | 24.69 | -3.78 | 22.61 | -1.48 | 13.97 | -4.06 | 26.31 |
| r107 | -2.3 | 37.15 | -1.95 | 35.36 | -0.73 | 32.17 | -1.77 | 27.9 |
| r108 | -1.62 | 29.12 | -2.33 | 43.88 | -2.39 | 38.89 | -1.21 | 31.45 |
| r109 | -6.5 | 34.15 | -3.71 | 44.71 | -4.93 | 22.66 | -1.95 | 27.49 |
| r110 | 0 | 29.37 | -0.75 | 19.72 | -3.61 | 3.6 | -0.54 | 42.5 |
| r111 | 0.54 | 33.11 | 0.45 | 40.99 | -0.29 | 35.03 | -1.91 | 0.3 |
| r112 | -3.91 | 35.66 | -0.75 | 8.89 | -1.15 | 30.99 | -0.96 | 8.43 |
| Average | -2.32 | 28.75 | -2.53 | 30.54 | -2.75 | 31.66 | -2.26 | 27.91 |
| r201 | -39.24 | 79.97 | -24.39 | 76.55 | -17.37 | 47.01 | -9.95 | 47.35 |
| r202 | -17.13 | 80.02 | -13.79 | 67.93 | -5.37 | 53.34 | -2.67 | 27 |
| r203 | -6.79 | 79.65 | -12.19 | 51.15 | -4.46 | 52.37 | -0.75 | 33.97 |
| r204 | -1.39 | 80.98 | -4.63 | 67.36 | 0 | 57.76 | 0 | 46.43 |
| r205 | -29.74 | 84.11 | -19.98 | 77.02 | -1.92 | 55.6 | 0 | 42.16 |
| r206 | -14.29 | 80.9 | -11.04 | 67.23 | -1.92 | 49.19 | 0 | 37.5 |
| r207 | -11.14 | 80.7 | -3.28 | 64.98 | -0.14 | 35.03 | 0 | 40.24 |
| r208 | -1.3 | 81.93 | -2.69 | 70.81 | 0 | 65.64 | 0 | 61.43 |
| r209 | -22.88 | 81.49 | -10.75 | 74.2 | -2.4 | 55.81 | 0 | 53.03 |
| r210 | -19.18 | 81.59 | -9.56 | 67.85 | -3.22 | 38.65 | 0 | 36.1 |
| r211 | -14.38 | 77.71 | -4.9 | 67.22 | 0 | 60.31 | 0 | 47.29 |
| Average | -16.13 | 80.82 | -10.65 | 68.39 | -3.35 | 51.88 | -1.22 | 42.95 |
| rc101 | -1.37 | 14.31 | -1.13 | 27.55 | -2.73 | 41.07 | -0.98 | 30.29 |
| rc102 | 0 | 23.68 | -1.65 | 9.4 | -1.31 | 41.21 | -1.42 | 23.58 |
| rc103 | 0.76 | 25.63 | -0.84 | 25.59 | 0.61 | 26.97 | -2.42 | 25.15 |
| rc104 | 0 | 22.98 | -1.41 | 37.34 | 0.1 | 28.47 | -3.57 | 34.16 |
| rc105 | -0.74 | 20.77 | -9.4 | 20.15 | -1.97 | 27.76 | -3.78 | 29.72 |
| rc106 | 0.24 | 24.33 | -2.88 | 15.88 | -4.61 | 24.55 | -0.05 | 17.63 |
| rc107 | -5.57 | 25.52 | -5.15 | 12.18 | -2.84 | 7.51 | -1.78 | 41.43 |
| rc108 | -8.05 | 32.2 | -2.23 | 41.73 | -3.38 | 28.88 | -2.56 | 35.32 |
| Average | -1.84 | 23.68 | -3.09 | 23.73 | -2.02 | 28.3 | -2.07 | 29.66 |
| rc201 | -18.85 | 72.51 | -22.3 | 72.02 | -17.64 | 53.19 | -9.63 | 22.34 |
| rc202 | -3.94 | 70.67 | -20.15 | 74 | -13.48 | 58.68 | -6.44 | 29 |
| rc203 | -5.77 | 77.62 | -6.32 | 72.39 | -6.15 | 54.22 | -0.87 | 46.88 |
| rc204 | 1.61 | 77.08 | -1.57 | 75.74 | -0.81 | 58.06 | 0 | 43.63 |
| rc205 | -13.74 | 73.78 | -18.45 | 77.35 | -11.45 | 60.27 | -5.34 | 45.56 |
| rc206 | -13.15 | 72.55 | -7.22 | 70.4 | -7.44 | 64.85 | 0 | 35.18 |
| rc207 | -10.09 | 71.29 | -5.7 | 72.06 | -5.57 | 59.23 | -0.58 | 30.09 |
| rc208 | -5.05 | 73.21 | -1.09 | 63.72 | 0 | 58.76 | 0 | 54.05 |
| Average | -8.62 | 73.59 | -10.35 | 72.21 | -7.82 | 58.41 | -2.86 | 38.34 |

Table 8. Comparison between RCRatio and CSCRoutes for Cordeau et al. instances

| | **1 tour Gap(%)** | | **2 tours Gap(%)** | | **3 tours Gap(%)** | | **4 tours Gap(%)** | |
|---|---|---|---|---|---|---|---|---|
| Name | Profit | Time | Profit | Time | Profit | Time | Profit | Time |
| pr01 | -19.44 | 54.4 | -10.12 | 33.91 | -0.97 | 56.94 | -0.61 | 44.06 |
| pr02 | -4.69 | 61.59 | -8.23 | 64.8 | -6.41 | 54.08 | -6.51 | 49.48 |
| pr03 | -10.37 | 48.59 | -9.19 | 55.42 | -9.13 | 55.31 | -5.27 | 51.88 |
| pr04 | -9.03 | 59.35 | -2.31 | 49.93 | -10.04 | 47.61 | -5.68 | 40.67 |
| pr05 | -21.65 | 66.12 | -16.99 | 64.41 | -9.42 | 53.73 | -4.73 | 54.22 |
| pr06 | -13.17 | 54.44 | -13.05 | 51.23 | -14.34 | 48.22 | -10.23 | 51.99 |
| pr07 | -1.42 | 49.42 | 0.54 | 57.27 | -3.36 | 52.77 | -7.27 | 57.23 |
| pr08 | -10.83 | 60.26 | -12.5 | 61.25 | -8.6 | 56.45 | -7.07 | 49.28 |
| pr09 | -6.34 | 55.83 | -8.75 | 51.38 | -2.24 | 50.61 | -8.07 | 39.16 |
| pr10 | -9.09 | 55.92 | -11.15 | 48.48 | -11.45 | 46.89 | -11.55 | 46.77 |
| pr11 | -3.91 | 49.24 | -2.23 | 40.63 | -1.07 | 56.03 | -0.18 | 56.12 |
| pr12 | -5.69 | 57.01 | -4.05 | 60.93 | -3.75 | 54.34 | -3.11 | 58.39 |
| pr13 | -3.43 | 57.13 | -2.51 | 57.49 | -8.64 | 53.51 | -8.29 | 35.14 |
| pr14 | -12.07 | 61.25 | -0.15 | 51.67 | -5.59 | 50.17 | -7.39 | 45.86 |
| pr15 | -3.01 | 69.77 | -11.97 | 55.52 | -8.03 | 44.08 | -5.92 | 51.44 |
| pr16 | -10.18 | 60.81 | -16.64 | 53.25 | -2.92 | 50.17 | -13.03 | 52.15 |
| pr17 | -7.95 | 53.99 | -3.52 | 58.94 | 0.68 | 57.57 | -0.52 | 43.77 |
| pr18 | -22.28 | 64.39 | -15.8 | 66.13 | -9.18 | 48.53 | -3.79 | 48.04 |
| pr19 | -5.77 | 57.74 | -7.34 | 57.61 | -9.12 | 52.09 | -6.78 | 43.95 |
| pr20 | -13.03 | 64.3 | -17.84 | 54.12 | -4.02 | 52.53 | -5.6 | 38.62 |
| Average | -9.67 | 58.08 | -8.69 | 54.72 | -6.38 | 52.08 | -6.08 | 47.91 |

Table 9. Comparison between ILS and RCRatio for new instances

| **t1** | | | **t2** | | |
|---|---|---|---|---|---|
| Name | Profit Gap(%) | Time Gap(%) | Name | Profit Gap(%) | Time Gap(%) |
| t101 | 0.1 | -18.06 | t201 | 1.2 | -104.09 |
| t102 | -1.14 | -18.65 | t202 | 0 | -30.17 |
| t103 | 3.16 | -6.81 | t203 | 3.1 | -109.74 |
| t104 | 1.3 | -2.84 | t204 | 0 | 6.3 |
| t105 | -0.79 | -44.07 | t205 | -0.72 | 61.17 |
| t106 | -0.6 | -16.02 | t206 | 2.76 | 5.02 |
| t107 | -1.17 | 32.31 | t207 | 7.01 | -58.08 |
| t108 | 0.17 | 21.21 | t208 | 8.64 | -82.69 |
| t109 | -2.53 | 5.78 | t209 | 1.19 | 37.01 |
| t110 | 0.4 | 14.65 | t210 | 3.28 | 23.05 |
| t111 | -1.32 | 19.82 | t211 | 0.25 | 32.27 |
| t112 | 0.65 | -47.94 | t212 | 0.91 | 17.51 |
| t113 | -3.04 | 36.4 | t213 | -0.04 | 7.76 |
| t114 | 1.97 | -53.06 | t214 | 4.97 | 19.86 |
| t115 | -0.98 | 0.08 | t215 | 0.71 | 4.38 |
| t116 | 1 | -61.25 | t216 | 3.97 | 6.31 |
| t117 | 1.46 | -3.91 | t217 | 0.43 | 55.1 |
| t118 | -0.61 | -36.04 | t218 | 0 | -48.42 |
| t119 | 0.89 | -50.46 | t219 | 5.37 | 6.24 |
| t120 | -2.7 | 27.44 | t220 | 1.92 | 52.19 |
| t121 | 3.02 | -82.62 | t221 | 8.64 | -13.64 |
| t122 | 1.62 | 32.9 | t222 | 1.31 | -7.83 |
| t123 | 0.84 | -78.6 | t223 | 17.6 | -186.76 |
| t124 | 8.18 | -142.43 | t224 | 1 | 32.23 |
| t125 | -1.04 | 5.45 | t225 | 1.95 | 40.74 |
| t126 | 0.63 | -28.41 | t226 | 1.58 | 28.09 |
| t127 | 0.64 | 28.72 | t227 | 0 | -105.59 |
| t128 | -1.35 | -4.21 | t228 | 0.93 | 3.56 |
| t129 | 2.08 | -80.07 | t229 | 0 | -50.14 |
| t130 | 1.08 | -49.7 | t230 | 3.13 | -23.33 |
| t131 | 3.4 | -50.61 | t231 | -0.48 | -5.36 |
| t132 | -0.05 | -36.01 | t232 | 4.48 | 33.26 |
| t133 | 1.35 | 23.8 | t233 | 16.11 | -59.69 |
| t134 | 1.12 | 28.02 | t234 | 4.92 | 34.78 |
| t135 | -1.31 | 36.17 | t235 | 3.06 | -21.69 |
| t136 | -0.13 | -49.37 | t236 | 0.23 | -76.2 |
| t137 | -2.11 | 0.8 | t237 | 1.01 | 30.02 |
| t138 | 0.28 | -14.83 | t238 | -0.19 | 0.44 |
| t139 | 4.95 | -15.12 | t239 | 0.2 | 51.05 |
| t140 | 2.16 | -26.53 | t240 | 6.67 | -30.81 |
| t141 | 2.15 | -8.28 | t241 | 1.18 | -140.71 |
| t142 | -0.69 | -21.65 | t242 | 0 | -23.4 |
| t143 | 1.11 | -94.98 | t243 | 17.29 | -130.56 |
| t144 | -2.02 | -3.51 | t244 | 2.3 | -37.94 |
| t145 | 2.91 | -85.66 | t245 | 2.75 | -4.23 |
| t146 | 0.76 | 0.91 | t246 | 1.54 | -12.66 |
| t147 | 0.35 | 26.14 | t247 | 1.08 | 28.86 |
| t148 | 0.47 | -86.44 | t248 | 4.72 | 12.27 |
| t149 | 0.02 | 24.46 | t249 | 0.88 | 38.52 |
| t150 | -0.9 | -42.13 | t250 | 0.4 | 18.58 |
| Average | 0.52 | -19.9 | Average | 2.98 | -13.54 |

## 5. Conclusions and Future Work

We introduced RCRatio and CSCRoutes, two cluster-based heuristic approaches for the TOPTW. The main design objectives of the two algorithms address the shortcomings of the best known so far real-time TOPTW algorithm, ILS. The main incentive behind our approaches is to favor visits to topology areas featuring high density of good candidate nodes. This is achieved through a clustering phase which groups nodes based on geographical criteria, thereby increasing the probability of visiting “promising” topology areas, even distant ones.

The comparison of RCRatio over the best known real-time TOPTW algorithm (ILS) demonstrated that RCRatio achieves higher quality solutions in comparable execution time, certainly satisfying the requirements of real-time TTDP solvers, even for large-scale topologies. As regards the comparison of CSCRoutes over ILS, the latter yields solutions with higher profit. However, CSCRoutes achieves the best performance results with respect to execution time, compared to both ILS and RCRatio. Notably, the performance gap of our algorithms over ILS increases when tested on realistic TTDP instances, wherein POIs typically feature wide, overlapping time windows and are located nearby each other, while the daily time budget is 5-10h.

Based on the above findings, our two cluster-based heuristics may be thought of as complementary TOPTW algorithmic options. We argue that the choice among RCRatio and CSCRoutes (when considering real-world online TTDP applications) should be determined by user-stated preferences. For instance, a user willing to partially trade the quality of derived solutions with fast obtained itineraries mostly including POIs located nearby each other should opt for the CSCRoutes algorithm.

Our future work will focus on variants of TOPTW to tackle more realistic TTDPs. For instance, tourists typically require relaxing and having breaks (e.g. for coffee and meal) in between of visits to POIs. Such breaks are typically specific in number, while respective recommendations may be subject to strict time window (e.g. meal should be scheduled around noon) and budget constraints. Further, we plan to incorporate max-n type (Souffriau and Vansteenwegen, 2010) restrictions to constrain the selection of POIs by allowing users to set a maximum number of certain types of POIs, per day or for the whole trip (e.g., maximum two museum visits on the first day). Likewise, mandatory visits (i.e. tours including at least one visit to a POI of certain type, such as a visit to a church) could also be asked for. Focused adjustments and refinements of our algorithms should be able to provide such features.

**Acknowledgments**

This work has been supported by the EU FP7/2007-2013 Programme under grant agreements no. 288094 (eCOMPASS - ”eco-Friendly Urban Multi-Modal Route Planning Services for Mobile Users”) and no. 621133 (HoPE - ”Holistic Personal public Eco-mobility”).

**References**

Archetti, C, Hertz, A and Speranza, M (2007) "Metaheuristics for the team orienteering problem," *Journal of Heuristics* 13, 49–76.

Bagirov, M (2008) "Modified global k-means algorithm for minimum sum-of-squares clustering problems," *Pattern Recognition* 41(10), 3192 – 3199.

Butt, SE and Cavalier, TM (1994) "A heuristic for the multiple tour maximum collection problem," *Computers & Operations Research* 21(1), 101 – 111.

Chao, IM, Golden, BL and Wasil, EA (1996) "The team orienteering problem," *European Journal of Operational Research* 88(3), 464 – 474.

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

Gambardella, L, Montemanni, R and Weyland, D (2012) "Coupling ant colony systems with strong local searches," *European Journal of Operational Research* 220(3), 831 – 843.

Gavalas, D, Konstantopoulos, C, Mastakas, K and Pantziou, G (2014) "A survey on algorithmic approaches for solving tourist trip design problems," *Journal of Heuristics* 20(3), 291–328.

Gavalas, D, Konstantopoulos, C, Mastakas, K, Pantziou, GE and Tasoulas, Y (2013) "Cluster-based heuristics for the team orienteering problem with time windows," in *Experimental Algorithms, 12th International Symposium, SEA 2013, Rome, Italy, June 5-7, 2013. Proceedings*, pp. 390–401.

Golden, BL, Levy, L and Vohra, R (1987) "The orienteering problem," *Naval Research Logistics (NRL)* 34(3), 307–318.

Guibadj, RN and Moukrim, A (2013) "Memetic algorithm with an efficient split procedure for the team orienteering problem with time windows," in *Artificial Evolution 2013*, pp. 6–17.

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

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

Labadi, N, Melechovský, J and Calvo, R (2010) "An effective hybrid evolutionary local search for orienteering and team orienteering problems with time windows," in R. Schaefer, C. Cotta, J. Kolodziej and G. Rudolph (eds.), *Parallel Problem Solving from Nature – PPSN XI*, Lecture Notes in Computer Science, Vol. 6239, pp. 219–228.

Labadi, N, Melechovský, J and Wolfler Calvo, R (2011) "Hybridized evolutionary local search algorithm for the team orienteering problem with time windows," *Journal of Heuristics* 17, 729–753.

Laporte, G and Martello, S (1990) "The selective travelling salesman problem," *Discrete Applied Mathematics* 26(2-3), 193 – 207.

Li, Z and Hu, X (2011) "The team orienteering problem with capacity constraint and time window," *The 10th International Symposium on Operations Research and its Applications (ISORA 2011)* , 157–163.

Likas, A, Vlassis, N and Verbeek, J (2003) "The global k-means clustering algorithm," *Pattern Recognition* 36(2), 451 – 461.

Lin, SW and Yu, VF (2012) "A simulated annealing heuristic for the team orienteering problem with time windows," *European Journal of Operational Research* 217(1), 94 – 107.

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

Solomon, M (1987) "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," *Operations Research* 35, 254–265.

Souffriau, W and Vansteenwegen, P (2010) "Tourist trip planning functionalities: State– of– the –art and future," in *Proceedings of the 10th International Conference on Current Trends in Web Engineering (ICWE'10)*, pp. 474–485, URL http://dx.doi.org/10.1007/978-3-642-16985-4_46.

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

Vansteenwegen, P (2008) *Planning in Tourism and Public Transportation - Attraction Selection by Means of a Personalised Electronic Tourist Guide and Train Transfer Scheduling*, PhD thesis, Katholieke Universiteit Leuven.

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

Vansteenwegen, P, Souffriau, W, Vanden Berghe, G and Van Oudheusden, D (2009) "Iterated local search for the team orienteering problem with time windows," *Computers & Operations Research* 36, 3281–3290.

Vansteenwegen, P, Souffriau, W, Vanden Berghe, G and Van Oudheusden, D (2011b) "The city trip planner: An expert system for tourists," *Expert Systems with Applications* 38(6), 6540 – 6546.

Vansteenwegen, P and Van Oudheusden, D (2007) "The mobile tourist guide: An or opportunity," *Operational Research Insight* 20(3), 21–27.

## Appendix A. Analytical Results

Table 10. Comparison between ILS and CSCRoutes for new instances

| t1 | | | t2 | | |
|---|---|---|---|---|---|
| Name | Profit Gap(%) | Time Gap(%) | Name | Profit Gap(%) | Time Gap(%) |
| t101 | -3.1 | 40.45 | t201 | -3.28 | -20.45 |
| t102 | -2.72 | 20.79 | t202 | 0 | -6.78 |
| t103 | 0.13 | 29.24 | t203 | 2.3 | -51.32 |
| t104 | -1.49 | 37.14 | t204 | 0 | 19.34 |
| t105 | 0 | -14.62 | t205 | -4.92 | 65.14 |
| t106 | -0.77 | -2.34 | t206 | 0 | 1.26 |
| t107 | -2.67 | 52.52 | t207 | 15.52 | -26.92 |
| t108 | 0.84 | 34.09 | t208 | 8.64 | -76.92 |
| t109 | -2.15 | 40.58 | t209 | -0.44 | 39.01 |
| t110 | 0.12 | 40.64 | t210 | 3.12 | 34.7 |
| t111 | -1.58 | 43.33 | t211 | 2.33 | 45.85 |
| t112 | 1.63 | -32.22 | t212 | 0.87 | 21.83 |
| t113 | -3.12 | 53.22 | t213 | -2.61 | 15.95 |
| t114 | -2.36 | 18.91 | t214 | 1.29 | 41.82 |
| t115 | -1.42 | 35.17 | t215 | -2.36 | 17.53 |
| t116 | -0.83 | -13.34 | t216 | 0.22 | 18.71 |
| t117 | 2.21 | 29.76 | t217 | 0 | 50.94 |
| t118 | 0 | -2.2 | t218 | 0 | -36.84 |
| t119 | -0.43 | -3.15 | t219 | 2.11 | 5.6 |
| t120 | -3.62 | 61.1 | t220 | -0.9 | 59.51 |
| t121 | 1.18 | -19.37 | t221 | 1.07 | -4.11 |
| t122 | 0.85 | 67.64 | t222 | -1.01 | -1.99 |
| t123 | 1.24 | -17.19 | t223 | 25.14 | -143.24 |
| t124 | 8.28 | -25.24 | t224 | 0.25 | 49.19 |
| t125 | -4.42 | 36.69 | t225 | 0.18 | 56.86 |
| t126 | -0.24 | 23.24 | t226 | 2.81 | 33.03 |
| t127 | -0.1 | 50.13 | t227 | 0 | -105.88 |
| t128 | -2.88 | 21.89 | t228 | -0.19 | 13.16 |
| t129 | 1.16 | 0 | t229 | -2.25 | -30.28 |
| t130 | -4.68 | -5.6 | t230 | 2.78 | -25.98 |
| t131 | -8.75 | 21.22 | t231 | -1.2 | 8.67 |
| t132 | -0.48 | 8.26 | t232 | 0.57 | 42.83 |
| t133 | 1.25 | 54.94 | t233 | 17.78 | -47.29 |
| t134 | 0.99 | 40.39 | t234 | 3.07 | 47.25 |
| t135 | -0.12 | 56.59 | t235 | 0.21 | 11.42 |
| t136 | 0.66 | 10.33 | t236 | 0 | -57 |
| t137 | -3.4 | 48.04 | t237 | 0.85 | 50.81 |
| t138 | -1.96 | 9.17 | t238 | -7.22 | 47.2 |
| t139 | 0 | 33.17 | t239 | 2.36 | 47.16 |
| t140 | 0 | 9.84 | t240 | 1.01 | -13.27 |
| t141 | -0.55 | 27.73 | t241 | 1.18 | -123.21 |
| t142 | -1.1 | 1.87 | t242 | 0 | -2 |
| t143 | 0.48 | -57.63 | t243 | 16.47 | -94.38 |
| t144 | -3.28 | 40.15 | t244 | 0.3 | 0 |
| t145 | 3.92 | 7.52 | t245 | 2.75 | -1.55 |
| t146 | 0.13 | 40.52 | t246 | 4.84 | 8.78 |
| t147 | 1.48 | 62.8 | t247 | -0.67 | 31.73 |
| t148 | 0.64 | 1.68 | t248 | 3.82 | 24.57 |
| t149 | -0.93 | 42.22 | t249 | -1.86 | 42.86 |
| t150 | -1.03 | 27.26 | t250 | -3.5 | 21.94 |
| Average | -0.66 | 21.75 | Average | 1.83 | 2.1 |

Table 11. Comparison between RCRatio and CSCRoutes for new instances

| **t1** | | | **t2** | | |
|---|---|---|---|---|---|
| Name | Profit Gap(%) | Time Gap(%) | Name | Profit Gap(%) | Time Gap(%) |
| t101 | -3.2 | 49.56 | t201 | -4.43 | 40.98 |
| t102 | -1.6 | 33.24 | t202 | 0 | 17.97 |
| t103 | -2.94 | 33.75 | t203 | -0.78 | 27.85 |
| t104 | -2.76 | 38.87 | t204 | 0 | 13.92 |
| t105 | 0.79 | 20.44 | t205 | -4.24 | 10.22 |
| t106 | -0.17 | 11.79 | t206 | -2.68 | -3.96 |
| t107 | -1.52 | 29.86 | t207 | 7.95 | 19.71 |
| t108 | 0.67 | 16.35 | t208 | 0 | 3.16 |
| t109 | 0.39 | 36.94 | t209 | -1.61 | 3.18 |
| t110 | -0.27 | 30.45 | t210 | -0.16 | 15.13 |
| t111 | -0.27 | 29.32 | t211 | 2.07 | 20.06 |
| t112 | 0.97 | 10.63 | t212 | -0.04 | 5.24 |
| t113 | -0.08 | 26.44 | t213 | -2.57 | 8.89 |
| t114 | -4.24 | 47.02 | t214 | -3.5 | 27.4 |
| t115 | -0.44 | 35.12 | t215 | -3.04 | 13.75 |
| t116 | -1.82 | 29.71 | t216 | -3.61 | 13.24 |
| t117 | 0.74 | 32.4 | t217 | -0.43 | -9.27 |
| t118 | 0.62 | 24.87 | t218 | 0 | 7.8 |
| t119 | -1.31 | 31.44 | t219 | -3.09 | -0.68 |
| t120 | -0.94 | 46.38 | t220 | -2.76 | 15.3 |
| t121 | -1.79 | 34.63 | t221 | -6.97 | 8.39 |
| t122 | -0.76 | 51.77 | t222 | -2.29 | 5.41 |
| t123 | 0.39 | 34.38 | t223 | 6.41 | 15.17 |
| t124 | 0.08 | 48.34 | t224 | -0.74 | 25.02 |
| t125 | -3.42 | 33.04 | t225 | -1.73 | 27.22 |
| t126 | -0.87 | 40.22 | t226 | 1.21 | 6.87 |
| t127 | -0.74 | 30.04 | t227 | 0 | -0.14 |
| t128 | -1.55 | 25.05 | t228 | -1.11 | 9.95 |
| t129 | -0.91 | 44.46 | t229 | -2.25 | 13.23 |
| t130 | -5.7 | 29.45 | t230 | -0.34 | -2.15 |
| t131 | -11.75 | 47.69 | t231 | -0.73 | 13.32 |
| t132 | -0.43 | 32.55 | t232 | -3.74 | 14.34 |
| t133 | -0.1 | 40.86 | t233 | 1.44 | 7.77 |
| t134 | -0.13 | 17.18 | t234 | -1.76 | 19.12 |
| t135 | 1.21 | 31.99 | t235 | -2.77 | 27.21 |
| t136 | 0.79 | 39.96 | t236 | -0.23 | 10.9 |
| t137 | -1.31 | 47.62 | t237 | -0.17 | 29.71 |
| t138 | -2.24 | 20.9 | t238 | -7.05 | 46.96 |
| t139 | -4.72 | 41.94 | t239 | 2.16 | -7.94 |
| t140 | -2.11 | 28.75 | t240 | -5.3 | 13.41 |
| t141 | -2.65 | 33.26 | t241 | 0 | 7.27 |
| t142 | -0.41 | 19.33 | t242 | 0 | 17.34 |
| t143 | -0.62 | 19.16 | t243 | -0.7 | 15.69 |
| t144 | -1.28 | 42.18 | t244 | -1.95 | 27.51 |
| t145 | 0.98 | 50.19 | t245 | 0 | 2.57 |
| t146 | -0.62 | 39.97 | t246 | 3.25 | 19.03 |
| t147 | 1.13 | 49.64 | t247 | -1.73 | 4.03 |
| t148 | 0.17 | 47.26 | t248 | -0.86 | 14.02 |
| t149 | -0.95 | 23.52 | t249 | -2.71 | 7.06 |
| t150 | -0.12 | 48.82 | t250 | -3.88 | 4.12 |
| Average | -1.16 | 34.18 | Average | -1.15 | 13.05 |

Table 12. Results for Solomon instances for 1 tour

| | | **ILS** | | **RCRatio** | | **CSCRoutes** | |
|---|---|---|---|---|---|---|---|
| Name | Clusters | Profit | CPU(ms) | Profit | CPU(ms) | Profit | CPU(ms) |
| c101 | 10 | 320 | 120 | 318 | 122.6 | 300 | 105 |
| c102 | 10 | 360 | 109 | 360 | 171.2 | 360 | 123 |
| c103 | 10 | 390 | 115 | 392 | 253 | 380 | 176 |
| c104 | 10 | 400 | 145 | 420 | 298 | 410 | 241 |
| c105 | 10 | 340 | 111 | 340 | 146 | 330 | 109 |
| c106 | 10 | 340 | 112 | 340 | 162.8 | 330 | 137 |
| c107 | 10 | 360 | 116 | 368 | 187.2 | 360 | 124 |
| c108 | 10 | 370 | 137 | 370 | 185.2 | 360 | 144 |
| c109 | 10 | 380 | 125 | 380 | 223.6 | 380 | 162 |
| c201 | 10 | 840 | 636 | 870 | 486.6 | 840 | 176 |
| c202 | 10 | 910 | 910 | 914 | 594 | 890 | 187 |
| c203 | 10 | 940 | 845 | 948 | 858.6 | 910 | 200 |
| c204 | 10 | 950 | 640 | 966 | 1134 | 970 | 250 |
| c205 | 10 | 900 | 470 | 900 | 578.8 | 890 | 214 |
| c206 | 10 | 910 | 485 | 920 | 665.8 | 910 | 229 |
| c207 | 10 | 910 | 722 | 926 | 741 | 920 | 239 |
| c208 | 10 | 930 | 534 | 940 | 800 | 920 | 232 |
| r101 | 10 | 182 | 70 | 185.8 | 85.8 | 180 | 84 |
| r102 | 10 | 286 | 130 | 286 | 167.2 | 282 | 122 |
| r103 | 10 | 286 | 116 | 292.4 | 219.2 | 289 | 147 |
| r104 | 10 | 297 | 135 | 303 | 244.2 | 303 | 168 |
| r105 | 10 | 247 | 153 | 247 | 130 | 238 | 93 |
| r106 | 10 | 293 | 124 | 293 | 176.6 | 279 | 133 |
| r107 | 10 | 288 | 115 | 295.8 | 216.4 | 289 | 136 |
| r108 | 10 | 297 | 194 | 308 | 234.2 | 303 | 166 |
| r109 | 10 | 276 | 143 | 277 | 164 | 259 | 108 |
| r110 | 10 | 281 | 165 | 281 | 178.4 | 281 | 126 |
| r111 | 10 | 295 | 148 | 295.4 | 207.8 | 297 | 139 |
| r112 | 10 | 295 | 130 | 296.6 | 231.6 | 285 | 149 |
| r201 | 10 | 788 | 829 | 783.4 | 659 | 476 | 132 |
| r202 | 10 | 880 | 1136 | 890.6 | 970.8 | 738 | 194 |
| r203 | 10 | 980 | 2081 | 980.6 | 1336.6 | 914 | 272 |
| r204 | 10 | 1073 | 1064 | 1062.8 | 1808.8 | 1048 | 344 |
| r205 | 10 | 931 | 1697 | 916.6 | 1044.8 | 644 | 166 |
| r206 | 10 | 996 | 824 | 990.6 | 1251.6 | 849 | 239 |
| r207 | 10 | 1038 | 926 | 1032 | 1554.6 | 917 | 300 |
| r208 | 10 | 1069 | 1791 | 1075 | 1947.6 | 1061 | 352 |
| r209 | 10 | 926 | 825 | 919.4 | 1286 | 709 | 238 |
| r210 | 10 | 958 | 1354 | 956.4 | 1265.4 | 773 | 233 |
| r211 | 10 | 1023 | 854 | 1013.8 | 1583.8 | 868 | 353 |
| rc101 | 10 | 219 | 108 | 219 | 106.2 | 216 | 91 |
| rc102 | 10 | 259 | 135 | 266 | 140.2 | 266 | 107 |
| rc103 | 10 | 265 | 114 | 264 | 150.6 | 266 | 112 |
| rc104 | 10 | 297 | 94 | 301 | 166.2 | 301 | 128 |
| rc105 | 10 | 221 | 132 | 242.8 | 130 | 241 | 103 |
| rc106 | 10 | 239 | 142 | 249.4 | 134.8 | 250 | 102 |
| rc107 | 10 | 274 | 145 | 276.4 | 154.4 | 261 | 115 |
| rc108 | 10 | 288 | 120 | 298 | 165.2 | 274 | 112 |
| rc201 | 10 | 780 | 685 | 778.8 | 538.4 | 632 | 148 |
| rc202 | 10 | 882 | 885 | 899.4 | 695.6 | 864 | 204 |
| rc203 | 10 | 960 | 885 | 956.2 | 987.4 | 901 | 221 |
| rc204 | 10 | 1117 | 997 | 1103.2 | 1479.2 | 1121 | 339 |
| rc205 | 10 | 840 | 656 | 841.6 | 629.2 | 726 | 165 |
| rc206 | 10 | 860 | 631 | 862.4 | 710.4 | 749 | 195 |
| rc207 | 10 | 926 | 1042 | 919.8 | 919.6 | 827 | 264 |
| rc208 | 10 | 1037 | 1434 | 1022.6 | 1175.8 | 971 | 315 |

Table 13. Results for Cordeau et al. instances for 1 tour

| | | **ILS** | | **RCRatio** | | **CSCRoutes** | |
|---|---|---|---|---|---|---|---|
| Name | Clusters | Profit | CPU(ms) | Profit | CPU(ms) | Profit | CPU(ms) |
| pr01 | 4 | 304 | 122 | 306.6 | 122.8 | 247 | 56 |
| pr02 | 9 | 385 | 431 | 392.4 | 299.4 | 374 | 115 |
| pr03 | 14 | 384 | 396 | 389.4 | 404.6 | 349 | 208 |
| pr04 | 19 | 447 | 1013 | 467.2 | 671.6 | 425 | 273 |
| pr05 | 24 | 576 | 1431 | 565.4 | 1136.4 | 443 | 385 |
| pr06 | 28 | 538 | 1403 | 543.6 | 1178.6 | 472 | 537 |
| pr07 | 7 | 291 | 135 | 295.2 | 172 | 291 | 87 |
| pr08 | 14 | 463 | 715 | 454.2 | 463 | 405 | 184 |
| pr09 | 21 | 461 | 798 | 466.6 | 708.6 | 437 | 313 |
| pr10 | 28 | 539 | 1443 | 541.2 | 1256.8 | 492 | 554 |
| pr11 | 4 | 330 | 158 | 337.2 | 157.6 | 324 | 80 |
| pr12 | 9 | 431 | 411 | 432.6 | 383.8 | 408 | 165 |
| pr13 | 14 | 450 | 550 | 448.4 | 594.8 | 433 | 255 |
| pr14 | 19 | 482 | 868 | 515.2 | 952.2 | 453 | 369 |
| pr15 | 24 | 638 | 1798 | 638.2 | 1647.4 | 619 | 498 |
| pr16 | 28 | 559 | 4891 | 591.2 | 1694.2 | 531 | 664 |
| pr17 | 7 | 346 | 285 | 359.6 | 243.4 | 331 | 112 |
| pr18 | 14 | 479 | 954 | 492.8 | 615 | 383 | 219 |
| pr19 | 21 | 499 | 1322 | 516.8 | 1031.8 | 487 | 436 |
| pr20 | 28 | 570 | 2156 | 600.2 | 1801 | 522 | 643 |

Table 14. Results for Solomon instances for 2 tours

| | | **ILS** | | **RCRatio** | | **CSCRoutes** | |
|---|---|---|---|---|---|---|---|
| Name | Clusters | Profit | CPU(ms) | Profit | CPU(ms) | Profit | CPU(ms) |
| c101 | 10 | 590 | 320 | 590 | 259.6 | 550 | 165 |
| c102 | 10 | 650 | 474 | 650 | 358 | 650 | 247 |
| c103 | 10 | 700 | 350 | 700 | 501.8 | 700 | 397 |
| c104 | 10 | 750 | 623 | 752 | 646.8 | 720 | 427 |
| c105 | 10 | 640 | 335 | 640 | 291 | 600 | 224 |
| c106 | 10 | 620 | 323 | 620 | 303.4 | 600 | 295 |
| c107 | 10 | 670 | 352 | 670 | 356.6 | 620 | 208 |
| c108 | 10 | 670 | 344 | 680 | 390 | 640 | 238 |
| c109 | 10 | 710 | 436 | 720 | 495.6 | 690 | 377 |
| c201 | 10 | 1400 | 1291 | 1428 | 1092.2 | 1420 | 490 |
| c202 | 10 | 1430 | 1027 | 1442 | 1323.6 | 1420 | 392 |
| c203 | 10 | 1430 | 989 | 1444 | 1593.6 | 1440 | 541 |
| c204 | 10 | 1460 | 1255 | 1450 | 2066.8 | 1460 | 621 |
| c205 | 10 | 1450 | 2013 | 1446 | 1284.2 | 1440 | 472 |
| c206 | 10 | 1440 | 1076 | 1462 | 1585.6 | 1460 | 502 |
| c207 | 10 | 1450 | 1263 | 1466 | 1484.8 | 1460 | 492 |
| c208 | 10 | 1460 | 1361 | 1470 | 1895.6 | 1470 | 638 |
| r101 | 10 | 330 | 239 | 343.4 | 174.4 | 325 | 127 |
| r102 | 10 | 508 | 339 | 506 | 328.2 | 501 | 192 |
| r103 | 10 | 513 | 335 | 513 | 376.8 | 504 | 305 |
| r104 | 10 | 539 | 403 | 543.4 | 515.2 | 529 | 338 |
| r105 | 10 | 430 | 290 | 453 | 265.8 | 422 | 191 |
| r106 | 10 | 529 | 477 | 529 | 311.4 | 509 | 241 |
| r107 | 10 | 529 | 388 | 533.4 | 456.4 | 523 | 295 |
| r108 | 10 | 549 | 399 | 558 | 589.8 | 545 | 331 |
| r109 | 10 | 498 | 437 | 501.6 | 340 | 483 | 188 |
| r110 | 10 | 515 | 533 | 509.8 | 403.6 | 506 | 324 |
| r111 | 10 | 535 | 703 | 535.6 | 405 | 538 | 239 |
| r112 | 10 | 515 | 603 | 535 | 488.4 | 531 | 445 |
| r201 | 10 | 1231 | 982 | 1215.4 | 1339.2 | 919 | 314 |
| r202 | 10 | 1270 | 1121 | 1293.4 | 1297.2 | 1115 | 416 |
| r203 | 10 | 1377 | 852 | 1373.4 | 1379.6 | 1206 | 674 |
| r204 | 10 | 1440 | 665 | 1430.2 | 1442.8 | 1364 | 471 |
| r205 | 10 | 1338 | 1067 | 1327.2 | 1745 | 1062 | 401 |
| r206 | 10 | 1401 | 781 | 1400.6 | 1556.4 | 1246 | 510 |
| r207 | 10 | 1428 | 692 | 1434 | 1633.4 | 1387 | 572 |
| r208 | 10 | 1458 | 536 | 1457.2 | 1524.6 | 1418 | 445 |
| r209 | 10 | 1345 | 1092 | 1362.4 | 1686 | 1216 | 435 |
| r210 | 10 | 1365 | 1072 | 1372.2 | 1536.4 | 1241 | 494 |
| r211 | 10 | 1422 | 776 | 1424.8 | 1702.2 | 1355 | 558 |
| rc101 | 10 | 427 | 587 | 423.8 | 223.6 | 419 | 162 |
| rc102 | 10 | 494 | 414 | 497.2 | 261.6 | 489 | 237 |
| rc103 | 10 | 519 | 318 | 522.4 | 349.4 | 518 | 260 |
| rc104 | 10 | 565 | 556 | 567 | 477.2 | 559 | 299 |
| rc105 | 10 | 459 | 380 | 474.6 | 244.2 | 430 | 195 |
| rc106 | 10 | 458 | 455 | 471.6 | 277 | 458 | 233 |
| rc107 | 10 | 515 | 566 | 520.8 | 353 | 494 | 310 |
| rc108 | 10 | 546 | 455 | 547.2 | 405 | 535 | 236 |
| rc201 | 10 | 1305 | 1011 | 1325.6 | 1168.6 | 1030 | 327 |
| rc202 | 10 | 1461 | 1229 | 1449 | 1365.4 | 1157 | 355 |
| rc203 | 10 | 1573 | 988 | 1537.2 | 1532 | 1440 | 423 |
| rc204 | 10 | 1656 | 768 | 1654 | 1681.8 | 1628 | 408 |
| rc205 | 10 | 1381 | 1332 | 1384.4 | 1434.6 | 1129 | 325 |
| rc206 | 10 | 1495 | 963 | 1460.4 | 1591 | 1355 | 471 |
| rc207 | 10 | 1531 | 1024 | 1509 | 1736 | 1423 | 485 |
| rc208 | 10 | 1606 | 1628 | 1616.6 | 1923.8 | 1599 | 698 |

Table 15. Results for Cordeau et al. instances for 2 tours

| | | **ILS** | | **RCRatio** | | **CSCRoutes** | |
|---|---|---|---|---|---|---|---|
| Name | Clusters | Profit | CPU(ms) | Profit | CPU(ms) | Profit | CPU(ms) |
| pr01 | 4 | 471 | 148 | 486.2 | 230 | 437 | 152 |
| pr02 | 9 | 660 | 800 | 687.6 | 769.8 | 631 | 271 |
| pr03 | 14 | 714 | 1083 | 711.4 | 874.8 | 646 | 390 |
| pr04 | 19 | 863 | 2552 | 866 | 1541.8 | 846 | 772 |
| pr05 | 24 | 1011 | 4043 | 1033.6 | 3208.6 | 858 | 1142 |
| pr06 | 28 | 997 | 4910 | 982.2 | 2852.2 | 854 | 1391 |
| pr07 | 7 | 552 | 483 | 554 | 400.2 | 557 | 171 |
| pr08 | 14 | 796 | 1355 | 795.4 | 1053 | 696 | 408 |
| pr09 | 21 | 867 | 3035 | 854.8 | 1803.8 | 780 | 877 |
| pr10 | 28 | 1004 | 5257 | 1049 | 3181.4 | 932 | 1639 |
| pr11 | 4 | 542 | 181 | 537 | 299.8 | 525 | 178 |
| pr12 | 9 | 727 | 746 | 725.4 | 837 | 696 | 327 |
| pr13 | 14 | 757 | 1683 | 780.6 | 1350.4 | 761 | 574 |
| pr14 | 19 | 925 | 2845 | 928.4 | 2168.6 | 927 | 1048 |
| pr15 | 24 | 1126 | 7239 | 1104.2 | 4102.6 | 972 | 1825 |
| pr16 | 28 | 1110 | 5949 | 1098.8 | 3954.8 | 916 | 1849 |
| pr17 | 7 | 624 | 1091 | 625 | 565 | 603 | 232 |
| pr18 | 14 | 877 | 1369 | 886 | 1535.4 | 746 | 520 |
| pr19 | 21 | 955 | 3792 | 942.2 | 2790.8 | 873 | 1183 |
| pr20 | 28 | 1056 | 5360 | 1139.2 | 4015.2 | 936 | 1842 |

Table 16. Results for Solomon instances for 3 tours

| | | **ILS** | | **RCRatio** | | **CSCRoutes** | |
|---|---|---|---|---|---|---|---|
| Name | Clusters | Profit | CPU(ms) | Profit | CPU(ms) | Profit | CPU(ms) |
| c101 | 10 | 790 | 814 | 802 | 466 | 770 | 300 |
| c102 | 10 | 890 | 766 | 896 | 611.2 | 890 | 466 |
| c103 | 10 | 960 | 893 | 962 | 821.4 | 930 | 507 |
| c104 | 10 | 1010 | 1128 | 1006 | 1169.8 | 1000 | 836 |
| c105 | 10 | 840 | 715 | 852 | 572.4 | 820 | 347 |
| c106 | 10 | 840 | 871 | 850 | 607 | 830 | 426 |
| c107 | 10 | 900 | 1307 | 904 | 652.2 | 870 | 415 |
| c108 | 10 | 900 | 2177 | 904 | 701.8 | 890 | 700 |
| c109 | 10 | 950 | 782 | 954 | 911.2 | 960 | 718 |
| c201 | 10 | 1750 | 1368 | 1746 | 1132.8 | 1770 | 486 |
| c202 | 10 | 1750 | 1105 | 1764 | 1069.8 | 1770 | 475 |
| c203 | 10 | 1760 | 680 | 1760 | 1200 | 1750 | 407 |
| c204 | 10 | 1780 | 610 | 1772 | 1282.2 | 1780 | 449 |
| c205 | 10 | 1770 | 737 | 1772 | 1220.8 | 1770 | 593 |
| c206 | 10 | 1770 | 629 | 1788 | 1323.4 | 1790 | 493 |
| c207 | 10 | 1810 | 1103 | 1788 | 1223 | 1800 | 561 |
| c208 | 10 | 1810 | 1030 | 1796 | 1342.2 | 1800 | 493 |
| r101 | 10 | 481 | 366 | 479.8 | 302.6 | 453 | 161 |
| r102 | 10 | 685 | 616 | 682.6 | 604.4 | 666 | 359 |
| r103 | 10 | 720 | 1032 | 723 | 755.6 | 699 | 413 |
| r104 | 10 | 765 | 1316 | 766.2 | 973.2 | 747 | 605 |
| r105 | 10 | 609 | 961 | 609.6 | 469.4 | 582 | 319 |
| r106 | 10 | 719 | 692 | 717.6 | 648.6 | 707 | 558 |
| r107 | 10 | 747 | 941 | 744.4 | 774 | 739 | 525 |
| r108 | 10 | 790 | 1889 | 785.8 | 1044 | 767 | 638 |
| r109 | 10 | 699 | 1298 | 701.6 | 656.8 | 667 | 508 |
| r110 | 10 | 711 | 811 | 720 | 722 | 694 | 696 |
| r111 | 10 | 764 | 1102 | 761.2 | 866.6 | 759 | 563 |
| r112 | 10 | 758 | 1190 | 764.8 | 973.8 | 756 | 672 |
| r201 | 10 | 1408 | 918 | 1407.4 | 1211.6 | 1163 | 642 |
| r202 | 10 | 1443 | 730 | 1447.8 | 1187.2 | 1370 | 554 |
| r203 | 10 | 1458 | 463 | 1458 | 959.4 | 1393 | 457 |
| r204 | 10 | 1458 | 366 | 1458 | 966 | 1458 | 408 |
| r205 | 10 | 1458 | 480 | 1458 | 1076.6 | 1430 | 478 |
| r206 | 10 | 1458 | 386 | 1458 | 936.8 | 1430 | 476 |
| r207 | 10 | 1458 | 373 | 1458 | 948.2 | 1456 | 616 |
| r208 | 10 | 1458 | 363 | 1458 | 992.4 | 1458 | 341 |
| r209 | 10 | 1458 | 417 | 1458 | 993.4 | 1423 | 439 |
| r210 | 10 | 1458 | 402 | 1458 | 948.6 | 1411 | 582 |
| r211 | 10 | 1458 | 372 | 1458 | 975 | 1458 | 387 |
| rc101 | 10 | 604 | 690 | 614.8 | 448 | 598 | 264 |
| rc102 | 10 | 698 | 1627 | 700.2 | 614 | 691 | 361 |
| rc103 | 10 | 747 | 884 | 752.4 | 605.2 | 757 | 442 |
| rc104 | 10 | 822 | 797 | 816.2 | 809.4 | 817 | 579 |
| rc105 | 10 | 654 | 564 | 669.2 | 473.4 | 656 | 342 |
| rc106 | 10 | 678 | 632 | 689.8 | 528.8 | 658 | 399 |
| rc107 | 10 | 745 | 805 | 754.4 | 580.6 | 733 | 537 |
| rc108 | 10 | 757 | 654 | 780.4 | 776.2 | 754 | 552 |
| rc201 | 10 | 1625 | 867 | 1629.4 | 1264.8 | 1342 | 592 |
| rc202 | 10 | 1686 | 779 | 1689.8 | 1261 | 1462 | 521 |
| rc203 | 10 | 1724 | 520 | 1724 | 1059.4 | 1618 | 485 |
| rc204 | 10 | 1724 | 415 | 1724 | 987.2 | 1710 | 414 |
| rc205 | 10 | 1659 | 706 | 1657.8 | 1389.4 | 1468 | 552 |
| rc206 | 10 | 1708 | 661 | 1716.8 | 1351.2 | 1589 | 475 |
| rc207 | 10 | 1713 | 731 | 1719.8 | 1319.6 | 1624 | 538 |
| rc208 | 10 | 1724 | 419 | 1724 | 1045.2 | 1724 | 431 |

Table 17. Results for Cordeau et al. instances for 3 tours

| | | **ILS** | | **RCRatio** | | **CSCRoutes** | |
|---|---|---|---|---|---|---|---|
| Name | Clusters | Profit | CPU(ms) | Profit | CPU(ms) | Profit | CPU(ms) |
| pr01 | 4 | 598 | 97 | 596.8 | 197.4 | 591 | 85 |
| pr02 | 9 | 899 | 925 | 895.4 | 1200 | 838 | 551 |
| pr03 | 14 | 946 | 2304 | 968.4 | 1756.4 | 880 | 785 |
| pr04 | 19 | 1195 | 7520 | 1212.8 | 3117.2 | 1091 | 1633 |
| pr05 | 24 | 1356 | 7142 | 1391 | 5340.2 | 1260 | 2471 |
| pr06 | 28 | 1376 | 14552 | 1395 | 5397.6 | 1195 | 2795 |
| pr07 | 7 | 713 | 998 | 721.2 | 681.8 | 697 | 322 |
| pr08 | 14 | 1082 | 2368 | 1070 | 2172.2 | 978 | 946 |
| pr09 | 21 | 1144 | 7294 | 1178.4 | 3547.2 | 1152 | 1752 |
| pr10 | 28 | 1473 | 13834 | 1434.2 | 6599.8 | 1270 | 3505 |
| pr11 | 4 | 632 | 88 | 632.8 | 232 | 626 | 102 |
| pr12 | 9 | 902 | 882 | 926.8 | 1482.6 | 892 | 677 |
| pr13 | 14 | 1046 | 2500 | 1076 | 2465.2 | 983 | 1146 |
| pr14 | 19 | 1197 | 5378 | 1255.2 | 3987.4 | 1185 | 1987 |
| pr15 | 24 | 1488 | 8033 | 1522.2 | 6888.6 | 1400 | 3852 |
| pr16 | 28 | 1478 | 10269 | 1507 | 7796.8 | 1463 | 3885 |
| pr17 | 7 | 808 | 405 | 795.6 | 773 | 801 | 328 |
| pr18 | 14 | 1165 | 2431 | 1171.6 | 2570.6 | 1064 | 1323 |
| pr19 | 21 | 1238 | 5886 | 1322.6 | 5140.8 | 1202 | 2463 |
| pr20 | 28 | 1514 | 17557 | 1531.6 | 8357.2 | 1470 | 3967 |

Table 18. Results for Solomon instances for 4 tours

| | | **ILS** | | **RCRatio** | | **CSCRoutes** | |
|---|---|---|---|---|---|---|---|
| Name | Clusters | Profit | CPU(ms) | Profit | CPU(ms) | Profit | CPU(ms) |
| c101 | 10 | 1000 | 972 | 994 | 649.4 | 990 | 531 |
| c102 | 10 | 1090 | 1405 | 1122 | 947.4 | 1090 | 663 |
| c103 | 10 | 1150 | 947 | 1162 | 1366.4 | 1140 | 979 |
| c104 | 10 | 1220 | 2411 | 1218 | 1752.2 | 1210 | 1024 |
| c105 | 10 | 1030 | 1815 | 1048 | 914 | 1020 | 541 |
| c106 | 10 | 1040 | 1100 | 1046 | 909.6 | 1020 | 792 |
| c107 | 10 | 1100 | 921 | 1104 | 1044.6 | 1060 | 714 |
| c108 | 10 | 1100 | 1525 | 1106 | 1024.2 | 1080 | 768 |
| c109 | 10 | 1180 | 1546 | 1164 | 1431.6 | 1160 | 1309 |
| c201 | 10 | 1810 | 364 | 1810 | 762 | 1810 | 376 |
| c202 | 10 | 1810 | 349 | 1810 | 784.6 | 1810 | 490 |
| c203 | 10 | 1810 | 352 | 1810 | 823.2 | 1810 | 409 |
| c204 | 10 | 1810 | 341 | 1810 | 863.2 | 1810 | 450 |
| c205 | 10 | 1810 | 339 | 1810 | 763.8 | 1810 | 383 |
| c206 | 10 | 1810 | 300 | 1810 | 782.2 | 1810 | 393 |
| c207 | 10 | 1810 | 339 | 1810 | 783 | 1810 | 380 |
| c208 | 10 | 1810 | 335 | 1810 | 790.6 | 1810 | 372 |
| r101 | 10 | 601 | 1013 | 607.8 | 609.4 | 576 | 318 |
| r102 | 10 | 807 | 1175 | 815.8 | 889 | 808 | 586 |
| r103 | 10 | 878 | 1017 | 892.6 | 1107.6 | 863 | 846 |
| r104 | 10 | 941 | 1474 | 949.6 | 1438.6 | 938 | 786 |
| r105 | 10 | 735 | 788 | 755 | 764.2 | 725 | 614 |
| r106 | 10 | 870 | 927 | 880.8 | 1019.2 | 845 | 751 |
| r107 | 10 | 927 | 1187 | 916.2 | 1273.2 | 900 | 918 |
| r108 | 10 | 982 | 1255 | 958.6 | 1464.6 | 947 | 1004 |
| r109 | 10 | 866 | 1346 | 871 | 1121.2 | 854 | 813 |
| r110 | 10 | 870 | 1228 | 880.8 | 1299.2 | 876 | 747 |
| r111 | 10 | 935 | 2882 | 922.6 | 1059.2 | 905 | 1056 |
| r112 | 10 | 939 | 2629 | 942 | 1413.2 | 933 | 1294 |
| r201 | 10 | 1458 | 474 | 1458 | 856.6 | 1313 | 451 |
| r202 | 10 | 1458 | 401 | 1458 | 848 | 1419 | 619 |
| r203 | 10 | 1458 | 332 | 1458 | 746.6 | 1447 | 493 |
| r204 | 10 | 1458 | 229 | 1458 | 621.6 | 1458 | 333 |
| r205 | 10 | 1458 | 305 | 1458 | 786.6 | 1458 | 455 |
| r206 | 10 | 1458 | 256 | 1458 | 691.2 | 1458 | 432 |
| r207 | 10 | 1458 | 218 | 1458 | 610.8 | 1458 | 365 |
| r208 | 10 | 1458 | 157 | 1458 | 588.6 | 1458 | 227 |
| r209 | 10 | 1458 | 251 | 1458 | 832.4 | 1458 | 391 |
| r210 | 10 | 1458 | 317 | 1458 | 749.6 | 1458 | 479 |
| r211 | 10 | 1458 | 220 | 1458 | 597.6 | 1458 | 315 |
| rc101 | 10 | 794 | 1282 | 795.8 | 691.4 | 788 | 482 |
| rc102 | 10 | 881 | 1617 | 887.6 | 747.2 | 875 | 571 |
| rc103 | 10 | 947 | 1236 | 952 | 956.6 | 929 | 716 |
| rc104 | 10 | 1019 | 1869 | 1035 | 1237.8 | 998 | 815 |
| rc105 | 10 | 841 | 854 | 852.2 | 772.6 | 820 | 543 |
| rc106 | 10 | 874 | 1030 | 881.4 | 761.2 | 881 | 627 |
| rc107 | 10 | 951 | 1338 | 957 | 1091 | 940 | 639 |
| rc108 | 10 | 998 | 2478 | 993.4 | 1117.8 | 968 | 723 |
| rc201 | 10 | 1724 | 501 | 1724 | 1052 | 1558 | 817 |
| rc202 | 10 | 1724 | 401 | 1724 | 857.8 | 1613 | 609 |
| rc203 | 10 | 1724 | 329 | 1724 | 792.6 | 1709 | 421 |
| rc204 | 10 | 1724 | 233 | 1724 | 644 | 1724 | 363 |
| rc205 | 10 | 1724 | 428 | 1724 | 968 | 1632 | 527 |
| rc206 | 10 | 1724 | 348 | 1724 | 823.8 | 1724 | 534 |
| rc207 | 10 | 1724 | 359 | 1724 | 858.2 | 1714 | 600 |
| rc208 | 10 | 1724 | 349 | 1724 | 759.6 | 1724 | 349 |

Table 19. Results for Cordeau et al. instances for 4 tours

| | | **ILS** | | **RCRatio** | | **CSCRoutes** | |
|---|---|---|---|---|---|---|---|
| Name | Clusters | Profit | CPU(ms) | Profit | CPU(ms) | Profit | CPU(ms) |
| pr01 | 4 | 644 | 79 | 653 | 228.8 | 649 | 128 |
| pr02 | 9 | 1014 | 927 | 1019.4 | 1377.8 | 953 | 696 |
| pr03 | 14 | 1162 | 3663 | 1146.4 | 2768 | 1086 | 1332 |
| pr04 | 19 | 1452 | 7982 | 1472.6 | 4832.4 | 1389 | 2867 |
| pr05 | 24 | 1665 | 22171 | 1700.4 | 8112.6 | 1620 | 3714 |
| pr06 | 28 | 1696 | 12486 | 1731 | 8953 | 1554 | 4298 |
| pr07 | 7 | 840 | 473 | 842.2 | 720.2 | 781 | 308 |
| pr08 | 14 | 1267 | 2764 | 1284.8 | 2768.4 | 1194 | 1404 |
| pr09 | 21 | 1460 | 8082 | 1490.2 | 5582 | 1370 | 3396 |
| pr10 | 28 | 1782 | 14071 | 1794.2 | 9552.2 | 1587 | 5085 |
| pr11 | 4 | 654 | 67 | 655.2 | 207.4 | 654 | 91 |
| pr12 | 9 | 1041 | 891 | 1055.8 | 1715.8 | 1023 | 714 |
| pr13 | 14 | 1263 | 4168 | 1283.4 | 3279.6 | 1177 | 2127 |
| pr14 | 19 | 1528 | 9987 | 1543 | 6099.4 | 1429 | 3302 |
| pr15 | 24 | 1818 | 10679 | 1859 | 11172 | 1749 | 5425 |
| pr16 | 28 | 1889 | 19914 | 1854.6 | 12503.4 | 1613 | 5983 |
| pr17 | 7 | 889 | 341 | 886.6 | 775.4 | 882 | 436 |
| pr18 | 14 | 1352 | 3549 | 1392.8 | 3549.2 | 1340 | 1844 |
| pr19 | 21 | 1560 | 8706 | 1601.6 | 7327.6 | 1493 | 4107 |
| pr20 | 28 | 1846 | 16212 | 1919.4 | 12377.2 | 1812 | 7597 |

Table 20. Results for t1* instances

| | | ILS | | RCRatio | | CSCRoutes | |
|---|---|---|---|---|---|---|---|
| Name | Clusters | Profit | CPU(ms) | Profit | CPU(ms) | Profit | CPU(ms) |
| t101 | 10 | 387 | 309 | 387.4 | 364.8 | 375 | 184 |
| t102 | 10 | 772 | 683 | 763.2 | 810.4 | 751 | 541 |
| t103 | 16 | 786 | 1368 | 810.8 | 1461.2 | 787 | 968 |
| t104 | 12 | 737 | 964 | 746.6 | 991.4 | 726 | 606 |
| t105 | 15 | 433 | 383 | 429.6 | 551.8 | 433 | 439 |
| t106 | 18 | 1167 | 2562 | 1160 | 2972.4 | 1158 | 2622 |
| t107 | 13 | 787 | 1607 | 777.8 | 1087.8 | 766 | 763 |
| t108 | 15 | 711 | 1669 | 712.2 | 1315 | 717 | 1100 |
| t109 | 10 | 1114 | 1752 | 1085.8 | 1650.8 | 1090 | 1041 |
| t110 | 16 | 807 | 1747 | 810.2 | 1491 | 808 | 1037 |
| t111 | 19 | 821 | 2167 | 810.2 | 1737.4 | 808 | 1228 |
| t112 | 14 | 800 | 869 | 805.2 | 1285.6 | 813 | 1149 |
| t113 | 15 | 1091 | 3243 | 1057.8 | 2062.4 | 1057 | 1517 |
| t114 | 12 | 467 | 386 | 476.2 | 590.8 | 456 | 313 |
| t115 | 10 | 1059 | 1413 | 1048.6 | 1411.8 | 1044 | 916 |
| t116 | 18 | 840 | 1042 | 848.4 | 1680.2 | 833 | 1181 |
| t117 | 19 | 452 | 850 | 458.6 | 883.2 | 462 | 597 |
| t118 | 15 | 1140 | 1905 | 1133 | 2591.6 | 1140 | 1947 |
| t119 | 18 | 1163 | 2030 | 1173.4 | 3054.4 | 1158 | 2094 |
| t120 | 17 | 1023 | 3411 | 995.4 | 2475 | 986 | 1327 |
| t121 | 16 | 424 | 351 | 436.8 | 641 | 429 | 419 |
| t122 | 14 | 468 | 859 | 475.6 | 576.4 | 472 | 278 |
| t123 | 15 | 404 | 285 | 407.4 | 509 | 409 | 334 |
| t124 | 12 | 435 | 206 | 470.6 | 499.4 | 471 | 258 |
| t125 | 18 | 1176 | 2693 | 1163.8 | 2546.2 | 1124 | 1705 |
| t126 | 12 | 413 | 340 | 415.6 | 436.6 | 412 | 261 |
| t127 | 11 | 1025 | 1853 | 1031.6 | 1320.8 | 1024 | 924 |
| t128 | 14 | 1111 | 2325 | 1096 | 2422.8 | 1079 | 1816 |
| t129 | 13 | 432 | 305 | 441 | 549.2 | 437 | 305 |
| t130 | 18 | 812 | 1124 | 820.8 | 1682.6 | 774 | 1187 |
| t131 | 16 | 400 | 377 | 413.6 | 567.8 | 365 | 297 |
| t132 | 19 | 420 | 581 | 419.8 | 790.2 | 418 | 533 |
| t133 | 13 | 798 | 1620 | 808.8 | 1234.4 | 808 | 730 |
| t134 | 18 | 1212 | 4370 | 1225.6 | 3145.4 | 1224 | 2605 |
| t135 | 16 | 823 | 2778 | 812.2 | 1773.2 | 822 | 1206 |
| t136 | 10 | 756 | 523 | 755 | 781.2 | 761 | 469 |
| t137 | 17 | 1119 | 2623 | 1095.4 | 2602 | 1081 | 1363 |
| t138 | 17 | 1222 | 2781 | 1225.4 | 3193.4 | 1198 | 2526 |
| t139 | 15 | 1115 | 2641 | 1170.2 | 3040.2 | 1115 | 1765 |
| t140 | 10 | 993 | 894 | 1014.4 | 1131.2 | 993 | 806 |
| t141 | 10 | 724 | 768 | 739.6 | 831.6 | 720 | 555 |
| t142 | 18 | 1185 | 2949 | 1176.8 | 3587.6 | 1172 | 2894 |
| t143 | 15 | 413 | 295 | 417.6 | 575.2 | 415 | 465 |
| t144 | 17 | 763 | 1432 | 747.6 | 1482.2 | 738 | 857 |
| t145 | 10 | 357 | 226 | 367.4 | 419.6 | 371 | 209 |
| t146 | 13 | 767 | 1123 | 772.8 | 1112.8 | 768 | 668 |
| t147 | 13 | 1078 | 2468 | 1081.8 | 1822.8 | 1094 | 918 |
| t148 | 14 | 468 | 298 | 470.2 | 555.6 | 471 | 293 |
| t149 | 13 | 1072 | 2418 | 1072.2 | 1826.6 | 1062 | 1397 |
| t150 | 16 | 487 | 554 | 482.6 | 787.4 | 482 | 403 |

Table 21. Results for t2* instances

| | | **ILS** | | **RCRatio** | | **CSCRoutes** | |
|---|---|---|---|---|---|---|---|
| Name | Clusters | Profit | CPU(ms) | Profit | CPU(ms) | Profit | CPU(ms) |
| t201 | 12 | 183 | 88 | 185.2 | 179.6 | 177 | 106 |
| t202 | 14 | 193 | 118 | 193 | 153.6 | 193 | 126 |
| t203 | 10 | 174 | 76 | 179.4 | 159.4 | 178 | 115 |
| t204 | 14 | 171 | 181 | 171 | 169.6 | 171 | 146 |
| t205 | 13 | 447 | 1245 | 443.8 | 483.4 | 425 | 434 |
| t206 | 17 | 196 | 239 | 201.4 | 227 | 196 | 236 |
| t207 | 14 | 174 | 104 | 186.2 | 164.4 | 201 | 132 |
| t208 | 19 | 162 | 104 | 176 | 190 | 176 | 184 |
| t209 | 17 | 455 | 1169 | 460.4 | 736.4 | 453 | 713 |
| t210 | 20 | 481 | 1271 | 496.8 | 978 | 496 | 830 |
| t211 | 12 | 472 | 820 | 473.2 | 555.4 | 483 | 444 |
| t212 | 10 | 461 | 458 | 465.2 | 377.8 | 465 | 358 |
| t213 | 17 | 498 | 959 | 497.8 | 884.6 | 485 | 806 |
| t214 | 14 | 310 | 440 | 325.4 | 352.6 | 314 | 256 |
| t215 | 13 | 424 | 502 | 427 | 480 | 414 | 414 |
| t216 | 12 | 463 | 700 | 481.4 | 655.8 | 464 | 569 |
| t217 | 11 | 463 | 956 | 465 | 429.2 | 463 | 469 |
| t218 | 10 | 155 | 57 | 155 | 84.6 | 155 | 78 |
| t219 | 16 | 473 | 875 | 498.4 | 820.4 | 483 | 826 |
| t220 | 13 | 334 | 689 | 340.4 | 329.4 | 331 | 279 |
| t221 | 18 | 280 | 365 | 304.2 | 414.8 | 283 | 380 |
| t222 | 19 | 396 | 603 | 401.2 | 650.2 | 392 | 615 |
| t223 | 15 | 183 | 74 | 215.2 | 212.2 | 229 | 180 |
| t224 | 13 | 402 | 618 | 406 | 418.8 | 403 | 314 |
| t225 | 18 | 543 | 1850 | 553.6 | 1096.4 | 544 | 798 |
| t226 | 19 | 569 | 1320 | 578 | 949.2 | 585 | 884 |
| t227 | 13 | 159 | 68 | 159 | 139.8 | 159 | 140 |
| t228 | 16 | 537 | 950 | 542 | 916.2 | 536 | 825 |
| t229 | 19 | 178 | 142 | 178 | 213.2 | 174 | 185 |
| t230 | 11 | 288 | 204 | 297 | 251.6 | 296 | 257 |
| t231 | 11 | 498 | 496 | 495.6 | 522.6 | 492 | 453 |
| t232 | 17 | 522 | 1534 | 545.4 | 1023.8 | 525 | 877 |
| t233 | 15 | 180 | 129 | 209 | 206 | 212 | 190 |
| t234 | 15 | 488 | 1020 | 512 | 665.2 | 503 | 538 |
| t235 | 14 | 484 | 543 | 498.8 | 660.8 | 485 | 481 |
| t236 | 15 | 175 | 100 | 175.4 | 176.2 | 175 | 157 |
| t237 | 11 | 473 | 1114 | 477.8 | 779.6 | 477 | 548 |
| t238 | 12 | 526 | 856 | 525 | 852.2 | 488 | 452 |
| t239 | 19 | 508 | 1970 | 509 | 964.4 | 520 | 1041 |
| t240 | 12 | 297 | 211 | 316.8 | 276 | 300 | 239 |
| t241 | 14 | 170 | 56 | 172 | 134.8 | 172 | 125 |
| t242 | 10 | 180 | 100 | 180 | 123.4 | 180 | 102 |
| t243 | 15 | 170 | 89 | 199.4 | 205.2 | 198 | 173 |
| t244 | 10 | 331 | 204 | 338.6 | 281.4 | 332 | 204 |
| t245 | 14 | 291 | 194 | 299 | 202.2 | 299 | 197 |
| t246 | 15 | 455 | 638 | 462 | 718.8 | 477 | 582 |
| t247 | 13 | 445 | 684 | 449.8 | 486.6 | 442 | 467 |
| t248 | 17 | 445 | 1099 | 466 | 964.2 | 462 | 829 |
| t249 | 11 | 431 | 553 | 434.8 | 340 | 423 | 316 |
| t250 | 20 | 200 | 310 | 200.8 | 252.4 | 193 | 242 |