 doi:10.20944/preprints202205.0301.v1

Article

An Iterated Local Search with Multiple Perturbation Operators and a Varying Perturbation Strength for the Capacitated Team Orienteering Problem with Time Windows

Hassan El Fahim [1,]

[1] Department of Mathematics and Computer Science, Hassan II University of Casablanca, Morocco; contact.dr.elfahimhassan@gmail.com

Abstract: The capacitated team orienteering problem with time windows (CTOPTW) is a NP-hard combinatorial optimization problem. In the CTOPTW, a set of customers is given each with a profit, a demand, a service time and a time window. A homogeneous fleet of vehicles is available for serving customers and collecting their associated profits. Each vehicle is constrained by a maximum tour duration and a limited capacity. The CTOPTW is concerned with the determination of a preset number of vehicle tours that begin and end at a depot, visit each customer no more than once while satisfying the time duration, time window and vehicle capacity constraints on each tour. The objective is to maximize the total profit collected. In this study we propose an iterated local search (ILS) algorithm to deal with the CTOPTW. ILS is a single solution based meta-heuristic that successively invokes a local search procedure to explore the solution space. A perturbation operator is used to modify the current local optimum solution in order to provide a starting solution for the local search procedure. As different problems and instances have different characteristics, the success of the ILS is highly dependent on the local search procedure, the perturbation operator(s) and the perturbation strength. The basic ILS uses a single perturbation operator and the perturbation strength remains the same during the optimization process. To address these issues, we use three different perturbation operators and a varying perturbation strength which changes as the algorithm progresses. The idea is to assign a larger perturbation strength in the early stages of the search in order to focus on exploring the search space. The perturbation strength is gradually decreased so that we focus more on exploitation. The computational results show that the proposed ILS algorithm is able to generate high quality solutions on the CTOPTW benchmark instances taken from the scientific literature, demonstrating its efficiency in terms of both the solution quality and computational time. Moreover, the proposed ILS produces 21 best known results and 5 new best solutions.

Keywords: combinatorial optimization; orienteering problem; meta-heuristic; iterated local search

Citation: El Fahim, H. . *Preprints* **2022**, *1*, 0. https://doi.org/

1. Introduction

Combinatorial optimization is a branch of mathematical optimization with a vast number of problems and applications. Over the last decades, it has grown into a mature field with strong links to various other disciplines like discrete mathematics, computer science and probability theory. Technically speaking, combinatorial optimization problems are concerned with finding among a finite set of solutions the best solution in terms of an objective function. The orienteering problem (OP) is a well-studied combinatorial optimization problem with many real-world applications such as routing of oil tankers [1] and reverse logistics [2]. It is a routing problem where travelling to all destinations is often not feasible due to time or budget constraints. It was first introduced by [3] as the problem of designing a tour, limited in travel time, that maximizes the profits collected by visiting a set of customers. Since then it has received a lot of attention; many researchers have worked and are still working on it as well as on its extensions and applications. The design of tourist itineraries is an example where the OP has been applied ([4], [5]). It seeks to solve the tourist trip design problem [6] whose objective is to select the most interesting combination of attractions to visit within a specific time span. Further applications can be found in the review by [7].

 doi:10.20944/preprints202205.0301.v1

The capacitated team orienteering problem with time windows (CTOPTW) is an extension of the OP that restricts the service to the time window and utilizes multiple vehicles with some operational constraints. In the CTOPTW, a profit is collected if there is a visit to the associated customer within the time window. In addition, each vehicle can perform at most one tour and the total demand of the customers visited in a tour cannot exceed a limited capacity Q. Moreover, the total duration of a tour including the travel and service times cannot exceed a preset time budget T_{max}. The objective is to design a set of m vehicle tours that maximize the total collected profit while satisfying the time window, time budget and vehicle capacity constraints on each tour. The OP is a NP-hard problem [8] so is the CTOPTW. Thus, heuristic methods have drawn the attention of researchers to design algorithms for the CTOPTW. [9] develop an easy to implement iterated local search heuristic for the CTOPTW. The heuristic was originally designed for a variant of the CTOPTW namely the multi-constrained team orienteering problem with time windows (MCTOPTW). This solution method is an adaptation of the heuristic designed by [10] for the well-known team orienteering problem with time windows (TOPTW). [11] develop a hybrid iterated local search algorithm for another variant of the CTOPTW namely the multi-constraint team orienteering problem with multiple time windows (MCTOPMTW). In this algorithm, a greedy randomized adaptive search procedure (GRASP) is embedded in an ILS algorithm to generate initial solutions. When this heuristic is tested on the CTOPTW benchmark instances, good results are obtained on relatively small computational times. [12] propose a variable neighborhood search algorithm for the CTOPTW. Two insertion heuristics are combined to generate initial solutions. The heuristic is able to obtain promising results on the CTOPTW benchmark instances. In [13], the variable neighborhood search algorithm by [12] is improved on some hard benchmark instances. On the other side, there is only one exact algorithm for the CTOPTW in the literature. [14] develop a branch and price algorithm to solve it to optimality. Among all the CTOPTW benchmark instances available in the literature they include just those with two vehicles in their computational study. This branch and price algorithm is able to find the optimal solution on 29 over 39. The branch and price algorithm by [14] is a good example of the unpredictability of exact methods since it can find the optimal solution for instances with 100 customers and fails on small instances.

In this study, we propose an iterated local search algorithm to deal with the CTOPTW. Iterated local search (ILS) is a local-search based meta-heuristic that was introduced by [15] to solve combinatorial optimization problems. It successively invokes a local search procedure to find a local optimum. A perturbation operator is used to modify the current local optimum solution in order to provide a good starting solution for the local search procedure. As different problems and instances have different characteristics, the success of the ILS is highly dependent on the local search procedure, the perturbation operator(s) and the perturbation strength. A small perturbation strength may lead the local search to return to previously visited solutions. If the perturbation strength is too large, this may lead the algorithm to behave as a random restart method, which typically leads to low quality solutions. Note that the basic ILS uses a single perturbation operator and the perturbation strength remains the same during the optimization process. To address these issues, we use three different perturbation operators and a varying perturbation strength which changes as the algorithm progresses. The idea is to assign a larger perturbation strength in the early stages of the search in order to focus on exploring the search space. The perturbation strength is gradually decreased so that we focus more on exploitation.

The rest of this paper includes four additional sections. Section 2 defines the mathematical notation and formulation of the CTOPTW. Section 3 describes the proposed iterated local search and its components. Section 4 compares the proposed iterated local search solution quality and computational time against published results. The last section is devoted to the conclusions.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 May 2022 doi:10.20944/preprints202205.0301.v1

2. Mathematical formulation

The CTOPTW may be defined in terms of a complete graph $G = (V, E)$, where $V = \{0, 1, \ldots, n\}$ is a set of vertices with vertex 0 representing a depot where m identical vehicles, each with a limited capacity Q, are located. The remaining vertices, denoted by $V \setminus \{0\}$, represent customers each having a non-negative profit p_i, a non-negative demand d_i, a non-negative service time s_i and a time window $[e_i, l_i]$. A time window is the period of time during which delivery can be made to a customer. It has two main characteristics: the earliest allowed arrival time denoted by e_i and the latest allowed arrival time denoted by l_i. The edge set connecting the vertices is given by $E = \{(i, j) \mid i, j \in V, i \neq j\}$. A distance matrix $C = \{t_{ij}\}$ is defined on E. In some contexts, t_{ij} may be interpreted as travel time or travel distance from vertex i to vertex j. The CTOPTW consists of designing a set of m vehicle tours such that:

- Each customer is visited at most once,
- Each tour starts from and ends at the depot,
- The total demand of a tour cannot exceed Q,
- The total duration of a tour cannot exceed T_{max},
- Arriving at a customer later than latest time of its time window is strictly forbidden,
- A waiting is incurred if the vehicle reaches to a customer before its earliest time window.

In order to formulate the CTOPTW as a mathematical programming model we need to define the decision variables.
x_{ijk}: binary variable equal to 1 if the vehicle k travels directly from vertex i to vertex j; 0 otherwise.
y_{ik}: binary variable equal to 1 if the vehicle k visits the customer i in its tour; 0 otherwise.
b_{ik}: time variable that specifies the beginning of the service at customer i visited by the vehicle k.
K: the vehicle set.
A: a large constant.

The CTOPTW can be formulated as the following mixed linear programming model.

The objective is to maximize the function f given by the following equation:

$$\max f = \sum_{k \in K} \sum_{i \in V \setminus \{0\}} p_i y_{ik} \tag{1}$$

Subject to the following constraints:

$$\sum_{k \in K} \sum_{j \in V \setminus \{0\}} x_{0jk} = \sum_{k \in K} \sum_{i \in V \setminus \{0\}} x_{i0k} = m \tag{2}$$

Constraint (2) guarantees that each tour must start from and end at the depot.

$$\sum_{i \in V} x_{ilk} = \sum_{j \in V} x_{ljk} = y_{lk} \quad \forall l \in V \setminus \{0\}, \forall k \in K \tag{3}$$

Constraint (3) is about the connectivity of each tour.

$$b_{ik} + s_i + t_{ij} - b_{jk} \leq A(1 - x_{ijk}) \quad \forall i, j \in V, \forall k \in K \tag{4}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 May 2022 doi:10.20944/preprints202205.0301.v1

Constraint (4) determines the time line of each tour.

$$\sum_{k \in K} y_{jk} \leq 1 \quad \forall j \in V \setminus \{0\} \tag{5}$$

Constraint (5) ensures that each customer is visited at most once.

$$\sum_{i \in V \setminus \{0\}} d_i y_{ik} \leq Q \quad \forall k \in K \tag{6}$$

Constraint (6) ensures that the capacity of each vehicle is not exceeded.

$$\sum_{i \in V} (s_i y_{ik} + \sum_{j \in V} t_{ij} x_{ijk}) \leq T_{max} \quad \forall k \in K \tag{7}$$

Constraint (7) confirms that the total travel time for each tour does not exceed the time budget T_{max}.

$$e_i \leq b_{ik} y_{ik} \leq l_i \quad \forall k \in K, \forall i \in V \setminus \{0\} \tag{8}$$

Constraint (8) restricts the start of the service to the time window.

$$x_{ijk}, y_{ik} \in \{0,1\}, \quad b_{ik} \in \mathbb{R}^+ \quad \forall i, j \in V, \forall k \in K \tag{9}$$

Constraint (9) is about the integrality constraints for the decision variables.

3. The proposed solution method

In this section, we first present the basic iterated local search and then we discuss the proposed method and its components.

3.1. Iterated local search

ILS is a meta-heuristic that invokes a local search procedure to explore the solution space. The idea behind this meta-heuristic is very simple: instead of iteratively restarting the local search procedure from randomly generated solutions, ILS produces the initial solution for the next iteration by perturbing the incumbent local optimum. We expect to produce an initial solution located in the basin of attraction of a local optimum that is better in terms of the objective function. ILS combines two mechanisms, intensification and diversification, to explore the solution space and guide the search to promising regions. Intensification aims to optimize the objective function as far as possible within a limited region, while diversification aims to drive the search to explore new promising regions of the solution space. The pseudo-code of the basic ILS is shown in Algorithm 1. ILS starts with an initial solution and performs a local search procedure (intensification stage) until a local optimum is found. Then, the current local optimum is perturbed by means of a perturbation operator (diversification stage) and another round of local search is performed from that solution. This mechanism is repeated until a stopping condition is met. That is the maximum number of iterations allowed and/or the maximum number of iterations between two improvements and/or the maximum computational time (CPU) allowed.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 May 2022 doi:10.20944/preprints202205.0301.v1

Algorithm 1 Pseudo-code of the basic ILS

```
S_0 ← InitialSolution();
S ← LocalSearch(S_0);
while Termination condition is not met do
    Ś ← Perturbation(S);
    Ŝ ← LocalSearch(Ś);
    S ← AcceptanceCriterion(S, Ŝ);
end while
return S
```

3.2. The proposed iterated local search

The proposed ILS follows the general scheme of the basic ILS. As different problems and instances have different characteristics, the success of the ILS is highly dependent on the local search procedure, the perturbation operator(s) and the perturbation strength. The basic ILS scheme presents some degrees of freedom in selecting the appropriate components such as the local search procedure and perturbation operator(s). Moreover, the basic ILS uses a single perturbation operator and the perturbation strength remains the same during the optimization process. To address these issues, we use three different perturbation operators and a varying perturbation strength which changes as the algorithm progresses. The following subsections discuss the components of the proposed ILS.

3.3. Initial solution generation procedure

The procedure used to generate an initial solution for our ILS algorithm is based on the sequential insertion heuristic developed by [16]. This sequential insertion heuristic uses the insertion criterion $c_1(i,u,j)$ to calculate for each unvisited customer u the best place for insertion between two adjacent vertices i and j in the current partial tour T. In a second step, the best customer according to the selection criterion $c_2(i,u,j)$ is selected and inserted between i and j. In this study we extend the heuristic of [16] by using additional components in the insertion and selection criterions.
The insertion criterion is expressed as follows:

$$c_1(i,u,j) = \delta_1 c_{11}(i,u,j) + \delta_2 c_{12}(i,u,j) + \delta_3 c_{13}(i,u,j) + \frac{d_u}{Q} \tag{10}$$

Note that as in [16] we require that the sum of the weighting factors δ_1, δ_2 and δ_3 should be 1.
The components of the insertion criterion are defined as follows:

$$c_{11}(i,u,j) = t_{iu} + t_{uj} - t_{ij} + \delta_4 s_u \tag{11}$$

The cost in equation (11) represents the additional time needed to insert customer u between i and j.

$$c_{12}(i,u,j) = b_j^{new} - b_j \tag{12}$$

The cost in equation (12) represents the time needed to serve customer u which is expressed as the difference between the new start of service time at customer j after inserting u, b_j^{new}, and the original start of service time at j, b_j.

$$c_{13}(i,u,j) = w_u + \sum_{l \in T} w_l \tag{13}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 May 2022 doi:10.20944/preprints202205.0301.v1

The cost in equation (13) is the sum of the waiting time at customer u given that it is inserted between i and j, w_u, and the original waiting time at T, $\sum_{l \in T} w_l$.
The selection criterion is expressed as follows:

$$c_2(i,u,j) = \frac{c_1(i,u,j)}{p_u^{\delta_5}} \tag{14}$$

The criterion in equation (14) gives priority to customers with highest profits that are close to the visited customers in terms of travel time and time window influence. The procedure of customer insertion is repeated until no further unvisited customer can be inserted into the tour under construction. A new tour is then initialized and the loop is performed until m tours are constructed. The insertion heuristic terminates by providing a feasible solution, a set of m vehicle tours, the customers that are assigned to each vehicle and the sequence in which customers are visited. This procedure is repeated with different values of $\delta_i, i = 1,2,3,4,5$ and the best overall solution is returned as the initial solution.

3.4. Perturbation operators

The perturbation operator has a large influence on the performance of an iterated local search and controlling the perturbation strength is quite important. It is responsible for providing, for each call of the local search procedure, a new starting solution by modifying the current local optimum solution in order to escape from the local optimum and to move the search to a new point in the search process. The perturbation size is controlled by two factors: the type of perturbation and the perturbation strength. In this work, we address these issues by proposing multiple perturbation operators. We utilize three perturbation operators to modify the current local optimum solution.
Let $S = \{T_1, T_2, \ldots, T_{m-1}, T_m\}$ be a solution, φ_{T_k} and ϑ_{T_k} are defined for each tour T_k, $(k = 1, \ldots, m)$ as follows:

$$\varphi_{T_k} = \left\lfloor \frac{\chi(T_k)}{2} \right\rfloor \tag{15}$$

$$\vartheta_{T_k} = \left\lceil \frac{\chi(T_k)}{2} \right\rceil \tag{16}$$

Where $\chi(T_k)$ is the number of visited customers on T_k.
The perturbation operators are:

- minProfit: removes the customer with the lowest profit from each tour.
- 2-positions: randomly selects two positions l_k and h_k from each tour T_k as follows: $l_k \in [1, \varphi_{T_k}]$, $h_k \in [\vartheta_{T_k}, \chi(T_k)]$. The customers associated to these positions are removed.
- Restart: performs on each tour T_k by randomly selecting one position f_k as follows: $f_k \in [1, \chi(T_k)]$. The customer associated with each position $l \in [1, f_k]$ is removed.

Note that after a removal, all customers followed the removed ones are shifted towards the beginning of the tour in order to avoid unnecessary waiting time and ensure the continuity of the tour.

3.5. Local search procedure

The local search procedure tries to fulfill the available room in the solution obtained through perturbation by inserting new unvisited customers. As one can intuitively expect, evaluating the possible insertion of each unvisited customer using the selection criterion given by equation (14) may increase the risk of inserting the customers just removed and then getting easily stuck on that solution. The local search procedure performs by inserting each unvisited customer not in its best place but in its first feasible place.
By doing so the following effects are observed:

- The position of a visited customer is changed in the same tour,
- Customers are interchanged between tours,
- New unvisited customers are inserted,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 May 2022 doi:10.20944/preprints202205.0301.v1

3.6. The proposed ILS algorithm

The pseudo-code of the proposed ILS algorithm is presented in Algorithm 2. The algorithm starts with an initial solution generated using the procedure described in 3.3. The parameter N is firstly initialized to 0. The algorithm repeats an outer loop during a fixed number of iterations N_{max}. At each iteration, the current solution is slightly perturbed using minProfit operator. Then it is improved using the local search procedure. The obtained result is used, if it is better, as the starting solution for the inner loop. The algorithm repeats an inner loop during a fixed number of iterations I_{max}. At each cycle, the current solution is firstly perturbed using Restart operator. The local search is then applied to the perturbation output. If the obtained solution improves the best overall solution then the search continues from that solution. Otherwise, the parameter I is incremented and the current solution is perturbed using 2-positions operator. The obtained solution is then improved using the local search procedure. The parameter N is incremented at the end of each cycle of the inner loop. As the reader can intuitively deduce, Restart perturbation operator is the main driver of our algorithm. This perturbation operator promotes long jumps in the solution space. The idea as earlier discussed is to assign a larger perturbation strength in the early stages of the search in order to focus on exploring the search space. The perturbation strength is gradually decreased so that we focus more on exploitation by using two different perturbation operators: 2-positions and minProfit.

Algorithm 2 Pseudo-code of the proposed ILS algorithm

```
1:  S ← InitialSolution();
2:  N ← 0;
3:  while N < N_max do
4:      X_0 ← minProfit(S);
5:      X ← LocalSearch(X_0);
6:      if f(X) > f(S) then
7:          S ← X;
8:          f(S) ← f(X);
9:      end if
10:     I ← 0;
11:     while I < I_max do
12:         Y_0 ← Restart(S);
13:         Y ← LocalSearch(Y_0);
14:         if f(Y) > f(S) then
15:             S ← Y;
16:             f(S) ← f(Y);
17:         else
18:             I ← I + 1;
19:             Z_0 ← 2-positions(S);
20:             Z ← LocalSearch(Z_0);
21:             if f(Z) > f(S) then
22:                 S ← Z;
23:                 f(S) ← f(Z);
24:             end if
25:         end if
26:     end while
27:     N ← N + 1;
28: end while
```

4. Computational results

In this section we report the computational results of our ILS algorithm on the CTOPTW benchmark instances. Our algorithm is implemented in Java language. All experiments are carried out on an Intel(R) Core(TM) i7-6500U CPU 2.50GHz with 12 Go of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 May 2022 doi:10.20944/preprints202205.0301.v1

RAM. In the rest of this section we present the benchmark instances, parameter settings and computational results.

4.1. Benchmark instances

We test our algorithm on the CTOPTW benchmark instances. These instances are based on the benchmark instances created by [16] for the vehicle routing problem with time windows (VRPTW) and those created by [17] for the periodic vehicle routing problem (PVRP). We refer to the instances of [16] as Solomon instances and to the instances of [17] as Cordeau instances. Solomon instances are divided into three categories: c1, r1 and rc1. Each instance has exactly 100 customers. These customers are clustered in category c1, randomly distributed in category r1 and a mix of randomly located and clustered customers in category rc1. The traveling time between customers is equal to the corresponding Euclidean distance. The instances are further categorised according to the scheduling horizon. The instances in category c1 have a long scheduling horizon: 1000 units of time. Whereas, the instances in categories r1 and rc1 have a short scheduling horizon: 230 and 240 units of time respectively. On the other side, Cordeau created 10 benchmark instances for the PVRP: pr01-pr10. The number of customers for Cordeau instances varies from 48 to 288 customers. Based on these instances, [9] designed instances for the MCTOPTW with 1 and 2 vehicles. In these instances, two additional attributes e_1 and e_2 are added to each customer. The first attribute e_{i1} is set equal to the index i of the customer (1 for the first customer, 2 for the second and so on). The value of the second attribute e_{i2} is based on the index i, it is calculated as follows: the value for the first five customers is set equal to 5, the next five customers get the value of 10, the next five the value of 15, the next five the value of 5 and so on. Being a constraint, each attribute e_l $(l = 1, 2)$ has an upper bound E_l. The CTOPTW benchmark instances are obtained as follows: $d_i = e_{i1}$ and $Q = E_1$ or $d_i = e_{i2}$ and $Q = E_2$.

4.2. Parameter settings

In the following, we investigate and justify the parameter settings used for the computation of the results. The proposed ILS algorithm has 7 parameters that need to be set in advance. We conduct two sets of experiments to set these parameters. In both sets, we randomly select 20 different instances for the purpose of parameter calibration. The instances are selected as follows: 5 instances from each category c1, r1, rc1 and pr. We aim to select instances that are representative of the characteristics found in the whole set of instances (number of customers, distribution of customers, scheduling horizon, time window density, etc.). First, we conduct a set of experiments to determine the values of the weighting factors δ_i, $i = 1, 2, 3, 4, 5$. We test all combinations of the following values: $\delta_i \in [0, 1]$, $(i = 1, 2, 3, 4)$ in increments of 0.1 unit and $\delta_5 = 1, 2, 3, 4$.
The results show that:

- By given more importance to δ_1 in the selection criterion we often obtain the best overall solution on Solomon instances especially on r1 and rc1 instances. This can be explained by the fact that customers in these categories are located far from one another and as a consequence the travel times are longer than in other categories.
- When δ_4 increases we obtain good results on Cordeau instances. The service time for these instances varies from one customer to another, whereas the service time is the same for all customers on Solomon instances.
- We obtain good results on Solomon instances when δ_3 is increased. This is due to the fact that these instances have tight time windows and then the waiting times might be long.
- The best overall solution can be obtained for different values of δ_5.
- We obtain good results with small values of δ_2.

The following values seem to have the best performance:
δ_1 = 0.0, 0.1, 0.2, 0.3, 0.4, 0.7, 0.9
δ_2 = 0.0, 0.1, 0.2, 0.3
δ_3 = 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 May 2022 doi:10.20944/preprints202205.0301.v1

δ_4 = 0.0, 0.1, 0.5, 0.6, 0.9, 1.0
δ_5 = 2, 3, 4
A second set of experiments are conducted to obtain the most appropriate values for the number of iterations I_{max} and N_{max}. We test all combinations of the following values: $N_{max} = 25, 100, 400$ and $I_{max} = 15, 30, 60$. As expected, the results show that increasing the values of these parameters may lead to small improvement of the solution quality at the expense of additional computational time. Thus, setting this pair to $(N_{max}, I_{max}) = (100, 15)$ realizes the best trade-off between solution quality and computational time.

4.3. Computational results on the CTOPTW benchmark instances

In this section we compare the proposed ILS solution quality and computational time against published results. The proposed ILS is compared to the following state-of-the-art algorithms:
B&P: the branch and price algorithm by [14].
ILS: the iterated local search algorithm by [9].
HILS: the hybrid iterated local search algorithm by [9].
VNS: the variable neighborhood search algorithm by [12].
Enhanced VNS: the enhanced variable neighborhood search algorithm by [13].
Tables 1 and 2 summarize the results for Solomon and Cordeau instances. The headings of these tables are defined as follows:
Instance-n: the name of the benchmark instance over which the algorithms are tested; the digits at the end give the number of customers.
BK: the best known profit.
B&P: the result of branch and price algorithm by [14] including the optimal profit, the percentual gap, Gap (%), with the best known profit and the computational time in seconds, CPU (s).
Gap (%) defines the relative deviation from the best known profit. It is expressed as follows:

$$\text{Gap (\%)} = \frac{\text{BK - profit}}{\text{BK}} \times 100 \tag{17}$$

ILS: the result of the iterated local search algorithm by [9] including the profit, the percentual gap with the best known profit and the computational time.
HILS: the average result over 10 runs of the hybrid iterated local search algorithm by [11] including the average profit, the associated percentual gap with the best known profit and the average computational time.
VNS: the average result over 10 runs of the variable neighborhood search algorithm by [12] including the average profit, the associated percentual gap with the best known profit and the average computational time.
Enhanced VNS: the average result over 10 runs of the enhanced variable neighborhood search algorithm by [13] including the average profit, the associated percentual gap with the best known profit and the average computational time.
The proposed ILS: the result over 10 runs of the proposed iterated local search algorithm including the worst profit, the best profit and its associated percentual gap with the best known profit, and the average profit, its associated percentual gap with the best known profit and the average computational time.
The row Avg reports the average percentual gap on each category and the average computational time required by each algorithm on each category. The N/A symbol means not available, that is, the result has not been published. A bold font indicates that we obtain new best value. In the present study as in all studies, the results of many algorithms are compared as follows:

- When comparing the results of two stochastic algorithms, the average results and average computational times of multiple runs, usually 5 or 10, are considered for comparison.

 doi:10.20944/preprints202205.0301.v1

- When comparing the results of a deterministic algorithm with those of a stochastic algorithm, the results of 1 run of the deterministic algorithm are compared with the average results of multiple runs of the stochastic algorithm.
- When the best results of multiple runs are used for comparison, it should be the total computational times of all runs that are used for comparison.

The results from Tables 1 and 2 show that:
On the benchmark instances with one vehicle, the proposed ILS algorithm performs on average better than ILS and VNS algorithms on c1 and r1 instances. The average computational time of the proposed ILS algorithm on c1 category for example is 0.1 seconds. On Solomon instances, HILS algorithm is on average better than the proposed ILS algorithm, VNS algorithm, ILS algorithm in terms of solution quality. The average gap of the proposed ILS algorithm, VNS algorithm, ILS algorithm and HILS algorithm on c1 category is 0.6%, 0.9%, 2.7% and 0.5% respectively. The proposed ILS algorithm succeeds in finding the best known solution on 11 instances, on c1 category it finds the best known solution on 6 instances over 9. On Cordeau instances, the proposed ILS algorithm outperforms all state-of-the-art algorithms. The average gap of the proposed ILS algorithm, HILS algorithm, ILS algorithm, VNS algorithm and enhanced VNS algorithm is 2.1%, 3.9%, 3.7%, 6.4% and 3.9% respectively. The average computational time of the proposed ILS algorithm, HILS algorithm, ILS algorithm, VNS algorithm and enhanced VNS algorithm is 0.5, 0.4, 4.3, 0.4, 1.4 and 1.8 seconds respectively. The proposed ILS algorithm is able to achieve 2 new best solutions.
For benchmark instances considering two vehicles, the proposed ILS algorithm is on average better than VNS algorithm both with regard to solution quality and computational time on c1 and r1 instances. On all Solomon instances, ILS algorithm outperforms on average all state-of-the-art algorithm using additional computational effort. The average computational time of the proposed ILS algorithm, VNS algorithm, ILS algorithm and HILS algorithm on c1 instances for example is 0.2, 0.4, 0.5 and 1.4 seconds respectively. On c1 instances, the proposed ILS algorithm is able to achieve the best known solution on 4 instances over 9. On Cordeau instances, the proposed ILS algorithm obtains the best performance among all algorithms. The average gap of the proposed ILS algorithm, VNS algorithm, enhanced VNS algorithm, ILS algorithm and HILS algorithm is 1.2%, 3.8%, 3.0%, 1.7% and 5.2% respectively. The average computational time of the proposed ILS algorithm, HILS algorithm, ILS algorithm, VNS algorithm and enhanced VNS algorithm is 1.0, 1.2, 30.0, 2.6 and 3.9 seconds respectively. The proposed ILS algorithm is able to achieve 3 new best solutions. The authors in [14] have failed in finding a solution for 9 over 10 of Cordeau instances which are known to be difficult to solve. Our 5 new best solutions are obtained on the following benchmark instances:
pr05-240 with one vehicle, the objective value of the new solution is 578.
pr08-144 with one vehicle, the objective value of the new solution is 462.
pr04-192 with two vehicles, the objective value of the new solution is 873.
pr05-240 with two vehicles, the objective value of the new solution is 1048.
pr08-144 with two vehicles, the objective value of the new solution is 801.
In summary, the proposed ILS algorithm is able to achieve high quality solutions on the CTOPTW benchmark instances especially on Cordeau instances which are known to be difficult to solve. It is able to find 21 best known solutions and 5 new best solutions (indicated in bold numbers).

5. Conclusion

In this paper an algorithm based on the iterated local search meta-heuristic has been discussed for the capacitated team orienteering problem with time windows. The solution method that we propose uses multiple perturbation operators and a varying perturbation strength that changes as the algorithm progresses. We have succeeded in finding 21 of the best known solutions on the CTOPTW benchmark instances and 5 new best solutions. In all test sets, Cordeau instances are solved better than Solomon instances which have tight

time windows. A cyclic exchange procedure may reduce this gap by cyclically exchanging paths of customers between the tours in order to make room and insert other unvisited customers. Since the chosen acceptance criterion has a critical influence on the balance between intensification and diversification of the search, a possible improvement of the algorithm could involve also considering worst solutions during the search. One possibility is to work with a simulated annealing acceptance criterion.

It should be noted that all the new best solutions we found were obtained on Cordeau instances. So the question we would like to ask is: what make these instances relatively easy for our algorithm ?

First, the instances in question are quite large (144, 192 and 240 customers) and have large time windows. Second, we know that the size of an instance determines the solution space size. Third, as the reader can intuitively deduce, Restart perturbation operator is the main driver of our algorithm. Indeed, this perturbation operator promotes long jumps in the solution space. Based on these points, a sophisticated fitness landscape analysis (FLA) should be conducted to find good insights into CTOPTW structure. This may be the subject of future works.

References

1. Golden, B.L; Asken, D; Tekin, M.T. The orienteering problem. *Naval Research Logistics* **1987** *34:3*, 307–318.
2. Aras, N; Levy, L; Vohra, R. Selective multi-depot vehicle routing problem with pricing. *Transportation Research Part C* **2011** *19*, 866–884.
3. Tsiligirides, T. Heuristic Methods Applied to Orienteering. *Journal of the Operational Research Society* **1984** *35:9*, 779–809
4. Vansteenwegen, P.; Van Oudheusden, D. The mobile tourist guide: an or opportunity. *OR Insight* **2007** *20:3*, 21–27.
5. Vansteenwegen, P.; Souffriau, W.; Berghe, G.V.; Van Oudheusden, D. The city trip planner: an expert system for tourists. *Expert Systems with Applications* **2011** *38:6*, 6540–6546.
6. Ruiz-Meza, J.; Montoya-Torres, J.R. A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines. *Operations Research Perspectives* **2022** *9*, 1–28.
7. Vansteenwegen, P.; Souffriau, W.; Van Oudheusden, D. The orienteering problem: a survey. *European Journal of Operational Research* **2011** *209:1*, 1–10.
8. Golden, B.L.; Levy, L.; Vohra, R. The orienteering problem. *OR Naval Research Logistics* **1987** *34:3*, 307–318.
9. Garcia, A.; Vansteenwegen, P.; Souffriau, W.; Arbelaitz, O.; Linaza, M.T. Solving Multi Constrained Team orienteering Problems to Generate Tourist Routes. **2009**.
10. Vansteenwegen, P.; Souffriau, W.; Berghe, G.V.; Van Oudheusden, D. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research* **2009** *36*, 3281–3290.
11. Souffriau, W.; Vansteenwegen, P.; Berghe, G.V.; Van Oudheusden, D. The Multiconstraint Team Orienteering Problem with Multiple Time Windows. *Transportation Science* **2013** *47:1*, 53–63.
12. Aghezzaf, B.; El Fahim, H. The multi-constraint team orienteering problem with time windows in the context of distribution problems: A variable neighborhood search algorithm. In Proceedings of International Conference on Logistics Operations Management, Rabat, Morocco, 2014, pp. 155–160.
13. Aghezzaf, B.; El Fahim, H. Solving the Capacitated Team Orienteering Problem with Time Windows Through Variable Neighborhood Search. *International Review on Computers and Software (IRECOS)* **2015** *10:11*, 1134–1142.
14. Park, J.; Lee, J.; Ahn, S.; Bae, J.; Tae, H. Exact Algorithm for the Capacitated Team Orienteering Problem with Time Windows. *Mathematical Problems in Engineering* **2017**.
15. Lourenço, H; Martin, O; Stützle, T. Iterated local search. In *Handbook of Metaheuristics, International Series in Operations Research and Management Science*; vol. 57. Kluwer Academic Publishers; 2002; pp. 321–353.
16. Solomon, M.M. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research* **1987** *35*, 254–265.
17. Cordeau, J.F.; Gendreau, M.; Laporte, G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* **1997** *30*, 105–119.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 May 2022 doi:10.20944/preprints202205.0301.v1

Table 1. Detailed results for the CTOPTW benchmark instances with $m = 1$

Instance-n	BK	B&P			ILS			HILS			VNS			Enhanced VNS			The proposed ILS					
		Results (1 run)			Results (1 run)			Avg results (10 runs)			Avg results (10 runs)			Avg results (10 runs)			Results (10 runs)					
																	Worst	Best		Avg		
		Profit	Gap (%)	CPU (s)	Profit	Gap (%)	CPU (s)	Profit	Gap (%)	CPU (s)	Profit	Gap (%)	CPU (s)	Profit	Gap (%)	CPU (s)	Profit	Profit	Gap (%)	Profit	Gap (%)	CPU (s)
c101-100	320	N/A	N/A	N/A	300	6.3	0.3	320.0	0.0	0.1	320	0.0	0.1	N/A	N/A	N/A	320	320	0.0	320.0	0.0	0.1
c102-100	360	N/A	N/A	N/A	360	0.0	0.4	360.0	0.0	0.2	360	0.0	0.1	N/A	N/A	N/A	360	360	0.0	360.0	0.0	0.1
c103-100	400	N/A	N/A	N/A	380	5.0	0.3	393.0	1.8	0.3	390	2.5	0.1	N/A	N/A	N/A	390	390	2.5	390.0	2.5	0.2
c104-100	420	N/A	N/A	N/A	400	4.8	0.3	409.0	2.6	0.3	420	0.0	0.0	N/A	N/A	N/A	420	420	0.0	420.0	0.0	0.2
c105-100	340	N/A	N/A	N/A	330	2.9	0.3	340.0	0.0	0.1	340	0.0	0.1	N/A	N/A	N/A	340	340	0.0	340.0	0.0	0.1
c106-100	340	N/A	N/A	N/A	340	0.0	0.3	340.0	0.0	0.1	340	0.0	0.0	N/A	N/A	N/A	340	340	0.0	340.0	0.0	0.1
c107-100	370	N/A	N/A	N/A	370	0.0	0.5	370.0	0.0	0.1	360	2.7	0.0	N/A	N/A	N/A	360	360	2.7	360.0	2.7	0.2
c108-100	370	N/A	N/A	N/A	350	5.4	0.3	370.0	0.0	0.2	360	2.7	0.0	N/A	N/A	N/A	370	370	0.0	370.0	0.0	0.2
c109-100	380	N/A	N/A	N/A	380	0.0	0.3	380.0	0.0	0.2	380	0.0	0.0	N/A	N/A	N/A	380	380	0.0	380.0	0.0	0.2
Avg						2.7	0.3		0.5	0.2		0.9	0.0						0.6		0.6	0.1
r101-100	198	N/A	N/A	N/A	182	8.1	0.1	198.0	0.0	0.1	198	0.0	0.0	N/A	N/A	N/A	198	198	0.0	198.0	0.0	0.1
r102-100	286	N/A	N/A	N/A	281	1.7	0.3	283.8	0.8	0.1	286	0.0	0.0	N/A	N/A	N/A	286	286	0.0	286.0	0.0	0.1
r103-100	293	N/A	N/A	N/A	286	2.4	0.4	289.8	1.1	0.2	286	2.4	0.0	N/A	N/A	N/A	286	286	2.4	286.0	2.4	0.1
r104-100	297	N/A	N/A	N/A	288	3.0	0.3	295.2	0.6	0.2	297	0.0	0.0	N/A	N/A	N/A	297	297	0.0	297.0	0.0	0.1
r105-100	247	N/A	N/A	N/A	247	0.0	0.3	247.0	0.0	0.1	247	0.0	0.0	N/A	N/A	N/A	247	247	0.0	247.0	0.0	0.1
r106-100	293	N/A	N/A	N/A	281	4.1	0.3	290.6	0.8	0.2	293	0.0	0.0	N/A	N/A	N/A	293	293	0.0	293.0	0.0	0.1
r107-100	294	N/A	N/A	N/A	289	1.7	0.4	292.2	0.6	0.2	288	2.0	0.0	N/A	N/A	N/A	288	288	2.0	288.0	2.0	0.1
r108-100	308	N/A	N/A	N/A	308	0.0	0.7	300.5	2.4	0.2	303	1.6	0.1	N/A	N/A	N/A	303	303	1.6	303.0	1.6	0.2
r109-100	277	N/A	N/A	N/A	276	0.4	0.3	277.0	0.0	0.1	276	0.4	0.0	N/A	N/A	N/A	276	276	0.4	276.0	0.4	0.1
r110-100	283	N/A	N/A	N/A	274	3.2	0.6	278.7	1.5	0.1	281	0.7	0.0	N/A	N/A	N/A	281	281	0.7	281.0	0.7	0.2
r111-100	297	N/A	N/A	N/A	295	0.7	0.5	296.4	0.2	0.2	283	4.7	0.0	N/A	N/A	N/A	283	283	4.7	283.0	4.7	0.2
r112-100	295	N/A	N/A	N/A	292	1.0	0.4	292.9	0.7	0.2	287	2.7	0.0	N/A	N/A	N/A	295	295	0.0	295.0	0.0	0.2
Avg						2.2	0.4		0.7	0.2		1.2	0.0						1.0		1.0	0.1
rc101-100	219	N/A	N/A	N/A	216	1.4	0.2	219.0	0.0	0.1	216	1.4	0.0	N/A	N/A	N/A	213	213	2.7	213.0	2.7	0.1
rc102-100	266	N/A	N/A	N/A	259	2.6	0.2	260.4	2.1	0.1	259	2.6	0.0	N/A	N/A	N/A	259	259	2.6	259.0	2.6	0.1
rc103-100	266	N/A	N/A	N/A	259	2.6	0.2	259.2	2.6	0.1	236	11.3	0.0	N/A	N/A	N/A	243	243	8.6	243.0	8.6	0.1
rc104-100	301	N/A	N/A	N/A	301	0.0	0.2	295.5	1.8	0.1	271	10.0	0.0	N/A	N/A	N/A	271	271	10.0	271.0	10.0	0.1
rc105-100	244	N/A	N/A	N/A	213	12.7	0.2	240.0	1.6	0.1	244	0.0	0.0	N/A	N/A	N/A	244	244	0.0	244.0	0.0	0.1
rc106-100	250	N/A	N/A	N/A	233	6.8	0.5	247.2	1.1	0.1	244	2.4	0.0	N/A	N/A	N/A	244	244	2.4	244.0	2.4	0.1
rc107-100	277	N/A	N/A	N/A	270	2.5	0.3	272.3	1.7	0.1	266	4.0	0.1	N/A	N/A	N/A	266	266	4.0	266.0	4.0	0.1
rc108-100	298	N/A	N/A	N/A	298	0.0	0.4	274.9	7.8	0.1	276	7.4	0.1	N/A	N/A	N/A	276	276	7.4	276.0	7.4	0.1
Avg						3.6	0.3		2.3	0.1		4.9	0.0						4.7		4.7	0.1
pr01-48	308	N/A	N/A	N/A	290	5.8	0.4	306.8	0.4	0.1	280	9.1	0.1	308.0	0.0	0.4	308	308	0.0	308.0	0.0	0.2
pr02-96	395	N/A	N/A	N/A	375	5.1	1.0	381.5	3.4	0.2	380	3.8	0.5	395.0	0.0	0.5	395	395	0.0	395.0	0.0	0.3
pr03-144	394	N/A	N/A	N/A	380	3.6	1.7	385.1	2.3	0.3	377	4.3	0.7	375.0	4.8	1.4	377	377	4.3	377.0	4.3	0.3
pr04-192	481	N/A	N/A	N/A	445	7.5	2.5	442.2	8.1	0.5	481	0.0	1.3	438.0	8.9	2.2	475	481	0.0	477.0	0.8	0.6
pr05-240	573	N/A	N/A	N/A	521	9.1	10.0	522.5	8.8	0.7	521	9.1	2.9	573.0	0.0	3.1	573	**578**	**-0.9**	**574.1**	**-0.2**	0.9
pr06-288	534	N/A	N/A	N/A	534	0.0	10.4	502.1	6.0	0.7	498	6.7	2.9	484.0	9.4	3.1	523	523	2.1	523.0	2.1	1.0
pr07-72	298	N/A	N/A	N/A	289	3.0	0.5	296.0	0.7	0.1	285	4.4	0.2	285.0	4.4	0.7	285	285	4.4	285.0	4.4	0.1
pr08-144	458	N/A	N/A	N/A	452	1.3	3.9	444.1	3.0	0.3	455	0.7	1.1	458.0	0.0	1.4	462	**462**	**-0.9**	**462.0**	**-0.9**	0.5
pr09-216	461	N/A	N/A	N/A	461	0.0	3.7	448.0	2.8	0.5	364	21.0	1.3	426.8	7.4	1.8	428	437	5.2	430.2	6.7	0.6
pr10-288	527	N/A	N/A	N/A	517	1.9	9.0	509.8	3.3	0.8	503	4.6	2.8	507.0	3.8	3.1	507	507	3.8	507.0	3.8	1.0
Avg						3.7	4.3		3.9	0.4		6.4	1.4		3.9	1.8			1.8		2.1	0.5

Table 2. Detailed results for the CTOPTW benchmark instances with $m = 2$

Instance-n	BK	B&P			ILS			HILS			VNS			Enhanced VNS			The proposed ILS					
		Results (1 run)			Results (1 run)			Avg results (10 runs)			Avg results (10 runs)			Avg results (10 runs)			Results (10 runs)					
																	Worst	Best		Avg		
		Profit	Gap (%)	CPU (s)	Profit	Gap (%)	CPU (s)	Profit	Gap (%)	CPU (s)	Profit	Gap (%)	CPU (s)	Profit	Gap (%)	CPU (s)	Profit	Profit	Gap (%)	Profit	Gap (%)	CPU (s)
c101-100	590	590	0.0	0.5	580	1.7	1.2	590.0	0.0	0.4	580	1.7	0.5	N/A	N/A	N/A	580	580	1.7	580.0	1.7	0.2
c102-100	660	660	0.0	30.3	650	1.5	1.3	645.0	2.3	0.5	650	1.5	0.4	N/A	N/A	N/A	650	650	1.5	650.0	1.5	0.2
c103-100	720	720	0.0	651.0	710	1.4	2.2	690.0	4.2	0.7	680	5.6	0.4	N/A	N/A	N/A	700	700	2.8	700.0	2.8	0.2
c104-100	760	760	0.0	2494.4	760	0.0	1.2	727.0	4.3	0.8	750	1.3	0.5	N/A	N/A	N/A	760	760	0.0	760.0	0.0	0.3
c105-100	640	640	0.0	2.0	640	0.0	1.2	640.0	0.0	0.4	640	0.0	0.2	N/A	N/A	N/A	640	640	0.0	640.0	0.0	0.2
c106-100	620	620	0.0	12.9	620	0.0	1.1	620.0	0.0	0.4	620	0.0	0.2	N/A	N/A	N/A	620	620	0.0	620.0	0.0	0.2
c107-100	670	670	0.0	17.1	660	1.5	1.3	668.0	0.3	0.5	640	4.5	0.2	N/A	N/A	N/A	640	640	4.5	640.0	4.5	0.1
c108-100	680	680	0.0	23.9	680	0.0	1.3	675.0	0.7	0.5	680	0.0	0.5	N/A	N/A	N/A	680	680	0.0	680.0	0.0	0.3
c109-100	720	720	0.0	37.4	710	1.4	1.5	699.0	2.9	0.6	700	2.8	0.5	N/A	N/A	N/A	700	700	2.8	700.0	2.8	0.2
Avg			0.0	363.3		0.8	1.4		1.6	0.5		1.9	0.4						1.6		1.6	0.2
r101-100	344	344	0.0	0.2	322	6.4	0.3	341.7	0.7	0.3	324	5.8	0.0	N/A	N/A	N/A	326	326	5.2	326.0	5.2	0.1
r102-100	508	508	0.0	57.2	508	0.0	1.3	500.8	1.4	0.4	481	5.3	0.2	N/A	N/A	N/A	484	484	4.7	484.0	4.7	0.2
r103-100	517	517	0.0	2698.3	512	1.0	1.4	508.6	1.6	0.5	513	0.8	0.4	N/A	N/A	N/A	513	513	0.8	513.0	0.8	0.2
r104-100	548	548	0.0	346.1	538	1.8	1.3	521.7	4.8	0.5	512	6.6	0.5	N/A	N/A	N/A	515	515	6.0	515.0	6.0	0.2
r105-100	438	438	0.0	1.2	434	0.9	0.6	437.3	0.2	0.3	411	6.2	0.2	N/A	N/A	N/A	400	400	8.7	400.0	8.7	0.1
r106-100	529	529	0.0	361.3	529	0.0	1.7	514.4	2.8	0.4	503	4.9	0.2	N/A	N/A	N/A	503	503	4.9	503.0	4.9	0.2
r107-100	531	531	0.0	1657.2	523	1.5	0.9	519.9	2.1	0.5	478	10.0	0.2	N/A	N/A	N/A	476	476	10.4	476.0	10.4	0.1
r108-100	555	555	0.0	4749.4	539	2.9	1.0	530.5	4.4	0.5	508	8.5	0.3	N/A	N/A	N/A	510	510	8.1	510.0	8.1	0.3
r109-100	506	506	0.0	7.3	498	1.6	1.7	486.3	3.9	0.4	474	6.3	0.3	N/A	N/A	N/A	482	482	4.7	482.0	4.7	0.2
r110-100	523	523	0.0	27.9	519	0.8	1.5	494.0	5.5	0.4	500	4.4	0.5	N/A	N/A	N/A	500	500	4.4	500.0	4.4	0.3
r111-100	544	544	0.0	83.6	536	1.5	1.1	532.4	2.1	0.5	523	3.9	0.5	N/A	N/A	N/A	525	525	3.5	525.0	3.5	0.2
r112-100	544	544	0.0	464.8	513	5.7	1.3	512.1	5.9	0.4	511	6.1	0.5	N/A	N/A	N/A	517	517	5.0	517.0	5.0	0.2
Avg			0.0	871.2		2.0	1.2		2.9	0.4		5.7	0.3						5.5		5.5	0.2
rc101-100	427	427	0.0	0.3	427	0.0	1.2	425.5	0.4	0.3	385	9.8	0.2	N/A	N/A	N/A	385	385	9.8	385.0	9.8	0.1
rc102-100	505	505	0.0	3.5	497	1.6	1.8	490.4	2.9	0.4	453	10.3	0.2	N/A	N/A	N/A	458	458	9.3	458.0	9.3	0.1
rc103-100	520	520	0.0	6.3	501	3.7	0.7	500.8	3.7	0.4	507	2.5	0.3	N/A	N/A	N/A	508	508	2.3	508.0	2.3	0.2
rc104-100	564	564	0.0	149.0	556	1.4	1.4	536.6	4.9	0.4	551	2.3	0.5	N/A	N/A	N/A	542	542	3.9	542.0	3.9	0.3
rc105-100	480	480	0.0	1.0	448	6.7	0.6	465.1	3.1	0.3	454	5.4	0.2	N/A	N/A	N/A	454	454	5.4	454.0	5.4	0.2
rc106-100	483	483	0.0	3.3	462	4.3	1.0	466.5	3.4	0.3	443	8.3	0.3	N/A	N/A	N/A	456	456	5.6	456.0	5.6	0.2
rc107-100	526	526	0.0	23.2	516	1.9	1.6	497.4	5.4	0.3	485	7.8	0.2	N/A	N/A	N/A	479	479	8.9	479.0	8.9	0.2
rc108-100	551	551	0.0	53.2	526	4.5	1.1	514.4	6.6	0.4	514	6.7	0.2	N/A	N/A	N/A	510	510	7.4	510.0	7.4	0.2
Avg			0.0	30.0		3.0	1.2		3.7	0.3		6.6	0.3						6.6		6.6	0.4
pr01-48	502	502	0.0	77.5	489	2.6	0.9	483.4	3.7	0.3	456	9.2	0.3	495.0	1.4	0.4	500	500	0.4	500.0	0.4	0.2
pr02-96	672	*	*	7200.0	654	2.7	4.4	654.9	2.5	0.6	651	3.1	0.9	659.0	1.9	1.2	661	662	1.5	661	1.6	0.5
pr03-144	701	F	F	7200.0	701	0.0	6.1	686.2	2.1	0.8	676	3.6	1.4	685.0	2.3	1.9	688	699	0.3	691.0	1.4	0.6
pr04-192	872	A	A	7200.0	872	0.0	18.1	827.4	5.1	1.3	847	2.9	2.7	811.3	7.0	6.1	862	**873**	**-0.1**	866.7	0.6	1.0
pr05-240	1040	I	I	7200.0	1002	3.7	42.7	938.0	9.8	2.0	1040	0.0	4.7	1012.3	2.7	10.0	1030	**1048**	**-0.8**	1034.0	0.6	1.5
pr06-288	989	L	L	7200.0	952	3.7	20.9	868.3	12.2	2.1	976	1.3	4.7	989.0	0.0	6.4	989	989	0.0	989.0	0.0	1.9
pr07-72	557	U	U	7200.0	547	1.8	2.8	549.4	1.4	0.4	546	2.0	4.1	547.0	1.8	0.6	538	538	3.4	538.0	3.4	0.3
pr08-144	794	R	R	7200.0	774	2.5	6.5	769.5	3.1	0.9	771	2.9	1.6	794.0	0.0	2.1	796	**801**	**-0.9**	**797.5**	**-0.4**	0.8
pr09-216	828	E	E	7200.0	828	0.0	19.8	786.4	5.0	1.4	763	7.9	2.9	750.2	9.4	3.3	804	806	2.7	804.1	2.9	1.2
pr10-288	998	*	*	7200.0	998	0.0	27.6	925.3	7.3	2.3	951	4.7	2.8	964.3	3.4	6.6	975	997	0.1	984.3	1.4	2.0
Avg						1.7	30.0		5.2	1.2		3.8	2.6		3.0	3.9			0.7		1.2	1.0