Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Thesis Title	Algorithmic techniques for the Tourist Trip Design Problem
Τίτλος Διατριβής	Αλγοριθμικές τεχνικές για το πρόβλημα σχεδιασμού τουριστικών διαδρομών
Ονοματεπώνυμο Φοιτητή	Γεώργιος Σανιδάς
Πατρώνυμο	Ζαφείριος
Αριθμός Μητρώου	ΜΠΠΛ/15062
Επιβλέπων	Χαράλαμπος Κωνσταντόπουλος, Αναπληρωτής Καθηγητής

Ημερομηνία Παράδοσης Μάιος 2019

Τριμελής Εξεταστική Επιτροπή

Χαράλαμπος Κωνσταντόπουλος Αναπλ. Καθηγητής	Άγγελος Πικράκης Επικ. Καθηγητής	Ιωάννης Τασούλας Επικ. Καθηγητής

Abstract

This paper concentrates on the Tourist Trip design problem, a practical application of the Team Orienteering problem, providing an efficient visit schedule of points-of-interest based on predetermined scores. A slightly modified version of a well-known algorithm based on the Iterated Local Search (ILS) is utilized: in contrast to the original ILS algorithm, the tours created must visit the selected Points of Interest (POIs) not only within their time windows but also the remaining time after reaching a POI should be at least the suggested visit time for this POI. Otherwise, the visit is considered unfeasible. Furthermore, the user is allowed to modify the score of each proposed POI by certain percent based on the category it belongs to and on how many POIs of the same category s/he has seen along the part of the tours completed so far.

Περίληψη

Η παρούσα εργασία επικεντρώνεται στο Πρόβλημα Σχεδιασμού Διαδρομών, μιας πρακτικής εφαρμογής του Team Orienteering Problem που προσφέρει ένα αποτελεσματικό προγραμματισμό διαδρομών σε σημεία ενδιαφέροντος (points of interest) βασισμένο σε προκαθορισμένες βαθμολογίες. Μια ελαφρώς τροποποιημένη εκδοχή ενός πολύ γνωστού αλγορίθμου βασισμένου στον Iterated Local Search (ILS) χρησιμοποιείται: σε αντίθεση με τον αρχικό ILS, οι δημιουργημένες διαδρομές πρέπει να επισκέπτονται τα επιλεγμένα σημεία ενδιαφέροντος όχι απλά μόνο μέσα στο χρονικό περιθώριο τους, αλλά ο υπολειπόμενος χρόνος που ακολουθεί τη μετάβαση στο σημείο πρέπει να είναι τουλάχιστον ίσος με τον προτεινόμενο χρόνο επίσκεψης του συγκεκριμένου σημείου. Αλλιώς, η επίσκεψη θεωρείται αδύνατη. Επιπλέον, ο χρήστης έχει τη δυνατότητα να τροποποιεί το σκορ κάθε προτεινόμενου σημείου κατά ένα συγκεκριμένο ποσοστό ανάλογα με την κατηγορία που αυτό ανήκει και με τον αριθμό των σημείων της κατηγορίας αυτής που έχει ήδη επισκεφθεί.

Table of Contents

1. Introduction

The orienteering problem is a subset of the Traveling Salesman Problem and consequently is a NP-hard problem, meaning no exact solution can be found. As such, an array of heuristic algorithms have been used to deliver near optimal solutions. This thesis will focus on the Orienteering problem, its most common extensions as well as its most famous practical application, the Tourist Trip Design problem.

The Orienteering problem (OP) has its roots on a group of sports that combines running and navigation in order to navigate quickly from point to point (also referred as nodes) in an unknown terrain. The element of required speed is introduced by racing against the clock, i.e. having a time limit. Within that time window, the participants are aiming to pass through as many nodes as possible; each node has a score associated with it, by visiting it the participant claims that score as her points. The goal then is to maximize the gathered points before the time limit is reached.

The major subcategories of OP are created by introducing additional constraints, for example attributing a time window to each node or having a different starting and ending point or requiring that each node can only be visited once. OP can also be viewed as a graph, so another example of added complexity is the decision whether the graph is directed or not.

This paper will introduce the most common variations of OP as they pertain to the practical application that it focuses on. Those are the Team Orienteering problem that adds multiple tours, the Orienteering problem with Time Windows, which allows a visit to node to be realized only within the node's time window and the Time dependent Orienteering problem. Then, the Iterated Local Search algorithm, which is used to solve the Team Orienteering problem with Time Windows, is presented along with two variations that group the nodes into clusters (CSCratio and CSCroutes). Finally, our modifications to ILS are introduced: the first forces the participant to stay at each node for the duration of its proposed visit, while the second groups the nodes into categories based on their type and allows the user to modify the score of each node according to the categories that interest him/her or have already been visited before.

2. Literature Review

2.1 The orienteering problem

First mentions of the orienteering problem have been from Tsiligirides, T. (1984)[1] and Golden, B. L. , Levy, L. , & Vohra, R. (1987)[2]. The orienteering problem (OP) at its core combines node selection .i.e. nodes with the task of pinpointing the shortest path between those nodes. In practice the orienteering problem comes with the assumption of a fixed (time) budget and the task of maximizing the total profit by visiting locations (nodes) with associated scores. Since not all nodes can be visited due to time limitations, the OP can be viewed as combining two other combinatorial problems, the Traveling Salesman (TSP) and the Knapsack problem. Naturally the OP is also a combinatorial NP-hard problem.

The OP has been studied extensively and has been given a number of extensions and practical applications. Vansteenwegen, P. , Souffriau, W. , & Van Oudheusden, D. (2011a)[3] , Feillet, D. , Dejax, P. , & Gendreau, M. (2005)[4] and Laporte, G. , & Rodríguez-Martín, I. (2007)[5] are some of the surveys tasked with summing up proposed solutions for OP and it's variants until 2009. A more recent survey (Aldy Gunawan, Hoong Chuin Lau ,Pieter Vansteenwegen (2016)) [6] attempted to cover more recent solutions as well as put more focus on specific practical applications.

2.1.1 Classical OP

Classical OP can be defined in the following way: Assuming a graph-like set of nodes N, with each node i∈N, each with a respective non-negative score with N[i=1] the start and N[i=n] the end, the goal is to design a path (or tour) that will maximize the total profit (the sum of the scores of the nodes visited) within a predetermined time frame. A further limitation prescribes that each node cannot be visited more than once.

OP is usually formulated mathematically, much like an integer problem thus:

First, there are two decision variables employed:

X_{ij} =1, assuming a visit from node i to node j is viable; it will be 0 if not

u_i = the position of node i in the tour

$$Max \sum_{i=2}^{N-1} \sum_{j=2}^{N-1} S_i x_{ij}\,, \tag{0}$$

$$\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1, \tag{1}$$

$$\sum_{i=1}^{N-1} x_{ik} = \sum_{j=2}^{N} x_{kj} \le 1; \quad \forall k = 2, \ldots, N-1, \tag{2}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij}\, x_{ij} \le T_{max}\,, \tag{3}$$

$$2 \le u_i \le N; \quad \forall i = 2, \ldots, N, \tag{4}$$

$$u_i - u_j + 1 \le (N-1)\big(1 - x_{ij}\big); \quad \forall i,j = 2, \ldots, N, \tag{5}$$

$$x_{ij} \in \{0,1\}; \quad \forall i,j = 1, \ldots, N \tag{6}$$

Function (0) maximizes the total collected profit of the path. Constraint(1) stipulates that the path begins at node 1 and ends at node N. Constraint (2) guarantees that there is no isolated node (all nodes are connected) and that each node cannot be visited more than once. Constraint (3) fixes the time window of every path at most at T_{max}, thus ensuring the time budget limit. Generic constraints (5) and (6) eliminate the possibility of subtour creation (see Miller, C., Tucker, A., Zemlin, R., 1960 [7]).

An important assumption of the generic formulation of OP is that travel time between nodes is symmetric according to Euclidian metric, that is $t_{ij} = t_{ji}$. This assumption means that OP as formulated so far represents an undirected complete graph. Most solution in the literature conform to this interpretation.

The proof that OP is NP-hard was given by Golden et al. (1987)[8]; they proved that no algorithm is expected to solve OP optimally. Not unlike any other NP-hard problem, OP proposed solutions are mostly heuristic and approximation ones; exact solutions would be simply too time consuming.

However, a few researchers have proposed exact algorithms to solve OP, albeit on instances with limited amount of nodes. Feillet et al. (2005a) produced a survey of sorts of exact algorithms. Chief among them, Laporte and Martello (1990) [9] utilized branch-and-bound algorithms on instances of up to 20 vertices, while Leifer and Rosenwein (1994) [10] built on their formulation by adding a cutting plane method to achieve better upper bounds. Branch-and-cut algorithms were later found to be able to solve instances of up to 500 vertices (Fischetti et al., 1998) [11]

The main focus of literature is however, as mentioned earlier, the heuristic algorithms. Tsiligirides (1984) who first introduced the term OP, based on the orienteering sport, suggested both a stochastic and a deterministic algorithm, while Golden et al.(1987) a centre-of-gravity algorithm. A 4-phase heuristic was introduced by Ramesh and Brown (1991) [12] and subsequently Chao et al.(1996b) use a 5-step algorithm to outperform any other algorithm mentioned so far.

While various other ideas regarding the OP were introduced, a new solution by Schilde et al.(2009) [13] which aimed to tackle the multi-objective variant was actually found to also outperform Chao et al.(1996b)[14] 5 -step heuristic.

More recent approaches to the OP include Sevkli and Sevilden(2010 a and b) [15][16] which focus on (Discreet) Strengthened Particle Swarm Optimization, Chekuri et al.(2012)-approximation algorithms[17] and a few others which didn’t really offer dramatic performance improvements. Dand et al’s(2013a) [18] branch-and-cut algorithm managed to improve 29 best-known-solution on Chao et al’s(1996b) datasets.

2.1.2 Team Orienteering Problem (TOP)

The most common extension to the OP is allowing for multiple (P) paths within the same graph, each starting and finishing at the predetermined respective positions (N[1] and N[n]) and having T_{max} available time budget . Such a variation is called Team Orienteering Problem (TOP) and was introduced by Chao et al (1996b).

TOP’s mathematical formulation is consequently very similar to the original OP’s.

Like in OP decision variables are employed:

x_{ijp} =1, assuming a visit from node i to node j in path p is viable ; it will be 0 if not

y_{ip} =1, if node i is visited in p

u_{ip} = the position of node i in path p

$$Max\ \sum_{p=1}^{P}\sum_{i=2}^{N-1} S_i y_{ip}\,, \tag{7}$$

$$\sum_{p=1}^{P}\sum_{j=2}^{N} x_{1jp} = \sum_{p=1}^{P}\sum_{i=1}^{N-1} x_{iNp} = P, \tag{8}$$

$$\sum_{p=1}^{P} y_{kp} \leq 1; \quad \forall k = 2, \ldots, N-1, \tag{9}$$

$$\sum_{i=1}^{N-1} x_{ikp} = \sum_{j=2}^{N} x_{kjp} = y_{kp}; \quad \forall k = 2, \ldots, N-1; \forall p = 1, \ldots, P, \tag{10}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} x_{ijp} \leq T_{max} \quad \forall p = 1, \ldots, P, \tag{11}$$

$$2 \leq u_{ip} \leq N; \quad \forall i = 2, \ldots, N; \ \forall p = 1, \ldots, P, \tag{12}$$

$$u_{ip} - u_{jp} + 1 \leq (N-1)\left(1 - x_{ijp}\right); \quad \forall i, j = 2, \ldots, N; \ \forall p = 1, \ldots, P, \tag{13}$$

$$x_{ijp}, y_{ip} \in \{0,1\}; \ \forall i, j = 1, \ldots, N; \ \forall p = 1, \ldots, P \tag{14}$$

These constraints establish similar requirements to the OP: namely the limit time budget available to each path, starting and end points for each path, a guarantee that each will be visited at most once and that no subtours will be generated. Objective function (7), like in OP, states the goal of maximizing the total realized profit.

Exact algorithms for the TOP were produced by Butt and Ryan(1999)[19] using column generation and aiming at solving instances of up to 100 nodes. Boussier et al.(2007)[20] combined column generation with branch-and-bound steps to significantly reduce computation times for instances of 100 nodes.

Chao et al.(1996a) updated their 5-step heuristic to solve the OP, while Tang and Miller-Hooks (2005)[21] utilized a tabu search heuristic in the context of an Adaptive Memory Procedure (AMP). Several other metaheuristics were presented, among others by Archetti et al. (2007), Ke et al. (2008), Vansteenwegen et al. (2009 b,c) and Souffriau et al. (in press). All four of them begin with a starting solution and try to formulate a second one, which will replace the original if found more profitable. In addition to profitability, emphasis is now placed on reducing computation times.

A common framework that is adopted by these so-called local search heuristics, includes five actions that aim to maximize the total profit and two actions that aim at reducing travel time between nodes.

Actions that aim at increasing total profit:

- Insert: This action utilizes cheapest insertion to add an extra node in any of the paths
- TwoInsert: Similar to above, but considering two extra nodes
- Replace: It examines all non-included nodes and inserts one if there is time budget available for insertion. If there is no budget available, the node to be inserted replaces a lower-score one that is already included.
- TwoReplace: Similar to the above, but now every combination of two non-included nodes is considered for insertion.
- Change: Is a different, more drastic approach as five included nodes are removed from a path. Subsequently, non-included nodes are inserted until there is no more time budget available. If the new path that is created by this process is more profitable than the original, the solution is saved, otherwise it is discarded.

Actions that aim at reducing computation times:

- 2-Opt: It replaces two edges included in the path with two new ones. If time reduction is achieved, the change is kept.
- Swap: It swaps one node from a path with another one from another path.

2.1.3 Orienteering Problem with Time Windows (OPTW)

OPTW introduces a concept that drastically alters the procedures mentioned so far needed to solve the OP. Time windows add an additional powerful constraint that a visit to a node can only begin during that window. The decision variables used to formulate this problem are:

X_{ij} =1, assuming a visit from node i to node j is viable; it will be 0 if not

y_i = 1, if node i is visited; it will be 0 if not

s_i = the start of visit at node i

M = a constant

$$Max \sum_{i=2}^{N-1} \sum_{j=2}^{N} S_i x_{ij}\,, \tag{15}$$

$$\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1, \tag{16}$$

$$\sum_{i=1}^{N-1} x_{ik} = \sum_{j=2}^{N} x_{kj} \leq 1; \quad \forall k = 2, \ldots, N-1, \tag{17}$$

$$s_i + t_{ij} - s_j \leq M\left(1 - x_{ij}\right) \quad \forall i, j = 1, \ldots, N, \tag{18}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij}\, x_{ij} \leq T_{max}\,, \tag{19}$$

$$O_i \leq s_i; \quad \forall i = 1, \ldots, N, \tag{20}$$

$$s_i \leq C_i; \quad \forall i = 1, \ldots, N, \tag{21}$$

$$x_{ij} \in \{0,1\}; \quad \forall i, j = 1, \ldots, N \tag{22}$$

The time window constraint practically means that solutions that target the OP can't solve the OPTW whereas OPTW solutions can still be utilized to solve OP (Tricoire et al.(2010))[22] . With this in mind, specific solutions of the OPTW started with Kantor and Rosenwein (1992)[23]. They produced an insertion heuristic that uses a "score over insertion time" ratio to choose inserted nodes while making sure that time windows are not violated. Mansini et al.(2006) [24] developed a neighborhood search heuristic that targets the case where the starting node is also the end node. Lastly, moving in the opposite direction Righini et Salasmi (2006,2009) [25][26] presented an exact algorithm for the OPTW which used dynamic programming to optimally solve instances.

2.1.4 The Team Orienteering Problem with Time Windows (TOPTW)

TOPTW is probably the most common OP extension among those presented this far as it can serve as the basis for a popular OP practical application, the Tourist Trip Design Problem.

Continuing with the same notation we have the following decision variables.

x_{ijp} =1, assuming a visit from node i to node j in path p is viable ; it will be 0 if not

y_{ip} =1, if node i is visited in p

s_{ip} = the start of visit at node i in path p

M = a constant

$$Max\ \sum_{p=1}^{P}\sum_{i=2}^{N-1} S_i y_{ip}\ , \qquad (23)$$

$$\sum_{p=1}^{P}\sum_{j=2}^{N} x_{1jp} = \sum_{p=1}^{P}\sum_{i=1}^{N-1} x_{iNp} = P, \qquad (24)$$

$$\sum_{i=1}^{N-1} x_{ikp} = \sum_{j=2}^{N} x_{kjp} = y_{kp};\quad \forall k = 2,\ldots,N-1; \forall p = 1,\ldots,P\ , \qquad (25)$$

$$s_{ip} + t_{ij} - s_{jp} \leq M\left(1 - x_{ijp}\right);\ \ \forall i = 1,\ldots,N-1; \forall p = 1,\ldots,P \qquad (26)$$

$$\sum_{p=1}^{P} y_{kp} \leq 1\ \forall k = 2,\ldots,N-1, \qquad (27)$$

$$\sum_{i=1}^{N-1}\sum_{j=2}^{N} t_{ij}\, x_{ijp} \leq T_{max}\ , \forall p = 1,\ldots,P\ , \qquad (28)$$

$$O_i \leq s_{ip};\ \ \forall i = 1,\ldots,N-1; \forall p = 1,\ldots,P \qquad (29)$$

$$s_{ip} \leq C_i;\ \ \forall i = 1,\ldots,N-1; \forall p = 1,\ldots,P \qquad (30)$$

$$x_{ijp}\,, y_{ip} \in \{0,1\};\ \ \forall i,j = 1,\ldots,N; \forall p = 1,\ldots,P \qquad (31)$$

Montemanni and Gambardella (2009) [27] developed new instances for TOPTW and produced solution of up to 4 tours based on ant colony optimization, a hierarchical generalization of the TOPTW. Vansteenwegen et al (2009d) [28] developed a fast metaheuristic called Iterated Local Search (ILS) to solve the instances proposed with Solomon , whose optimal solutions are known. ILS is the focal point of this thesis and will be covered in much greater detail later on. Tricoire et al (2010) [29] proposed a Variable Neighborhood Search (VNS) algorithm; their results showed that they managed to produce quality solutions for instances of up to 100 nodes with two tours in one minute of computation time. Tricoire et al (2010) actually worked on the Multi Period Team Orienteering Problem with Time Windows, a generalization of TOPTW.

2.2 The Time Dependent Orienteering Problem (TDOP)

TTDP does away with what has been a major assumption so far, that travel time between two nodes is a constant value. Practically, this is an oversimplification that ignores unforeseen events or network properties that may result in phenomena like congestion. Far more common occurrence is of course the waiting in a station typically associated with public transport systems. As usual, the formulation of the problem begins with the introduction of decision variables.

X_{ijt} = 1: assuming a departure from node i in order to reach node j happens in time slot *t*, it will be 0 otherwise

W_{ijt} : the actual time of the departure within timeslot *t* from node i in order to reach node j

θ_{ijt} : slope coefficient of the linear time-dependent travel time

η_{ijt} : intercept coefficient of the linear time-dependent travel time

τ_{ijt} : lower limit of time slot *t* for arc (*i* , *j*)

T_{iJ} : number of time slots for arc (*i* , *j*)

The objective function that maximizes the total profit is

$$Max \sum_{i=2}^{N-1} \sum_{j=2}^{N} \sum_{t=1}^{T_{ij}} S_i X_{ijt}, \tag{32}$$

The following constraints must be respected:

$$\sum_{j=2}^{N} X_{1j1} = \sum_{i=1}^{N-1} \sum_{t=1}^{T_{iN}} X_{iNt} = 1, \tag{33}$$

$$\sum_{i=1}^{N-1} \sum_{t=1}^{T_{ih}} X_{iht} = \sum_{j=2}^{N} \sum_{t=1}^{T_{hj}} X_{hjt} \leq 1; \ \forall h = 2, \dots, N-1, \tag{34}$$

$$\sum_{i=1}^{N-1} \sum_{t=1}^{T_{ih}} [\, W_{iht} + (\theta_{iht} W_{iht} + \eta_{iht} X_{iht}\,)\,] = \sum_{j=2}^{N} \sum_{t=1}^{T_{hj}} W_{hjt}; \ \forall h = 2, \dots, N-1, \tag{35}$$

$$X_{1j1} \tau_{ijt} \leq W_{ijt} \leq X_{ijt} \tau_{ij(t+1)}; i = 1 \dots, N-1, j = 2 \dots . N; \ \forall t \tag{36}$$

$$\sum_{i=1}^{N-1} \sum_{J=2}^{N} \sum_{t=1}^{T_{ij}} [\, \theta_{ijt} W_{ijt} + \eta_{ijt} X_{ijt}\,)\,] \leq T_{max} \tag{37}$$

$$W_{1i1} = 0; \ \forall i = 1, \dots, N, \tag{38}$$

$$0 \leq W_{ijt} \leq T_{max}; \ \forall \, t, i, j = 1, \dots, N \tag{39}$$

Constraint (33) forces the path to start and end at nodes 1 and N, while constraint (34) ensures that each node is visited at most once. Constraint (35) removes any waiting time: it does so by guaranteeing that the departure time of a subsequent node is the sum of the departure time of the preceding node and the travel time needed to reach the former. Constraints (36) and (37) enforce the limited travel time: they do that by categorizing the departure time in the right time slot. Constraint (38) forces each path to start in the first time slot, while constraint (39) ensures that all departure times are less than T_{max} .

Li(2012) employed network planning as well as dynamic node labeling programming to construct an algorithm, while assuming a realistic transportation system with given start and end nodes. Verbeeck et al.(2014a) utilized the concept of an Ant Colony System combined with a (time-dependent) local search procedure with its own evaluation metric. Gunawan et al.(2014) solve TDOP through a different prism, that of a practical application giving directions inside a large leisure facility e.g. a museum, etc. Four metaheuristics are used to reach solutions with acceptable computational times: a restart greedy algorithm, a restart Variable Neighborhood Descent heuristic, a basic ILS and a modified ILS which uses an adaptive perturbation size.

As expected, a common extension of TDOP is the Time Dependent Orienteering Problem with Time Windows (TDOPTW). Garcia et al.(2010,2013), Abbaspoor and Samadzadegan(2011) and Gavalas et al(2014) are among the TDOPTW proposed solutions.

3. Iterated Local Search procedure for the Team Orienteering Problem with Time Windows

3.1 Introduction

ILS is widely regarded as the most popular solution for the TOPTW. Its popularity derives from the balance between computational time and high quality solutions that it achieves. It was introduced by Pieter Vansteenwegen et al. in 2009. The paper was based on the idea of having an electronic assistance application to help tourists plan their trip. It is thus a Tourist Trip Design problem, with TOPTW its simplified version.

The premise is that the tourist-user needs help in organizing trips for each day he has at his disposal. Each day trip will be relegated to a route. The goal then is to maximize the total score collected by the fixed number of routes. A route naturally must be comprised by timely visits to points of interest (POI) and must have a fixed duration. It is common for a route to have the same starting and final point-node, for example a hotel, but this is not explicitly required.

An additional and very important requirement is the ability to quickly recalculate a proposed trip because of altered real life circumstances. For example, if the user stays at a POI longer than originally accounted for, a recalculation of the entire solution must be undertaken in order to utilize this new information. This is a crucial point for the metaheuristic that will be employed to reach the solution, as it is unlikely that a computational time of more than a few seconds will be deemed acceptable by the user every time such a recalculation is needed. The introduction of this requirement is what made ILS such an important metaheuristic in the literature.

3.2 Mathematical formulation

The mathematical formulation for the TOPTW was already given in literature but will be repeated here for continuity.

As already stated, the locations to be visited are points of interest. Each POI in a set of n locations, which can also be seen as a node *i* in a graph, has a score that is realized by visiting it, a visiting time T_i , and a time window in which it can be visited, i.e. an opening time and a closing time $[O_i, C_i]$. The first node (1) and the end node (*n*) of every tour must be fixed and as already mentioned may or may not be the same. Since each route usually corresponds to a single day so naturally it has a time budget T_{max} which means that not all nodes can be visited in a route and probably not even across a number of routes. The specific goal for each route then is to maximize the total profit realized by visiting as many nodes as the time budget allows within their respective windows. A node should only be visited once and the arrival at it can happen before it's opening time, but in this case a waiting time must be allowed for since the visit can't actually occur before the node's opening time.

Due to the nature of the constraints imposed on it, TOPTW is a rather difficult problem to solve; in fact Golden et al. have already proved that is an NP-hard problem, meaning an optimal

solution can’t be reached in polynomial time. In this context, developing a heuristic that can produce a near-optimal solution in mere seconds is not a simple task.
Based on the notation introduced so far, TOPTW mathematical formulation begins with the following decision variables:

x_{ijp} =1, when in route p node j is visited after the visit to node *i* ; it will be 0 if not

y_{ip} =1, if node *i* is visited in route p; it will be 0 if not

s_{ip} = the start of visit at node *i* in route p

M = a large constant

The following constraints reproduce the requirements mentioned above:

$$Max\ \sum_{p=1}^{P}\sum_{i=2}^{N-1} S_i y_{ip}\,, \tag{40}$$

$$\sum_{p=1}^{P}\sum_{j=2}^{N} x_{1jp} = \sum_{p=1}^{P}\sum_{i=1}^{N-1} x_{iNp} = P, \tag{41}$$

$$\sum_{i=1}^{N-1} x_{ikp} = \sum_{j=2}^{N} x_{kjp} = y_{kp}; \quad \forall k = 2,\dots,N-1; \forall p = 1,\dots,P\,, \tag{42}$$

$$s_{ip} + t_{ij} - s_{jp} \le M\left(1 - x_{ijp}\right); \quad \forall i = 1,\dots,N-1; \forall p = 1,\dots,P \tag{43}$$

$$\sum_{p=1}^{P} y_{kp} \le 1\ \forall k = 2,\dots,N-1, \tag{44}$$

$$\sum_{i=1}^{N-1}\sum_{j=2}^{N} t_{ij}\, x_{ijp} \le T_{max}\,,\ \forall p = 1,\dots,P\,, \tag{45}$$

$$O_i \le s_{ip}; \quad \forall i = 1,\dots,N-1; \forall p = 1,\dots,P \tag{46}$$

$$s_{ip} \le C_i; \quad \forall i = 1,\dots,N-1; \forall p = 1,\dots,P \tag{47}$$

$$x_{ijp}\,, y_{ip} \in \{0,1\}; \quad \forall i,j = 1,\dots,N; \forall p = 1,\dots,P \tag{48}$$

Objective function (40) demands the maximization of total collected score S. Constraint (41) ensures that all tours start at node 1 and end at node N. Constraint (42) guarantees the connectivity of each tour while (43) it’s timeline. Constraint (44) ensures that each node cannot be visited more than once and constraint (45) limits each tour’s duration to the predetermined time budget T_{max}. Constraint (46) states that each visit cannot start before a POI’s opening time while (47) demands that the visit cannot start after a POI’s closing time.

Vansteveegen et al(2009) seminal paper introduced a very fast local search procedure that also performs very well on the available data sets. The procedure is based on an insertion step and a removal (shaking) step to avoid local optima.

3.3 Methodology

3.3.1 Insertion step

The insertion step aims to add one after another all possible visits in a tour while simultaneously respecting the time budget available. In addition, after each node insertion it must ensure that all previously inserted nodes whose visits happen after the just inserted one still have their time windows respected. This is a key point for the computation speed of the whole algorithm, as there will be needed approximately as many such evaluations as there are possible nodes to be visited. A way had to be found to simplify and increase the speed of these calculations. The proposed solution was to record two helper variables for each included node, the Wait and the MaxShift. Intuitively, Wait represents the time that will have to pass before an actual visit to node can be started if someone arrives at it before it's opening time. If on the other hand the arrival a_i is during the node's time window, then Wait is zero.

$$Wait_i = \max[0, O_i - a_i]; \qquad (49)$$

MaxShift represents the time a visit completion can be delayed while simultaneously respecting both its and all the following nodes time windows. MaxShift of node i is the sum of the Wait and the MaxShift of the following node i+1 or the duration of its own visit as defined by its time window, whatever is less.

$$MaxShift_i = \min[C_i - S_i, MaxShift_{i+1} + Wait_{i+1}]; \qquad (50)$$

Knowing the MaxShift of all nodes already included in a tour means that any evaluation regarding a new candidate node will take constant time instead of linear.

The delay each new node insertion will impose to the consequent nodes i.e., the total time expenditure of inserting a new node j between nodes *i* and *k* is given by the following formula:

$$Shift_i = c_{ij} + Wait_j + T_j + c_{jk} - c_{ik}\ ; \qquad (51)$$

In order for j to be eligible to be inserted between *i* and *k*, $Shift_j$ must be less than or equal to the sum of $MaxShift_k + Wait_k$ of node k. Additionally, node j 's own time window must be respected.

The insertion procedure first calculates the best possible place of insertion in the tour for all not already included nodes by minimizing their possible Shift. Then an insertion metric for each node is calculated, which has the form of the following ratio:

$$Ratio_i = (S_i)^2 / Shift_i \qquad (52)$$

The node with the highest ratio will be chosen for insertion in the tour. The ratio places more focus on the score of each node rather than it's time consumption because of the time windows constraint and that is manifested by having the square of the score rather than the score itself included in the calculation.

After the node with the highest ratio is inserted into the tour, a number of variables for the nodes already in the tour will need to be updated to facilitate the next insertion. Particularly important are the nodes that represent visits that are to happen after the recently inserted one. For these nodes, their arrival, waiting time, start of the actual visit, shift and MaxShift need to be updated. These variables are updated through the following formulas:

$$Wait_{k^*} = \max\left[0, Wait_k - Shift_j\right];$$

$a_{k^*} = a_k + Shift_j;$
$Shift_k = \max[0, Wait_j - Wait_k];$
$s_{k^*} = s_k + Shift_k;$
$MaxShift_{k^*} = MaxShift_k - Shift_k;$

All subsequent nodes will be updated through these formulas until the shift will be zero: after this point no other node would be affected by the insertion. MaxShift will also be updated for node j as well as its previous nodes.

The following figure presents the pseudocode of the insertion step.

For each not already included node:
    Calculate shift and best possible insertion position;
    Calculate ratio;
Insert node with highest ratio;
For inserted node j:
    Calculate Arrival, Start (of actual visit), Wait;
For each node after recently inserted node j:
    Update Arrival, Start (of actual visit), Wait, Shift, MaxShift;
For inserted node j:
    Calculate MaxShift;
For each node before inserted node j;
    Update MaxShift;

3.3.2 Shake Step

The insertion step is finished when there are no more possible insertions available. At that time, a shake step is introduced in order to avoid local optima. Specifically, a number of nodes will be removed from the tour to allow reinsertion in the pursuit of optimality. Two integers are used to represent two decisions needed at this point, the number of consecutives nodes to be removed (R_p) and the position in the tour that the removal will begin (S_p). If the sequence of nodes to be removed includes the end node, the process will pick up with the starting node.

After the removal of one or more nodes, a gap will occur in the tour that will generate unnecessary waiting time. For that purpose, all nodes will be shifted towards the beginning of the tour to close that gap. However, if a node's time window doesn't allow for that node's shifting, the shifting stops and all subsequent nodes remain unchanged. Once the shake step is completed, all affected nodes will be updated by a process similar to the one used after the insertion step. Once again, nodes before the shaking sequence need only have their MaxShift updated. In the end, a gap that minimizes waiting time will have been introduced to the tour which will allow the insertion of more profitable nodes than the ones removed.

The following figure presents the pseudocode for the shake step:

For each tour:
    Remove nodes between i and j inclusive;
    Calculate Shift;
    For each node after j:
        Shift node towards starting node;
        Update Arrival, Start, Shift, MaxShift, Wait;

```
For each node before i:
    Update MaxShift;
```

3.3.3 The Heuristic

The ILS heuristic combines the two previously mentioned steps in the following way:

Since the problem at hand is TOPTW, we begin with a set of empty tours sized from 1 to m. The two parameters of the shake step (R_p, S_p) are initialized to 1. The heuristic records the best found solution after every iteration and will loop until no improvement appears after a given number of times. Every iteration begins with an insertion step that will loop until a so called local optimum is reached. If the current tour is more profitable than the so far best solution recorded, it replaces it as best incumbent solutions. In either case, the heuristic will move on to the next step, shake. After every shake S_p is increased by the current R_p, while R_p itself is increased by 1. The shake step will then be performed with these parameters as input. S_p and R_p have been given upper limits through testing the algorithm until the best solutions were produced. Specifically, S_p cannot be more than the size of the smallest tour in the set, at which point this size will be subtracted from it in order to continue the process. On the other hand, if R_p reaches $\frac{n}{3*m}$ where n= number of nodes and m=numbers of tours to be created, it will reset to 1. The only predetermined decisions then are the number of nodes available for consideration, the number of tours to be created and the number of iterations for the heuristic, after which no further improvement is needed. The latter is of course the result of the balance to be decided between the computation time and the quality of the produced solution. Through testing, the number that seems to be soft cap regarding the quality of the solution without adding unnecessary computation time is 150.

The pseudocode for the heuristic is the following:

```
S_p=1;
R_p=1;
NumberOfIterations=0;
while NumberOfIterations<150:
    while localOptimum not reached:
        Perform InsertionStep;
    if currentSolution better than bestFoundSolution:
        bestFoundSolution= currentSolution;
        R_p=1;
        NumberOfIterations=0;
    else:
        NumberOfIterations = NumberOfIterations + 1;
    Perform Shake (S_p, R_p);
    S_p = S_p+ R_p;
    R_p = R_p+ 1;
    if S_p >= Size of smallest tour in set:
        S_p = S_p - Size of smallest tour in set;
    if R_p== n/(3*m):
        R_p= 1;
return bestFoundSolution;
```

3.3.4 Visit time modification

In the original ILS context, a visit is considered to take place if the arrival to one node happens anytime before the node's closing time. Thus, the visit's proposed duration is not always respected; a visit that lasts just one second is acceptable. Our approach is much stricter in this regard: a visit's proposed duration is always enforced. As a consequence we can expect lower total scores than the original ILS's results, since some of our tours will possibly contain fewer visits in order to accommodate our much stricter criteria.

3.3.5 Point-of-interest categorization and user choice

Another modification we introduce is the classification of POI's in categories based on their particular touristic purpose. For example, a POI can be classified as a museum, a theater, an open space and so on. The user is then given the choice to control the relative value each category has to them. Specifically, if a particular user has only a passing interest in museums, he can choose to reduce the proposed score of subsequent museum visits after a certain limit of already included museums in the designed tour is reached. Conversely, if a user is specifically interested in sights in open spaces, he can choose not to reduce the proposed score of subsequent open spaces visits no matter how many such visits are already planned. In summation, this modification allows the user to favor POI's of particular interest to him, while limiting those he is indifferent to, while maintaining the highest overall score of the whole tour possible.

4. Cluster based Heuristics for the Team Orienteering Problem with Time Windows

4.1 Introduction

Gavalas et al.[30] added another dimension to the TOPTW rooted in the ILS solving procedure: the division of the set of available nodes in clusters based on geographical criteria. The reasoning behind this addition is that ILS, during the evaluation of candidate nodes for insertion, will disregard high profit areas of nodes if these are far from the current solution in geographical terms. This of course happens because the ratio takes into account the time consumption of each visit through Shift, even though there is an attempt to mitigate this by squaring the score to place more emphasis in it. However, the problem still persists because ILS is evaluating each node individually. The proposed solution is to cluster nearby nodes creating profitable areas to increase those nodes attractiveness. The main idea is that if a high profit node is visited it's nearby nodes can also be visited without significant additional travel time. Having the concept of the Tourist Trip Design Problem in mind, this can mean that these nodes can be reached by walking, a highly desirable trait for a tourist.

There are two algorithms developed to handle this process, CSCRatio and CSCRoutes, both based on ILS. Both algorithms use the same procedure to form the clusters of nodes by employing the global k-means algorithm developed by Likas et al.[31], through which empty clusters are initialized during a preprocessing phase. Afterwards, in a phase common to both algorithms called RouteInitPhase the m requested routes are each assigned exactly one node from the clusters. Since it is reasonable that the number of clusters will be greater than the number of routes, a decision must be reached over which clusters will provide the routes with nodes during this step. One approach would be to simply rank the clusters formed on total profit and pick the best ones. A more optimal one would be to be flexible at this stage and try various combinations of the best clusters and stick with the ones giving better solutions at the algorithm stage. Whatever the approach decided, RouteInitPhase takes as argument a m number of clusters from the listOfClusterSet and after finding the node with the highest score in it, delegates it to one of the m requested routes-tours. By doing this, the process ensures that various geographical areas from the set of available nodes will be represented and avoid getting trapped in high-scoring local nodes. The process then continues, much like ILS, with an insertion and a shake step, either by employing the CSCRatio or CCSCRoutes algorithm. Our stricter criterion of respecting a visit's proposed duration are also applied here.

4.2 Cluster Search Cluster Ratio

The insertion step of the CSCRatio algorithm has additional argument in a parameter called clusterParameter $\geq$ 1. ClusterParameter represents the emphasis decided to be given on the clustering of the nodes. The greater it is, the higher the chance a visit to a node will be accompanied by a visit to another node of the same cluster. The way this emphasis is introduced in the ILS is by creating a modified version of $Shift_i$ called $shiftCluster_i$ which is calculated by $\frac{Shift_i}{\text{clusterParameter}}$. When clusterParameter is 1, $shiftCluster_i$ equals $Shift_i$, in which case we have the standard case of ILS. However, as clusterParameter increases the time consumption of a visit to a node in the same cluster of the current one decreases, making it more likely to be chosen compared to a node of a different cluster. This happens because the evaluation ratio is now given by the following type: $Ratio_i = \frac{Score_i^2}{shiftCluster_i}$. CSCRatio begins with the clusterParameter set at 1.3 and gradually decreases it by 0.1 at each quarter of the total iterations it will go through. So, in the beginning a much greater emphasis is placed on visiting nodes within the same cluster and as the iterations progress this emphasis declines until the cluster play no role in evaluating nodes at the last quarter of iterations. A balance then is reached between the benefits of visiting inside a cluster and the diversification that ILS provides.

The shake step in the CSCRatio algorithm is very much like the respective step of ILS; a modification is made regarding the number of nodes that are to be moved. Specifically, in CSCRatio R_p is limited to half the size of the largest tour in the solution and not $\frac{n}{3*m}$ which reduces computation time since the local optimum is reached faster than in the ILS having a smaller portion of the solution removed at each iteration. This reduction in computation time allows more iterations of the heuristic to be completed without taking more total execution time than the ILS.

The pseudocode of CSCRatio is given in the figure below:

Perform k-means algorithm : intiliaze k amount of clusters
Construct the listOfClusterSets
it1= $\frac{maxIterations}{4}$*;*
it2= $\frac{2*maxIterations}{4}$*;*
it3= $\frac{3*maxIterations}{4}$*;*

while listOfClusterSets not empty:
  remove all nodes included in currentSolution;
  theClusterSetIdToInsert= listOfClusterSets.pop();

  RoutInitPhase(theClusterSetIdToInsert)
  S_p*=1;*

```
Rp=1;
NumberOfIterations=0;
while NumberOfIterations < maxIterations:
    if NumberOfIterations <it2:
        if NumberOfIterations <it1:
            clusterParameter = 1.3;
        else:
            clusterParameter = 1.2;
    else:
        if NumberOfIterations <it3:
            clusterParameter = 1.1;
        else:
            clusterParameter = 1.0;

    while localOptimum not reached:
        CSCRatio_Insertion (clusterParameter);
    if currentSolution better than bestFoundSolution:
        bestFoundSolution= currentSolution;
        Rp=1;
        NumberOfIterations=0;
    else:
        NumberOfIterations = NumberOfIterations + 1;
    if Rp > currentSolution.sizeOfLargestTour / 2:
        Rp=1;
    Perform Shake (Sp, Rp);
    Sp = Sp+ Rp;
    Rp = Rp+ 1;
    if Sp >= Size of smallest tour in set:
        Sp = Sp - Size of smallest tour in set;
    if Rp== n / (3*m):
        Rp= 1;
Return bestFoundSolution;
```

4.3 Cluster Search Cluster Routes algorithm

Within a tour p in a proposed solution of TOPTW we define a sub-tour of consecutive nodes that belong in the same cluster as a Cluster Route (CR) of p associated with cluster C and denoted as CR_C^p. CR_C^p is greater than 1 and less than or equal to the size of cluste r C. A key difference of CSCRoutes from the CSCRatio is that it doesn't allow to return to a previously visited cluster, with the exception of the starting and ending node being in the same cluster. In that case, a reentry to the cluster is allowed. This difference means that for a tour p there can only be one CR_C^p sub-tour and also that a node can't just be entered at any position in the tour; it has to be in proximity to the other nodes of the same cluster. This restriction will lead to lower-quality solutions compared to the ILS and CSCRatio but it will also take significant less execution time because the evaluation needed at each iteration will be a lot less in CSCRoutes insertion step.

The pseudocode of CSCSRoutes is presented in the following figure:

```
Perform k-means algorithm : intiliaze k amount of clusters
                            Construct the listOfClusterSets
```

```
while listOfClusterSets not empty:
    remove all nodes included in currentSolution;
    theClusterSetIdToInsert= listOfClusterSets.pop();

    RoutInitPhase(theClusterSetIdToInsert)
    S_p=1;
    R_p=1;
    NumberOfIterations=0;
    while NumberOfIterations < maxIterations:


        while localOptimum not reached:
            CSCRoutes_Insert;
        if currentSolution better than bestFoundSolution:
            bestFoundSolution= currentSolution;
            R_p=1;
            NumberOfIterations=0;
        else:
            NumberOfIterations = NumberOfIterations + 1;
        if R_p > currentSolution.sizeOfLargestTour / 2:
            R_p=1;
        Perform Shake (S_p, R_p);
        S_p = S_p+ R_p;
        R_p = R_p+ 1;
        if S_p >= Size of smallest tour in solution:
            S_p = S_p - Size of smallest tour in solution;
        if R_p== n / (3*m):
            R_p= 1;
    Return bestFoundSolution
```

5 Experimental results

5.1 Test Instances

The modified ILS, CSCRatio and CSCRoutes algorithms are tested on the widely used datasets of Solomon. All data sets have 100 possible nodes to be visited and a fixed proposed visit duration for each node. It should be noted that these data sets are a particular fit for TOPTW problems; they are not suited for algorithms designed to solve TOP cases.

5.2 Results

All computations are carried out on a personal portable computer with a Intel Core 2 Duo CPU @ 2.0GHz with 3.0GB Ram. These specs are actually lower than the ones used to run the original ILS and so meaningful comparison can be made regarding the computation time recorded.

Four sizes of tours for each instance were examined (m=1,2,3,4) to keep the tests directly related to the original ILS and also because in view of the Tourist Trip Design Problem, these sizes are contextually valid. Tables 1-4 present the results of our ILS variant with the stricter constraints on

POI visits (ILS_V.1) and contrast them with the original ILS. Tables 5-8 contrast our modified ILS with the accordingly modified version of the CSCRatio algorithm to examine possible benefits from employing clustering heuristics, while tables 9-12 do the same with the modified CSCRoutes algorithm.

Tables 13-16 display the impact the POI classification and the on-the-fly POI profit adjustments (ILS_V.2) have on our modified ILS. Our tighter constraints from ILS_V.1 regarding the visit time are respected here as well. The results are based on an classification in 3 groups with the following parameters:

a. First group: POI's belonging to this group have a 40% reduction in their scores if more than 1/3 of the nodes already included in that tour belong to it
b. Second group: POI's belonging to this group have a 70% reduction in their scores if more than 1/3 of the nodes already included in that tour belong to it
c. Third group: POI's belonging to this group have a 10% reduction in their scores if more than 1/3 of the nodes already included in that tour belong to it

The approach selected clearly rewards POI's of the third group while "punishing" those in the first group. All of the parameters selected are arbitrary and chosen randomly, while the user is free to change them at will.

As expected and demonstrated in the following tables, our modified ILS is slightly underperforming the original ILS due to the enforcing of much stricter criteria regarding the completion of each node visit in its suggested duration. The difference is on average a 1-10% reduction in the score of each tour. It is a behavior observed when we move to the respective comparisons of the CSCRatio and CSCSRoutes algorithms.

The variant of ILS considering POI categorization and dynamic POI profits has very good results when the number of tours is less than 3. As that number increases, the algorithm struggles to find quality nodes after the score reduction and the overall tour score drops significantly.

Table 1 – Comparison between Original ILS and ILS_V.1 for m=1

	Original ILS			ILS_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	320	10	0.4	300	10	0.04700
c102	360	11	0.3	320	11	0.07800
c103	390	10	0.5	380	11	0.07800
c104	400	10	0.3	390	11	0.06300
c105	340	10	0.3	310	9	0.04700
c106	340	10	0.3	310	10	0.04800
c107	360	11	0.3	320	10	0.06200
c108	370	11	0.3	320	10	0.06300
c109	380	11	0.3	340	11	0.06200
r101	182	7	0.1	186	8	0.09400
r102	286	11	0.2	247	10	0.06300
r103	286	10	0.2	252	10	0.09300
r104	297	11	0.2	268	11	0.07800
r105	247	11	0.1	215	9	0.06300
r106	293	11	0.2	258	10	0.14100
r107	288	10	0.2	243	11	0.12500
r108	297	11	0.2	233	11	0.07900
r109	276	11	0.2	264	11	0.09400
r110	281	11	0.3	247	11	0.07800
r111	295	11	0.2	276	11	0.09400
r112	295	11	0.2	274	11	0.07800
rc101	219	9	0.2	203	8	0.06200
rc102	259	9	0.2	245	10	0.06300
rc103	265	11	0.3	245	10	0.06300
rc104	297	11	0.3	240	10	0.05500
rc105	221	11	0.2	162	7	0.06300
rc106	239	11	0.2	200	8	0.04700
rc107	274	11	0.2	240	10	0.06300
rc108	288	11	0.2	240	10	0.06200

Table 2 – Comparison between Original ILS and ILS_V.1 for m=2

	Original ILS			ILS_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	590	21	1.4	540	18	0.18700
c102	650	22	0.9	610	21	0.32200
c103	700	22	1.2	660	22	0.12500
c104	750	22	1.5	700	22	0.22200
c105	640	21	0.8	540	18	0.12500
c106	320	20	0.8	550	18	0.18800
c107	670	22	1.4	570	18	0.12500
c108	670	22	0.8	570	19	0.14100
c109	710	22	0.9	640	20	0.18700
r101	330	13	0.4	307	13	0.14000
r102	508	21	0.9	455	18	0.18800
r103	513	20	0.9	450	19	0.15600
r104	539	22	1.5	483	21	0.15600
r105	430	18	0.8	369	16	0.12500
r106	529	21	0.9	438	19	0.10900
r107	529	21	1	483	20	0.14000
r108	549	24	1.4	486	21	0.13600
r109	498	22	0.5	412	18	0.10900
r110	515	22	1	435	19	0.14100
r111	535	23	0.6	471	20	0.12500
r112	515	21	0.5	450	20	0.12900
rc101	427	19	0.6	293	11	0.09400
rc102	494	20	0.8	326	14	0.10900
rc103	519	20	1.1	394	16	0.23400
rc104	565	22	0.7	459	18	0.14100
rc105	459	22	0.8	289	12	0.15600
rc106	458	20	0.6	377	15	0.12500
rc107	515	21	0.5	436	18	0.10900
rc108	546	23	0.6	464	19	0.12500

Table 3 – Comparison between Original ILS and ILS_V.1 for m=3

	Original ILS			ILS_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	790	29	1.1	730	25	0.21900
c102	890	32	2.1	800	29	0.37800
c103	960	33	2.2	920	32	0.20300
c104	1010	34	1.3	950	33	0.21800
c105	840	30	1	780	26	0.20400
c106	840	30	1.1	780	26	0.22500
c107	900	33	1.5	760	25	0.19400
c108	900	33	1.2	820	27	0.25000
c109	950	33	2	860	28	0.20300
r101	481	21	0.8	415	18	0.20300
r102	685	31	1	606	25	0.25000
r103	720	31	2	630	27	0.17200
r104	765	34	1.5	662	29	0.17200
r105	609	27	2.3	519	22	0.16600
r106	719	32	2.1	604	26	0.28900
r107	747	33	1.1	611	26	0.21600
r108	790	36	3.1	691	31	0.21700
r109	699	31	1.8	580	24	0.24600
r110	711	32	1.4	618	26	0.25000
r111	764	34	1.8	654	28	0.26500
r112	758	34	1.1	665	29	0.21900
rc101	604	29	1.4	448	17	0.18700
rc102	698	30	1.3	486	20	0.18800
rc103	747	30	1.1	588	22	0.21800
rc104	822	33	1.3	690	27	0.18500
rc105	654	28	0.8	422	17	0.25000
rc106	678	31	1	546	21	0.18800
rc107	745	31	0.9	602	24	0.17200
rc108	757	29	1.1	648	26	0.20500

Table 4 – Comparison between Original ILS and ILS_V.1 for m=4

	Original ILS			ILS_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	1000	39	3.8	910	31	0.31700
c102	1090	43	1.8	980	35	0.37500
c103	1150	44	2.5	1080	39	0.23500
c104	1220	45	3	1160	42	0.35900
c105	1030	40	1.8	950	33	0.25800
c106	1040	40	2.1	940	32	0.25000
c107	1100	43	2	950	33	0.31300
c108	1100	44	3.6	970	34	0.26600
c109	1180	45	2.5	1030	36	0.51500
r101	601	28	1.4	507	22	0.23500
r102	807	39	1.7	687	29	0.26600
r103	878	42	2.2	782	33	0.37500
r104	941	45	3.8	808	36	0.26500
r105	735	35	2.9	634	27	0.28200
r106	870	41	3.5	725	31	0.28100
r107	927	44	3.3	765	34	0.23000
r108	982	47	3.2	865	38	0.23500
r109	866	40	2.1	763	33	0.27700
r110	870	42	2	768	34	0.48400
r111	935	45	2	805	35	0.21900
r112	939	44	3.1	830	37	0.23500
rc101	794	37	1.9	598	23	0.28100
rc102	881	42	2.3	656	27	0.31300
rc103	947	42	2	777	30	0.25000
rc104	1019	43	1.7	833	33	0.34100
rc105	841	37	1.5	601	23	0.32800
rc106	874	37	2.5	728	28	0.28100
rc107	951	42	1.9	773	30	0.26700
rc108	998	43	2	837	33	0.23400

Table 5 – Comparison between ILS_V.1 and CSCRatio_V.1 for m=1

	ILS_V.1			CSCRatio_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	300	10	0.047	300	10	1.885
c102	320	11	0.078	330	11	2.745
c103	380	11	0.078	390	11	2.915
c104	390	11	0.063	390	11	2.811
c105	310	9	0.047	320	10	2.54
c106	310	10	0.048	310	10	2.56
c107	320	10	0.062	300	10	3.405
c108	320	10	0.063	320	10	2.903
c109	340	11	0.062	350	10	4.256
r101	186	8	0.094	142	7	0.186
r102	247	10	0.063	246	10	0.271
r103	252	10	0.093	268	11	0.297
r104	268	11	0.078	266	12	0.337
r105	215	9	0.063	159	7	0.261
r106	258	10	0.141	258	10	0.371
r107	243	11	0.125	267	11	0.475
r108	233	11	0.079	265	12	0.515
r109	264	11	0.094	238	10	0.581
r110	247	11	0.078	243	11	0.495
r111	276	11	0.094	272	11	0.68
r112	274	11	0.078	262	12	0.701
rc101	203	8	0.062	180	7	2.453
rc102	245	10	0.063	222	9	4.156
rc103	245	10	0.063	222	9	4.234
rc104	240	10	0.055	243	10	5.109
rc105	162	7	0.063	162	7	2.75
rc106	200	8	0.047	200	8	4.219
rc107	240	10	0.063	220	9	4.781
rc108	240	10	0.062	240	10	4.375

Table 6 – Comparison between ILS_V.1 and CSCRatio_V.1 for m=2

	ILS_V.1			CSCRatio_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	540	18	0.187	540	18	9.247
c102	610	21	0.322	610	21	13.097
c103	660	22	0.125	650	22	10.513
c104	700	22	0.222	700	23	9.516
c105	540	18	0.125	520	18	6.328
c106	550	18	0.188	550	18	6.273
c107	570	18	0.125	540	18	6.781
c108	570	19	0.141	560	19	7.187
c109	640	20	0.187	620	21	8.408
r101	307	13	0.14	251	11	1.139
r102	455	18	0.188	419	18	1.706
r103	450	19	0.156	463	19	2.077
r104	483	21	0.156	507	22	2.391
r105	369	16	0.125	362	15	1.711
r106	438	19	0.109	429	18	2.136
r107	483	20	0.14	473	20	3.085
r108	486	21	0.136	521	23	3.918
r109	412	18	0.109	411	18	3.374
r110	435	19	0.141	419	19	3.528
r111	471	20	0.125	462	20	4.176
r112	450	20	0.129	485	20	4.359
rc101	293	11	0.094	344	13	6.36
rc102	326	14	0.109	385	16	8.094
rc103	394	16	0.234	440	17	8.906
rc104	459	18	0.141	478	19	10.641
rc105	289	12	0.156	369	15	11.625
rc106	377	15	0.125	370	15	12.841
rc107	436	18	0.109	429	18	11.797
rc108	464	19	0.125	447	17	10.547

Table 7 – Comparison between ILS_V.1 and CSCRatio_V.1 for m=3

	ILS_V.1			CSCRatio_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	730	25	0.219	730	25	10.564
c102	800	29	0.378	780	28	22.648
c103	920	32	0.203	920	32	21.702
c104	950	33	0.218	920	32	19.339
c105	780	26	0.204	780	26	21.529
c106	780	26	0.225	780	26	17.257
c107	760	25	0.194	810	27	22.382
c108	820	27	0.25	760	26	14.899
c109	860	28	0.203	840	28	20.18
r101	415	18	0.203	386	17	4.481
r102	606	25	0.25	563	24	7.035
r103	630	27	0.172	642	27	6.458
r104	662	29	0.172	661	29	6.858
r105	519	22	0.166	494	22	5.134
r106	604	26	0.289	598	25	5.906
r107	611	26	0.216	642	28	6.729
r108	691	31	0.217	707	30	7.546
r109	580	24	0.246	601	25	9.655
r110	618	26	0.25	610	27	13.391
r111	654	28	0.265	695	30	12.82
r112	665	29	0.219	630	29	14.068
rc101	448	17	0.187	500	19	13
rc102	486	20	0.188	546	22	15.687
rc103	588	22	0.218	632	26	19.547
rc104	690	27	0.185	688	27	24.438
rc105	422	17	0.25	483	20	13.89
rc106	546	21	0.188	528	21	16.203
rc107	602	24	0.172	602	24	18.625
rc108	648	26	0.205	678	27	23.343

Table 8 – Comparison between ILS_V.1 and CSCRatio_V.1 for m=4

	ILS_V.1			CSCRatio_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	910	31	0.317	870	31	20.144
c102	980	35	0.375	910	33	20.698
c103	1080	39	0.235	1070	39	24.596
c104	1160	42	0.359	1150	42	26.85
c105	950	33	0.258	900	32	34.904
c106	940	32	0.25	910	31	30.429
c107	950	33	0.313	920	32	22.453
c108	970	34	0.266	950	33	22.625
c109	1030	36	0.515	1000	35	24.937
r101	507	22	0.235	458	22	9.766
r102	687	29	0.266	663	28	9.369
r103	782	33	0.375	708	31	11.163
r104	808	36	0.265	804	35	12.487
r105	634	27	0.282	653	28	9.685
r106	725	31	0.281	727	32	18.817
r107	765	34	0.23	764	33	16.447
r108	865	38	0.235	899	39	16.058
r109	763	33	0.277	747	32	14.724
r110	768	34	0.484	752	33	18.671
r111	805	35	0.219	844	37	15.625
r112	830	37	0.235	814	36	19.068
rc101	598	23	0.281	619	23	18.546
rc102	656	27	0.313	736	30	29.828
rc103	777	30	0.25	832	32	28.344
rc104	833	33	0.341	869	34	29.531
rc105	601	23	0.328	621	24	23.125
rc106	728	28	0.281	711	27	26.844
rc107	773	30	0.267	813	31	33.64
rc108	837	33	0.234	838	33	30

Table 9 – Comparison between ILS_V.1 and CSCRoutes_V.1 for m=1

	ILS_V.1			CSCRoutes_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	300	10	0.047	250	9	0.281
c102	320	11	0.078	340	11	0.281
c103	380	11	0.078	350	11	0.359
c104	390	11	0.063	390	11	0.313
c105	310	9	0.047	300	9	0.25
c106	310	10	0.048	290	9	0.25
c107	320	10	0.062	310	10	0.25
c108	320	10	0.063	350	11	0.266
c109	340	11	0.062	340	10	0.437
r101	186	8	0.094	131	5	0.36
r102	247	10	0.063	275	10	0.359
r103	252	10	0.093	263	11	0.391
r104	268	11	0.078	288	12	0.391
r105	215	9	0.063	177	7	0.25
r106	258	10	0.141	279	11	0.546
r107	243	11	0.125	263	11	0.343
r108	233	11	0.079	265	11	0.375
r109	264	11	0.094	234	10	0.359
r110	247	11	0.078	225	10	0.265
r111	276	11	0.094	243	10	0.157
r112	274	11	0.078	253	11	0.266
rc101	203	8	0.062	189	7	0.172
rc102	245	10	0.063	222	9	0.281
rc103	245	10	0.063	222	9	0.25
rc104	240	10	0.055	229	9	0.313
rc105	162	7	0.063	173	7	0.094
rc106	200	8	0.047	174	7	0.219
rc107	240	10	0.063	231	9	0.22
rc108	240	10	0.062	271	10	0.234

Table 10 – Comparison between ILS_V.1 and CSCRoutes_V.1 for m=2

	ILS_V.1			CSCRoutes_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	540	18	0.187	470	16	0.407
c102	610	21	0.322	410	16	0.141
c103	660	22	0.125	630	22	0.406
c104	700	22	0.222	670	22	0.609
c105	540	18	0.125	510	17	0.391
c106	550	18	0.188	520	18	0.39
c107	570	18	0.125	540	18	0.453
c108	570	19	0.141	540	19	0.328
c109	640	20	0.187	600	21	0.391
r101	307	13	0.14	239	10	0.218
r102	455	18	0.188	439	18	0.313
r103	450	19	0.156	465	19	0.329
r104	483	21	0.156	499	21	0.422
r105	369	16	0.125	298	13	0.218
r106	438	19	0.109	428	19	0.391
r107	483	20	0.14	455	20	0.328
r108	486	21	0.136	499	21	0.578
r109	412	18	0.109	403	17	0.437
r110	435	19	0.141	437	19	0.328
r111	471	20	0.125	487	21	0.297
r112	450	20	0.129	491	21	0.375
rc101	293	11	0.094	311	12	0.187
rc102	326	14	0.109	350	14	0.266
rc103	394	16	0.234	423	17	0.438
rc104	459	18	0.141	448	18	0.313
rc105	289	12	0.156	352	14	0.234
rc106	377	15	0.125	371	15	0.266
rc107	436	18	0.109	430	17	0.297
rc108	464	19	0.125	447	18	0.328

Table 11 – Comparison between ILS_V.1 and CSCRoutes_V.1 for m=3

	ILS_V.1			CSCRoutes_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	730	25	0.219	660	23	0.359
c102	800	29	0.378	760	28	0.5
c103	920	32	0.203	880	31	0.562
c104	950	33	0.218	950	33	0.844
c105	780	26	0.204	710	24	0.531
c106	780	26	0.225	660	24	0.641
c107	760	25	0.194	770	26	0.5
c108	820	27	0.25	770	25	0.532
c109	860	28	0.203	870	29	0.5
r101	415	18	0.203	347	14	0.171
r102	606	25	0.25	606	25	0.563
r103	630	27	0.172	645	28	0.546
r104	662	29	0.172	688	30	0.781
r105	519	22	0.166	490	21	0.312
r106	604	26	0.289	585	24	0.547
r107	611	26	0.216	651	28	0.453
r108	691	31	0.217	697	30	0.734
r109	580	24	0.246	598	25	0.61
r110	618	26	0.25	625	27	0.641
r111	654	28	0.265	661	28	0.469
r112	665	29	0.219	697	30	0.719
rc101	448	17	0.187	476	18	0.406
rc102	486	20	0.188	534	21	0.453
rc103	588	22	0.218	614	24	0.407
rc104	690	27	0.185	649	26	0.375
rc105	422	17	0.25	502	20	0.328
rc106	546	21	0.188	523	20	0.25
rc107	602	24	0.172	632	25	0.343
rc108	648	26	0.205	623	24	0.453

Table 12 – Comparison between ILS_V.1 and CSCRoutes_V.1 for m=4

	ILS_V.1			CSCRoutes_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	910	31	0.317	860	30	0.312
c102	980	35	0.375	930	34	0.469
c103	1080	39	0.235	1050	38	0.531
c104	1160	42	0.359	1150	42	0.765
c105	950	33	0.258	860	30	0.766
c106	940	32	0.25	880	32	0.469
c107	950	33	0.313	950	33	0.75
c108	970	34	0.266	930	33	0.765
c109	1030	36	0.515	980	35	0.766
r101	507	22	0.235	421	18	0.171
r102	687	29	0.266	643	27	0.563
r103	782	33	0.375	595	26	0.359
r104	808	36	0.265	799	35	0.578
r105	634	27	0.282	620	26	0.328
r106	725	31	0.281	730	31	0.547
r107	765	34	0.23	728	33	0.532
r108	865	38	0.235	829	36	0.516
r109	763	33	0.277	711	30	0.594
r110	768	34	0.484	726	33	0.64
r111	805	35	0.219	833	36	0.516
r112	830	37	0.235	818	37	0.453
rc101	598	23	0.281	540	21	0.297
rc102	656	27	0.313	649	25	0.36
rc103	777	30	0.25	749	29	0.391
rc104	833	33	0.341	844	33	0.391
rc105	601	23	0.328	624	24	0.313
rc106	728	28	0.281	732	28	0.438
rc107	773	30	0.267	798	31	0.484
rc108	837	33	0.234	820	32	0.453

Table 13 – Comparison between ILS_V.2 and ILS_V.1 for m=1

	ILS_V.2			ILS_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	288	9	0.09	300	10	0.04700
c102	300	10	0.11	320	11	0.07800
c103	370	11	0.11	380	11	0.07800
c104	390	11	0.11	390	11	0.06300
c105	290	9	0.09	310	9	0.04700
c106	280	8	0.15	310	10	0.04800
c107	300	9	0.17	320	10	0.06200
c108	300	9	0.09	320	10	0.06300
c109	340	10	0.11	340	11	0.06200
r101	168	7	0.19	186	8	0.09400
r102	244	10	0.3	247	10	0.06300
r103	191	9	0.2	252	10	0.09300
r104	203	9	0.24	268	11	0.07800
r105	202	8	0.12	215	9	0.06300
r106	217	10	0.29	258	10	0.14100
r107	207	9	0.13	243	11	0.12500
r108	192	10	0.18	233	11	0.07900
r109	192	10	0.17	264	11	0.09400
r110	210	9	0.14	247	11	0.07800
r111	239	9	0.2	276	11	0.09400
r112	223	10	0.28	274	11	0.07800
rc101	151	7	0.06	203	8	0.06200
rc102	200	9	0.09	245	10	0.06300
rc103	200	9	0.09	245	10	0.06300
rc104	195	9	0.09	240	10	0.05500
rc105	138	7	0.09	162	7	0.06300
rc106	200	8	0.1	200	8	0.04700
rc107	199	10	0.1	240	10	0.06300
rc108	191	10	0.1	240	10	0.06200

Table 14 – Comparison between ILS_V.2 and ILS_V.1 for m=2

	ILS_V.2			ILS_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	432	15	0.17	540	18	0.18700
c102	456	17	0.2	610	21	0.32200
c103	610	21	0.23	660	22	0.12500
c104	583	21	0.26	700	22	0.22200
c105	480	15	0.31	540	18	0.12500
c106	500	16	0.29	550	18	0.18800
c107	500	16	0.31	570	18	0.12500
c108	457	16	0.24	570	19	0.14100
c109	550	18	0.27	640	20	0.18700
r101	250	10	0.29	307	13	0.14000
r102	321	15	0.31	455	18	0.18800
r103	322	16	0.31	450	19	0.15600
r104	395	17	0.32	483	21	0.15600
r105	311	14	0.2	369	16	0.12500
r106	330	15	0.23	438	19	0.10900
r107	322	16	0.25	483	20	0.14000
r108	352	18	0.21	486	21	0.13600
r109	331	18	0.37	412	18	0.10900
r110	354	16	0.31	435	19	0.14100
r111	367	16	0.23	471	20	0.12500
r112	354	17	0.18	450	20	0.12900
rc101	251	12	0.35	293	11	0.09400
rc102	270	14	0.29	326	14	0.10900
rc103	279	15	0.42	394	16	0.23400
rc104	313	17	0.64	459	18	0.14100
rc105	268	11	0.39	289	12	0.15600
rc106	308	14	0.73	377	15	0.12500
rc107	306	16	0.71	436	18	0.10900
rc108	304	16	0.62	464	19	0.12500

Table 15 – Comparison between ILS_V.2 and ILS_V.1 for m=3

	ILS_V.2			ILS_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	474	21	0.46	730	25	0.21900
c102	538	23	1.12	800	29	0.37800
c103	760	27	0.39	920	32	0.20300
c104	721	30	0.38	950	33	0.21800
c105	630	21	0.34	780	26	0.20400
c106	617	23	0.39	780	26	0.22500
c107	670	23	0.36	760	25	0.19400
c108	522	22	0.33	820	27	0.25000
c109	676	25	0.79	860	28	0.20300
r101	324	16	0.34	415	18	0.20300
r102	393	19	0.34	606	25	0.25000
r103	437	23	0.4	630	27	0.17200
r104	510	25	0.51	662	29	0.17200
r105	388	20	0.33	519	22	0.16600
r106	412	21	0.55	604	26	0.28900
r107	452	23	0.32	611	26	0.21600
r108	470	25	0.3	691	31	0.21700
r109	437	23	0.31	580	24	0.24600
r110	406	21	0.39	618	26	0.25000
r111	465	23	0.38	654	28	0.26500
r112	452	23	0.31	665	29	0.21900
rc101	326	17	0.48	448	17	0.18700
rc102	349	18	0.68	486	20	0.18800
rc103	306	20	0.35	588	22	0.21800
rc104	324	21	0.49	690	27	0.18500
rc105	338	16	0.41	422	17	0.25000
rc106	372	19	0.46	546	21	0.18800
rc107	333	22	0.42	602	24	0.17200
rc108	317	22	0.39	648	26	0.20500

Table 16 – Comparison between ILS_V.2 and ILS_V.1 for m=4

	ILS_V.2			ILS_V.1		
Name	Score	Visits	Comp. time	Score	Visits	Comp. time
c101	549	23	0.67	910	31	0.31700
c102	555	28	0.34	980	35	0.37500
c103	829	31	0.36	1080	39	0.23500
c104	728	35	0.26	1160	42	0.35900
c105	740	27	0.39	950	33	0.25800
c106	664	28	0.34	940	32	0.25000
c107	759	28	0.6	950	33	0.31300
c108	541	27	0.34	970	34	0.26600
c109	705	32	0.31	1030	36	0.51500
r101	401	20	0.38	507	22	0.23500
r102	427	22	0.52	687	29	0.26600
r103	475	26	0.29	782	33	0.37500
r104	480	28	0.28	808	36	0.26500
r105	470	24	0.32	634	27	0.28200
r106	473	25	0.31	725	31	0.28100
r107	525	30	0.28	765	34	0.23000
r108	529	29	0.26	865	38	0.23500
r109	453	28	0.28	763	33	0.27700
r110	508	27	0.29	768	34	0.48400
r111	456	29	0.31	805	35	0.21900
r112	477	28	0.25	830	37	0.23500
rc101	402	22	0.51	598	23	0.28100
rc102	372	22	0.65	656	27	0.31300
rc103	318	25	0.61	777	30	0.25000
rc104	327	25	0.56	833	33	0.34100
rc105	370	20	0.46	601	23	0.32800
rc106	417	23	0.46	728	28	0.28100
rc107	346	27	0.74	773	30	0.26700
rc108	322	27	0.83	837	33	0.23400

6. Conclusions

The first most obvious contribution of the thesis is the inclusion of the realistic “requirement” to spend a minimum amount of time at a point of interest in order for it to be considered “visited”. The time proposed is tailored for each POI specifically and can even be modified according to the user’s preference, making it a quite valuable feature for any tourist trip planner.

The second meaningful contribution also involves taking into account user preferences: the user is invited to place emphasis on a particular category of POI’s, to avoid another one or potentially choose a balanced approach. This kind of features really applies a practical perspective to ILS.

As far as performance is concerned, our implementation of ILS is very close to the original. Naturally, the features we introduced impose powerful constraints to the performance of the algorithm, which is still very much within acceptable limits from a practical point of view.

7. References

[1] Tsiligirides T. Heuristic methods applied to orienteering ,Journal of the Operational Research Society1984;35:797–809

[2] Golden B, Levy L ,Vohra R The orienteering problem, Naval Research Logistics 1987;34:307–18.

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

[4] Feillet, D. , Dejax, P. , & Gendreau, M. (2005). Traveling salesman problems with prof- its. Transportation Science, 39 (2), 188–205

[5] Laporte, G. , & Rodríguez-Martín, I. (2007). Locating a cycle in a transportation or a telecommunications network. Networks, 50 (1), 92–108

[6] Orienteering Problem: A survey of recent variants, solution approaches and applications
Aldy Gunawan a , ∗, Hoong Chuin Lau a , Pieter Vansteenwegen b

[7] Miller, C., Tucker, A., Zemlin, R., 1960 Integer Programming Formulation of Traveling Salesman Problems, Journal of the ACM (JACM), Volume 7 Issue 4, Oct. 1960 ,Pages 326-329

[8] GoldenB, LevyL, VohraR, The orienteering problem, Naval Research Logistics 1987;34:307–18.

[9] Laporte G, Martello S, The selective travelling salesman problem, Discrete Applied Mathematics 1990;26:193–207.

[10] Leifer and Rosenwein, Strong linear programming relaxations for the orienteering problem, European Journal of Operational Research, 1994, vol. 73, issue 3, 517-523

[11] Fischetti et al, Solving the Orienteering Problem through Branch-and-Cut, Informs Journal on Computing 10(2):133-148 · May 1998

[12] R.Ramesh & Kathleen M.Brown, An efficient four-phase heuristic for the generalized orienteering problem, Computers & Operations Research, Volume 18, Issue 2, 1991, Pages 151-165

[13] Schilde et al.(2009), Metaheuristics for the bi-objective orienteering problem, Swarm Intelligence 3(3):179-201, May 2009

[14] Chao I, Golden B ,Wasil E, A fast and effective heuristic for the orienteering problem, European Journal of Operational Research 1996 ;88:475–89.

[15] Sevkli, Z. , & Sevilgen, F. E. (2010a). Discrete particle swarm optimization for the orienteering problem. In Proceedings of the IEEE congress on evolutionary compu- tation (CEC 2010), Barcelona, Spain (pp. 3234–3241) .

[16] Sevkli, Z. , & Sevilgen, F. E. (2010b). StPSO: Strengthened particle swarm opti- mization. Turkish Journal of Electrical Engineering & Computer Sciences, 18 (6), 1095–1114

[17] Chekuri, C. , Korula, N. , & Pál, M. (2012). Improved algorithms for orienteering and related problems. ACM Transactions on Algorithms, 8 , 661–670
[18] Dang, D.-C. , El-Hajj, R. , & Moukrim, A. (2013a). A branch-and-cut algorithm for solving the team orienteering problem. In C. Gomes, & M. Sellmann (Eds.), Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. Lecture Notes in Computer Science: 7874 (pp. 332–339). Springer
[19] Butt, S., Ryan, D., 1999. An optimal solution procedure for the multiple tour maximum collection problem using column generation. Computers and Operations Research 26, 427–441.
[20] Boussier, S., Feillet, D., Gendreau, M., 2007. An exact algorithm for the team orienteering problem. 4OR 5, 211–230.
[21] Tang, H., Miller-Hooks, E., 2005. A tabu search heuristic for the team orienteering problem. Computer and Operations Research 32, 1379–1407.
[22] Tricoire, F., Romauch, M., Doerner, K., Hartl, R., 2010. Heuristics for the multi-period orienteering problem with multiple time windows. Computers and Operations Research 37 (2), 351–367
[23] Kantor, M., Rosenwein, M., 1992. The orienteering problem with time windows. The
Journal of the Operational Research Society 43 (6), 629–635
[24] Mansini, R., Pelizzari, M., Wolfer, R. 2006. A granular variable neighbourhood search heuristic for the tour orienteering problem with time windows. Technical Report R.T 2006-02-52, University of Brescia, Italy.
[25] Righini, G., Salani, M. 2006. Dynamic programming for the orienteering problem with time windows. Technical Report 91 2006, Dipartimento di Tecnologie dell'Informazione, Universita degli Studi Milano, Crema, Italy.
[26] Righini, G., Salani, M., 2008. New dynamic programming algorithms for the resource constrained elementary shortest path. Networks 51 (3), 155–170.
[27] Montemanni, R., Gambardella, L., 2009. Ant colony system for team orienteering problems with time windows. Foundations of computing and Decision Sciences 34 (4), 287–306.
[28] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D., 2009d. Iterated local search for the team orienteering problem with time windows. Computers and Operations Research 36 (12), 3281–3290
[29] Tricoire, F., Romauch, M., Doerner, K., Hartl, R., 2010. Heuristics for the multi-period orienteering problem with multiple time windows. Computers and Operations Research 37 (2), 351–367.
[30] Gavalas, D. , Konstantopoulos, C. , Mastakas, K. , Pantziou, G. , & Tasoulas, Y. (2013). Cluster-based heuristics for the team orienteering problem with time windows. In V. Bonifaci, C. Demetrescu, & A. Marchetti-Spaccamela (Eds.), Experimental algorihtms. Lecture Notes in Computer Science: 7933 (pp. 390–401). Springer
[31] A. Likas, N. Vlassis, and J. Verbeek. The global k-means clustering algorithm.
Pattern Recognition, 36(2):451- 461, 2003.