A Fast Algorithm for Personalized Travel Planning Recommendation

Aldy Gunawan · **Hoong Chuin Lau** · **Kun Lu**

Abstract With the pervasive use of recommender systems and web/mobile applications such as TripAdvisor and Booking.com, an emerging interest is to generate personalized tourist routes based on a tourist's preferences and time budget constraints, often in real-time. The problem is generally known as the Tourist Trip Design Problem (TTDP) which is a route-planning problem on multiple Points of Interest (POIs). TTDP can be considered as an extension of the classical problem of Team Orienteering Problem with Time Windows (TOPTW). The objective of the TOPTW is to determine a fixed number of routes that maximize the total collected score. The TOPTW also considers the time window constraints when the visit at a particular node has to start within a predefined time window. In the context of the TTDP, the utility score for a particular node can be treated as the user's preference on a POI. In this paper, we propose a mathematical model for the TTDP that extends the TOPTW constraints by incorporating more real-world constraints, such as different total travel time budgets, different start and end nodes for routes. We then propose an Iterated Local Search (ILS) algorithm to solve both TTDP and TOPTW. We implement our ILS to provide tour guidance in the Singapore context. We show experimentally that ILS is able to solving real-world problem instances within a few seconds, and our ILS can improve 19 best known solution values on the benchmark TOPTW instances.

Keywords Recommender System · Tourist Trip Design Problem · Team Orienteering Problem with Time Windows · Iterated Local Search

1 INTRODUCTION

Recommender systems and apps are pervasive in the world of consumer choice, where different people who use a recommender system expect to get their own recommendations based on their own preferences [22]. In the e-Commerce world, we see

A. Gunawan, H.C. Lau and K. Lu
Singapore Management University
E-mail: aldygunawan, hclau, kunlu@smu.edu.sg

increasing proliferation of product recommender systems such as those available on Amazon and taobao. Likewise, in the travel world, we have recommender systems in portals such as TripAdvisor, Booking.com and AirBnB to name a few. Much of recommendations provided by these websites are information-centric, based on ratings given by past travellers. These portals generally do not provide planning capabilities that enable a travel itinerary to be generated based on personalized preferences time as well as monetary budget constraints.

From a planning perspective, what is needed when a tourist visits a country or a city is to have a schedule for multiple Points of Interest (POIs) that maximizes his/her experience based on his preferences and constraints such as time budget, the opening and closing days/times of POIs, distances between POIs. This problem is generally called the Tourist Trip Design Problem (TTDP) [36].

The OP is first introduced by Tsiligirides [31]. The objective of OP is to determine a route that consists of a subset of nodes and define the sequence of the selected nodes so that the total collected score from nodes is maximized and the time budget is not exceeded. The Team OP (TOP) is the extension of the OP to multiple routes. The TTDP can also be considered as an extension of the Team OP with Time Windows (TOPTW) [29] which needs to produce multiple routes subject to time window constraints arising from the opening and closing times of POIs. In the context of the TTDP, each route can be interpreted as a day trip. A node represents a particular POI. The score of a node represents the tourist's preference for that particular POI.

This paper is concerned with developing a fast algorithm that can be embedded within a (real-time) personalized tour planning recommender system for use by website/mobile apps. We introduce an extended mathematical model for TTDP that includes some additional constraints, such as different start and end nodes and the time budget for each route. We then propose an Iterated Local Search (ILS) approach to solve both TTDP and TOPTW. The proposed algorithm is an extension of ILS proposed by Gunawan et al. [8] for solving the OPTW. An initial feasible solution is constructed by a greedy heuristic. The initial solution is further improved by Iterated Local Search (ILS). Several additional components and operators of ILS are added.

In summary, the contribution of the paper is threefold:

- We introduce an extended mathematical model of the TTDP that addresses the main drawbacks of the classical TOPTW mathematical model.
- We propose an ILS algorithm for solving the TTDP and TOPTW.
- We show experimentally that our ILS algorithm is effective in providing good and fast solutions for the TTDP as well as improving 19 best known solution values of the benchmark TOPTW instances.

The remainder of this paper is organized as follows. Section 2 provides a literature review. In Section 3, we present the description and the mathematical model of the TTDP. Section 3 is also devoted to the proposed algorithm to solve the TTDP instances and the benchmark TOPTW instances. Section 4 provides the computation results together with the analysis of the results. Section 5 concludes the paper and summarizes directions for further research.

2 LITERATURE REVIEW

An evidence of the emerging importance in travel planning recommendation is demonstrated by the popularity of web and mobile applications that provide personalized tourist routes. Some web-based recommender systems are *Otium* (`http://sinai.ujaen.es/otium`) [19] and *City Trip Planner* (`http://www.citytripplanner.com/en/home`) [35]. One of the latest mobile apps built for the iOS platform is *GUIDEME* system [32].

The underlying problem for travel planning is the TTDP, and Gavalas et al. [5] provide a comprehensive survey. In the following, we summarize key contributions related to the TTDP. Souffriau et al. [28] proposed Guided Local Search (GLS) for solving the TTDP. The algorithm is applied to a real data set from the city of Ghent. It is concluded that GLS is able to produce good solutions within shorter computational times. Abbaspour and Samadzadegan [1] studied a more complex TTDP problem in a large urban area. A nested architecture in which tour planning routine calls multi-modal shortest path subroutine to generate an itinerary based on user preferences and restrictions of interesting points is proposed. Two engines of both the tour planner and the multi-modal shortest pathfinder are based on adapted Genetic Algorithms. The real dataset of city of Tehran is used to test the proposed algorithm.

Maervoet et al. [18] studied another type of the TTDP problem, namely the Outdoor Activity Tour Suggestion Problem. The problem considers the total path attractiveness which is based on the sum of the average arc attractiveness and the sum of the node scores in the path. A heuristic based on spatial filtering is presented. They conclude that it is a promising approach with low computational impact. Divsalar et al. [4] introduced another variant of the TTDP, namely the OP with Hotel Selection and Time Windows, to model multi-day trips to tourism regions. The main assumption is the location of the hotel may affect which POIs would be visited during the day. They propose a hybrid Genetic Algorithm with an embedded Variable Neighborhood Descent algorithm. Hasuike et al. [10] formulated the sightseeing route planning problem by introducing fuzzy random variables for travel times, satisfactions and the tiredness-dependency of the user. The objective is to minimize the maximum weighted total travel time and sightseeing under some possible conditions. Hasuike et al. [11] introduced the bi-objective Sightseeing Route Planning by considering two objectives: to minimize the maximum weighted total travel time and sightseeing and to maximize the total satisfaction value.

The Team OP with Time Windows (TOPTW) is a simplified version of the TTDP. Vansteenwegen et al. [33] provided an excellent review of the OP and its variants that has been done up to 2009. Gunawan et al. [9] summarized the most recent literature about the orienteering problem and its variants, including the proposed approaches and applications.

In recent years, heuristic and metaheuristic algorithms have been proposed to tackle the TOPTW efficiently for real-time applications, such as Iterated Local Search [34], Variable Neighborhood Search [30], Ant Colony System [21], a hybridization of a Greedy Randomized Adaptive Search Procedure and Evolutionary Local Search [15], LP-based Granular Variable Neighborhood Search [16], Simulated Annealing [25], a hybridization of Greedy Randomized Adaptive Search Procedure (GRASP)

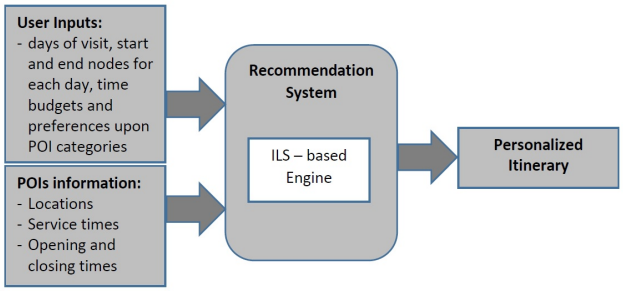

Fig. 1: TTDP Recommendation System

and Iterated Local Search [27], a hybridization of a local search procedure and Simulated Annealing [12] and Artificial Bee Colony [3]. For the TOPTW with the number of route = 1 (namely OPTW), there are some algorithms that have been proposed, such as the pulse algorithm [17] and Iterated Local Search [8].

3 PERSONALIZED TRAVEL PLANNING RECOMMENDATION

In this section, we describe our approach for constructing a personalized TTDP recommender system. A high level architecture is given in Figure 1. The user provides essential personal information such as days of visit, start and end nodes for each day, time budgets and preferences for the different POI categories. Then, the route planning engine calls the ILS algorithm to find the routes. Finally, an itinerary is recommended to the user. In the following, we present the underlying mathematical model followed by our proposed algorithm for generating the itinerary.

3.1 Mathematical Model

The TTDP is formally defined as follows. We are given a graph defined by a set of nodes $N = \{1, 2, \cdots, |N|\}$ where each node $i \in N$ is associated with score u_i and service time T_i. The non-negative travel time between nodes i and j is represented as t_{ij}. Our goal is to find a set of routes $M = \{1, 2, \cdots, |M|\}$ that maximizes the sum of scores of visited nodes from all routes. In the context of the TTDP, one route can be interpreted as one day trip. It is common for a tourist to visit a particular country / city for multiple days / routes.

To define the mathematical model for TTDP, some additional notations are defined as follows:

- $N^{start} = \{n_1^{start}, n_2^{start}, \cdots, n_{|M|}^{start}\}$, where n_m^{start} represents the start node in route m ($n_m^{start} \in N, m \in M$).
- $N^{end} = \{n_1^{end}, n_2^{end}, \cdots, n_{|M|}^{end}\}$, where n_m^{end} represents the end node in route m ($n_m^{end} \in N, m \in M$).

Each node i is associated with a time window $[e_i, l_i]$, where e_i and l_i are the earliest and latest times allowed for starting the service at node i. We assume that $e_{n_m^{start}} = 0$ $(\forall n_m^{start} \in N^{start}, m \in M)$, $e_{n_m^{end}} = 0$ $(\forall n_m^{end} \in N^{end}, m \in M)$, $l_{n_m^{start}} = T_m^{max}$ $(\forall n_m^{start} \in N^{start}, m \in M)$, $l_{n_m^{end}} = T_m^{max}$ $(\forall n_m^{end} \in N^{end}, m \in M)$. T_i is set to 0 $(\forall i \in (N^{start} \cup N^{end}))$.

Our mathematical model uses the following decision variables:

- X_{ijm} = 1 if a visit to node i is followed by a visit to node j in route m, 0 otherwise.
- Y_{im} = 1 if a visit to node i in route m.
- S_{im} = the start time at node i in route m.
- W_{im} = the waiting time before entering node i in route m.

The TTDP mathematical programming model is presented below. The model can also be considered as the extension of the classical TOPTW mathematical model.

$$Maximize \sum_{m \in M} \sum_{i \in N \setminus (N^{start} \cup N^{end})} u_i Y_{im} \tag{1}$$

$$\sum_{j \in N \setminus N^{start}} X_{ijm} = 1 \ \ \forall m \in M, i \in N^{start} \tag{2}$$

$$\sum_{i \in N \setminus N^{end}} X_{ijm} = 1 \ \ \forall m \in M, j \in N^{end} \tag{3}$$

$$\sum_{i \in N \setminus N^{end}} X_{ikm} = Y_{km} \ \ \forall k \in N \setminus (N^{start} \cup N^{end}), m \in M \tag{4}$$

$$\sum_{j \in N \setminus N^{start}} X_{kjm} = Y_{km} \ \ \forall k \in N \setminus (N^{start} \cup N^{end}), m \in M \tag{5}$$

$$\sum_{m \in M} Y_{im} \leq 1 \ \ \forall i \in N \setminus (N^{start} \cup N^{end}) \tag{6}$$

$$Y_{im} = 1 \ \ \forall m \in M, i \in (N^{start} \cup N^{end}) \tag{7}$$

$$S_{im} \geq e_i \ \ \forall m \in M, i \in N \tag{8}$$

$$S_{im} \leq l_i \ \ \forall m \in M, i \in N \tag{9}$$

$$S_{im} + T_i + t_{ij} - S_{jm} \leq \hat{L}(1 - X_{ijm}) \ \ \forall i, j \in N, m \in M \tag{10}$$

$$S_{jm} - (S_{im} + T_i + t_{ij}) \leq \hat{L}(1 - X_{ijm}) + W_{jm} \ \ \forall i, j \in N, m \in M \tag{11}$$

$$\sum_{i \in N} T_i Y_{im} + \sum_{i \in N \setminus N^{end}} \sum_{j \in N \setminus N^{start}} t_{ij} X_{ijm} + \sum_{i \in N} W_{im} \leq T_m^{max} \ \ \forall m \in M \tag{12}$$

$$S_{im}, W_{im} \geq 0 \ \ \forall i \in N, m \in M \tag{13}$$

$$X_{ijm}, Y_{im} \in \{0, 1\} \ \ \forall i, j \in N, m \in M \tag{14}$$

The objective function 1 is to maximize the total collected score from visited nodes from all routes. Constraints 2 and 3 ensure that each route m starts from its start node n_m^{start} and ends at its end node n_m^{end}. Constraints 4 and 5 determine the connectivity of each route m. Constraints 6 prevent each node i to be visited on more than one route.

Constraints 7 ensure that the start node n_m^{start} and end node n_m^{end} in each route m are visited. Constraints 8 and 9 ensure that the start time at node i in route m is within time window $[e_i, l_i]$. Constraints 10 ensure that if nodes i and j are visited consecutively, then the start time at node j has to be greater than or equal to the start time at node i plus the service time at node i and the travel time from nodes i to j. In constraints 11, the waiting time before entering node j in route m, W_{jm}, is at l east equal to the start time at node j minus the arrival time at node j. Note that $\hat{L}$ is a very large constant value. Constraints 12 limit the time budget for each route m by T_m^{max}. Constraints 13 are the non-negativity condition for S_{im} and W_{im}. The binary conditions for X_{ijm} and Y_{im} are constrained by constraints 14. The binary condition also ensures that each node is only visited at most once.

Note that unlike the classical TOPTW, the above mathematical model allows different start and end nodes and different time budgets for routes. Simpler mathematical formulations of the TOPTW, when the start and end nodes are the same and the values of T_m^{max} for all routes are constant, can be found in [16] and [34].

Furthermore, in the TTDP context, e_i and l_i refer to the opening and closing times of a particular POI i, respectively. We need to ensure that the visitor has to leave the POI before l_i. In this scenario, constraints 9 are replaced by constraints 15.

$$S_{im} + T_i \leq l_i \;\; \forall m \in M, i \in N \tag{15}$$

3.2 Algorithm

We introduce a simple yet effective Iterated Local Search (ILS) algorithm that works well in real-world problem instances and improves the state-of-the-art results. This algorithm is an extension of ILS proposed by Gunawan et al. [8] for solving the OPTW. In a nutshell, our algorithm constructs an initial feasible solution via a greedy heuristic. The initial solution is further improved by Iterated Local Search (ILS).The algorithm is briefly described in the following sub-sections. We still include some explanations which are taken from the original paper [8] so the readers can have a better understanding.

3.2.1 Greedy Construction Heuristic

The basic idea is to start with an empty solution as the current solution and insert nodes iteratively to the current solution until there is no node can be inserted. The greedy construction heuristic is outlined in Algorithm 1.

Let N' and N^* be the sets of unscheduled and scheduled nodes, respectively ($N' \cup N^* = N$) and let F be the set of feasible candidate nodes to be inserted. N^* is initialized by N^{start} and N^{end}, while N' consists of unscheduled nodes. S_0 refers the current feasible solution obtained so far, represented as m-vector. Each row m is initialized with $n_m^{start} \in N^{start}$ and $n_m^{end} \in N^{end}$.

Let $P(m)$ represents a set of scheduled node positions in route m. We examine all possibilities of inserting an unscheduled node in position $p \in P(m)$ and store all feasible insertions into F. Each element in F, which represents a feasible insertion

Algorithm 1 CONSTRUCTION (N, M)

$N' \leftarrow N \setminus \{N^{start} \cup N^{end}\}$
$N^* \leftarrow \{N^{start} \cup N^{end}\}$
Initialize $S_0 \leftarrow N^*$
$F \leftarrow$ UPDATEF(N', M)
while $F \neq \emptyset$ **do**
  $\langle n^*, p^*, m^* \rangle \leftarrow$ SELECT(F)
  $S_0 \leftarrow \langle n^*, p^*, m^* \rangle$
  Update $P(m)$
  $N' \leftarrow N' \setminus \{n^*\}$
  $N^* \leftarrow N^* \cup \{n^*\}$
  $F \leftarrow$ UPDATEF(N', M)
end while
return S_0

of node n in position p of route m, is represented as $\langle n, p, m \rangle$. For each $\langle n, p, m \rangle$, we calculate the benefit of insertion $ratio_{n,p,m}$ by using equation 16. $\Delta_{n,p,m}$ represents the difference between the total time spent before and after the insertion of node n in position p of route m. All elements in F would be sorted in descending order based on $ratio_{n,p,m}$ values and we only keep f elements. The idea of generating F is summarized in Algorithm 2.

$$ratio_{n,p,m} = \left(\frac{u_n^2}{\Delta_{n,p,m}} \right) \tag{16}$$

If F is not an empty set, Algorithm 3 is run in order to select $\langle n^*, p^*, m^* \rangle$ to be inserted. Each $\langle n, p, m \rangle$ corresponds to a probability value, $prob_{n,p,m}$, which is calculated by Equation 17:

$$prob_{n,p,m} = \left(\frac{ratio_{n,p,m}}{\sum_{\langle i,j,k \rangle \in F} ratio_{i,j,k}} \right) \tag{17}$$

Selecting $\langle n^*, p^*, m^* \rangle$ from F is based on the Roulette-Wheel selection [6]. The probability of selecting a candidate insertion is proportional to its benefit, $ratio_{n,p,m}$. S_0, N' and N^* will be updated after the insertion. The greedy construction heuristic is terminated when $F = \emptyset$.

3.2.2 Iterated Local Search

Given the initial solution S_0 generated from the greedy construction heuristic, we propose ILS to improve the quality of S_0. Three components of ILS PERTURBATION, LOCALSEARCH and ACCEPTANCECRITERION are considered. Let S^* be the best found solution so far. The outline of ILS is presented in Algorithm 4.

In PERTURBATION, we implement two steps: EXCHANGEROUTE and SHAKE. If the number of iterations without improvement, NOIMPR, is larger than THRESHOLD2 and (NOIMPR + 1) Mod THRESHOLD3 = 0, EXCHANGEROUTE would be executed; otherwise, SHAKE would be selected. THRESHOLD2 and THRESHOLD3 are constant parameters need to be set.

Algorithm 2 UPDATEF (N', M)

$F \leftarrow \emptyset$
for all $n \in N'$ **do**
  for all $m \in M$ **do**
    for all $p \in P(m)$ **do**
      if insert node n in position p of route m is feasible **then**
        calculate $ratio_{n,p,m}$
        $F \leftarrow F \cup \langle n,p,m \rangle$
      end if
    end for
  end for
end for
Sort all elements of F in descending order based on $ratio_{n,p,m}$
Select the best f number of elements of F and remove the rest
return F

Algorithm 3 SELECT (F)

$SumRatio \leftarrow 0$
for all $\langle n,p,m \rangle \in F$ **do**
  $SumRatio \leftarrow SumRatio + ratio_{n,p,m}$
end for
for all $\langle n,p,m \rangle \in F$ **do**
  $prob_{n,p,m} \leftarrow ratio_{n,p,m}/SumRatio$
end for
$U \leftarrow rand(0,1)$
$AccumProb \leftarrow 0$
for all $\langle n,p,m \rangle \in F$ **do**
  $AccumProb \leftarrow AccumProb + prob_{n,p,m}$
  if $U \leq AccumProb$ **then**
    $\langle n^*,p^*,m^* \rangle \leftarrow \langle n,p,m \rangle$
    break
  end if
end for
return $\langle n^*,p^*,m^* \rangle$

In the EXCHANGEROUTE step, all nodes of one route are moved to the other route in the same order and vice versa. The strategy of selecting two different routes are based on generating of permutations by adjacent transposition method [14]. The SHAKE step is adopted from [34] with some modifications. During SHAKE step, one or more nodes will be removed in each route m, which depends on two integer values, CONS and POST. CONS indicates how many consecutive nodes to remove for a particular route while POST indicates the first position of the removing process in a particular route. If we reach the last scheduled node, the process will then be back to the first node after the start node n_m^{start}. Both CONS and POST are initially set to 1. After each SHAKE step, POST is increased by CONS. CONS would also be increased by one after a fixed number of consecutive iterations.If POST is greater than the size of the smallest route, POST is subtracted with the size of the smallest route to determine the new position POST. If CONS is greater than the size of the largest route, or S^* is updated, CONS is reset to one. After removing CONS nodes, we update N' and N^*

Algorithm 4 ILS (N,M)

```
S0 ← CONSTRUCTION(N,M)
S0 ← LOCALSEARCH(S0,N*,N',M)
S* ← S0
NOIMPR ← 0
while TIMELIMIT has not been reached do
   S0 ← PERTURBATION(S0,N*,N',M)
   S0 ← LOCALSEARCH(S0,N*,N',M)
   if S0 better than S* then
      S* ← S0
      NOIMPR ← 0
   else
      NOIMPR ← NOIMPR + 1
   end if
   if (NOIMPR+1) MOD THRESHOLD1 = 0 then
      S0 ← S*
   end if
end while
return S*
```

accordingly. F is then regenerated based on Algorithm 2 and an unscheduled node that needs to be inserted is selected using Algorithm 3. This is repeated until $F = \emptyset$.

Table 1: LOCAL SEARCH operations

Operations	Descriptions
SWAP1	Exchange two nodes within one route
SWAP2	Exchange two nodes within two routes
2-OPT	Reorder the sequence of certain nodes within one route
MOVE	Move one node from one route to another route
INSERT	Insert nodes into a route
REPLACE	Replace one scheduled node with one unscheduled node

In LOCALSEARCH, we run six different operations consecutively, as shown in Table 1. Take note that four operations: SWAP1, 2-OPT, INSERT and REPLACE are taken from [8]. In this paper, we only explain two additional operations, SWAP2 and MOVE.

The idea of exchanging two scheduled nodes within one particular route SWAP1 is extended to two different routes with the lowest and the second lowest unused time budget, namely SWAP2. The swapping is executed if the total of unused time budgets of both selected routes is increased. All possible combinations of selecting two different nodes are examined.

MOVE is performed by reallocating one node from one route to another one. It is started from the first scheduled node n^* from the first route m^*. The idea is to generate F using Algorithm 2 where $N' = \{n^*\}$ and $M = M \setminus \{m^*\}$. If $F \neq \emptyset$, node n^* would be reallocated based on Algorithm 3. Otherwise, we continue to the next scheduled node. This operation would be terminated if node n^* is moved or we have reached the last scheduled node of the last route $|M|$.

If S^* is not updated for a certain number of iterations,((NOIMPR+1) MOD THRESHOLD1 = 0), we apply an intensification strategy. This strategy restarts the search from

the best found solution, S^*. Finally, the entire algorithm will be run within a given computational budget, TIMELIMIT.

4 COMPUTATIONAL EXPERIMENTS

4.1 Instances and Experimental Setup

Table 2 summarizes the details of the instances used in our paper. The experiments were carried out on a personal computer Intel Core i7 - 4770 with 3.4 GHz processor and 16 GB RAM.

Table 2: Benchmark Instances

Reference	Problem	Category	Name	Instance sets	$\|N\|$	$\|M\|$
-	TTDP	-	Real-world	$R1$-$R5$	50	1 to 5
[24]	TOPTW	INST-M	Solomon 100	c^*_100, r^*_100, rc^*_100	100	1 to 4
			Cordeau 1-10	$pr01$ - $pr10$	[48, 288]	
[20]	TOPTW	INST-M	Solomon 200	c^*_200, r^*_200, rc^*_200	100	1 to 4
			Cordeau 11-20	$pr11$ - $pr20$	[48, 288]	
[34]	TOPTW	OPT	Solomon 100 & 200	c^*_100, r^*_100, rc^*_100	100	3 to 20
				c^*_200, r^*_200, rc^*_200	100	
			Cordeau 1-10	$pr01$ - $pr10$	[48, 288]	

4.1.1 Real-world TTDP Instances

We introduce a set of real-world TTDP instances derived from the city of Singapore with 50 Point of Interests (POIs). The score for each POI was derived from local knowledge, which can be adjusted according to user's preferences. In order to obtain a quick response, we limit the run time to 5 seconds for each instance. The ILS was executed in 1 run for each instance.

The start and end nodes are different for each route m and the time budget for each route m, T_m^{max}, may vary. The time window $[e_i, l_i]$ for POI i represents the opening and closing times of POI i. The user has to leave POI i before the closing time l_i. The score related to the attractiveness of POI u_i is assumed to be known. The travel time between two POIs, the service time, the opening and closing time for POIs are also known.

4.1.2 Benchmark TOPTW Instances

Since test instances for the extended TOPTW problem are not available, we use benchmark TOPTW instances to validate the performance of ILS. The OPTW instances were initially proposed by Righini and Salani [24]. 48 test problems were generated from Solomon's instances [26] and 10 instances were adapted from Cordeau's instances [2]. Both Solomon's and Cordeau's instances were originally designed for vehicle routing problems with time windows and multi-depot vehicle routing problems, respectively. [20] developed another set of 37 instances for the OPTW. The

Table 3: Estimation of single-thread performance [12]

Algorithm	Experimental environment	*SuperPi*	Estimate of single-thread performance	Number of trials for each instance
IterILS	Intel Core 2 with 2.5 gigahertz CPU, 3.45 gigabytes RAM	18.6	0.53	1
VNS	Intel Pentium 4 with 2.4 gigahertz CPU, 4 gigabytes RAM	unknown	0.53	10
EACS	Dual AMD Opteron 250 2.4 gigahertz CPU, 4 gigabytes RAM	unknown	0.22	5
SSA	Intel Core 2 CPU, 2.5 gigahertz	18.6	0.53	1
GVNS	Intel Pentium (R) IV, 3 gigahertz CPU	44.3	0.22	10
GRILS	Intel Xeon with 2.5 gigahertz CPU, 4 gigabytes RAM	unknown	0.53	10
I3CH	Intel Xeon E5430 with 2.66 gigahertz CPU, 8 gigabytes RAM	14.7	0.67	1
ILS	Intel Core i7-4770 with 3.4 GHz processor, 16 gigabytes RAM	9.8	1	10

TOPTW instances are constructed by extending the OPTW instances with different values of routes: $m = 2$, 3 and 4. This set of instances is considered under the "INST-M" category [12].

Vansteenwegen et al. [34] added more benchmark instances based on instances of [26] and [2]. Those instances are considered more difficult instances, with the number of routes $|M|$ is set to the number of vehicles. The optimal solution for these instances are known due to the specific setting on the number of provided vehicles, which is equal to the total score collected from all customers. Hu and Lim [12] considered this set of benchmark instances as the "OPT" category.

ILS was executed in 10 runs with different random seeds for each benchmark instance. The performances of the proposed ILS in solving benchmark TOPTW instances are compared to the state-of-the-art algorithms: IterLS [34], VNS [30], EACS [21], SSA [25], GVNS [16], GRILS [27] and I3CH [12].

Gunawan et al. [8] and Hu and Lim [12] used the *SuperPi* benchmark to compare the computer speeds and ensure the fairness among algorithms. *SuperPi* is a single-threaded program that computes the first 1 million digits of π of a particular processor. Table 3 summarizes the experimental environment of each algorithm [12] for comparing the speed of the computers. Labadie et al. [15] show the comparability of processors used by ACS* and GVNS since the *SuperPi* for ACS* is not available. The same conclusion is applied to VNS [30] and GRILS [27]. For the details, please refer to [12].

Take note that only EACS uses one hour computational time for each instance, while others use the number of iterations. We decide to implement EACS's computational time as our reference since we are more concerned with solution quality. We adjust the computational time and decide to use only 35% of the adjusted EACS's computational time ($\sim$3600 seconds) for solving each instance. Therefore, the computational time for each instance is set to $35\% \times 0.22 \times 3600$ seconds = 272 seconds using our processor.

We also implement another scenario by referring to the computational times of I3CH. I3CH is considered as the best among the state-of-the-art algorithms [12].

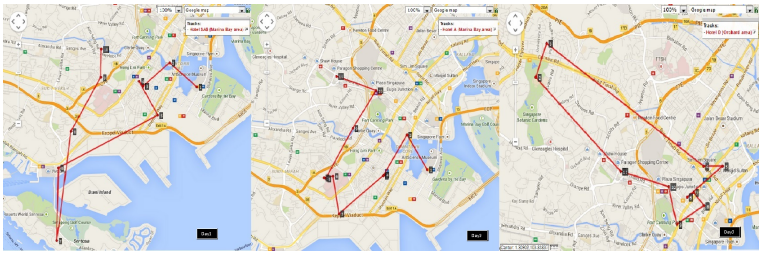

Fig. 2: Example Routes of $R3$

4.2 Experimental Results

The summary of results obtained are reported in the following sub-sections. The details of results can also be downloaded at `http://centres.smu.edu.sg/larc/Orienteering-Problem-Library`.

4.2.1 Real-world TTDP Instances

As described in Section 4.1.1, we generate different instances to simulate different user behaviors. Table 4 provides the details of those five different instances. We also vary the number of routes, the start and end nodes and the time budget for each route. The last column of Table 4 represents the solutions in terms of the number of POIs that can be visited. In general, the higher the value of m, the more POIs can be visited. For example, the total numbers of POIs are 11, 20 and 26 POIs for instances $R1, R2$ and $R3$, respectively. The number of POIs visited is slightly decreased for $m = 4$ (instance $R4$) since the user is moved to a hotel which is closer to the airport and far from most of POIs.

Table 5 illustrates POIs that can be visited for a three-days trip ($R3$) generated by ILS. Figure 2 visualizes the proposed routes for $R3$. We conclude that ILS is effective for solving real-world problems related to the TTDP within a few seconds. Take note that the scores of POIs can always been adjusted with respect to the user preferences.

Table 4: Real-world instances

Instance	$\|M\|$	Route (day) m	Start node	End node	Start time	End time	T_m^{max} (mins)	Number of POIs that can be visited
$R1$	1	1	Hotel A (Marina Bay area)	Hotel B (China Town area)	9.00	18.00	540	11
$R2$	2	1	Hotel A (Marina Bay area)	Hotel C (Bugis Area)	9.00	18.00	540	10
		2	Hotel A (Marina Bay area)	Hotel D (Orchard area)	9.00	18.00	540	10
$R3$	3	1	Hotel A (Marina Bay area)	Hotel B (China Town area)	9.00	18.00	540	8
		2	Hotel A (Marina Bay area)	Hotel D (Orchard area)	9.00	18.00	540	9
		3	Hotel D (Orchard area)	Hotel D (Orchard area)	9.00	18.00	540	9
$R4$	4	1	Hotel A (Marina Bay area)	Hotel B (China Town area)	9.00	18.00	540	10
		2	Hotel A (Marina Bay area)	Hotel C (Bugis Area)	9.00	18.00	540	8
		3	Hotel A (Marina Bay area)	Hotel E (East area)	9.00	18.00	540	6
		4	Hotel E (East area)	Hotel E (East area)	9.00	15.00	360	1
$R5$	5	1	Hotel A (Marina Bay area)	Hotel B (China Town area)	9.00	18.00	540	10
		2	Hotel A (Marina Bay area)	Hotel C (Bugis Area)	9.00	18.00	540	7
		3	Hotel A (Marina Bay area)	Hotel F (Sentosa area)	9.00	18.00	540	8
		4	Hotel F (Sentosa area)	Hotel E (East area)	9.00	18.00	540	3
		5	Hotel E (East area)	Hotel E (East area)	9.00	15.00	360	1

Table 5: The detailed routes of $R3$

Route	POIs visited
1	(1) Hotel A (Marina Bay area) → (2) Former Empress Place Building → (3) Sri Mariamman Temple → (4) Chinatown Heritage Centre → (5) Former Singapore Conference Hall and Trade Union House → (6) Former St James Power Station → (7) Capella Singapore → (8) Church of St Teresa → (9) WANGZ Hotel → (10) Hotel B (China Town area)
2	(1) Hotel A (Marina Bay area) → (2) Esplanade Park Memorials → (3) Thian Hock Kheng Temple → (4) Keng Teck Whay (5) Tanjong Pagar Railway Station → (6) Bowyer Block → (7) Tan Teck Guan Building → (8) College of Medicine Building (9) Singapore Tyler Print Institute → (10) MacDonald House → (11) Hotel D (Orchard area)
3	(1) Hotel D (Orchard area) → (2) Former Raffles College → (3) Command House → (4) St Joseph's Church (5) Malay Heritage Centre → (6) Church of St Peter and St Paul → (7) Church of Our Lady of Lourdes → (8) Civilian War Memorial (9) Civil Defence Heritage Gallery → (10) The Cathay → (11) Hotel D (Orchard area)

4.2.2 Benchmark TOPTW Instances

Table 6 reports the new best known solutions (new *BK*s) obtained by ILS. Most of new *BK*s are from $m = 1$ and 2, while the rest are from $m = 3$ and 4. Table 7 summarizes the overall results of "INST-M" instances solved by IterILS, VNS, EACS, GVNS, SSA, GRILS, I3CH and our proposed ILS. The table reports the average relative percentage deviation (AG), which refers to the average of percentage gap between *BK* and the average solutions obtained by a particular algorithm across all benchmark instances. We observe that ILS can produce a better AG against those of VNS, EACS, GVNS and GRILS. However, IterILS, SSA and I3CH only reported their best known solution obtained; therefore, we calculate the best relative percentage deviation (BG) that represents the average of percentage gap between *BK* and the best solution obtained across all instances. In average, ILS provides a better BG value compared with those of other methods, except VNS. Although the BG of VNS is smaller, but it requires more computational time.

Table 6: New best known solution values found by ILS

Instance	m	Old *BK*	New *BK*	Instance	m	Old *BK*	New *BK*
r203	1	1021	1026	r107	2	536	538
r204	1	1086	1093	r205	2	1380	1385
r209	1	950	956	r209	2	1405	1406
r211	1	1046	1049	rc206	2	1546	1552
rc202	1	936	938	r104	3	777	778
rc206	1	895	899	rc104	3	834	835
rc208	1	1053	1057	r104	4	972	973
pr15	2	1219	1220	rc107	4	980	985

The computational times of EACS and ILS represent the average of computational times to obtain the best found *BK* (in seconds). IterILS, VNS, SSA, GVNS, GRILS and I3CH report the average of computational times for solving each instance. ILS requires shorter computational times than VNS and EACS do. On the other hand, ILS needs longer computational times compared against the ones of GVNS and GRILS. Table 8 reports the results of the "OPT" category. In terms of BG values, ILS result is comparable to that of SSA, although it is worse than that of I3CH.

We compare the performance of ILS against I3CH in solving the "INST-M" instances by setting the computational times to those of I3CH. The results are shown in

Table 7: Comparison for "INST-M"

Algorithm	AG(%)	BG(%)	Time (seconds)
IterILS	-	3.50	1.6
VNS	1.42	0.48	197.9
EACS	2.33	1.69	310.3
SSA	-	1.09	46.5
GVNS	1.74	1.00	14.2
GRILS	3.87	2.81	5.6
I3CH	-	0.69	134
ILS	1.19	0.54	126.6

Table 8: Comparison for "OPT"

Algorithm	AG(%)	BG(%)	Time (seconds)
IterILS	-	1.30	3.5
SSA	-	0.45	76.6
GVNS	0.74	-	4.9
I3CH	-	0.15	183.0
ILS	0.82	0.45	188.8

Table 9: Comparison with the same computational time

Algorithm	AG(%)	BG(%)	Time (seconds)
I3CH	-	0.69	134.7
ILS	1.33	0.53	134.7

Table 10: New best known solution values found by ILS using I3CH computational time

Instance	m	Old BK	New BK	Instance	m	Old BK	New BK
pr15	1	707	708	r206	2	1440	1442
r209	2	1405	1407	pr04	2	925	926

Table 9. More new BKs especially for $m = 2$ instances have been found, as listed in Table 10. The best known of instance r209 ($m = 2$) has also been further improved to 1407.

5 CONCLUSION

We develop a fast algorithm based on Iterated Local Search (ILS) for generating a personalized tour planning recommendation. We introduce a TTDP mathematical model which extends the TOPTW model by including several additional real-world constraints, such as different maximum total travel times and different start and end nodes. A factorial experimental design is implemented for tuning the parameter values of ILS. Computational results have shown that our approach is capable of generating good and fast itineraries for the TTDP, as well as improving 19 best known solution values among benchmark instances of the TOPTW.

Our future focus is to improve the performance of ILS by modifying it in order to solve larger values of m, e.g. focusing on allocating nodes with shorter time windows that are more difficult to allocate. It would be interesting to apply the proposed algorithm to other variants of the OP, e.g. the Time Dependent OP and the Multi-Constraint Team OP with (Multiple) Time Windows.

Acknowledgements This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative.

References

1. Abbaspour, R.A., Samadzadegan, F.: Time-dependent personal tour planning and scheduling in metropolises. Expert Systems with Applications **38**(10), 12,439–12,452 (2011)
2. Cordeau, J.F., Grendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks **30**(2), 105–119 (1997)
3. Cura, T.: An artificial bee colony algorithm approach for the team orienteering problem with time windows. Computers and Industrial Engineering **74**, 270–290 (2014)
4. Divsalar, A., Vansteenwegen, P., Chitsaz, M., Sörensen, K., Cattrysse, D.: Personalized multi-day trips to touristic regions: a hybrid GA-VND approach. In: C. Blum, G. Ochoa (eds.) Proceeding of EvoCOP 2014, *Lecture Notes in Computer Science*, vol. 8600, pp. 194–205. Springer (2014)

5. Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G.: A survey on algorithmic approaches for solving tourist trip design problems. Journal of Heuristics **20**(3), 291–328 (2014)
6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, Massachusetts (1989)
7. Gunawan, A., Lau, H.C., Lindawati: Fine-tuning algorithm parameters using the design of experiments approach. In: C.A.C. Coello (ed.) proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION5), *Lecture Notes in Computer Science*, vol. 6683, pp. 278–292. Springer (2011)
8. Gunawan, A., Lau, H.C., Lu, K.: An iterated local search algorithm for solving the orienteering problem with time windows. In: G. Ochoa, F. Chicano (eds.) proceedings of the 15th European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoStar 2015), 8-10 April 2015, Copenhagen, Denmark, *Lecture Notes in Computer Science*, vol. 9026, pp. 61–73. Springer-Verlag, Berlin, Germany (2015)
9. Gunawan, A., Lau, H.C., Vansteenwegen, P.: Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research (2016). DOI 10.1016/j.ejor.2016.04.059
10. Hasuike, T., Katagiri, H., Tsubaki, H., Tsuda, H.: Sightseeing route planning responding various conditions with fuzzy random satisfactions dependent on tourist's tiredness. In: S.I. Ao, O. Castillo, C. Douglas, D.D. Feng, J.A. Lee (eds.) Proceedings of the International MultiConference of Engineers and Computer Scientists 2014 (IMECS 2014), *Lecture Notes in Engineering and Computer Science*, vol. 2210, pp. 1232–1236. Newswood Limited (2014)
11. Hasuike, T., Katagiri, H., Tsubaki, H., Tsuda, H.: Biobjective sightseeing route planning with uncertainty dependent on tourist's tiredness responding various conditions. In: G.C. Yang, S.I. Ao, O. Huang X. Castillo (eds.) Transcations on Engineering Technologies, pp. 169–179. Springer (2015)
12. Hu, Q., Lim, A.: An iterative three-component heuristic for the team orienteering problem with time windows. European Journal of Operational Research **232**(2), 276–286 (2014)
13. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research **36**, 267–306 (2009)
14. Johnson, S.M.: Generation of permutations by adjacent transposition. Mathematics of Computation **17**(83), 282–285 (1963)
15. Labadie, N., Mansini, R., Melechovskỳ, J., Calvo, R.W.: Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. Journal of Heuristics **17**(6), 729–753 (2011)
16. Labadie, N., Mansini, R., Melechovskỳ, J., Calvo, R.W.: The team orienteering problem with time windows: an LP-based granular variable neighborhood search. European Journal of Operational Research **220**(1), 15–27 (2012)
17. Lozano, L., Duque, D., Medaglia, A.L.: An exact algorithm for the elementary shortest path problem with resource constraints. Transportation Science (2014). URL `http://hdl.handle.net/1992/1181`
18. Maervoet, J., Brackman, P., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G.: Tour suggestion for utdoor activities. In: S. Liang, X. Wang, C. Claramunt (eds.) Web and Wireless Geographical Information Systems, *Lecture Notes in Computer Science*, vol. 7820, pp. 54–63. Springer (2013)
19. Montejo-Ráez, A., Perea-Ortega, J.M., García-Cumbreras, M.A., Martínez-Santiago, F.: Otium: A web based planner for tourism and leisure. Expert Systems with Applications (8) (2011)
20. Montemanni, R., Gambardella, L.M.: Ant colony system for team orienteering problem with time windows. Foundations of Computing and Decision Sciences **34**(4), 287–306 (2009)
21. Montemanni, R., Weyland, D., Gambardella, L.M.: An enhanced ant colony system for the team orienteering problem with time windows. In: Proceedings of 2011 International Symposium on Computer Science and Society (ISCCS), pp. 381–384 (2011)
22. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM **40**(3) (1997)
23. Ridge, E., Kudenko, D.: Tuning the performance of the MMAS heuristic. In: T. Stützle, M. Birattari, H.H. Hoos (eds.) proceedings of the workshop of Stochastic local search algorithms (SLS2007), *Lecture Notes in Computer Science*, vol. 4638, pp. 46–60. Springer (2007)
24. Righini, G., Salani, M.: Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. Computers and Operations Research **36**(4), 1191–1203 (2009)
25. Shih, W.L., Yu, V.F.: A simulated annealing heuristic for the team orienteering problem with time windows. European Journal of Operational Research **217**(1), 94–107 (2012)

26. Solomon, M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research **35**(2), 254–265 (1987)
27. Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., Van Oudheusden, D.: The multiconstraint team orienteering problem with multiple time windows. Transportation Science **47**(1), 1–11 (2013)
28. Souffriau, W., Vansteenwegen, P., Vertommen, J., Vanden Berghe, G., Van Oudheusden, D.: A personalised tourist trip design algorithm for mobile tourist guides. Applied Artificial Intelligence **22**(10), 964–985 (2008)
29. Sylejmani, K., Dika, A.: Solving touristic trip planning problem by using taboo search approach. International Journal of Computer Science **8**(5), 139–149 (2011)
30. Tricoire, F., Romauch, M., Doerner, K.F., Hartl, R.F.: Heuristics for the multi-period orienteering problem with multiple time windows. Computers and Operations Research **37**(2), 351–367 (2010)
31. Tsiligirides, T.: Heuristic methods applied to orienteering. Journal of the Operational Research Society **35**(9), 797–809 (1984)
32. Umanets, A., Ferreira, A., Leite, N.: GuideMe - a tourist guide with a recommender system and social interaction. Procedia Technology **17**, 407–414 (2014)
33. Vansteenwegen, P., Souffriau, W., Oudheusden, D.V.: The orienteering problem: A survey. European Journal of Operational Research **209**(1), 1–10 (2011)
34. Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: Iterated local search for the team orienteering problem with time windows. Computers and Operations Research **36**(12), 3281–3290 (2009)
35. Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: The city trip planner. Expert Systems with Applications **38**(6), 6540–6546 (2011)
36. Vansteenwegen, P., Van Oudheusden, D.: The mobile tourist guide: an OR opportunity. OR Insights **20**(3), 21–27 (2007)