8-2014

A mathematical model and metaheuristics for Time Dependent Orienteering Problem

Aldy GUNAWAN
Singapore Management University, aldygunawan@smu.edu.sg

Zhi YUAN
Helmut Schmidt University

Hoong Chuin LAU
Singapore Management University, hclau@smu.edu.sg

10th International Conference of the Practice and Theory of Automated Timetabling
PATAT 2014, 26-29 August 2014, York, United Kingdom

A Mathematical Model and Metaheuristics for Time Dependent Orienteering Problem

Aldy GUNAWAN* · **Zhi YUAN*** ·
Hoong Chuin LAU

Abstract This paper presents a generalization of the Orienteering Problem, the Time-Dependent Orienteering Problem (TDOP) which is based on the real-life application of providing automatic tour guidance to a large leisure facility such as a theme park. In this problem, the travel time between two nodes depends on the time when the trip starts. We formulate the problem as an integer linear programming (ILP) model. We then develop various heuristics in a step by step fashion: greedy construction, local search and variable neighborhood descent, and two versions of iterated local search. The proposed metaheuristics were tested on modified benchmark instances, randomly generated problem instances, and two real world problem instances extracted from two popular theme parks in Asia. Experimental results confirm the effectiveness of the developed metaheuristic approaches, especially an iterated local search with adaptive perturbation size and probabilistic intensified restart mechanism. It finds within an acceptably short computation time, the optimal or near optimal solutions for TDOP instances of realistic size as in our target application.

Keywords Time-Dependent Orienteering Problem · Integer Linear Programming · Metaheuristics · Iterated Local Search

1 Introduction

The Orienteering Problem (OP) is originated from the sport game of orienteering [2]. The main goal is to find a single route by visiting as many nodes as possible

* Contributed equally

A. Gunawan and H.C. Lau
School of Information Systems, Singapore Management University, Singapore
E-mail: aldygunawan, hclau@smu.edu.sg
Z. Yuan
Professorship of Applied Mathematics, Department of Mechanical Engineering, Helmut Schmidt University, Hamburg, Germany
E-mail: yuanz@hsu-hh.de

that maximizes the total collected score subject to a given time budget frame. It is assumed that the starting point and the end point are fixed. Many OP applications are described in the literature: selective travelling salesperson problem [21]), home fuel delivery problem [8], single-ring design problem [20], and mobile tourist guide [17].

Several variants of the OP include: 1) Team Orienteering Problem (TOP) [2, 18], 2) Orienteering Problem with Time Windows (OPTW) [15], 3)Team Orienteering Problem with Time Windows (TOPTW) [23], 4) Time-Dependent Orienteering Problem (TDOP) [5].

In this paper, we study the Time-Dependent Orienteering Problem (TDOP), which is a generalization of OP. In the classical OP, the changes to the network over time are not taken into account. However, in certain networks, the route between two nodes actually depends on the network properties, such as congestion levels, construction zone on certain segments, etc., which will affect the travel time between two nodes. Our target application of this work is to provide automatic tour guidance to theme park visitors, taking into account the waiting time of a theme park varies over time. The goal is to maximize the overall utility of the visited attractions within the tourist's available visiting period.

We formulate the TDOP as an Integer Linear Programming (ILP) model. Due to the computational inefficiency in solving large-scale instances with a commercial ILP solver, we then develop various metaheuristics, including a greedy construction heuristic, two local search operators and variable neighborhood descent, and two versions of iterated local search: a basic version and a further improved version by adaptive perturbation strength and probabilistic intensification mechanism. All these approaches were tested on modified benchmark instances, randomly generated instances and two case studies extracted from real world theme park data.

The paper is organized as follows. We first provide a brief review of the OP and TDOP in Section 2. We then describe the TDOP and formulate it as an Integer Linear Programming model in Section 3. In Section 4, metaheuristics are proposed to solve the problem. Section 5 provides the computational results together with the analysis of the results. Finally, we provide concluding perspectives and directions for future research.

2 Literature Review

The Orienteering Problem (OP) [21], also known as the selective travelling salesperson problem [11] or traveling salesman problem with profits [3], has received increasing attention among researchers during recent decades. A comprehensive survey of OP can be found in [24]. An earlier work [3] also provided a survey of different classes of applications, modeling approaches and solution techniques.

[21] is the first to introduce a general description of the sport of orienteering and develop heuristic approaches based on a Monte Carlo technique for the OP. [9] introduced a new procedure which is based on four concepts: center of gravity, randomness, subgravity, and learning. Several metaheuristics for solving the OP have been proposed by researchers, such as Tabu Search [7], Genetic Algorithm (GA) [19] and Ant Colony Optimization [16].

Time-Dependent Orienteering Problem (TDOP) is an extension of OP by taking into account changes to the network over time. The travel time from one node

to another node varies with time and depends on the start time. It was first proposed by [5], where a $(2+\varepsilon)$-approximation algorithm is also proposed for solving it. However, [5] considers only equal utility for all nodes, thus the problem actually reduces to visiting as many nodes as possible rather than maximizing total collected utilities; besides, the proposed algorithm has not been empirically studied. A very recent work [25] proposed ant colony system for solving a variant of TDOP based on speed model, where the travel time between two nodes does not depend on the starting time but on the speed given at each time step during the travel. The speed for travelling between two nodes at a time step is assumed invariant under different starting time. This speed model is very interesting since it preserves the FIFO property: a vehicle that starts earlier will always arrive earlier. And this is practical in many real-world applications. However, this speed model may not hold in the general TDOP. For example, if the travel time between two nodes includes waiting for a shuttle that arrives according to a fixed time table, then it is not straightforward to compute the waiting time by the speed model due to the speed invariance. In this work, we follow the more general time-dependence defined in [5] that the travel time between two nodes depends only on the starting time, and these travel times are assumed to be given. Besides, no assumption of FIFO property is assumed from the input travel time data.

Other related work for TDOP includes [12] that proposed the Time-Dependent Team Orienteering Problem; [1] that proposed two genetic algorithms for the Time-Dependent Orienteering Problem with Time Windows which stems from the application of tour itinerary planning in complex and large urban areas in Tehran; [6] that presented the Time-Dependent Team Orienteering Problem with Time Windows which originated from the development of personalised electronic tourist guides by integrating the tourist planning problem and the use of public transportation, and two different approaches based on Iterated Local Search are proposed to solve a set of test instances based on real data for the city of San Sebastian, Spain.

3 Time Dependent Orienteering Problem

A graphical description of the Orienteering Problem (OP) [24] can be briefly introduced as follows. Given a set of nodes $N := \{1, 2, \ldots, n\}$, where node 1 is the starting point, and n is the end point; also given the utility of each node $U := \{u_i : i \in N\}$, the distance matrix $D := \{d_{i,j} : i, j \in N\}$ representing the travel time between any two nodes i, j, and a maximum travel time budget T_{max}, the objective is to find a path P that starts from node 1 and ends at node n before T_{max}, such that the total utility collected at all visited nodes in path P is maximized.

In the time dependent orienteering problem (TDOP), the travel time from nodes i to j depends on the time when the trip starts. Given k time horizons $H := \{h_1, h_2, \ldots, h_k\}$ with $h_i = \underline{h_i}, \underline{h_i}+1, \ldots, \overline{h_i}$, where the travel time within each horizon is constant $D := \{d_{i,j,h} : i, j \in N, h \in H\}$.

In our target practical application, theme park tour guidance, the travel time $d_{i,j,h}$ from node i to node j includes traveling from i to j, waiting time at j, and the service time at j. The length of $d_{i,j,h}$ mainly depends on the waiting time at attraction j. The high-utility attractions are usually preferred by most of the

Table 1 Parameters and decision variables

Notations	Descriptions
u_i	the utility score for node i
T_{max}	the time budget to leave node n; it also specifies the number of time steps.
$d_{i,j,t}$	the travel time from node i to node j, started at time period t.
X_{ijt}	$= 1$ if travel occurs from node i to node j at time period t; otherwise, 0

tourists, and thus usually have a long queue during busy hours. If our tour guidance is used by a small portion of the visitors, taking into account their attraction preference as utility value, and the historical data or real-time crowd statistics as prediction of the waiting time, an optimal tour is expected to maximize the total utility of the visited attractions, while avoiding visiting a popular attraction at its most crowded hour.

We formulate the TDOP as an integer linear programming (ILP) model. Table 1 presents parameters and decision variables required to formulate the ILP model.

$$Maximize \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{t=1}^{T_{max}} u_i \times X_{i,j,t} \tag{1}$$

The objective function (1) is to maximize the total collected utility score when visiting nodes at certain time periods.

$$\sum_{i>1}^{n} \sum_{t=1}^{T_{max}} X_{i,1,t} = 0 \tag{2}$$

$$\sum_{j>1}^{n} \sum_{t=1}^{T_{max}} X_{1,j,t} = 1 \tag{3}$$

Constraint (2) ensures that there is no return trip to the start point and constraint (3) ensures that the start point is node 1.

$$\sum_{j=1}^{n-1} \sum_{t=1}^{T_{max}} X_{n,j,t} = 0 \tag{4}$$

$$\sum_{i=1}^{n-1} \sum_{t=1}^{T_{max}} X_{i,n,t} = 1 \tag{5}$$

Constraints (4) and (5) ensure that the last visited node is node n.

$$\sum_{\substack{i=1 \\ i \neq e}}^{n-1} \sum_{t=1}^{T_{max}} X_{i,e,t} = \sum_{\substack{j=2 \\ j \neq e}}^{n} \sum_{t=1}^{T_{max}} X_{e,j,t} \; \forall e = 2, 3, \ldots, (n-1) \tag{6}$$

Constraint (6) guarantees the connectivity of the path for each node visited.

$$\sum_{\substack{j=2 \\ j \neq i}}^{n} \sum_{t=1}^{T_{max}} X_{i,j,t} \leq 1 \; \forall i = 2, 3, \ldots, (n-1) \tag{7}$$

Constraint (7) ensures that each node is visited at most once.

$$\sum_{e \neq i,j} \sum_{u=t+d_{i,j,t}}^{T_{max}} X_{j,e,u} \geq X_{i,j,t}; \forall i,j = 1,\ldots,n-1, i \neq j, j \neq 1, t \leq T_{max} - d_{i,j,t} \quad (8)$$

The constraint (8) enforces if a trip from i to j starts at time t, and j is not the end point, then a trip must start from j at a time period later than after visiting j.

$$X_{i,j,t} = 0, \forall i \neq j, t > T_{max} - d_{i,j,t} \quad (9)$$

(9) removes infeasible trips that starts too late.

4 Metaheuristics

As proved by Golden et al. in 1987 [8] that orienteering problem (OP) is NP-hard, i.e. no polynomial time algorithm could be designed to solve this problem to optimality. As a generalisation of the OP with time-dependent travel time, the TDOP is also NP-hard. The mathematical model introduced in Section 3 can be regarded as a time-expanded graph of the OP, which substantially increases the problem dimension, making computation even more challenging.

In this paper, our target application of the TDOP is to provide real-time tour guidance to theme park visitors, so it is practically infeasible to make the tourists wait minutes or even hours for an optimal route. Therefore, a fast and effective heuristic approach is essential for devising a practical theme park routing tool in a dynamic environment.

4.1 Greedy Construction Heuristic

A greedy construction heuristic is a myopic strategy that always chooses one solution component that is with the best immediate desirability based on a greedy criterion. In TDOP, A path P is initialized with the starting node 1 and end node n. Then, the construction proceeds iteratively by adding a solution component, in our case, one unvisited node, to P. In each iteration, the selected unvisited node is added to the end of the path, right before the end node n.

One important feature in the TDOP is to handle the time-dependent travel time $d_{i,j,h}$. Since travel time changes as the starting time changes, and no FIFO property is assumed, we are faced with the following non-trivial subproblems:

- EarliestArrival: given a starting time t^{start} to travel from node n^{prev} to current node n^{cur}, find the earliest arrival time t^{arr} at n^{cur}; and
- LatestDeparture: given an arrival time t^{end} at node n^{next}, find the latest departure time t^{dep} from node n^{cur}.

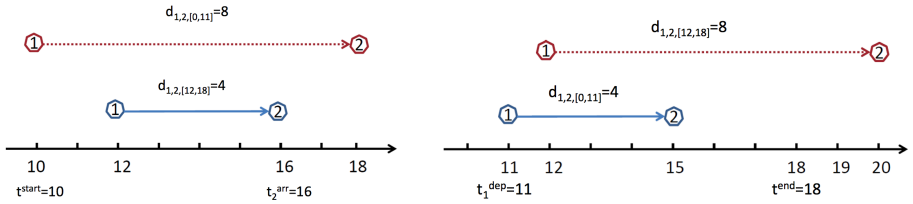
An example of these two subproblems can be illustrated in Figure 1. They are handled is as follows: EarliestArrival, as outlined in Procedure 1, iteratively checks the earliest possible arrival time of each proceeding time horizon, until a horizon surpasses current earliest possible arrival time; LatestDeparture in

Procedure 1 EARLIESTARRIVAL $(t^{start},\ n^{prev},\ n^{cur},\ D,\ H)$

$h^{cur} \leftarrow$ the time horizon that contains t^{start}
$t^{end} \leftarrow t^{start} + d_{n^{prev},n^{cur},h^{cur}}$
for all $h \in \{h^{cur}+1,\ldots,H\}$ **do**
  if $t^{end} < \underline{h}$ **then**
    break
  else
    if $\underline{h} + d_{n^{prev},n^{cur},h} \leq t^{end}$ **then**
      $t^{end} \leftarrow \underline{h} + d_{n^{prev},n^{cur},h}$
    end if
  end if
end for
return t^{end}

Procedure 2 LATESTDEPARTURE $(t^{end},\ n^{cur},\ n^{next},\ D,\ H)$

$h^{cur} \leftarrow$ the time horizon that contains t^{end}
$t^{start} \leftarrow t^{end} - d_{n^{prev},n^{cur},h^{cur}}$
for all $h \in \{h^{cur}-1,\ldots,1\}$ **do**
  if $t^{start} > \overline{h}$ **then**
    break
  else
    $t^{start} \leftarrow \min(t^{end} - d_{n^{prev},n^{cur},h}; \overline{h})$
  end if
end for
return t^{start}



(a) Starting from node 1 at time 10, the earliest arrival time at node 2 is 16.

(b) Requiring arrival at node 2 at time 18, the latest departure time at node 1 is 11.

Fig. 1 Example of finding the earliest arrival and latest departure time in time dependent travels. The travel time from node 1 to node 2 is 4 if it starts within time interval $[0, 11]$, and is 8 within [12, 18].

Procedure 2 on the contrary, iteratively checks backwards each time horizon until one horizon h in which a trip can be started, then the latest departure time is set to either $t^{end} - d_{n^{prev},n^{cur},h}$, or the horizon boundary $\overline{h}$, whichever starts first.

In each construction iteration, a node is feasible to be appended to the path only if its earliest arrival time computed by Procedure 1 is no later than its latest departure time computed by Procedure 2. From the set of all feasible nodes N^*, a best node n^* is then selected by the following greedy criterion:

$$n^* \leftarrow \arg\max_{i \in N^*} \frac{u_i^\alpha}{\delta_i^\beta} \cdot \text{rand}(\frac{1}{\gamma}, 1). \tag{10}$$

That is, the desirability of a node $i \in N^*$ depends on two terms: the utility value u_i, and the distance δ_i which is calculated as the difference between the earliest

Procedure 3 ForwardPropagation $(P,\ n_s)$

$t^{start} \leftarrow t^{arr}_{n_{s-1}}$
for all $i \in \{s, s+1, \ldots, m\}$ **do**
  $t^{arr}_{n_i} \leftarrow$ EarliestArrival$(t^{start}, n_{i-1}, n_i)$
  $t^{start} \leftarrow t^{arr}_{n_i}$
end for

Procedure 4 BackwardPropagation $(P,\ n_s)$

$t^{end} \leftarrow t^{dep}_{n_{s+1}}$
for all $i \in \{s, s-1, \ldots, 1\}$ **do**
  $t^{dep}_{n_i} \leftarrow$ LatestDeparture(t^{end}, n^i, n^{i+1})
  $t^{end} \leftarrow t^{dep}_{n_i}$
end for

arrival time at i and earliest possible leaving time at the previous node. α and β are parameters determining the impact of utility and distance, respectively. The rand$(\frac{1}{\gamma}, 1)$ is the noise term that generates a uniformly random number ranging from $\frac{1}{\gamma}$ to 1, where $\gamma \geq 1$ is a parameter that allows candidates of γ times worse than the best one to be selected. If γ is set to 1, it selects deterministically the most desirable node at each step. The greedy construction terminates when either the visitation time budget T_{max} is finished, or no more unvisited node can be appended.

4.2 Local Search and Variable Neighborhood Descent

Two types of basic local search operators are adopted in our work, the insert and replace operators. We also consider hybridizing the two operators within a variable neighborhood descent framework. In order to make the feasibility check of each operator more efficient, a starting time propagation procedure is used and described below.

4.2.1 Starting Time Propagation

Given a path P of m nodes, $P := \{n_i : i = 1, 2, \ldots, m\}$, the starting time propagation concerns assigning the earliest arrival time t^{arr}_i and the latest departure time t^{dep}_i for each node $n_i \in P$. It can be classified into two different procedures, the ForwardPropagation in Procedure 3 that propagates the earliest arrival time in the forward direction, and the BackwardPropagation in Procedure 4 that propagates the latest departure time in the backward direction. Note that both procedures can also propagate for a partial path, starting from a certain index n_s.

The ForwardPropagation procedure iteratively takes the earliest arrival time of the previous node to obtain the earliest arrival time of the current node by using the EarliestArrival procedure, while the BackwardPropagation procedure iteratively takes the latest departure time of the next node to compute the latest departure time of the current node by the LatestDeparture procedure.

Procedure 5 INSERT (P, N)

```
for all i ∈ random nodes of N do
  for all j ∈ random positions of P do
    t_i^arr ← EARLIESTARRIVAL(t_{n_j}^arr, n_j, i)
    t_i^dep ← LATESTDEPARTURE(t_{n_{j+1}}^dep, i, n_{j+1})
    if t_i^arr ≤ t_i^dep then
      insert i into path P at position j
      N ← N \ {i}
      FORWARDPROPAGATION(P, j + 1)
      BACKWARDPROPAGATION(P, j − 1)
      break
    end if
  end for
end for
```

4.2.2 Local Search Operators: Insert and Replace

The random first improvement strategy is applied for the insert operator, as outlined in Procedure 5. Each random unvisited node $i \in N$ is tried to be inserted between random position j and $j+1$ in the path P. The earliest arrival time of node i is computed based on the earliest arrival time at node j, and the latest departure time of node i is computed based on the latest arrival time of node $j+1$. If the node i can arrive earlier than its latest departure time, it is inserted at position j, and earliest arrival time of the nodes after j will be updated using ForwardPropagation and the latest departure time of the nodes before j will be updated using BackwardPropagation.

Similarly, the replace operator also uses a random first improvement strategy. An unvisited node i is considered better than node n_j at position j of the path P, either when its utility value is strictly better, or in the case of equal utility, the difference between its earliest arrival time and latest departure time is strictly larger. In such case, the two nodes are exchanged, and the starting time of the rest of the nodes are updated by propagation.

4.2.3 Variable Neighborhood Descent

The basic idea of Variable Neighborhood Descent (VND) [14] is to apply a set of local search operators iteratively, such that the final solution obtained is locally optimal with respect to all local search operators (subject to iteration order). In such a way, variable neighborhoods, such as insert and replace operators in Section 4.2.2, can be hybridized. Note that VND starts with a complete solution and returns a modified solution, hence itself can also be regarded as a local search operator.

4.3 Iterated Local Search

Iterated local search (ILS) [13] is a simple yet effective, general-purpose metaheuristic. It starts with an initial solution and a local search, and then iterate the three components of ILS: perturbation, local search, and acceptance criterion.

Procedure 6 PERTURBATION (P, N)

randomly remove s nodes from P
tabu the removed nodes for one local search iteration
FORWARDPROPAGATION(P, 1)
BACKWARDPROPAGATION(P, $|P|$)

Procedure 7 ACCEPTANCEBASIC (P, P^{gb}, P^{rb}, I^{max})

if Incumbent P is better than restart best P^{rb} **then**
  $P^{rb} \leftarrow P$
  if Incumbent P is better than global best P^{gb} **then**
    $P^{gb} \leftarrow P$
  end if
else
  $P \leftarrow P^{rb}$
  if consecutive non-improved iteration count exceeds maximum I^{max} **then**
    Restart by a greedy construction followed by a variable neighborhood descent
  end if
end if

Here, the initial solution is constructed by the greedy method in Section 4.1, the variable neighborhood descent in Section 4.2.3 is adopted as the subsidiary local search procedure. In the PERTURBATION procedure outlined in Procedure 6, s random nodes are removed from the incumbent path P. Note that a node can be removed only if the earliest arrival time at the next node is not delayed. This is usually not an issue in a Euclidean-distance graph, however, it may not hold if the travel time on an edge is time dependent due to traffic conditions as in some benchmark instances mentioned in Section 5.1. s is a parameter reflecting the perturbation strength. A high value of s may result in slow convergence, while a smaller value of perturbation strength may quickly lead to a good solution at the beginning but is more likely to be trapped in a deep local optimum. These s removed nodes are tabued for one local search iteration, so that they cannot be immediately inserted or replaced back to the incumbent path, allowing more diversification.

We have considered two versions of the iterated local search in this work, a basic version named Basic ILS and a modified version named Adaptive ILS. Their main difference lies in the acceptance criterion, or more precisely, in how to handle algorithm stagnation. The stagnation is referred to when the maximum number of non-improved iterations I^{max} is reached. In Basic ILS, the algorithm is restarted by a greedy construction and variable neighborhood descent, as detailed in Procedure 7. Adaptive ILS in Procedure 8 first increments the perturbation strength s, and resets the iteration counter to zero, until the maximum perturbation strength s^{max} is reached, then restarts the search. However, since Adaptive ILS allows more non-improved iterations before restart, if it is trapped in an uninteresting region, many iterations will be wasted. To this end, we developed a probabilistic intensification mechanism. For each unsuccessful iteration, with an intensification probability p^{in}, the best-so-far solution, instead of the restart best solution, will be copied into the incumbent.

Procedure 8 AcceptanceAdaptive (P, P^{gb}, P^{rb}, p^{in}, I^{max}, s^{max})

```
if Incumbent P is better than restart best P^rb then
  P^rb ← P
  Set perturbation strength to minimum s ← s^min
  if Incumbent P is better than global best P^gb then
    P^gb ← P
  end if
else
  if rand(0,1) < p^in then
    P ← P^gb        // Intensify by copying the global best solution to incumbent
  else
    P ← P^rb
  end if
  if consecutive non-improved iteration count exceeds maximum I^max then
    Increment perturbation strength s ← s + 1
    if s > s^max then
      Restart by a greedy construction followed by a variable neighborhood descent
    end if
  end if
end if
```

5 Computational Results

The comprehensive computational results including experimental setup will be described below.

5.1 Instance Setup

Three classes of time dependent orienteering problem (TDOP) instances are considered in this study:

- `Benchmark`. The benchmark instances are adopted from [22]. These instances were initially developed by [2]. The number of nodes in these instances varies from 21 to 102. These instances were further adapted by Verbeek et al. [25] by varying travel time at different time horizons. Each instance has a time span from 7 am to 9 pm. These 14 hours were divided into four different time horizons: 7 am to 9 am, 9 am to 5 pm, 5 pm to 7 pm, and 7 pm to 9 pm, since the travel time on each edge may depend on the traffic load at different period of the day. In order to use our time-expanded model for these benchmark instances, the original travel time is discretized by a unit of $\mu = 1, 5, 15, 30$ minutes, which corresponds to a total number of time steps $T_{max} = 840, 168, 56, 28$. In order to guarantee the feasibility, the discretization of a travel time d is by rounding up after divided by the time unit, $\lceil d/\mu \rceil$. The larger the time unit, the less the number of time steps, and thus, easier for the time-expanded model to be solved.
- `Random`. The second class of instances is randomly generated with varying values of the parameters: number of nodes n and time budget T_{max}. Each time step is considered a time horizon. The utility score for each node is generated randomly between 1 to 9. The start and end nodes are allocated with zero utility scores.

Table 2 Characteristics of problem instances and their optimal or best-known solution computed by CPLEX with the mathematical program model. The instance characteristics include number of nodes $|N|$, number of discrete time steps T_{max}, and number of time horizons k.

Class	Name	$\|N\|$	T_{max}	k	CPLEX Result Optimal	CPLEX Result Time
`Benchmark`	$TDOP$1-{1, 5, 15, 30}	32	{840, 168, 56, 28}	4	220‡	18.47 mins
	$TDOP$2-{1, 5, 15, 30}	21	{840, 168, 56, 28}	4	355‡	55.69 secs
	$TDOP$3-{1, 5, 15, 30}	33	{840, 168, 56, 28}	4	590‡	1.88 hours
	$TDOP$4-{1, 5, 15, 30}	100	{840, 168, 56, 28}	4	623‡*	>24 hours
	$TDOP$5-{1, 5, 15, 30}	66	{840, 168, 56, 28}	4	850‡	13.11 hours
	$TDOP$6-{1, 5, 15, 30}	64	{840, 168, 56, 28}	4	768‡	21.23 hours
	$TDOP$7-{1, 5, 15, 30}	102	{840, 168, 56, 28}	4	690‡*	>24 hours
`Random`	$Rand10 \times 10$	10	10	10	20	0.17 secs
	$Rand10 \times 20$	10	20	20	41	0.47 secs
	$Rand10 \times 30$	10	30	30	46	2.06 secs
	$Rand10 \times 40$	10	40	40	37	5.60 secs
	$Rand20 \times 10$	20	10	10	42	1.20 secs
	$Rand20 \times 20$	20	20	20	80	2.14 mins
	$Rand20 \times 30$	20	30	30	84	10.73 mins
	$Rand20 \times 40$	20	40	40	107	14.10 mins
	$Rand30 \times 10$	30	10	10	47	1.94 secs
	$Rand30 \times 20$	30	20	20	103	3.49 mins
	$Rand30 \times 30$	30	30	30	151*	>24 hours
	$Rand30 \times 40$	30	40	40	143*	>24 hours
	$Rand40 \times 10$	40	10	10	63	10.13 secs
	$Rand40 \times 20$	40	20	20	145	26.26 mins
	$Rand40 \times 30$	40	30	30	176*	>24 hours
	$Rand40 \times 40$	40	40	40	217*	>24 hours
`Real world`	$Real1$	17	36	9	195	15.3 mins
	$Real2$	40	42	42	208*	>24 hours

‡ optimal or best known solution for discrete time unit 30.
* best known solution obtained after 24 hours

– `Real world`. Two real-world instances are obtained from two of the most popular theme parks in Asia. Each node represents an attraction, the utility vector of each attraction is derived from user preferences data, and the travel time from one attraction to another includes the traveling time (by shuttle or on foot), and service time at an attraction, and the waiting time that varies over time.

The details of instance characteristics can be referred to in Table 2.

5.2 Computational Results of Mathematical Model

The mathematical programming model in Section 3 is solved by commercial solver CPLEX 10.2 on a computing server with multi-core Intel Xeon CPU ES-2667 at 2.90 GHz with 256GB RAM running Microsoft Server 2008 R2 Enterprise. Up to 24 threads are used per run.

The last two columns of Table 2 summarizes the optimal solution and computation time obtained by CPLEX for each instance. The computational scalability of the problem is best illustrated in the `Random` class of instances. Although most

instances can be solved to optimality, the computation time explodes quickly as the number of nodes and number of time steps increase. Instances with more than 30 nodes and 30 time steps cannot be solved to provable optimality within 24 hours. It takes minutes of computation time to compute an optimal route for instances from 20 nodes and 20 time steps. In the instance class `Benchmark`, CPLEX is only applied to instances discretized by time unit 30 minutes, resulting in 28 time steps in 14 hours.[1] Note that since the number of time horizons k is reduced to 4, the computational time required is also reduced noticeably. Here, instances with up to 66 nodes and 28 time steps can be solved to provable optimality within 24 hours. However, the computation time required is very long: It takes over 1 hour to solve instances from 33 nodes. Concerning the two `real world` theme park instances, the smaller one *Real*1 with 17 nodes and 36 time steps is solved to optimum in around 15 minutes, while the larger one *Real*2 cannot be solved to provable optimality within 24 hours.

Although most of the instances considered can be solved optimally by CPLEX, the computation time is unpractically long, usually minutes to hours for a realistic problem size. However, our target application is a time-critical problem: each instance is generated for each visitor on the fly based on their personal preference and available time, and then the tour guidance system is expected to compute a good solution within an acceptable time, i.e., maximum one second. Besides, the coarse time discretization required by the mathematical model also reduces the accuracy of travel time input. Therefore, the mathematical programming may not be an ideal approach for this application, however, the optimal or best known solution computed in this section can be a good reference in assessing the quality of our metaheuristic approaches in the next section.

[1] For instances with smaller discretization time units such as 15, 5, and 1 minutes, most of the instances cannot be solved to optimality within 24 hours.

Table 3 The computational results of the four metaheuristics: greedy, variable neighborhood descent (VND), basic iterated local search (Basic ILS), and adaptive iterated local search (Adaptive ILS). Each metaheristic is run 30 trials on each instance, and best, mean, and worst performance of each 30 runs are listed below. Percentage deviation from optimal or best-known solution is listed, where available in Table 2, or else the objective value is listed. Statistically significantly best results are marked in bold face.

Instance	Greedy			VND			Basic ILS			Adaptive ILS		
	Best	**Mean**	**Worst**	**Best**	**Mean**	**Worst**	**Best**	**Mean**	**Worst**	**Best**	**Mean**	**Worst**
$TDOP$1-1	280	273.8	270	280	280	280	285	**284.3**	280	285	**284.8**	280
$TDOP$1-5	270	264.7	260	280	273.8	270	280	**280**	280	280	**280**	280
$TDOP$1-15	245	238.7	235	260	251.8	245	260	257.5	255	260	**260**	260
$TDOP$1-30	2.27%	4.02%	6.82%	0.00%	1.29%	4.55%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%
$TDOP$2-1	450	437.7	430	450	**450**	450	450	**450**	450	450	**450**	450
$TDOP$2-5	430	422.0	415	440	430.7	430	440	**440**	440	440	**440**	440
$TDOP$2-15	385	376.3	375	395	**395**	395	395	**395**	395	395	**395**	395
$TDOP$2-30	0.00%	2.96%	5.63%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%
$TDOP$3-1	760	744.3	740	770	762.7	750	780	**778.3**	770	780	**777.7**	770
$TDOP$3-5	730	725.7	720	740	731	730	750	**742.3**	730	750	**744.3**	730
$TDOP$3-15	660	647.7	640	670	660.7	650	670	**665**	660	670	**663**	660
$TDOP$3-30	0.00%	0.90%	3.39%	0.00%	**0.06%**	1.69%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%
$TDOP$4-1	1014	980.2	964	1032	1003.3	989	1056	1032.2	1012	1085	**1048.5**	1015
$TDOP$4-5	935	910.9	891	956	927.2	909	964	942	921	989	**953.9**	925
$TDOP$4-15	791	777.1	761	824	791	776	831	800.8	771	842	**812.7**	783
$TDOP$4-30	2.25%	4.03%	5.78%	3.21%	4.20%	5.14%	1.93%	4.56%	5.94%	1.61%	**3.55**%	6.10%
$TDOP$5-1	1495	1458.2	1410	1495	1468.7	1425	1505	1488.8	1460	1505	**1497.3**	1465
$TDOP$5-5	1260	1233.8	1195	1260	**1238.2**	1200	1260	**1244.7**	1220	1260	**1246.8**	1215
$TDOP$5-15	865	851.7	835	870	854	845	870	859.3	845	870	**865**	850
$TDOP$5-30	0.59%	1.78%	2.94%	0.59%	1.47%	3.53%	0.00%	0.69%	1.76%	0.00%	**0.24**%	1.18%
$TDOP$6-1	1326	1303.4	1290	1326	1316.4	1302	1338	1328.8	1320	1344	**1337.6**	1332
$TDOP$6-5	1224	1186.4	1158	1236	1198.4	1176	1242	1221	1206	1254	**1237.0**	1212
$TDOP$6-15	1182	1153.2	1134	1206	1180.2	1164	1236	1211	1194	1254	**1233.6**	1206
$TDOP$6-30	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
$TDOP$7-1	1182	1164.6	1152	1232	1211.6	1192	1285	1260.1	1236	1318	**1283.2**	1253
$TDOP$7-5	1084	1065.2	1050	1113	1093.6	1078	1182	1140	1116	1195	**1159.1**	1134
$TDOP$7-15	960	940.9	920	1000	982.7	967	1026	1005.8	980	1046	**1019.8**	983
$TDOP$7-30	1.30%	1.82%	2.46%	0.58%	1.42%	2.03%	0.87%	1.46%	1.59%	0.58%	**1.14%**	1.59%
$Rand10 \times 10$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
$Rand10 \times 20$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
$Rand10 \times 30$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
$Rand10 \times 40$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
$Rand20 \times 10$	7.14%	7.14%	7.14%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%
$Rand20 \times 20$	0.00%	0.80%	1.27%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%
$Rand20 \times 30$	0.00%	0.16%	1.19%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%
$Rand20 \times 40$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
$Rand30 \times 10$	6.38%	6.38%	6.38%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%
$Rand30 \times 20$	3.88%	3.88%	3.88%	0.00%	1.62%	1.94%	0.00%	**0.45%**	0.97%	0.00%	**0.58%**	1.94%
$Rand30 \times 30$	3.29%	4.30%	4.61%	1.32%	2.11%	3.29%	0.00%	**1.64%**	2.63%	0.00%*	**1.36%**	2.63%
$Rand30 \times 40$	0.00%	0.40%	0.70%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%	0.00%	**0.00%**	0.00%
$Rand40 \times 10$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
$Rand40 \times 20$	0.00%	**0.62%**	0.69%	0.00%	**0.67%**	0.69%	0.00%	0.74%	1.38%	0.00%	0.83%	2.07%
$Rand40 \times 30$	0.56%	1.98%	3.95%	0.56%	1.53%	2.26%	0.56%	1.54%	2.82%	0.00%‡	**1.21%**	2.26%
$Rand40 \times 40$	2.30%	2.69%	3.23%	0.46%	1.03%	1.38%	0.46%	0.89%	1.38%	0.00%	**0.68**%	1.38%
$Real1$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
$Real2$	0.96%	1.54%	1.92%	0.00%	**0.19%**	0.48%	0.00%	**0.18%**	0.48%	0.00%	**0.14%**	0.48%

* New best known solution found: 152.

‡ New best known solution found: 177.

5.3 Computational Results of Metaheuristics

The metaheuristics introduced in Section 4 were implemented in Java, compiled by JDK 7, and run on a MacBook Air with 1.7GHz Intel Core i7 and 8GB memory. Only single thread is used for each metaheuristic run.

Four metaheuristics are considered here: a restart greedy append construction heuristic (Greedy), a restart variable neighborhood descent (VND), a basic version of iterated local search with restart (Basic ILS), and a modified iterated local search with adaptive perturbation size and probabilistically intensified restart (Adaptive ILS). The following parameter settings are adopted for these algorithms. For greedy construction, the weight of utility and distance is set to $\alpha = \beta = 1$ following most of the existing work on orienteering problem; the noise factor γ is set to 1 in the first run to make it deterministic, and set to 5 in all the following runs. For Basic ILS, the perturbation strength s is set to 3, and the maximum non-improved iteration I^{max} is set to 10. For Adaptive ILS, I^{max} is also set to 10, and the minimum perturbation strength s^{min} is set to 3, and the maximum $s^{max} = \lceil |P|/4 \rceil$, i.e. the roundup of one fourth of the incumbent path size; the intensification probability p^{in} is set to 0.05.

Each metaheuristic is allowed a maximum runtime of one second, and is performed 30 independent runs on each instance. The best, mean and worst performance of 30 runs are recorded in Table 3. For each instance, the significantly best performing metaheuristics by the Wilcoxon's signed rank test at 0.05 level is marked in bold face. As is clearly shown, Adaptive ILS is the significantly best performing algorithms for all benchmark instances and real world instances. Especially in the largest benchmark instances $TDOP4$ and $TDOP7$, Adaptive ILS is usually over 1% better than the Basic ILS, and around 2 to 5% better than VND and greedy. It remains the significantly best performing algorithm for all but one random instances, where it was slightly but statistically significantly outperformed by greedy and VND. Basic ILS as runner-up significantly outperforms VND, which in turn performs significantly better than greedy.

Comparing the metaheuristic approaches with the mathematical programming approach, regardless of only one second computation time, Adaptive ILS is able to improve the best known solutions of random instances $Rand30 \times 30$ and $Rand40 \times 30$ that are computed by CPLEX 10.2.0 solver in 24 hours. The best run of Adaptive ILS has found the optimum of all random instances, real world instances, and the benchmark instances up to size 66. It misses the optimum of the largest instances $TDOP4$-30 and $TDOP7$-30 composed of 100 nodes, leaving a gap of 0.6 to 1.6 % at its best run, or 1.1 to 3.6% at its average. One second is probably too short for instances of such size. Considering mean performance, for instance size under 30, Adaptive ILS finds the optimal solution in each of the 30 runs; the average deviation from the optimum is less than 1.4% for the random instances up to size 40, 0.14% for the real world instances, and 0.24% for the benchmark instances up to size 66.

Another important advantage of the metaheuristic approaches over the time-expanded mathematical model is that, it does not require a time discretization, and thus can use the travel time of arbitrary accuracy. Since the travel time is discretized by rounding up to guarantee feasibility, it compromises the quality of the obtained solution. Comparing the result of benchmark instances with time unit of 1 minute in Table 3 to optimal result with time unit 30 minutes in Ta-

ble 2, the solution quality loss is from over 20% as on instance $TDOP2$ with size 21, to around 45% on instance $TDOP5$ with size 66. A finer time discretization than 30 minutes suffers the time-expanded mathematical model, however, using a coarser discretization at the expense of solution quality also demeans the original motivation of using mathematical model, which is to guarantee optimality. The metaheuristic approaches explored in this work, especially the Adaptive ILS, appear to be practical and promising for our application problem.

6 Conclusion

This paper presents a general form of the Time-Dependent Orienteering Problem (TDOP). We formulated this problem by an integer linear programming (ILP) model based on time-expanded graph. We further adopted two real-world instances from two most popular theme parks in Asia, together with modified benchmark instances, and randomly generated instances to study the scalability. The computational difficulty turns out to explode quickly for a commercial ILP solver as problem size increases.

As the underlying application problem is time critical, the development of a good metaheuristic is essential. Several heuristics are developed. From experimental results, we showed that our proposed approach, iterated local search with adaptive perturbation size and probabilistic intensified restart, appears to be fast and effective: within one second's computation time, it manages to find the optimal solution for most of the instances considered. It even improves for two instances the best known solutions computed by CPLEX for over 24 hours.

An interesting idea to extend our current ILS is to consider a hierarchical iterated local search approach [10], as well as using automatic algorithm configuration tool to determine the algorithm setting. It will be also interesting to compare with some state-of-the-art approaches such as ant colony systems [25]. From mathematical programming point of view, it would be interesting to consider further Branch-and-Cut techniques as in [4]. In the application aspect, our future research includes extracting more instances from our real world theme park tour guidance data, and extending the applicability of our approach to other types of real-world variants including time-dependent utility score, and the time-dependent team orienteering problem.

Acknowledgements This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA). Zhi Yuan acknowledges support by BMBF Verbundprojekt E-Motion (grant number 05M13GBA).

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. Chao, I.M., Golden, B.L., Wasil, E.A.: Theory and methodology - the team orienteering problem. European Journal of Operational Research **88**(3), 464–474 (1996)
3. Feillet, D., Dejax, P., Gendreau, M.: Traveling salesman problems with profits. Transportation Science **39**(2), 188–205 (2005)

4. Fischetti, M., Salazar, J.J., Toth, P.: Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing **10**, 133–148 (1998)
5. Fomin, F.V., Lingas, A.: Approximation algorithms for time-dependent orienteering. Information Processing Letters **83**, 57–62 (2002)
6. Garcia, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., Linaza, M.T.: Integrating public transportation in personalised electronic tourist guides. Computers and Operations Research **40**(3), 758–774 (2013)
7. Gendreau, M., Laporte, G., Semet, F.: A tabu search heuristic for the undirected selective travelling salesman problem. European Journal of Operational Research **106**(2-3), 539–545 (1998)
8. Golden, B., Levy, L., Vohra, R.: The orienteering problem. Naval Research Logistics **34**(3), 307–318 (1987)
9. Golden, B., Wang, Q., Liu, L.: A multifaceted heuristic for the orienteering problem. Naval Research Logistics **35**(3), 359–366 (1988)
10. Hussin, M.S., Stützle, T.: Hierarchical iterated local search for the quadratic assignment problem. In: M. Blesa, et al. (eds.) Proceeding of Hybrid Metaheuristics (HM 2009), *Lecture Notes in Computer Science*, vol. 5818, pp. 115–129. Springer (2009)
11. Laporte, G., Martello, S.: The selective travelling salesman problem. Discrete Applied Mathematics **26**(2-3), 193–207 (1990)
12. Li, J.: Model and algorithm for time-dependent team orienteering problem. In: S. Lin, X. Huang (eds.) Communications in Computer and Information Science, *Communications in Computer and Information Science*, vol. 175, pp. 1–7 (2011)
13. Lourenço, H., Martin, O., Stützle, T.: Iterated local search. In: Handbook of metaheuristics, pp. 320–353. Springer (2003)
14. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research **24**(11), 1097–1100 (1997)
15. 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)
16. Souffriau, W., Vansteenwegen, P., Berghe, G., Oudheusden, D.V.: Automated parameterisation of a metaheuristic for the orienteering problem. In: C. Cotta, M. Sevaux, K. Sörensen (eds.) Adaptive and multilevel metaheuristics, *Studies in Computational Intelligence*, vol. 136, pp. 255–269 (2008)
17. Souffriau, W., Vansteenwegen, P., Vertommen, J., Berghe, G.V., Oudheusden, D.V.: A personalised tourist trip design algorithm for mobile tourist guides. Applied Artificial Intelligence **22**(10), 964–985 (2008)
18. Tang, H., Miller-Hooks, E.: A tabu search heuristic for the team orienteering problem. Computer and Operations Research **32**(6), 1379–1407 (2005)
19. Tasgetiren, M.: A genetic algorithm with an adaptive penalty function for the orienteering problem. Journal of Economic and Social Research **4**(2), 1–26 (2001)
20. Thomadsen, T., Stidsen, T.: The quadratic selective travelling salesman problem. Informatics and mathematical modelling technical report IMM-Technical Report-2003-17, Technical University of Denmark (2003)
21. Tsiligirides, T.: Heuristic methods applied to orienteering. Journal of the Operational Research Society **35**(9), 797–809 (1984)
22. Vansteenwegen, P.: TDOP Format http://www.mech.kuleuven.be/en/cib/op/#section-20 (2013)
23. Vansteenwegen, P., Souffriau, W., Berghe, G.V., van Oudheusden, D.: Iterated local search for the team orienteering problem with time windows. Computers and Operations Research **36**(12), 3281–3290 (2009)
24. Vansteenwegen, P., Souffriau, W., Oudheusden, D.V.: The orienteering problem: A survey. European Journal of Operational Research **209**(1), 1–10 (2011)
25. Verbeeck, C., Sörensen, K., Aghezzaf, E.H., Vansteenwegen, P.: A fast solution method for the time-dependent orienteering problem. European Journal of Operational Research **236**(2), 419–432 (2014)