International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 07 | July 2022 www.irjet.net p-ISSN: 2395-0072

Multi-Objective Trip Planning With Tourist Attractions, Restaurant Selection And Public Transportation Facility Details

Ajith S Kumar[1]

Student, Dept. Of Computer Science and Engineering, Indira Gandhi Institute Of Engineering And Technology, APJ Abdul Kalam Technological University, Kerala, India

---

Abstract - *The Trip Planner web application helps the trip planners, such as tourists, tour companies, and government agencies, to plan their trip with user priorities. Trip planner application sequences an optimal set of point of interest (POIs) and then generates a planned route that maximizes their pleasure. However, the traditional Tourist Trip Design Problems (TTDP) does not include the break period at a local restaurant, which causes the rest of the itinerary in the afternoon to shift. Moreover, as tourism contributes to high greenhouse gas emissions, especially from its transportation, minimizing the itinerary's total distance is also considered. Unfortunately, this objective conflicts with the profit scores, So I included the public transportation facilities in the region. The tourists can choose their medium of transportation and if possible they can select some of the public transportation facilities in the area. To address the real-world issues in the itinerary selection, I formulate a new variant of the well-known orienteering problem with time windows (OPTW) called the multi-objective orienteering problem with TimeWindows, Restaurant Selection, and Compulsory POIs (MOPTW-RSCP). The proposed problem is provided with a mathematical formulation and two exact algorithms for solving them, i.e., greedy and branch-and-cut Pareto-based techniques.*

Key Words: (Size 10 & Bold) Key word1, Key word2, Key word3, etc (Minimum 5 to 8 key words)…

1.INTRODUCTION

Travellers visit tourist destinations for a limited time period, and it is not possible to visit all the attractions in an area. Therefore, they need an itinerary with option to select POIs based on their preferences and constraints, So tourists can make an itinerary with required POIs for a convenient time period.

The tourist trip design problem (TTDP) is proposed to describe the issue. It takes a list of POIs, a POI-to-POI distance table, and a tour period as the inputs. Each POI has a profit score and a visiting period, these profit score describes a satisfying level in the POI. Therefore, the solution of TTDP is a planned route having a subset of POIs together with the highest total profit that satisfies the tour period. Many TTDPs also consider the time window of each POI (Operational hours) as a constraint to reflect the real world. However, making a most profitable travel route is a complex, too long, and time-consuming process.

TTDP is a very complex problem, and the requirements may be varying. Several modified models of TTDP have been proposed in the past decade to achieve real-world requirements. During the literature review, I found variants of TTDP models that attempted to meet the real world scenarios, but there were some limitations. Firstly, most of them are avoiding the lunch period. The ignorance of this time period may cause the whole POIs to visit in the afternoon to be forced to shift. In addition, the user preference of restaurant selection is not considered. Moreover, Some POIs are usually put in the itinerary by the tourist's personal preferences or the recommendation from other sources. Unfortunately, most of the existing TTDP models do not support these compulsory POIs in the itinerary.

A walking tour has its characteristic that the long total distance to be covered in a walking tour may cause the tourists to be tired. So a trade-off between the total profit and distance is needed. However, the suitable amount of distance varies from person to person and from time to time. In other words, in any situation there is no single suitable total distance that suits for a tourist. Hence, the total distance should be considered as important as the total profit. Therefore, the multi-objective optimization model should able to consider both aspects equally.

Multi-objective optimization model is not only providing a single best itinerary but a set of non-dominated itineraries that let the trip planners choose the most appropriate one. However, If there are no preferences available, a solution ranking technique can support the decision.

My idea proposes a multi-objective orienteering problem with time windows, restaurant selection, and compulsory POIs (MO-OPTW-RSCP) based on user preferences to solve the problems above. Following are the main contributions of this paper.

1. A real-world MO-OPTW-RSCP model is formulated and introduced for the first time. The model supports the real-world issues by including

lunchtime with restaurant selection and compulsory POIs.

2. Two Pareto-based multi-objective algorithms, branchand-cut and greedy-based, are proposed for solving the model.

3. A solution ranking technique to help trip planners choose outstanding itineraries from the Pareto front containing a set of Pareto-optimal solutions.

2. METHODOLOGY

2.1 Background And Related Works

2.1.1 Orienteering Problem

Orienteering problem is a combination of node selection and determining the most effective path between the selected nodes. The tourist trip design problem (TTDP) can be viewed as an application of the orienteering problem (OP). The OP was originated from a sport in which competitors can choose to visit the places with various profit scores in the forest. The one who can collect the highest total profit from a subset of locations within a limited period will win. In 1984, Tsiligirides did the mathematical formulation of the problem. The problem is an NP-hard problem. Several well-known variants of the problem have been modelled, including OP with time window (OPTW), team OP (TOP), time-dependent OP (TDOP), team OPTW (TOPTW), TDOP with time window (TDOPTW), and team TDOPTW (TD-TOPTW).

2.1.2 OP for TTDP

TTDP can be mapped to the OP by replacing visiting places with POIs and competitors with tourists. Furthermore, to make the planned route more practical, each POI has its own open and close time as time window. Hence, the most popular models of TTDP are based on OPTW for single-day itineraries and TOPTW or multi-day itineraries. In this project, I will focus only on the OPTW for a single day travel. TTDP itself has several specified models which have been proposed to capture some unique real-world aspects. For example, Divsalar, presented OP with hotel selection (OPHS) to choose the hotel that makes the cost of traveling lowest . Having lunch for tourists is also vital for many trip planners to specify a restaurant in the itinerary explicitly for reasons such as the preferences of the tourists, beliefs and religions, and limitations such as food allergy. The original base paper of TTDP by Vansteenwagen and Oudheusden has also considered having lunch break in a restaurant. Then, Vansteenwegen formalized the concept by designing the lunch break by enabling virtual POIs without locations. Tenemaza assigned the lunch break at the current POI, assumed that every POI has a place to eat, and allowed the current POI to overlap the lunch break, then optimized for minimizing that overlap period. Finally, Exposito proposed the model TTDP with clustered POIs (TTDP-clustered) that separate POIs into several categories, including a restaurant. The model supports restaurant selection and lunch break by specifying the constraints of the restaurant category. Some POIs are preferable compared to the remaining. For example, some point of interests highlight that everyone should visit first-time tourists, or some POIs match the trip planner's preferences. Consequently, some POIs are compulsory POIs so they must be included in the planned route. The concept was introduced into OP by Gendreau. Palomo-Martínez model the OP with mandatory visits and conflict (OPMVC). The model supports both compulsory and conflicted POIs. If conflicted POIs are there only one can be selected. For example, only one palace can be added to the itinerary. In the same year, Palomo-Martínez solved the orienteering problem by using VNS and GRASP. Li and merged OPTW with compulsory POIs and modelled it as the orienteering problem with compulsory POIs and time windows (OPCNTW). Lin and Yu introduced the compulsory POIs to the TOPTW model and named it TOPTW with mandatory visits (TOPTW-MV).

2.1.3 Multi-Objective OPTW

The orienteering problem with time windows (OPTW) deals with the problem about selecting a set of points of interest and then finding the route to visit the POIs under the particular time window. But some models having more than one objective to be optimized. Then, they are classified as multi-objective optimization models with more than one goal. Popular techniques to address this problem are to merge multiple objectives and create a single objective, hierarchical ordering of the objectives, or using a Pareto-based method. The first two techniques may lead in to a single best solution, whereas the last technique may give multiple non-dominated solutions. In my approach, I only consider the Pareto-based model. The Pareto-based multi-objective optimization is based on Pareto dominance which works like If we have two itineraries itinerary A and B, then if itinerary A has at least one objective superior to itinerary B with no worse objective, then I can say that itinerary A dominates itinerary B, and itinerary B may eliminated. When I did pair wise tournament comparisons, only the non-dominated itineraries are left. They are the solutions to the problem, which can be considered as a Pareto front. Pareto front the plot with the axes are objectives. Fig. 2.1 represent the concept of Pareto front for a specific TTDP, which has two objectives: first one is the total profit (the higher, the better) and the second one is the total distance (the lower, the better). The Pareto dominance's concept of an itinerary is demonstrated in the figure as the dominance area formed by an itinerary and the most unsuccesfull point at the top-left corner of the graph. Eliminate all itineraries within the region. Hence, only two solutions, non-dominated itineraries, are left and it forms a Pareto front. Srinivas and Deb come up with the concept of the Pareto front. Since that time, it has become a research focus. Nowadays, so many

state-of-the-art algorithms have been proposed including VEGA, MOEA/D, and NSGA-II. Objectives for TTDP may be varied in these. One of the most well-known objectives is having multiple scores for each POI, such as level of interest, entrance fees, relaxation, readiness, etc. Schilde proposed a generic bi-objective POI scores model for orienteering problem with the result as a Pareto front. The papers contributed on this type of problem are Martí, Purevsuren, Rezki and Aghezzaf, Aghezzaf, and Rezki, Martín-Moreno and Vega-Rodríguez Rezki and Aghezzaf. Chenand Hu proposed the [T]OPTW variant of this type of problem. By using a local search, minimizing the total distance of the trip may be implicitly achieved. However, bringing the trade-off itineraries with varying satisfaction total profit (maximizing) and total distance (minimizing) will help the trip planner to balance. Unfortunately, there are only a few papers that address the problem. They are Karimi and Bashiri, Hapsari and Falco applied TOPTW with five objectives: maximize profit score, time of the tour, minimize the total distance number of preferred POIs of the tourists, and number of recommended POIs. On the other hand, Mirzaei attempted to balance the satisfaction level for each day. Hence for a TOP model, the objectives are to maximize the score, minimize the difference between the highest and lowest scores. Table 2.1 presents the summary of the related works and the proposed model. I am focused on these four points: restaurant selection, compulsory POIs, lunch period, and bi-objective of total profit and distances. The table reveals that none of the existing multi-objective models address the compulsory POIs, restaurant selection, or lunch period. Hence, the proposed model is proposed to fill in these research gaps

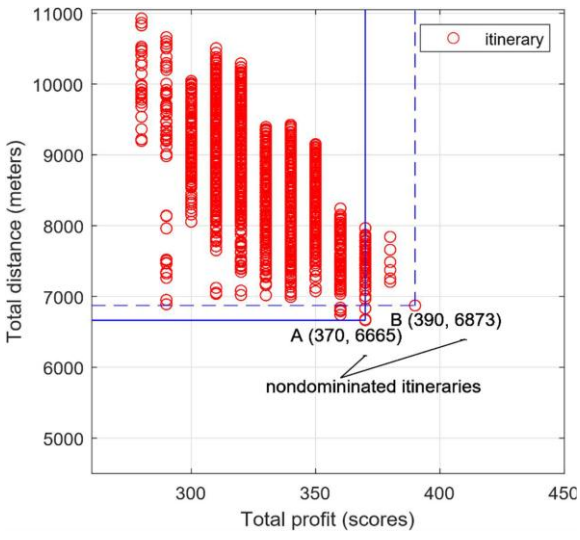

Figure 2.1 The concept of the Pareto front.

Base model	references	Lunch period	Restaurant	Compulsory	Objective 1	Objective 2
OP	1998 [14], 2017 [15], 2017 [16]			✓	P	
	2009 [23], 2015[24], 2015 [25], 2017 [26], 2018 [27], 2018 [28], 2020 [29]				P_1	P_2
OPTW	2007 [4]	✓			P	
	2011 [17]			✓	P	
	2016 [30]				P_1	P_2
	The proposed model	✓	✓	✓	**P**	**(D)**
TOPTW.	2011 [11]	✓			P	
	2019 [13]	✓	✓	✓	P	
	2015 [34], 2016 [35]				*	*
	2018 [31]				P_1	P_2
	2018 [32], 2019 [33]				P	(D)

The works with blanked objective2 are single-objective and an objective in parenthesis is to minimize and otherwise is to maximize
P, P_1, P_2 , and D denote total profit, profit1, profit2 and distance, respectively.
* The two papers have five objectives.

Table 2. 1 Summary of related works and the proposed model.

2.1.4 Itinerary Selection From Pareto Front

From the given Pareto front, the trip planners may select only one outstanding itinerary to implement, which depends on the trip planners route preference. The two obvious preferences available in route planning are the itinerary with minimum distance and maximum profit. The first one is the itinerary that having maximum total profit which is often selected when the tourists focus on exploring as many as possible highlighted POIs, near to the target destination, especially when visiting for the first time. Moreover, If the tourists travel with travel agents they sometimes organize familiarization trips on almost all possible highlighted POIs. Then, the travel partners can design an itinerary that matches the customer expectations. The second one is when the tourists consist of elders or young children in tours the shortest total distance is often selected.

Virtually all itineraries on the Pareto front dominate the inferior itineraries, i.e., they are non-dominated itineraries, so Selecting an outstanding itinerary from a Pareto front when no preference is available is not an important task. Sanyapong Petchrompo did a clustering technique by a cut-off technique and discovered an outstanding solution for each cluster. However, many researchers agree on knee solution which is an outstanding, well-balanced trade-offs and diversity solution. The knee solutions concept can be depicted for a convex shape of a Pareto front of both objectives: distance and total profit Fig. 2.2. The figure illustrates Pareto front with three knee itineraries.

The three most popular techniques for finding knee solutions are the maximum distance from the hyperplane focus, utility-based focus, and angle-based focus. The maximum distance from the hyperplane focus was firstly described by Das. First, Das defined a hyperplane that connects the extreme solutions. Then, the knee solution is the solution with the most extended length from itself perpendicular to the hyperplane. This technique is the basic for several knee searching algorithms.

E Branke introduced the utility-based focus based on the trade-off of objective functions in his research "Parallel Problem Solving From Nature". The knee solution has a small gain of an object that causes the highest loss of the other objective.

E Branke also seeded the angle-based focus. The typical Pareto dominance can be geographically viewed as the area of domination to be rectangular shaped, so it is the right angle at the solution. However, Scope of domination is extended if the angle is more expansive. Therefore, some non-dominated solutions are eliminated, and the rest are considered as knee solutions. The knee solution acts as decision support, not a mandatory itinerary to select in the real world Choachaicharoenkul and Wattanapongsakorn provide the angle-based focus solution ranking that will rank the solutions on a Pareto front to empower the trip planners.

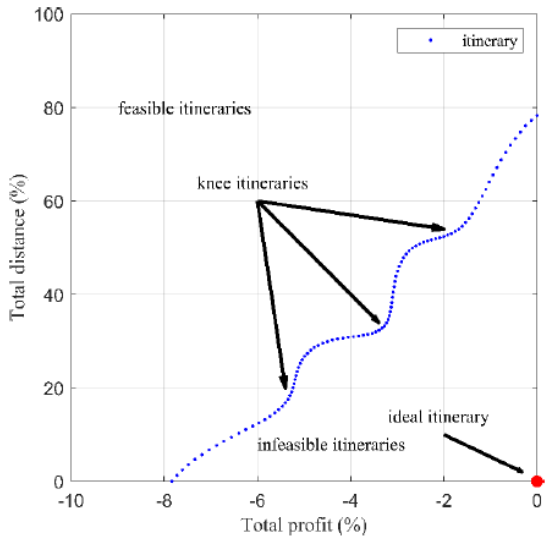

Figure 2.2 Knee itinerary concept

2.2 Problem Description And Formulation

This section proposes the extension of the Orienteering Problem with Time Windows (OPTW) called Multi-objective Orienteering Problem with TimeWindows, Restaurant Selection, and Compulsory POIs (MOPTW-RSCP). I focus OPTW on the TTDP application for a one-day trip with restaurant selection. MOPTW-RSCP extends the OPTW in below aspects.

Firstly, the model has two objectives: minimizing the total distance and maximizing the total profit of the selected POIs in the itineraries. So, the model becomes a multi-objective. However, the objectives conflict, as the usual characteristic of the multiobjective model. In this particular case, the total profit conflicts with the total distance; therefore, both objectives can't satisfy simultaneously in any single itinerary. Set of trade-offs between non-dominated itineraries will be the solution to this problem. If a tourist focuses on the shorter distance, the profit tends to be higher and vice versa. The solution set can be visualized as a Pareto front, which an objective is represented by each axis. Typically, the trip planners need only a single planned route from the solution according to their requirements.

Secondly, the model includes an explicit lunch break and a small set of restaurants in the area. The list of restaurants is filtered according to the POIs selected.

Thirdly, the model create itineraries based on the compulsory POIs. The compulsory POIs are the highlighted POIs that need to be traversed in between source and destination.

Finally, the model included public transportation facility details in the area as part of CO2 emission control.

2.2.1 Mathematical Model

Assuming that n attractions are there in between the source and destination, these attractions are separated into three types: point of interest (POI), restaurant, and start and end locations. Then, the attractions are encoded into an array that can be represented in Fig. 2.4. *Sp* and *Ep*, *Sr* and *Er* , and *Sh* and *Eh* keep track of the start and stop indexes of POIs, restaurants, and start and end locations, respectively. The meaning of symbols are mentioned in the two tables, table 2.2 for the list of parameters and table 2.3 for the list of decision variables. There are two objectives in MOPTW-RSCP, one for maximizing total profit and the other for minimizing the total distance that can be represented as follows

$$\text{Maximize} \sum_{i=S_p}^{E_p} \sum_{j=s_p}^{E_p} P_j x_{i,j} \quad (1)$$

$$\text{Minimize} \sum_{i=S_p}^{E_p} \sum_{j=s_p}^{E_p} D_{i,j} x_{i,j} \quad (2)$$

$$\text{subject to} \sum_{j=S_p}^{E_p} x_{S_D,j} = \sum_{i=S_p}^{E_p} x_{i,E_D} = 1 \quad (3)$$

$$\sum_{i=S_p}^{E_r} x_{i,k} = \sum_{j=S_p}^{E_r} x_{k,j} \leq 1; \quad \forall k = S_p, \ldots, E_r \quad (4)$$

$$O_i - W_i \leq a_i \leq C_i - J_i; \quad \forall i = S_P, \ldots, E_h \quad (5)$$

$$a_{S_d} = T_{min} \quad (6)$$

$$a_i + V_{i,j} - a_j \leq \infty \times (1 - x_{i,j}); \quad \forall i, j = S_p, \ldots, E_h \quad (7)$$

$$T_{min} + \sum_{i=S_P}^{E_D} \sum_{j=S_P}^{E_D} V_{ij} x_{i,j} \leq T_{max} \quad (8)$$

$$\sum_{i=S_p}^{E_p} x_{i,k} = \sum_{j=S_p}^{E_p} x_{k,j} = 1; \quad \forall k = S_r, \ldots, E_r \tag{9}$$

$$\sum_{i=S_r}^{E_r} x_{i,k} = \sum_{j=S_r}^{E_r} x_{k,j} = 0; \quad \forall k = S_r, \ldots, E_r \tag{10}$$

$$\sum_{i=S_p}^{E_h} x_{i,M_k} = \sum_{j=S_p}^{E_h} x_{M_k,j} = |M|; \quad \forall k = 1, \ldots, |M| \tag{11}$$

$$\sum_{i=S_p}^{E_h} x_{i,i} = 0 \tag{12}$$

Constraint (3) makes sure that the itinerary starts at the start location and ends at the end location. Each place must visit only once, constraint (4) forces the chance to include a POI at most once into the itinerary. Each POI has a time window and open time, Constraint (5) states that the arrival time to each POI in the itinerary must be within that time window or within z1 minutes before the open time of the POI. Constraints (6) sets the start time of the tour at *Tmin*. Constraint (7) ensures the validity of path between two attractions if the path is included in the itinerary. Constraint (8) provides that the itinerary is within the time budget, we need itineraries for a single day . Constraint (9) and (10) have the scope to modify in the set of equations it ensure that the itinerary must include exactly one restaurant. Constraint(11) provides that the itinerary comprises of all compulsory POIs. Finally, constraint (12) make sure that no POI is connected to itself.

Symbol	Meaning
S_p, E_p	Start and end index of POIs
S_r, E_r	Start and end index of restaurants
S_h, E_h	Start and end index of depots
T_{min}, T_{max}	Start and end time of the tour
L_{min}, L_{max}	Stand and end of lunchtime
b, e	Attraction index; $b, e = S_p, \ldots, E_h$
N_p	Number of POIs.
N_m	The number of compulsory POIs.
N_a	The number of attractions.
$W_{S_p}, \ldots, W_{E_h}$	The amount of waiting time allowed before the open time of the attraction
$P_{Sp}, \ldots, P_{Ep}$	The profit score of each POI
$O_{Sp}, \ldots, O_{Ep}$	Start operating time of each POI
$C_{Sp}, \ldots, C_{Ep}$	Close operating time of each POI
$J_{Sp}, \ldots, J_{E_h}$	Time used in each attraction
$M_1, \ldots, M_{N_m}$	Compulsory POIs
$V_{b,e}$	Travel cost from attraction b to attraction e (minutes)
$D_{b,e}$	Distance cost from attraction b to attraction e (meters)

Table 2.2 Parameters

Symbol	Meaning
$x_{b,e}$	One if attraction from attraction b to attraction e is included in the itinerary or zero otherwise
$a_{S_p}, \ldots, a_{E_d}$	Arrival time at each attraction

Table 2.3 Decision variables

2.2.2 Case Study Area And Dataset

The research is intended to match as much as possible real-world problems. So, I selected certain regions of Kerala, India. Kerala is one of the most magnificent tourist destinations in the world.

I collected 10 datasets with nearly 30 POIs. Then, I seek the help of a tour company to fill up the times used in minutes and profit scores in the range of 1 to 100 based on popularity among tourists for each selected POI. I also collected most preferable restaurants in the 10 selected areas by locals as well as tourists. Finally, I collect detailed description of the POIs and available public transportation details in the area to give the tourists more possibilities in travel.

As a result, the map with POIs in the districts Ernakulam, Kottayam, Alappuzha, and Kollam is shown in Fig. 2.3. From the figure, the red location symbol denotes a POI, where the camera icon with big location symbol denotes POIs having more profit score: Kumarakam bird sanctuary, Thottappally beach Alappuzha, Munroe island etc.

The idea in the project is to plan a single day itinerary; the region proposed having POIs to cover for a week, so I created 10 different datasets with 2 or 3 combinations of POIs. So the user can select whichever source and destinations from the 10 source and destination combinations from the region and these 10 datasets have different combinations of POIs.

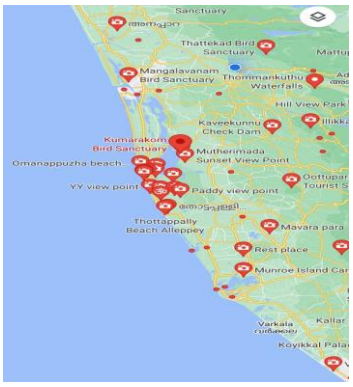

Figure 2.3 The POI map of a certain region of Kerala.

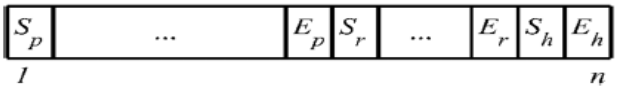

Figure 2.4 The structure of attractions.

3. PROPOSED ALGORITHMS

I proposed two algorithms for this work: branchand-cut and a greedy algorithm. The branch-and-cut algorithm guarantees the optimal itineraries, but computational complexity is high. At the same time, the greedy algorithm can produce high-quality itineraries, not always the optimal solutions, though, but the computational complexity is much lower.

3.1 Branch-And-Cut Optimization Algorithm

The branch-and-cut algorithm have a straightforward concept; it is a tree-based, recursive, and branch-and-cut algorithm that enhances performance using increment searching and tree pruning. The increment searching starts at the start location and then appends all possible attractions to fulfil the itineraries. Due to the characteristic of the incremental process, the need for re-calculation from the start location is eliminated. The pruning technique will stop traversing down the tree's deeper levels if any of the following conditions are met.

- The lunch period is over, but no restaurant is included in the itinerary.
- A POI with the closing time is overdue
- There is not enough time to append the compulsory POIs to the itinerary.
- A restaurant if the lunch period is fulfilled

When a valid itinerary is found, the algorithm will perform the Pareto-dominance testing for the newly found itinerary against the current solution set of itineraries to guarantee that the solution set always keeps the non-dominated itineraries. Algorithm 3.1.1 implements the above concept by eliminating the itineraries in the current solution set; if the new itinerary dominates them, the new itinerary will be appended to the solution set. . However, the algorithm will not insert the itinerary into the solution set if any itinerary in the solution set can dominate it. The eliminate formula is at line 4 which each relational operator results in one for true and zero for false. Thus, the new itinerary dominates in the solution set if r is positive and the reverse is negative, but both are non-dominated solutions if r is zero.

field	datatype	description
p	array of integers	the order of POIs
f	array of integers	the candidates POIs to be inserted
$\hat{p}$	integer	accumulation of the profits
$\hat{d}$	integer	accumulation of the distances
l	boolean	there is a restaurant in the path or not
t	integer	the current time of the last appended POI to the path.
$\hat{c}$	integer	the total of time used for all compulsory POIs.

Table 3.1 The data structure of an itinerary used in algorithms.

Input:
nondominated solution set ss and a valid itinerary it
Output: *nondominated solution set*
1 $isND \leftarrow true$
2 *for i in size (rs) down to* 1
3 $isND \leftarrow true$ *and break if* $sort\ (it.p) = sort(ss_i.p)$
4 $r \leftarrow (it.\hat{p} > ss_i.\hat{p}) + (it.\hat{d} < ss_i.\hat{d}) + (ss_i.\hat{p} > it.\hat{p}) + (ss_i.\hat{d} < it.\hat{d})$
5 $ss \leftarrow ss - ss_i$ *if* $r > 0$
6 $isND \leftarrow true$ *and break if* $r < 0$
7 *end*
8 $ss \leftarrow ss \cup it$ *if* $isND = true$
9 *return ss;*

Algorithm 3.1.1 ParetoDominance

Algorithms 3.1.2 and 3.1.3 depict the main and the core recursive algorithm of the branch-and-cut algorithm, respectively. The main algorithm prepares an empty solution set and initializes an itinerary to have only the start depot in the path. The itinerary structure is presented in Table 3.1. The core recursive algorithm, Algorithm 3.1.2, is a recursive branch-and-cut algorithm that attempts to trial all possibilities of the sequence of POIs to search for all valid itineraries. Lines 1-4 check for a valid itinerary; if the algorithm can append the end depot, the itinerary is a valid itinerary. The Pareto dominance algorithm is performed to include the itinerary to the solution set. Line 5 is the tree pruning that executes when lunchtime is overdue, but no restaurant is included, or the time left for the tour is less than the time to visit the rest compulsory POIs. Lines 6-17 append each of the candidate POIs into the current itinerary if the current time plus the time cost to the candidate POI is between the operational time of the POI. If the candidate POI can be appended, then create a new itinerary with the POI and the new candidate POIs excluding the POI and all restaurants, if the POI is a restaurant. Finally, the algorithm will call itself in line 16 to traverse down the whole tree.

```
    Input: All Parameter from table 2
    Output: nondominated solution set ss
1   ss ← []
2   it ← new itineraray structure
3   it.p ← [S_h]
4   it.f ← {1, ..., N_A} − {S_h}
5   it.p̂ ← it.d̂ ← 0
6   it.ĉ ← Σ_{i=1}^{N_M} J_{M_i}
7   it.t ← T_min
8   it.l ← false
9   ss ← BranchAndCutRecursive(it, rs)
10  return ss;
```

Algorithm 3.1.2 BranchAndCutMain

```
    Input: an uncompleted itinernary it, solution set ss
    Output: nondominated solution set ss
1   if it.p_end = E_h
2      ParetoDominance(ss, it) if it.l = true and it.ĉ = 0
3      return ss
4   end if
5   return ss if (it.t ≥ L_max and it.l = false) or T_end − it.t
                ≤ it.ĉ
6   for each f in it.f
7      continue if not(O_f − W_f ≤ it.t + V_{it.p_end,f} ≤ C_f)
8      ff ← it.f − f;
9      ff ← ff − {S_R, ..., E_R} if f ∈ {S_R, ..., E_R}
10     it2 ← new itinerary structure
11     it2.d̂ ← it.d̂ + D_{it.p_end,f}
12     it2.p ← append(it.p, f)
13     it2.f ← ff
14     it2.l ← it.l ∨ f ∈ {S_R, ..., E_R}
15     it2.p̂ ← it.p̂ + P_f
16     ss ← BranchAndCutRecursive(it2, ss)
17  end for
18  return ss
```

Algorithm 3.1.3 BranchAndCutRecursive

3.2 Greedy Algorithm

The greedy algorithm builds up the itineraries by filling them with higher profit POIs into all starter paths segments made from each restaurant's permutation and compulsory POIs. The recursive technique is employed to fill up a POI per level of calling, and the algorithm will stop traversing down if a valid itinerary is found. Thus, the algorithm is fast and can obtain high-quality itineraries; it cannot guarantee optimal tanneries.

The main algorithm is shown in Algorithm 3.2.1. It builds the permuted itineraries that contain only essential POIs: a restaurant and compulsory POIs. The process is done in lines 3-5. Consequently, for each sequence of POIs (p); the corresponding unassigned POIs (f) are assigned and sorted by high-to-low profits for being injected into p. Line 7, Evaluate function converts p into an itinerary and then feed to the core recursive algorithm.

Algorithm 3.2.2 depicts the core recursive algorithm that injects each POI from f into the given itinerary. Line 1, if the current itinerary is completed, no deeper level is traversed, the ParetoDominance function is called to attempt to insert it into the solution set. After that, the position of the restaurant (r) is determined in line 3, and attempts to find the best orders to be inserted and traversed down for the before noon period (Lines 4 and 5) and afternoon period (Lines 6 and 7). Line 8 and 9, The algorithm also ignores the current best unassigned POI for giving the chance to produce the itineraries without that POI.

```
    Input: All Parameter from table 2
    Output: nondominated solution set ss
1   ss ← []
2   for r in S_r, ..., E_r
3      for p in permutation of {S_r, ..., E_r, r, M_{1,...,N_M}}
4         p ← concat(S_H, p, E_H)
5         f ← {S_p, ..., E_p} − p
6         f ← sort f by profit, descending
7         ss ← GreedyRecursive(Evaluate(p, f), ss)
8      end for
9   end for
10  return ss
```

Algorithm 3.2.1 GreedyMain

```
    Input: an uncompleted itinernary it,
    nondominated solution set ss
    Output: nondominated solution set ss
1   return ParetoDominance (ss, it) if it.s = valid
3   r ← position of the restuarant in it.p
4   it2 ← FindBestInterpolate(it, 1, r)
5   ss ← GreedyRecursive (it2, ss) if it2.s ≠ invalid
6   it2 ← FindBestInterpolate(it, r, |it.p|)
7   ss ← GreedyRecursive (it2, ss) if it2.s ≠ invalid
8   it.f ← it.f − f_1
9   return GreedyRecursive (it, ss)
```

Algorithm 3.2.2 GreedyRecursive

Algorithm 3.2.3 computes the variables from the structure of an itinerary (Table 3.1) from a given POI list. The algorithm also determines the POI list that is valid, semi-valid, or invalid. A valid itinerary is a usable itinerary, whereas the invalid itinerary violates the constraints: reach a POI after its close time (line 6), more than one restaurant (line 10). Moreover, the number of compulsory POIs is not fulfilled (line 15), but no semi-valid POI is found, also considered an invalid itinerary. The semi-valid is when the constraint of the open time of a POI is not satisfied; the POI is reached too early. For this case, it is possible to insert another POI before this one (line 7). The main loop in lines 5-14 considers each POI in the POI list and gathers nc the number of compulsory POIs, nr the number of restaurants, and the variables in the itinerary structure.

Input: a *POI list p*
Output: *an itinerary it*

1 $s, t \leftarrow valid, T_{min}$
2 $nr \leftarrow nm \leftarrow \hat{p} \leftarrow \hat{c} \leftarrow 0$
3 $it \leftarrow$ *new itineraray structure*
4 $it.s, it.\hat{d} \leftarrow invalid, \infty$
5 *for i in* $2, \ldots, |p|$
6 *return it if* $t + V_{i-1,i} > C_i$
7 $s \leftarrow$ *semivalid if* $t + V_{i-1,i} < O_i - W_i$
8 $t \leftarrow (O_i \text{ if } t + V_{i-1,i} < O_i \text{ else } t + V_{i-1,i}) + j_i$
9 $nr \leftarrow nr + 1$ *if* p_i *is in* $S_r, \ldots, E_r$
10 *return it if* $nr > 1 or (nr \text{ is zero and } t > L_{max})$
11 $nc \leftarrow nc + 1$ *if* p_i *in* M
12 $\hat{p} \leftarrow \hat{p} + P_{p_{i-1},p_i}$
13 $\hat{d} \leftarrow \hat{d} + D_{p_{i-1},p_i}$
14 *endfor*
15 *return it if* $nc < |M|$
16 $it.s, it.p, it.\hat{p}, it.\hat{d} \leftarrow s, p, \hat{p}, \hat{d}$
17 *return it*

Algorithm 3.2.3 Evaluate

Algorithm 3.2.4 converts a sequence of POIs into an itinerary. Loop in lines 5-14 will try to insert the current highest profit (f1) into each segment and the valid itinerary with the shortest total distance will be returned to the caller.

Input: *an itinerary it, start search position* S_F, *and end search position* E_F
Output: *an itinerary it*

1 *best* $\leftarrow$ *new itiner aray structure*
2 $best.\hat{d} \leftarrow \infty$
3 $best.s \leftarrow invalid$
4 *for i in* $S_F, \ldots, E_F$
5 $p \leftarrow$ *insert* $it.f_1$ *to* $it.p$ *at* i
6 $it2 \leftarrow Evaluate(p)$
7 $best \leftarrow it2$ *if* $best.\hat{d} < it2.\hat{d}$
8 *end for*
9 *return best*

Algorithm 3.2.4 FindBestInterpolate

Algorithm 3.2.5 is used for the local search. It attempts to replace a POI in each itinerary of the solution set with a POI in the unassigned POIs that minimizes the total distance. Lines 1-3 are for looping all POIs in each itinerary in the solution set. The POIs in the loop must not be the compulsory POIs nor the restaurant, the for loop is implemented with this check in line 4. Then, the algorithm will check if any POIs having equal profit scores with the unassigned POIs, if do, a POI that makes the lowest total distance will be replaced with the original one. Finally, the algorithm will return the local search solution set in line 15.

Input: *solution set ss*
Output: *solution set*

1 *for it in ss*
2 $best \leftarrow copy(it)$
3 *for i in* $2, \ldots, |it.p| - 1$
4 *continue if* $it.p_i \in \{S_r, \ldots, E_r, M_{1,\ldots,|M|}\}$
5 *for j in* $S_p, \ldots, E_p$
6 *continue if* $j \in it.porP_j \neq P_{it.p_i}$
7 $p \leftarrow copy(it.p)$
8 $p_i \leftarrow j$
9 $it2 \leftarrow Evaluate(p)$
10 $best \leftarrow it2$ *if* $best.\hat{d} < it2.\hat{d}$
11 *end for*
12 *end for*
13 *replace it with best in ss*
14 *end for*
15 *return ss*

Algorithm 3.2.5 Local Search

3.3 Solution Ranking

A Pareto front consist of multiple itineraries so the trip planners have more choices to choose from. Based on their preferences they can choose the most suitable itinerary. For example, if they want to enjoy the trip as much as possible, they may select the itinerary with the highest total profit or choose the lowest total distance if they do not want to spend much time walking. Unfortunately, choosing the most suitable itinerary might not be a very important task when they do not provide a preference.

I embrace the concept of knee solutions to solve this problem, the widely accepted technique when no preference is provided. Moreover, knee-based solution ranking is employed to highlight outstanding itineraries and offer trip planners more choices. For this reason, I rank the itineraries from the Pareto fronts that generated from the previous sub-sections by using an angle-dominance knee-based solution ranking algorithm called RADA. However, I have post-processed the result from RADA not to index the extreme solutions as they are always good choices for the trip planners to be considered.

variable	meaning	value
T_{min}, T_{max}	tour time	8:00-17:00
L_{min}, L_{max}	lunch period	11:30-13:00
$W_{S_p,\ldots,E_p}$	wait time allowed before a POI open.	15 mins
$W_{S_r,\ldots,E_r}$	wait time allowed before a restaurant open	15 mins
W_{E_h}	Amount of time allowed to finish the tour before end tour time	30 mins

Table 3.3 Parameters setting.

Compulsory POIs	POIs	GD	IGD	Time used hh:mm:ss	
				greedy	branch-and-cut
None	13	0.0411	0.0695	00:00:01	00:00:15
	15	0.0264	0.0551	00:00:04	00:04:01
	20	0.0715	0.0721	00:00:51	21:12:51
	30	-	-	00:01:11	time out
	40	-	-	00:00:37	time out
	50	-	-	00:00:28	time out
{C1}	13	0.0263	0.0411	00:00:01	00:00:13
	15	0.0059	0.0059	00:00:01	00:03:33
	20	0.1494	0.0964	00:00:11	21:24:40
	30	-	-	00:00:20	time out
	40	-	-	00:00:11	time out
	50	-	-	00:00:07	time out
{C1, C2}	13	0.0263	0.0411	00:00:00	00:00:12
	15	0.0059	0.0059	00:00:00	00:03:36
	20	0.0480	0.0521	00:00:01	23:03:26
	30	-	-	00:00:03	time out
	40	-	-	00:00:01	time out
	50	-	-	00:00:01	time out
{C1, C2, C3}	13	0.0526	0.0432	00:00:00	00:00:12
	15	0.0000	0.0000	00:00:00	00:03:36
	20	0.0530	0.0654	00:00:01	15:43:18
	30	-	-	00:00:02	time out
	40	-	-	00:00:01	time out
	50	-	-	00:00:01	time out

Table 3.2 Time used for greedy and branch-and-cut algorithms against the dataset.

4. EXPERIMENTS AND RESULTS

4.1 Parameter Setting

In the experiments, I produce 5 subsets from Alappuzh district in the the proposed region, as mentioned in Section 2.2.2, each subset having a combination of POIs. The route having 3 compulsory POIs, they are mentioned below..

- Alappuzha Beach {A}
- Krishnapuram Palace {B}
- Pathiramanal {C}

The compulsory POIs are found by the reference of more than one reliable tourist websites of the region. There are other major attractions in the area including,

- Kuttanadu
- Karumadi
- Ambalappuzha Temple
- Arthunkal Church
- Mannarsala temple

I'm able to get as many needed combinations from the region. Figure 4.1 shows the pareto fronts of the region. Which include the best itineraries can be found for a single day trip with the compulsory POIs.

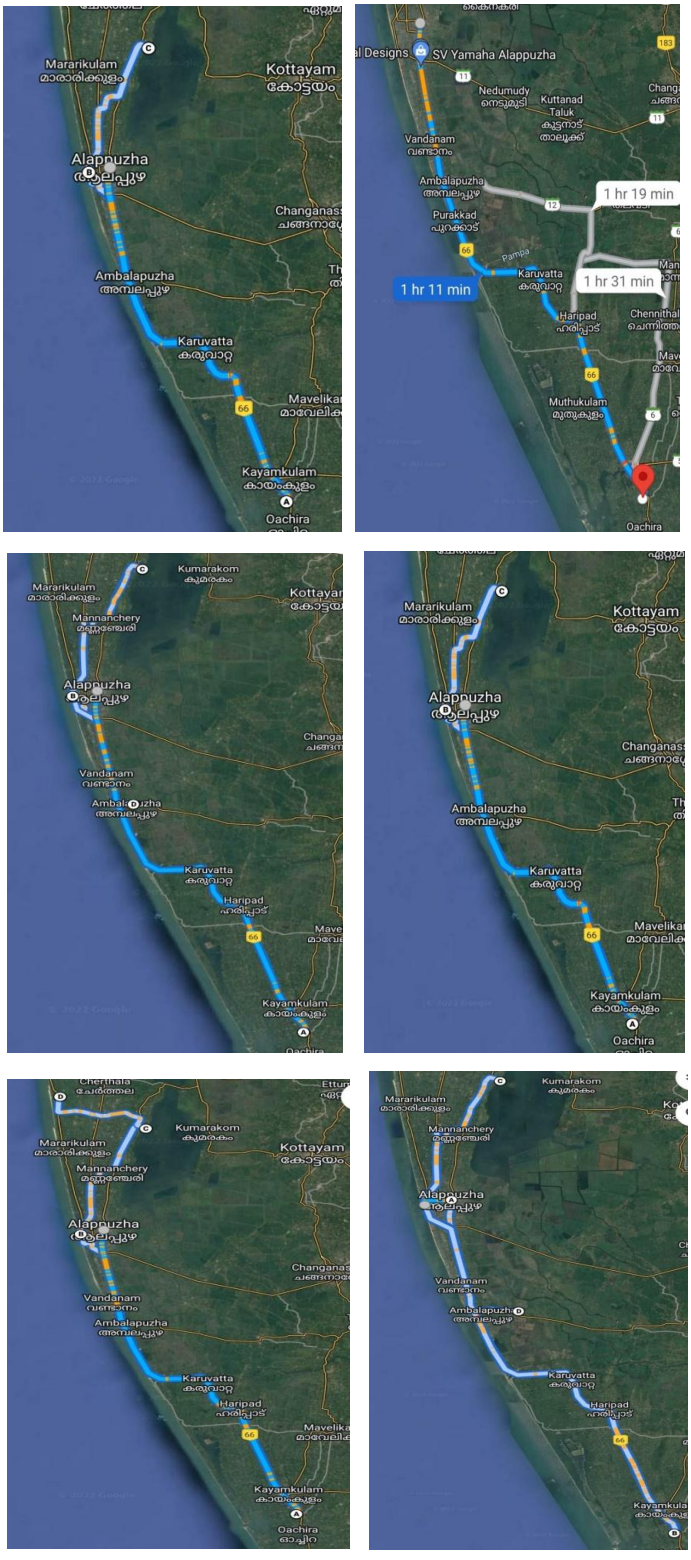

Figure 5.1 Pareto fronts of the selected region.

4.2 Evaluation Metrics

This research chooses the top three indicators to evaluate the greedy algorithm against the benchmarked branch-and-cut algorithm from the standard indicators for multi-objective optimization. These indicators are probably the most accepted indicators for Pareto-based multiobjective optimization. First, I pre-processed the results by negating the total profit to minimize both objectives. Then the two objectives were normalized to the values between zero and one. The G and E denote the set of greedy and branchand-cut solution sets, respectively.

4.2.1 Generational Distance (Gd.)

Generational distance is defined as the average of the distance of each itinerary in the greedy solution set to its closest itinerary in the branch-and-cut solution set. Zero is the ideal solution that implies that two Pareto fronts are precisely the same. The definition of GD is shown as Eq. V-B3.

$$GD(G,E) = \sum_{g \in G} MIN(\|g - e\|)/\lceil G \rceil; \quad \forall e \in E \quad (14)$$

5.2.2 Inverted Generational Distance (IGD)

The inverted generational distance (IGD) is the inverse version of GD, defined as the average of the total distance of each itinerary in the branch-and-cut solution set to its closest itinerary in the Greedy solution set. The concept of the values is the same as GD; Zero is the ideal value. The definition of IGD is shown as Eq. (15).

$$GD(G,E) = \sum_{e \in E} MIN(\|e - g\|)/\lceil E \rceil; \quad \forall g \in G \quad (15)$$

4.3 Itinerary Selection

The above set of POIs are generated on the basis of research done on almost all the available tourist websites of the region. The POIs listed above have the largest profit scores. All of the itineraries generated on the basis of compulsory POIs. The web application is able to list the restaurants of the region also, so each itinerary will have its on corresponding set of restaurants. A tourist have so many priorities to choose an itinerary, food is one of the highest priority.

In this particular experiment , I considered Ambalappuzha will have the lunch break and listed 3 famous restaurants in the POI.

4.3.1 Itinerary Analysis

We can analyze itineraries from the pareto front, following is the itinerary consists of only compulsory POIs.

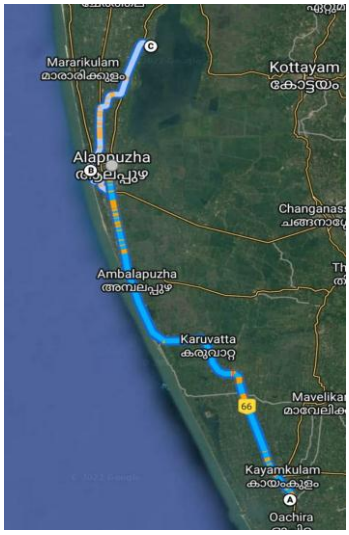

Fig 4.2 Itinerary only having compulsory POIs

Kuttanadu is one of the famous attractions of the area , I tried to add the POI to our recommended route, and below is the corresponding itinerary.

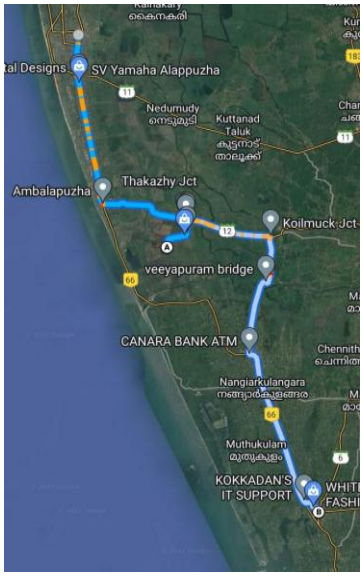

Fig 4.3 Unreliable itinerary

So It is clear from the itinerary that Kuttanadu will give higher profit to the trip, but it is not fit in the itinerary with the other 3 compulsory POIs and is will make the trip to miss other POIs.

The itinerary only having compulsory POIs is enough to plan a oneday trip, but we are trying to include all the possible POIs that will satisfy all the requirements. So from the above combination of POIs, below is the one itinerary that include the compulsory POIs and by adding one POI (D) along the path is not affecting the total distance but its giving more profit in the travel.

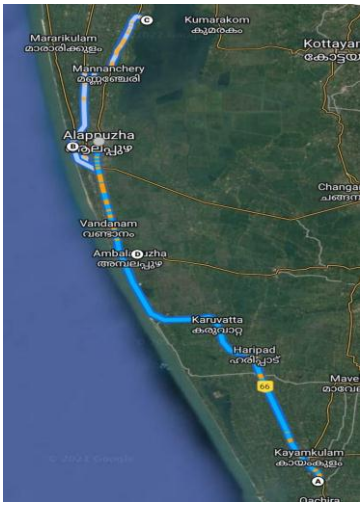

Fig 4.4 Most suitable itinerary

Once the POIs are fixed by the user, the website will give a suitable description about the itinerary area. Fig 4.5 is the description user get for the current itinerary.

Alappuzha is one of the destinations in Kerala, which any traveller will never miss, while his trip to this 'God's Own Country'. Alappuzha is unique with its ever beautiful backwaters, greenish paddy fields, coconut groves etc. The serene ambience and tranquility that you find here will never leave you even after you reach home. There are beautiful places in Alappuzha that will always remain in your memory. Lets start the journey...

Fig 4.5 Itinerary description

The application also included the public transportation details of the area in a description box as in Fig 4.6.

Bus Details	Time And Duration	Price
KSRTC (Kerala) A/C Seater (2...	07:35 PM 1h 10m 08:45 PM	₹ 139 onwards Select seats
KSRTC (Kerala) SUPER FAST (...	07:26 PM 1h 15m 08:41 PM	₹ 47 onwards Select seats
KSRTC (Kerala) SUPER FAST (...	04:40 PM 1h 15m 05:55 PM	₹ 47 onwards Select seats
KSRTC (Kerala) NON A/C Seat...	11:30 PM 0h 50m 12:20 AM	₹ 74 onwards Select seats

Fig 4.6 Public transportation details

3. CONCLUSIONS

Introduced the new real-world tourist trip design problem called the multi-objective orienteering problem with restaurant selection and compulsory POIs (MOPTW-RSCP). I proposed two exact multi-objective algorithms for solving the problem and developed the web application in reactJS by inspiring from the algorithms.

By theoretical basis, the branch-and-cut algorithm can search for all optimized solutions. But, the time complexity of the algorithm is exponentially high. The time complexity of the branch-and-cut algorithm measured in a time limit of 48 hours and found it can easily search for the optimal Pareto fronts for 13 and 15 POIs, but for 20 POIs, it took 15-23 hours runtime. Hence, the branch-and-cut algorithm failed to find the solutions when the number of POIs is more than 20. Conversely, the greedy algorithm could efficiently find solutions within the time limit.

I used reactJS and css to develop the UI, The ui having option to enter source and destination then it will populate the well known attractions in the area. After selection of the POIs user have the option for restaurant selection near by the POIs. The web application include the description of the area and as part of the CO2 emission control, I have included the available public transportation details, So the traveler can rely on the public transportation facilities instead of the private vehicle whenever possible.

REFERENCES

[1] G. Li andW.-Y. Chung, ``Combined EEG-gyroscope-tDCS brain machine interface system for early management of driver drowsiness,'' IEEE Trans. Human-Mach. Syst., vol. 48, no. 1, pp. 50_62, Feb. 2014

[2] Tourism United Behind the Glasgow Declaration on Climate Action at COP26, UNWTO, Glasgow, U.K., 2021.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[3] S. Sohrabi, K. Ziarati, and M. Keshtkaran, ``A greedy randomized adaptive search procedure for the orienteering problem with hotel selection,'' Eur. J. Oper. Res., vol. 283, no. 2, pp. 426440, Jun. 2020.

[4] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. Van Oudheusden, ``The city trip planner: An expert system for tourists,'' Expert Syst. Appl., vol. 38, no. 6, pp. 65406546, 2011.