applied sciences

Article

A Novel Application Based on a Heuristic Approach for Planning Itineraries of One-Day Tourist

Agostino Marcello Mangini [†], **Michele Roccotelli** [*,†] and **Alessandro Rinaldi** [†]

Department of Electric and Information Engineering, Faculty of Engineering, Politecnico di Bari, Via Orabona 4, 70126 Bari, Italy; agostinomarcello.mangini@poliba.it (A.M.M.); alessandro.rinaldi@poliba.it (A.R.)
* Correspondence: michele.roccotelli@poliba.it
† These authors contributed equally to this work.

Abstract: Technological innovations have revolutionized the lifestyle of the society and led to the development of advanced and intelligent cities. Smart city has recently become synonymous of a city characterized by an intelligent and extensive use of Information and Communications Technologies (ICTs) in order to allow efficient use of information. In this context, this paper proposes a new approach to optimize the planning of itineraries for one-day tourist. More in detail, an optimization approach based on Graph theory and multi-algorithms is provided to determine the optimal tourist itinerary. The aim is to minimize the travel times taking into account the tourist preferences. An Integer Linear Programming (ILP) problem is introduced to find the optimal outward and return paths of the touristic itinerary and a multi-algorithms strategy is used to maximize the number of attractions (PoIs) to be visited in the paths. Finally, a case study focusing on cruise tourist in the city of Bari, demonstrates the efficiency of the approach and the user interaction in the determination of the itinerary.

Keywords: itinerary planning; smart tourism; graph theory; heuristic approach

check for updates

Citation: Mangini, A.M.; Roccotelli, M.; Rinaldi, A. A Novel Application Based on a Heuristic Approach for Planning Itineraries of One-Day Tourist. *Appl. Sci.* **2021**, *11*, 8989. https://doi.org/10.3390/app11198989

Academic Editor: Paola Pellegrini

Received: 9 August 2021
Accepted: 17 September 2021
Published: 27 September 2021

1. Introduction

The planning of touristic itineraries is a typical decision making process for tourists visiting a city in a limited time period. The selection of the most valuable Points of Interests (PoIs) is not simple.

In the last years mobile applications are offering a variety of services from vacation planning to mobile tourist guides and tourism recommender systems [1,2]. The design of flexible, efficient, and user-friendly applications for mobile devices has a great interest from both a commercial and a research point of view. The authors in [3] propose a mobile application based on a hybrid multiobjective genetic algorithm to smartly generate feasible itineraries. The algorithm incorporates an advanced heuristic to build a route, with a start and an arrival time passing from a set of locations each characterized by a score measuring its attractiveness, an opening and a closing time, and visit duration. Vansteenwegen et al. [4,5] present an advanced mobile tourist guide, capable of suggesting a near-optimal and feasible selection of attractions and a route passing among them. The related optimization problem is solved by using a combination of guided local search metaheuristics. Booth et al. [6] develop a data model for trip planning in multimodal transportation systems and Navabpour [7] plans a trip with multimodal transportation based on Service Oriented Architecture (SOA). In addition, Andre et al. [8] design a journey planning system based on safety, weather and specific travel time for individual user. Gonzalez et al. [9] propose a fastest-path computation system on a road network using a traffic mining approach. However, while the above papers mainly focus on the mobile application architecture, the following two subsections analyze existing contributions focusing on methodology and parameters.

1.1. Related Works: Methodology-Based Classification

This section groups and analyses research works that provide rigorous description of heuristic and metaheuristic approaches to solve the Tourist Trip Design Problem (TTDP). These approaches result the only viable methods to efficiently optimize the travel itinerary, by analyzing the problem from different perspectives, with different problem variables and constraints. The objective in TTDP modeling is to identify a set of near-optimal itineraries to maximize tourist satisfaction. The baseline combinatorial optimization problem for TTDP is the orienteering problem (OP). The OP can be used to model the TTDP where the PoIs are associated with a profit and the goal is to find a single tour that maximizes the profit collected within a given time budget. In the OP, given a starting node s, a terminal node t and a positive time limit (budget), the goal is to find a path from s to t such that the total profit of the visited nodes is maximized. In the related literature, Garcia et al. [10] propose an intelligent routing system that defines an optimization problem including multiple paths to move from one location to another. Such a system, by exploiting an iterated local search metaheuristic method, suggests a personalized tour combining information about the local attractions, weather forecasting and public transportation. Gavalas analyzes the models, algorithmic approaches and methodologies about tourist trip design problems [11]. Recent approaches are reported aiming at taking into account a multitude of realistic PoIs attributes and user constraints. In this context, Gunawan et al. focus on the most recent works about the Orienteering Problem (OP) and its variants [12]. The authors focus on a comprehensive and thorough survey of recent variants of the OP, including the proposed solution approaches. The work reports the new variants of the OP, such as the Stochastic OP, the Generalized OP, the Arc OP, the Multi-agent OP, the Clustered OP and others. The authors summarize several interesting applications which are related to the mobile crowdsourcing problem, the Tourist Trip Design Problem, the theme park navigation problem and others. The authors in [13] provide a detailed explanation about operation on tour routes only qualitatively. An optimizer is proposed in [14], where a multiobjective evolutionary algorithm is used to identify the near-optimal solutions to the planning of multiple-day routes in a reasonable computational time. In the contribution [15], the authors present a heuristic procedure for the generalization of a optimization problem to plan personalized recommendations for daily sightseeing itineraries for mobile tourist guides.

Extensions of the OP have been applied to model more complex versions of TTDP: the OP with time windows (OPTW) considers visits to locations within a predefined time window; this allows modeling opening days/hours of PoIs. The time-dependent OP (TDOP) considers time dependency in the estimation of time required to move from one location to another; therefore, it is suitable for modeling multimodal transports among PoIs. In particular, Cotfas considers a more complex variant of the tourist trip design problem i.e., the time-dependent in the estimation of the time required to move from one location to another for planning daily tours according to tourist's preferences [16].

The team orienteering problem (TOP) is the extension of the OP to multiple tours. The TOP with time windows (TOPTW) has been mostly commonly studied among the aforementioned OP variants since it is useful for modeling several real-life optimization problems. Vansteenwegen et al. propose a metaheuristic algorithm to tackle a more effective extension of the optimization problem [17]. The proposed algorithm performs a planning of a multipleday tour by considering a set of PoIs, a visiting duration, and a set of multiple opening and closing times per day combined with the trip constraints of the tourist.

Other studies propose approaches based on Graph Theory [18–21], applied in tourism. The authors in [19] deal with typical tourist attractions in urban destinations, as pedestrian zones, market areas or urban areas of architectural, cultural and scenic value rather than only visiting sites of restricted access or taking the fastest route to move among city landmarks. Herein, the authors introduce Scenic Athens, a context- aware mobile city guide for Athens (Greece) which provides personalized tour planning services to tourists.

Scenic Athens derives near-optimal sequencing of PoIs along recommended tours, taking into account a multitude of travel restrictions and PoI properties, so as to best utilize time available for sightseeing. The authors in [20] define tourist routes by means of graph theory. The authors also calculate some relative indexes (e.g., the circle number, circle ratio, line-point ratio etc.) to make quantitative evaluation of tourist routes. Chen et al. apply the Graph theory to optimize tour path and tour flows to provide practical solutions to tourist guides [21].

1.2. Related Works: Parameter-Based Classification

Another classification that can be proposed is based on the works that emphasizes the study of the effect of key parameters of the TTDP on the final solution, such as: (i) the selection of transport modes to reach the different PoIs; (ii) the choice of PoIs; (iii) the number of tours to be generated, on the basis of visiting duration; (iv) the visit duration of a PoI; (v) the travel times among PoIs; (vi) the daily time budget that a tourist wishes to spend on visiting a PoI; (vii) the weather conditions.

Transport is a critical and dynamic process of tourism, which facilitates physical movement to points of interest [22–24]. Transportation affects the accessibility to the tourist destination, the distance travelled, and the comfort of the trip [25,26]. The authors in [27] develop a genetic algorithm (GA) to solve the TTDP that included multimodal transport and real traffic parameters and time constraints.

In [14] the influencing factors of the tour route choices of tourists are analyzed by means of a questionnaire survey. Moreover, tour routes multiobjective optimization functions are prompted for the tour route design with the aim of maximise the user satisfaction with the minimum tour distance. The authors in [17] analyze the planning of a multipleday tour by considering a set of PoIs, a visiting duration, and a set of multiple opening and closing times per day combined with the trip constraints of the tourist. The authors in [28] apply an evolutionary algorithm to solve the tour planning problem in time-dependent urban areas. Gavalas et al. [29] develop a tool for tourist itineraries that considered the departure time and the mode of transport on the tourist route. Wu et al. [30] develop a mathematical model to consider the selection of transport modes, the travel budget, and the maximum travel times. Zheng et al. [31] design a multi-objective model of one-day urban tourist routes, taking into account the transport modes and the complexity of urban tourism transport systems, as congestion, and the transport needs of tourists. Zhang et al. [32] develop a model for the construction of itineraries in scenic routes considering the modes of transport. The authors in [33] analyze the environmental implications of tourist itineraries by creating groups of tourists that use a single mode of transportation (i.e., taxis). Some works study the use of electric vehicles (EV) for the generation of more environmentally friendly tourist itineraries, such as [34–36].

Other works focus on planning trips for tourist group [33,37], that consider the individual preferences of each tourist. The authors in [38] develop a model for the route design problem for various cycle-tourists. The model consider the preferences of each tourist who incorporates different benefits on the same route. Finally, the authors in [33] develop a route planning model that considers multiple days, urban tourism, PoI categories, and heterogeneous preferences for a group of tourists that maximise profit and minimises travel time, distance, and cost.

1.3. Contribution of the Paper

From the analysis of the above reported studies, the OP is not suitable in case the PoIs need to be selected and exchanged among different itineraries, like outward and return paths of a one day trip, because of time constraint. In this case, it is necessary to implement a multi-level algorithm to be able to consult the tourist on any relocation of PoIs in the tour. To these aims, we applied the Travel Salesman Problem (TSP) [39–41] method that involves finding the shortest route through n nodes that begins and ends at the same city and visits every node. The TSP is among the best-known combinatorial optimization problems and has been intensely studied by researchers in various research fields.

In this paper, we aim to determine the optimal itinerary for the one-day tourist, maximizing the number of PoIs to be visited in the outward and return parth, and at the same time minimizing the travel times taking into account the tourist preferences and hard time constraints. The idea is to allow the tourist to select the preferred PoIs to be visited on the first part of the day, i.e., in the outward trip, and on the second part of the day, i.e., in the return trip, respectively. We formulate our optimization problem on the basis of Graph theory, TSP and multi-level algorithms. We model the city PoIs network on the basis of the graph theory, where the nodes represent the various attractions (PoI) of the city and two separate graphs are derived. The tourist can select the PoIs of the starting graph to be visited with high priority in the outward and return journeys, respectively. In our application the tourist is part of the multi-algorithms approach interacting with it and taking decisions for one-day tourist. The proposed approach plans the tourist itinerary, minizing travel times based on the TSP algorithm, taking into account the priority list of PoIs and the decisions of the user. The TSP is used in this paper since it allows to consider a first itinerary solution including all the PoIs of the city that is refined by the multi-level algorithms interacting with the tourists. In detail, compared with the analyzed works, this paper presents the following novelties:

- an innovative multi-level algorithm approach is proposed to determine the optimal roundtrip path: the outward and return journeys are specified and customized, minimizing the total travel time, including the visiting time of each PoI.
- the number of attractions to be visited is maximized and is splitted between the outward and return path in order to improve the visiting experience on the basis of user preferences.
- the tourist is seen as an active and informed user who directly interacts with the system for the optimal planning of both the outward and return journeys, not only providing initial inputs and preferences but taking decisions at intermediate stages of the approach.

The rest of the paper is organized as follows: Section 2 describes the one-day tourist itinerary planning problem; Section 3 presents the Multi-level algorithm approach for the itinerary planning while Section 4 provides the analysis of the algorithms performance and complexity; Section 5 demonstrates the effectiveness of the proposed approach by a case study focusing on the cruise tourist in the city of Bari and Section 6 provides the conclusions and future works perspectives.

2. The One-Day Tourist Itinerary Planning Problem

The one-day tourist, having reached a stage of his journey through the airplane, train, car or cruise ship etc., wonders how to spend at best his/her time in the city in a short time period (e.g., one day).

Due to the limited time, it is therefore necessary to pay attention to the organization of the visits and excursions. The tourist can opt for a tour pre-organized by the operator or he/she can plan it on his/her own. In the first case, one of the advantages concerns the mere observation of the predefined roadmap to visit the city, without any worries. This case, on the other hand, does not always satisfy the personal interests of the individual tourist who must follow the visiting group and, in addition, can not personally manage the route and the stops. In the second case, however, the tourist has more freedom of choice but he/she must plan independently the trip in a city and respect the departure times that are mandatory. Instead of relying on the tours organized by the company, sometimes with unsatisfactory results, the tourist by use only a smartphone can select the preferred attractions.

Today, there are numerous online travel planning systems that allow to automatically generate a selection and routing plan to visit PoIs that suit the tourist's personal interests [42]. These systems implement various functionalities that aim to satisfy different profiles of tourist interest [43,44]. Therefore, considering a tourist discovering the city, in addition to walking through its most famous streets, he/she wants to head, for example, to a restaurant near an attraction to have lunch and taste the typical dishes of the place, and then resume the tour and return back. For instance, by simply accessing an app from the smartphone, he/she can set the time available to carry out the tour from a starting point to a restaurant and the time to return back. The visiting times must also include the stop times for activities such as take photos in front of a monument, go shopping, visit a museum and so on. The tourist can also select the preferred PoIs to be visited on the first part of the day, i.e., in the outward trip, and on the second part of the day, i.e., in the return trip, respectively. In addition, the tourist can also indicate the PoIs that are less important and that can be deleted by the roundtrip in case of time unavailability.

Then, let us describe an example in order to present the addressed problem. Firstly, the following assumptions are made:

- the tourist is an active user who wants to interact with the application in order to customize the daily roundtrip;
- the PoIs of the city are initialized by the application;
- the tourist indicates the starting and destination PoIs, the travel modes and time preferences as well as the PoIs to be visited with high and secondary priority, in the first and second part of the day, respectively.

Let us consider the case of a cruise tourist who wants to visit the city in one day, without loss of generality. When arriving at the port, the tourist needs to have a plan for the daily tour. In particular, he/she needs to decide which PoIs to visit based on the available time and in which order, also making a priority list to be sure to visit the most important ones. There can be also the necessity to specify which PoIs to be visited in the first part of the day, that is usually lightful and more appropriate to visit outdoor spaces like parks, before to have a lunch, usually in a typical restaurant. The tour for the second part of the day, starts after lunch allowing to complete the city visit going towards the final destination point, i.e., the port. Of course, an application is needed to help the tourist at planning the less time consuming roundtrip, respecting the preferences. In our scenario, the application initializes the PoIs network and shows to the user the map of the city PoIs with related information, including traveling times among each PoI couple based on transport means. Different trip solutions can be provided by the application based on the user choice regarding the stop time at each PoI and preferred way of transport: (1) fastest, (2) by foot, (3) by metro/bus. The tourist is also asked to indicate the starting and destination points of the roundtrip, that are different from the origin/final point (i.e., the port), as well as the time deadline for the roundtrip. In addition, the tourist is asked to decide which PoIs to be visited in the first and second part of the day, indicating the priority and the desired time to dedicate to the visit. On this basis, the application try to generate the customized outward and return tours of the day by applying the heuristic procedure presented in Section 3. If the deadline time both for the outward and return trips are respected, the heuristic procedure investigates the addition of secondary importance PoIs and generates the final roundtrip itineraries. On the contrary, if the deadline time of the outward and/or the return tours is violated, the heuristic procedure can exchange PoIs between the first and the second tour. In case some feasible solutions are determined, i.e., the deadlines of the outward and return trips are satisfied, the procedure delegates to the user the choice of a solution from a list created by the application. Afterwards, the heuristic procedure determines the final customized outward and return tours of the day including possible addition of secondary PoIs. Finally, if the deadline time of the outward and/or the return tours is violated and no feasible solution is achievable, a PoI deletion procedure must be implemented in order to respect the deadline travel time. Let us summarize the necessary input and output information and data of the proposed itinerary planning application:

Initial inputs from the user:

- starting and ending PoI;
- deadline time both for the outward and return trips;
- stop times;
- priority list of PoI;
- list of PoIs both for the outward and return trips;
- preferred mode of travel;

Moreover, other inputs are required to the user from the application while running in order to refine the roundtrip customization as described in detail in Section 3.

Real time inputs from the user:

- preferred itinerary from a list of feasible solutions determined by the proposed automatic procedure.

Outputs by the application:

- outward and return paths;
- outward and return travel times.

3. The Multi-Level Algorithm Approach for Tourist Itinerary Planning

In this Section, we want to present an adequate solution to the problem of the one-day tourist, whose goal is to visit the greatest number of attractions and carry out activities of his own liking, respecting the times available for visiting. First of all the city PoI network needs to be modeled in order to connect all the PoIs and decide the best itinerary. We apply the Graph theory [21,45] to model and study the PoIs network which in this paper is modeled as a weighted connected graph [46]. From each graph a path is determined ensuring that the tourist will visit only once those nodes representing the essential PoI: (i) the first path, called outward path, is from the source to the destination; (ii) the second path, called return path, is from the destination to the source.

In particular, the nodes of the graph represent the city attractions. In addition, the weight of an arc connecting two nodes represents the travel time between two attractions. More in detail, the proposed approach uses two graphs G_o and G_r that are built considering the following tourist inputs: the starting and ending PoIs of the outward path (they correspond to the ending and starting PoIs of the return path), the other preferred PoIs to visit during the outward and return path, the preferred transport mode. The starting PoI (ending PoI) of the outward path is represented by the source node (destination node) v_s (v_d) as shown in Figure 1. Moreover, the starting PoI is the place that the tourist firstly reaches after leaving the airport, port or station that are respresented in Figure 1 with the origin node V_{origin}. The origin node is not included in the set of nodes of G_o and G_r. Finally, each arc of graphs is weighted by the travel time between two PoIs and the time depends by the preferred transport mode chosen by the tourist.

The proposed approach to solve the tourist problem is based on a multi-level algorithm approach [39]. The proposed Algorithms are modeled by means of UML diagrams. UML is a standard highly recognized language widely used to visually describe software programs and algorithms [47]. More specifically, there is the main algorithm, so called Algorithm 1, that is responsible for the data initialization and for the determination of the initial paths. Moreover, Algorithm 1 makes use of two sub-algorithms to find an optimal planning of the itinerary based on the tourist needs in term of time and places of interest, by applying the TSP algorithm. The TSP is about a traveling man who wants to visit only once each PoI of the list returning to the initial PoI through the least cost route. The TSP is suitable to be modeled through a graph in which the nodes are the PoI and each arc connects a couple of PoI (i, j) including a travel cost from i to j. The total lenght of a journey is given by the sum of the arc weights included in the round-trip of the traveler. In order to formulate the generic version of the asymmetric TSP, the following binary variables are needed:

$$x_{ij} = \begin{cases} 1 & \text{if arc (i,j) is in the tour} \qquad i,j \in \{1,\dots,m\} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

with m total number of PoIs. Now, according to the Dantzig–Fulkerson–Johnson formulation the TSP can be formalized as the following integer linear programming problem:

$$min \sum_{i=1}^{m} \sum_{j=1,j\neq i}^{m} c_{ij}x_{ij}$$

$$s.t.$$

$$\begin{cases} \sum_{i=1,i\neq j}^{m} x_{ij} = 1 \qquad j = 1,\dots,m & \text{(2a)} \\ \sum_{j=1,j\neq i}^{m} x_{ij} = 1 \qquad i = 1,\dots,m & \text{(2b)} \\ \sum_{i\in K} \sum_{j\in K,j\neq i} x_{ij} \leq |K| - 1 \qquad \forall K \subset \{1,\dots,m\}, |K| \geq 2 & \text{(2c)} \end{cases}$$

with $c_{ij} > 0 \; \forall i,j \in \{1,\dots,m\}, i \neq j$ time cost to travel from i to j, K nonempty subset of the set of m PoIs and $m(m-1)$ number of binary variables. In particular, constraints (2c) ensures that no subset K can generate sub-tours, i.e., only a single tour will be generated. In order to obtain the symmetric version of the TSP it is necessary to have $c_{ij} = c_{ji} \; \forall i,j \in \{1,\dots,m\}, i \neq j$. It holds that the number of variables in the symmetric TSP is halves with respect to the asymmetric TSP. In this paper, we consider the symmetric TSP inside the proposed heuristic approach modeled with an undirect graph to find the optimal travel times and paths associated to the outward and return tours.

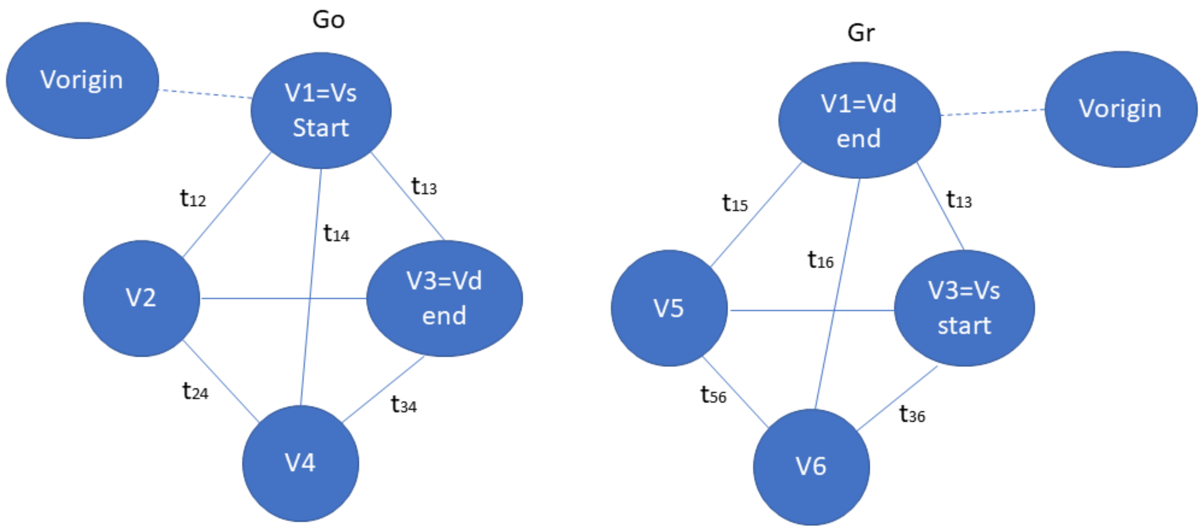


Figure 1. G_o and G_r graph examples.

3.1. *The Proposed Heuristic Approach*

The proposed approach starts by Algorithm 1 described by the UML diagram in Figure 2 that is the upper level Algorithm that executes two phases: (1) initialization phase; (2) itinerary planning phase.

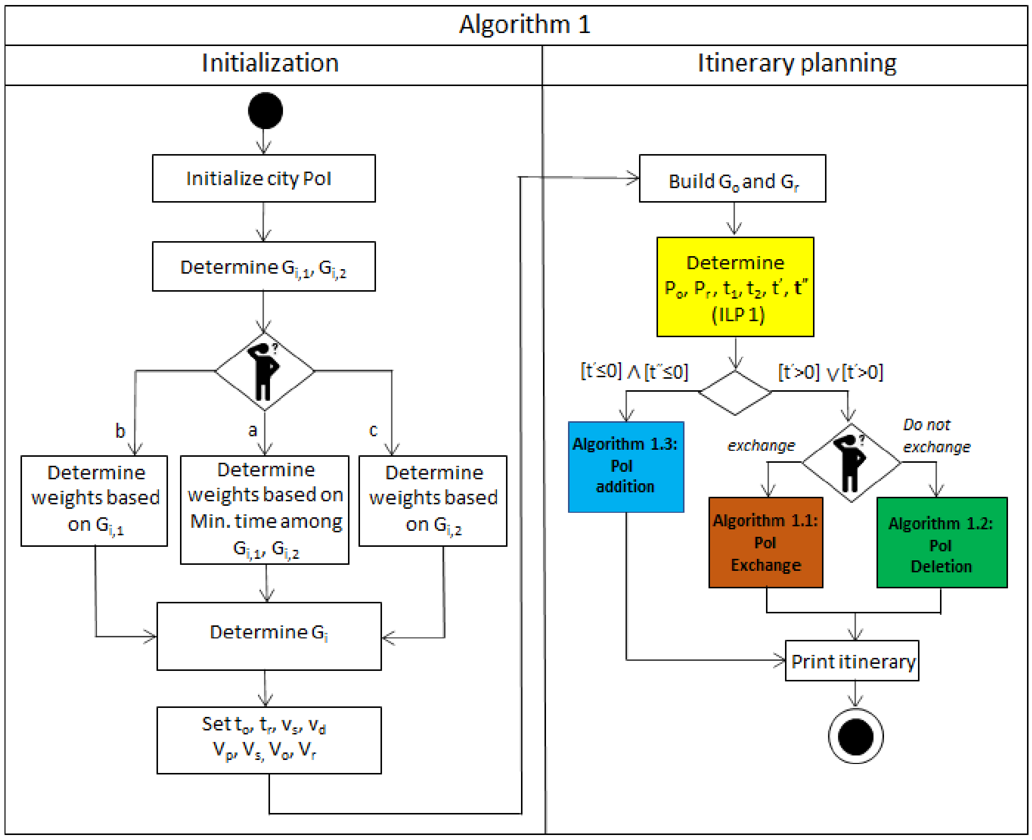

Algorithm 1
Initialization
Itinerary planning
Initialize city PoI
Determine $G_{i,1}$, $G_{i,2}$
b
a
c
Determine weights based on $G_{i,1}$
Determine weights based on Min. time among $G_{i,1}$, $G_{i,2}$
Determine weights based on $G_{i,2}$
Determine G_i
Set t_o, t_r, v_s, v_d V_p, V_s, V_o, V_r
Build G_o and G_r
Determine P_o, P_r, t_1, t_2, t', t'' (ILP 1)
$[t' \leq 0] \wedge [t'' \leq 0]$
$[t' > 0] \vee [t' > 0]$
Algorithm 1.3: PoI addition
exchange
Do not exchange
Algorithm 1.1: PoI Exchange
Algorithm 1.2: PoI Deletion
Print itinerary

Figure 2. The UML diagram of Algorithm 1: data initialization and itinerary planning.

3.1.1. Initialization Phase

The first phase of the algorithm concerns the initialization of data and it is divided into two parts. In the first part, the algorithm, by knowing the city map, creates the initial graphs of the city tourist attractions: a tourist attraction is associated with a node and all the nodes are connected to each other through indirect arcs. For each pair of nodes the time needed to go from one attraction to another is specified, on the basis of the transport means, with a weight associated with the arc that connects the nodes. In addition, the times to reach each attraction starting from the origin PoI and vice versa are also provided. In particular, two weighted graphs are initially considered, respectively named $G_{i,1}$ and $G_{i,2}$, composed by the same nodes and arcs, i.e., $V_{i,1} = V_{i,2}$ and $E_{i,1} = E_{i,2}$, where each node represents a PoI of the city, each arc indicates the connection between two nodes and the arc weight represent respectively the travel times by foot in $G_{i,1}$ and by bus/metro in $G_{i,2}$. Considering two nodes a and b of $G_{i,2}$, here we assume that the trip from a to b is performed mainly by bus and/or metro, with the possibility that a short segment of this trip must be traveled by foot. Moreover, for each node of $G_{i,1}$ and $G_{i,2}$ we are assuming that the travel time to go from the origin PoI to the node is equal to the travel time to go from the node to the origin PoI. After that, the graph $G_i = \{V_i, E_i\}$ is defined composed by the same nodes and arcs of $G_{i,1}$ and $G_{i,2}$. On each arc of G_i the weight is set among the following three possibilities, based on the user preferences:

(a) time preference: the weight of the arc is given by the minimum travel time among the corresponding ones of $G_{i,1}$ and $G_{i,2}$;
(b) foot preference: the weight of the arc is given by the travel time by foot on the corresponding arc of $G_{i,1}$;
(c) bus/metro preference: the weight of the arc is given by the travel time by bus/metro on the corresponding arc of $G_{i,2}$.

In addition, the resulting graph G_i also keeps track of the transport means used on each trip segment, i.e., by foot or by bus/metro.

In the second part of the initialization phase, the tourist sets the following preferences:

- t_o, time available for the outward trip and t_r, time available for the return trip;
- t_s, stop time at each node $v \in V_i$;
- define set $V_p \subseteq V_i$ of nodes of high priority and set $V_s \subseteq V_i$ of nodes of secondary priority with $V_p \cap V_s = \emptyset$;
- v_s source node and v_d destination node, among the nodes $v \in V_p$;
- define set $V_o \subset V_p$ and $V_r \subset V_p$, $V_o \cap V_r = \{v_d, v_s\}$, of the nodes to be visited on the outward and return journeys, respectively.

3.1.2. Itinerary Planning Phase

On the basis of the input from the initialization phase, the Algorithm 1 proceeds with the construction of two separate graphs G_o and G_r to determine the *outward* and *return* paths, respectively. The graph G_o and G_r are composed by the nodes $v \in V_o$ and $v \in V_r$, respectively. Moreover, the graphs G_o and G_r are of order N_o (cardinality of V_o) and N_r (cardinality of V_r), respectively, and are implemented through the adjacency matrices. Since the graphs are not oriented, connected and complete, the adjacency matrices are symmetric with a null diagonal. We solve the TSP (2) for the graphs G_o and G_r, respectively, in order to find the minimum path P_o and P_r and the associated travel time cost t_1 and t_2. Now, let us define the following integer linear programming problem ILP1 in order to maximize the available travel times for the outward and return paths:

$$F(\lambda) = \max \lambda$$

$$s.t.$$

$$\begin{cases} \lambda \leq t^{'} = t_1^{'} - t_o & \text{(3a)} \\ \lambda \leq t^{''} = t_1^{''} - t_r & \text{(3b)} \\ t_1^{'} = t_1 + t_{stop}(P_o) + y_1 * t_{stop,s} + y_2 * t_{stop,e} + t_p & \text{(3c)} \\ t_2^{''} = t_2 + t_{stop}(P_r) + y_3 * t_{stop,s} + y_4 * t_{stop,e} + t_p & \text{(3d)} \\ y_1 + y_3 = 1 & \text{(3e)} \\ y_2 + y_4 = 1 & \text{(3f)} \\ \lambda \in \mathcal{R} & \text{(3g)} \\ y_1, y_2, y_3, y_4 \in \{0, 1\} & \text{(3h)} \end{cases}$$

where λ is the real decision variable that has to be maximized in order to maximize the difference between the effective travel times $t_1^{'}$ and $t_2^{''}$ and the respective available times t_o and t_r. Let us specify that for the outward path P_o, the source PoI is set as the initial node while the destination PoI as the final node. On the contrary, for the return journey P_r the destination PoI is set as the initial node and the source PoI as the final node. Moreover, with constraints (3c) and (3d) ILP1 takes into account the following variables: the sum of the stop time period at each PoI of P_o and P_r, respectively called $t_{stop}(P_o)$ and $t_{stop}(P_r)$; the time period t_p to reach the origin PoI from the final point; the stop time period for visiting the source and destination nodes denoted respectively by $t_{stop,s}$ and $t_{stop,e}$. In particular, constraints, (3c), (3d), (3e) and (3f) are introduced to ensure that the source and end PoI can be visited only one time respectively on the outward or on the return journey. More precisely, constraint (3e) states that if the tourist stops for visiting the source PoI on the outward path, he/she will not repeat the visit on the return path: on the return path, the tourist will just pass through the source PoI without stopping there. The same statement of (3e) is done for the end/destination PoI by applying constraint (3f).

The resulting paths P_o and P_r are not definitive and a further analysis is required to satisfy the tourist preferences.

The travel times $t_1^{'}$ and $t_2^{''}$, must not exceed the time available for visiting t_o and t_r, respectively. Consequently, the following algorithms will manage the itinerary by adding and/or removing none, one or more PoI (nodes) from the path P_o and/or P_r so that the time constraints are respected. Now, considering that t_o and t_r are the available travel times to complete the outward and return journeys, respectively, two cases which needs to be managed can arise:

1. travel times exceed available times: $t^{'} > 0$ OR $t^{''} > 0$;
2. travel times do not exceed available times: $t^{'} \leq 0$ AND $t^{''} \leq 0$;

In particular, the management of case 1 is performed by Algorithm 1.1 and Algorithm 1.2, respectively described by the UML diagrams in Figures 3 and 4, while case 2 is managed through Algorithm 1.3 described by the UML diagram of Figure 5 in the following. At the end of Algorithm 1.1 and Algorithm 1.3, Algorithm 1 displays the final itinerary to the tourist.

Algorithm 1.1: PoI Exchange Procedure

In case 1, it is necessary to manipulate the outward and return paths, P_o and P_r, in order to respect the time constraints, to avoid delay in the origin PoI. An attempt is made to keep all the PoI of high priority by exchanging nodes between those selected for the outward and the return journeys. To this aim, a node belonging to the outward graph G_o is exchanged with a node belonging to the return graph G_r. Once the exchange has been made, the Algorithm 1.1 determines the new paths P_o and P_r and the travel time t_1 and t_2 by applying ILP1. Afterwards, it checks if the times $t^{'}$ and $t^{''}$ are positive or negative and one of the two cases can occur, as highlighted in Figure 3.

If case 1 occurs, Algorithm 1.1 updates the table FS of feasible solutions, i.e., records the paths P_o and P_r obtained by feasible nodes exchange. Afterwards, all the possible nodes exchange are tried between the two graphs (see Figure 3) and all the feasible solutions are recorded in Table FS. If case 2 occurs the Algorithm 1.1 ignores the obtained solution beacuse it is not feasible and proceeds with the node exchange procedure until other combinations are no longer possible.

After all possible nodes exchange have been made, the Algorithm 1.1 checks if suitable solutions have been found. If table FS is not empty, Algorithm 1.1 asks the tourist to indicate one of the solution in table FS and Algorithm 1.1 goes to Algorithm 1.3. On the contrary, if table FS is empty, i.e., no feasible node exchange are possible, Algorithm 1.1 goes to Algorithm 1.2 to start the node deleting procedure.

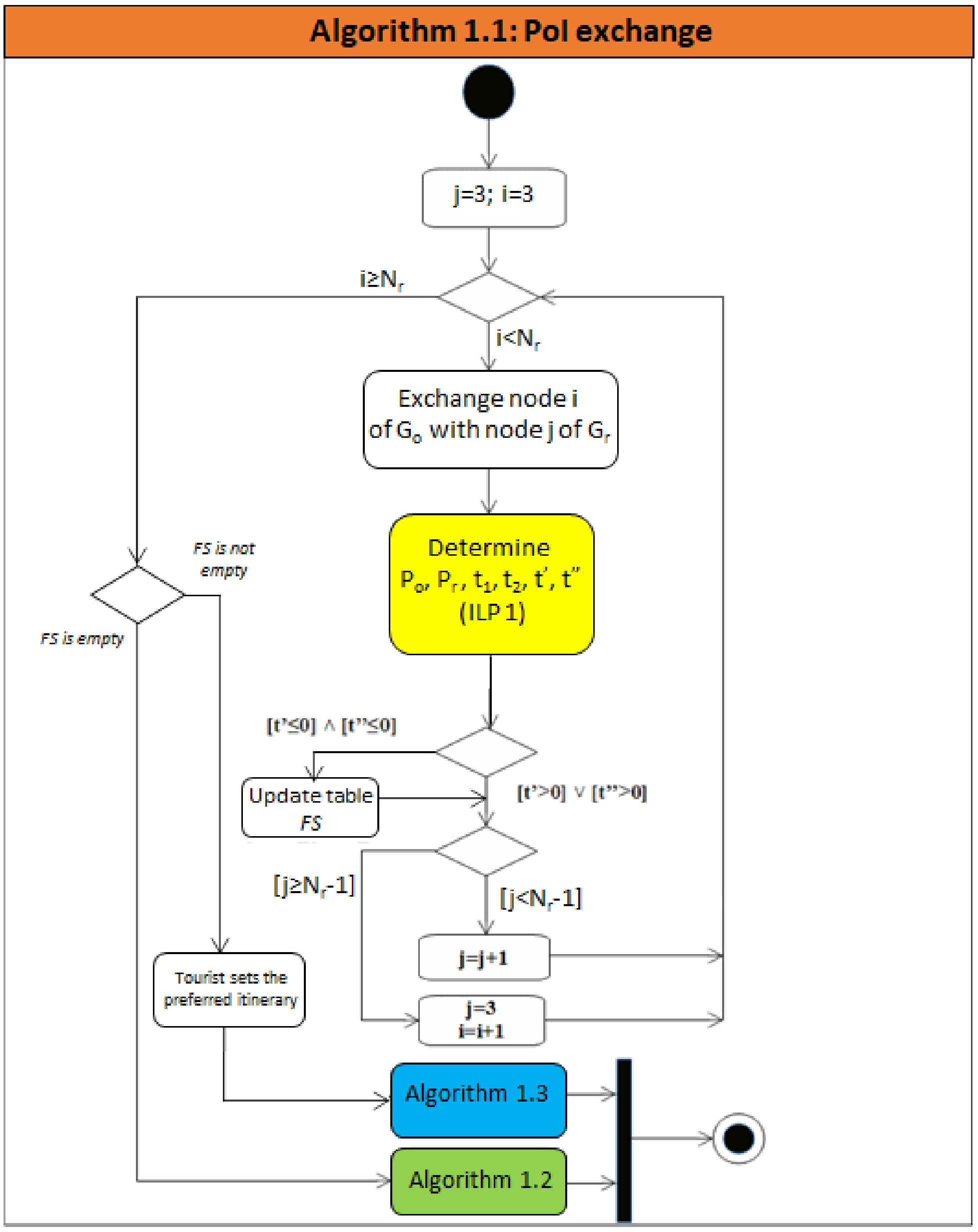

Algorithm 1.1: PoI exchange
j=3; i=3
i≥N_r
i<N_r
Exchange node i of G_o with node j of G_r
Determine P_o, P_r, t_1, t_2, t', t'' (ILP 1)
FS is not empty
FS is empty
[t'≤0] ∧ [t''≤0]
Update table FS
[t'>0] ∨ [t''>0]
[j≥N_r-1]
[j<N_r-1]
j=j+1
j=3 i=i+1
Tourist sets the preferred itinerary
Algorithm 1.3
Algorithm 1.2

Figure 3. The UML diagram for the node exchange procedure.

Algorithm 1.2: PoI Deletion Procedure

Algorithm 1.2 starts the node deleting procedure (see UML diagram of Figure 4). If $t' > 0$, a node is eliminated from G_o and, in case $t'' > 0$, a node is simultaneously deleted from G_r. After that, Algorithm 1.2 computes P_o and t_1, P_r, t_2 and, in particular, t' and t'' by applying ILP1. Then, the algorithm checks if the time constraints on t' and t'' are satisfied. These steps are repeated iteratively by Algorithm 1.2 until time constraints are not respected or no more node can be deleted. The Algorithm 1.2 displays an error message in case, after all possible nodes of G_o or G_r have been deleted, it still holds $t' > 0$ or $t'' > 0$, respectively. On the contrary, if $t' \leq 0$ and $t'' \leq 0$, the PoI elimination is not necessary anymore and the Algorithm 1.2 goes to Algorithm 1.3 where the possible addition of other points is evaluated.

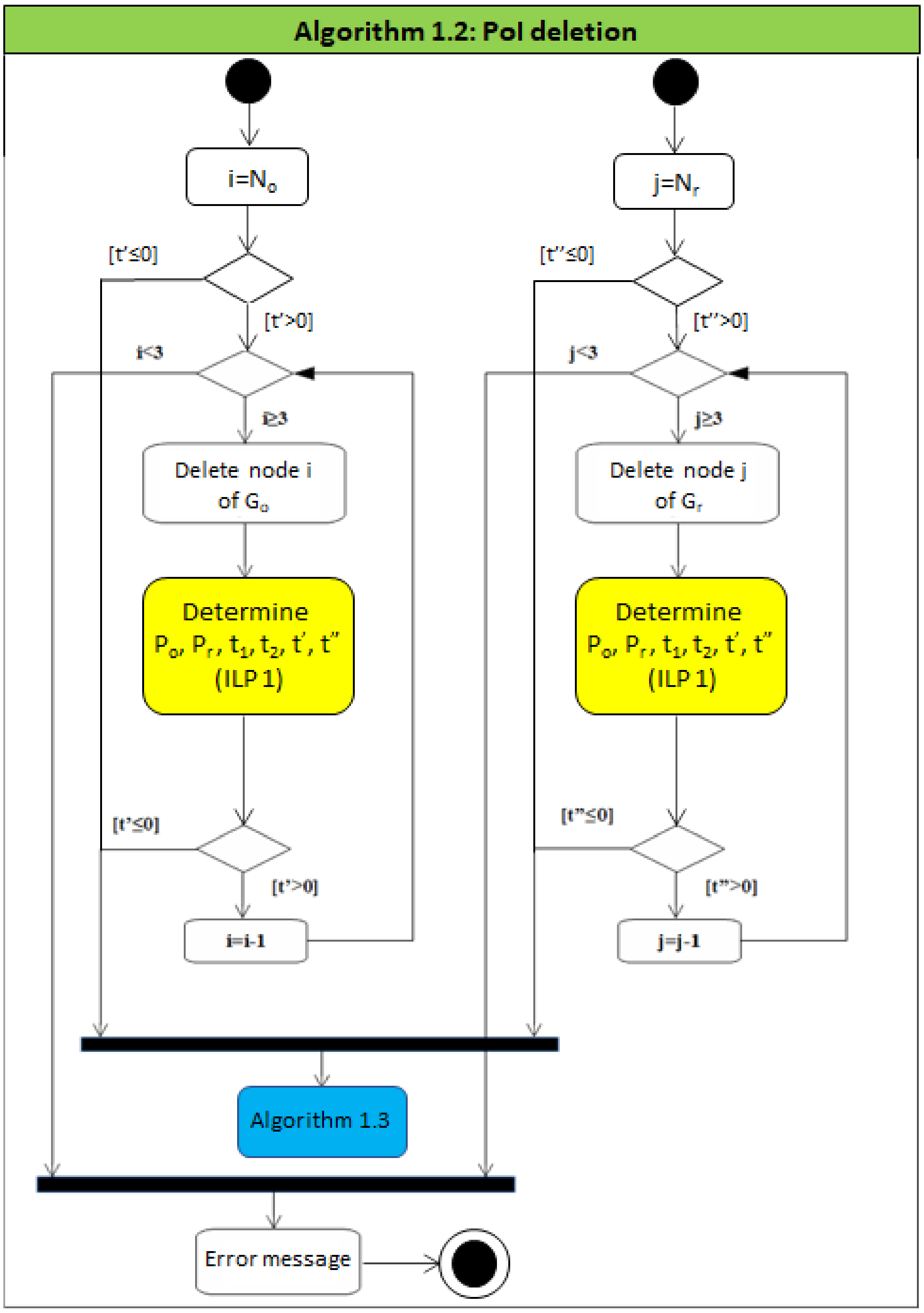


Figure 4. The UML diagram for the node deletion procedure.

Algorithm 1.3: PoI Addition Procedure

In the case $t^{'} \leq 0$ and $t^{''} \leq 0$, it is reasonable to add one or more nodes to the paths P_o and P_r. Thus, a node $v \in V_s$ with secondary priority is temporarily added to G_o and G_r. Note that N_s is the cardinality of V_s. At this point, if condition $t^{'} \leq 0$ and $t^{''} \leq 0$ is still verified the Algorithm 1.3 proceeds by adding another node $v \in V_s$, until no more nodes $v \in V_s$ can be added. On the other hand, if the addition of a node does not satisfy the time constraints, the added node is removed from the relative path and the Algorithm 1.3 checks whether it is possible to insert other nodes by repeating the operation until all nodes $v \in V_s$ are examined (see the UML diagram in Figure 5).

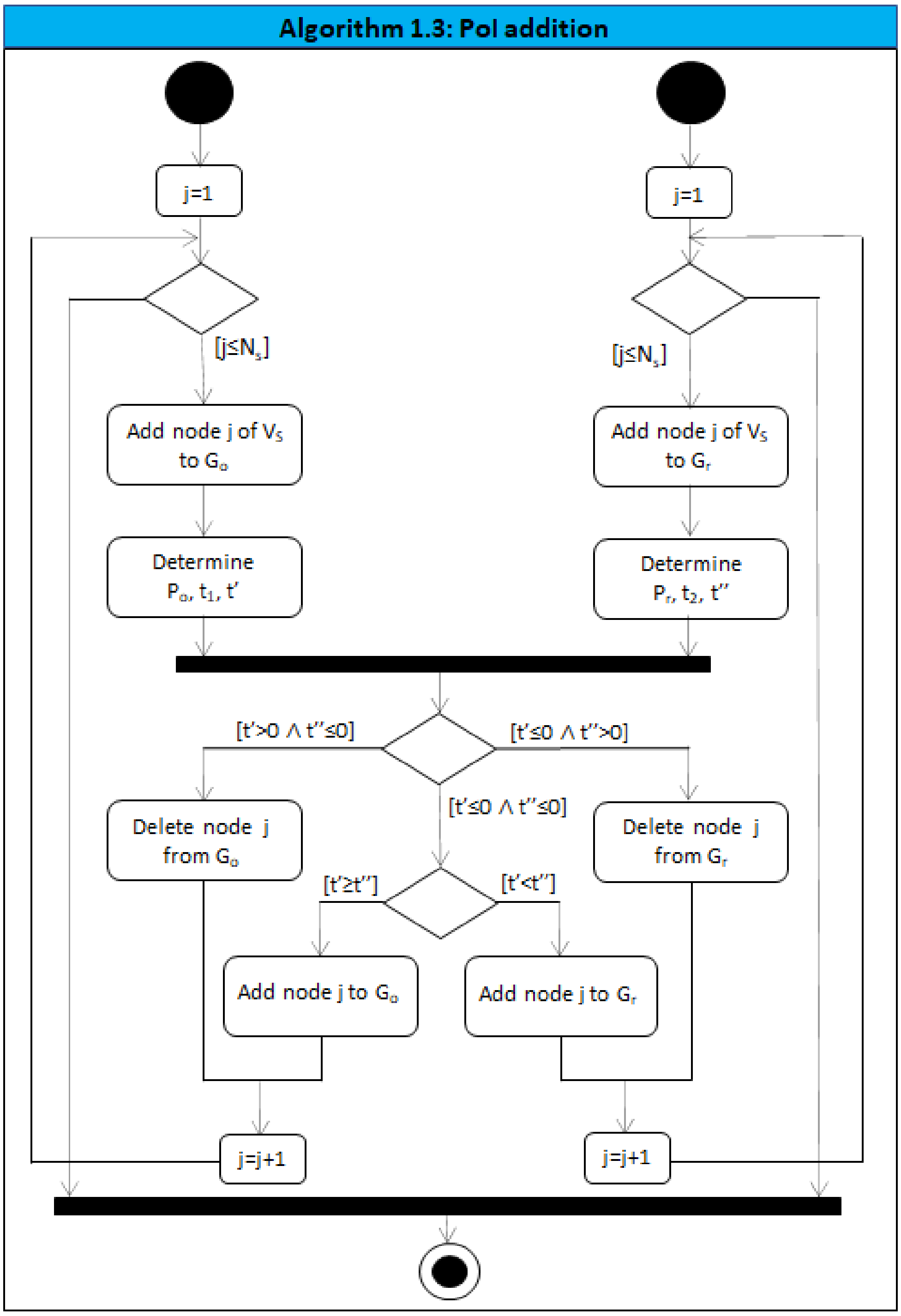


Figure 5. The UML diagram for the node addition procedure.

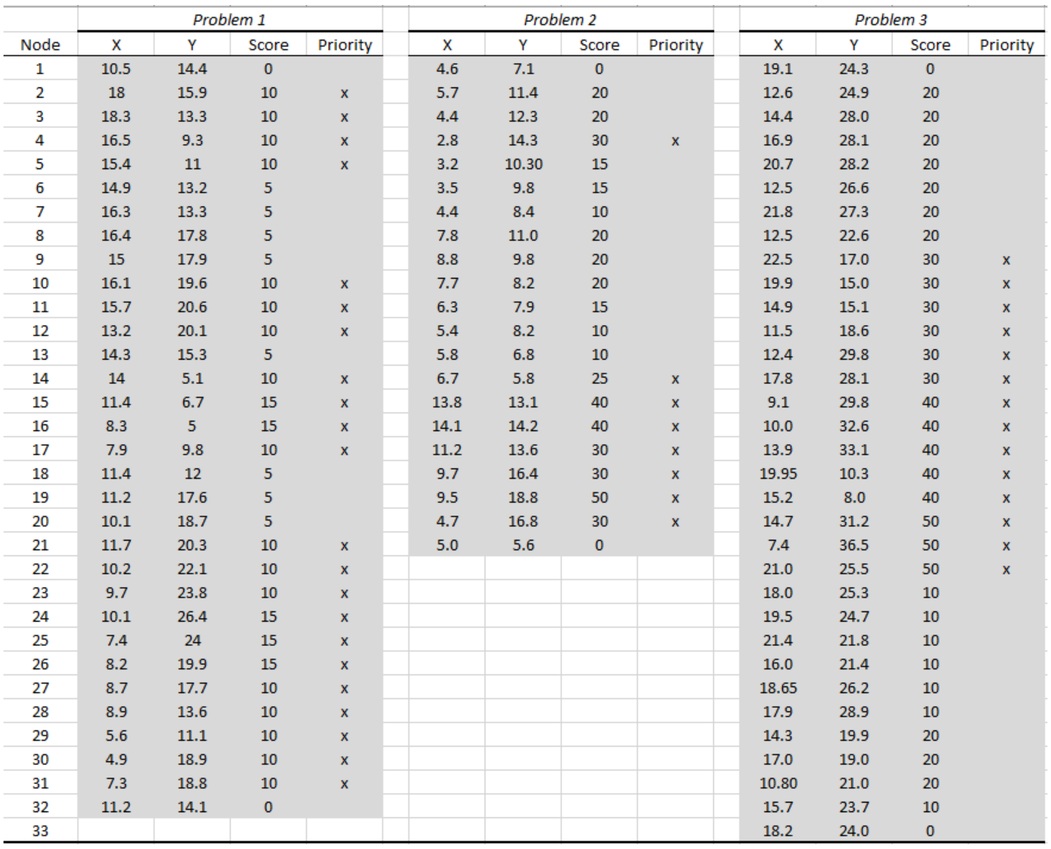
4. Complexity and Performance Analysis

In this section, the analysis of the complexity and performance of the proposed algorithms are provided. The following results describe the algorithms complexity:

- the Algorithm 1 requires the implementation of the ILP 1 problem including two TSP and 4 decision variables. Hence, considering a branch-bound approach, the complexity of Algorithm 1 is $O(2^4) + 2 * O(K) = O(K)$, with $O(K)$ TSP complexity;
- the Algorithms 1.1, 1.2 and 1.3 show complexity $O(N * K)$ since the ILP 1 problem is included in a N-dimensional "while" loop, where N is the number of PoI of graph G_i.

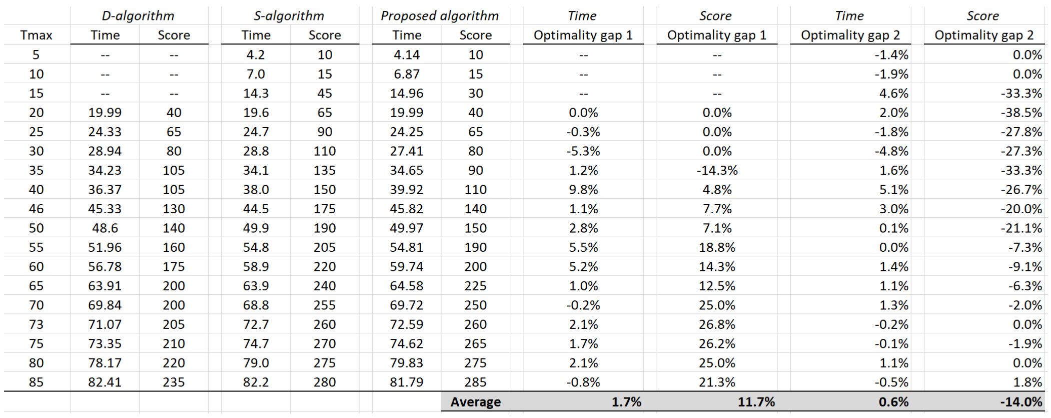
We can conclude that the complexity of the heuristic approach is $O(N * K)$. In order to be compliant with application time constraints, the TSP has been implemented using the Lin-Kernighan algorithm that often keeps its tours within 2% of the Held-Karp lower bound [48], then $K = O(N^{2.2})$. Therefore, our heuristic approach shows complexity $O(N^{3.2})$. Let us underline that the proposed application interacts with users to find the final best solution. Therefore, the time to complete the heuristic approach application and provide the final solution strongly depends on the velocity of the user given the necessary inputs to the application.

The performance of the proposed algorithms are validated with benchmark orienteering algorithms presented in [49]. In particular, the data set used in [49] and reported in [50] is used as benchmark (see Figure 6). Since, in the considered benchmark data set the PoI importance is defined by a score, we associate the priorities to the higher score values as reported in Figure 6. Note that in Figure 6, X and Y are the cartesian coordinates of the PoI and the PoIs distance is computed using the Euclidean distance formula [49]. Moreover, for comparison purpose, we assume that (i) the PoI V_d is selected as to minimize the total travel time, (ii) the Euclidean distance includes the stop time for visiting in our approach, (iii) the user preferences are randomly set in the instances simulation. Now, let us report the comparison results between the proposed heuristic and the D-algorithm and S-algorithm proposed by [49]. In particular, Figures 7–9 report respectively the comparison results considering data set of problem 1, 2 and 3, where Tmax represents the total travel time. Comparing the score and time values it can be concluded that the proposed approach performs much better than D-algorithm and little worse than the S-algorithm. However, in case only addition of PoI are needed by Algorithm 1.3, the proposed approach performs better than S-algorithm too. It is also remarked that the performance of Algorithm 1.2 can be further improved when a considerable number of PoIs must be deleted.



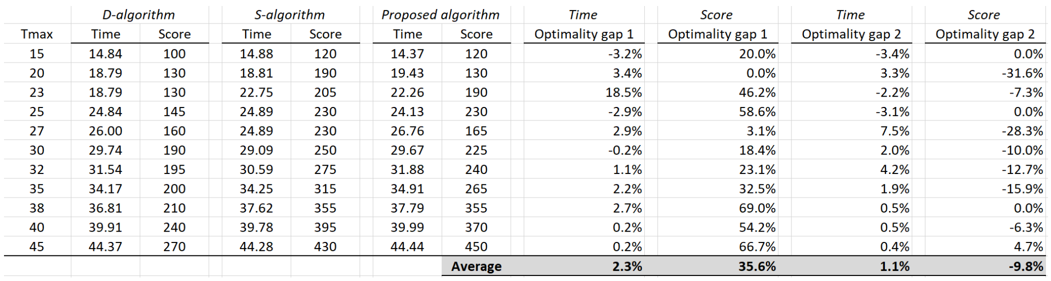
	Problem 1				*Problem 2*				*Problem 3*			
Node	X	Y	Score	Priority	X	Y	Score	Priority	X	Y	Score	Priority
1	10.5	14.4	0		4.6	7.1	0		19.1	24.3	0	
2	18	15.9	10	x	5.7	11.4	20		12.6	24.9	20	
3	18.3	13.3	10	x	4.4	12.3	20		14.4	28.0	20	
4	16.5	9.3	10	x	2.8	14.3	30	x	16.9	28.1	20	
5	15.4	11	10	x	3.2	10.30	15		20.7	28.2	20	
6	14.9	13.2	5		3.5	9.8	15		12.5	26.6	20	
7	16.3	13.3	5		4.4	8.4	10		21.8	27.3	20	
8	16.4	17.8	5		7.8	11.0	20		12.5	22.6	20	
9	15	17.9	5		8.8	9.8	20		22.5	17.0	30	x
10	16.1	19.6	10	x	7.7	8.2	20		19.9	15.0	30	x
11	15.7	20.6	10	x	6.3	7.9	15		14.9	15.1	30	x
12	13.2	20.1	10	x	5.4	8.2	10		11.5	18.6	30	x
13	14.3	15.3	5		5.8	6.8	10		12.4	29.8	30	x
14	14	5.1	10	x	6.7	5.8	25	x	17.8	28.1	30	x
15	11.4	6.7	15	x	13.8	13.1	40	x	9.1	29.8	40	x
16	8.3	5	15	x	14.1	14.2	40	x	10.0	32.6	40	x
17	7.9	9.8	10	x	11.2	13.6	30	x	13.9	33.1	40	x
18	11.4	12	5		9.7	16.4	30	x	19.95	10.3	40	x
19	11.2	17.6	5		9.5	18.8	50	x	15.2	8.0	40	x
20	10.1	18.7	5		4.7	16.8	30	x	14.7	31.2	50	x
21	11.7	20.3	10	x	5.0	5.6	0		7.4	36.5	50	x
22	10.2	22.1	10	x					21.0	25.5	50	x
23	9.7	23.8	10	x					18.0	25.3	10	
24	10.1	26.4	15	x					19.5	24.7	10	
25	7.4	24	15	x					21.4	21.8	10	
26	8.2	19.9	15	x					16.0	21.4	10	
27	8.7	17.7	10	x					18.65	26.2	10	
28	8.9	13.6	10	x					17.9	28.9	10	
29	5.6	11.1	10	x					14.3	19.9	20	
30	4.9	18.9	10	x					17.0	19.0	20	
31	7.3	18.8	10	x					10.80	21.0	20	
32	11.2	14.1	0						15.7	23.7	10	
33									18.2	24.0	0	

Figure 6. The benchmark data set.



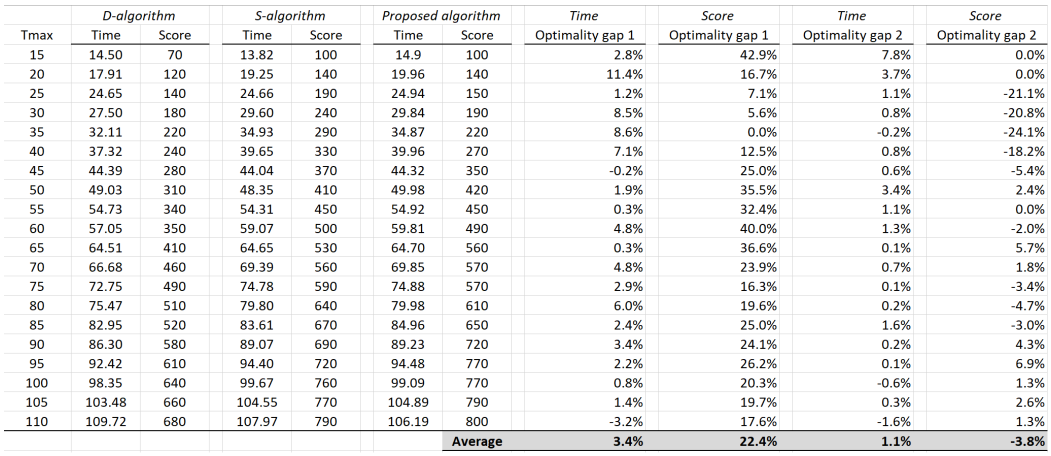
	D-algorithm		*S-algorithm*		*Proposed algorithm*		*Time*	*Score*	*Time*	*Score*
Tmax	Time	Score	Time	Score	Time	Score	Optimality gap 1	Optimality gap 1	Optimality gap 2	Optimality gap 2
5	--	--	4.2	10	4.14	10	--	--	-1.4%	0.0%
10	--	--	7.0	15	6.87	15	--	--	-1.9%	0.0%
15	--	--	14.3	45	14.96	30	--	--	4.6%	-33.3%
20	19.99	40	19.6	65	19.99	40	0.0%	0.0%	2.0%	-38.5%
25	24.33	65	24.7	90	24.25	65	-0.3%	0.0%	-1.8%	-27.8%
30	28.94	80	28.8	110	27.41	80	-5.3%	0.0%	-4.8%	-27.3%
35	34.23	105	34.1	135	34.65	90	1.2%	-14.3%	1.6%	-33.3%
40	36.37	105	38.0	150	39.92	110	9.8%	4.8%	5.1%	-26.7%
46	45.33	130	44.5	175	45.82	140	1.1%	7.7%	3.0%	-20.0%
50	48.6	140	49.9	190	49.97	150	2.8%	7.1%	0.1%	-21.1%
55	51.96	160	54.8	205	54.81	190	5.5%	18.8%	0.0%	-7.3%
60	56.78	175	58.9	220	59.74	200	5.2%	14.3%	1.4%	-9.1%
65	63.91	200	63.9	240	64.58	225	1.0%	12.5%	1.1%	-6.3%
70	69.84	200	68.8	255	69.72	250	-0.2%	25.0%	1.3%	-2.0%
73	71.07	205	72.7	260	72.59	260	2.1%	26.8%	-0.2%	0.0%
75	73.35	210	74.7	270	74.62	265	1.7%	26.2%	-0.1%	-1.9%
80	78.17	220	79.0	275	79.83	275	2.1%	25.0%	1.1%	0.0%
85	82.41	235	82.2	280	81.79	285	-0.8%	21.3%	-0.5%	1.8%
						Average	**1.7%**	**11.7%**	**0.6%**	**-14.0%**

Figure 7. Algorithms comparison using data set of problem 1.



	D-algorithm		*S-algorithm*		*Proposed algorithm*		*Time*	*Score*	*Time*	*Score*
Tmax	Time	Score	Time	Score	Time	Score	Optimality gap 1	Optimality gap 1	Optimality gap 2	Optimality gap 2
15	14.84	100	14.88	120	14.37	120	-3.2%	20.0%	-3.4%	0.0%
20	18.79	130	18.81	190	19.43	130	3.4%	0.0%	3.3%	-31.6%
23	18.79	130	22.75	205	22.26	190	18.5%	46.2%	-2.2%	-7.3%
25	24.84	145	24.89	230	24.13	230	-2.9%	58.6%	-3.1%	0.0%
27	26.00	160	24.89	230	26.76	165	2.9%	3.1%	7.5%	-28.3%
30	29.74	190	29.09	250	29.67	225	-0.2%	18.4%	2.0%	-10.0%
32	31.54	195	30.59	275	31.88	240	1.1%	23.1%	4.2%	-12.7%
35	34.17	200	34.25	315	34.91	265	2.2%	32.5%	1.9%	-15.9%
38	36.81	210	37.62	355	37.79	355	2.7%	69.0%	0.5%	0.0%
40	39.91	240	39.78	395	39.99	370	0.2%	54.2%	0.5%	-6.3%
45	44.37	270	44.28	430	44.44	450	0.2%	66.7%	0.4%	4.7%
						Average	**2.3%**	**35.6%**	**1.1%**	**-9.8%**

Figure 8. Algorithms comparison using data set of problem 2.



	D-algorithm		*S-algorithm*		*Proposed algorithm*		*Time*	*Score*	*Time*	*Score*
Tmax	Time	Score	Time	Score	Time	Score	Optimality gap 1	Optimality gap 1	Optimality gap 2	Optimality gap 2
15	14.50	70	13.82	100	14.9	100	2.8%	42.9%	7.8%	0.0%
20	17.91	120	19.25	140	19.96	140	11.4%	16.7%	3.7%	0.0%
25	24.65	140	24.66	190	24.94	150	1.2%	7.1%	1.1%	-21.1%
30	27.50	180	29.60	240	29.84	190	8.5%	5.6%	0.8%	-20.8%
35	32.11	220	34.93	290	34.87	220	8.6%	0.0%	-0.2%	-24.1%
40	37.32	240	39.65	330	39.96	270	7.1%	12.5%	0.8%	-18.2%
45	44.39	280	44.04	370	44.32	350	-0.2%	25.0%	0.6%	-5.4%
50	49.03	310	48.35	410	49.98	420	1.9%	35.5%	3.4%	2.4%
55	54.73	340	54.31	450	54.92	450	0.3%	32.4%	1.1%	0.0%
60	57.05	350	59.07	500	59.81	490	4.8%	40.0%	1.3%	-2.0%
65	64.51	410	64.65	530	64.70	560	0.3%	36.6%	0.1%	5.7%
70	66.68	460	69.39	560	69.85	570	4.8%	23.9%	0.7%	1.8%
75	72.75	490	74.78	590	74.88	570	2.9%	16.3%	0.1%	-3.4%
80	75.47	510	79.80	640	79.98	610	6.0%	19.6%	0.2%	-4.7%
85	82.95	520	83.61	670	84.96	650	2.4%	25.0%	1.6%	-3.0%
90	86.30	580	89.07	690	89.23	720	3.4%	24.1%	0.2%	4.3%
95	92.42	610	94.40	720	94.48	770	2.2%	26.2%	0.1%	6.9%
100	98.35	640	99.67	760	99.09	770	0.8%	20.3%	-0.6%	1.3%
105	103.48	660	104.55	770	104.89	790	1.4%	19.7%	0.3%	2.6%
110	109.72	680	107.97	790	106.19	800	-3.2%	17.6%	-1.6%	1.3%
						Average	**3.4%**	**22.4%**	**1.1%**	**-3.8%**

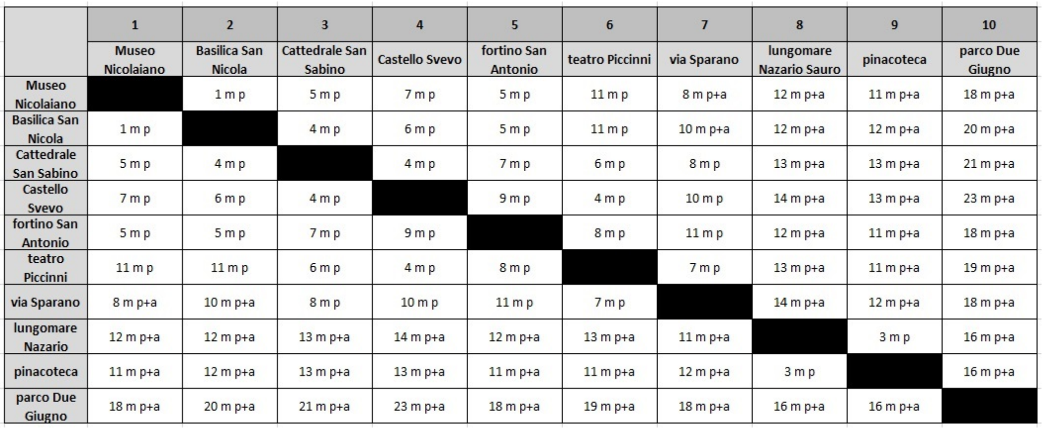
Figure 9. Algorithms comparison using data set of problem 3.

5. Case Study

This section presents a case study where the proposed multi-level algorithm approach is applied to solve the cruise tourist problem in the metropolitan port city of Bari. Bari is the capital city of Apulia Region and the second biggest city in southern Italy. Since the roman epoch, Bari became an important commercial center, during the Saracen domination. From 1071, it became a big maritime center and still today it is an important port hub of the Mediterranean sea.

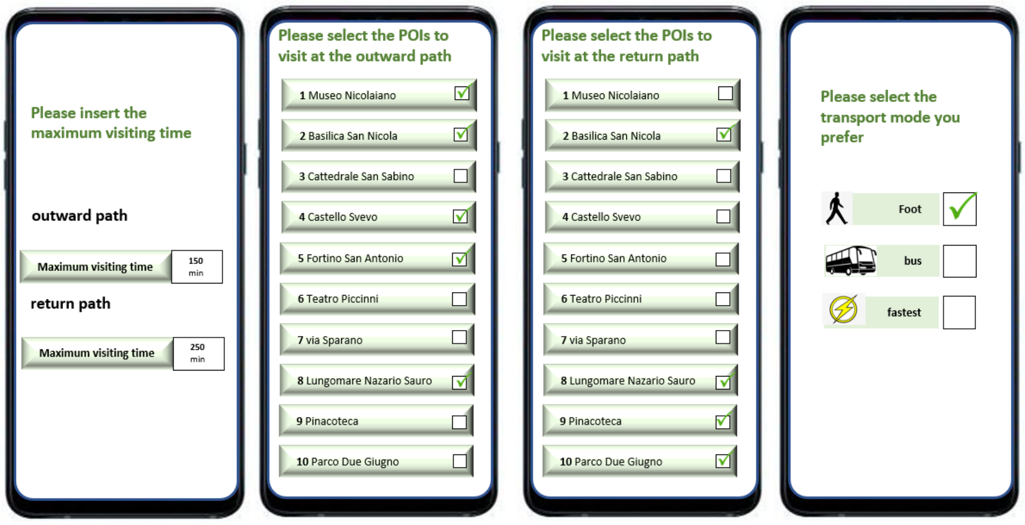
In the initialization phase of Algorithm 1, all the main attractions of Bari are determined and represented by the adjacency matrix of the graph G_i as it shown in Figure 10. Note that the weight of arcs are decided on the basis of the time preference (a) described in Section 3, according to the user. For better understanding of Figure 10 let us consider two examples: "1mp" in the box from PoI 2 to PoI 1 means that the tourist should travel by foot for 1 min; "10m p+a" in the box from PoI 2 to PoI 7 means that the tourist should travel by foot and by bus for 7 min total. The tourist inputs are provided in the initialization phase of the mobile application, where the tourist inserts the preferences and constraints related to: (i) the maximum available time for the outward and return path; (ii) the PoIs to visit during the outward and return path; (iii) the preferred transport mode, i.e., by foot, by bus or the fastest way (see Figure 11). Moreover for each node of the PoI network, the tourist can edit the selection of transport mode according to his/her preferences (see Figure 12).

Let the Saint Nicolas Basilica (node 2) be the starting point and Lungomare Nazario Sauro (node 8) be the final point of the tourist itinerary. Note that for path P_o the starting point is node 2 and the final point is node 8. On the contrary, for P_r the starting point is node 8 and the final point is node 2. In particular, let us define the nodes of priority level 1 (maximum priority) $V_p = \{1, 2, 4, 5, 8, 9, 10\}$, and the secondary nodes with $V_s = \{3, 6, 7\}$. Furthermore, the PoI to be visited on the outward and return paths and the stop time at each PoI are also reported in Figures 10 and 13.



	1	2	3	4	5	6	7	8	9	10
	Museo Nicolaiano	Basilica San Nicola	Cattedrale San Sabino	Castello Svevo	fortino San Antonio	teatro Piccinni	via Sparano	lungomare Nazario Sauro	pinacoteca	parco Due Giugno
Museo Nicolaiano	■	1 m p	5 m p	7 m p	5 m p	11 m p	8 m p+a	12 m p+a	11 m p+a	18 m p+a
Basilica San Nicola	1 m p	■	4 m p	6 m p	5 m p	11 m p	10 m p+a	12 m p+a	12 m p+a	20 m p+a
Cattedrale San Sabino	5 m p	4 m p	■	4 m p	7 m p	6 m p	8 m p	13 m p+a	13 m p+a	21 m p+a
Castello Svevo	7 m p	6 m p	4 m p	■	9 m p	4 m p	10 m p	14 m p+a	13 m p+a	23 m p+a
fortino San Antonio	5 m p	5 m p	7 m p	9 m p	■	8 m p	11 m p	12 m p+a	11 m p+a	18 m p+a
teatro Piccinni	11 m p	11 m p	6 m p	4 m p	8 m p	■	7 m p	13 m p+a	11 m p+a	19 m p+a
via Sparano	8 m p+a	10 m p+a	8 m p	10 m p	11 m p	7 m p	■	14 m p+a	12 m p+a	18 m p+a
lungomare Nazario	12 m p+a	12 m p+a	13 m p+a	14 m p+a	12 m p+a	13 m p+a	11 m p+a	■	3 m p	16 m p+a
pinacoteca	11 m p+a	12 m p+a	13 m p+a	13 m p+a	11 m p+a	11 m p+a	12 m p+a	3 m p	■	16 m p+a
parco Due Giugno	18 m p+a	20 m p+a	21 m p+a	23 m p+a	18 m p+a	19 m p+a	18 m p+a	16 m p+a	16 m p+a	■

Figure 10. Adjacency matrix of graph G_i.



Please insert the maximum visiting time

outward path

Maximum visiting time 150 min

return path

Maximum visiting time 250 min

Please select the POIs to visit at the outward path

- 1 Museo Nicolaiano ✓
- 2 Basilica San Nicola ✓
- 3 Cattedrale San Sabino ☐
- 4 Castello Svevo ✓
- 5 Fortino San Antonio ✓
- 6 Teatro Piccinni ☐
- 7 via Sparano ☐
- 8 Lungomare Nazario Sauro ✓
- 9 Pinacoteca ☐
- 10 Parco Due Giugno ☐

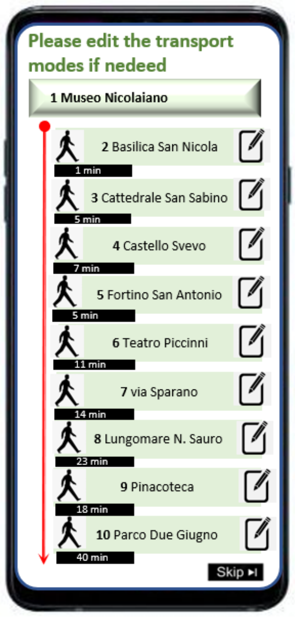
Please select the POIs to visit at the return path

- 1 Museo Nicolaiano ☐
- 2 Basilica San Nicola ✓
- 3 Cattedrale San Sabino ☐
- 4 Castello Svevo ☐
- 5 Fortino San Antonio ☐
- 6 Teatro Piccinni ☐
- 7 via Sparano ☐
- 8 Lungomare Nazario Sauro ✓
- 9 Pinacoteca ✓
- 10 Parco Due Giugno ✓

Please select the transport mode you prefer

- Foot ✓
- bus ☐
- fastest ☐

Figure 11. The GUI of the user preferences.


Please edit the transport modes if nedeed

1 Museo Nicolaiano

2 Basilica San Nicola
1 min

3 Cattedrale San Sabino
5 min

4 Castello Svevo
7 min

5 Fortino San Antonio
5 min

6 Teatro Piccinni
11 min

7 via Sparano
14 min

8 Lungomare N. Sauro
23 min

9 Pinacoteca
18 min

10 Parco Due Giugno
40 min

Figure 12. The GUI showing the travel times in the PoIs network from museo Nicolaiano. Preferred transport mode can be edit.

	1. Museo Nicolaiano	2. Basilica san Nicola	3. Cattedrale San Sabino	4. Castello Svevo	5. Muraglia e Fortino Sant'Antonio	6. Teatro Piccinni	7. Via Sparano	8. Lungomare Nazario Sauro	9. Pinacoteca Corrado Giaquinto	10. Parco due Giugno
stop time	60 min	30 min	30 min	50 min	10 min	60 min	120 min	40 min	60 min	10 min

Figure 13. Stop time at each PoI.

The Algorithm 1 creates the two graphs G_o and G_r by using only the nodes of G_i of priority level 1 (maximum priority) and calculates the travel times t_1 and t_2, as it is reported in Figure 14. It implies $V_o = \{1, 2, 4, 5, 8\}$ and $V_r = \{2, 8, 9, 10\}$. The travel times t_1 and t_2 do not include the time needed to go from the port to the first stop of the tour, i.e., node 2, and vice versa that is equal to 10 min. In Figure 15 the vector of the times to reach the port from each node is shown.

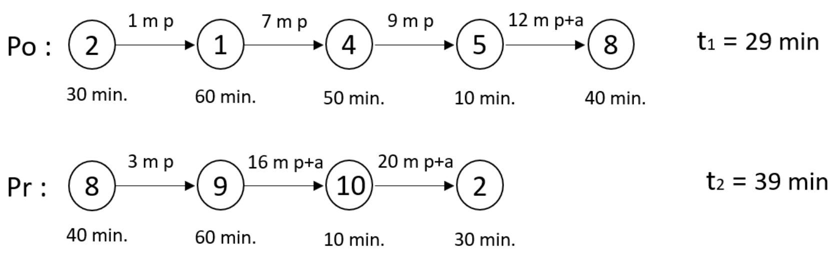


Figure 14. The Graphs G_o and G_r and the travel times t_1 and t_2.

	1. Museo Nicolaiano	2. Basilica san Nicola	3. Cattedrale San Sabino	4. Castello Svevo	5. Muraglia e Fortino Sant'Antonio	6. Teatro Piccinni	7. Via Sparano	8. Lungomare Nazario Sauro	9. Pinacoteca Corrado Giaquinto	10. Parco due Giugno
travel time to the port	9 min P	10 min P	12 min P	10 min A	11 min A	11 min A	15 min A	15 min A	15 min A	20 min A

Figure 15. Travel times from each node of G_i to the port.

Let us set $t_o = 150$ min and $t_r = 250$ min. Since $t' > 0$, as $t_1' = 159$ (obtained by the sum of the travel and stop times from one node to another in the outward path, except the stop times of node 2 and 8) and $t_2'' = 189$ (obtained by the sum of the travel and stop times from one node to another in the return path, considering also the stop times of node 2 and 8), this solution is not feasible.

Therefore, Algorithm 1 goes to Algorithm 1.1 that tries to exchange the nodes between graph G_o and G_r in order to obtain all the feasible solutions that are stored in *FS*. In the table FS, two solutions are stored which are obtained by the following two nodes exchanging: (i) node 4 of G_o with node 10 of G_r, (ii) node 1 of G_o with node 10 of G_r. In Figure 16, P_o and P_r of solution (i) obtained by solving ILP1 are shown. In this case $t' < 0$ and $t'' < 0$, since $t_1' = 130$ and $t_2'' = 212$ (node 2 and 8 are visited during the return path).

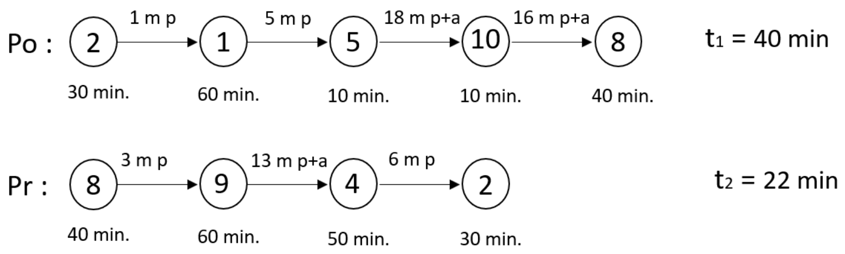


Figure 16. Algorithm 1.1 exchanges node 4 of G_o with node 10 of G_r

In Figure 17, P_o and P_r of solution (ii) obtained by solving ILP1 are shown. In particular, in this case $t' < 0$ and $t'' < 0$, since $t_1' = 129$ (node 2 and 8 are visited during the outward path) and $t_2'' = 215$.

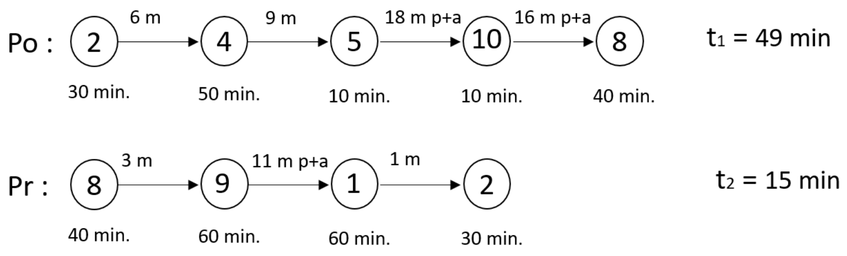


Figure 17. Algorithm 1.1 exchanges node 1 of G_o with node 10 of G_r

The two resulting feasible solutions are shown to the user that can select the preferred one in the mobile application (see Figure 18). Let us suppose that the tourist selects path (i). Therefore, Algorithm 1.1 goes to Algorithm 1.3 that tries to add nodes of priority 2 both on the outward and on the return paths. Since it implies that $t' > 0$, node 3 cannot be added to the route and it is removed. The adding of node 3 is checked for the outward path (see Figure 19). In this case, there is time available since $t'' < 0$ and $t_2'' = 244$. Therefore, node 3 with priority 2 is added to G_r. Moreover, since there are no other nodes of G_i to be added, the outward journey does not change.

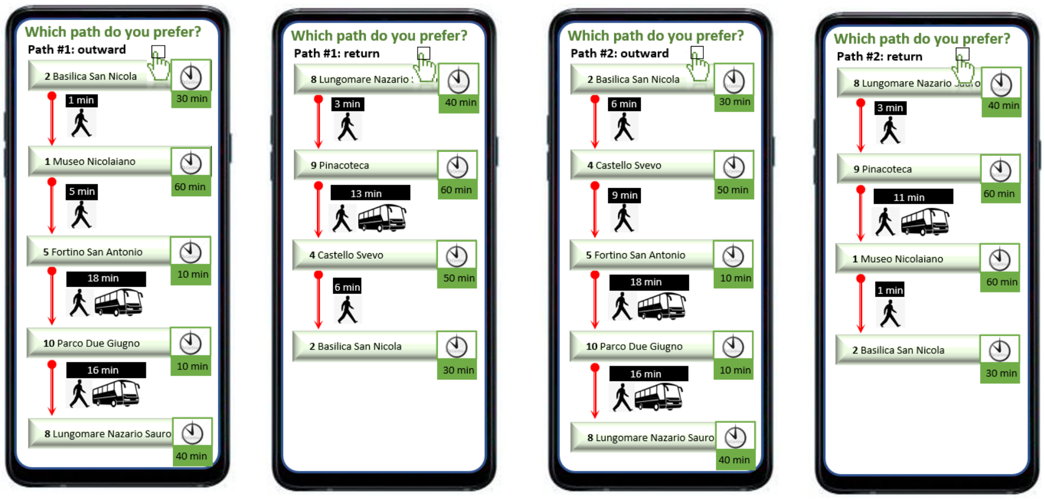


Figure 18. The GUI of the feasible solutions.

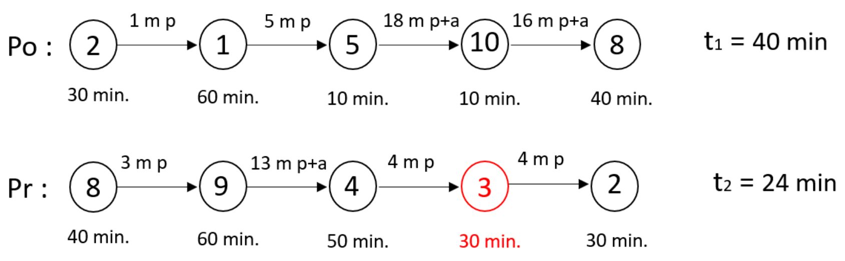


Figure 19. The outward path after adding node 3 to G_r.

Figures 20 and 21 depicts the GUI of the mobile application showing the final outward and return paths, respectively, connecting the identified PoIs.

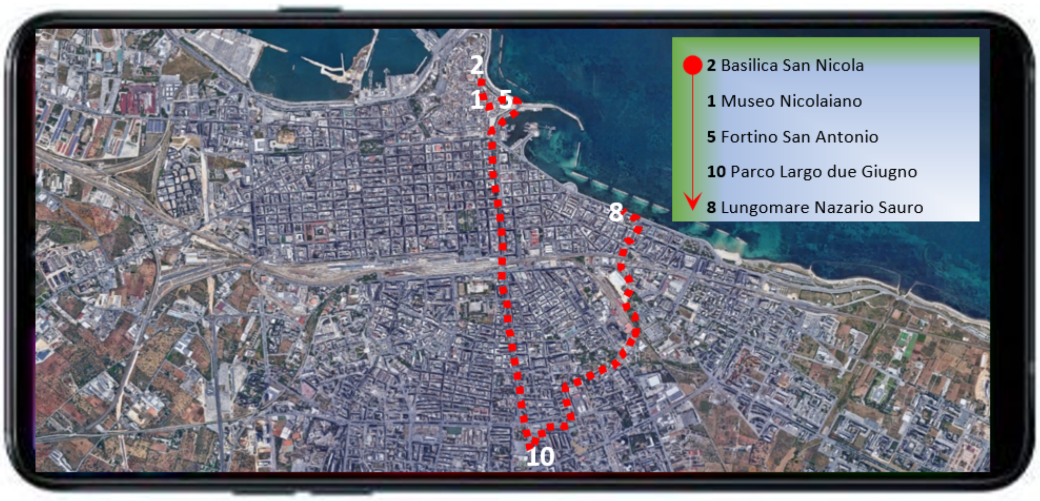


Figure 20. The GUI of the outward path.

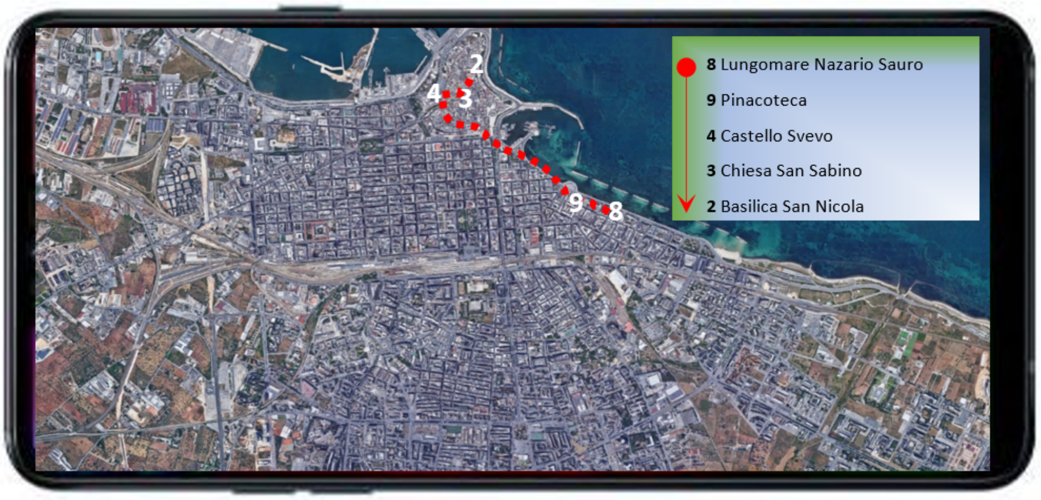


Figure 21. The GUI of the return path.

The procedure ends and the tourist can get the sequence of attractions in both trips with the relative means of transport. In the proposed case study, the mathematical calculations to obtain the final solution are performed in about 2 seconds.

The presented application demonstrates the effectiveness of the proposed heuristic approach in planning the itinerary for the one-day tourist, customizing the outward and return paths of the roundtrip in the city of Bari. In particular, the application can manage the map of city PoIs, showing the travel means and times for each PoI pair, decided based on the user preferences. The tourist preferences can initially be input through specific app pages like presented in Figures 11 and 12, where it is possible to decide which PoIs to be visited in the outward and return paths, respectively, and the preferred traveling mode. In particular, in the proposed case study it is evident how the Algorithm 1.1 recovers the initial unfeasible solution providing to the tourist alternative feasible trips including all the high priority PoIs. After the user choice, made as in the app page in Figure 20, Algorithm 1.3 further improves the solution adding node 3 belonging to secondary priority list, given one more PoI to be visited.

Let us remark that the obtained solution by applying the proposed procedure can be sub-optimal because of the human intervention. Indeed, with respect to a classical orienteering problem, the user subdivides the PoIs between outward and return trip and can choose a feasible itinerary according to the preferences affecting the real time procedure. Nevertheless, even if the obtained solution can be non optimal, it is surely customized based on the user preferences. Moreover, in Section 4, we enlighten that the proposed heuristic procedure shows better performances than other algorithms such as D-algorithm and S-algorithm [49] in some specific cases.

6. Conclusions

This paper is aimed at providing a tool to help the one-day tourist in the difficult choice to plan an itinerary in a city. Indeed, a tourist often renounces relying on professionals of the sector who offer a service although complete, often pre-packaged and not taking into account her/his passions and preferences.

The proposed approach builds a network of points of interest (PoIs), proposing the city attractions but leaving the choice of the PoIs to be visited by the tourist. Based on the obtained graph, a multi-algorithms approach is provided to determine the optimal itinerary. The tourist is part of the multi-algorithms approach interacting with it and taking decisions.

An Integer Linear Programming (ILP) problem is introduced to find the optimal outward and return paths of the touristic itinerary and the multi-algorithms strategy is used to maximize the number of PoIs to be visited in the paths. Moreover, a case study demonstrates the approach efficiency and the steps of the procedure.

Finally, an app prototype has been developed and several use cases are being tested to iron out bugs before writing the final code. Future works will focus on useful developments of the application: integrating the possibility of making reservations at hotels, purchasing entrance tickets for the various sites and promoting sustainable mobility by providing simple but complete and updated information on how to get around the city (bus/metro time schedule, opening/closing hours of attractions, etc.); locating bike-sharing stations and bicycle parking; considering the cost as additional objective function to be minimized. Furthermore, Algorithm 1.2 will be object of further studies in order to improve its performance.

Author Contributions: Conceptualization, A.M.M.; methodology, M.R.; software, A.R.; validation, A.M.M., M.R. and A.R.; formal analysis, A.R.; investigation, M.R.; resources, A.M.M.; data curation, A.R.; writing–original draft preparation, A.M.M. and M.R.; writing–review and editing, M.R. and A.R.; visualization, A.R.; supervision, M.R.; project administration, A.M.M.; funding acquisition, A.M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TSP	Traveling Salesman Problem
PoI	Point of Interest
UML	Unified Modeling Language
ILP	Integer Linear Programming

References

1. De Falco, I.; Scafuri, U.; Tarantino, E. Optimizing personalized touristic itineraries by a multiobjective evolutionary algorithm. *Int. J. Inf. Technol. Decis. Mak.* **2016**, *15*, 1269–1312. [CrossRef]
2. Angskun, T.; Angskun, J. A travel planning optimization under energy and time constraints. In Proceedings of the 2009 International Conference on Information and Multimedia Technology, Jeju, Korea, 16–18 December 2009; pp. 131–134.
3. Diosteanu, A.; Cotfas, L.A.; Smeureanu, A.; Dumitrescu, S.D. Natural language processing applied in itinerary recommender systems. In Proceedings of the 10th WSEAS International Conference on Applied Computer and Applied Computational Science, Venice, Italy, 8–10 March 2011; World Scientific and Engineering Academy and Society (WSEAS): Athens, Greece, 2011; pp. 260–265.
4. Vansteenwegen, P.; Souffriau, W.; Berghe, G.V.; Van Oudheusden, D. Iterated local search for the team orienteering problem with time windows. *Comput. Oper. Res.* **2009**, *36*, 3281–3290. [CrossRef]
5. Souffiau, W.; Maervoet, J.; Vansteenwegen, P.; Berghe, G.V.; Van Oudheusden, D. A mobile tourist decision support system for small footprint devices. In Proceedings of the International Work-Conference on Artificial Neural Networks, Salamanca, Spain, 10–12 June 2009; pp. 1248–1255.
6. Booth, J.; Sistla, P.; Wolfson, O.; Cruz, I.F. A data model for trip planning in multimodal transportation systems. In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, Saint Petersburg, Russia, 24–26 March 2009; pp. 994–1005.
7. Navabpour, S.; Ghoraie, L.S.; Malayeri, A.A.; Chen, J.; Lu, J. An intelligent traveling service based on SOA. In Proceedings of the 2008 IEEE Congress on Services-Part I, Honolulu, HI, USA, 6–11 July 2008; pp. 191–198.
8. André, P.; Wilson, M.L.; Owens, A.; Smith, D.A. Journey planning based on user needs. In Proceedings of the CHI'07 Extended Abstracts on Human Factors in Computing Systems, San Jose, CA, USA, 28 April–3 May 2007; pp. 2025–2030.
9. Gonzalez, H.; Han, J.; Li, X.; Myslinska, M.; Sondag, J.P. Adaptive fastest path computation on a road network: A traffic mining approach. In Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, Vienna, Austria, 23–27 September 2007; Association for Computing Machinery, Inc.: New York, NY, USA, 2007; pp. 794–805.
10. Garcia, A.; Linaza, M.T.; Arbelaitz, O.; Vansteenwegen, P. *Intelligent Routing System for a Personalised Electronic Tourist Guide*; ENTER: Amsterdam, The Netherlands, 2009; pp. 185–197.

11. Gavalas, D.; Konstantopoulos, C.; Mastakas, K.; Pantziou, G. A survey on algorithmic approaches for solving tourist trip design problems. *J. Heuristics* **2014**, *20*, 291–328. [CrossRef]
12. Gunawan, A.; Lau, H.C.; Vansteenwegen, P. Orienteering problem: A survey of recent variants, solution approaches and applications. *Eur. J. Oper. Res.* **2016**, *255*, 315–332. [CrossRef]
13. Kai, W. Operational Research Problems in Tour Itinerary Design and Optimization. *Tour. Sci.* **2004**, *1*. Available online: https://en.cnki.com.cn/Article_en/CJFDTotal-LUYX200401008.htm (accessed on 10 September 2021).
14. Han, Y.; Guan, H.; Duan, J. Tour route multiobjective optimization design based on the tourist satisfaction. *Discret. Dyn. Nat. Soc.* **2014**, *2014*, 603494. [CrossRef]
15. Kenteris, M.; Gavalas, D.; Pantziou, G.; Konstantopoulos, C. Near-optimal personalized daily itineraries for a mobile tourist guide. In Proceedings of the IEEE symposium on Computers and Communications, Riccione, Italy, 22–25 June 2010; pp. 862–864.
16. Cotfas, L.A. Collaborative itinerary recommender systems. *Acad. Econ. Stud. Econ. Inform.* **2011**, *11*, 191.
17. Vansteenwegen, P.; Souffriau, W.; Berghe, G.V.; Van Oudheusden, D. The city trip planner: An expert system for tourists. *Expert Syst. Appl.* **2011**, *38*, 6540–6546. [CrossRef]
18. Evans, J.R. *Graph Theory and Networks*; Industrial Engineering-New York Basel-Marcel Dekker Incorporated: New York, NY, USA, 1996; Volume 20, pp. 271–306.
19. Gavalas, D.; Kasapakis, V.; Konstantopoulos, C.; Pantziou, G.; Vathis, N. Scenic route planning for tourists. *Pers. Ubiquitous Comput.* **2017**, *21*, 137–155. [CrossRef]
20. Geng, J.; Ye, Q.; Wu, D.; Yang, H. Research of the evaluation and optimization of tourism route based on graph theory in Tibet. *Areal Res. Dev.* **2011**, *30*, 104–109.
21. Chen, J.; Tang, D. Tour routes optimization based on Graph theory for improving instruction services in scenic spots. In Proceedings of the ICSSSM11, Tianjin, China, 25–27 June 2011; pp. 1–3.
22. Le-Klaehn, D.T.; Hall, C.M. Tourist use of public transport at destinations—A review. *Curr. Issues Tour.* **2015**, *18*, 785–803. [CrossRef]
23. Van Truong, N.; Shimizu, T. The effect of transportation on tourism promotion: Literature review on application of the Computable General Equilibrium (CGE) Model. *Transp. Res. Procedia* **2017**, *25*, 3096–3115. [CrossRef]
24. Tussyadiah, I. A review of research into automation in tourism: Launching the Annals of Tourism Research Curated Collection on Artificial Intelligence and Robotics in Tourism. *Ann. Tour. Res.* **2020**, *81*, 102883. [CrossRef]
25. Gavalas, D.; Konstantopoulos, C.; Mastakas, K.; Pantziou, G.; Vathis, N. Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Comput. Oper. Res.* **2015**, *62*, 36–50. [CrossRef]
26. Pellegrini, A.; Scagnolari, S. The relationship between length of stay and land transportation mode in the tourism sector: A discrete–continuous framework applied to Swiss data. *Tour. Econ.* **2021**, *27*, 243–259. [CrossRef]
27. Abbaspour, R.; Samadzadegan, F. Itinerary planning in multimodal urban transportation network. *J. Appl. Sci.* **2009**, *9*, 1898–1906. [CrossRef]
28. Abbaspour, R.A.; Samadzadegan, F. Time-dependent personal tour planning and scheduling in metropolises. *Expert Syst. Appl.* **2011**, *38*, 12439–12452. [CrossRef]
29. Gavalas, D.; Kasapakis, V.; Konstantopoulos, C.; Pantziou, G.; Vathis, N.; Zaroliagis, C. The eCOMPASS multimodal tourist tour planner. *Expert Syst. Appl.* **2015**, *42*, 7303–7316. [CrossRef]
30. Wu, X.; Guan, H.; Han, Y.; Ma, J. A tour route planning model for tourism experience utility maximization. *Adv. Mech. Eng.* **2017**, *9*, 1687814017732309. [CrossRef]
31. Zheng, W.; Liao, Z.; Lin, Z. Navigating through the complex transport system: A heuristic approach for city tourism recommendation. *Tour. Manag.* **2020**, *81*, 104162. [CrossRef]
32. Zhang, R.; Liu, Z.; Feng, X. A novel flexible shuttle vehicle scheduling problem in scenic areas: Task-divided graph-based formulation and algorithm. *Comput. Ind. Eng.* **2021**, *156*, 107295. [CrossRef]
33. Kargar, M.; Lin, Z. A socially motivating and environmentally friendly tour recommendation framework for tourist groups. *Expert Syst. Appl.* **2021**, *180*, 115083. [CrossRef]
34. Wang, Y.W.; Lin, C.C.; Lee, T.J. Electric vehicle tour planning. *Transp. Res. Part D Transp. Environ.* **2018**, *63*, 121–136. [CrossRef]
35. Karbowska-Chilinska, J.; Chociej, K. Optimization of multistage tourist route for electric vehicle. In *Computer Science On-Line Conference*; Springer: Cham, Switzerland, 2018; pp. 186–196.
36. Karbowska-Chilinska, J.; Chociej, K. Genetic Algorithm for Generation Multistage Tourist Route of Electrical Vehicle. In Proceedings of the International Conference on Computer Information Systems and Industrial Management, Bialystok, Poland, 16–18 October 2020; pp. 366–376.
37. Sylejmani, K.; Dorn, J.; Musliu, N. Planning the trip itinerary for tourist groups. *Inf. Technol. Tour.* **2017**, *17*, 275–314. [CrossRef]
38. Malucelli, F.; Giovannini, A.; Nonato, M. Designing single origin-destination itineraries for several classes of cycle-tourists. *Transp. Res. Procedia* **2015**, *10*, 413–422. [CrossRef]
39. Korte, B.; Vygen, J. The traveling salesman problem. In *Combinatorial Optimization*; Springer: Cham, Switzerland, 2012; pp. 557–592.
40. Miller, C.E.; Tucker, A.W.; Zemlin, R.A. Integer Programming Formulation of Traveling Salesman Problems. *J. ACM* **1960**, *7*, 326–329. [CrossRef]

41. Abbatecola, L.; Fanti, M.P.; Mangini, A.M.; Ukovich, W. A decision support approach for postal delivery and waste collection services. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1458–1470. [CrossRef]
42. Silvestri, B.; Rinaldi, A.; Berardi, A.; Roccotelli, M.; Acquaviva, S.; Fanti, M.P. A Serious Game Approach for the Electro-Mobility Sector. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 674–679.
43. Souffriau, W.; Vansteenwegen, P. Tourist trip planning functionalities: State–of–the–art and future. In *Proceedings of the International Conference on Web Engineering*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 474–485.
44. Sylejmani, K.; Dika, A. A survey on tourist trip planning systems. *Int. J. Arts Sci.* **2011**, *4*, 13.
45. Bender, E.A.; Williamson, S.G. *Lists, Decisions and Graphs*; Williamson, S.G., Ed.; University of California: San Diego, CA, USA, 2010.
46. Costantino, N.; Dotoli, M.; Falagario, M.; Fanti, M.P.; Mangini, A.M.; Sciancalepore, F.; Ukovich, W. A fuzzy programming approach for the strategic design of distribution networks. In Proceedings of the 2011 IEEE International Conference on Automation Science and Engineering, Trieste, Italy, 24–27 August 2011; pp. 66–71.
47. Pender, T. *UML Bible*; John Wiley & Sons: Hoboken, NJ, USA, 2003.
48. Nilsson, C. Heuristics for the traveling salesman problem. *Linkop. Univ.* **2003**, *38*, 85–89.
49. Tsiligirides, T. Heuristic methods applied to orienteering. *J. Oper. Res. Soc.* **1984**, *35*, 797–809. [CrossRef]
50. The Orienteering Problem: Test Instances Benchmark Dataset. Available online: https://www.mech.kuleuven.be/en/cib/op (accessed on 10 September 2021).