An Efficient Algorithm for Recommending Personalized Mobile Tourist Routes

Damianos Gavalas[1], Michael Kenteris[1], Charalampos Konstantopoulos[2] and Grammati Pantziou[3]

[1] Department of Cultural Technology and Communication
University of the Aegean
Mytilene, Greece
Email: dgavalas@aegean.gr, m.kenteris@ct.aegean.gr

[2] Department of Informatics, University of Piraeus
Piraeus, Greece
Email: konstant@unipi.gr

[3] Department of Informatics, Technological Educational Institution of Athens,
Athens, Greece
Email: pantziou@teiath.gr

Abstract
This article deals with the problem of deriving personalized recommendations for daily sightseeing itineraries for tourists visiting any destination. Our approach considers selected places of interest that a traveller would potentially wish to visit and derives a near-optimal itinerary for each day of visit; the places of potential interest are selected based on stated or implied user preferences. Our method enables the planning of customised daily personalised tourist itineraries considering user preferences, time available for visiting sights in daily basis, opening days of sights and average visiting times for these sights. Herein, we propose a heuristic solution to this problem addressed to both web and mobile web users. Evaluation and simulation results verify the competence of our approach against an alternative method.

Keywords: Online Web Application; Itinerary; Team Orienteering Problem; Route Planning; Maps; Mobile Tourism.

I. INTRODUCTION

Tourists that visit a destination for one or multiple days are unlikely to visit every tourist sight; rather, tourists are dealt with the dilemma of which points of interest (POIs) would be more interesting for them to visit. These choices are normally based on information gathered by tourists via the Internet, magazines, printed tourist guides, etc. After deciding of which sights to visit, tourists have to decide on which route to take, i.e. the order in which to visit each POI, with respect to the visiting time required for each POI, the POI's visiting days/hours and the time available for sightseeing in daily basis.

Tourists encounter many problems following this procedure. The information contained in printed guide books is often outdated (e.g. the opening times of some museums might have changed or some other memorial sites might be closed due to maintenance works, etc), the weather conditions might be prohibitive during one of

the visiting days to visit an important POI, etc [8]. The selection of the most important and interesting POIs for visiting also requires fusion of information typically provided from separate -often non credible- sources. Usually tourists are satisfied if a fairly attractive or feasible route is derived, yet, they cannot know of any alternative routes which would potentially be better to follow. Some tourist guides do acknowledge such problems and try to propose more generalized tourist routes to a city or an area. Of course these routes are designed to satisfy the likes of the majority of its readers but not those with specialized interests, needs or constraints [4].

Mobile tourist guides may be used as tools to offer solution to these types of problems [5],[15],[19]. Based on a list of personal interests, up-to-date information for the sight and information about the visit (e.g. date of arrival and departure, accommodation address, etc), a mobile guide can suggest near-optimal and feasible routes that include visits to a series of sights, as well as recommending the order of each sight's visit along the route [24]. Generalized tourist routes do not take into consideration the context of the user e.g. the starting or ending point of the user, the available time the user affords, the current time, predicted weather conditions while on journey, etc. Taking into account the parameters of context and location awareness brings forward a challenge for the design of appropriate tourist routes [18]. Kramer et al. [17] analyzed the interests in the profiles of each tourist and concluded that they particularly varied from each other. This conclusion supports the argumentation for deriving personalized instead of generalized tourist routes.

Given a list of sights of some tourist destination in which a user-tourist would potentially be interested in visiting, the problem involves deriving the order in which the tourist should visit the selected POIs, for each day the tourist stays at that destination. We term this problem as the 'tourist itinerary design problem' (TIDP). Interestingly, the TIDP presents similarities to problems which have arisen in the past in the field of operational research; such problems reside upon the mathematical theory of graphs (graph theory) and comprise variations of the well-known travelling salesman problem (TSP).

For instance, the team orienteering problem (TOP) appoints an initial and final point as well as *N* points for visiting, where each point is associated with a 'score' or 'profit'. Given a particular time margin for each of the *M* team members, the TOP determines *M* routes (from the initial to the end point) via a subset of *N* points, aiming at maximizing the overall profit of visited points [3]. The TOP cannot be solved in polynomial time (NP-complete) [22], hence heuristics deriving near-optimal solutions are the only realistic way to tackle such problems, especially when considering online applications. TOP can be thought of as a starting point to model TIDP whereby the *M* team members are reduced to the number of days available for the tourist to stay and the profit of a sight signifies the potential interest (or degree of satisfaction) of a particular tourist visiting the POI within a given time span available for sightseeing daily (therefore, TOP considers the time spent while visiting each POI as well as the time needed to travel from one POI to another).

Nevertheless, TOP does not take into consideration the POIs' visiting days and hours. Therein, the resemblance of TIDP with another operational research problem (travelling salesman problem with time windows, TSPTW) [7] comes forward. TSPTW concerns the minimum cost path for a vehicle which visits a set of nodes. Each node must be visited only once and the visit must be carried out inside an allowed time interval (time window). The correlation of time windows with the POIs visiting days/hours is obvious. However, TSPTW involves planning of only one route (i.e. not *M*, as many as days available to the tourist to visit POIs), while it requires the vehicle to visit the whole set of nodes. A generalization of TOP and TSPTW is referred to as team orienting problem with time windows (TOPTW) [23] and considers multiple vehicles (i.e. itineraries) that should visit a subset of nodes, each within its allowed time window.

The main contribution of this paper lies in modelling and investigating a generalization of TOPTW through introducing a novel heuristic that provides near-optimal solutions to TIDP: the Daily TouRist Itinerary Planning (DailyTRIP). It is noted that some preliminary ideas of our technique have also been presented in [16].

The remaining of this article is organized as follows: Related work is discussed in Section II. The modelling, design and implementation of DailyTRIP are presented in Sections III, IV and V, respectively, Section VI presents simulation results while Section VII draws conclusions and grounds for future work.

II. Related Work

The issue of personalized tourist itineraries has not been looked at in the electronic and mobile tourism literature, with the exception of the algorithms proposed in [21] and [22]. In [21], Souffriau et al. proposed a heuristic solution for the orienteering problem, i.e. they only consider a single tourist itinerary. The algorithm presented in [22] deals with TOPTW; however, it does not take into account neither the opening days of sites nor the time needed to visit a sight, i.e. it makes the unrealistic assumption of zero visiting duration.

Other relevant research projects with respect to tourist itineraries have been reported in [1], [9]. In [1] a method for deriving a single multi-modal tourist itinerary is proposed considering a variety of constraints and following a genetic algorithm-based approach. However, derived route recommendations are not personalized as user preferences about specific types of sites are not taken into account. P-Tour [20], Dynamic Tour Guide (DTG) [13] and City Trip Planner [6] address these shortcomings; however, they derive routes day-by-day. The City Trip Planner currently offers the most advanced route planning functionalities considering several user constraints. Google city tours application [10] represents another interesting approach along the same line suggesting multiple daily itineraries through the familiar Google maps interface. Yet, the suggested itineraries are not personalized. Furthermore, both [1] and [9], city tours implementations are only provided through a web interface and have not been tested on mobiles; hence they lack location-based and context-aware features. On the other hand, P-Tour and DTG have been implemented on mobiles, but can only deal with a small number of POIs (i.e. their scalability is questionable).

III. DailyTRIP Modelling

DailyTRIP modelling involves the definition and the description of the user model, visit model and the sight (POI) model (see Figure 1) taking into consideration parameters/ constraints like those listed below:

- User Model:
  - device (e.g. screen resolution, available storage space, processing power, etc);
  - language of content, localization;
  - personal ‘demographic’ data (e.g. age, educational level);
  - interests (explicit declaration or implicitly collected);
  - disability (e.g. blind, deaf, kinetic disability);
  - budget threshold willing to spend on sightseeing.
- Visit Model:
  - geographical location of accommodation;
  - period of stay (arrival and departure date);
  - time constraints (e.g. available time each day to tour, number and duration of desirable breaks, etc);
  - means of travel (e.g. walking, driving, bus, metro, etc).
- Sight (POI) Model:
  - category (e.g. museum, archaeological site, monument, etc);
  - available multimedia resources (collection of texts, video, audio, etc, localized in different languages;
  - geographical position (coordinates);
  - weight or ‘objective’ importance (e.g. the Acropolis of Athens is thought to be ‘objectively’ more important of the Coin Museum of Athens, hence the Acropolis is assigned a larger weight);

- average duration of visit (e.g. the Archaeological Museum of Athens typically takes longer to visit than the city's Coin Museum due to size difference and the nature of exhibition);
- rating/comments of users;
- opening days/hours (time windows), which could be provided by the web service of an administrative body or the Ministry of Culture;
- whether it is a indoor/outdoor site;
- whether it is a accessible from people with disabilities;
- admission price (ticket prices).

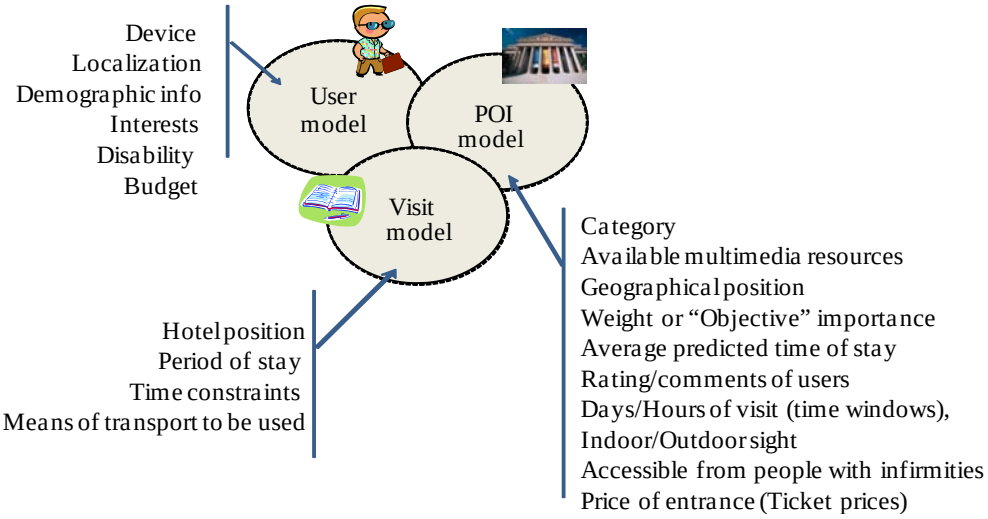


Figure 1. Description of user, visit and sight models in TIDP.

Notably, the above stated parameters/constraints are not exhaustive. From those parameters the below listed elements may be easily derived:

- The topological distance (or Manhattan distance) among the POIs and also among the accommodation and the POIs, based on their geographical coordinates and the local map.
- The number of routes that must be generated are based upon the period of stay of the user at the tourist destination.
- The anticipated duration of visit of a user at a POI derives from the average duration and the user's potential interest (concluded by examining the user's profile).
- The ability to visit open air sites in a particular day during the user's visit, e.g. outdoor sites are not recommended to visit during a rainy day (meteorological forecasts can be retrieved from an Internet web service).

The problem's definition also includes the 'profit' of a POI, calculated as a weighted function of the objective and subjective importance of each POI (subjectivity refers to the users' individual preferences). Our algorithmic solution maximizes the overall profit, i.e. it enables the construction of personalized routes which include the most important (for each tourist) sights under specific constraints (opening hours, weather conditions, time available for sightseeing). The most crucial constraint in seeking sound algorithmic solutions is the daily time limit *T* which a tourist wishes to spend on visiting sights; the overall daily route duration (i.e. the sum of visiting times plus the overall time spent moving from a POI to another which is a function of the topological distance) should be kept below *T*.

IV. DailyTRIP: A Heuristic for Deriving Near-Optimal Personalized Daily Tourist Itineraries

A. Problem Statement

The TIDP problem involves a complete graph *G=(V,E),* $|V| = n$, where each node i, $i=0,\ldots,n-1$, in V corresponds to a POI and each edge *(i,j)* in E corresponds to the shortest path (in terms of Manhattan distance $d_{i,j}$) linking individual POIs i and j.

Each POI $i \in V$ is associated with a weight w_i which denotes the 'objective' importance of the POI and a profit vaue p_i, which reflects the importance of that POI for a particular user and depends on her personal preferences. Each POI i is also associated with a set of days $D_c(i)$ when visiting is not feasible (e.g. Mondays and during some bank holidays) and the anticipated visit duration of the user at the POI $t_v(i)$; similarly to the profit, $t_v(i)$ also depends on the user's personal preferences (for instance, someone interested in archaeology is expected to take longer to visit an archaeological museum than others).

The cost of each edge *(i,j)* $c_{i,j}$, namely the cost of visiting j after visiting i, is a weighted function of travelling time from i to j $t_{i,j}$ (the latter depends on the Manhattan distance $d_{i,j}$ between i and j and the means of travel), the profit of the arriving node p_j and the duration of visit at the arriving node $t_v(j)$: $c_{i,j} = a_1 \cdot t_{i,j} - a_2 \cdot p_j + a_3 \cdot t_v(j)$, where a_1, a_2 and a_3 are weight coefficients. This formula signifies that being on node i, the next itinerary stop j has to be a node of relatively high profit that takes short to arrive and visit. Notably, $c_{i,j} \neq c_{j,i}$.

Travelers typically plan to visit the area for a set of days, *D.* Users also define a starting and ending time for their daily itineraries, T_{start} and T_{end}, which denote what time she prefers to depart from his starting point S and arrive at her end point (destination) E. Hence, a daily time budget devoted to visiting sights may be easily calculated: $T = T_{end} - T_{start}$. Without loss of generality, we assume that that the starting and end points of the $|D|$ daily itineraries coincide, i.e. $S \equiv E$ (typically these will coincide with the user's accommodation H).

Summarizing, the objective of DailyTRIP is to derive $|D|$ itineraries I_i that maximize the overall profit $\sum_{i=1}^{|D|} \sum_{j=1}^{|I_i|} p_j$, ensuring that the time needed to complete each itinerary does not exceed the user-defined daily time budget T, i.e. $T(I_i) \leq T$.

B. The DailyTRIP algorithm flow

DailyTRIP comprises the following execution phases:

Phase 1: Definition of the problem's model

The first phase first involves the definition of problem's space, i.e. the nodes of G, the nodes' weight w_i and the travelling time matrix $t_{i,j}$ that denotes the time needed to travel between node pairs (see Figure 2a); notably, $t_{i,j} \neq t_{j,i}$, since the route $i \rightarrow j$ differs from the route $j \rightarrow i$ due to considering one-way roads. Taking into consideration personalization issues (e.g. in a simplified scenario, user preferences upon POIs' categories), the cost matrix (i.e. the cost values $c_{i,j}$ associated with the two-directional edges) as well as the nodes' profit p_i and visit duration $t_v(i)$ with respect to a specified user are also computed.

Phase 2: Reduction of the problem's space

The initial set of sights around the tourist destination is sorted in decreasing order of profit p, where the value of p mainly depends on its category (i.e. whether the POI is museum or an architectural monument) and the user's preference upon this category. To reduce the computational effort required to reach valid solutions (i.e. to reduce the problem's space) we discard:

- nodes (POIs) with profit p_i smaller than a threshold value $p_{\min}$

- POIs located too far from the origin point *H*, i,e, every node *v* for which $t_{H,v} > t_{max}$, where t_{max} is an upper time limit (see Figure 2b).

An alternative approach would be to exclude the relatively low-profit POIs located far from *H*, i.e. exclude every POI *i* for which $a_1 * p_i - a_2 * d_{i.H} < t$, where a_1 and a_2 are weight coefficients and *t* a threshold value.

Phase 3: Selection of first daily itinerary nodes

DailyTRIP determines the $|D|$ POIs that will be the first to include in the $|D|$ daily itineraries I_i, where *i*=1.. $|D|$. We select the set of $|D|$ nodes $\{N_i\}$, where *i*=1.. $|D|$, located furthest apart from one another, i.e. those for which the minimum distance from one another is the maximum among any other permutation of $|D|$ nodes. For instance, in the example topology of Figure 2c, assuming that $|D|$=3, we select the nodes *i*, *j* and *k* that: $\max_{i,j,k} min\{d_{i,j}, d_{i,k}, d_{j,k}\}$. Then, the $|D|$ daily itineraries are initialized, each incorporating one of those nodes: $I_i = \{N_i\}, \forall i = 1..|D|$. The philosophy behind this approach is to achieve a geographical 'segmentation' of the tourist destination in separate zones, as it makes sense, for example, for a tourist that visits a city for two days, to focus on POIs located on the northern part of the city on her first day and on POIs located on the southern part on her second day.

Phase 4: Construction of itinerary trees

On each of the following algorithm's steps, itineraries I_i are considered interchangeably incorporating a new node *N* not yet included in any of the I_i. In particular, for each I_i, the candidate node *N* with the minimum connection cost $c_{j,N}$ to any of the nodes $j \in I_i$ joins I_i (through accepting the *j*→*N* edge), given that the daily time budget *T* condition is not violated for this itinerary (see Figure 2d). Notably, as the candidate node *N* may be connected to any of the I_i nodes (i.e. not necessarily to the edge nodes of the itinerary), I_i grows as a tree structure rather than a multipoint line. The time T_i corresponding to the completion of the itinerary I_i is calculated first by temporarily connecting *H* with the I_i node nearest to *H*, then converting the I_i itinerary tree to a multipoint line (through a post-order tree traversal) and finally calculating: $T(I_i) = t_{H,1} + \sum_{k=1}^{|I_i|}(t_v(k) + d_{k,k+1})$. Namely, $I_i = I_i \cup N$, if $T(I_i) \leq T$. This is illustrated in Figure 2e.

Hence, on each step itineraries I_i grow interchangeably (see Figure 2f,g), typically approaching the start/end point *H*, until no further insertion is feasible. Upon completion, each itinerary is connected to the 'hotel' node *H*, i.e. the edge *j*→*N* is accepted, where *j* is the itinerary's node nearest to *H* (see Figure 2h).

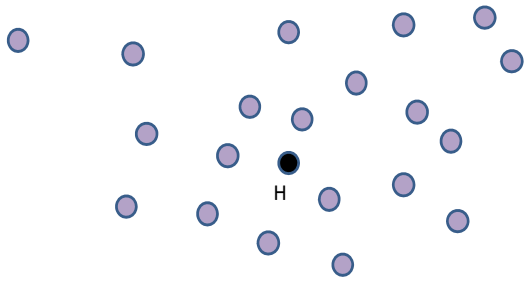
It is noted that the acceptance of candidate nodes also depends on the corresponding POIs' scheduled visiting days. In particular, for each joining node *i* that may not be visited during the days $D_c(i)$, the 'excluded' days of the itinerary *I* joined by *i* is adapted excluding those days: $D_c(I) = \bigcup_{i=1}^{|I|} D_c(i)$, signifying that during those days the itinerary is not feasible either. Apparently, a POI *i* may join an itinerary *I* if the intersection of their valid days (i.e. those when visiting is feasible) is not null and also this intersection includes at least one of the *D* days of visit, namely if $D_v(I) \cap D_v(i) \cap D \neq \emptyset$.

Phase 5: Rearrangement of itinerary trees

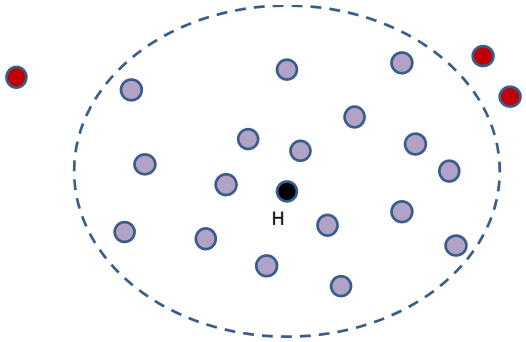
Phase 5 is optional and aims at improving the solutions derived in the previous phase, i.e. either increasing the overall profit or maintaining the same profit while reducing the itinerary completion time $T(I)$ (see Figure 2i). Improved solutions are searched for every itinerary by: (a) substituting each itinerary tree node by any node not included in any itinerary at the end of the previous phase, (b) by swapping nodes included on different itineraries. In any case, the new itinerary solutions should satisfy the daily time budget constraint.

Phase 6: Traversal of itinerary trees

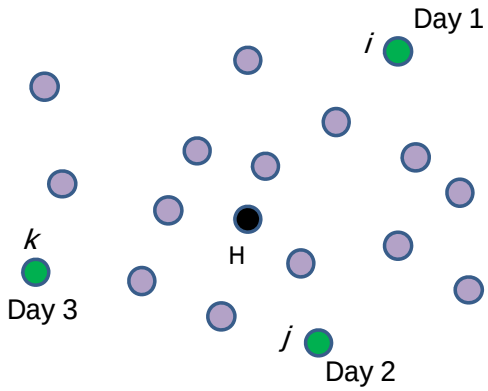
Notably, the outcome of the previous phases is not a set of itineraries, but rather a set of itinerary trees. Hence, the last phase of DailyTRIP involves the conversion of the $|D|$ trees to multipoint lines I_i through executing a heuristic TSP algorithm [14] upon each itinerary tree (see Figure 2j).


H

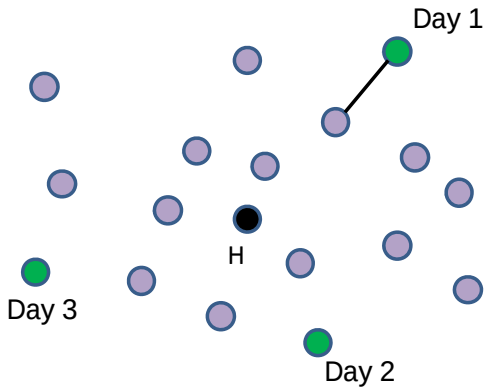
(a)


H

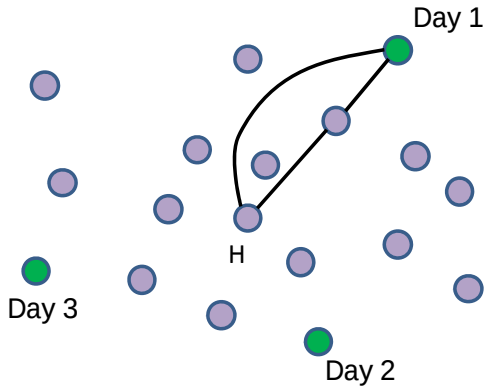
(b)


Day 1
i
k
H
Day 3
j
Day 2

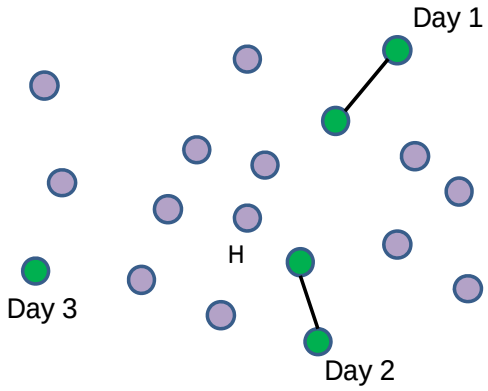
(c)


Day 1
H
Day 3
Day 2

(d)


Day 1
H
Day 3
Day 2

(e)


Day 1
H
Day 3
Day 2

(f)

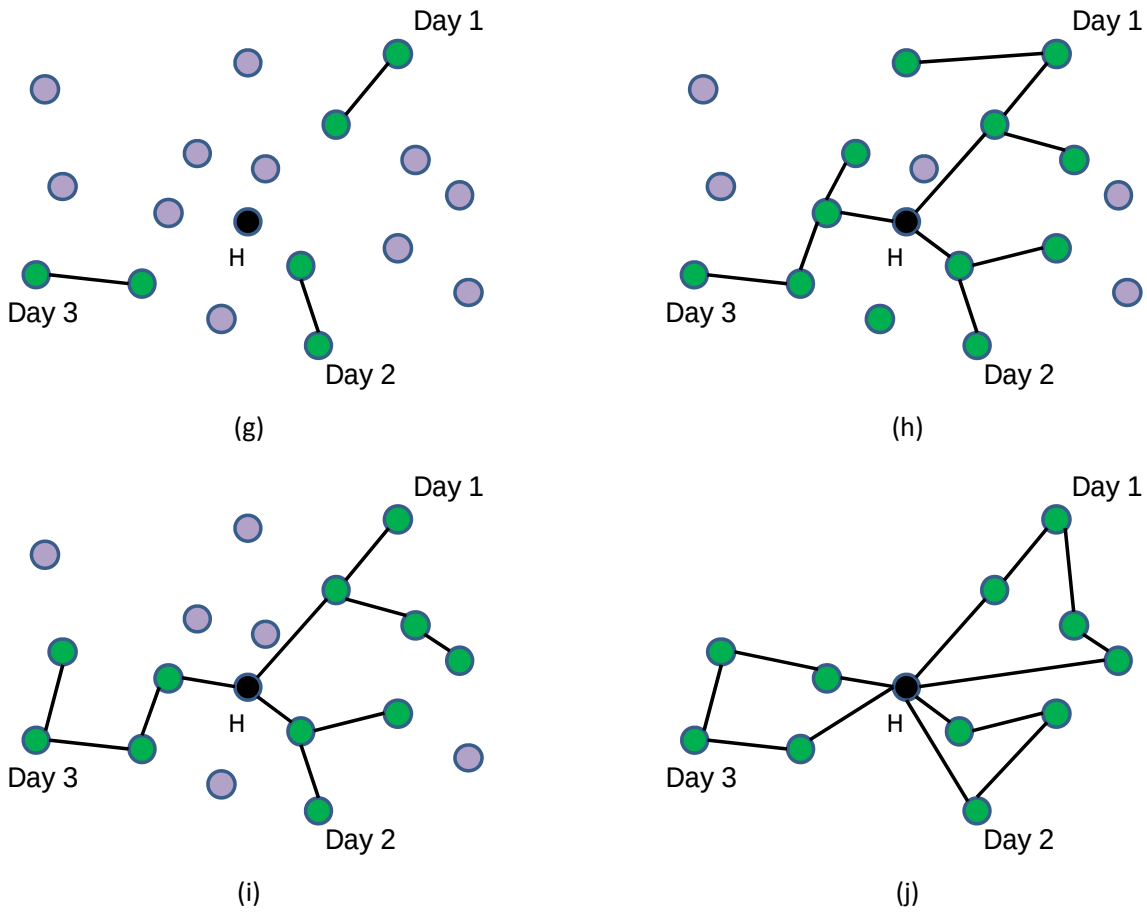


Figure 2. Execution phases of DailyTRIP.

V. Implementation Details of DailyTRIP

DailyTRIP has been developed using JSP/MySQL web technologies and Google Maps as the main user interface. The user first provides some personal demographic data and preferences upon tourist content items, i.e. she may state preference in visiting museums, archaeological sites, monuments, etc. Further, she points the location *H* of her accommodation, the period of visit, the hours available for visit, the means of transport and the radius around the hotel she is willing to move in order to visit a POI. In the mobile application case, the user is provided the option to receive recommendations for itineraries that originate at her current location (instead from her hotel). The user is then shown a list of the initially selected POIs based on her preferences, which she is allowed to modify adding/ removing POIs.

The algorithm filters the POIs left out from the problem's space (due to their distance from the user's accommodation, their incompatibility with the user's preferences or their intentional removal by the user) and populates the travelling time matrix $t_{i,j}$ for the remaining nodes through first computing the distance matrix entries and considering the average expected velocity *v* of the selected means of transport. Distances amongst pairs of nodes are found by means of using the shortest-route functionality of the Google Maps API [11] which refers to Manhattan distances and takes into account one-way roads.

Our implementation is based on the following assumptions: (a) each daily itinerary starts and ends at the same node, which typically coincides with the user's accommodation; (b) among all possible routes between a

pair of nodes we only consider the shortest route in terms of length, although this might not be shortest in time; (c) the user is assumed to move with constant velocity regardless of the traversed edge or the time of day (admittedly, this is a valid assumption only for tourists walking around a city); (d) the POIs are assumed to be open for visiting during the hours available to the tourist for sightseeing.

As stated in Section IV, phase 6 of DailyTRIP involves the conversion of the |D| trees to multipoint lines through a heuristic TSP algorithm. The latter is based on the Lin-Kernighan heuristic [14] and was implemented in Java. The Lin-Kernighan heuristic is considered as one of the most efficient algorithms for deriving near optimal solutions for the symmetric travelling salesman problem.

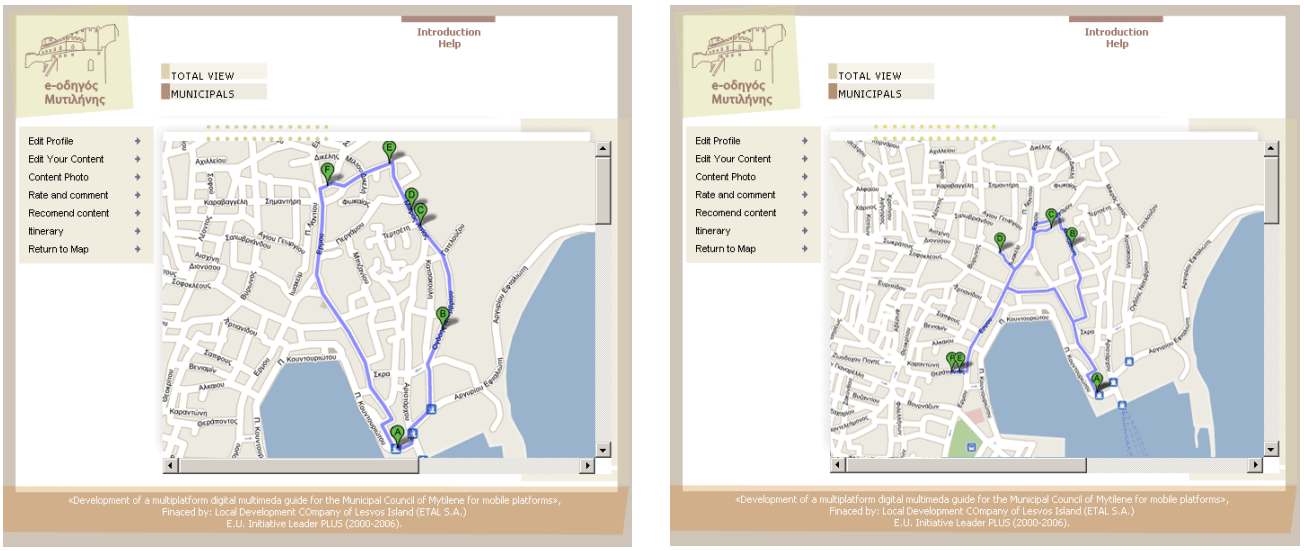
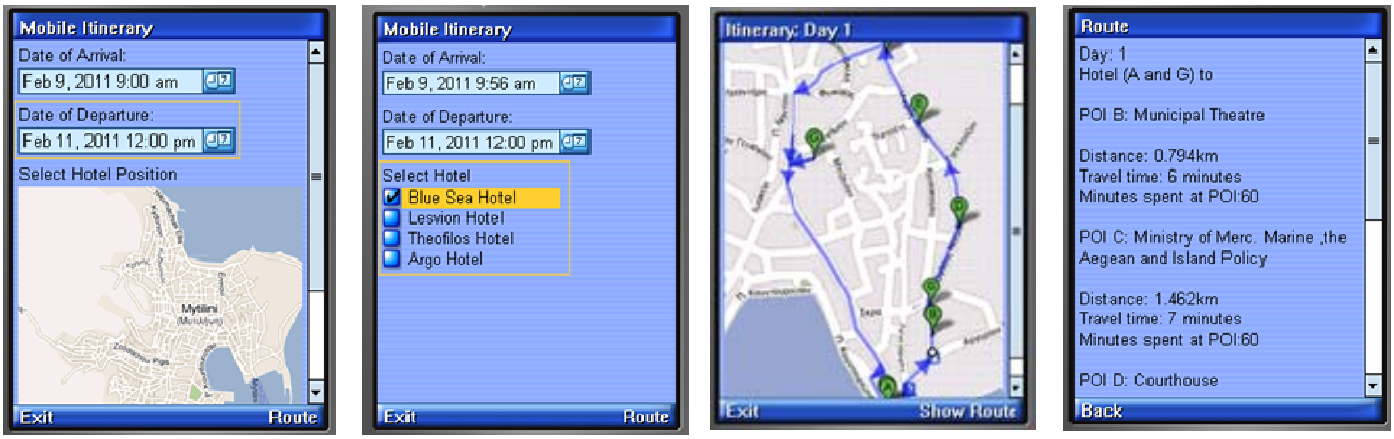


Figure 3. Output of DailyTRIP for two daily itineraries on Google Maps (point 'A' denotes the user's accomodation location, i.e. the start/end point of the two itineraries).

The output of DailyTRIP is sketched on a Google Maps interface, with each itinerary drawn on separate screen and the order of visiting POIs denoted by the alphabetical order of characters representing POIs (see Figure 3). The maps derived by the web application are then converted to static images using the Google Static Maps API [12] in order to display on mobile phone screens. The mobile application (see Figure 4) prompts the user for specifying the period of stay at the destination and determining the start/end location of the itineraries (that may be the current location of the user). The derived routes are then displayed upon a static map, accompanied by a textual description.



(a) (b) (c) (d)

Figure 4. Screens of the DailyTRIP mobile web application: (a) accommodation's selection through a map interface; (b) accomodations' selection through a set of predefined choices; (c) visualization of a daily itinerary; (d) textual description of a daily itinerary.

It is noted that mobile users may update their itineraries at any time, if the deviate from the originally calculated itineraries. Those updates take into account contextual parameters, such as the remaining time budget and the sites already visited by the user. Currently, we are working on an intelligent mechanism that will trigger automated itinerary updates when significant deviations from the original schedules are observed.

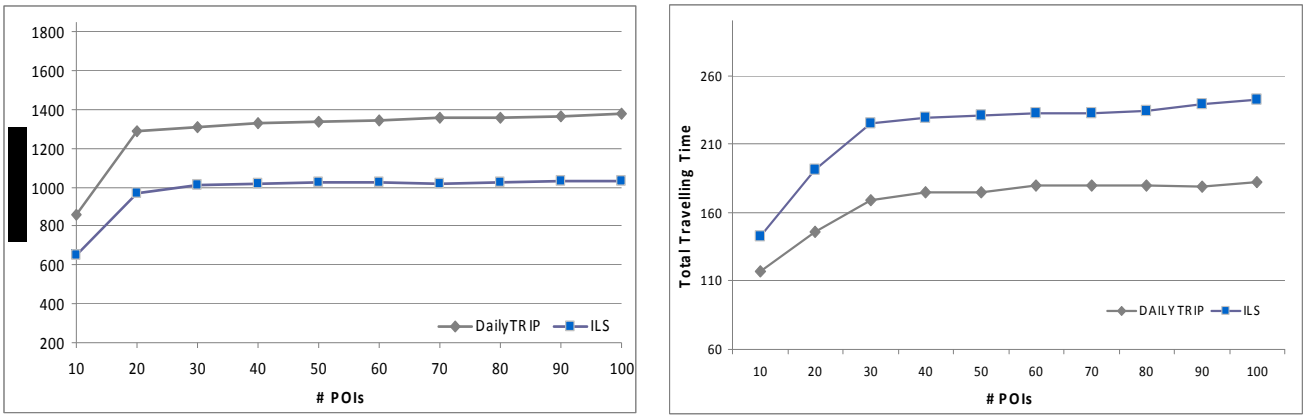
VI. Simulation Results

In addition to testing DailyTRIP in realistic environments, we have compared the performance of DailyTRIP against ILS through extensive simulation results. ILS serves as a benchmark against which to compare the results of DailyTRIP for the same topologies and TIDP parameter values. ILS algorithm and DailyTRIP algorithms differ substantially. DailyTRIP is based on trees for consolidating the POIs that belong to the same itinerary and then converts each 'itinerary tree' to a near-optimal route after running a TSP algorithm. In contrast, ILS uses a greedy approach [2], wherein the non-selected POI with the greatest score is chosen as the most suitable to be appended to the itinerary, thus removing it from the list of available candidate POIs. Using this approach, the ILS algorithm does not have to go through a large number of link selection options and permutations (i.e. it creates multi-point lines rather than trees, hence, it only examines candidate links deriving from the last inserted node), but it fails to examine likely more suitable candidate nodes from a global perspective. Thus, ILS represents a fair compromise in terms of speed versus deriving routes that approximate optimality. Another difference is that DailyTRIP considers node insertions interchangeably among constructed itineraries, while ILS first finalizes each itinerary before proceeding to the next one. In several topology scenarios, this may lead to 'greedily' incorporating some nodes to the currently examined itinerary, although those nodes could find a better match (i.e. yield improved overall profit) if connected to another itinerary.

Simulation results have been executed on a Java-based simulation tool, which takes into account most of the TIDP parameters discussed in Section IV and includes a visual interface for displaying the step-by-step output of the investigated algorithms. The simulator allows easy specification of simulation parameters and graphically illustrates the output of the DailyTRIP and ILS algorithm, while also recording their respective overall itinerary length, visit order of POIs and respective travel times. It also takes into account a number of constraints, for instance the attributes a_1, a_2, a_3 that determine the weight of travelling time, visiting time and profit of POIs in the calculation of the cost matrix (see Section IV.A), the number of days to visit, the minimum/maximum weight

to assign to each POI and the minimum/maximum time to visit a POI (stay time). The assignment of weight and stay time for each POI follows a uniform distribution. The network topology (i.e. POIs' coordinates) is randomly generated, given the dimensions of the city field, while the start/end point's position is explicitly set by the user.

Unless otherwise stated, in all our simulation tests, the weight of each POI is a random number between 20 and 100, the time a user can spend at a POI ranges from 5 to 120 minutes, while daily itineraries start at 09:00 and end at 18:00. In order to denote a high importance in regards to distance, the distance coefficient has been set to 20% (0,2), while the profit coefficient was set to 80% (0,8) in order to increase the importance of the profit in the POIs selection process. As a result, the visiting time coefficient was set to 0. The main reason for setting those values has been to come to agreement with the implementation of ILS algorithm, which does neither take into account the distance among POIs nor their corresponding visiting time (i.e. it only considers POIs' profit when designing the itineraries). The simulation results presented herein have been averaged over ten simulation runs (i.e. for ten different network topologies) to ensure statistical validity.



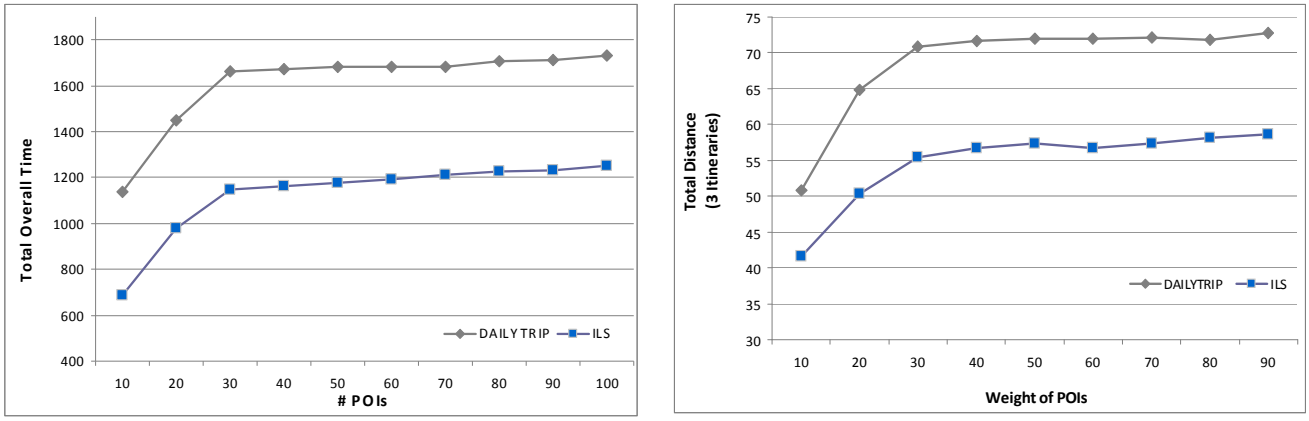
(a) (b)

Figure 5. (a) Comparison of the overall profit of DailyTRIP vs ILS for 2 itineraries; (b) comparison of the total travel time derived from DailyTRIP and ILS for 3 Itineraries.

Figure 5a illustrates the overall profit for the DailyTRIP in comparison to ILS as the number of POIs increase. For this simulation we assumed 2 days of stay (i.e. two itineraries). The overall profit of DailyTRIP is shown to improve the solutions obtained from ILS. This is basically due to the greedy nature of ILS. It should be stressed that the total itinerary profit of DailyTRIP very much depends on the value of profit coefficient, that is decreasing the profit coefficient results in decreased total profit (it signifies that the profit of POIs to visit becomes less important) and vice versa. Both algorithms' curves increase more mildly when crossing a threshold of around 20 POIs. This is because no more POIs may be accommodated within the 2 itineraries (typically a maximum of ~10 POIs can fit per itinerary). However larger topologies create opportunities to substitute some POIs with others having larger profit values, hence, larger topologies yield incremental overall profit values.

Figure 5b shows the total travel time for all 3 days (i.e. 3 itineraries) as a function of the increasing number of POIs. Clearly, as the number of POIs increase (i.e. itineraries are prolonged) so does the total travel time. The DailyTRIP yields improved results which are mainly due to the TSP algorithm executed on its last phase, which ensures near-optimality of each daily itinerary in terms of length. However, the total itinerary length of DailyTRIP very much depends on the value of distance coefficient, that is decreasing the travelling time coefficient resulting in increased total itinerary length (it signifies that the time spent by the user for travelling is less important) and vice versa. That represents an interesting trade-off among profit and travelling time. It also noted that after crossing a threshold number of POIs (around 30 POIs) small fluctuations are observed on both algorithms in the total travel time (as some POIs are substituted by others).

Figure 6a compares DailyTRIP against ILS in terms of their respective overall itinerary time. That sums up the overall time spent for travelling (i.e. moving from one POI to another) and visiting POIs. As expected, this figure resembles Figure 5b as the total visiting time is added to the total travelling time values. Figure 6b compares the total itineraries length of DailyTRIP and ILS (that is the sum of individual daily itinerary lengths). Evidently, the illustrated distance values are proportional to the travelling times of Figure 5b.

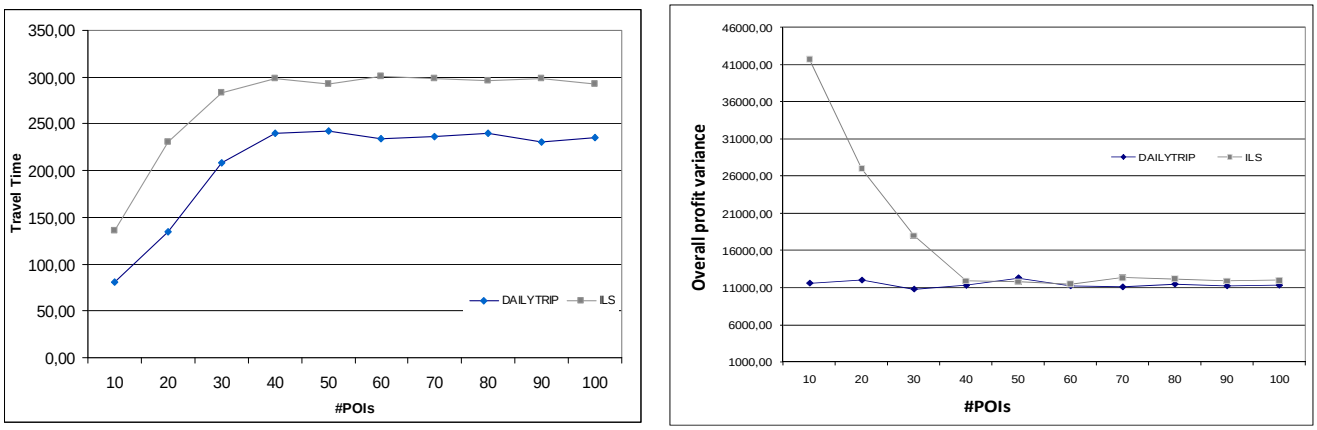


(a) (b)

Figure 6. (a) Comparison of the overall time derived from DailyTRIP and ILS algorithms for 3 itineraries; (b) comparison of the total distance for 3 itineraries.

Figure 7a compares the DailyTRIP against ILS with regards to the total travelling time, that is, the sum of individual daily itineraries travelling time. In this test, we consider 4 days of stay (i.e. 4 daily itineraries). DailyTRIP performs better than ILS as it yields reduced total travelling time. This is because ILS only considers POIs' profit as the only criteria for appending a new POI in an itinerary, whereas DailyTRIP considers both profit and topological distance (i.e. travelling time). Both algorithms are shown to increase their travelling time up to a threshold of ~40 POIs, which is the maximum number of POIs that can be accommodated within the 4 itineraries. Above this threshold, both DailyTRIP and ILS travelling time exhibit some fluctuations as the itineraries are modified (some POIs are substituted by others, hence, the itineraries' length is modified).

Last, Figure 7b portrays the variance among overall daily profits (i.e. the sum of profits of POIs visited on each day). Herein we assume 4 daily itineraries (i.e. staying 4 days at the destination). Small variance values denote fairly balanced itinerary profit values over the four itineraries while larger variance values denote that some itineraries are much more 'profitable' than others. Notably, DailyTRIP yields smaller variance values in comparison to ILS for small-scale topologies. This is because DailyTRIP considers the four itineraries interchangeably while inserting new nodes, hence, derived itineraries comprise approximately the same number of POIs (±1), ensuring a balanced overall itinerary profits. On the other hand, ILS first completes an itinerary before proceeding to the next one, which implies that in scenarios with few POIs and relatively large number of itineraries, ILS derives some 'full' and some 'empty' itineraries (hence, large variance values). In other words, DailyTRIP distributes visited POIs over all available days of stay, while ILS derives 'overloaded' days in the beginning and relatively 'relaxed' days in the end of the staying time. Notably, as the number of POIs increase, ILS obtains improved distributions of POIs into individual itineraries (i.e. its itineraries are gradually 'filled' with POIs) which in turn decreases variance values.



(a) (b)

Figure 7. (a) Total travel time using 4 days itinerary simulation; (b) variance of the individual itineraries' overall profit per day for both algorithms.

The satisfactory performance of DailyTRIP suggests it is suitable for online usage. In particular the algorithm requires less than 2 sec for topologies spanning up to 25 nodes, which represents a reasonable number of POIs to visit at any destination to derive a solution (excluding the time required to draw the solution on Google Maps) that deviates less than 7% from the optimal solution.

VII. CONCLUSIONS AND FUTURE RESEARCH

This paper introduced DailyTRIP, a heuristic approach for deriving personalized recommendations of daily tourist itineraries for tourists visiting any tourist destination. The algorithm targets both web and mobile users, with the latter offered location-aware recommendations. DailyTRIP considers selected POIs that a traveller would potentially like to visit and derives a near-optimal itinerary for the traveller for each day of visit. Our approach takes into account user preferences, time available for visiting sights in daily basis, opening days of sites and average visiting times for these sites. The objective of DailyTRIP is to maximize the overall profit associated with visited POIs (where individual profits are calculated as a function of the POIs' 'objective' importance and the user's potential interest for the POI) while not violating the daily time budget for sightseeing. Our algorithm has been implemented and proved suitable for online applications (i.e. real-time design of itineraries), while simulation results validated its performance gain over ILS algorithm.

Our future research will focus on variations of DailyTRIP algorithm that will incorporate additional TIDP problem parameters and constraints, e.g. weather conditions while on travel, financial budget (for transport and POIs admission charges), etc. We will also investigate the use of a combination of transport modalities, e.g. walking and bus service, taking into account various aspects of alternative transport services (e.g. walking time to the nearest metro station, time-dependent metro service frequencies, etc). Currently, we are working on an intelligent mechanism that will trigger automated itinerary updates when significant deviations from the original schedules are observed. Methods for fast, automated itinerary updates will also be considered, either for mobile users that have deviated from the suggested itinerary or due to sudden weather changes, unexpected public transportation schedule changes, etc. Last, the mobile application will incorporate various contextual parameters, such as time of day, predicted queuing delay and parking availability at each POI, etc.

REFERENCES

[1] Abbaspour R.A., Samadzadegan F. (2009). Itinerary Planning in multi-modal Urban Transportation Network. Journal of Applied Sciences, 9(10): 1898-1906.

[2] Black P.E. (2005). Greedy algorithm. Dictionary of Algorithms and Data Structures, U.S.National Institute of Standards and Technology, http://www.itl.nist.gov/div897/sqg/dads/HTML/greedyalgo.html. Last accessed: January 30, 2011.

[3] Chao, I.-M., Golden B. L., Wasil E.A. (1996). The Team Orienteering Problem. European Journal of Operational Research, 88(3): 475-489.

[4] Cheverst K., Davies N., Mitchell K., Friday A., Efstratiou C. (2000). Developing a Context-Aware Electronic Tourist Guide: Some Issues and Experiences. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: 17-24.

[5] Cheverst K., Mitchell K., Davies N. (2002). The Role of Adaptive Hypermedia in a Context-Aware Tourist GUIDE. Communications of the ACM 45(5): 47-51.

[6] City Trip Planner, http://www.citytripplanner.com/. Last accessed: January 30, 2011.

[7] Dumas Y., Desrosiers J., Gelinas E., Solomon M. M. (1995). An Optimal Algorithm for the Traveling Salesman Problem with Time Windows. Operational Research, 43(2): 367-371.

[8] Dunlop M., Ptasinski P., Morrison A., McCallum S., Risbey C., Stewart F. (2004). Design and Development of Taeneb City Guide - From Paper Maps and Guidebooks to Electronic Guides. Proceedings of the International Conference Information and Communication Technologies in Tourism (ENTER'2004): 58-64.

[9] Garcia A., Linaza M.T., Arbelaitz O., Vansteenwegen P. (2009). Intelligent Routing System for a Personalised Electronic Tourist Guide. Proceedings of the International Conference in Information and Communication Technologies in Tourism 2009 (ENTER'2009):185-197.

[10] Google city tours, http://citytours.googlelabs.com/. Last accessed: January 30, 2011.

[11] Google Maps. Google Maps API Concepts, http://code.google.com/apis/maps/documentation/. Last accessed: January 30, 2011.

[12] Google Static Maps API, http://code.google.com/apis/maps/documentation/staticmaps/. Last accessed: January 30, 2011.

[13] Hagen K., Kramer R., Hermkes M., Schumann B. Mueller P (2005). Semantic Matching and Heuristic Search for a Dynamic Tour Guide. Proceedings of the International Conference in Information and Communication Technologies in Tourism: 149-159.

[14] Helsgaun K, (2000). An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic, European Journal of Operational Research, 126(1): 106-130

[15] Kenteris M, Gavalas D, Economou D. (2009). An Innovative Mobile Electronic Tourist Guide Application. Personal and Ubiquitous Computing, 13(2):103-118.

[16] M. Kenteris, D. Gavalas, G. Pantziou and C. Konstantopoulos (2010). Near-Optimal Personalized Daily Itineraries for a Mobile Tourist Guide. Proceedings of the 15th IEEE Symposium on Computers and Communications (ISCC'2010): 862-864.

[17] Kramer R., Modsching M., ten Hagen K. (2006). A City Guide Agent Creating and Adapting Individual Sightseeing Tours Based on Field Trial Results. International Journal of Computational Intelligence Research 2(2): 191-206.

[18] Krüger A., Malaka R. (2004). Artificial Intelligence Goes Mobile. Applied Artificial Intelligence, 18: 469-476.

[19] Malaka R., Zipf A. (2000). DEEP MAP - Challenging IT Research in the Framework of a Tourist Information System., Proceedings of the International Conference on Information and Communication Technologies in Tourism (ENTER'2000).

[20] Maruyama A., Shibata N., Murata Y., Yasumoto K., Ito M. (2004). P-tour: A Personal Navigation System for Tourism. Proceedings of 11th World Congress on ITS:18-21.

[21] Souffriau W., Maervoet J., Vansteenwegen P., Vanden Berghe G., Van Oudheusden D. (2009). A Mobile Tourist Decision Support System for Small Footprint Devices, Proceedings of the 10th International Work-Conference on Artificial Neural Networks (IWANN'2009): 1248-1255.

[22] Souffriau W., Vansteenwegen P., Vertommen J., Berghe G.V., Oudheusden D. (2008). A Personalized Tourist Trip Design Algorithm for Mobile Tourist Guides. Applied Artificial Intelligence, 22(10): 964-985.

[23] Vansteenwegen P. (2008). Planning in tourism and public transportation. PhD thesis, Katholieke Universiteit Leuven.

[24] Vansteenwegen, P., Van Oudheusden D. (2007). The Mobile Tourist Guide: An OR Opportunity. Operational Research Insight, 20(3): 21-27.