Time-Dependent Tourist Tour Planning with Adjustable Profits

Felix Gündling
Technical University of Darmstadt, Germany
guendling@cs.tu-darmstadt.de

Tim Witzel
Technical University of Darmstadt, Germany
witzel@cs.tu-darmstadt.de

Abstract

Planning a tourist trip in a foreign city can be a complex undertaking: when selecting the attractions and choosing visit order and visit durations, opening hours as well as the public transit timetable need to be considered. Additionally, when planning trips for multiple days, it is desirable to avoid redundancy. Since the attractiveness of activities such as shopping or sightseeing depends on personal preferences, there is no one-size-fits-all solution to this problem. We propose several realistic extensions to the Time-Dependent Team Orienteering Problem with Time Windows (TDTOPTW) which are relevant in practice and present the first MILP representation of it. Furthermore, we propose a problem-specific preprocessing step which enables fast heuristic (iterated local search) and exact (mixed-integer linear programming) personalized trip-planning for tourists. Experimental results for the city of Berlin show that the approach is feasible in practice.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Mathematics of computing → Graph algorithms; Theory of computation → Randomized local search

Keywords and phrases tourist tour planning, orienteering problem, TDTOPTW, mixed integer linear programming, iterated local search, computational study

Digital Object Identifier 10.4230/OASIcs.ATMOS.2020.14

Supplementary Material Evaluation data provided at `https://github.com/motis-project/berlin-pois`

Funding This work was partially supported by Deutsche Bahn.

1 Introduction

When planning a tourist trip to a foreign city, there are often many activities to choose from. Selecting a subset of these, while keeping in mind their opening hours as well as the alternatives to get from one point of interest (PoI) to the next, can be a daunting and time-consuming task. Planning activities for multiple days (each day within a fixed time horizon) is even more challenging because one probably wants to avoid redundancy.

Opening hours may have potentially zero (closed) to multiple different time windows each day. While public statues and monuments can be visited anytime of the day, a special place to enjoy the sunset should be visited when the sun goes down. Public transport (containing regular as well as irregular services) is a popular option to move between PoIs. Thus, the problem definition has to be time-dependent. In addition to time-dependent means of transportation (public transit), many attractions are reachable by non-time-dependent means of transportation such as walking.

While some PoIs, like a statue, can be experienced within minutes, others (like a zoo or museum) can be entertaining for hours. Events like a theater or opera have a fixed start and end time. Modeling these properties requires a duration dependent profit function for each PoI. This profit function needs to be capable of enforcing a minimum required visit time

and be able to model a "saturation effect". It should not only take into account the type of PoI but also the personal preferences of the tourist: a family with children probably will not want to spent the same amount of time at an art museum as an elderly person.

To realistically model PoIs, it is important to consider multiple locations for entries and exits. The problem definition has to respect the time required to get from one entry/exit to another. For example, a large zoo, park, or shopping street can have various entries where each one can be reached with different public transport lines. Additionally, such areal PoIs may contain further PoIs (like statues or famous shops, bars, cafes).

In this paper, we propose a mathematical formulation of the aforementioned problem in the form of a mixed integer linear program (MILP). Furthermore, we present an iterated local search (ILS) approach to solve the problem fast enough for practical planning purposes (i.e. in a web-based or mobile planning service for tourists).

The remainder of this paper is organized as follows: Section 2 gives an overview over related work. Section 3 outlines our contribution to the topic of realistic tourist trip planning. In Section 4 we describe how we model the Time-Dependent Team Orienteering Problem with Time Windows (TDTOPTW) with our problem specific extensions as a Mixed Integer Linear Program (MILP). Section 5 contains a description of our approach to solve the problem. In Section 6, we present the results of our experimental study with data from the city of Berlin. Finally, Section 7 contains a conclusion and outlines ideas for future work.

2 Related Work

The (informal) problem description from Section 1 is close to the functionalities of the Next Generation Mobile Tourist Guide (MTG) envisioned in [34], and can be formally defined as a variation of the Orienteering Problem (OP) (also known as the selective traveling salesman problem [24]) which is proven to be NP-hard [19]. There has been extensive research regarding the OP and extensions thereof. In this section, we will discuss the general algorithmic research regarding the OP as well as the literature that specifically deals with tourist trip planning.

For a much more detailed overview of the state of the art, we refer to the mentioned survey papers [17, 21, 33] as well as the recent textbook [31]. The first computationally feasible mathematical formalization of the sport of orienteering [6] is given in [30]: participants have limited time to visit predefined checkpoints starting and finishing at a specific control point. Each checkpoint is associated with a score. The goal is to maximize the total score of all visited checkpoints. From this basic problem definition, several variations evolved. In the following, we will discuss those variants that are relevant for the problem introduced in Section 1.

Optimizing multiple tours (each limited in time) with the requirement that every checkpoint should still be visited only once is called the Team Orienteering Problem (TOP) which was introduced in [7]. The restriction that checkpoints may only be visited within specified time windows was first introduced in [5]. The multi-period OP with multiple (arbitrary) time windows is presented in [29]; [27] shows an extension with extra knapsack constraints. The combined problem is named (Team) Orienteering Problem with Time Windows (T)OPTW which is closely related to the Selective Vehicle Routing Problem with Time Windows (SVRPTW) [20]. The SVRPTW limits the vehicle capacity as well as the maximum distance traveled. The Time-Dependent Orienteering Problem (TDOP) is presented in [12]. The combination of the aforementioned problems is the Time Dependent Team Orienteering Problem with Time Windows (TDTOPTW) which was first presented in [14]. As some PoIs

require a specific continuous amount of time spent for the visit, this induces the OP with Variable Profits (OPVP) which is studied in [11] and applied in [35] for the city of Istanbul with 20 PoIs to maximize time at PoIs and minimize time spent to travel between PoIs. However, the other extensions (time dependency, "team" version, time windows) are missing here.

One of the practical applications of the OP besides vehicle routing is the Tourist Trip Design Problem (TTDP). The basic OP can be regarded as the most simplistic TTDP [33]. However, to model realistic tours, the variations described before are useful: the team version to compute multiple tours with non-overlapping sets of activities, time-dependency to support using public transport between points of interest, as well as time windows to consider opening times of attractions. An overview of the latest research regarding the TTDP can be found in [16, 21]. Most approaches used to solve realistic instances of the TTDP employ heuristic algorithms such as evolutionary genetic algorithms [1, 3] ([1] was evaluated with data of the city of Tehran; [3] was evaluated on 15 major cities in Iran – both employ a shortest path routing routine as subroutine of the tour optimization), iterated local search (ILS) [2, 13, 15, 32], or simulated annealing [25]. There are formulations in the form of a Mixed Integer Linear Program (MILP) of some variations of the OP (e.g. the TDOP in [21] and the OPVP [35]). The system proposed in [28] takes real-time information such as traffic and queue length at the attractions (manually provided by administrators of the system) into account.

3 Contribution

In this paper, we propose several realistic extensions to state-of-the-art variations of the orienteering problem. These extensions are specifically relevant to compute practical solutions when optimizing tourist trips. To the best of our knowledge, we present the first combination of the TDTOPTW and the OPVP with arbitrary time windows. The profit functions are personalized depending on the properties of each PoI as well as the preferences of the respective tourist. Additionally, our formulation of the problem supports multiple entries and exits for PoIs covering a widespread area. This is relevant in practice because especially for large PoIs like a zoo or a park, each entry/exit may be served by different public transport lines. Solutions computed by our approach respect the time required to walk from the entry to the exit of the PoI. Existing models associate each PoI with exactly one geographic coordinate which can lead to suboptimal routes in such cases.

We present the first Mixed Integer Linear Program (MILP) representation of the TD-TOPTW with the aforementioned extensions. Furthermore, we present a iterated local search (ILS) algorithm that solves the problem fast enough for practical purposes (e.g. as a backend for a web-based or mobile app service for tourists). Approaches based on ILS have been proven to be well suited to efficiently compute feasible solutions for the TDTOPTW and to produce high quality results [2, 18, 32]. In our evaluation on data for the city of Berlin with 41 diverse PoIs we compare the results of the ILS-based approach with the results of a MILP solver.

Visiting a park is worthwhile on its own. Therefore parks can be a PoI. However, parks may as well contain more PoIs (e.g statues). Thus, our model also supports PoIs in parks or other areal PoIs.

4 Modeling the Problem

4.1 Profit Function for Points of Interest

In this section, we define a generalized profit function which takes the visit time as input and returns the profit gained when visiting the PoI for this amount of time. As mentioned in Section 1, there are many different profit functions. Most PoIs require a certain amount of time to achieve any profit. Then, the accumulation of profit will flatten out and finally staying longer at a PoI will not yield any further profit. To model this behavior, we introduce a piecewise linear function

$$p(t) = \begin{cases} 0 & t < t_{\text{minvisit}} \\ p_{\min} + (t - t_{\text{minvisit}}) \cdot \text{ppt} & t_{\text{minvisit}} \leq t \leq t_{\text{maxvisit}} \\ p_{\max} & t_{\text{maxvisit}} < t \end{cases}$$

where:

- t_{minvisit} is the minimum time a tourist needs to visit a PoI before profit can be gained
- t_{maxvisit} is the maximum amount of time. Staying longer should not accumulate any further profit.
- $p_{\min}$ is the minimum profit a tourist gains when staying at least t_{minvisit} at the PoI
- $p_{\max}$ is the maximum profit a tourist can gain by visiting the PoI
- *ppt* is the profit per time unit gained at the PoI after the minimum visit time is exceeded. It can be calculated with the two points $(t_{\text{minvisit}}, p_{\min})$ and $(t_{\text{maxvisit}}, p_{\max})$.

A movie theater would have $p_{\min}$ set equal to $p_{\max}$ to ensure the visit does not last shorter but at the same time also not longer as the movie duration (plus some time buffer). As the attractiveness of a PoI depends on the preferences of the user, we multiply the profit values $p_{\min}$, $p_{\max}$ with the preference value of the user for this PoI. For each category (e.g. "shopping", "museum" or "public monument"), the user rates their interest on a scale from 0 (not interested) to 10 (highly interested). Each PoI is tagged by at least one category. The preference value of the user for each PoI is then calculated by dividing the sum of the preference values given by the user for each category of the PoI by the total number of categories of the PoI (as a normalization to not give weight to PoIs with multiple categories).

4.2 Mixed Integer Linear Program

In our definition of the MILP, we make use of the following notation. Inputs are noted as capital letters, output variables are lower case. A "location" is used synonymously to entry/exit of a PoI and is therefore associated with exactly one PoI. ℓ is used as an arbitrary large number. As input variables we use:

P	set of all PoIs
M	set of all tours
N_m	set of all locations for tour m
Z_{ip}	1 if location i belongs to PoI p, otherwise 0
T_m	set of all discrete timeslots for tour m
T_{ij}^{walk}	walking duration from location i to j
T_{ijt}^{travel}	travel time from location i to j departing in timeslot t
T_t^{slot}	starting time of timeslot t

T_m^{start}	starting time for tour m
$T_m^{\max}$	time limit for tour m
$F_i(x)$	profit function for location i
$P_i^{\max}$	maximum profit at location i
$P_i^{\min_t}$	minimum visit time at location i
W_{im}	set of time windows for location i and tour m
O/C_{iwm}	opening/closing time for location i, tour m, time window w

As output variables we use:

p_{im}	profit accumulated at location i in tour m
y_{ijmt}	location i is left to location j in timeslot t for tour m
x_{pm}	number of visits to PoI p in tour m
t_{im}^{poi}	time spent visiting location i in tour m
s_{im}	arrival time at location i in tour m
j_{im}	helper variable for our piecewise linear profit function
g_{iwm}	helper variable for modeling multiple time windows

We define the objective function as: $\max \sum_{m \in M} \sum_{i \in N_m} p_{im}$ which sums up the profit gained over all PoIs in all tours. The profit is 0 if the PoI is not visited on the tour. Otherwise, the gained profit is the value of the profit function defined in Section 4.1. Locations 1 and N are the starting and ending point provided by the user. These may differ for each tour. All other locations are fixed. Constraint 1 ensures that the first location is left exactly once and the last location is reached exactly once:

$$\sum_{j \in N_m} \sum_{t \in T_m} y_{1jmt} = \sum_{i \in N_m} \sum_{t \in T_m} y_{iNmt} = 1 : \forall m \tag{1}$$

To ensure that every PoI is entered at most once and left the same number of times (i.e. once or not at all), we introduce the following constraints ($y_{ijmt} \cdot Z_{jp}$ connects locations used in y with their PoIs in p):

$$\sum_{i \in N_m} \sum_{j \in N_m} \sum_{t \in T_m} y_{ijmt} \cdot Z_{jp} = \sum_{i \in N_m} \sum_{j \in N_m} \sum_{t \in T_m} y_{ijmt} \cdot Z_{ip} = x_{pm} : \forall m, \forall p \tag{2}$$

$$\sum_{m=1}^{M} x_{pm} \leq 1 : \forall p, \tag{3}$$

As we allow for entering and leaving PoIs at different locations, we introduce the following constraint to ensure that the minimum walking time between the two locations of the PoI is taken into account. The product of $Z_{kp} \cdot Z_{ip} \cdot T_{ik}^{\text{walk}}$ yields 1 only for locations of the same PoI.

$$(1 - \sum_{l \in N_m} \sum_{t \in T_m} y_{klmt} - \sum_{x \in N_m} \sum_{t \in T_m} y_{ximt}) \cdot Z_{kp} \cdot Z_{ip} \cdot T_{ik}^{walk} \leq t_{i,m}^{poi} \forall i, k, p, m \tag{4}$$

The following constraint ensures that each tour duration is limited to $T_m^{\max}$ given by the user. The duration of each tour is the sum of the time spent at PoIs and the time required

to travel between PoIs. The time at the last location of the tour (e.g. the hotel) should be smaller than the sum of the start time and the maximum travel time.

$$s_{Nm} \leq T_m^{\text{start}} + T_m^{\max} \tag{5}$$

The following constraints are needed to ensure that the correct departure time is used at each PoI - i.e. the PoI is left after the visit is finished (and not before). Both constraints are automatically fulfilled by the right hand side (given any big number M), if the corresponding y variable is 0 or in case of Constraint 6 if locations k and i do not belong to the same PoI.

$$s_{im} + t_{im}^{\text{poi}} \leq T_t^{\text{slot}} + \ell(1 - y_{kjmt} \cdot Z_{ip} \cdot Z_{kp}) : \forall i, j, k, p, m, t \tag{6}$$

$$T_t^{slot} + T_{ijt}^{\text{travel}} \leq s_{jm} + \ell(1 - y_{ijmt}) : \forall i, j, m, t \tag{7}$$

To ensure that the time-dependent travel times between PoIs are respected, the following constraint is required. This constraint is automatically fulfilled by the right hand side (given any big number ℓ), if either k is never left towards j or if k and i are not locations of the same PoI.

$$s_{im} + t_{im}^{\text{poi}} + T_{kjt}^{\text{travel}} - s_{jm} \leq \ell(1 - y_{kjmt} \cdot Z_{ip} \cdot Z_{kp}) : \forall i, j, k, p, m, t \tag{8}$$

To model the profit function for each location, we use the following constraints. We introduce j as helper variable which (due to Constraint 11) is 1 *iff* the visiting time is less than the minimum visit duration. Therefore, Constraint 13 forces the gained profit to be 0 if this is the case.

$$p_{im} \leq F_i(t_{im}^{\text{poi}}) + \ell j_{im} \qquad \forall i, m \tag{9}$$

$$p_{im} \leq P_i^{\max} x_{im} \qquad \forall i, m \tag{10}$$

$$t_{im}^{poi} + \ell j_{im} \geq P_i^{\min_t} \qquad \forall i, m \tag{11}$$

$$p_{im} \geq 0 \qquad \forall i, m \tag{12}$$

$$p_{im} \leq \ell - \ell j_{im} \qquad \forall i, m \tag{13}$$

To model multiple time windows per PoI/location and ensure that every visit of a PoI takes place within a time window of this respective PoI, we introduce the following constraints. Constraint 14 ensures, that a visit s_{im} at location i in tour m starts after an opening time O_{iwm} (w is the index of the specific time window) whereas Constraint 15 does this analogously for closing times. Constraint 16 ensures only opening and closing times of the same time window are matched by introducing variable g_{iwm}. Thus, either index i in O_{iwm} and C_{iwm} matches or Constraints 14 and 15 are always true

$$O_{iwm} \cdot g_{iwm} \leq s_{im} \qquad \forall i, w, m \tag{14}$$

$$s_{im} \leq C_{iwm} + \ell(1 - g_{iwm}) \qquad \forall i, w, m \tag{15}$$

$$\sum_{w \in W_{im}} g_{iwm} \leq 1 \qquad \forall i, m \tag{16}$$

Finally, we set the starting location and ensure positive visit times through the following constraints:

$$s_{1m} = T_m^{start} \tag{17}$$

$$t_{im}^{poi} \geq 0 \quad \forall i, j, m, t \tag{18}$$

This form is suitable for MILP solvers and was programmed in Gurobi [23] for the experimental study of this paper presented in Section 6.

5 Approach

In this section, we will describe the preprocessing required to deliver real-time response times for user queries as well as different heuristic approaches to solve the problem defined mathematically in Section 4.2.

5.1 Preprocessing

Like in [14], we use an offline preprocessing step which does not have real-time requirements. We precompute intermodal time-dependent shortest paths for public transport and walking connections from every location to every other location for every timeslot. Our routing is based on [22][1] and respects realistic transfer times, allows for walking between stations, and does not assume periodicity of the timetable. We use a realistic pedestrian routing based on OpenStreetMap data for the path between the PoI location and the next public transport stop. Therefore, we precompute shortest paths from every PoI location to every public transport stop and vice versa.

The fact that most shortest path algorithms for public transit routing (like RAPTOR [8], Connection Scanning [9], and all graph-based Dijkstra variations like [10]) are inherently computing shortest paths to all targets at the same time, can be exploited here. Here, each shortest path problem is independent from every other shortest path problem. Thus, this task is perfectly suited to be carried out in parallel.

The result is a time-expanded directed acyclic event-activity graph where every node is either an arrival or a departure at a PoI location (in a discrete timeslot). Arrivals and departures at the same location are connected by *visit* edges. To model entering and leaving a PoI at different entries/exits, arrivals and departures at different locations of the same PoI are connected by *intra* edges. *Intra* edges and *inter* edges need to conform to the computed walking durations and travel times respectively. *Visit* edges and *intra* edges are only created for visit times greater or equal the minimum visit time specified for the respective PoI profit function. Finally, *inter* edges connect departure and arrival nodes of different PoIs.

Start and end location are both provided by the user. Thus, these can either be limited to a set of known hotels and public transport stations where tourists typically start their trip, which allows us to include them in the preprocessing. If this is not an option, four additional queries need to be carried out online at query time: from the start location to every location, using the start time as earliest departure time (forward search one to all) and *to* the last location from every location, taking the start time plus the maximum trip time as latest arrival time (backward search all to one). These two queries need to be done for the pedestrian routing (to all locations and public transport stops) as well as the public transport routing (initiating the labels with the results from the pedestrian routing). Both, forward and backward direction can be computed in parallel. These nodes and edges are added to the event-activity graph.

5.2 Heuristic Algorithm

To supplement computing solutions to problem using a MILP solver (which is time-consuming as discussed in the Section 6), we develop heuristic algorithms: as a baseline, we present a Basic Greedy Algorithm (BGA) and an Advanced Greedy Algorithm (AGA). As outlined in Section 2, Iterated Local Search (ILS) based approaches were able to compute high-quality

[1] The latest version is available as Open Source Software at `https://motis-project.de/`

results in real-time. Therefore, we decided to implement an ILS approach to solve the problem at hand on the graph described in Section 5.1. We present our Basic Iterated Local Search (BILS) as well as our Specialized Iterated Local Search (SILS).

5.3 Basic Greedy Algorithm

This algorithm solves the problem sequentially, building a tour from start to end by adding one greedily chosen activity at a time. For each expansion, the BGA has to choose the next PoI, a visit time, and a location to exit the PoI at. This can be done efficiently on the event-activity graph described in Section 5.1. To compute valid tours where no PoI is visited twice, the algorithm has to keep a set of already visited PoIs and prevent expansions which would result in revisiting PoIs which were already visited. This set is kept for all tours that will be planned. Additionally, the algorithm requires a "dead-end protection": there are PoIs from where it is not possible to reach the end location within the maximum tour duration. To prevent this, the graph can be pruned by removing nodes that have no transitive path to the end location. A backward BFS starting from the end location nodes can mark all feasible nodes. Nodes not marked within this run are omitted in the expansion step of the BGA. Another solution would have been to introduce a backtracking step if the algorithm visited a dead-end. The BGA chooses the next PoI based on a weight function $p/(w \cdot T_{\text{travel}} + T_{\text{visit}})$ where w controls the influence of the travel time. Note that this algorithm greedily selects only steps that look locally promising. However, a globally optimal solution may contain steps which will not be chosen with any w value.

5.4 Advanced Greedy Algorithm

To improve upon the BGA, the AGA also makes (locally) suboptimal steps and keeps a list of multiple active solutions. The basic properties of the BGA (duplicate PoI prevention and dead-end protection) stay the same. However, it makes multiple expansions in each step - each one with a different value for w. After each complete step, it cuts off all solutions with a lower profit per time duration than the best solution times the cutoff threshold. A high cutoff threshold implies many cut off paths and therefore a better computing time but also a decreased chance to find better solutions (i.e. paths from the start location to the end location in the event-activity DAG). A cutoff threshold of zero combined with a unlimited list of active solutions would result in listing all feasible solutions. This would yield the optimal solution but is not feasible in practice for realistic problem instances.

5.5 Basic Iterated Local Search

A ILS basically uses a Local Search to find a local optimum. After that, the local optimum is perturbed sufficiently enough to be able to escape the previous local optimum and find a local optimum. The algorithm terminates if the Local Search cannot find a new local optimum after a certain number of perturbations.

In our case, the search can be either seeded with an empty route (respectively multiple empty routes if we are planning more than one tour) from start to finish (without visiting PoIs) or the result of one of the previously described greedy algorithms. We define our neighborhood for the local search step as all solutions which can be produced by integrating a visit to a new PoI while still keeping the solution feasible. All existing visits keep their arrival and departure time. We decide to insert always the (locally) best PoI visit (i.e. the PoI which has the best profit per time including travel time) using the maximum visit time.

This will be done until it is not possible to add yet another PoI. The perturb step removes a varying number of PoI visits from the current solution. The remaining PoIs are then shifted forward in time (i.e. towards the start of the route) as much as possible. The number of removed PoI visits is incremented (to improve the chance to find a new local optimum) if the new solution is equal or worse (regarding the profit value) than the previous solution.

5.6 Advanced Iterated Local Search

Since the previously presented heuristic algorithms always select the maximum visit time (by locally optimizing the profit value), it would be interesting to introduce options to lengthen (in the local search) or shorten (in the perturb step) a visit at a PoI. We add options to extend the visit of a PoI to the Local Search neighborhood. This is done by moving the arrival time to an earlier point in time or by moving the departure time to a later point in time. The visit time is extended by 5 minutes in each step. Since the extension step is called repeatedly, the algorithm should eventually be able to find new optima. The perturbation step is now capable of shortening all PoIs from the front or back. As previously noted, only feasible solutions are allowed as Local Search and perturbation step result. Still, the best neighbor is chosen and the Local Search step continues until the neighborhood does not contain any improvement.

6 Experimental Results

In this section, we will present the results of our experiments. As MILP solver, Gurobi [23] was used and executed on a computer with an Intel® Xeon® CPU E3-1245 V2 processor (3.4GHz) and 32 GB of RAM. Everything else was run on a computer with an Intel® Core® M i3-5005U processor and 8 GB of RAM. The greedy and ILS algorithms are implemented in C++.

The test instance are 41 hand-picked PoIs in Berlin from various categories with manually researched opening and closing times [2]. The main categories were defined as “Museum”, “Monument”, “Panorama”, and “Experience”. More details can be derived from the theme category “Art”, “Nature”, “History”, “Famous”, and “Shopping”. Each PoI can have multiple categories. It is also possible for a tourist to set a high preference value for only a single category - e.g. if they are interested in a tour of famous landmarks of the city of Berlin (e.g. the Brandenburger Tor, pieces of the Berlin Wall, etc.). This could also be used to generate interesting ideas for theme tours for so called “Hop-On Hop-Off” buses (albeit with a street routing algorithm to generate the event activity DAG).

We manually picked 25 different queries covering a diverse set of combinations of maximum duration (between 2-10 hours), number of tours (one or two) with four possible start and end locations. We chose to evaluate the algorithms with a balanced profile as a single high preference value for one category eliminates all but a few PoIs which produces unvaried tours and makes the problem much easier to solve. This would not make for a good benchmark.

The timetable for the city of Berlin was kindly provided by Deutsche Bahn for research purposes.

Table 1 Preprocessing Computation Times.

Preprocessing Step	Computing Time
Data Initialization	46 ms
Calculating Walk Times between PoIs	11.5 min
Calculating Travel Times between PoIs	2.5 min
Building Event-Activity DAG	6,6 s
Precomputing Paths for Query Positions	3.5 min
Integrate Query Data	12 ms
Total	15 min 23 s

Table 2 Graph Information for Different Granularity Settings.

	Granularity 1 min	Granularity 5 min	Granularity 10 min
Arrival Nodes	81,452	67,963	58,815
Departure Nodes	88,470	17,694	8847
Inter Edges	9,382,766	1,877,691	939,597
Visit Edges	5,130,816	929,892	438,198
Intra Edges	4,044,902	727,959	329,015

Preprocessing

In Table 1, we report the time it takes to finish the preprocessing step described in Section 5.1. The total duration (approximately 15min) is in a range where the preprocessing can even be repeated with minimal effort when the timetable or pedestrian routes change.

Table 2 shows the size of the event-activity DAG for different granularities of the timeslots.

MILP Solver

We ran every query with the MILP solver for 10 hours each. The solver was seeded with the best greedy solution. For the simplest query (a two hour tour), the solver did find an optimal solution. For all other queries, the solver provided the best solution known so far as well as an upper bound for the profit of the best solution possible. The difference between the upper bound and the currently known best solution ranges between 6% and 20%.

Greedy Algorithm

The BGA from Section 5.3 was evaluated using 13 different travel time weights (0-10, 15, and 20). In general, extreme travel time weights such as 0, 15, and 20 performed badly as it is not reasonable to chose only very close PoIs or only high profit PoIs (ignoring travel time completely). For long tours, lower travel time values seem to outperform higher values whereas for short tours, the opposite is the case. This makes sense because for short tours, long travel times leave not much time for the actual visits. Therefore, it could be useful to select the travel time weight depending on the tour length. The BGA from Section 5.3 takes about 2-4ms to complete. Comparing the result with those from the MILP solver, we see that the MILP solver consistently outperforms the BGA by 5-10 pp. For one query, the gap is even 16 pp. The gap is especially high for queries with long maximum travel times or even multiple tours because the solution space increases drastically.

[2] The data is freely available at `https://github.com/motis-project/berlin-pois`.

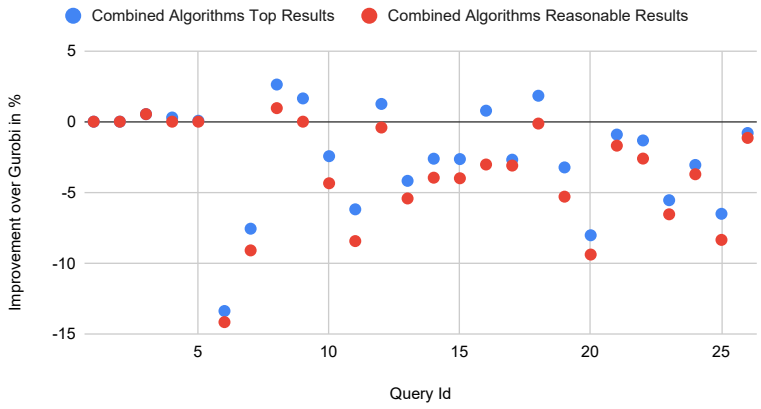

Figure 1 Improvement over Gurobi Results for the Combined Algorithms: the highest profit solution in blue and the highest solution which can be computed in real-time in red.

The AGA from Section 5.4 was tested with different numbers of active solutions (100, 1.000, 10.000), different cutoff thresholds (0.25 and 0.5) as well as different numbers of chosen candidates (1, 3, 5). This yields 18 variations which we supplemented with one further combination: 100.000 active solutions, cut-off threshold 0.5 and 3 chosen candidates. The best solutions were found with the latter parameterization (15 times), closely followed by 10.000/0.25/5 (active solutions / threshold / chosen candidates). The best configuration with 1.000 active solutions was 1.000/0.25/5 which produced the best known solution in 12 cases. The main driving factor for the processing time is the number of active solutions: the AGA takes around 1 second for 100 solutions, 10 seconds for 1.000 solutions, 50 seconds for 10.000 solutions, and 500 seconds for 100.000 solutions. The other parameters do not influence the processing time significantly. Comparing the AGA with the MILP solver, the solver still outperforms the result quality of the greedy algorithm by a huge margin of up to 15 pp. Interestingly, for five queries, the AGA was able to compute slightly better solutions (2-3 pp) than the MILP solver (which was halted after 10 hours).

Iterated Local Search

The BILS presented in Section 5.5 was not able to improve upon the seeds from the best greedy algorithm except in one case, where the profit was marginally improved (from 1231 to 1235 profit). As the best greedy algorithms are also very slow, we differentiate more between the different greedy algorithm parameterizations and observe that the BILS is quite capable of improving upon bad seeds from extreme travel time weights (0, 15, and 20) for the BGA. Although the improvement upon the highest quality seeds is marginal, the BILS is nonetheless interesting due to its fast computation times averaging around 1.5 seconds.

The AILS described in Section 5.6 was not able to improve upon the previously known best solutions of our heuristic algorithms except for a slight improvement for one query (1402 to 1403). Compared with the basic ILS, the query runtime of 5 seconds on average does not yield a worthwhile benefit.

Overall Comparison

Figure 1 shows an overall comparison of the heuristic algorithms with the best solution found by the MILP solver after 10 hours. For simple queries (1-5), the solutions do not differ much. However, there are queries where the best solution found by a heuristic algorithm is not even close (more than 10 pp difference) to the solution found by the solver. Interestingly, in some cases, the heuristic algorithms were able to find slightly better solutions (2-3 pp) in some cases.

Granularity Analysis

Previous instances were reported with a 5 minute granularity for timeslots. Now, we also vary the granularity and test the values 1 minute, 5 minutes, and 10 minutes. The results show a strong correlation between query runtime and granularity: computing results with a one minute granularity takes about 5 times as long as for the 5 minute granularity while at the same time, the 10 minute granularity made the processing about twice as fast as the 5 minute granularity. The profits for the 1 minute granularity only improve between 0.5 pp to 1 pp (depending on the query) compared to the 5 minute granularity. However, the increase of profit value from the 10 minute granularity compared to the 5 minute granularity ranges from 0.5 pp to 5 pp. All in all, the 5 minute granularity seems to be a good trade-off between result quality and processing time.

7 Conclusion and Future Work

In this paper, we presented several realistic extensions to the previously known definition of the TDTOPTW (a variation of the TTDP) to make tourist trip planning more feasible in practice and combine the TDTOPTW with the OPVP to account for variable personalized PoI profit functions. For instance, the problem definition presented in this paper supports multiple entries and exits for each PoI. We presented the first MILP modeling of the TDTOPTW including the described realistic extensions. The approach is split into two phases: the preprocessing phase has no real-time requirements and computes a time-expanded event-activity DAG by routing optimal public transport and walking connections from every PoI entry/exit to every other PoI entry/exit at every time with different granularity (here, we used 1, 5, and 10 minutes). This allows for efficient trip planning at query time and eliminates the need for repair steps as required by most previous approaches.

As the MILP solver takes quite long with the current definition, an interesting research direction would be to search for ways to improve the representation in order to solve the problem online in real-time.

In the future, the system could be extended to support adaptions of the profit functions of PoIs depending on the weather forecast (i.e. prefer indoor activities for rainy days). Additionally, the tour can be split further into smaller parts to allow for lunch and/or dinner. Note that both of these extensions neither require any adjustment of the MILP nor any changes to the ILS algorithm but can be encoded into the input. Furthermore, the algorithm described and implemented in this paper could be used as a backend service for interfaces presented in [4]. Combining our approach with [26] to guess preferences based on previous ratings and activities can make for an even more satisfying tourist experience.

References

1 Rahim A Abbaspour and Farhad Samadzadegan. Time-dependent personal tour planning and scheduling in metropolises. *Expert Systems with Applications*, 38(10):12439–12452, 2011.

2 Victor Anthony Arrascue Ayala, Kemal Cagin Gülsen, Marco Muñiz, Anas Alzogbi, Michael Färber, and Georg Lausen. A delay-robust touristic plan recommendation using real-world public transportation information. In *CEUR Workshop Proceedings*, volume 1906, 2017.

3 Ali Azizi, Farid Karimipour, and Ali Esmaeily. Time-dependent, activity-based itinerary personal tour planning in multimodal transportation networks. *Annals of GIS*, 23(1):27–39, 2017.

4 Joan Borràs, Antonio Moreno, and Aida Valls. Intelligent tourism recommender systems: A survey. *Expert Systems with Applications*, 41(16):7370–7389, 2014.

5 Sylvain Boussier, Dominique Feillet, and Michel Gendreau. An exact algorithm for team orienteering problems. *4or*, 5(3):211–230, 2007.

6 I-Ming Chao, Bruce L Golden, and Edward A Wasil. A fast and effective heuristic for the orienteering problem. *European journal of operational research*, 88(3):475–489, 1996.

7 I-Ming Chao, Bruce L Golden, and Edward A Wasil. The team orienteering problem. *European journal of operational research*, 88(3):464–474, 1996.

8 Daniel Delling, Thomas Pajor, and Renato F Werneck. Round-based public transit routing. *Transportation Science*, 49(3):591–604, 2015.

9 Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. Intriguingly simple and fast transit routing. In *International Symposium on Experimental Algorithms*, pages 43–54. Springer, 2013.

10 Yann Disser, Matthias Müller-Hannemann, and Mathias Schnee. Multi-criteria shortest paths in time-dependent train networks. In *International Workshop on Experimental and Efficient Algorithms*, pages 347–361. Springer, 2008.

11 Güneş Erdoğan and Gilbert Laporte. The orienteering problem with variable profits. *Networks*, 61(2):104–116, 2013.

12 Fedor V Fomin and Andrzej Lingas. Approximation algorithms for time-dependent orienteering. *Information Processing Letters*, 83(2):57–62, 2002.

13 Ander Garcia, Olatz Arbelaitz, Maria Teresa Linaza, Pieter Vansteenwegen, and Wouter Souffriau. Personalized tourist route generation. In *International Conference on Web Engineering*, pages 486–497. Springer, 2010.

14 Ander Garcia, Pieter Vansteenwegen, Olatz Arbelaitz, Wouter Souffriau, and Maria Teresa Linaza. Integrating public transportation in personalised electronic tourist guides. *Computers & Operations Research*, 40(3):758–774, 2013.

15 Damianos Gavalas, Vlasios Kasapakis, Charalampos Konstantopoulos, Grammati Pantziou, Nikolaos Vathis, and Christos Zaroliagis. The ecompass multimodal tourist tour planner. *Expert systems with Applications*, 42(21):7303–7316, 2015.

16 Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. Mobile recommender systems in tourism. *Journal of network and computer applications*, 39:319–333, 2014.

17 Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, 20(3):291–328, 2014.

18 Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou, and Nikolaos Vathis. Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers & Operations Research*, 62:36–50, 2015.

19 Bruce L Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318, 1987.

20 Cyrille Gueguen. *Méthodes de résolution exacte pour les problèmes de tournées de véhicules*. PhD thesis, Châtenay-Malabry, Ecole centrale de Paris, 1999.

21 Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

22 Felix Gündling, Mohammad Hossein Keyhani, Mathias Schnee, and Karsten Weihe. Fully realistic multi-criteria multi-modal routing, December 2014. URL: `http://tuprints.ulb.tu-darmstadt.de/4298/`.

23 LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020. URL: `http://www.gurobi.com`.

24 Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Discrete applied mathematics*, 26(2-3):193–207, 1990.

25 Shih-Wei Lin and F Yu Vincent. A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 217(1):94–107, 2012.

26 Antonio Moreno, Aida Valls, David Isern, Lucas Marin, and Joan Borràs. Sigtur/e-destination: ontology-based personalized recommendation of tourism and leisure activities. *Engineering applications of artificial intelligence*, 26(1):633–651, 2013.

27 Wouter Souffriau, Pieter Vansteenwegen, Greet Vanden Berghe, and Dirk Van Oudheusden. The multiconstraint team orienteering problem with multiple time windows. *Transportation Science*, 47(1):53–63, 2013.

28 V Subramaniyaswamy, R Logesh, V Vijayakumar, K Keerthana, K Rakini, and B Swarnamalya. Dynamo: Dynamic multimodal route and travel recommendation system. In *2018 International Conference on Recent Trends in Advance Computing (ICRTAC)*, pages 47–52. IEEE, 2018.

29 Fabien Tricoire, Martin Romauch, Karl F Doerner, and Richard F Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37(2):351–367, 2010.

30 Theodore Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9):797–809, 1984.

31 Pieter Vansteenwegen and Aldy Gunawan. *Orienteering problems: Models and algorithms for vehicle routing problems with profits*. Springer Nature, 2019.

32 Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290, 2009.

33 Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

34 Pieter Vansteenwegen and Dirk Van Oudheusden. The mobile tourist guide: an or opportunity. *OR insight*, 20(3):21–27, 2007.

35 Jingjin Yu, Javed Aslam, Sertac Karaman, and Daniela Rus. Optimal tourist problem and anytime planning of trip itineraries. *arXiv preprint arXiv:1409.8536*, 2014.