Flexible Multicriterial Agenda Planning in Public Transit Systems
An Intelligent Agent for Mobility-oriented Agenda Planning

Felix Schwinger[1], Fabian Ohler[1] and Karl-Heinz Krempels[1,2]
[1]*Fraunhofer Institute for Applied Information Technology FIT, Aachen, Germany*
[2]*Information Systems, RWTH Aachen University, Aachen, Germany*

Keywords: Agenda Planning, Information System, Intelligent Agent, Mobility Planning, Operations Research, Optimization Problem, Orienteering Problem, Public Transportation, Travel Planning.

Abstract: Planning a mobility-oriented agenda is a time-consuming and tedious task for many travelers. A person is required to collect information from different sources such as a map service, a business register, a calendar and a journey planner. However, she is mostly not interested in either planning the agenda or the journeys between different locations of the agenda; but is more interested in completing the tasks of the agenda. Therefore, we propose an intelligent agenda planning agent that aims to support people with this task. We integrate public transit schedules with additional spatial information from OpenStreetMap to create an information database for the agent. The agent can then plan tasks and appointments and the mobility between those items. First brief evaluations with a survey have shown, that the algorithm finds shorter agendas than most manually found agendas. However, participants of the survey criticized the temporal placement of tasks in the agenda.

1 INTRODUCTION

Urban transportation systems are often large, complicated to use for the first time and when regarding multimodal transportation systems, also difficult to utilize effectively (Zografos and Androutsopoulos, 2008). In recent years, there has been a surge of new information systems that provide useful information for passengers of public transit systems. The developed information resources include map services, business directories, and online route planners which try to decrease the usage complexity of urban transportation systems. A user can resort to these services when planning her agenda; however, the main problem with these services is that they do not provide an easy to understand, integrated solution to the whole problem of mobility agenda planning, and only offer segmented information (Baena-Toquero et al., 2014). A mobility agenda does not only include the agenda items, i. e., appointments and tasks, but also the mobility between the different locations of the agenda. The user, therefore, needs access to multiple services when planning her agenda: She could check nearby locations on a map service, then inquire their opening times in a business directory and finally plan the journeys between the locations with a route planner. Additionally, she might check different journey planners for different types of mobility, i. e., car-sharing, walking or public transportation. A user might, however, not even be interested in manually planning her agenda, but is more interested in an agenda that allows her to perform all tasks she wants to complete on a day. An integrated intelligent system that allows the user to specify her needs, which then calculates an optimal mobility agenda that fits the passenger's schedule would help to drastically reduce the complexity. This paper introduces an algorithm for computing intra-city itineraries that focuses on public transportation systems.

The remainder of this paper is structured as follows: In Section 2, we propose the *Mobility Agenda Planning Problem* and in Section 3, the related work is introduced and discussed. Section 4 introduces a model for mobility-oriented agenda planning and Section 5 describes an algorithm for it. We show first brief results of a performed survey in Section 6 and finally conclude the paper with an outlook and a summary in Section 7.

2 PROBLEM DESCRIPTION

In this paper, we regard the *Mobility Agenda Planning Problem (MAPP)* as follows: The input for the prob-

Schwinger, F., Ohler, F. and Krempels, K.
Flexible Multicriterial Agenda Planning in Public Transit Systems.
DOI: 10.5220/0006795305580567
In *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2018)*, pages 558-567
ISBN: 978-989-758-293-6

lem is a number of appointments and tasks. Appointments are entirely fixed, whereas tasks are movable in at least one dimension, i. e., temporally or spatially. Each insertable agenda item has an associated score and a completion time. The score of the agenda item may represent the priority of the task to the person, or a combined score of various factors. The goal of a MAPP algorithm is to find an optimized mobility agenda, containing all appointments and a number of tasks. Additionally, the algorithm computes the mobility between all locations in the agenda. In this paper, we focus on walking and public transportation journeys between agenda items. The task for the problem is then to optimize the mobility agenda by maximizing the score of the agenda while minimizing the total time it takes to complete the agenda. The total time includes possible wait times at public transit stations, wait times at *Points of Interest (POIs)*, travel times between different locations, and the completion time for agenda items. All appointments have to be reached on time, and all public transit transfers must be feasible. Furthermore, the algorithm has to consider category or location constraints for agenda items and potential dependencies between agenda items. Each location, therefore, also has an associated category, so that the agent can determine suitable locations for tasks. It should also be possible to define dependencies between agenda items so that one agenda item can only be inserted into the agenda, if another agenda item has been completed before it. To compute mobility-oriented agendas, the algorithm needs to optimize the spatial placement of tasks and their sequence. To this end, the algorithm needs to know the location of POIs including their category and be able to generate routes in a road network, as well as in a public transit network. The agent can then compute a mobility itinerary, containing a list of agenda items and a travel plan on how to reach the different location either by walking, by public transit, or by a combination of both travel modes.

This work aims to investigate whether it is possible to create an intelligent agent to support people in the agenda planning process and combine the process with mobility planning in a real-life setting.

3 RELATED WORK

Not much research has been conducted on general mobility-oriented agenda planning. The research on combining mobility and agenda planning also incorporating public transportation has, to the best of our knowledge, only recently started. Other optimization problems such as the Tourist Trip Design Problem (Vansteenwegen and Van Oudheusden, 2007) are, however, a specialization of the Mobility-Agenda Planning Problem, which focuses on tourist agendas.

3.1 Model of Mobility Agenda Planning

A model for mobility agenda planning has been proposed by (Wienken et al., 2014; Wienken et al., 2017). The authors discuss the possibility of combining mobility planning and agenda planning. This integration is possible due to a large similarity between both aspects.

Table 1: Similarity between information needs for agenda planning and mobility planning. (Wienken et al., 2014)

Information need of	
Agenda planning	Mobility planning
Purpose	Location
Date/Time	Date/Time
Location	Sequence of mobility
Sequence	Means of transportation
Social	Mobility impairments

Table 1 shows the identified information needs. The table indicates that both agenda items and mobility elements share some information needs such as the location and the time. A mobility agenda consists of agenda items and the mobility between the different agenda items. An agenda item has a purpose which determines whether it is a task or an appointment. The problem for the mobility agenda planning agent is to compute a mobility-oriented agenda that allows the user to be on time to all her appointments while performing an optimal number of tasks at their respective locations. Between the agenda items, the mobility agenda planner should also plan the mobility with different choices of transportation. The algorithm may tailor the selection of transportation and agenda items to a specific user profile, respecting the user's preferences.

As an example, a user gives the agenda planner a list of fixed appointments on that day and a list of tasks she wants to perform that day. The function of the mobility agenda planner is then to find a suitable location for the tasks if a location for the task is not specified. The mobility agenda planner should choose places in such a way that the resulting mobility agenda is optimal to the user, for example, as short as possible, while still allowing for the completion of all tasks.

3.2 Relevant Optimization Problems

In the literature, we have not found optimization problems that deal with computing mobility-oriented agendas in a general way, but we have found several more

specialized problems that can be generalized and be used as a basis for a mobility agenda planning algorithm.

Tourist Trip Design Problem. The MAPP is a generalization of the *Tourist Trip Design Problem (TTDP)* (Vansteenwegen and Van Oudheusden, 2007). The TTDP is a route planning problem that focuses on tourists, which want to visit multiple POIs in a city. The objective of the TTDP is to select a subset of POIs that match the user's profile and to maximize the user's satisfaction with the route. The authors suggested to calculate the satisfaction of a user with various parameters and constraints such as the choice of POIs, the order of the visits, the distance between POIs, the visiting time for POIs, the opening hours of POIs and further information such as entrance fees or weather conditions. The main difference between the TTDP and the MAPP is that the TTDP does not differentiate between appointments and tasks. Furthermore, the TTDP does not necessarily model dependencies between tasks and mostly does not distinguish between various categories of POIs. The mobility-oriented agenda planning agent must place certain tasks of the user, i. e. buying bread, at certain locations, such as a supermarket or a bakery, which is why the algorithm needs to distinguish between different categories and must map the intent of the user to certain POI categories. For a MAPP query it should, furthermore, be possible to plan certain tasks not before other tasks have been completed, e. g. visit the pharmacy after visiting the doctor.

Orienteering Problem. The TTDP has often been modeled as an *Orienteering Problem (OP)* in the literature (Vansteenwegen and Van Oudheusden, 2007). The OP, also known as the *Selective Traveling Salesman Problem*, can be seen as a combination of the *Traveling Salesman Problem (TSP)* and the *Knapsack Problem (KP)*. In the OP we have a graph with vertices and arcs. The vertices represent POIs, while the arcs represent the travel time between POIs. Each vertex is annotated with a score and a visiting time, and each arc is annotated with a travel time. The task of the OP is to compute a path visiting a certain subset of vertices in such a way, that the aggregated score of the vertices is maximized. The path is constrained by the time budget and the visiting times of the vertices added to the travel times on the arcs is not allowed to exceed the time budget. Furthermore, the score of each vertex is only allowed to be collected once. (Golden et al., 1987) have proven that the OP is NP-hard. Hence, it is infeasible to calculate exact results for larger problem instances and heuristics need to be found to determine quality solutions to the OP. Nearly all recent papers published on the OP and its variations describe an approximation algorithm for the problem.

More advanced approaches model the TTDP as a *Team Orienteering Problem with Time Windows (TOPTW)* (Boussier et al., 2007). The TOPTW is an extension to the OP, where the task is not to find a single route in the graph that maximizes the total score, but *r* routes, which maximize the aggregated score of all routes. Each route has a time budget and the score of a route is the score of the route's POIs. The score of each vertex is only allowed to be collected once. This extension of the OP was used to allow computing tourist trips spanning multiple days. Additionally, the OP has been modified to support time windows of vertices, so collecting the score of a vertex is only possible, if it is visited during its time window. With this extension, we can model the opening and closing times of POIs in the TTDP.

Furthermore, our approach should integrate public transportation as a means to travel. Later, we want to extent this support to new mobility offers such as car- or bike-sharing. Public transport, however, is schedule-based and therefore the travel times from one location to another are time-dependent. Depending on when a person arrives at the bus station, the travel time may differ. For dealing with time-dependency, the *Time-Dependent Team Orienteering Problem with Time Windows (TDTOPTW)* has been proposed. The problem description itself is the same as for the TOPTW, but the travel times on the arcs of the graph are now a function, whose value depends on the time for which it is traversed. Computing exact solutions, even for small problem instances of the Time-Dependent Orienteering Problem, is not feasible (Gunawan et al., 2014). The problem was attempted to be solved with the commercial *Integer Problem (IP)* solver *CPLEX*, which was not able to compute exact solutions to small problem instances in less than 24 hours. As interactions with an intelligent agent should be possible in nearly real-time, the runtime of the algorithm should be limited to less than a few seconds, making exact solutions not feasible for our use case.

For solving a TDTOPTW there are two main approaches discussed in the literature: The key idea of the first approach is to reduce a TDTOPTW to a TOPTW by assigning each arc in the graph the average travel time from one location to another (Garcia et al., 2013). Removing the time-dependency from the problem reduces its computational complexity. The TOPTW is then solved with an *Iterated Local Search (ILS)* algorithm. An ILS algorithm is a meta-heuristic algorithm applicable to a wide range of optimization problems (Lourenço et al., 2003). It operates in several iterations and performs local search steps in each iter-

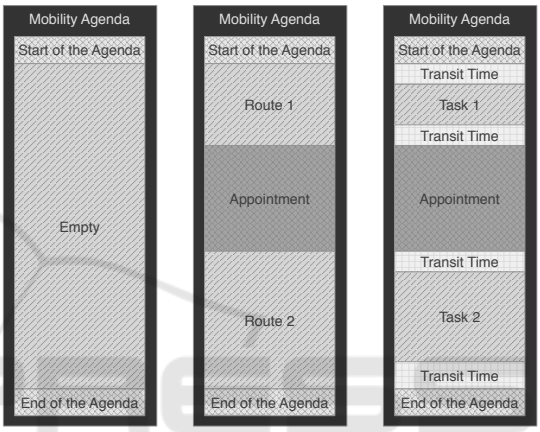
ation in an attempt to improve the best-found solution so far. While this approach reduces the runtime of the algorithm, the downside is that it may return infeasible mobility agendas, because it does not consider the actual schedule of the public transit vehicles. To handle these cases the authors in (Garcia et al., 2013) implemented a repair procedure, which uses a time-dependent route planner, to validate the solution. If an error is found, it attempts to repair the created route by removing visits from it.

The second approach also solves the TDTOPTW with an ILS algorithm (Gavalas et al., 2015). The key difference is, that this solution directly integrates the time-dependency and the problem is not reduced to a TOPTW, but is rather directly solved. The authors improved the runtime of the algorithm by using a clustering approach, which limits the search space of the algorithm and therefore shortens its runtime. The evaluation showed that this clustering heuristic only minimally affects the solution's quality, but greatly improves the runtime, when compared to the default ILS heuristic.

Finding Optimal Sequences. Another aspect of mobility agenda planning is to find optimal sequences of POIs. For this, we have researched optimization algorithms. The *Trip Planning Query* (Li et al., 2005), *Optimal Sequence Route* (Sharifzadeh et al., 2008), and *Multi-Request Route Planning* (Lu et al., 2017) problems all deal with finding a sensible sequence for visiting a number of POIs. They do, however, not include the functionality to reduce the requested POIs to a subset, but rather plan an optimal route containing all requested task, similar to the Traveling Salesman Problem. This means that there is no notion of a time budget, which the algorithm could consider. Additionally, as there is no time budget, there is also no score of the POIs, the algorithm rather minimizes the travel time required for the route, but does not consider a kind of score for the various POIs. Furthermore, these approaches are designed for road networks, meaning that the travel times are assumed to be time-independent and symmetric, which is both not the case for a public transit network.

4 MODELING OF THE PROBLEM

We propose to model the MAPP as a TDTOPTW and transform it the following way: As appointments are entirely fixed an inflexible, we can construct a corresponding TDTOPTW problem, where appointments are already handled. A graphical illustration is shown in Figure 1. Step 1 creates an empty solution with the starting and ending location of the agenda with the respective starting and ending times. In Step 2 we create a new route for each appointment in the agenda. This means, that we model the time of an appointment as the time difference between two routes. This modeling ensures, that the TDTOPTW algorithm respects the time windows and locations of the appointments. For example, the complete mobility agenda of a user starts at 8:00, ends at 18:00, and there is an appointment from 14:00 to 16:00. We model this problem as a TDTOPTW with two distinct routes: The first route r_1 starts at 8:00 and ends at 14:00, whereas the second routes r_2 starts at 16.00 and ends at 18:00.



(a) Step 1 (b) Step 2 (c) Step 3

Figure 1: Creation of a TDTOPTW instance from a MAPP instance, consisting of two tasks and one appointment. The corresponding TDTOPTW therefore consists of two routes.

The constructed problem ensures that the user is on time to her appointment because route r_1 ends at the location of the appointment at the time of the appointment. For each appointment of the user, we therefore, end a route, at the time when the appointment begins and start a new route, for the time when the appointment ends. The respective ending and starting locations of the routes are the locations of the appointments. As the CSCRoutes algorithm only inserts a task, if no route time window is violated, the timely arrival at the appointment location is guaranteed. After inserting the appointments into the solution, the algorithm can start to add tasks into the routes. The algorithm then has two routes available to insert tasks into. The insertion of tasks is performed in Step 3. Step 3 is done with the modified CSCRoutes algorithm, while Step 1 and 2 is performed by a pre-processing algorithm constructing the associated TDTOPTW problem. The CSCRoutes algorithm is introduced in more detail in Section 5.2. The modifications include the support of an order of agenda items and category constraints on the POIs.

We identified various other features that may be important for an agenda planner to support such as: Mutual exclusion of agenda items, deadlines, preferred order of agenda items, mandatory POIs (not an appointment), time-dependent scores for POIs, multimodality, and connection between agenda planners (i. e. plan a task together with other people). Some of them can easily be integrated into our approach such as deadlines, for example; while the integration of multimodality is more complex.

Scoring of Agendas. The main task of the agenda planner is to help the user plan tasks on a specific day. We have two ratings that are important for an agenda plan, the score of the agenda, denoting the preference to the user and the completion time of the agenda, i. e., the time it takes for the user to complete the agenda. Both of these ratings have to be correlated and be used for rating the agenda, otherwise unreasonable agendas may be returned. We have decided to use a weighted linear combination of the score of the agenda and the completion time. The score of the individual agenda tasks, however, must also be correlated to the time taken to complete them, otherwise the agenda planner may prefer low-scored short tasks over high-scored long tasks. Furthermore, the score of the individual agenda tasks is also an combined score consisting of several factors such as the priority of the task, the preference of the user and maybe further external information such as the weather forecast, for example. A user may be more inclined to perform tasks outside, for a nice day and prefer doing indoor activities on a rainy day. When using the weather forecast as input for the score of an agenda item, the score must also become time-dependent, as it may change during the day. The weather forecast may also influence the mobility choice of the user i. e. prioritize the bus on rainy days.

5 AGENDA PLANNING ALGORITHM

We compute mobility-oriented agendas with the help of a modified version of the *Cluster Search Cluster Routes (CSCRoutes)* algorithm proposed in (Gavalas et al., 2015). These modifications include the addition of various categories based on the OpenStreetMap category model, handling of dependencies between various insertable locations, computing the journey times between various locations on-the-fly during the execution of the algorithm and improving the algorithm's runtime by designing further insertion heuristics. CSCRoutes is a heuristic algorithm for the TDTOPTW based on the ILS meta-heuristic.

5.1 Workflow

The mobility-oriented agenda planning agent requires data from multiple information sources as it integrates agenda planning with mobility planning: A mobility-oriented agenda planner needs to have access to POI information because we require the POI's name, its category, its location, its opening hours, and a unique identifier. The POI information is used for the planning of tasks, as they may need to be performed at special locations, i. e. buying bread at a bakery. Furthermore, the algorithm must be able to compute public transit and walking journeys between the locations of different agenda items. To this end, we also need the schedule of the public transportation service. The workflow of the algorithm, which is illustrated in Figure 2, is then as follows: We first obtain the POI and road information from OpenStreetMap and extract the relevant information from it. Then, we also obtain the relevant public transportation information as a GTFS file[1] and combine it with the OpenStreetMap extract using Graphhopper to obtain routing information. Graphhopper is an open source route planning library, which recently added support for public transit planning. The CSCRoutes algorithm queries Graphhopper during the execution and no routing information between POIs is pre-computed. In contrast to that, the POI database is pre-computed by the algorithm from the OpenStreetMap extract. Afterward, we use the POI information and the route planner to run the adapted version of the CSCRoutes algorithm (Gavalas et al., 2015).

5.2 Optimization Algorithm

In the first step of the MAPP algorithm, before executing the optimization algorithm, we transform it into a TDTOPTW as explained in Section 4. The transformation already ensures, that all appointments are correctly planned. An optimization algorithm is then executed on the remaining spare time in order to find suitable tasks for this spare time. The optimization algorithm itself is mostly adopted from the *CSCRoutes* algorithm (Gavalas et al., 2015), with various changes to adapt it to the domain of agenda planning. The algorithm is changed to allow an enforced order on agenda items and to be able to handle POIs of different categories. CSCRoutes is based on the *Iterated Local Search (ILS)* meta-heuristic from (Lourenço et al., 2003) and employs several local search steps. The pseudocode of the ILS algorithm is shown in Algorithm 1. The ILS meta-heuristic consists of four methods that are required to be implemented: *GenerateInitialSolution*,

[1] https://developers.google.com/transit/gtfs/

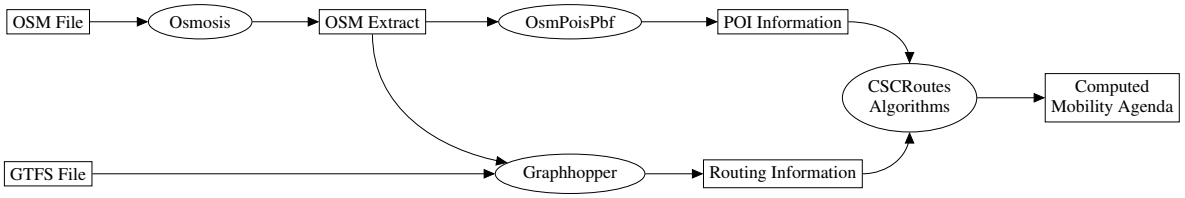


Figure 2: Workflow of the MAPP Algorithm. The rectangles show information, and the ellipses depict a modification of information.

LocalSearch, *Perturbation* and *AcceptanceCriterion*. At first, we generate an initial solution with the GenerateInitialSolution method and search for an local optimum using the LocalSearch method. After we initialized the solution the algorithm is repeated until the termination criterion is met. In this loop the algorithm attempts to escape from the local optimum with the Perturbation method from the previous iteration, reaches a new local optimum with the LocalSearch method and selects the starting solution for the next iteration with the AcceptanceCriterion method. When the termination criterion is met, the algorithm returns the best found solution.

Algorithm 1: Iterated Local Search Algorithm. Adapted from (Lourenço et al., 2003).

1: $s_0 \leftarrow$ GenerateInitialSolution
2: $s \leftarrow$ LocalSearch(s_0)
3: **repeat**
4: $\quad s' \leftarrow$ Perturbation$(s,$history$)$
5: $\quad s'' \leftarrow$ LocalSearch(s')
6: $\quad s \leftarrow$ AcceptanceCriterion$(s,s'',$history$)$
7: **until** termination criterion met

The CSCRoutes algorithm implements the different methods of the ILS algorithm. CSCRoutes has a *route initialization phase* which is run at the beginning that initializes the different routes with first tasks. Afterward, local search operations are performed in order to reach the local optimum. CSCRoutes has two local search operations implemented, the *insertion step* and the *replace step*. The insertion step inserts new tasks into the different route, until no insertion is possible anymore, while the replace step attempts to replace already inserted tasks with higher scored non-inserted tasks. Perturbation is implemented in the *shake step*. The shake step removes a certain number of tasks from the routes, so that new tasks can be inserted in the next iteration. So that the CSCRoutes algorithm does not get trapped in the same local optimum over and over again, it always continues its search from its last found solution, After reaching a pre-defined number of non-improving iterations, the CSCRoutes algorithm terminates.

Heuristics. As the search space for the agenda planning algorithm is large, especially if it has a nationwide database of POIs, heuristics need to be used to decrease the algorithm's runtime. Hence, the CSCRoutes algorithm only checks certain POIs for insertion in each iteration. For this, it assigns clusters to all POIs in the dataset based on their geolocation in a preprocessing step. The clusters are then used to constrain the insertion and replacement step of the algorithm. With the exception of the starting and ending location, this constraint forbids it to revisit clusters in a route. This property is called the cluster route constraint (Gavalas et al., 2015). Once a POI belonging to a cluster c has been inserted into the routes, all POIs belonging to c can only be inserted into the route, when it is adjacent to another POI belonging to cluster c in the route.

5.3 Problems with Existing Approaches

In Section 4, we have seen that the TDTOPTW needs to be extended to compute useful mobility agendas. The TDTOPTW lacks the definition of categories of POIs and dependencies between agenda items. In our use case, categories are required to find suitable locations for tasks. Furthermore, CSCRoutes preprocesses the data in such a way that using a larger number of POIs is infeasible (Gavalas et al., 2015). As a agenda planner should not only be usable in a confined area, but preferably work worldwide, the number of POIs may become large. The CSCRoutes algorithm, however, computes the fastest journey between all pairs of POIs for every time step and obtains a large travel-time matrix. (Gavalas et al., 2015) used 1440 time steps per day, one per minute. These time steps are needed to deal with the time-dependent nature of public transit journeys. However, the pre-processing step only needs to store the non-dominated shortest travel times between each pair of POIs and not a travel time for each time step. A departure dep_1 dominates another departure dep_2, if it starts later than dep_2, but arrives at the same time or earlier at the destination than dep_2. This pre-processing step is very costly, especially with larger number of POIs. The advantage of the preprocessing is that it allows to acquire relevant

travel times by a simple lookup in the matrix.

However, pre-calculating all travel times is infeasible, when regarding a larger number of POIs.

Instead of pre-calculating all pair-wise travel times, we utilize the GraphHopper[2] routing engine to find journeys from one location to another location. In theory, the regions to compute mobility agendas in, can then be significantly larger. The major downside of this approach, however, is that the computation of point-to-point journeys takes much longer than a matrix lookup. While we only need to read a single value from the travel-time matrix obtained by preprocessing the data, we need to compute a viable journey on-the-fly using a multimodal router. The computation of journeys takes several orders of magnitude longer compared to looking up the relevant information in the matrix. Hence, a trade-off between computation time and storage requirements exists, but the more POIs are supported, the less attractive the pre-processing step becomes.

Another significant advantage of the GraphHopper approach is that, we know the locations of transfers and the waiting time at transfer stations. This information is lost, when only storing the travel time and disregarding the transfers itself. Lastly, determining the required information for the user when calculating the route on-the-fly, is significantly simpler. A user expects to have an itinerary which tells her where to enter and to exit vehicles and which roads she should take to her location. The travel-time matrix itself only stores the travel time, therefore, there is a need for a post-processing step that attempts to recreate the meta-information of the route. By directly calculating the journeys with a route planner, we can easily extract the routing information for the user and exactly guide her to her locations.

5.4 Additional Heuristics

As the number of POIs that a mobility-oriented agenda planner needs to consider is much larger than a solver for the TTDP, we had to implement further heuristics to the algorithm to reduce the computation time. We have a trade-off between quality of the solution and the algorithm's runtime. The more the heuristic limits the search space, the worse the computed result will be, nevertheless we attempted to find heuristics that try to reduce the negative impact on the result's quality. The CSCRoutes algorithm spends most of its runtime in the scoring function of possible insertions, as it needs to calculate the additional time an insertion would cost. To score an insertion of a POI p_k between a POI p_i and p_j, two journeys need to be computed: The traveling time from p_i to p_k and from p_k to p_j.

Most of the time it is not important for the scoring whether the travel time is slightly inaccurate, as long as the calculated value for the travel time is close enough to the real travel time. The idea here is to only approximate the cost of an insertion in an attempt to minimize the number of journey computations. As we have already clustered the POIs, we will also use the clustering step to enable a better use of caching. In the scoring step, instead of calculating the travel time from one POI directly to another POI, we will use the travel time from a POI to the other POI's cluster centroid. On the one hand this approach results in more cache-hits and allows to reduce the number of costly travel time computations, whereas, on the other hand the scoring of insertions is less accurate as the travel time between POIs is only approximated. Nevertheless, the correctness of the results is ensured, because we will calculate the actual travel time between the locations, when inserting the selected POI into the agenda. As long as we only regard intra-city mobility agendas at the moment, this heuristic returns reasonable results, as most POIs are not far away from their respective centroids. When extending the support to inter-city mobility agendas, a better heuristic needs to be found.

Another heuristic regards the selection of possible POI insertions. The solution of (Gavalas et al., 2015) regarded most POIs for insertion, only limited by the cluster constraint. Therefore, we propose to limit the area in which the algorithm is able to select POIs for insertion from. This is problematic for the general OP, as the score of a POI and its location are completely independent. For the MAPP, however, an approximation like this is slightly less problematic as we assume that the score of POIs in the same category does not drastically differ. We restrict the algorithm, when inserting POI p_k between POIs p_i and p_j to only search for POIs which are in a certain radius around p_i or p_j, or along a corridor on the path from p_i to p_j. In this way we reduce the search space of the algorithm, while still allowing for many of the most important POIs to be inserted. If no suitable POI could be found for insertion, the search area can be widened. The downside of this heuristic is, that POIs that are further away, but where many other highly-scored POIs are located might be missed, when they are outside of the search area of the algorithm. This is not a problem for an agenda with many appointments and only short time windows for tasks, but becomes a larger problem when a lot of spare time is available. When more time is available, it might be more sensible to travel further to reach a cluster with several highly scored POIs, i. e. a mall, because the travel time inside the cluster will be small.

[2]https://github.com/graphhopper/graphhopper

If not much time is available in the route, traveling to a cluster that is far away does not make much sense, as performing all tasks at that cluster may not be possible due to the time constraints of the next appointment. Setting the search distance for POIs, therefore, could depend on the available spare time in that specific time slot in the route.

6 EVALUATION

In the following section, we will evaluate the practical feasibility of mobility-agenda planning and our agenda planning agent.

6.1 Methodology

To evaluate the agenda planning algorithm, we have designed a short survey with which we attempted to answer the following questions:

- Which aspects of agenda planning and mobility planning are important and how do they interact with each other?
- Which features of a mobility-oriented agenda are most important to users?
- Which features of a mobility-oriented agenda planning agent are most important to users?
- How well does the agent planned agenda fare against manually planned agendas by the participants?

The survey was structured into four different parts, for each of the above listed research questions. We aimed to evaluate mobility agenda planning, both qualitatively and quantitatively. Hence, the fourth part of the survey was interactive in which we have asked the participants to plan a mobility agenda of a fictive persona. We have designed a persona and a scenario as the agenda planner does not have a user interface as the moment. This allowed us to ask the participants to plan an agenda for the same query as our agenda planning agent. We were then able to compare the manually planned solution to the solution of the agenda planning algorithm. Additionally, we have asked the participants to evaluate the agent and manually planned mobility agenda themselves and list advantages and disadvantages of both agendas.

To ease the effort of planning an agenda for the respondents, we designed and implemented a journey planner based on GraphHopper that overlays the relevant POI information on a map, on an interactive website. On the one hand, this simplified the question for the participants, as only a single website was required to be used. While on the other hand, this allowed the respondents to work on the same data basis as the optimization algorithm. The fictive agenda that had to be planned, consisted of one appointment and seven tasks. This scenario was chosen, because planning appointments is not difficult, as they are completely inflexible. The interesting aspect is the temporal and spatial placement of tasks. Furthermore, two tasks had to be completed at specific POIs and in a specific time frame, whereas the others could be completed any time at their respective category. Additionally, we have added a single dependency between two tasks. The agenda then consists of two distinct routes. The total visiting time of all tasks (without the appointment) is 101 minutes, which does not include the journey time between the various POIs.

6.2 Results

We implemented a prototype agenda planning agent in *Python 3.5.2* with *SciPy* for the clustering algorithms and helper functions, *joblib* for the parallelization of the algorithm and *GraphHopper* for computing public transit journeys. The software was run inside a virtual machine configured with 8 vCPUs (Intel Xeon E5-2650 clocked at 2.20GHz) and 8GB of RAM. As this survey only attempts to show first insights into the topic and is not meant to be representative in any way, most of the 14 respondents have been students. For most of the respondents the location of the scenario was not well-known. Unfortunately, the sample group is too homogeneous for a meaningful qualitative feedback, while the sample size is too small for an expressive quantitative feedback.

The short evaluation of mobility agenda planners showed that most respondents think that an agenda planner would help them to organize and plan their day. They expect the agenda planner to be especially useful in unfamiliar locations or cities, whereas they only expect limited utility in known places. The minimization of unintended breaks in the agenda and the completion time was listed as crucial. For agenda planning in general, optimizing the agenda as a whole is more important to the participants, than optimizing the distinctive journeys between agenda items. Most of the respondents indicated that influencing the means of transportation is not very important to them, as long as the choice is sensible. A few respondents, however, expect to have full control over the choice of transportation. Specifying own time frames for certain tasks and an enforced order of agenda items were named as the most important features. Embedding external information, such as a weather-dependent score was listed as the least important feature.

Figure 3 shows the completion time of the created agenda. It can be seen, that the automatically planned agenda is slightly better than the average of manually created agendas. However, one participant of the survey has created a faster agenda than our implemented agenda planner, showing that there is still room for improvement for the agenda planning algorithm. It shows that, at least when only comparing the completion time of the agenda, the agenda planner is able to compute agendas that are slightly better than most of the manually planned agendas.

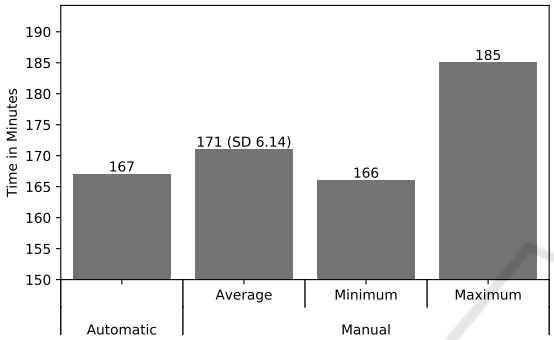


Figure 3: Completion time of the manually and automatically generated agendas.

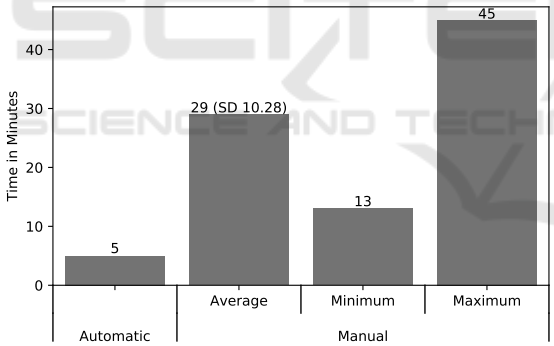


Figure 4: Planning of the manually and automatically generated agendas.

Figure 4 shows the time it took for planning the respective agenda. The agenda planner was a lot faster than the participants. This comparison, however, is not representative as the respondents had to use a newly developed tool, that they were not familiar with. Furthermore, the way the agenda had to be planned was not as it would be in a real-life setting. Nevertheless, it shows that planning an agenda is a time-consuming task, especially in a not so well known location. The agenda planner took around 5 minutes to compute a single agenda after 10 non-improving iterations. For an interactive agent system, these 5 minutes are far too long. Most of the runtime is spent in assigning scores to the possible insertions of tasks and in the computation of journeys between two locations, this part of the algorithm needs to be improved.

Even though the completion time is similar, people preferred aspects of their own agenda. One of the most noted aspect is that the agenda planner planned more tasks more towards the evening of the day, whereas most respondents said that they prefer doing their tasks in the morning. The agenda planner planned the tasks mostly in the evening, because it was slightly faster to visit them there. This is due to the fixed location tasks that had to be visited in the morning and in the evening, as the user was then already in the vicinity of other suitable locations. The manually planned agendas by the participants were mostly as compact as possible, i. e. they started the agenda early and ended it early. In contrast to that, the agenda planner placed more free time in the beginning of the agenda and placed more tasks into the afternoon. Therefore, the continuous free time was smaller than that of the manually planned agendas, which was perceived as a negative aspect in the survey. Since the agenda planner favors agenda items from the same clusters, the algorithm found clusters where several of the tasks could be completed, which people mostly missed. People liked this aspect of the agenda planner, because this means that the number of public transit journeys is reduced and locations which are reachable with a short amount of walking are preferred. Another aspect that people have criticized is that there is not enough room for customizability of the query at the moment. Overall, the computed agenda was rated with an average of 4.23 on a scale from 1 to 5, and half of the people remarked that the computed agenda is better than their manually planned agenda.

7 CONCLUSION AND FUTURE WORK

While no expressive conclusion can be drawn from the evaluation, it shows that planning mobility-oriented agendas with an intelligent agent is feasible in practice. The computed agendas were mostly shorter than the manually planned ones and were rated only slightly worse by the some of the participants of the survey. For the future, the evaluation and the algorithm's performance have to be improved. For this paper, we have evaluated the mobility agenda planner only with a single agenda, which only shows that it is able to compute mobility agendas in this one scenario. Therefore, a better evaluation method has to be implemented for our planner. Furthermore, the approach was only tested for intra-city mobility itineraries.

For future work we have, thus, identified two main aspects. The first aspect is to improve the evaluation

of potential solutions to the problem. For this, a user interface needs to be designed, which allows people to plan their own itineraries with the agent, so that more scenarios are covered. In addition to more scenarios, the review of people's own agendas will be more useful as people have a stronger opinion about their own mobility itineraries. Missing functionality and features can, therefore, be more easily detected. Research on the design on such an interface has already been started in (Wienken et al., 2017). Future work should evaluate these design guidelines and implement a user interface for the agenda planner so that people can interact with it. A user interface, however, may not be restricted to a graphical user interface. Currently intelligent personal assistants, such as Amazon Alexa, Apple's Siri, and Google Assistant are becoming more popular. They allow users to interact with software using natural speech. Such a electronic personal agent may support the user then in planning their mobility-agenda. As the personal assistant already has access to the calendar and the preferences of the user, it already has much of the required information. These personal agents already individually support the user in agenda planning and mobility planning, but an integrated solution that combines these aspects is missing. The intelligent agent will act more like a recommender system in such a scenario, because it does not automatically compute the whole agenda, but suggests date and locations for certain tasks of the user.

The other aspect of future work focuses on the optimization problem itself. Furthermore, the mentioned drawbacks and missing functionalities have to be addressed in future work, so that people can instruct the agent more precisely in how it should plan the mobility-oriented agenda. The algorithm's runtime also has to be improved, as waiting for 5 minutes in an interactive scenario for the answer of the agent is unrealistic. Additionally, the algorithm can be designed to respect user profiles, which we have not regarded at all. A user profile may either be obtained by explicitly letting the user complete a survey which then constructs the user profile or by learning the user profile while she interacts with the agent.

ACKNOWLEDGMENTS

This work was partially funded by the German Federal Ministry of Transport and Digital Infrastructure (BMVI) for the project "Digitalisierte Mobilität – die Offene Mobilitätsplattform" (19E16007B).

REFERENCES

Baena-Toquero, M. J., Muros-Cobos, J. L., Rodríguez-Valenzuela, S., and Holgado-Terriza, J. A. (2014). Towards sustainability in multi-modal urban planners. In *Connected Vehicles and Expo (ICCVE), 2014 International Conference on*, pages 492–497. IEEE.

Boussier, S., Feillet, D., and Gendreau, M. (2007). An exact algorithm for team orienteering problems. *4OR: A Quarterly Journal of Operations Research*, 5(3):211–230.

Garcia, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., and Linaza, M. T. (2013). Integrating public transportation in personalised electronic tourist guides. *Computers & Operations Research*, 40(3):758–774.

Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., and Vathis, N. (2015). Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers & Operations Research*, 62:36–50.

Golden, B. L., Levy, L., and Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, 34(3):307–318.

Gunawan, A., Yuan, Z., and Lau, H. C. (2014). A mathematical model and metaheuristics for time dependent orienteering problem. In *Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling*, pages 202–217. Research Collection School Of Information Systems.

Li, F., Cheng, D., Hadjieleftheriou, M., Kollios, G., and Teng, S.-H. (2005). On trip planning queries in spatial databases. In *International Symposium on Spatial and Temporal Databases*, pages 273–290. Springer.

Lourenço, H. R., Martin, O. C., and Stutzle, T. (2003). Iterated local search. *International series in operations research and management science*, pages 321–354.

Lu, E. H.-C., Chen, H.-S., and Tseng, V. S. (2017). An efficient framework for multirequest route planning in urban environments. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):869–879.

Sharifzadeh, M., Kolahdouzan, M., and Shahabi, C. (2008). The optimal sequenced route query. *The VLDB Journal — The International Journal on Very Large Data Bases*, 17(4):765–787.

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

Wienken, T., Krömker, H., and Spundflasch, S. (2017). Agenda planning-design guidelines for holistic mobility planning. In *International Conference on Human-Computer Interaction*, pages 713–720. Springer.

Wienken, T., Mayas, C., Hörold, S., and Krömker, H. (2014). Model of mobility oriented agenda planning. In *International Conference on Human-Computer Interaction*, pages 537–544. Springer.

Zografos, K. G. and Androutsopoulos, K. N. (2008). Algorithms for itinerary planning in multimodal transportation networks. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):175–184.