

Recommendation of Personalized Routes with Public Transport Connections

Bernd Ludwig, Bjørn Zenker, Jan Schrader

Chair for Artificial Intelligence
Friedrich-Alexander-University Erlangen-Nuremberg
Haberstraße 2, D-91058 Erlangen

ludwig@cs.fau.de, bjoern.zenker@cs.fau.de, jan.schrader@cs.fau.de

Abstract. ROSE (ROuting SErvice) is an application for mobile phones, which suggests events and locations to the user and guides him to them via public transportation. Many different systems partly incorporating recommending and navigation features exist. However, no system exists which combines event recommendation and pedestrian navigation with (live) public transport support. In this paper we describe ROSE a mobile application which combines these features. Our motivation is to free the passenger from many tedious tasks, e.g. finding an interesting event and navigating to it. ROSE determines the best possible transport link and then accompanies the passenger throughout his entire journey. It reacts in real time to delays in the public transport system and calculates alternative routes when necessary. For route planning in this context, we will propose a $h_\epsilon u$ -optimal algorithm for incorporating non-monotone multi dimensional user preferences in an A*-like algorithm. We also present an assignment of theoretical foundations to real world route planning problems.

1 Introduction

The development of mobile hardware has lead to a new variety of navigation systems. There are situations in which this is not the case, for example, tourists in unfamiliar cities can only use the navigation system, after they have looked up the places available to visit, e.g. a famous castle. Also city residents often do not know where to go or what to do. If they want to enjoy a jazz concert, they first have to look up when and where there is such a concert and then they have to plan the trip to it. For this, they have to use at least two different information sources: one to search for the concert and a navigation tool. In on-the-move situations this is not practical.

To ease the whole process of trip planning, we are developing ROSE which combines the recommendation of events and locations with navigation. To account for the weak hardware of mobile phones we use a client server architecture, where the ROSE server processes all data from different service providers like for example live public transport data, event and location directories or map services.

1.1 Combining Pedestrian Navigation with Event Recommendation and Live Public Transport Routing

We separated the system in three main parts:

Recommendation Part To get a recommendation, the user enters a query, like 'eat pizza', into his mobile phone. The recommender then generates a list of suggestions based on the user input and the user preferences. In this example it would likely be a list of restaurants which sell pizza.

For recommendation we use currently stemmed string matching, results are ranked using the Okapi BM25 measure [1].

The Okapi measure helps us normalizing documents of different length, because different event provider tend to send event descriptions of different sizes. Normally (without Okapi) the longer texts tend to score higher and shorter descriptions are getting unjustified penalized.

Route Generation Part After the user chose one of the presented options, the system calculates a route from the current location to the selected goal. To consider a diverse set of user preferences in route generation, we propose a $h_{\epsilon u}$ -optimal algorithm in section 2.

To ease the traveling, public transportation is also considered. The system calculates a route from the user's current position to the nearest public transport option, which means of transportation to take, where to change transportation and how to walk from the last stop to the goal location.

Navigation Part Figure 3 shows, how the route is displayed on a map on the mobile phone. If the route includes public transport, the next possible departure time is shown and the user is informed, whether he has to hurry to catch a bus. The system also informs the user if he has to hurry to reach his goal. As map-data we are using OpenStreetMap.

All three parts can be loosely coupled: the results of the recommendation are the input (goals) of the route generation. The result of the route generation is the input (way) of the navigation service. Such a loose coupling lacks the flexibility needed in many special situations, especially when errors occur, or the user behaves in an unpredicted manner. For example, if the user misses a bus, the system has to decide what to do: wait for the next bus, take another line, or just walk. Another case occurs, when the user wants to get a recommendation for multiple events, a route which goes through these events respectively. Often, the events that match well with the user preferences are located too far away from each other. This illustrates, loose coupling of recommendation and route generation to be disadvantageous.

This is an example of the disadvantage of loose coupling of recommendation and route generation.

To address these problems, e.g. considering distance when recommending multiple events, we propose a closer coupling of recommendation, route generation and navigation. Close coupling results in a theoretical problem formulation, as it can be seen in Section 3.

1.2 Current State of the ROSE System

To address the limitations of mobile devices like limited computational power and slow and expensive Internet access, we constructed a client server-architecture. Expensive calculations are moved to the server and the transferred amount of data is minimized. The ROSE system consists of the ROSE server, a J2EE application which integrates different services from multiple service providers and offers them as web services to the ROSE client, as you can see on the left hand side in Figure 1. The connections to 3rd party data is shown on the right hand side. In the middle you can see the ROSE databases which contain preprocessed data from various providers.

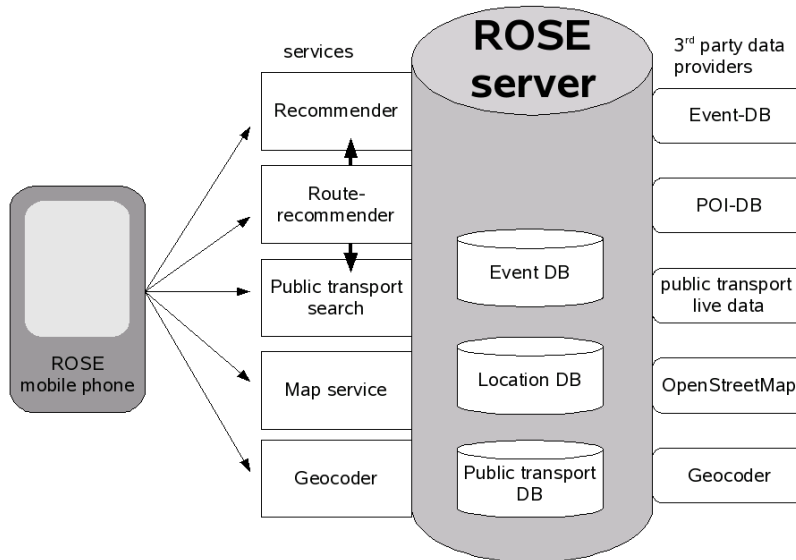


Fig. 1. Overview of the ROSE server

At the moment, we are developing two prototypes of the client, running on J2ME enabled mobile phones with GPS receiver and on Google Android.

The client sends the query over HTTP to the ROSE server. The server then gathers, based on the user's location, information from various providers, pre-processes it, and sends the result to the client. Routes, timetables, and live public transport data is obtained via VPN from a local public transport company.

On the client side, different localization services are integrated to cover a large number of devices and to allow navigation in various locations. As the start and end of a journey are often in buildings, and to support navigation also in subways a suitable indoor localization technique is needed. Therefore, we incorporate the Fraunhofer WiFi-localization module [2]. It is unique in that way as it works autonomously on the mobile terminal and does not need access to the WiFi access points. This protects user's privacy.

2 ALGORITHMS FOR COMFORTABLE ROUTES

Criteria for evaluating the quality of a route are limited mostly by formal constraints dictated by the algorithm used to find optimal paths. Efficient greedy graph search algorithms require the heuristic function to be monotone; A* even requires the heuristics to be optimistic, i.e. to never overestimate real costs of a path. In practice however, such constraints for heuristics are not adequate to reason about user preferences. In a survey conducted at our computer science institute among public transport users, the following criteria were marked as important by the test candidates:

- No long waiting time until departure
- Short duration of the trip
- Short foot walks
- Few changes of transportation
- No long waiting time during changes

Optimistic estimates for these criteria are just the function $f(x) = 0$; this amounts to omitting the criterion completely, which is an undesired consequence.

A second important finding of our study is that users do not evaluate the utility of a route on a one-dimensional scale (where there is always an optimum in a finite set or closed interval of utility values), but try to find a compromise between multiple attributes that are not comparable among each other. This finding is supported by investigations and studies in the field of psychology of decision making reported in [3-5]: Users accept locally sub-optimal proposals if they optimize the benefits of a proposal and minimize its risk globally (so-called noncompensatory decisions).

For example, somebody traveling with a lot of luggage accepts using a bus line that arrives some minutes later at the train station than the fastest one, but is much less crowded. Obviously, in this context, the comfort of the trip is valued higher than the duration. From an algorithmic point of view, this means that the standard shortest path approach cannot be applied successfully in order to satisfy the user needs as good as possible. However, searching according to a heuristic function that forces the search procedure to visit (almost) the whole search space is no attractive option for developing programs intended to run in real-time.

2.1 An $h_\epsilon u$ -optimal Algorithm for Comfortable Routes

The key to an efficient solution that belongs to the same complexity class as A^* and retains its soundness and completeness is therefore to use a) a monotone heuristic function h for computing correct solutions and b) to incorporate a non-monotone, multi-attribute heuristic u for the user preferences into the search procedure.

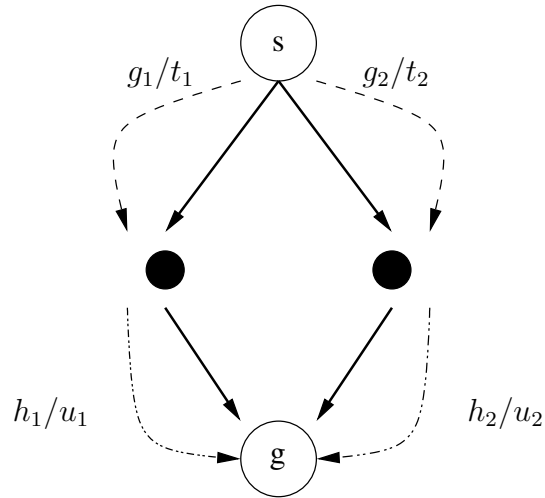


Fig. 2. $h_\epsilon u$ -optimal Expansion of the Search Space

In Fig. 2 you can see two paths from the start node s to a goal node g . g_i denotes the actual costs from s to the node which is currently being expanded, h_i the estimated costs to g according to the heuristic function h . u_i denotes the actual valuation from s to the node which is currently being expanded, t_i the estimated valuation to g according to the multi-attribute heuristic function u (user preferences).

If there is exactly one optimal successor state with $|g_1 + h_1 - g_2 - h_2| > \epsilon$, the search procedure works as usual. In the case of $|g_1 + h_1 - g_2 - h_2| \leq \epsilon$, both paths from s to g in Fig. 2 are undistinguishable in h .

In practical applications, if $|g_1 + h_1 - g_2 - h_2| \leq \epsilon$ then often other criteria become relevant for decisions in the sense of [5]. In order to explain the meaning of ϵ , let us consider the two public transport connections from s to g in Fig. 2: one takes 50 minutes, the other one 47 minutes. However, the first connection requires fewer changes of transportation than the second one. If the user dislikes changes, the first connection is optimal according to the user preferences u under the assumption of an ϵ -tolerance $\epsilon \geq 3$ minutes.

In order to select a path that is both optimal in the sense of h given ϵ and in the sense of u , we compute the ranking $p_1 = t_1 + u_1$ and $p_2 = t_2 + u_2$ of both options. The vectors p_1 and p_2 represent all user preferences in u . Each dimension represents a single preference (cf. the list of criteria in section 2). As the preference space to which p_1 and p_2 belong is partially ordered, in general there is no unique minimal element. Therefore, we compute a pareto-optimal set which includes all paths which cannot be distinguished neither by h nor by u . The final selection of a path which is h_ϵ -optimal and u -optimal is performed by applying a decision procedure (as described in section 2.3). We call the result $h_\epsilon u$ -optimal.

2.2 Approximation of the $h_\epsilon u$ -optimal Algorithm

Instead of implementing the $h_\epsilon u$ -optimal algorithm directly, we simulated it's behavior by having the path search procedure compute the n -best list of solutions for a given destination. Each of these n solutions is evaluated according to the user preferences. The final set of $m < n$ solutions is the set of m routes which are $h_\epsilon u$ -optimal.

In our first prototype, we implemented the n -best approach in order to obtain an evaluation platform as fast as possible. The algorithm works in two steps:

1. Compute n best results

Just computing the n best routes using always the same heuristics often leads to proposals that only differ minimally among each other — in particular, if just one line serves as public transport to the destination. Therefore, it is better to compute n routes using n different (optimistic) heuristics and to compare the n resulting best routes. This observation has also been made by [6].

2. Rank the n best lists

In order to get a global score for each user preference and each route, we sum up the contributions of each segment of the route to each criterion. The sum is called the rating of the route corresponding to the criterion under investigation. Finally, a total score is computed: The easiest approach to take multiple criteria into account is to compute a weighted sum of all criteria by multiplying each rating with the weight for the preference as entered by the user in the configuration dialog for the ROSE system (see Figure 1 *left*).

A more elaborate approach is to compute a pareto-optimum for the rating of the n routes. Beyond that, it is possible in our system to apply other multi-attribute decision rules, as the *Take The Best* decision rule proposed by [7].

We have implemented the described approach in JavaME for a standard Nokia N95. Figure 1 *right* shows a screenshot for a calculated route in Nürnberg (Germany) town centre. On the average, the algorithm takes just a few seconds to compute at least five solutions for the user's request. The underlying public transportation network consists of about one hundred bus stops. There are ca. 20

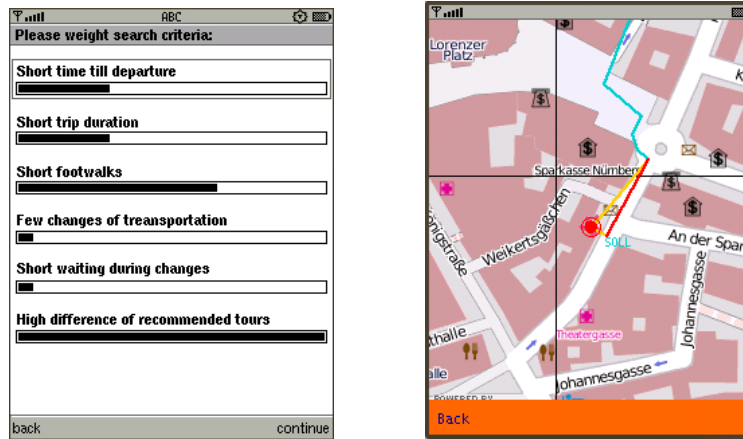


Fig. 3. *left:* Configuration Dialog, *right:* Calculated route on mobile phone (map from OpenStreetMap under CC-BY-SA)

bus lines, each of them departing from 5 to 10 times per hour. As a consequence, the memory consumption of the search procedure is high: the frequency of the bus lines leads to large open and close lists. To reduce the search space, we implemented a number of domain-dependent pruning strategies.

2.3 Multi-Attribute Decisions

A common and often used decision rule for multi-attribute decisions is the weighted sum. As known from decision theory, weighted sums are problematic as incommensurable data is mapped onto a unique scale. For example, the duration of a trip and the amount of space in a bus available for a passenger's luggage do not have the same units. Therefore, just multiplying such values with a weight factor and adding it to some other weighted value makes it hard for the user to understand the system's decision. Anyway, it is difficult for the designer of such a system to improve the selection process by adjusting the weights, as it is impossible to state that the user always prefers one criteria over others, e.g. short waiting time is preferred over trip duration.

In order to better model the way users take their decisions among several proposed routes, we implemented a module for multi-attribute decisions. Its basis is formed by a decision-oriented programming language. Figure 2 shows a code fragment for a decision in this language called MADL (Multi Attributive Decision Language). The decision named 'doWhat', using the pareto decision rule, consists of three alternatives, namely taking bus 293, waiting and taking bus 288 or walking. The goal is to minimize the three given attributes 'waiting Time', 'duration' and 'trip duration'.

The advantages of using MADL are: all decision specific information is stored in one designated place and easily understandable and adaptable by humans. It

```

ParetoDecision doWhat
{
  ALT [bus293]
  ALT [wait, bus288]
  ALT [walk]

  GOAL MIN!(waitingTime)
  GOAL MIN!(changes)
  GOAL MIN!(tripDuration)
}

```

Fig. 4. MADL decision

allows configuring the parameters that influence a decision and the strategy (decision rule) used to make a decision in a given situation. Furthermore, MADL allows combination and inheritance of decisions. It fosters the usage and evaluation of different decision rules, as they are already implemented and can be used out of the box.

2.4 Comparison

Our algorithmic idea is similar to the approaches of various recommendation and navigation systems. COMPASS [8] and Magitti [9] employ different prediction strategies in recommendation and rate the yielded results based on user preferences. P-TOUR [10] uses a genetic algorithm to find different near-best solutions and presents them to the user in a k-means clustered overview, from which he can select his preferred route. RouteCheckr [11] employs a multi-criteria Dijkstra based algorithm, which remains limited to the usage of the weighted sum. As it is not using an estimation function, it does not have to cope with non-monotone and optimal criteria, but is presumably slower as an algorithm using such a heuristic. Hochmair [12] studies, which decision rules bicyclists utilize in route planning and discusses different decision rules. He concludes that a compensatory decision rule should be used, but he does not implement this concept in a new algorithm.

In contrast, PECITAS [6] is a mobile personalizable navigation system, which advises routes using means of public transportation. However, it does not include recommendation of events or locations and routes are restricted to one starting point and one destination point only. For user adaptation, PECITAS generates multiple routes by using different heuristics (e.g. fastest route, or not taking any bus) and ranks them according to user preferences (walking preferences, number of bus changes, arrival at destination, sightseeing).

Compared to these systems, the unique features of ROSE are:

- routing to multiple destinations,
- integration of recommendation and route generation with live public transport support and navigation

- usage of various compensatory decision rules.

3 ASSIGNMENT TO THEORETICAL PROBLEMS

The presentation of the ROSE system and its capabilities highlights the fact that applications such as ROSE should offer two modes of usage:

Single destination mode: the user wants to reach a single location or complete just one tasks and needs assistance in finding the appropriate location and a transportation link to it.

Multiple destination mode: the user wants to reach more then one location or complete several tasks. The locations or the respective tasks are partially ordered leading to some time constraints that have to be respected by ROSE when it recommends locations and transportation links.

In order to give an overview about the algorithmic complexity for the functionality that has to be provided by ROSE, in this section we sketch the graph theoretical problem for each (level of) functionality. The most fundamental problem finding the shortest path from one location to another is known as the shortest path problem and investigated in depth. It solves the problem of finding a transportation link without respecting a time table. An extension to this problem includes public transport. A theoretical definition of this problem is known as time-dependent shortest path problem (TDSP) and according algorithms can be found in [13] and [14].

A different extension is to find a best path between multiple, differently valued locations. This is known as the orienteering problem (OP) [15], which is NP-hard. Furtheron, considering opening hours of locations leads to the orienteering problem with time windows (OPTW). [16] gives an overview. This problem is extended for the tourist route recommendation context by [17] which takes multiple constrained attributes into account, e.g. a maximum price of the trip. [16] presents an iterated local search based algorithm which yields good results ('[...] below 2 seconds for problems with up to 100 locations [...]'). For our application scenario we need another extension to support public transport in route finding. As [14] shows, normal shortest path problems have little in common with public transport best path problems. [18] introduces the time-dependent orienteering problem (TDO) which adds public transport support to the OP.

By taking time windows into account, TDO is extended to the Multi Path Orienteering Problem with Time Windows (MPOPTW). [19] recently proposed an algorithm for this problem.

Table 1 shows an overview about some route planning problems, their corresponding theoretical problems and adequate algorithms.

4 CONCLUSION AND OUTLOOK

We introduced ROSE, a mobile system for recommending events and planning routes. For considering arbitrary user preferences when using A* like search,

Route Planning Problem	Graph-Theoretical Problem	Most Efficient Algorithm
Single destination		
	Shortest Path	Dijkstra, A*
with public transport	Time-dependent shortest path (TDSP)	PFS [14] , Ding [13]
Multiple destination recommendation		
	Orienteering Problem (OP)	Fischetti, Salazar, Toth [15]
with time windows	Orienteering Problem with time windows(OPTW)	For an overview see [16]
with time windows and constrained attributes	Multi-constrained team orienteering problem with time windows (MCTOPTW)	Garcia et al. [16]
with public transport	Time-dependent orienteering problem (TDO)	Fomin, Lingas [18]
with public transport and time windows	Multi Path Orienteering Problem with Time Windows (MPOPTW)	Garcia et al. [19]

Table 1. Assignment of route planning problems to theoretical problems

we developed a $h_{\epsilon u}$ -optimal algorithm. Combined with MADL it allows easy integration of multiple criteria and different decision rules into its heuristic.

We studied mappings of different real world problems to theoretical problems and found out, that at the moment, no theoretical problem formulation for our scenario exists.

Our next steps are developing a suitable heuristic algorithm for solving the MPOPTW problem in our context and to conduct user studies to find criteria for the heuristic and more information about decision rules in the combination of recommendation and route finding.

Additionally, the impact of close and loose coupling of the recommendation and route finding services on the user satisfaction shall be researched. As we presume that the algorithmic complexity of close coupling is too big and that loose coupling is not satisfactory, we want to research, how an intermediate coupling between these extremes could be realized.

References

1. Singhal, A.: Modern information retrieval: A brief overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering **24**(4) (2001) 35–42
2. Meyer, S., V.T.H.S.: Wi-fi coverage and propagation for localization purposes in permanently changing urban areas. International Journal of Information Technology and Web Engineering, Amsterdam, The Netherlands (2008)

3. Tversky, A.: Elimination by aspects: A theory of choice. *Psychological Review* **79** (1972) 281–299
4. Tversky, A.: Choice by elimination. *Journal of Mathematical Psychology* **9** (1972) 341–367
5. Kahneman, D., Slovic, P., Tversky, A.: *Judgement under uncertainty - Heuristics and biases*. Cambridge University Press, Cambridge (1981)
6. Tumas G., Ricci, F.: Personalized mobile city transport advisory system. *ENTER Conference 2009* (2009)
7. Gigerenzer, G., G.D.G.: Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review* **103** (1996) 650–669
8. van Setten, M., P.S.K.J.: Context-aware recommendations in the mobile tourist application compass. *Adaptive Hypermedia and Adaptive Web-Based Systems (1111)* 235–244
9. Bellotti, V.e.a.: Activity-based serendipitous recommendations with the magitti mobile leisure guide. *CHI 2008 Proceedings* (2008)
10. Maruyama, A., S.N.M.Y.Y.K.: P-tour: A personal navigation system for tourism. *Proc. of 11th World Congress on ITS* (2004)
11. Voelkel, T., W.G.: Routecheckr: personalized multicriteria routing for mobility impaired pedestrians. *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility* (2008) 185–192
12. Hochmair, H.: Decision support for bicycle route planning in urban environments. *Proceedings of the 7th AGILE Conference on Geographic Information Science*, Crete University Press, Heraklion, Greece (2004)
13. Ding, B., X.Y.J.Q.L.: Finding time-dependent shortest paths over large graphs. *EDBT Proceedings* (2008)
14. Huang, R.: A schedule-based pathfinding algorithm for transit networks using pattern first search. *Geoinformatica, Greece* **11** (2007) 269–285
15. Fischetti, M., S.J.T.P.: Solving the orienteering problem through branch-and-cut. *INFORMS J. Comput.* **10** (1998) 133–148
16. Garcia, A., V.P.S.W.L.M.T.A.O.: Iterated local search applied to the multi-constrained team orienteering problem with time windows. submitted to *Computers & Industrial Engineering* (to appear)
17. Vansteenwegen, P., S.W.G.A.: Personalised tourist guide: Multi-constraint team orienteering problem with time windows. *Proceedings of ORBEL2009, Leuven, Belgium* (2009)
18. Fomin, F. V., L.A.: Approximation algorithms for time-dependent orienteering, information processing letters. *Information Processing Letters* **83** (2002) 57–62
19. Garcia, A., A.O.O.O.V.P., Linaza, M.: Public transportation algorithm for an intelligent routing system. 16th ITS world congress, September 21-25, Stockholm, Sweden: to appear (2009)