2011 35th IEEE Annual Computer Software and Applications Conference

Selection and Ordering of Points-Of-Interest in Large-Scale Indoor Navigation Systems

Martin Werner
Mobile and Distributed Systems Group, Ludwig-Maximilians University Munich
Email: martin.werner@ifi.lmu.de

***Abstract*—Indoor navigation systems guide users through complex buildings, which they do not know in advance. The complexity of public buildings such as airports, train stations or hospitals leads to new variants of well-studied NP-hard optimization problems such as the Traveling Salesman Problem, where most of the classical approximations are not directly applicable for fundamental geometric reasons. Fortunately we are able to solve the upcoming small instances of these problems to optimality in a very short timeframe. With this paper we define the Partially Ordered Traveling Salesman Problem, explain how it comes up in indoor navigation applications and present two algorithms, which solve or approximate the Partially Ordered Traveling Salesman Problem for small and medium size instances in a timeframe of less than one second and hence allow for a real time and context-aware indoor navigation experience. We then evaluate both algorithms using realistic data modelling the public area of Munich airport spanning nearly 200.000 square meters.**

I. Introduction

Global growth of the use of navigation applications for organization, documentation and optimization of processes in the outdoor area has led to increasing interest in providing comparable services for indoor environments. However the situation indoors is fundamentally different: First of all there is no cheap and everywhere available positioning system in contrast to the outdoor situation where GPS is globally available. The second major difference is the availability and complexity of maps. While for the outdoor area street maps are more or less freely available and ready to use for navigation, the situation for indoor areas is fundamentally different. Usually no map data is available and there is no possibility to generate usable maps from cheap sources, such as provided by satellite images for the outdoor case.

Nevertheless it is possible to overcome all these problems nowadays. As there is no cheap and high quality mobile positioning technology one can use a fixed positioning technology based on screen interaction. This idea first appeared in [1], where a grid of autonomous displays is spread out over the navigation space used for user interaction and positioning, and was extended in [2], which is the basis of this research. In a cooperation project with Munich airport we are developing a large-scale indoor navigation system using a distributed user interface for positioning and enhanced CAD floor plans and lists of points of interest for navigation.

In this system the potential user interacts with a large set of displays distributed over the area of Munich airport. To initiate a navigation session, the user stands in front of a terminal and can either use a video conference with an employee of the airport to explain his needs (flight number, additional interest in shopping and services, etc.) or use a simple touch-screen user interface to define the points of interest for his session. The system then dispenses a cheap RFID-card to the user which is used to identify his navigation session. The user is then presented with an animated map of the airport showing the way to the next point of interest and the remaining time until boarding if applicable. The user then can interact with the system by putting his RFID-card on the reader of any of the distributed displays and get information about his route to the next point of interest and a touch-screen user interface similar to the user interface of the prototype.

With this navigation system prototype we are facing at larger and much more complex indoor surroundings than most of the previous work. The indoor areas we are facing at are

- **very large:** the data used for the evaluation spans nearly 200.000 m^2
- **very restrictive:** not all reachable places are reachable to anyone and some connections are only allowed in one direction
- **semantically complex:** there are many special places such as shops, elevators, escalators and repeating points of interest such as toilets

To support these facts, we chose to use a grid-based search space representation of uniform high resolution. The main navigation structure is a big graph which does not have any of the properties common to navigation graphs: It is not planar, it does not possess an embedding into a metric space such that this metric space is locally equivalent to Euclidean space (which we would need to estimate times using distances in a sensible way).

The most complex problem which this indoor navigation system has to solve is the topic of this research. Given a set of points of interests at the airport, find a total ordering on this set such that the resulting tour is reasonably short and such that you are allowed to walk every part of the tour. At an airport it is for example impossible to walk backwards through security checks. In comparison to the first prototype, which was able to handle only five points of interest in a small area solving a simpler problem, we manage to handle up to ten points of interest in a correct way below one second for the

0730-3157/11 $26.00 © 2011 IEEE
DOI 10.1109/COMPSAC.2011.71

full navigation space and much more points of interest using an approximation of the problem which leads to acceptable quality results.

A very important consideration in this project is that the topological information defining the navigation information has a long lifetime and is expensive to generate. Hence the navigation system should be set up to transparently provide service to the complete indoor area and not only to a limited interconnection network of points of interest. In this project we designed and developed an indoor navigation service that can calculate shortest routes between any two points in the navigation space as well as order points of interest in admissible order in a limited time-frame. Due to the fact that the system is context-aware and can for example detect gate changes and work with waiting times at security checks, it is clear, that no calculations can be cached and reused and that for each interaction with the screen all this information has to be regenerated on the fly which imposes strong performance requirements on the navigation system.

The rest of this paper is organized as follows: In the next section, we define the problem in detail and compare it with the related well-known Traveling Salesman Problem and give hints on important differences to the usual Traveling Salesman situation. In section III we review related work in the area of the Traveling Salesman Problem. We do not review related work in the area of indoor navigation; the interested reader is referred to [1]–[7]. In section IV we explain two algorithms that can be used to solve or approximate the Partially Ordered Traveling Salesman Problem. In section V both algorithms are evaluated using realistic data modelling the complete public indoor area of Munich airport. In a final section we draw conclusions and give hints on open questions for further research.

II. Problem Statement

The problem which we want to solve with this paper is coming up in indoor navigation applications such as the one we are working on. Due to the similarity to the Traveling Salesman Problem, we call this problem the Partially Ordered Traveling Salesman Problem.

Assume we are given a weighted graph G and a partially ordered subset $P \subseteq V(G)$ of the vertices of G. Defining a tour to be given by a total ordering of the vertices in P and the length of such a tour as the sum of the length of the shortest ways interconnecting the vertices of P in this respective order, find the shortest tour.

Note, that the graph need not and usually will not have any good properties such as being connected or allowing an embedding into a metric space such that the weights are compatible with the distance of the metric space and that the graph will usually be very large.

This problem is to some extent similar to the well-known Traveling Salesman Problem (TSP). A detailed discussion of the Traveling Salesman Problem and approximations to this problem is postponed to section III. For now, we just formulate the Traveling Salesman Problem and discuss some differences, their origin and their impact on the problem. The most classical Traveling Salesman Problem formulation is the problem of a salesman to decide in which order he should travel a set of cities to minimize his time of travel and could be formally described as:

Assume we are given a weighted complete graph G with non-negative weights. Defining a tour to be given by a total ordering of the vertices in G and the length of such a tour as the sum of the length of the edges interconnecting the vertices of P in this total ordering (returning home from the last to the first vertex), find the shortest tour.

There are some important differences between the Traveling Salesman Problem and our Partially Ordered Traveling Salesman Problem. Obviously the classical Traveling Salesman Problem does not have any order relation that a solution should fulfill. Moreover due to the nature of a salesman returning home, we are looking for a solution which does return home. Note that it is not important to know the home of the salesman as every shortest tour could be rotated to start with any of its vertices. In other words: The search space is by factor n smaller than in our problem, because two tours are equivalent for the original Traveling Salesman Problem, if they can be rotated to be equal (where rotation means to move the home city into one direction around the tour). Unfortunately in our situation these rotations are not admissible, because they do not respect the partial ordering.

In contrast to our general formulation of the Traveling Salesman Problem, the problem is often only treated in more special cases. As these special cases are very important for most well-studied approximation algorithms, we want to explain the most important ones. The Metric Traveling Salesman Problem is a Traveling Salesman Problem, where the weights have to fulfill a triangle inequality $\mathrm{w}(ac) \leq \mathrm{w}(ab) + \mathrm{w}(bc)$. For the Euclidean Traveling Salesman Problem the graph possesses an embedding into two-dimensional space R^2 such that the weights and the Euclidean distance coincide.

It is easy to see, that for indoor areas and for a graph with an embedding such that the weights are locally proportional to the Euclidean distance, none of these good properties can hold. A single ascending escalator for example leads to a short travel time upstairs but to a completely different (longer) way downstairs. Hence the weights can not be Euclidean as they are not even symmetric.

For the case of the Metric Traveling Salesman problem, see figure 1. In the depicted triangle, the travel time for edge b is high due to the stairs, the travel time for edge c is small due to the speed of the escalator and the travel time for the remaining edge a can be neglected. Now it is obvious that the relation $b \leq a + c$ can be broken by speeding up the escalator or by longer stairs. In natural language this relation reads: ”Walking a staircase is always faster if you already stand in front of it than using an escalator next to the staircase”, which is obviously wrong.

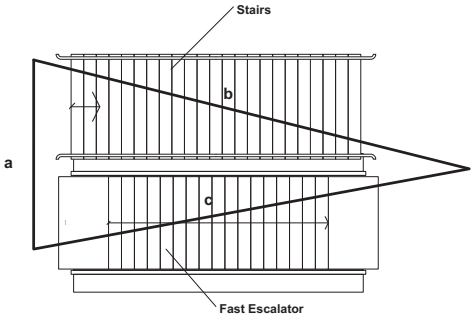


Fig. 1. An example showing that indoor navigation areas do not lead to Metric Traveling Salesman Problems

III. Related Work

As our problem is related to the Traveling Salesman Problem, we want to review in some detail, how the Traveling Salesman Problem has been approximated and explain the basic idea of genetic algorithms.

A. The Traveling Salesman Problem

The Traveling Salesman Problem is a very well studied NP-hard problem (follows from the remarkable work of Karp [8] and the well-known fact that the Traveling Salesman Problem is equivalent to the Hamiltonian Circuit Problem). Hence we do not know, whether a polynomial time algorithm exists. The simplest correct algorithm for the Traveling Salesman Problem has a time complexity of $O(n!)$ and consists of exhaustively searching the space of all tours in the graph and remember the shortest tour. Beside this, many approximation algorithms have been studied. Approximation algorithms try to solve the problem in polynomial time with a known relative error bound. It has been shown, that for the case of the Traveling Salesman Problem no k-factor approximation can exist, unless P=NP [9]. This means, that there is no polynomial time algorithm A such that

$$\frac{1}{k}\,\mathrm{Opt}(T) \leq A(T) \leq k\,\mathrm{Opt}(T) \text{ for any } k \geq 1$$

where T is an instance of the problem, $\mathrm{Opt}(T)$ denotes the length of the optimal solution and $A(T)$ denotes the length of the result of applying the algorithm A to the instance T.

To handle this situation, the algorithms for solving the Traveling Salesman Problem generally make use of simplifications of the problem. A very common case is that the Traveling Salesman Problem is metric, which means, that all triangles built from edges in the graph fulfill a suitable triangle inequality as explained in the introduction. In this case the simpler problem of finding a closed walk with possibly doubled edges suffices to solve the problem. In this situation the Christofides-Algorithm [10] is the best known approximation algorithm with a performance ratio of $k = \frac{3}{2}$ and a running time of $O(n^3)$.

In contrast to this geometric simplification, there are also problem simplifications related to the technique of local search. The basic idea is to start with some heuristically generated initial tour and continue using "local" modification to the tour. The worst-case performance of these algorithms is really bad (i.e. the tour can be very long for some instances and the running time is exponential). Due to difficulties of using such local search techniques while respecting a partial ordering, we did not investigate them further.

Another successful technique is called Branch and Bound and tries to quickly subdivide the set of all admissible solutions. This technique is based on two algorithmic steps. In the first step all admissible tours are divided into at least two classes such that in the second step a lower bound on the length of any solution in the subsets can be computed. Unfortunately we did not (yet) find a way to calculate a lower bound on a set of possible solutions in feasible time taking into account the restrictions imposed by the partial ordering and hence are not able to apply this idea now.

A fundamentally different idea to tackle with the Traveling Salesman Problem comes from the area of Genetic Algorithms. The main ingredient here is a sufficiently random process combining partial solutions (often called "genetic ideas" in this context) to generate better ones. In general a genetic algorithm uses a coding to translate a possible solution (e.g., a tour) into a string over a finite alphabet (e.g., a sequence of numbers identifying a city) which can be evaluated by a cost function (e.g., the length of the tour). Then based on the fitness (e.g., the value of the cost function) of the chromosomes, pairs of chromosomes are chosen with a slight randomness in each generation and a suitable crossover of genetic material (parts of the string) is performed. To prevent this algorithm from getting stuck in a local optimum a mutation operation is introduced with a specific probability. Running this game of life for some time, the best individual chromosome of the population becomes stronger and stronger if the crossover operation is able to combine ideas to better ideas (i.e., if the probability that a crossover between strong chromosomes is stronger than before is high). For this type of algorithm it is usually impossible or very difficult to calculate approximation factors. Nevertheless they provide good solutions in short computation times.

IV. A Solution and an Approximation to the Partially Ordered Traveling Salesman Problem

In this section, we define two algorithms which we used to handle the Partially Ordered Traveling Salesman Problem in our navigation system. When these algorithms are invoked, the framework has generated some information for them: A list of points of interest, a complete distance map for the interconnection of these points, and an order number for each point of interest. The partial ordering comes from the less-or-equal ordering on these ordering numbers. Note, that the same ordering number can come up multiple times.

A. The only known solution: Exhaustive Search

The first algorithm generates all permutations of the cities and for each permutation it checks, whether this permutation is

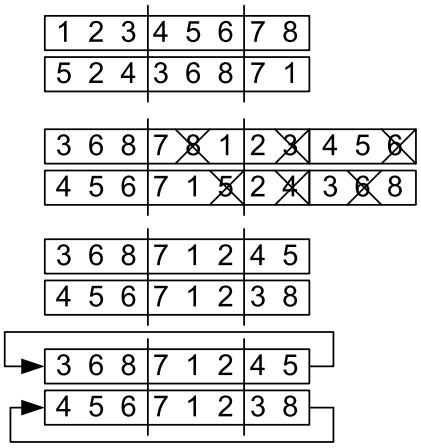


Fig. 2. An example for the steps of the crossover operator

well-ordered and calculates the total distance keeping track of the ordering with the shortest total length. One might wonder here, why we are creating all permutations and not only the admissible ones. The reason is, that we are not able to do this in a faster way. The first reason is that by the definition of the navigation interface we would need a sort operation to calculate the equivalence classes defined by equality in the partial ordering and then generate iterated permutations inside these classes. The second reason is, that many search operations will use a trivial partial ordering (e.g. all points of interests except the first, where we are currently positioned, are in the public area of the airport) and hence the partial ordering does not efficiently reduce the search space size for these requests.

This algorithm is correct and can be used on sufficiently small instances to calculate the optimum solution. We will exploit this fact to calculate relative errors for our genetic algorithm in section V.

B. A Variant of Genetic Search for the Partially Ordered Traveling Salesman Problem

As explained in the related work section, a Genetic Algorithm is based on a survival of the fittest strategy using partial and randomized information exchange between a set of possible solutions. In our case a chromosome is defined to be a list of integer values identifying the points of interest. We are working on a fixed size population of such chromosomes in the following way: For a fixed number of generations we calculate a fitness value for each chromosome by calculating the length of the encoded tour. Then we choose two individuals from the current generation using a roulette-wheel technique, where the probability of being chosen is equivalent to the fitness of a chromosome in comparison to the sum over the fitness values of all individuals. Having selected two individuals, we perform a new crossover operation defined below and generate two outcomes. We then choose the two fittest individuals from the chosen two and the outcome to survive and insert them into the next generation. As a mutation step, we replace a cromosome with a new one which is randomly generated and correctly ordered.

The most important operation for this genetic algorithm is the crossover routine. Many crossover routines have been proposed for the Traveling Salesman Problem. The Partially Matched Crossover (PMX) appeared in [11] calculates two crossover points and swaps the chromosome portion inbetween. Then in the doubled cities outside the crossover are being repaired using the same swaps as those done in the crossover region. Unfortunately, a large crossover area keeps only a small part of information intact.

A better crossover operator in this case is the Ordered Crossover (OX) [11]. Ordered Crossover works similar by choosing two crossover points and swapping the genes inbetween the two points. But then from the remaining genes those genes are taken which are not in the crossover area leading to a valid tour which remains much more of the sequential structure of a partial solution.

Another popular crossover operation uses heuristic information. The Heuristic Crossover [12] chooses a random city and inserts the shortest edge leaving this city in the parent chromosomes into the outcome. With this edge chosen, the Heuristic Crossover takes again the shortest of the outgoing edges of the second city unless this choice introduces a cycle and hence leads to an invalid tour. In this case a random edge is chosen which does not lead to a cycle. This process is then repeated until a complete offspring has been generated. This crossover operation is not applicable in our case without using some backtracking for the case that there is no valid edge that can be chosen taking into account the predefined partial ordering. This would make the crossover step very complex and hence is not applicable in our setting. Moreover the nearest neighbor strategy used for crossover does not pay off in many situations. A mixture of this greedy crossover with another more stable and faster crossover method could however be a good choice.

For this research we are adopting the main idea of the OX crossover operation, which is to retain the relative ordering of the cities to some extent. This is however not enough to generate well-ordered outcome. However we will see, that we can rotate the outcome of the OX operator such that it is well-ordered if the parent chromosomes are both well-ordered.

We define our crossover operator as follows:

Given two chromosomes $A = (a_i)_{i=0..n}$ *and* $B = (b_i)_{i=0..n}$*, we start by choosing two random crossover points* $0 \leq k < l \leq n$*. We construct the outcome* C *and* D *by setting* $c_i = b_{i+k}$ *for* $i = 0..l-k$ *and* $d_i = a_{i+k}$ *for* $i = 0..l-k$*. The remaining elements of* C *(resp.* D*) are filled in the order as they appear in* A *(resp.* B*) after the second crossover point (continuing at the beginning) and dropping those elements that are already in* C *(resp.* D*).*

An example for this crossover operation is given in figure2. It is now important to repair this crossover operator such that it completely respects the order relation. The following fact is easy to see:

For two well-ordered chromosomes $A = (a_i)_{i=0..n}$ and $B = (b_i)_{i=0..n}$ (i.e. $a_i \leq a_{i+1}$ and $b_i \leq b_{i+1}$ for $i = 0..n-1$ in the induced ordering) the outcome of the crossover operator admits a well-ordered rotation.

We are recombining everything in the same order as it appeared in the parent chromosomes. If we have to leave out some element, it is a simple application of transitivity of the partial ordering to conclude that everything is well-ordered except at the place where we continue from the beginning. We can now rotate until this single place is between the end and the beginning and get a well-ordered outcome.

Now that we have described the main ingredients of this algorithm there remains only one non-trivial step: Generate uniformly random well-ordered chromosomes for the initialization and mutation steps. This task is a bit tricky, if you want to do this with high performance. We decided to randomly generate a tour and check where the first contradiction to the partial ordering occurs and try to shuffle this element back. If for some fixed shuffles the length of the well-ordered subtour does not increase, we are generating a completely random new one. It is important to see, that if this type of generation is given an unsolvable problem, it will not terminate. Nevertheless the decision problem, whether there exists a well-ordered tour is difficult to solve. In essence we did not find a way to do this without exhaustively constructing the space of well-ordered chromosomes. But if we do this in a sensible time, we could directly solve the Partially Ordered Traveling Salesman Problem in this time-frame.

In practice our mechanism of finding a well-ordered chromosome has shown to work. One reason for this is, that we have only some few order classes and that most connections inside the same ordering class are of finite weight. Putting this in other words: The space of well-ordered chromosomes is - in the case of an airport - relatively big compared to the space of all tours and a random tour has a good chance of being well-ordered.

V. Evaluation

For the evaluation of our algorithms, we have implemented all this functionality into a productive navigation system modelling Munich airport. As described before, the navigation system is based on a big weighted, directed graph with a grid-structure of uniform high resolution. An instance of our problem can be defined and computed using a socket interface to a dedicated routing server. On this socket interface a list of coordinates (inside the navigable space), an order number defining the partial ordering and a label can be transmitted. The navigation server then computes an asymmetric distance map using n Dijkstra searches and starts with one of our algorithms above. It reports back the best ordering found and a status code. This status code can be used to distinguish between the following three cases:

- **OK:** The solution is correct
- **Too Complex:** We did not find enough well-ordered tours or the maximal computation time for the exhaustive search has exceeded
- **Unsolvable:** The present problem does not have a solution

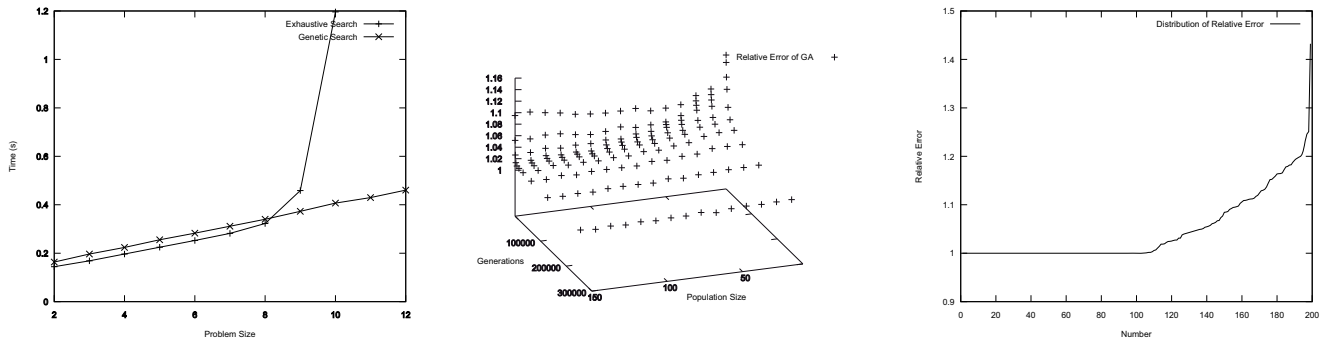
The case **Unsolvable** is only reported in case of exhaustive search being unable to find a solution. This can be the case, if the partial ordering contradicts an ordering of points-of-interests enforced by edges which do only go in one direction (e.g. passport control). An unsolvable problem is reported as **Too Complex** in the genetic case.

The graph in use consists of nearly 65.000 vertices interconnected by approximately 415.000 edges modelling the accessible terrain. This graph has been generated automatically from enhanced CAD drawings of Munich airport. For the comparison of the complete system speed, we created a set of 220 sample problems, 20 for each problem size ranging from 2 to 12 points of interest. These sample problems consist of randomly chosen vertices in the graph and we took care, that they are solvable.

The results of running all these benchmark problems in sequence on common desktop hard- and software (Intel Core2 Duo P8400 @ 2.26 GHz running Linux) is depicted in figure 3(a). The genetic algorithms was configured to use 50 chromosomes, 2000 generations and a mutation probability of 0.01. At first it seems to be unnatural, that the exhaustive search is faster than the genetic variant at all. The reason for this is, that the construction of the initial population for the genetic approach takes more time than the exhaustive solution for small instances. The number of loops involved in creating well-ordered chromosomes and running a fixed number of generations is larger than for exhaustive search. For instances with more than 9 points of interest, exhaustive search hits the one second limit (problems of size 9 were solved in an average time of 1.19s) while genetic search keeps below half a second for all problems.

From this we can conclude that for our specific navigation systems and graph we should use exhaustive search for all requests consisting of less than 10 points of interest. For larger instances we should switch to the genetic approximation. For general indoor navigation systems we can conclude that it makes sense to use exhaustive search as an algorithm for small instances, as it can be faster than genetic search and the well-known fast approximations of the Traveling Salesman Problem can not be applied.

As the genetic search is not correct and its performance (in terms of speed and quality) depends on many factors such as the number of chromosomes in the population, the mutation probability and the crossover operator, we performed another benchmark using a set of 200 solvable random problems, 25 for each problem size ranging from 4 to 11 points of interest. The average relative error of the solutions produced by genetic search is plotted against the two most important configuration variables defining the population size and the generation count. The results are depicted in figure 3(b). A relative error of 1 means, that the genetic algorithm was able to find the best solution. As you can clearly see, bigger populations and more generations lead to better results. In the average case, with the configuration of 2000 generations and 50 chromosomes



(a) Speed of Exhaustive Search compared to Genetic Search

(b) Relative error of GA

(c) Relative error distribution

Fig. 3. Results of the empirical evaluation using a benchmark set of 200 solvable random PoI-ordering problems

used for the first benchmark, we have an average relative error of 1.046, which means, that we find tours that are less than 5% longer than the optimal tour in average. Unfortunately, there were five problems (out of 200 problems) which solved to relative errors of inbetween 20% and 25% and one with a relative error of 45%. Note however, that the best known approximation algorithm for the metric Traveling Salesman Problem has a relative error bound of $\frac{3}{2}$ which translates to a relative error of below 50% which is not stronger than our result. The majority (52%) solved to the optimum and 82% solved to tours of length only 10% longer than the optimum tour. These results can be seen in the error histogram given in figure 3(c).

VI. Conclusion and Future Research

The most important result of this research is the negative result, that for indoor navigation an approximation of the Partially Ordered Traveling Salesman Problem using genetic search is not a good choice for sufficiently small instances. The genetic algorithm with our strongly order preserving crossover operation worked well but without self-adjustment of parameters it was slower than exhaustive search for small problems.

In the setting of a general indoor navigation system based on a graph structure the process of constructing the problem (e.g. constructing the complete graph from a general big navigation graph) takes a considerable amount of time for small instances and is linearly increasing (this is major reason for the linear increase of the genetic algorithm running time in figure 3(a)).

From this research, many questions arise which we did not cover here. The most important questions are: How can we estimate the quality of a genetic solution without knowing the best tour in advance. For a navigation system server the question arises, whether and how we can reuse partial knowledge created in previous genetic search instances. A different problem which is strongly related to this problem is the good selection of a point of interest out of a list of equivalent ones (e.g. choosing the "best" toilet to be inserted into a Partially Ordered Traveling Salesman Problem out of a list of available toilets).

Another question goes into the direction of context-awareness. A major strength of genetic techniques is the blindness. This means, that the properties of the graph are not used inside the search unless for evaluating a complete tour for calculating the fitness function. If one considers including routing over periodic or timetable services such as a bus, this is a very good property. However the crossover operation should take into account that changes in the estimated time of arrival at a partial solution can reduce the partial solution quality.

References

[1] C. Kray, G. Kortuem, and A. Krüger, "Adaptive navigation support with public displays," in *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*. New York, NY, USA: ACM, 2005, pp. 326–328.

[2] P. Ruppel, F. Gschwandtner, C. K. Schindhelm, and C. Linnhoff-Popien, "Indoor navigation on distributed stationary display systems," *Computer Software and Applications Conference, Annual International*, vol. 1, pp. 37–44, 2009.

[3] A. Butz, J. Baus, A. Krüger, and M. Lohse, "A hybrid indoor navigation system," in *IUI*, 2001, pp. 25–32.

[4] Y. Chen and H. Kobayashi, "Signal strength based indoor geolocation," in *Proceedings of IEEE International Conference on Communications (ICC '02)*, vol. 1, 2002, pp. 436 –439.

[5] F. Evennou and F. Marx, "Advanced integration of wifi and inertial navigation systems for indoor mobile positioning," *EURASIP J. Appl. Signal Process.*, vol. 2006, pp. 164–164, uary.

[6] P. Bahl and V. N. Padmanabhan, "Radar: an in-building rf-based user location and tracking system," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2. IEEE, 2000, pp. 775–784. [Online]. Available: http://dx.doi.org/10.1109/INFCOM.2000.832252

[7] M. Werner and M. Kessel, "Organisation of indoor navigation data from a data query perspective," *in press; accepted for publication in the conference proceedings of UpinLBS 2010*, 2010.

[8] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, pp. 85–103, 1972.

[9] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the ACM*, vol. 23, pp. 555–565, 1976.

[10] N. Chrisofides, "Worst-case analysis of a new heuristic for the traveling salesman problem," *Technical Report 388, Graduiate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh*, 1976.

[11] D. E. Goldberg, *Genetic Algorithms*. Addision Wesley Longman, Inc., 1989.

[12] K. Bryant, "Genetic algorithms and the traveling salesman problem," 2000.