Methods for Optimal Pedestrian Task Scheduling and Routing

Srihari Narasimhan
IPVS, Universität Stuttgart
Universitätsstr. 38
70569 Stuttgart, Germany

Hans-Joachim Bungartz
Institute for Informatics, TU München
Boltzmannstr. 3
85748 Munich, Germany

Abstract

Today, sensors and cameras are often used to monitor the movement and behavior of pedestrians, especially where there are a huge number of visitors. The classical usage of such devices, for example in a theme park, is to identify the queue size in front of each attraction and thereby to predict the time it takes until the visit can be completed. Work has been done in the past to use statistical data that resembles the data collected by such devices to simulate the pedestrian behavior. As a result, the congestions as well as the queue sizes at different times can be predicted. This work aims in using the data obtained from the simulation to optimally schedule a list of tasks to be executed as well as to find an optimal path between each destination. As an example, one might think of a scenario where a customer enters a theme park would wish to visit as many attractions as possible in the alloted time or a large clinic where a patient has to be routed through various departments such as registration, OP, X-Ray, ward, etc. The problem involves finding the optimal sequence of the tasks and determining the fastest path between the destinations, both combined. Since the data varies over time, the problem is time dependent or dynamic. In the past, several methods have been proposed to solve dynamic shortest path algorithms and scheduling problems. However, due to the stochastic nature of the available data, it is not necessary to find the best schedule and route that takes the minimum amount of time but, it is rather important to find an optimal solution in a short time. In this paper, we study and compare different combinatorial optimization methods and heuristics that can used to determine an optimal schedule.

Introduction

Pedestrians are one of the most commonly seen entities in day to day life. The most common activity of a pedestrian involves navigating his way to the desired destinations and executing a specific task at each destination, whatever the task may be. The pattern of the pedestrian movement along the paths as well as its behavior at the destination (participation in the queue) can be modeled and simulated (Narasimhan & Bungartz 2006; 2005). Once an analysis is made from the simulation, the statistics collected can be used to optimize the pedestrian activity. As an example, one might think of a commercial center where a customer has to execute a set of n tasks denoted as $T_j | j = 1, 2, \ldots, n$. This process not only involves waiting in the queues but also navigate between each destinations. The customer would obviously want to take the shortest route with least congestions and also complete the tasks in the shortest possible time. This paper proposes some of the methods that are used to optimize a visit based on the statistics derived from the pedestrian simulation.

The focus of this paper lies in using statistical data to estimate traffic congestions and waiting times, which can in turn be used to prepare an optimal plan for a new visit. Therefore, the first step involves a careful analysis of the pedestrian behavior in the given scenario. Since the geometry of the scenario makes an impact on the pedestrian model, the geometric aspects must also be taken into account when modeling pedestrian behavior. The pedestrian model must be strongly coupled with the geometry of the scenario such that the interactions with the geomety is also modeled. Once such a model is built, various scenarios can be simulated and an overview of congestions and waiting times across each scenario is recorded for different time periods. Once the simulation output data is available, the next step involves applying scheduling and optimization methods on the data obtained. Furthermore, with the help of sensors and modern communication systems to transmit spontaneous events to the model, it is possible to simulate the pedestrian behavior even more precisely, thereby improve the efficiency of the optimization methods proposed here.

This paper is structured as follows. The following section gives an overview of the pedestrian model implemented so far and the analysis made by simulation. The results generated from the simulation are then used for pedestrian navigation and task scheduling. Next, the problem of scheduling and routing by using the simulation data is discussed extensively. Following that section, different methods to optimize routing and scheduling problems are studied and analyzed. A comparison of the methods are then made with an example scenario. This paper concludes by presenting similar topics that exist elsewhere and also an outlook towards possible extensions to this work.

Pedestrian Simulation and Analysis

In order to analyze the pedestrian behavior, two different pedestrian processes must be modeled. First, the queues at the destination, where a customer waits in the queue before executing his task, and second, the movement of the pedestrian along the path. Both these processes are represented as a queuing system. In case of visiting a destination, the queue model consists of one or more service counters to serve the customer, and a queue where pedestrians arrive and wait if all service counters are busy. Similarly, the queuing model of a path consists of a number of service counters (the number is derived from the path capacity), and a queue for pedestrians to wait in case the path is congested. The queuing system is modeled using the discrete event simulation methodology.

Before any such queuing system can be modeled, it is first important to analyze the geometry in which the pedestrians or customers (hereafter also called as an entity) are modeled. The geometry provides spatial data to non-spatial context. For this purpose, the CAD model of the given scenario is analyzed. Geometrical data such as the paths along which the pedestrians move, possible destinations, and several other architectural parameters such as the type of the path (stairs, ramp, etc.), the capacity of the paths, properties of the destinations, capacity of each destination, etc., are extracted.

Geometric Structure and Modeling

Pathscan (Drexl 2003) is a flexible tool, which accepts a given CAD model, parses the geometry data, and identifies all possible paths and destinations where an entity can move. These paths are interconnected and exported as a graph structure. Furthermore, the flexibility of Pathscan tool also allows us to redefine the graph properties such that additional attributes can be incorporated into the graph. The resulting graph contains a minimized list of paths, list of possible destinations, and the required properties of the paths and the nodes. Fig.1 shows a snapshot of graph extraction using Pathscan tool. For test purposes, the CAD model of the new computer science building at the Universität Stuttgart is used (Giesecke, Stier, & Grumbein 2004).

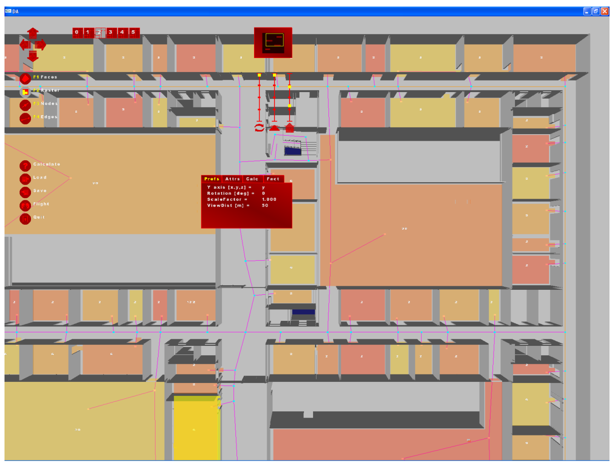

Figure 1: Snapshot of Pathscan tool used for graph extraction

Embedding of Queuing Systems into Geometry Model

Now, the necessary geometry parameters to model a queuing system are available. A queuing system consists of an arrival event (arrival of the entities to the queue) and the service process (task execution). The necessary input data are generated by using random variate functions. The available architectural data includes statistics on the type and usage of each destination. These statistics can be used to estimate the time an entity needs to execute a specific task. However, the behavior of an entity (or the customer) plays a major role in modeling a pedestrian simulation scenario. Therefore, several customer profiles are generated that contain parameters such as possible destinations an entity would visit, any specific timings to consider during the day, any restrictions (certain paths or destinations may not be accessible to all entities), etc. It is therefore possible to model the desired scenario by choosing from the appropriate customer profiles and specifying their distributions.

When modeling a queue at the destination, the arrival event is the time at which the entity leave the last path before entering a destination, and the service time is determined from both the customer profile and the destination statistics. Similarly, when modeling the movement along a path, the arrival event is the departure time of the entity from the previous path and the service time is derived from both the walking speed and the customer profile. Several statistics are available to estimate the walking speed of a pedestrian (Teknomo 2002; Knoblauch, Pietrucha, & Nitzburg 1996) for different situations (stairs, ramps, various pedestrian densities and congestions, uni- and bi-directional, etc.,). These statistics are used to generate the service time when modeling a path, based on the current path situation. The initial arrival is modeled as a Poisson process. The arrival time is hence negative exponentially distributed.

Now that a queuing system is available to model both destinations as well as paths in the given scenario, each path and the destination from the graph is replaced with a queuing system. Finally, the queuing systems are interconnected to form a queuing networks that span across the scenario. The model is now simulated for a specific scenario (decided on combining different customer profiles) and the status of the queuing systems are periodically recorded. Snapshots for each time unit (in this case, the time unit is in terms of minutes) are captured for each path and destination. Even though the data collected is discrete and not continuous, care is taken that fluctuations within two snapshots are as well recorded and adjusted with the existing snapshots. The resulting data is therefore very extensive. The result gives an overview of congestions and waiting lines that occur in the scenario during different times of the day. The output data contains either the sojourn time (waiting time and service time) or the queue size (or congestion in case of a path). The resulting data is then stored in form of a 2D Matrix (as shown in Table.1), which lists the congestions or waiting times for each path and node for several time stamps during the simulation.

Time	Edges 1, 2, ...	Nodes 1,2, ...
9:01	waiting time or congestion	...
9:02	...	...
...	...	...
17:00	...	...

Table 1: The resulting 2D Matrix when simulated for 8 hours (9am to 5pm)

Pedestrian Navigation Using a Graph

The tight coupling of the pedestrian model and the geometry context opens the possibility for a number of useful applications. One such application is the possibility of pedestrian navigation system. Work has been done in the past to identify the position of the entity as well as the choice of destinations to visit, thereby providing a framework for a pedestrian navigation (Narasimhan, Mundani, & Bungartz 2006). The graph data obtained from the CAD model is represented in an octree structure. The graph is partitioned in a 3D space recursively until each voxel of the octree contains utmost one node as illustrated in Fig.2.

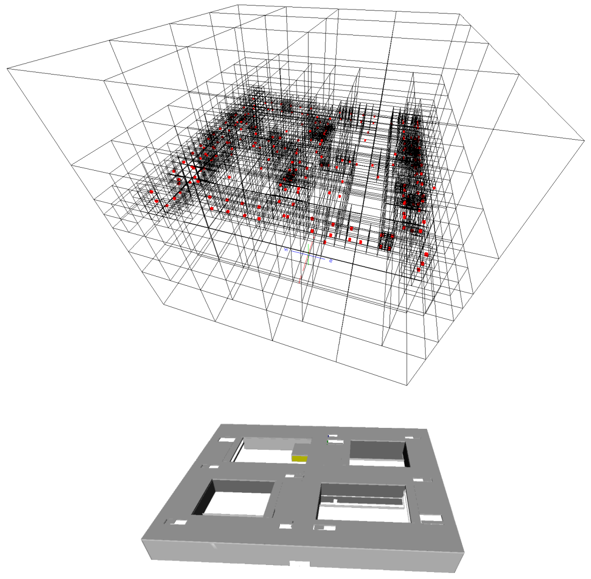

Figure 2: Octree model for the graph extracted from the CAD model

The method to partition the graph into an octree is described in (Mundani 2005). Each node of the graph is then associated with the respective co-ordinates (x, y, z) within the CAD model. The position of the customer in the octree is identified from the co-ordinates of the customer (identified from a mobile navigation device). The identified position is then translated to the corresponding node in the graph. Similarly, the choice of destinations can be listed by searching a specific region around the customer. The hierarchical storage of the graph structure enhances the search possibilities, specially for very large regions (e. g. a graph covering the whole city). Due to the hierarchy, the efficiency of the neighbor search improves to $\mathcal{O}(\log n)$ as compared to a linear search of $\mathcal{O}(n)$. Once the position of the entity and the choice of destinations are identified, a path search algorithm is performed to identify the shortest paths to each destination and sort the destination choices according to the distance from the actual customer position. With the use of simulation data obtained above, it is also possible to identify the paths with the least congestion as well as the destination with the shortest waiting line.

Pedestrian Visit Planning

The simulation of the pedestrian model so far gives an overview of the status of the paths and destinations at each simulation time stamp. It is now possible to determine the time it takes to move from the current location to a specific destination at time t and the expected waiting time at the destination at time $t + t(P_i)$ (where $t(P_i)$ denotes the time to walk along the i^{th} path. The waiting time at the queue and the actual service time at the service counter i are denoted by $t(WQ_i)$ and $t(ST_i)$. The time necessary to execute a task is denoted as $t(T_i) = t(P_i + WQ_i + ST_i)$.

The index i is used to represent the i^{th} task T_i. Since each task involves the transition to the task, waiting before the task and service of the task, the index i shall be used to also represent the path, queue and service station of the respective task. The index j is used to represent a set of tasks where $T_i \in T_j$

A task can denote any arbitrary activity that a pedestrian would perform in a scenario (shopping, working, relaxing). Using the statistical data obtained from the pedestrian simulation, we investigate the possibilities to schedule and navigate the tasks to be executed by a pedestrian.

Let us consider the scenario of a commercial building where several number of visitors are expected regularly. The scenario is assumed to contain several types of rooms and service stations (e.g. offices, shops, restaurants, etc.), and each service station has different properties. It is also assumed that statistics regarding pedestrian arrivals and their distributions are available as input models for pedestrian simulation. The scenario once simulated, gives us an overview of congestions and queue status in the commercial building throughout the simulation time period. Now, let us assume that a new pedestrian arrives in the commercial building with an intention to execute n set of tasks within a specific period of time $t_S \leq t_{\max}$ (for example, before closing time) or the least possible time. For experimental purpose, we assume that the aim is to minimize the time it takes to execute n tasks. Considering the congestions along the path and possible waiting times at the destination, we determine a schedule for the pedestrian visit such that $S = \min\{\sum_{i=1}^{n} t(T_i)\}$. Let list of tasks be $T_j | j = 1, 2, \ldots, n$. The visitor starts his visit at $source$, visits n destinations and traverses to the $sink$. Therefore, the set of paths, denoted as $P_j | j = 1, 2, \dots, n, n+1$, consists of n paths to each destination and the last path to the $sink$ (see figure 3).

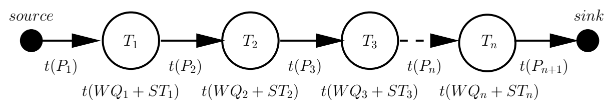

Figure 3: Set of tasks to be executed by a pedestrian

The pedestrian visit planning problem consists of two components namely, scheduling the tasks and routing between each task. In the example scenario considered here, the pedestrian would wish to obtain a feasible schedule either in advance, e.g. through a web-based interface, or via some hand-held devices. In either cases, it is important compute a feasible itinerary within a short span of time. In this section, we present the various techniques that were used to solve these two components.

Pedestrian Routing: Fastest Path Problem

The task scheduling problem involves both sequencing of the tasks and movement of the pedestrian between the tasks. To reach a destination node, the pedestrian must find his way from his current position to the destination node. A path must be found such that the pedestrian is able to walk easily without congestions and reach his destination in the shortest possible time. The path search algorithm makes use of the congestion data obtained from the pedestrian simulation. Even though the edge value is dynamic, the pedestrian simulation records the edge data as discrete and not as continuous values. A generalized static version of the Dijkstra's algorithm can be used to calculate the fastest path between two points in a discrete dynamic graph (Chabini 1998; Dean 1999). This algorithm works in the same way as the static algorithm and has a complexity of $\mathcal{O}(m + n \cdot \log(n))$. However, in reality, the computational needs and the running time for a discrete dynamic shortest-path algorithm is high. This is because of the construction and use of a time-expanded network (the edges of the graph should contain all discrete data). Also in a one-to-all shortest path problem, such data network is necessary even though the actual number of destinations is much lesser than the actual number of nodes present in the model (30% in the experimental model used for this work). Also due to the stochastic nature of the simulation, the edge values are more an estimate and not exact values.

Alternatively, we use a much simpler algorithm to compute the fastest-path without a time-expanded network by carefully manipulating the graph with certain stochastic parameters. The fastest path is determined as follows. Initially, the weights of the edges of the graph is replaced with the time it takes to cross the path (depending on the average walking speed, type of the path and the length of the path). The shortest-path algorithm now gives us the fastest path between two points. Now, any congestions that occur along this path will reduce the total time needed to reach the destination. Experiments have been made and it was found out that the congestions increase or decrease gradually and continuously. In the test model used here, it was found that the longest walk (the farthest points) do not take more than about 10 minutes to walk. Therefore, we consider the traffic that occur along the paths during the next 10 minutes and stochastically determine a mean value such that the edge weights of the graph is replaced with the stochastic values. This method has a complexity of $\mathcal{O}(n)$ and saves the trouble of creating a time-expanded graph.

Pedestrian Task Scheduling: Task Sequencing

We now have a generalized path search method that identifies an optimal path between two nodes. The next stage involves finding a schedule (sequencing the tasks) such that the total time taken to complete the visit is within the alloted time limit. The schedule S involves sequencing of tasks $T_j | j = 1, 2, \dots, n$ and we assume that the time taken to complete each task includes the $t(P_i)$, $t(WQ_i)$ and $t(ST_i)$.

Brute-Force Search Theoretically, the only accurate solution will be to use a brute-force search. A brute-force approach identifies the optimal sequence of tasks by determining the time taken by all possible sequences $S_j | j = 1, 2, \dots, n!$ and identifying the sequence which takes the least time. Even though, the brute-force search is a straight forward method and gives us the accurate solution from the available data, the computing resources required are enormous. The brute-force approach for scheduling the tasks have a computational complexity of $\mathcal{O}(n!)$. Even for small n (e.g. $n = 10$ or $n! = 3,628,800$ comparisons), modern computer requires many hours of computation. However, for very small problems (e.g. $n = 4$ takes about 0.2 s), brute-force search can be used, specially when a large scenario can be decomposed into smaller domains. Since a pedestrian requests for a schedule on-the-fly, brute-force search is inappropriate for pedestrian task scheduling.

Due to the stochastic nature of the simulation data, there is no guarantee that the schedule determined from a brute-force search is indeed the fastest and the most optimal schedule. This is because, the simulation attempts to capture the congestions and waiting time situations in a macroscopic level and a precise behavior modeling of each pedestrian is therefore not possible

it is not possible to predict the occurrence of a slow moving group at a specific time and path that would decrease the walking speed.

Also, if considering such negligible details, a small change could create an impact to the final schedule. That is, if it is assumed that the pedestrian would start at time t, arrive at the destination at $t + \Delta t$, start the task execution at $t + \Delta t + \Delta t_2$, an unexpected change in the Δ value will then render the schedule invalid and require a new computation.

Greedy Heuristic We use the greedy heuristic to optimize the pedestrian schedule. In a greedy heuristic, the algorithm follows the problem solving meta-heuristic of making the locally optimum choice at each stage with a hope of finding the global optimum. There are three different ways to perform greedy heuristic for the task scheduling problem and they are listed as follows. An analysis with an example is made later.

***Greedy Path Search*:** In the first approach, the algorithm always looks for the task that can be reached fastest. The generic path search algorithm is used to measure the time it takes to walk between the starting point and all destinations. The destination which can be reached fastest is chosen. The greedy path search algorithm has a complexity of $\mathcal{O}(n)$ for the first time and $\mathcal{O}(n-1), \mathcal{O}(n-2), \ldots$ and so on for consecutive searches. The major disadvantage with this method is that there exists no control in the sequencing of the tasks. Therefore, the pedestrian might choose a destination that is most crowded currently.

***Greedy Sequencing*:** In contrast to the greedy path search method, the next approach chooses the next task that can be executed the fastest. The service time ST_i of any task is generally consistent at any time of execution. So, the algorithm identifies the current waiting time at each destination, and chooses the task, which requires the least waiting time. The computational complexity is same as the greedy path search method. In a greedy sequencing algorithm, the location of the destination (distance from the current position) is ignored. Therefore, a pedestrian might have to walk long distances between each task.

***Greedy Routing and Scheduling*:** This approach essentially combines both greedy path search and greedy sequencing methods to choose the appropriate destination by counting the length of the path as well as the waiting time at the destination. In this method, the sum of waiting time and walking time the time $t(P_i + WQ_i)$ is calculated for each destination and the destination with the least walking and waiting time is chosen.

Since the future states of the destinations are not taken into account, the greedy approach does not always give us the appropriate solution. Also, the use of such methods are decided based on the actual scenario. For example, if there are no waiting times, a greedy path search method is appropriate.

Stochastic Optimization

The greedy search mentioned above is a local search technique. The drawback of a greedy search can be avoided by iteratively determining a schedule and checking if the new schedule is faster. However, a greedy search always chooses the fastest task and any number of iterations will result in the same schedule. Therefore, a probabilistic approach is made to determine if a faster schedule exists. We use simulated annealing to determine the optimal pedestrian schedule.

In a simulated annealing method, many iterations are made to determine the optimal solution. At each step, some neighbor sequence S' of the current sequence S is chosen using probabilistic methods. S' is accepted as an optimal solution if it turns out to be better than S. The probabilities are chosen such that the system finally tends to result in a faster schedule. The process is iterated until the an optimized solution is identified (with a termination condition). The probabilistic methods used to determine S' is not completely at random. Instead, the profiles and the input data that are used for the simulation are analyzed and the times when the congestion is at minimum, are identified. The probabilistic methods used to determine S' are coupled with these data such that the number of iterations are minimized

Counting on Constraints and Preconditions

In typical shop scheduling problems, the job characteristics β are considered for scheduling. Similarly, the pedestrians arriving at a scenario may have different priorities and preferences (referred to as constraints) such as precedence relation for tasks $T_i \prec T_j$, path preferences (avoid stairs or escalators), queue priorities (preemptive service, queue balking and reneging), and a specific time of execution (execute task T_i at time δ). Such constraints are defined within the pedestrian profiles and are incorporated when scheduling the tasks.

Apart from pedestrian constraints, a careful analysis of the pedestrian simulation data can give us specific situations such as times when certain paths are heavily congested or times when certain destination nodes are free of any activities. Such preconditions can be used together with the scheduling methods discussed above. In general cases, the congestions in a path or destination changes continuously and gradually. Rapid fluctuations are seldom seen along these paths. However, due to stationary processes, such fluctuations along the paths or nodes are possible. A stationary process is a stochastic process whose probability distribution at a fixed time or position is the same for all times or positions. As a result, parameters such as the mean and variance also do not change over time or position. However, the time before and after a stationary process experiences a rapid fluctuation near the geographical location of the process. A classical example of a stationary process is a cinema hall. In case of a cinema hall, the probability distribution between the between the start and end of the process (a show) remains constant. Therefore, pedestrians arrive shortly before the start of the process and the inter-arrival times reduces rapidly before the process. This process causes congestions along the paths in the vicinity of the destination (cinema hall). Similar fluctuation is seen once the process is complete is over (see Fig.4). From the simulation data, it is possible to identify stationary processes and thereby identify the congested paths at a given point of time.

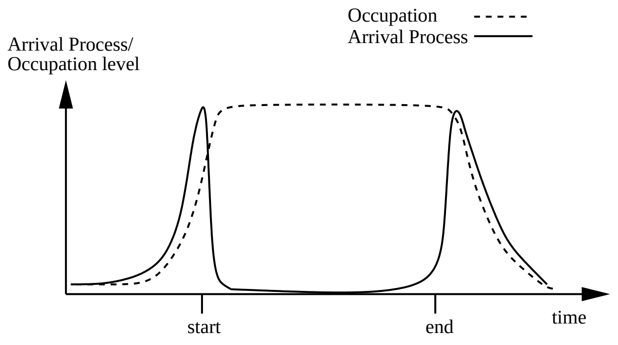

Figure 4: The arrival pattern before and after a stationary process

Similarly, long waiting times that occur in the scenario can be detected from the simulation data. These observations can be used to ensure that the task is scheduled such that the lengthy queues are avoided. In one of the test models simulated here, the following parameters were considered.

- About 3000 customers visit the building, each with a specific list of tasks to execute (derived from the customer profile).
- Utmost 1000 customers are present in the building at any given point of time.
- A restaurant, with a capacity for 40 people is chosen for analysis, and simulations are run to study its occupation during different times of the day.
- The probability of the time of visit is determined from the customer profile. In general, the probability of visit is high during noon.
- It is assumed that the entities wait in the queue as long as they are served. Also, it does not deter further additions (queue reneging and balking are deactivated).
- The simulation is run for a period of 8 hours (9am to 5pm).

The plot in Fig.5 shows the number of customers (entities in the queue and the entities being served) during different times of the day. From the plot, it can be seen that the restaurant is overloaded shortly after noon and takes more than an hour to fall below the threshold line. Such information can be used as a precondition when planning a schedule. For instance, if the schedule includes visit to a restaurant, the visit can be fixed shortly before noon.

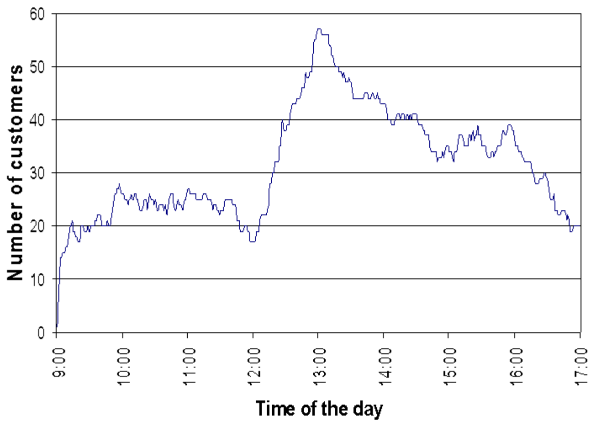

Figure 5: Number of visitors for a sample restaurant (capacity=40) recorded during the simulation

A *tabu* list of the above mentioned preconditions and constraints can be made and the list can be used by the scheduling algorithms to improve the efficiency of the algorithm. The implementation details are not discussed in this paper.

Nearest Neighbor Search

The nearest neighbor search attempts to find the next closest destination (in terms of distance) to visit. Nearest neighbor search is similar to greedy path search method excepting that the nearest neighbor search is a static algorithm. In practical situations, it can be observed the customer tends to stay in the region and finish the tasks located within the region before moving to the next region. The classical example is a theme park where the attractions are grouped and sorted according to the theme and a visitor normally makes a tour from the first area to the next closest and so on. The nearest neighbor search makes an attempt to localize the search for optimal path by splitting the graph into several subgraphs (regions), optimizing the plan for each subgraph and finding the shortest connection between each subgraph. The scheduling methods explained earlier are used to schedule the tasks optimally within the subgraph.

Domain Decomposition using Octrees Octrees are very efficient method to partition and store geometry data hierarchically. Octrees for very large graphs (greater than level 12) can be generated on the fly (Mundani *et al.* 2003). For particularly large graphs (such as a whole campus network), octrees are used to partition the graph. The octants of the octrees are mapped with each other through a Lebesgue curve. This leaves each domain to be processed individually and the interconnection between the domains are connected using space filling curves. Furthermore, there is a possibility to parallelize the optimization algorithm. The analysis of the nearest neighbor search and decomposition of larger graphs are not within the scope of this paper and hence not discussed here.

Results and Analysis

An hypothetical pedestrian scenario, which occurs within the sample geometric model used for test cases, is modeled and simulated. As mentioned earlier, the new building erected for the computer science department will be used as a test environment here. The dimensions of the building measure approximately $67m \times 79m \times 16m$ ($l \times b \times h$) including basement. The building consists of a total of four floors. The graph extracted using the Pathscan tool consists of a total of 600 nodes and 540 edges. The path types (stairs, ramps, private areas) and the path capacities are defined within the graph data. In the test scenario used here, the customer has a choice of about 250 destinations to visit. There are about 15 different classes of destinations and each class differs with the expected service time of a task to be executed. All 250 destinations fall under these 15 categories. The capacity of each destination is also defined.

Different pedestrian profiles and their distributions were created. The pedestrian profile chosen was a mix of regular employees (tend to stay longer and restrict the visit to office rooms) and visitors (tend to stay shorter and visit the public area). Differences exist in the set of tasks to be executed and the personal behavior (walking speed, disabilities and restricted access if any). Each pedestrian executes on an average of about 10 tasks and the average service times are set in the range of 5-20 minutes per task (office stay takes longer). During the simulation run time, about 5000 pedestrians visited the scenario and a maximum of about 500 pedestrians were present in the building at any given point of time. The scenario was simulated for a period of 8 hours (9am to 5pm) and the simulation took 117 seconds to complete on a Pentium 4 processor with 3 GHz speed and 1 GB main memory.

Once the simulation is complete, statistical data of transition times along each path and waiting times at each destination, similar to the layout shown in table 1, were collected. Now, a pedestrian arrives in the scenario with a set of tasks to execute. For test reasons, a random list of tasks were generated. The different methods shown in this paper were used to determine the optimal sequence in which the tasks must be executed (sequence involves both transition along the path and execution of the task). It was assumed that there exists no specific constraints. It was also assumed that there are no available preconditions to use with the scheduling algorithm.

About 10 destinations, where a task can be executed, were generated at random. Different problem sizes were chosen (the first 4,6,8 and 10 tasks) and the scheduling methods were implemented for each problem set. Initially, the time taken to execute the tasks in the generated sequence is computed. Then different methods were used to determine the corresponding solutions and a comparison was made with the original sequence. Figure.6 shows the results from the methods used in this thesis. The above defined scenario had generally less waiting times due to the short service times and availability of many rooms.

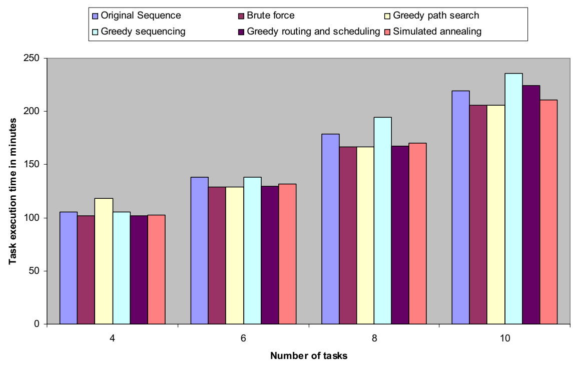

Figure 6: The comparison of the results obtained from the various methods used to schedule the tasks

The greedy sequencing approach did not differentiate with marginal differences in waiting times. Therefore, greedy sequencing method is found to be inappropriate in pedestrian task scheduling. The results from the greedy path search method was often close to the best result. However, there existed inconsistencies in several other experiments performed. Of all the greedy search methods, the method that combined the path search and waiting time, performed most consistently. But, for large problem sizes, inconsistencies were noticed. The simulated annealing method produced very consistent results for different scenarios and problem sizes. The results were close to the optimal solution but never the same as the optimal solution.

The computing times were measured for each method and different problem sizes. The greedy search had the least complexity and therefore took the least amount of time. The computing time for all greedy search methods were linear with respect to the problem size (see figure.7). Due the search mechanism, the greedy approach takes the least time.

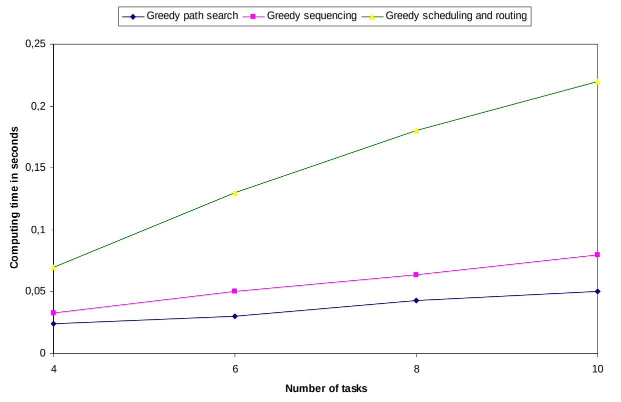

Figure 7: Comparison of the computing speed of the greedy methods used to schedule the tasks

The brute force search always yields the optimal solution but there is a factorial increase in the running time (see in figure.8). For a problem size of $n = 10$, the brute force search lasted about 18.5 hours in comparison with just 0.2 seconds for $n = 4$. The computing time of the simulated annealing method depends on the number of iterations performed. In this example, between 100 and 230 iterations were performed for different problem sizes.

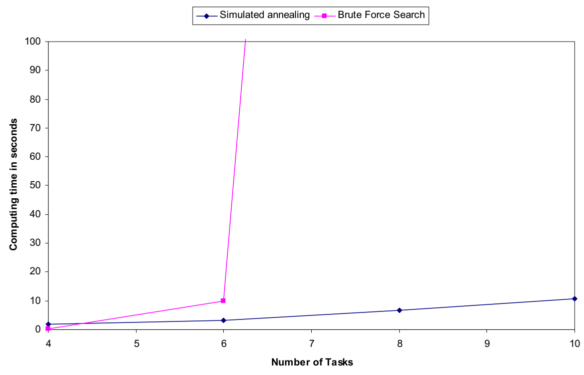

Figure 8: Computing speed comparison of brute force and simulated annealing

Conclusion and Outlook

So far, various methods have been proposed to optimize a pedestrian visit. The aim of this work is not to impose a control on every pedestrian such that entire system functions optimally, but rather to provide a service to the customer by planning the visit – either a priori before the visit or in real-time through mobile navigation devices. Also, the idea is to compute the task sequence by considering the expected congestions (crowded restaurant during noon) rather than considering all waiting times and path conditions to determine the best solution. The integration of the pedestrian simulation into the geometry context is the key to enhance the system into an intelligent navigation system. This work is purely driven by the pedestrian simulation and the statistics produced out of it. It must be noted that the quality of pedestrian simulation purely depends on the accuracy of input modeling. Care has been taken to carefully model and validate the statistical data used for the simulation. However, spontaneous changes such as emergency situations, accidents, breakdown, etc., are not computed through simulation. With the use of sensors and modern communication devices to transmit the actual status of congestions, the simulation can be altered dynamically. The NeXuS project (Spatial World Models for Mobile Context-Aware Applications) at the Universität Stuttgart (Hohl *et al.* 1999) for instance enhances mobility by inducing spatial-aware application. In this project, work has also been done in using communication devices to transmit information between each other and therby constantly update the service database. By integrating the pedestrian simulation model and the task scheduling strategies with the context model database, updated information can continuously be received, thereby leading to a more intelligent task scheduling and pedestrian navigation. This however is still an open avenue for research.

References

Brucker, P. 1998. *Scheduling Algorithms*. Springer-Verlag, 2nd edition.

Chabini, I. 1998. Discrete Dynamic Shortest Path Problems in Transportation Applications. *Transportation Research Records* 1645:170–175.

Dean, B. C. 1999. Continuous-Time Dynamic Shortest Path Algorithms. Master's thesis, Massachusetts Institute of Technology.

Drexl, T. 2003. Entwicklung intelligenter Pfadsuchsysteme für Architekturmodelle am Beispiel eines Kiosksystems (Info-Point) für die FMI in Garching. Master's thesis, Technische Universität München, Garching bei München, Germany.

Giesecke, S.; Stier, M.; and Grumbein, S. 2004. Pfadsuche in Architekturmodellen und Stereoprojektion. Software Praktikum, Universität Stuttgart.

Hohl, F.; Kubach, U.; Leonhardi, A.; Rothermel, K.; and Schwehm, M. 1999. Next Century Challenges: Nexus – An Open Global Infrastructure for Spatial-Aware Applications. In *Proc. of the fifth Annual Intl. Conf. on Mobile Computing and Networking (MobiCom'99)*.

Knoblauch, R.; Pietrucha, M.; and Nitzburg, M. 1996. Field Studies of Pedestrian Walking Speed and Start-up Time. *Transportation Research Record* 1538.

Mundani, R.-P.; Bungartz, H.-J.; Rank, E.; Romberg, R.; and Niggl, A. 2003. Efficient Algorithms for Octree-Based Geometric Modelling. In *Proc. of the Ninth Int. Conf. on Civil and Structural Engineering Computing. Civil-Comp Press*.

Mundani, R.-P. 2005. *Hierarchische Geometriemodelle zur Einbettung verteilter Simulationsaufgaben*. Ph.D. Dissertation, Faculty of Computer Science, Electrical Engineering and Information Technology, Universität Stuttgart.

Narasimhan, S., and Bungartz, H.-J. 2005. Congestion-Aware Optimization of Pedestrian Paths. In Hülsemann, Frank and Kowarschik, Markus and Rüde, Ulrich., ed., *Proceedings of the 18th Symposium Simulationstechnique ASIM 2005*, 242 – 247. Erlangen, Germany: Erlangen: SCS Publishing House.

Narasimhan, S., and Bungartz, H.-J. 2006. A Framework for A Graph- and Queuing System-Based Pedestrian Simulation. In H. R. Arabnia., ed., *Proceedings of the 2006 International Conference on Modeling, Simulation and Visualization Methods (MSV'06)*, 87 – 93. Las Vegas, USA: CSREA Press.

Narasimhan, S.; Mundani, R.-P.; and Bungartz, H.-J. 2006. An Octree- and A Graph-Based Approach to Support Location Aware Navigation Services. In H. R. Arabnia., ed., *Proceedings of the 2006 International Conference on Pervasive Computing and Systems (PSV'06)*, 24 – 30. Las Vegas, USA: CSREA Press.

Reinelt, G. 1994. *The Traveling Salesman: Computational Solutions for TSP Applications*. Lecture Notes in Computer Science. Springer.

Spall, J. C. 2003. *Introduction to Stochastic Search and Optimization*. Wiley-VCH.

Teknomo, K. 2002. *Microscopic Pedestrian Flow Characteristics: Development of an Image Processing Data Collection and Simulation Model*. Ph.D. Dissertation, Tohoku University, Japan.