Deadline Driven Behavior in Intelligent Virtual Environments

Djura S. Smits

Front cover illustration by Djura Smits.

Djura S. Smits
Student number: 5619807
Master's thesis
Credits: 42 EC

Master's in Artificial Intelligence
University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisors
Arnoud Visser
Philip Kerbusch

Intelligent Systems Lab
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

Change

December 20th, 2014

This research has been published:

[1] Djura Smits, Arnoud Visser, and S C.A. Groen, Frans. Modeling pedestrians in an airport scenario with a time-augmented petri net. In Sukhan Lee, Hyungsuck Cho, Kwang-Joon Yoon, and Jangmyung Lee, editors, *Intelligent Autonomous Systems 12*, volume 194 of *Advances in Intelligent Systems and Computing*, pages 543–551. Springer Berlin Heidelberg, 2013.

Todo list

0.1 Acknowledgements

I would like to thank my supervisors Philip Kerbush from TNO, and Arnoud Visser from the UvA. Furthermore, I would like to thank Ernst Bovenkamp for providing the data and videos from Rotterdam airport. I would also like to thank my friends and family for the support, and my classmates, many of whom I've enjoyed working with for many years.

Contents

Chapter 1

Introduction

Artificial intelligence is the branch of computer science that aims to create or simulate intelligence in machines. The range of methods to attempt this is endless. One of the many sub-fields in AI is aimed at reproducing human behavior. The problem of recreating human behavior can be tackled from many different perspectives. For example, one could attempt to get an understanding of the workings of the human mind, and try to imitate this in a computer program. Another approach would be to treat the human brain as a black box, and mainly focus on recreating external behavior. In this approach it does not matter whether the inner workings of this computer program are simular to that of an actual human being. The disadvantage of this approach is that it does not help you learn how the human mind works. However, when used for practical purposes, approaching the problem like this is favorable, since it is more likely to run real-time and take up less resources. In our research, we are attempting to recreate a specific kind of human behavior. Namely, the kind of behavior that pedestrians would show in a public environment like an airport, where it is very important to not lose sight of time.

For the last decade or so, there has been an increased interest in reinforcing security in

Figure 1.1: People going about their everyday business

public environments. The recent advances in technology have increased the feasibility of many different methods. An important approach in this area of interest is the automatic detection of suspicious behavior from camera images. The approaches in this area can vary, but one important aspect that they have in common, is the necessity of camera footage to be able to test the approaches. Since it can be extremely tedious to collect and annotate this data it would be very rewarding to be able to generate annotated testing data automatically. We would like to focus on the behavior of non-suspicious people, be-

Figure 1.2: The main hall of Rotterdam Airport.

cause we assume these usually greatly outnumber terrorists or other suspicious individuals. People with suspicious behavior could then later be scripted and added in by hand. In other words, we would like to be able to quickly generate crowds of people doing normal, everyday behavior.

Research focussing on simulating pedestrians can tackle the problem from different angles. For some purposes such as simulating panic situations, it is important to view the pedestrians as a large crowd. These types of simulations are mainly concerned with with how people find the exit out of a building and how they move through doors and other openings. Other researches focus more on treating the pedestrians as individuals, and crafting a specific behavior for every individual. In between these methods there is a whole spectrum of approaches, some of which focus on larger groups of people and general movement, others focus more on smaller groups or individuals and treat more sophisticated behavior.

TNO & SIMOBS

This research was done as part of an internship at TNO, the Netherlands Organisation for Applied Scientific Research. TNO is an independent organization founded to enable business and government to apply knowledge. More information can be found at `http://www.tno.nl`.

SIMOBS is a plugin for the military simulation toolkit VR-forces (`http://www.mak.com/products/simulate/vr-forces.html` which is a tool for generating and executing battlefield scenarios. SIMOBS is developed by Philip Kerbush at the Modelling, Simulation and Gaming department at TNO to quickly generate inhabitants of scenario to add realism to the otherwise mostly empty simulated areas. SIMOBS allows users to quickly draw residential areas on a map to indicate where the simulated people should live. It is also possible to indicate where these people should go to work. The behavior of these inhabitants is determined by *Daily Motion Patterns*, which specify where certain types of inhabitants need to go at specified times. SIMOBS formed the starting point of our research.

Our research is intended to help inhabit simulated scenarios developed at the *Modelling, Simulation and Gaming* department at TNO and is an extension of their SIMOBS

(simulated observables) project. The department *Intelligent Imaging* has been so kind to provide both camera footage and manually tracked locations of pedestrians at the main hall at Rotterdam airport (pictured in figure 1.2). This footage has also been used to research how to automatically spot suspicious behavior. That is why we attempt to find an approach to suit an airport type scenario. Our results can then be compared to the available footage. Because of the nature of the footage and the desire of TNO to quickly generate large quantities of non-suspicious people in everyday environments, we chose to tackle the problem of pedestrian simulation with an approach that lies somewhere in the middle of the spectrum of approaches. The behavior of the simulated people should not be designed individually per person, but our approach should result in more varied behavior than only the general movement of crowds through openings. We would like to be able to specify behaviors by indicating them in the environment, instead of specifying them per individual. For example, when a food stand is present in the area, we would like to be able to indicate that the agents that are placed in the neighborhood of that stand are able to do certain food-buying behavior. In our research, we specified a toilet behavior, that causes two agents to go to the area near the toilets, where then one person enters the bathroom and the other one waits untill the first one comes back again. An additional requirement is that we would like to indicate a certain *deadline* for an agent which is the time at which its goal is not reachable any more. This deadline-drivenness should result in roughly two types of behavior, *hurried* and *relaxed*. When the deadline approaches, less and less actions are likely to be done, since actions that take much time would result in not reaching the goal in time.
Keeping these requirements in mind we created a system where we design what we call *situations* in an environment. These are areas where certain behavior is appropriate. The design of the behavior is facilitated by our choice to model them in *Petri nets*, which is a mathematical modelling language used to model distributed systems. Moreover, the choice of the pedestrians to execute certain behaviors should be weighted by how much time they still have left to reach their goal, which is to check in, in our airport-type scenario. We will mimick behaviors that we found at Rotterdam airport, and then validate them using the real-life footage. We will also show that Petri nets are particularly suitable for designing the type of human behavior we need, especially when a situation requires interaction between multiple individuals.

1.1 Research Question

The question we are going to base this research around is the following:

> How can an intelligent virtual environment for simulated pedestrians be extended to deal with time-restricted destinations?

Some of these terms may need some clarification:

- *Intelligent Virtual Environment*:
  IVE is a broad term, but in this particular case we mean that a large portion of the intelligence needed for the pedestrians to walk around is placed in the environment, instead of in the pedestrians walking in it. For example, the information about interacting with a food stand could be placed in the design of a certain "food stand object" that is placed somewhere in the environment, instead of in the design of the pedestrians. There are different ways to construct an IVE, which is something we have to look at as well.

- *Time-restricted destinations*:
  We would like to create a framework that is able to deal with departure hall-like situations. That means that pedestrians will have a destination (e.g. a train, or an airplane, etc.) that will be available for a limited amount of time (until it departs). It can also be used to create simulations with a pattern that is more realistic when run for a long period of time (such as a whole day). Furthermore, in many situations, only the time is known when the person has to arrive at his destination. In those cases it is more intuitive to define the time of arrival, instead of determining at what time someone has to leave his starting point. In the rest of the article, we will refer to these time restrictions as *deadlines*.

We are going to solve this question by trying to answer the following subquestions:

- *To what extent can pedestrians be simulated realisticly?*
  If it were feasible to model the complete human brain, we would probably get the most lifelike behavior. However, since this is not possible, we have to simplify the model somehow. Models can be made in varying levels of complexity. Most often, a higher complexity means slower performance. That is why we have to think about getting the right balance between realism and performance. We also have to decide how we define realism. Do we take the inner model into account, or do we purely compare the resulting behavior to real people in similar situations?

- *How do we let the pedestrians make decisions based on time left to reach the destination?*
  In situations such as departure halls, people have a destination (e.g. airplane, train) that is only available for a limited amount of time. Some actions might take a very short amount of time, some may need more. How do we let these pedestrians decide between the different options?

- *Is it possible to have emergent behavior based on time restrictions?*
  Ideally, our model should lead to behavior that we have not expressly implemented in our system. In the context of time restrictions, we would like to see the behavior of the pedestrians to look hurried or relaxed based on the amount of time they have left to reach their goal.

- *Is it possible to quickly generate these virtual pedestrians without much tweaking for each environment?*
  We aim at creating pedestrians that can be used in many different environments without much additional scripting. Is it possible to do this and still have varied behavior between environments?

In the following chapters we will first discuss the relevant literature that helped us develop our method that will answer these questions. Consequently, we will describe our system for simulating pedestrians in detail. After that we will test the implementation of our method both qualitatively (by comparing the trajectories of simulated pedestrians to real pedestrians), and quantitatively (by following the frequencies of hurried and relaxed behavior over time). Lastly we will discuss our findings.

Chapter 2

Related Work

When studying existing methods for modeling pedestrian behavior, the amount of available literature is quite overwhelming. This is not surprising as a large part of AI research focuses on the imitation of human behavior, and pedestrian behavior can imply many different kinds of behavior. However, not all means of simulating pedestrian behavior are developed for the same purpose. What we would like to have in our system, is a method to design behavior in an environment in the same way that other objects in the environment are designed. That is, it should be possible to spatially place the behaviors in the environment. As a consequence, we have to divide our search into two categories, namely methods that describe how to design individual behaviors and methods for designing behavior in relation to others, or to the environment.

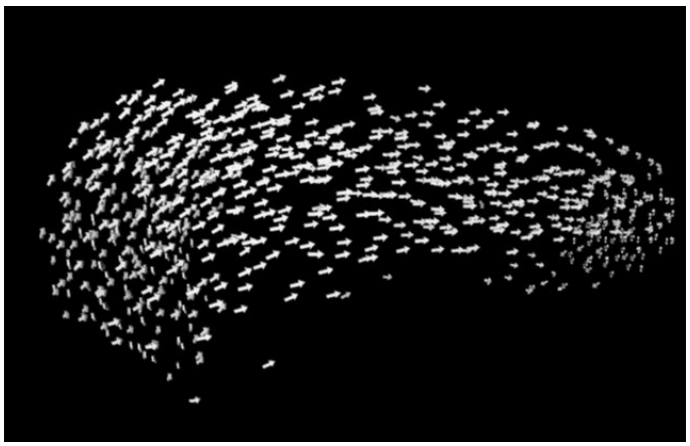


Figure 2.1: An image from an implementation of Reynolds boids model. This particular implementation can be found at `http://vvvv.org/contribution/boids-3d`.

2.1 Group Interaction

The research of interaction in groups started with Reynolds' *boids* [21] , where members of the group were seen as particles that exert both repulsive and attracting forces on the other members of the flock, depending on the distance to one another, and forces inwards from the outer contour of the flock, in order to remain in a certain shape. This behavior is driven by three simple rules, namely *separation* (avoiding crowding local flockmates), *cohesion* (move toward center of mass of local flockmates), and *alignment* (steering towards average

heading of local flockmates). Even though the rules the individual flock members adhere to are simple, it led to emergent group behavior that imitated actual group behavior in nature. However, this flocking behavior is more suitable for modelling behavior of more primitive animals such as fish, and will not give a very plausible result when it is used to model humans. Because humans usually act in a way more complicated than a flock. However, many researches have built upon this idea of modelling groups of people by viewing the members as particles. This approach is particularly suitable for simulating large crowds. Other researches divide the crowd up in smaller groups, or treat the pedestrians on an individual level. This angle asks for different approaches.

2.1.1 Large Crowds

In the collection of researches concerned with simulating large crowds, many models can be found that focus on crowds in panic situations. An important of simulating panic situations can be found in the article of Helbing, Farkas, and Vicsek [11]. In their model, people exert a repulsive interaction force to stay away from each other, an additional *body force* slowing to counteract body compression, and a *sliding friction force* when a pedestrian comes in contact with another pedestrian or the wall. The resulting change in velocity over time can be expressed in accelleration equation 2.1.

$$m_i \frac{d\boldsymbol{v}_i}{dt} = m_i \frac{\boldsymbol{v}_i^0(\boldsymbol{t})\boldsymbol{e}_i^0(\boldsymbol{t}) - \boldsymbol{v}_i(t)}{\tau_i} + \sum_{j(\neq i)} \boldsymbol{f}_{ij} + \sum_w \boldsymbol{f}_{iw} \tag{2.1}$$

Where pedestrians $1 \leq i \leq N$ of mass m_i want to move with a certain desired speed v_i^0 in a certain direction $\boldsymbol{e}_i^0$. They tend to adapt their instantanious velocity $\boldsymbol{v}_i(t)$ within time interval τ_i. The interaction force $\boldsymbol{f}_{ij}$ is added to simulate the tendency of the pedestrians to want to keep their distance from others. Interaction force $\boldsymbol{f}_{iw}$ makes the pedestrians tend to keep away from the walls. This model can lead to several effects known to occur in real panic situations. Focussing on large crowds though does not always mean one is unconcerned with the individual agents. One example of large crowd simulation while taking individuality into account is the research by Pelechano and Badler [18]. They successfully show that Helbings model can be combined with a high-level wayfinding model that bases decisions of individuals on their mental map of the building. Individuals are even divided into three categories: *trained leaders*, who have complete knowledge about the building the simulation takes place, *untrained leaders*, who handle stress well, help others and will explore the building, and *untrained non-leaders*. In later research they extend the psychological model even more by using the PMFServ software system [19]. PMFServ is a model based on established psychological principles [23]. Braun et al. [5] also used the model of Helbing as a basis, but introduced additional features for creating group behaviors, such as family members, dependence level, altruism level, and desired speed of the agent.

While these methods give a good insight into the movements in those particular panic situations, they are less suitable for experiments simulation non-threatening situations. Only in those few moments of panic, or when crowds are very dense, do these models represent a crowd realistically. What we are looking for, is a framework that gives realistic behavior over longer periods of time. Pelechano, Allbeck and Badler [17] have simulated high-density crowds for normal situations. They base the movement of the crowds on a simple wayfinding algorithm and a number of different psychological (impatience, panic, personality attributes, etc.) and physiological traits(e.g. locomotion and energy level). Furthermore, the agent is given perception and will react to objects and other pedestrians

in the nearby space.
Bayazit, Lien and Amato approached the subject of crowd simulation in a very different way [3]. Instead of letting an entity such as one pedestrian or a group do the navigation on the fly, global roadmaps are used. In this method, a map is generated beforehand defining where the pedestrians can walk. During simulation, the pedestrians are given a goal location, and will then explore the paths defined by the map, based on which direction exercises the highest force on the pedestrians. The pedestrians can update this map in real-time indicating if a path is favorable or not when trying to get to the goal. When two paths exert an equal amount of force on the pedestrians, they will split up and both paths will be explored simultaneously.
However, global roadmaps are not the only way an environment can be defined with respect to behavior. Various methods have been developed that use a combination of multi-agent techniques and cellular automata [8][10]. Here, the behavior stems from a combination of basic multi-agent systems, combined with information about how the pedestrians should be distributed over a grid. This grid follows the rules typical to cellular automata, where the value of a single square in the grid at time t depends on the value of the surrounding squares in the grid at $t-1$. Figure 2.2 shows how Hamagami et al. structured their implementation of this idea.

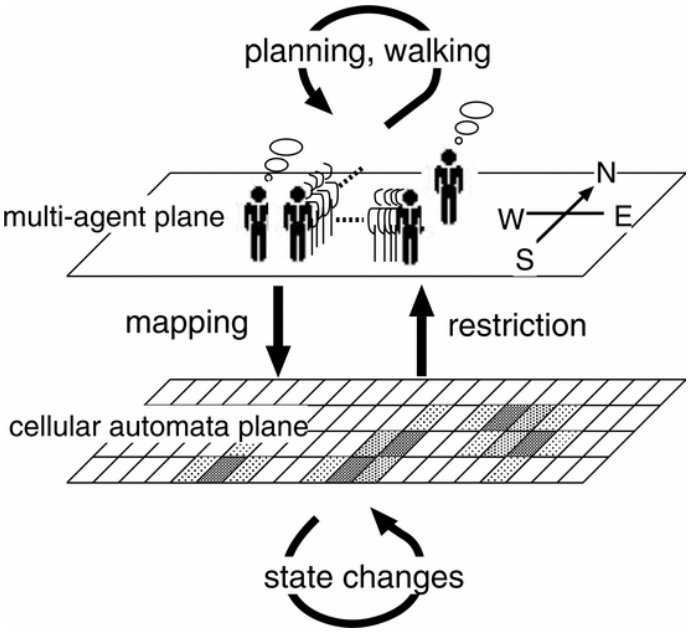


Figure 2.2: Crowd simulation model using cellular automata by Hamagami et al. [10]

.

2.1.2 Smaller Groups and Individual Approaches

The previously mentioned approaches focus on movements of large groups of people. This might generate realistic effects when the crowds are very dense, but when the pedestrians are more sparsely scattered in the environment, these models will not suffice. That is why there have been many researches focusing more on crowds as a collection of smaller groups. In an urban environment, a lot of pedestrians move around together with a couple of other pedestrians and very few move around on their own. An important step in this direction has been made by Li, Jeng and Chang [25], who proposed a leader-follower model, in which

one person in a group gets the role of *leader*, who has the job to decide on the destination and has to plan the path. This leader will exert an attractive force on the *followers*, who will continuously follow this leader around. Hostetler and Kearny chose an approach in which all members of the group cast an equal vote on which direction to head in [12]. The walkways have been modelled as ribbons to define the geometry of the surface, and which create a conduit that channels pedestrian traffic into parallel streams. Every member of a group casts a vote on which way to turn and how to adjust the speed based on a discretized action space. The group will then collectively follow the action that has the highest vote. Peters, Ennis and O'Sullivan decided to have a more direct approach to the formation of groups [20]. They studied a large video corpus of prototypical walking areas and concluded groups always occur in certain formations. Subsequently, they designed a number of formations in a *formation template* that represent discrete formations that the pedestrian groups may adopt, such as walking completely abreast, or in a staggered formation (see figure 2.3). The distance between the group members is defined by a cohesion matrix, which describes the cohesion between every two members in the group. Another factor contributing to the distance between each member is the minimum frontal aspect the formation can have, which describes the width of the formations.

Figure 2.3: (a) Examples of formations where people walk completely abreast or staggered. (b) Group dynamics. From Peters et al. [20].

Until now, we have only seen models that deal with crowds in terms of walking behavior. Interpersonal relationships may have been somewhat defined, but were only expressed through spacial positions. Bcheiraz and Thalmann have attempted to express these interpersonal dynamics through a set of animations that express a persons mood through body language [4].

2.1.3 Mixed Approaches

Farenc et al. even devised a hierarchical framework that incorporates a number of different models managing the crowd on different levels (crowd behavior, group specification, group behavior, and individual behavior). A framework such as this is very suitable for incorporating multiple crowd behavior techniques. While a single method of the previously mentioned techniques might not generate a satisfactory result, a hierarchic combination of several methods might be able to do the job.

2.2 Interaction with the Environment

As previously mentioned, most researches with the focus on crowds as a group of individuals or viewed globally do not address the problem of how to interact with the environment except for some collision detection. While this results in realistic behavior in panic situations, it is less suitable for scenarios when people have more freedom to choose their actions. The methods for crowd simulation we've looked at in the previous sections that don't focus on emergency scenarios are mostly concerned with interactions between people, and less about interaction with the environment. Because it is important for our purposes to have pedestrians with behavior tailored to the specific environment we need to look further. The obvious solution to more interaction with the environment is to extend the knowledge of the pedestrians with instructions about how to deal with the objects that are present in the simulation. This has been successfully done for instance by Shao and Terzopoulos [22]. They used an extensive psychological model to determine the behavior of the individuals. This led to a simulation in which the behavior looks very realistic, even when one individual is followed for a long time. The downside of this method is that a different behavior needs to be programmed for the pedestrians for every simulation environment. This is unfortunately quite time-consuming.
It would be easier to generate the virtual human agents if the environment would automatically decide for the agents what interactions are possible and appropriate. The first step towards this focus was made by Kallmann and Thalmann who introduced the principle of *smart objects* [14]. In their simulations, the information about interaction with objects is stored in the objects themselves, instead of in the human agents. This way, additional behavior can be added to a pedestrian by simply placing a new object in the environment. The object also keeps track about how many agents can interact with it at the same time and if the interaction should be the same for all agents. For instance, an elevator modeled as a smart object will make the first agent interacting with it press the button, but not the next agents that approach this object. By using smart objects, the internal model of the pedestrians can be kept very simple, because they do not need to remember specific information about how to interact with the objects. Furthermore, this means that the pedestrians do not have to be specifically designed for the current simulation environment, because the environment will tell them how to act. In the most basic approach to smart objects, the agents lose all their autonomy when they approach a smart object. When these agents interact with an object , they become "slaves" of these objects, as it were, because these objects provide the animations that the agents need to execute.

A lot of research has been built upon the idea of smart objects. For instance, Kallmann, de Sevin and Thalmann have extended this model to have agents that have their own motivations and needs [13]. This model uses five main motivation types: eat, drink,

Figure 2.4: An human agent interacting with a tv in the videogame *The Sims 4*.

rest, work, and go to toilet. These motivations control the action through a hierarchical decision graph. Information about which objects fullfill these different needs are added to the specification of the objects. The fact that the general idea of smart objects has been implemented in the very successful video game series "The Sims" (see a screenshot in figure 2.4) might be seen as an illustration of its effectiveness. By modelling the in-game human behavior through smart objects, the game developers have been able to add expansion after expansion of new items (with new required behavior) with great ease.The initial implementation of the agents doesn't need to be modified, but their behavior will still be enriched.

Another slightly different approach to modelling interaction in the environment is the use of a situation based control structure (Sung, Gleicher and Chenny [24]). A situation is an area in the environment that requires the pedestrians to act in a certain way. The behavior of a pedestrian is described by a finite state machine in which a state is defined as follows:

$$s = \{t, \mathbf{p}, \theta, \mathbf{a}, \mathbf{s}^-\}$$

In which t is the time, $\mathbf{p}$ is the current position in two dimensional space, θ is the orientation, a is an action, and s^- is a list of previous states. By "action" they mean a particular animation clip that has to be played at this state. Note that this definition of finite state machines differs from the standard definition. More information about finite state machines can be found in section 2.3. Situations extend the pedestrians' finite state machine with situation-specific actions. The probability distribution of the actions the pedestrian can take is multiplied with the probability distribution given by the situation. Situations are divided into two categories: *spatial* situations for stationary objects or areas, and *non-spatial* situations to describe concepts such as friendship with another pedestrian. In figure 2.5 an example is shown of an area around a crosswalk where the behavior is designed with the situations based control structure. The numbered areas are (1) a "crossing street" situation, (2) a "traffic sign situation and (3) an "in a hurry" situation where the street is crossed without a crosswalk.

2.3 Behavior Modeling Languages

For some Pedestrian simulatinos, such as Helbings, it is sufficient to have one mathematical equation that can describe all the movement. However, when one wants to be more

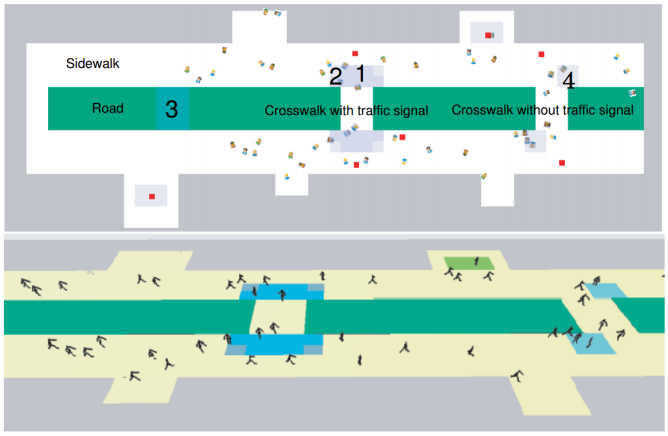


Figure 2.5: A screenshot of pedestrian simulation based on the situation based control structure by Sung et al.

specific about how an agent interacts with an environment, it might be advantageous to choose a modeling language to express behavior. In this thesis, two different languages are important: Finite State Machines and Petri nets. They will be described in detail below.

Finite State Machines

Finite-state machines (FSMs) are behavioral models that are composed of a number of states associated to transitions [9]. They can be used to describe a sequence of actions. A finite state machine moves from state to state by doing sets of actions associated with certain transitions. Both *deterministic* and *non-deterministic* finite state machines are defined by a 5-tuple $(Q, \Sigma, \delta, q_o, F)$ consisting of:

- Q: a finite set of states
- Σ: a finite input alphabet of symbols
- $\delta : Q \times \Sigma \rightarrow Q$: a transition function
- $q_0 \in Q$ a start state
- $F \subseteq Q$a set of accept states

The difference between deterministic and non-deterministic FSMs is that in non-deterministic FSM, a transition function can give multiple states for one combination of a state and input symbol. Finite state machines are widely used in a variety of applications such as electronic design automation, but also for language parsing. Many variations on FSMs exist for a variety of purposes such as deterministic and non-deterministic. The latter can also be seen as a representation of a *Markov chain*. Finite state machines have been used extensively as tools for behavior modeling. The previously mentioned situations based control structure for example uses finite state machines to describe the behavior of a pedestrians as is illustrated in figure 2.6. Here, extensible states are attached and detached in the course of a simulation, depending on where a pedestrian is located.

2.3.1 Petri nets

Petri nets are a mathematical modeling language used for the description of distributed systems. A petri net is a bipartite graph consisting of two types of nodes: places and

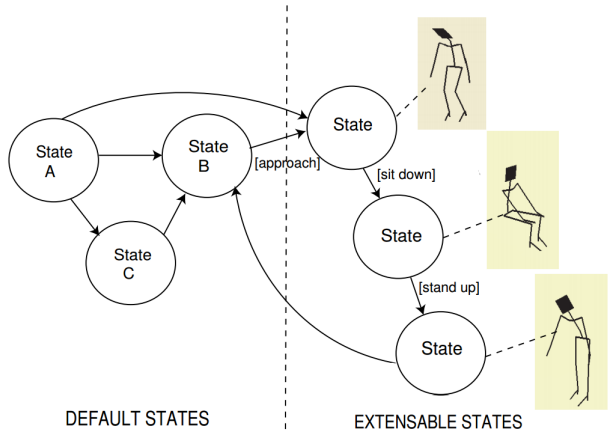


Figure 2.6: An example of the use of finite state machines in Sungs research.

transitions. These nodes are connected by directed arcs. An arc can run from either a place to a transition, or from a transistion node to a place, but never from a place to a place, or between two transitions. Activity in a Petri net is expressed by by the movement of tokens from place to place, through transitions. Input arcs (from place to transition) denote which places need to contain tokens in order to enable the transition. When a transition is enabled, it consumes the tokens from the input places, and produces tokens in the place indicated by the output arc. The basic Petri nets that we describe in this thesis are called place-transition nets, or p/t nets [6]. P/t nets can be described by a four-tuple:

$$PN = (P, T, F, M_0)$$

which comprises of

- a set of places $P = (p_1, p_2, ..., p_m)$,
- a set of transitions $T = (t_1, t_2, ..., t_m)$,
- a set of directed arcs, $F \subseteq (S \cup T) \times (S \cup T)$, satisfying $F \cap (S \times S) = F \cap (T \times T) = \emptyset$
- an initial marking $M_0 = (m_{01}.m_{02}, \ldots, m_{0m})$.

Graphically, places are indicated with circles, transitions as rectangles and arcs as arrows. All places and transitions are elements of N. Furthermore, it is important to know the definitions of *pre-sets* and *post-sets*:
For an element x of N, its *pre-set* ${}^\bullet x$ is defined by

$${}^\bullet x = \{y \in N \mid (y, x) \in F_N\}$$

and its post-set is defined by

$$x^\bullet = \{y \in N \mid (x, y) \in F_N\}$$

P/t nets have been extended in many ways in order to accommodate many different functionalities. An example of this are *stochastic Petri nets*. In this extension, there are two types of transitions: *immediate* and *timed* transitions. The Stochastic Petri net (SPN) model can be described as a six-tuple:

$$SPN = (P, T, F, M_0, \Lambda) \tag{2.2}$$

where (P, T, I, O, M_0) is the marked untimed PN underlying the SPN, and $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is an array of (possibly marking dependent) firing rates associated with transitions.

Immediate transitions always have priority over timed transitions, and the likelihood of firing a timed transition is dependent on a parameter called the *firing rate* of the transition. This rate indicates the firing delay of the timed transition. This firing rate may be marking-dependent, so it should be written as $\lambda_i(M_j)$. The average firing delay of a transition t_i in marking M_j is $[\lambda_i(M_j)]_{-1}$. Immediate transitions fire in zero time once they are enabled, while timed transitions fire after a random, exponentially distributed enabling time.

Another way an aspect of time is incorporated are *Timed Petri nets*[26]. Here, Petri nets are accompanied by a firing time function that assigns a positive rational number to each transition. This number indicates how much time the firing of a transition costs. When a transition is fired, it consumes and holds its input tokens, until this time has past, and the firing is terminated when the tokens have been produced in the transition's output places.

Köhler et al. show us an example of how Petri nets can be used to model elaborate social situations [15]. They use a high level variant of Petri nets called *reference nets* to model the decision making processes in universities. Reference nets are built up of multiple layers of Petri nets that represent both emergent aggegration of processes on a micro level and more high-level rules and structures on a macro level. In their multi-agent system, they make the distinction between *system nets* and *object nets*, where tokens of a system net correspond to Petri nets on a lower level, called object nets. In this structure of *nets within nets* these lower level Petri nets move around in the higher level system nets. They manage to apply this modelling language to the decision problem of recruiting new employees in universities.

2.3.2 Finite State Machines vs. Petri nets

We think Sungs situations framework, is very promising, but for our purposes it would benefit from using Petri nets instead of finite state machines. There are serveral properties of Petri nets that FSMs lack that would be very beneficial for us. We will discuss those properties below, but first it might be important to ask, is it difficult to substitute FSMs in a system with Petri nets? Petri nets and FSMs don't only look similar, but FSMs can even be considered a subclass of Petri nets. A finite state machine can namely be described as a Petri net that contains only one token. According to Desel et al., a (finite) state machine can be defined as a Petri net that "has no *branching transitions*, and hence does not allow for *synchronization*". The lack of the possibility to describe synchronous behavior is one of the reasons that we prefer Petri nets over FSMs. This means that with Petri nets it would be possible to model behavior that makes an agent do several tasks at once. For instance, it could be possible to model our pedestrians' Petri nets so that a token would represent an arm or a leg. That way, a pedestrian could execute multiple basic tasks independently. More importantly, Petri nets could be used to model complex interactions between several agents. It would never be possible to achieve this kind of behavior with a finite state machine, unless a state would be described for every possible combination of activities. An example of how interaction between agents could be modeled can be found in 2.7.

However, this is not the only advantage Petri nets have over finite state automata. The aspect of time is also very important for our research. There are several ways with which a sense of time can be incorporated into Petri nets, while are harder to find for finite state machines. Timed automata do exist [2], but more approaches exist for Petri nets.

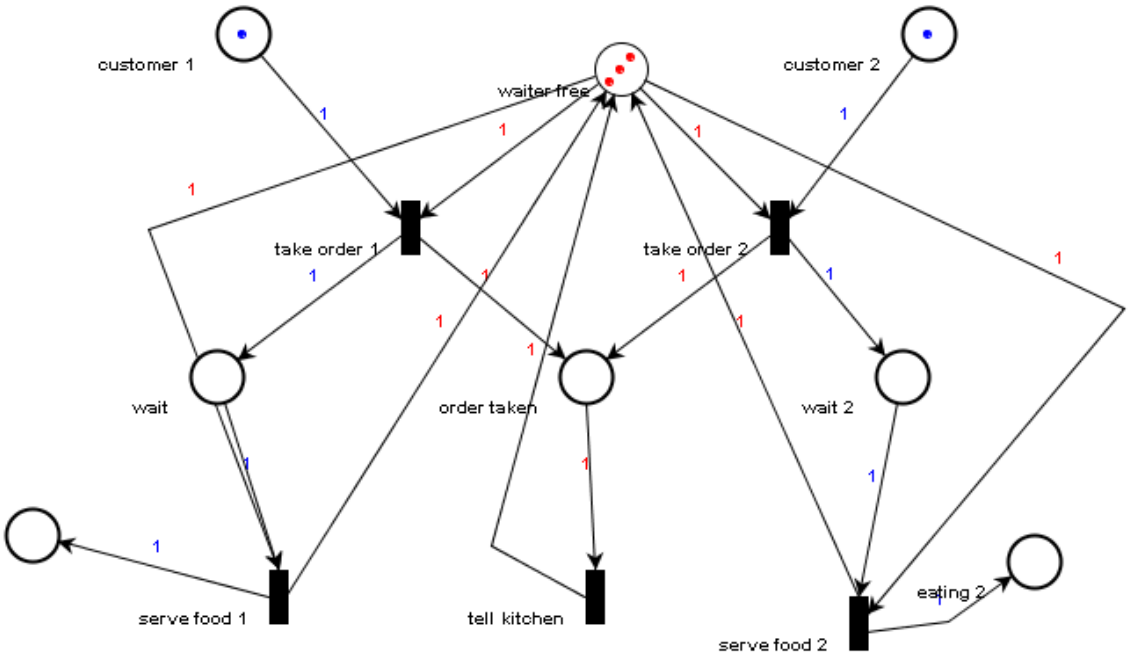


Figure 2.7: An example of a behavioral model of a restaurant expressed in a Petri net

Another essential issue which has to be implemented in the method we are going to choose, is non-determinism, which is one of the basic properties of a Petri net. When multiple transitions are enabled at the same time, any of them may fire. Finite state machines allow for non-determinism as well, so this doesn't have to be the deciding property for choosing between both, but together with the other advantages, it seems that Petri nets suit our needs the best. Especially the issue of synchronicity makes us certain that we should use Petri nets

2.4 Time Planning

The essential extension to the situations framework that is proposed in this thesis adds an element of time to the system. This is needed to enable the system to deal with daily motion patterns. An important element without which the system cannot succeed is knowledge about how long actions are going to take. Only when this information is known to the agent (or system) it can be decided whether taking a certain action will result exceeding the deadline for the goal. Both stochastic Petri nets and timed Petri nets deal with a part of the time issue, but when we specifically want our agents adapt their behavior to *deadlines*, these methods still fall too short. Unfortunately, not much research has been done regarding the planning of actions or behavior under time restrictions. More often the restrictions of processing time are more important than the restriction of the executing time of behavior. That is why we decided to design our own method that can

deal with this problem.

2.5 Which Techniques Can We Use?

We have seen many different techniques for many different purposes. We can immediately eliminate most global approaches to crowds, since for most environments, we do not want the pedestrians to move as one big mass, all pedestrians heading into the same direction. Using purely attracting and repulsive forces will also generally not work when the pedestrians are too far apart, so we have another reason to rule out most global techniques. Of course, the global roadmaps approach does enable the pedestrians to split up and walk in different directions, but using this system on its own will not create very realistic behavior, since it does not have the capability to deal with interpersonal relations, and interaction with objects.
The techniques focusing on smaller groups do show some potential. Ideally, people should interact with each other based on a variety of social relationships such as "parent" and "friend". However, this will require much configuration beforehand, and may increase processing time for each decision the pedestrians have to make. In this case, configuration may not be the issue, since these relationships can be defined and then used for many different simulations, but processing time is a larger problem, increasing with the number of pedestrians present in the simulation.
For our purposes though, we rule out the techniques that don't put more importance in the environment except for collision detection, because with those techniques it will be impossible for a designer of the environment to have any influence on the behavior of the pedestrians, which is an important requirement for us. Of all the possible existing techniques, the "situations" approach of Sung et al. seems the most appealing. However, it is not directly usable for our purposes. One downside to the situations approach (at least how it is described in the paper) is that state transitions are mainly low-level animations. However, it should be possible to adapt the state transitions to correspond to higher-level actions, such as change of goal. Path planning should then be delegated to another module. In this way, it is very likely that the state representation can be reduced from $s = \{t, \mathbf{p}, \theta, \mathbf{s}^-\}$ to $s = \{t, \mathbf{p}, \theta\}$. The s^- was used to remember which actions were taken previously, so that the pedestrian remembers which way it was walking. When the actual walking is delegated to another module, it seems unnecessary to keep s^-.

Chapter 3

Method

In the upcoming sections will be described how we decided to design our system. We decided to base our model around Sungs situations framework [24]. We chose this framework because it allows for probabilistic behavior, and puts emphasis on defining behavior through the environment. This framework will have to be extended to deal with time restrictions. However, we won't work with finite state machines, as Sung et al. do, since these offer too little functionality to enable us to work with time restrictions. Furthermore, FSMs are not suitable for describing interactions between multiple agents. That is why we model behavior in a more advanced way in our system. Instead of modelling the system with finite state machines, we describe the behavior of the pedestrians with Petri nets. The toolkit we will use for this is *Platform Independent Petri net Editor 2* (PIPE2). `http://pipe2.sourceforge.net/` Further down below we describe how we decided to use this toolkit.

In short, this chapter will describe how we combined the situations framework with Petri nets instead of finite state machines, and added a mechanism with which we compute a "utility" measure for behavior.

In our method, a number of concepts will be mentioned that have a very specific meaning within our research. These concepts might sometimes be easily confused with each other.

- *Scene*
  A scene or scenario is the area where the simulation takes place. For example, in our experiments the scene is Rotterdam airport.

- *Situation*
  In our research, the term situation is used to indicate an area in a scene where a specific behavior is applicable. For example, a situation could be the area around a drinks machine. A situation as a number of properties: the area in the environment where it is placed, the behavior, and whether an instance of the behavior is shared with other agents so they can interact or is exclusive to one agent.

- *Behavior*
  A behavior is a group of consecutive actions. In our framework a situation (as previously described) always has a accompanying behavior. The behavior for the drinks machine situation would be going up to the machine and buying a drink.

- *Action*
  Actions are the building blocks in which a behavior is described.

- *Petri net*
  Petri nets are the modelling language in which we describe our behaviors.

- *Steps or Timesteps*
  Lengths of time are expressed in timesteps. The reason for this is that this is that our multiagent simulation uses this measure of simulation time. If a different simulation would have been used, the system could have been easily adapted to deal with the time indications that would be used there.

It is important to keep in mind the difference between behaviors and actions when covered in our thesis. These two terms are not interchangeable. With *actions* we indicate movement of a pedestrian that is described in a single transition in a Petri net. *Behaviors* on the other hand, indicate the whole set of actions that are encompassed within a whole situation. Another way a behavior can be viewed is the whole sequence of actions that take place when a token of a pedestrians Petri net leaves the base place until it comes back again.
For example, a pedestrian might have entered the region of the toilet situation. A new *behavior*, namely the Petri net corresponding to the toilet situation, is then attached to the pedestrian. A pedestrian might then decide to fire the transition that produces a token in this newly attached Petri net, thus making the pedestrian execute the behavior corresponding to the toilet situation. This Petri net consists of several transitions, corresponding to various *actions*, such as walking to the toilet.
Now that some of the terminology has been clarified, we can move on to how our system has been designed.

3.1 Petri nets

Our method of designing behavior for large groups of pedestrians is largely based on the situation framework by Sung et al., but the finite state machines are replaced by Petri nets. Our Petri nets are based on place/transition Petri nets (p/t-nets)[6]. However, we did make some changes to the way one transition is selected from a set of enabled transitions. These changes were necessary to be able to incorporate a way of dealing with time pressure. In the following subsections we will introduce the tool we use for designing Petri nets, and how we incorporate these Petri nets in the overall structure of our system.

3.1.1 PIPE2

PIPE2 is a tool written in Java to create and analyse Petri nets. We chose this toolkit for a number of reasons. First of all, PIPE2 is written in Java, which makes it easier to integrate with VR-Forces, because VR-Forces is made to communicate with Java. Secondly, this toolkit promised a number of Petri net extensions, the most important of which is the capability to create *Generalized Stochastic Petri nets*. Generalized stochastic Petri nets is an extension that adds timed transitions to standard p/t nets. Unfortunately, at a later time it seemed that the generalized stochastic petrinets that we planned to use did not behave as described in the specifications, so that is one of the reasons we stick to use modified, basic, place/transition Petri nets. Another reason that extend regular p/t Petri nets is that it allows us to have more control over frequencies of behavior. PIPE2 also provides a clear and simple GUI with which Petri nets can be designed (see figure 3.1). This is very important because the design of the behavior in our method should be fast and easy. It should require minimal effort to place large amounts of agents in a environment.

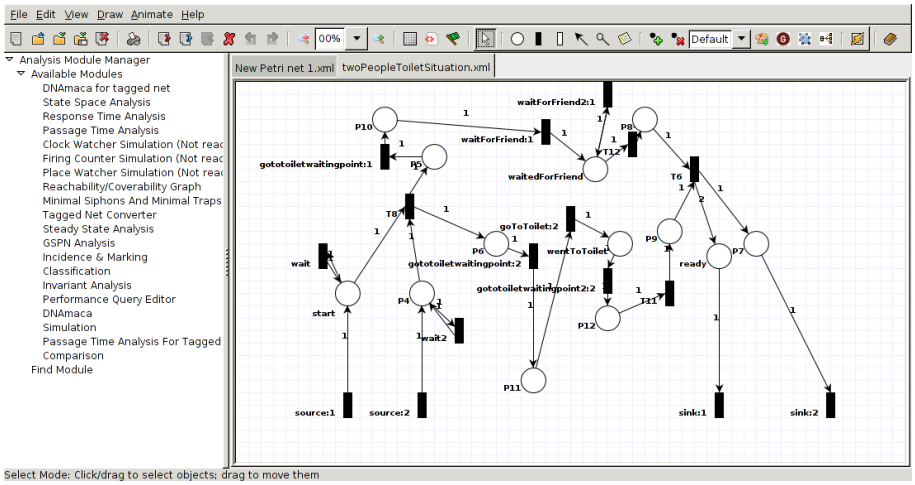


Figure 3.1: The PIPE2 interface.

3.1.2 Separating the Petri Nets of Situations and Pedestrians

Because a pedestrian should only be able to do behavior when he is in the associated situation, we should keep the Petri nets belonging to the pedestrian and the nets that describe the behavior for a situation separate, like in figure 3.2. In the middle of this figure we can see the base place, to which all Petri nets eventually loop back, except for the gotogoal behavior. The large circles in this example indicate the situation Petri nets. These have to attach to and detach from the pedestrian Petri net as he walks in and out of the associated situation. We do this through specially labeled *source* and *sink* transitions. A *source* transition is a transition $t \in T$ of which the pre-set ${}^\bullet t = \emptyset$, whereas a *sink* transition is a transition $t \in T$ of which the post-set $t^\bullet = \emptyset$. An example of a place with a source and sink transition attached can be found in figure 3.3
When a pedestrian walks into a certain situation, two new transitions are attached to the base place. One of these transitions has an arc pointed from itself to the base place, this is a source transition. The other transition, which has an arc pointed the other way around is a sink transition. Because a source transition does not consume any tokens, but only produces them, it is always enabled and can fire at any time. Sink transitions only consume tokens.

In our system, we generate source and sink transitions when a pedestrian Petri net is attached to a situation Petri net, and treat these differently from other transitions. These new transitions will be associated to sink and source transitions of the situation Petri net. This way, when a newly attached sink transition is fired in a pedestrian Petri net, the associated source transition of the situation Petri net will be found and fired as well. This way tokens can be "transported" from pedestrian Petri nets to situation Petri nets. Situation Petri nets have a fixed amount of special source and sink transitions, so no new ones are generated through the simulation. These sources and sinks indicate where a pedestrian Petri net should "attach" its sink and source transition when the pedestrian enters the associated situation. Whether a source or sink transition in a situation Petri net is meant to be attached to a pedestrian Petri net can be seen in the name of these transitions. These sources and sinks always come in pairs with the names **source:n** and **sink:n**. There is another way in which these source and sink transitions are treated differently. Regular source transitions are always enabled. Therefore, if these transitions would be treated as other transitions, attaching a situation to a pedestrian would create a

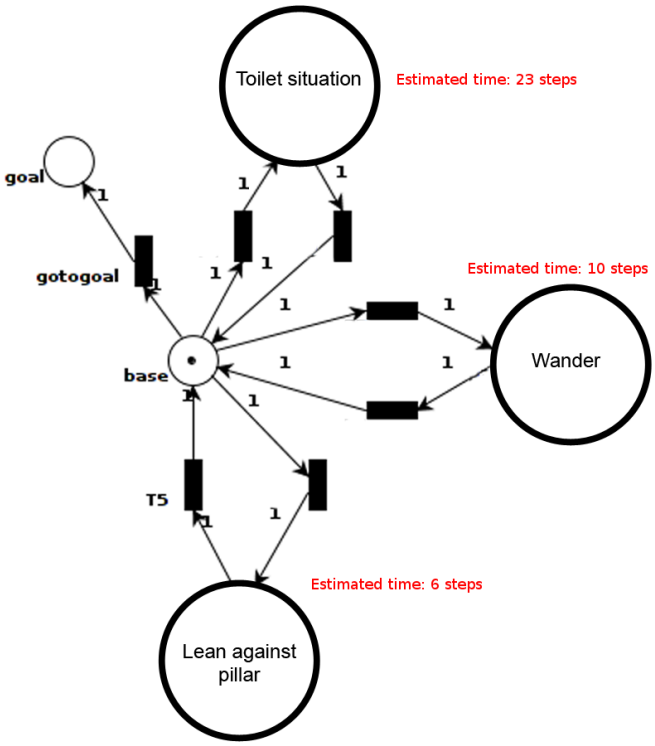


Figure 3.2: Our plan for separating pedestrian Petri nets from situation Petri nets.

transition that releases an endless amount of tokens into the Petri nets, which would upset our system. It is possible implement for example a situation in which a regular source transition is incorporated. Whether the system will work correctly is left to the judgement of the designer of the Petri net. An important rule though that all designs should adhere to is that a pedestrian net should always be restricted to transporting only one token to a situation Petri net, and only one should be transported back. This is because the system, expects this, and the behavior will become unpredictable otherwise. In figure 3.4 shows the process of connecting to a situation Petri net and exchanging tokens.

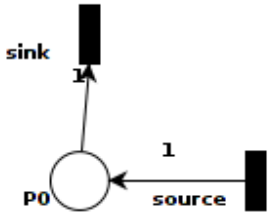
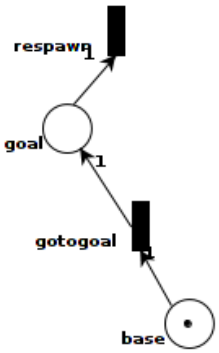
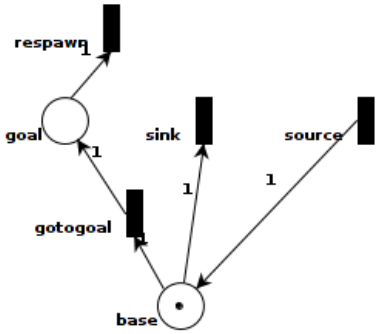


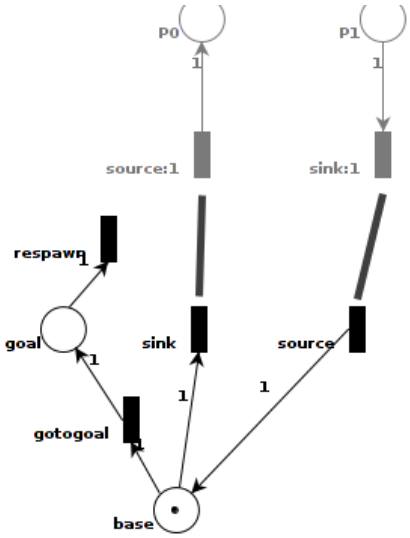
Figure 3.3: A place with a source and sink transition.



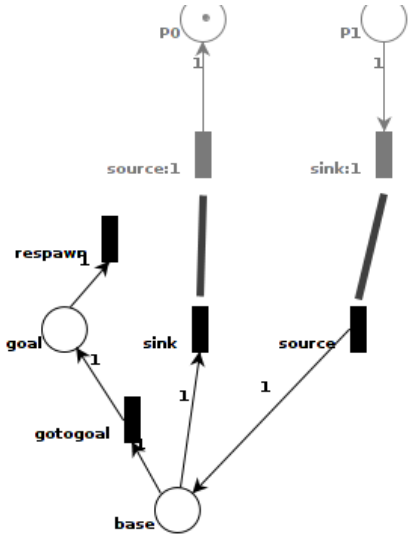
(a) Pedestrian Petri net with no situations attached.



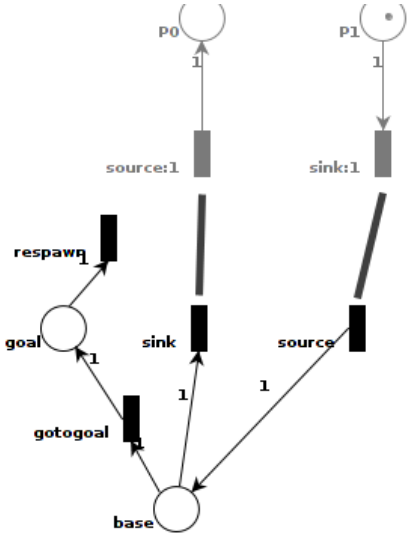
(b) Source and sink transitions are generated to attach to situation Petri net.



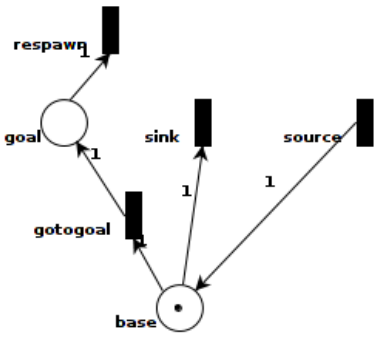
(c) Sink of pedestrian net is associated with source of situation net and vice versa.



(d) When the sink transition of the pedestrian net is fired, the source transition of the situation net is fired simultaneously.



(e) The final place that is attached to the special sink transition in the situation net is reached.



(f) The sink transition of the situation net is fired simultaneously with the source transition of the pedestrian net.

Figure 3.4: The process of connecting and exchanging tokens between pedestrian and situation Petri nets.

3.2 Time Planning & Decision Mechanism

In this section will be described how the pedestrian agents' behavior is decided upon using information about their deadlines and time spent doing the various behaviors. What is important to remember is that pedestrian Petri nets always have one *base place* from which it is always possible to reach the (time constrained) goal. This place should also be labeled **base** by the designer so our system can find it. In the following subsections, we will first describe how we estimate how much time a certain behavior costs, then we will show how we use this information together with the deadline time, a utility function and the current time to compute the probabilities of various behaviors, and finally we will propose various utility functions that we think will result in the desired behavior.

3.2.1 Estimating the Time of Behaviors

In order to save computing time during a simulation, we make a few computations before the simulation has started. An important step is the calculation of the distance in time between all the places in a Petri net and a destination place. This distance can easily be computed using the *Dijkstra shortest path algorithm* [7]. Dijkstra's algorithm is a graph search algorithm that can produce a shortest path tree for a single source, for a graph with nonnegative edges. In our system we can use this to compute the time from any place to the destination place (the source). We then pick the place from all the places in the Petri net the one that has the highest distance to the destination. We use this information to make an estimate of how long a pedestrian will be stuck to the behavior of a certain situation. However, it will never be more than an estimate, since it is possible to design Petri nets with (possibly) infinite loops, since the Petri nets are probabilistic, it will never be possible to give an exact prediction of the time it takes to execute a certain behavior. In figure 3.2 the estimates of the behaviors described in section 3.3 are shown.

3.2.2 Deciding on Behavior

Though the probabilities of the various behaviours a pedestrian can have are dependent on what kind of utility function we are going to use, the underlying mechanism will always be as described in the algorithm in figure 3.5. This algorithm is used to choose one of the various behaviors that are currently available to a pedestrian at a certain timestep. Once one is chosen, this behavior will be executed from the current timestep until it has finished, and the token has moved back to the base place of the pedestrian. This algorithm is only run for the pedestrians whose token is currently in the base place. When the token is in another place, the decision about which transition to choose follows the basic p/t Petri net rules.

The mechanism works as follows: first of all, we subtract the current simulation time from the deadline time of a specific pedestrian. When we have computed this t_d, we know how much time this pedestrian has left to reach its goal.

Next, the algorithm computes for every situation how much time is left would the pedestrian choose to execute this behavior. This remaining time is then used to compute the *utility* of these behaviors. The utility is computed with the utility function U, and can have various distributions. In section 3.2.3 the various utility functions that we experiment with are described. We will try out behavior various options for U to see which will give the most realistic behavior. When the utilities for all the behaviors have been computed, they are normalized, and based on the resulting probabilities, one of the behaviors is randomly chosen.

Algorithm 1 $\delta \leftarrow$ deadline time - current time
for Every connected transition numbered $t_0, t_1 \dots t_n$ **do**
$\epsilon \leftarrow time\_estimate(t_x)$ ▷ Estimated time of behavior connected to t_x
$\tau \leftarrow \delta - \epsilon$ ▷ Time left after doing behavior.
if $name(transition)$ ='gotogoal' **then**
$u_x \leftarrow U_{goal}(\tau))$
else
$u_x \leftarrow U_{non-goal}(\tau))$
end if
end for
$normalize(u_0 \dots u_n)$
Select randomly from situations $0 \dots n$, weighted by $u_0 \dots n$
end

Figure 3.5: The behavior decision mechanism

As one may have noticed, this approach does not give a completely watertight solution to the planning problem. However, since we have to be able to model large crowds, we cannot create an overly complex planning system, because then we would not be able to run the simulation real-time. Furthermore, we do not aim at finding an optimal solution to the planning problem, but rather the most lifelike behavior. In real life, people make errors in judgement, so creating pedestrians who can look ahead perfectly would not be realistic. It is impossible to make an exact definition of realism for our purposes, but what we try to do, is to copy certain specific behavior found in real-life footage. In our experiments we will try different utility functions to influence the probability of going to the goal or do something else, and attempt to assess which function will be most suitable.

3.2.3 The Utility Functions

Below the various utility functions we use will be described.

Sigmoid Function

A sigmoid function is an S-shaped curve that has a progression that accelerates and approaches a climax over time. This function can be found in many natural processes, such as learning curves. This function closely resembles how we reason that the behavior will shift when a pedestrian approaches a deadline. Our sigmoid function is defined as follows.

$$U(\tau) = \eta \frac{1}{1 + e^{-\frac{\tau - \mu}{\omega}}} \tag{3.1}$$

Our sigmoid function includes parameters for translation and transformation: η to multiply in the y direction, ω to multiply in the τ direction and μ to translate in the τ direction.

Linear Function

We first chose to use a sigmoid function, because intuitively it felt like it would reflect real-life behavior best. However, it could be the case that the sophistication of this function is lost in practice. If this is the case, we might as well use the simpler linear function. Our linear function was of the following form, where parameter d is the time of the deadline:

$$\text{If } 0 \leq \tau \leq d\text{: } P(\tau) = a\tau + b$$
$$\text{If } \tau < 0\text{: } P(\tau) = P(0)$$
$$\text{If } \tau > d\text{: } P(\tau) = P(d)$$

However, we do not specify parameters a and b directly. What matters the most to us, is the value $U(\tau)$ at $\tau = 0$, which we call U_0 and the value of U at the end where $\tau = d$, which we call U_d. Consequently, we compute a and b as follows:

$$a = \frac{U_d - U_0}{d}$$
$$b = U_0$$

Gaussian Function

We also used a Gaussian function to model the behavior. A Gaussian distribution (or normal distribution) is a continuous probability distribution with a bell-shaped probability density function and has mean (μ and variance (σ^2) as parameters.

$$U(\tau) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\tau-\mu)^2}{2\sigma^2}}$$

We chose to try a Gaussian function as well because of the following reasoning: A pedestrian might not care about going to it's goal until it is approximately the time of the deadline. With this we mean that going to the goal is a priority *around* the time of the deadline, and will also decline when the deadline has passed for a while, and the pedestrian hasn't reached its goal yet. With this reasoning, the "go-to-goal" behavior frequency should increase when approaching the deadline, and should peak *just* before the deadline, and decline thereafter.

3.3 Rotterdam Airport

In our research, we focus on behavior found at Rotterdam airport. In order to know what the behavior at this location looks like, we have observed footage from the security cameras at Rotterdam airport that the Intelligent Imaging department at TNO was so kind to provide. From these observations we have established a couple of specific behaviors that are recurrent in Rotterdam airport and managed to recreate the behavior in Petri nets.

- *Standing still, figure 3.6.*
  People stand still and do nothing very often while waiting.

- *Wandering, figure 3.7.*
  People also wandered around randomly.

- *Leaning against a pillar, figure 3.8.*
  Another recurring behavior we saw is that people lean against the pillars in the hall.

- *Going to the toilet, figure 3.9.*
  In the videos, we observed that a typical behavior that manifests itself multiple times in the video material is that one person goes to the toilet, and another one waits until this person has come back. After that, they move on to do something else. The Petri net can be found in figure

As can be seen, the standing still behavior in figure 3.6 was easiest to recreate, closely followed by the "wandering" behavior in figure 3.7 and "leaning against a pillar" behavior. The exact appearance in the simulation mostly depends on the implementation of the actions indicated in the places of the Petri nets. Whether there will be e.g. collision detection or other lower-level functionality depends on which kind of multiagent simulation is used. The "go-to-toilet" behavior in figure 3.9 is the most interesting, because this Petri net can actually be shared between two pedestrians. We see that this fairly complicates the structure of the Petri net. In comparison to the other Petri nets, there are relatively more places here that do not map to an actual action. These places are necessary to make sure that one pedestrian does not move forward to its next action too soon. This will cause the pedestrians to have more "idle" actions between their other actions than there would be with the simpler behavior. This is not a problem though, because the pedestrians will not remain idle longer than one or two timesteps.

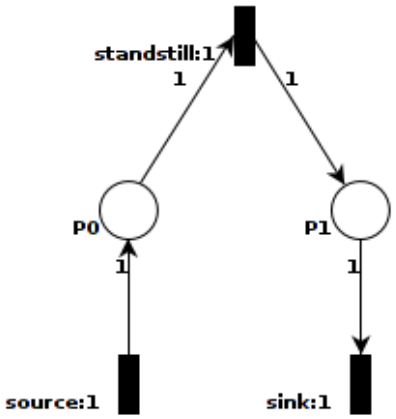


Figure 3.6: Pedestrian standing still.

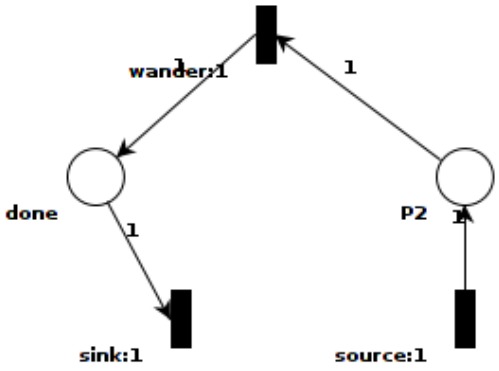


Figure 3.7: Random wandering around

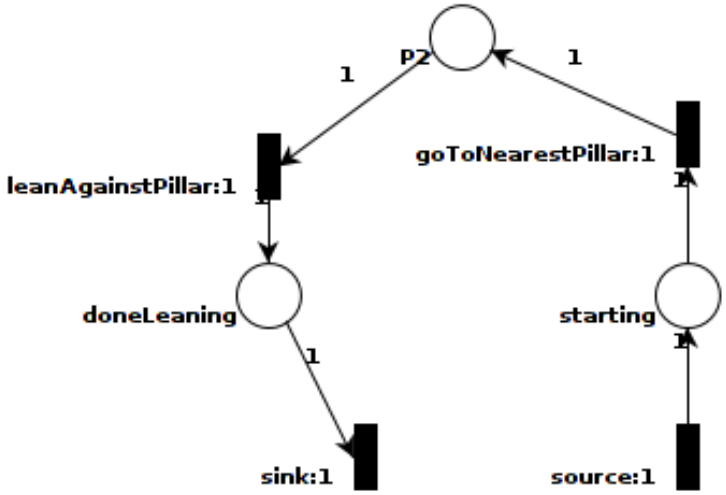


Figure 3.8: Going to the nearest pillar and leaning against it.

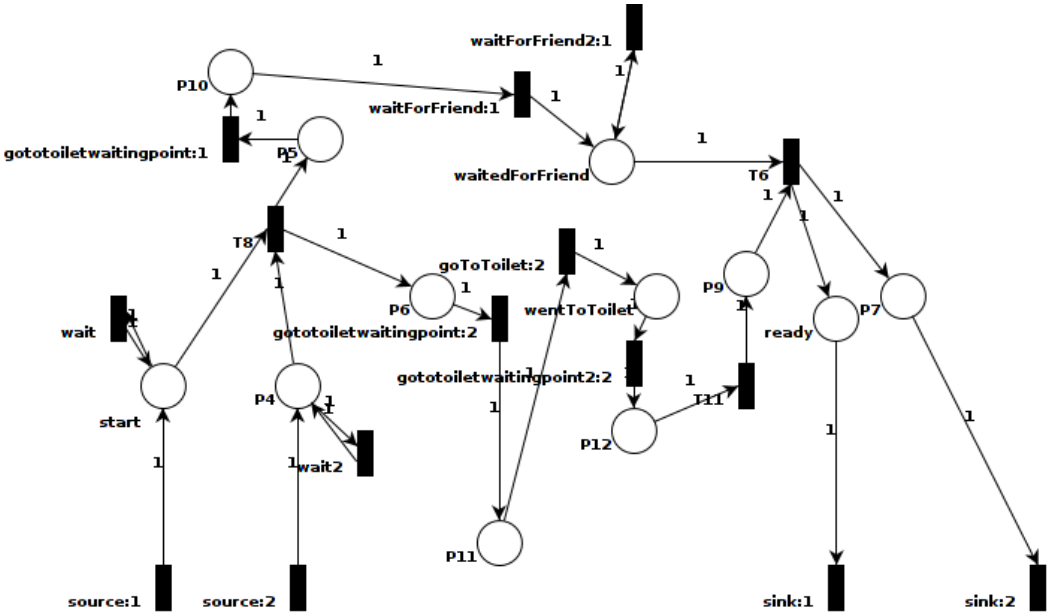
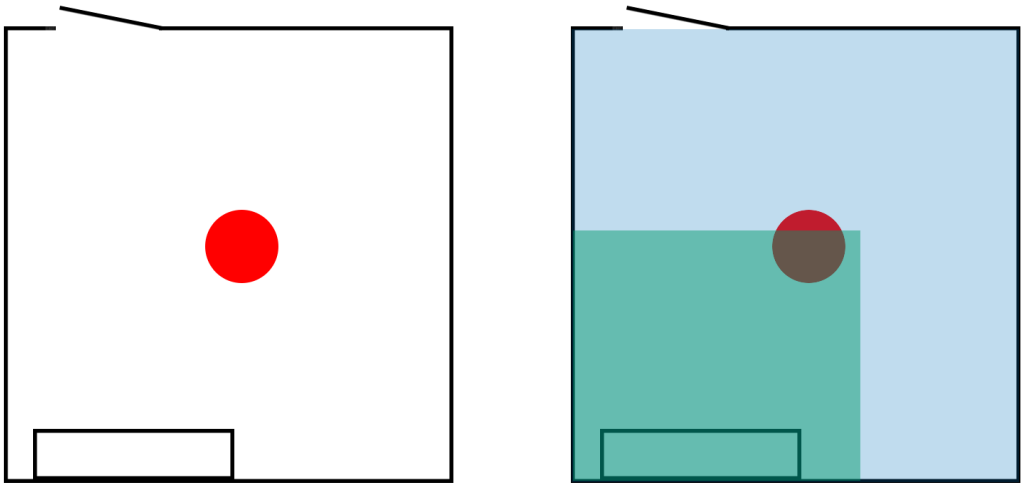


Figure 3.9: One person going to the toilet and one person waiting for the other.



(a) Room with agent **(b)** Room with agent and situation areas.

Figure 3.10: A hypothetical room that could be simulated.

3.4 Example

In order to give a clear insight into how our method is used in practice we will give a small example. Let us say that we need to simulate only one person, who has to wait in a room. At a certain point in time (let us say 100 steps), he has to advance through a door. There are a few objects in this room with which he can interact. In figure 3.10a we see a schematic representation of this room. The red circle in the middle is the agent, the rectangle on the lower left side is a bench and on the upper left side we can see a door going out of this room. our agent can do a couple of things in this room. First of all, he can wander around the room. In our representation it is possible to decide to do this standing in any position in this room. Another possibility is to sit on the bench. For the purpose of this example it is only possible for the agent to decide to do this when in close proximity of the bench. In figure 3.10b the situation areas for these two behaviors can be seen. Our agent is not planning to stay in this room forever though. He wants to leave the room before 100 (time) steps, so leaving the room will be our go-to-goal"gotogoal" action.

The high level behavior is described in Petri nets. Our pedestrian comes with only a very basic Petri net, with its only transition being the gotogoal transition as shown in figure 3.11. The bench and wander situation have their associated situation Petri nets, as shown in figure 3.12 and 3.7 respectively. The red numbers indicate how much timesteps the designer of the Petri net thought these transitions would take. When the simulation is run, the time in steps from every place in the situation Petri nets to the sink transition are computed, and the maximum time per situation are registered as heuristic. These heuristics are quite easy to compute for our example Petri nets. For the bench situation it would be $1 + 3 + 10 + 1 + 1 = 16$. For the wander situation $1 + 3 + 1 = 5$. Of course, our simulation also needs goal and non-goal utility functions. Let us say they are the following:

$$U_{gotogoal} = -\frac{1}{100}\tau + 1$$

$$U_{nongoal} = \frac{1}{100}\tau \tag{3.2}$$

After this the preprocessing of the actual simulation starts. Our pedestrian stands in the situation area of both the wander situation and the bench situation. That means the Petri nets of both will be attached to the pedestrian. How they are connected through sources and sinks can be seen in figure 3.14.

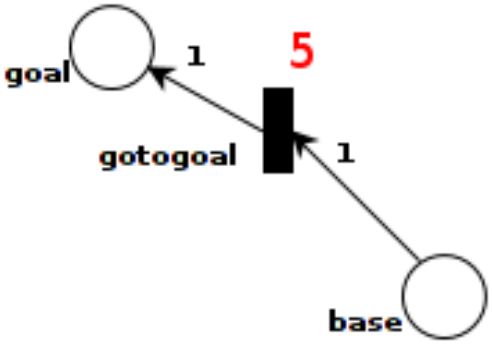


Figure 3.11: Our agent's basic pedestrian Petri net.

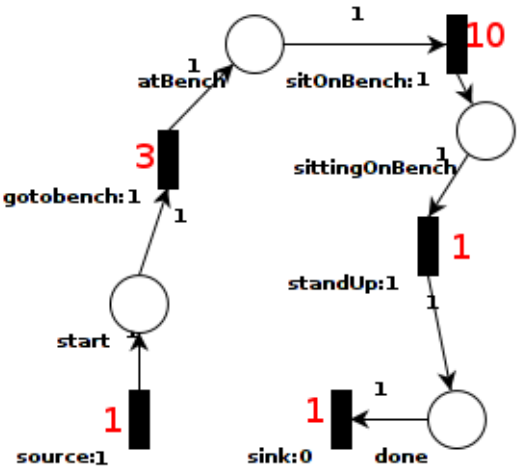


Figure 3.12: The Petri net of the bench situation.

The token of the pedestrian is in the base place. That means transitions connected to Petri nets of the situations can be chosen. He still has 100 timesteps left. The estimated time left (named τ after doing the wander behavior would be $100 - 5 = 95$ for the bench situation $100 - 16 = 84$. The time left after executing the go-to-goal behavior would be $100 - 5 = 95$. The utilities are computed as follows:
$U_{bench} = \frac{1}{100} * 84 = 0.84$
$U_{wander} = \frac{1}{100} * 95 = 0.95$
$Ugotogoal = -\frac{1/100}{*}95 + 1 = 0.05$

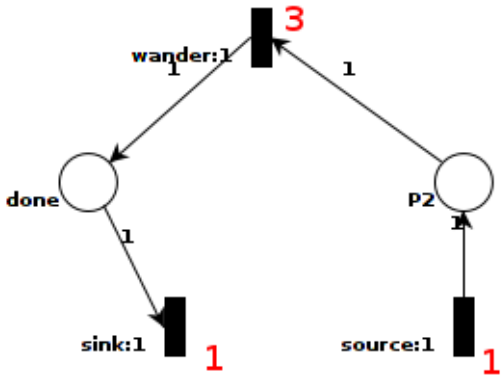


Figure 3.13: The Petri net of the wander situation.

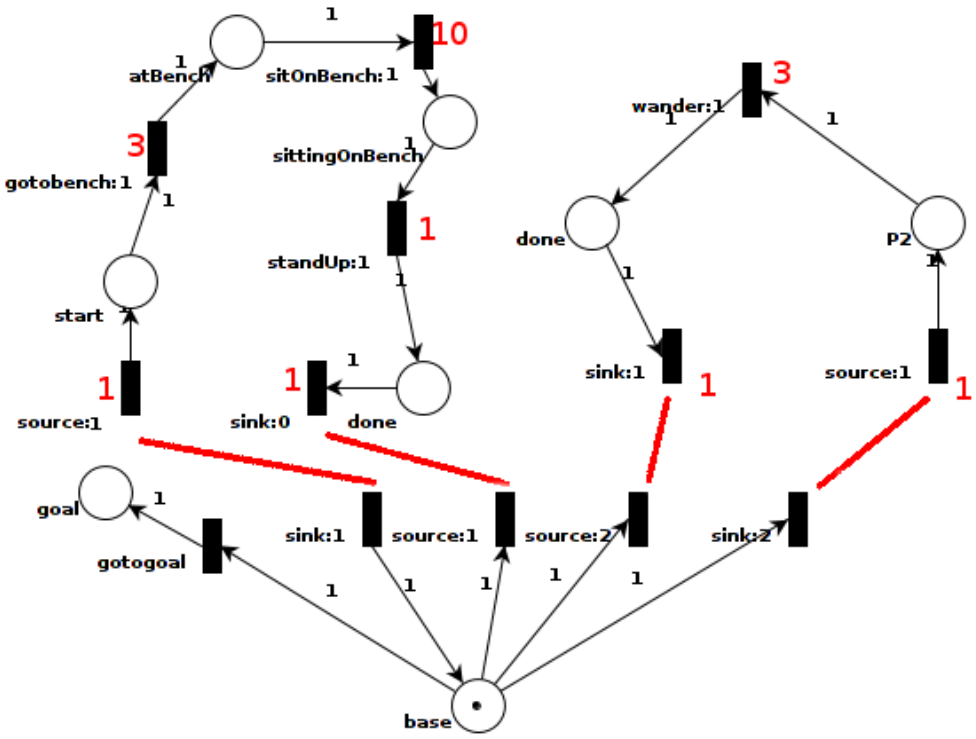


Figure 3.14: Example of how the Petri nets are connected.

To compute the probabilities for selecting the transitions, these utilities are normalized, which leads to the following values:
$P(choosing\_bench\_transition) \approx 0.46$
$P(choosing\_wander\_transition) \approx 0.51$
$P(choosing\_gotogoal\_transition) \approx 0.03$
So most likely the agent will wander around or sit on the bench. Let us say the sink transition connected to the wander Petri net will be randomly chosen. Now the token gets consumed by sink transition *sink:2* and produced by *source:1* 1 in the Petri net with wander behavior. Every time an action needs to be taken, the active transition in the Petri net will be fired. First it will be *wander:1*, then *source:2*, and the cycle starts anew,

until the go-to-goal transition is chosen and the pedestrian moves through the door.

Chapter 4

Experiments

Although it can be difficult to decide whether a group of people walks around "realistically", we will certainly give it a try. We will attempt to test our framework in two different ways; first of all, we will assess our framework by doing a qualitative comparison with real-life footage of Rotterdam Airport. We have access to both camera footage and manually tracked locations of the visitors. Secondly, we will assess how the the different utility functions we choose for the go-to-goal action will affect the frequency of other actions. In other words, we will attempt to investigate whether our method leads to *emergent* behavior.

4.1 Qualitative Experiment

First of all, we will see if it is possible to recreate real-life behavior in Rotterdam airport with our framework. As mentioned before, we have observed footage from the security cameras at Rotterdam airport. From these observations we established a couple of specific behaviors that are recurrent in Rotterdam airport, as mentioned in section 3.3. We will briefly mention them here again:

- *One person going to the toilet, other person waiting.*
  We have observed this behavior 2 times in 10 minutes
- *Leaning against a pillar* We have observed this behavior approximately 2 times in 10 minutes.
- *Standing still* This behavior happens almost constantly
- *Wandering* This behavior happens constantly. Although it is sometimes hard to assess whether a pedestrian is randomly wandering around because he has to wait or actually has a goal in mind.

In order to make a fair comparison, we searched for these behaviors in the tracked data, and drew their approximate trajectories, which can be found in figure 4.1, 4.2 4.3, and 4.4. The trajectories are divided into different steps.

Modify toiletsituation so it has 4 steps just like in the results

In this qualitative experiment, we will try to imitate these behaviors using our Petri nets. We expect that our Petri nets will be capable of imitating these behaviors.

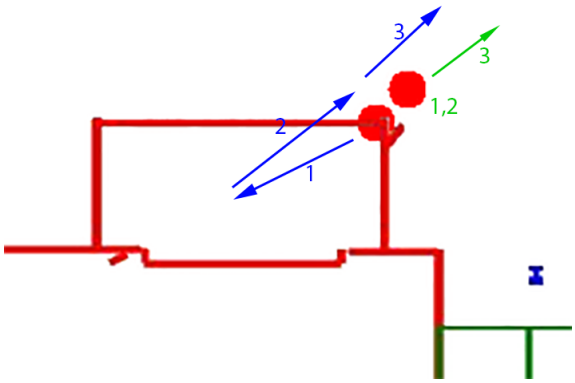


Figure 4.1: Approximate trajectories of one person going to the toilet, other person waiting.

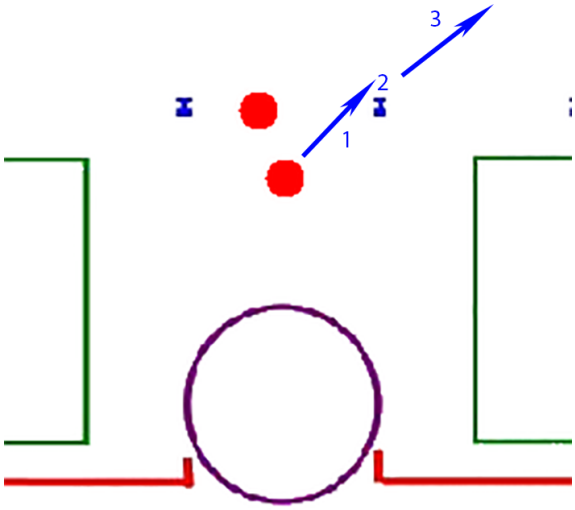


Figure 4.2: Approximate trajectory of a person going to a pillar and standing against it.

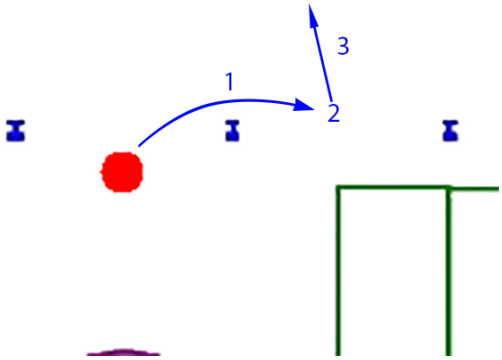


Figure 4.3: Approximate trajectory of a person standing still for a while and then moving on.

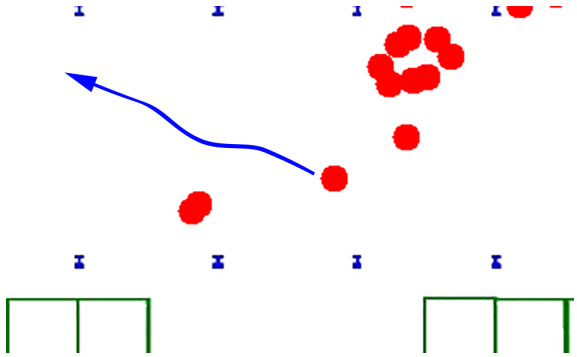


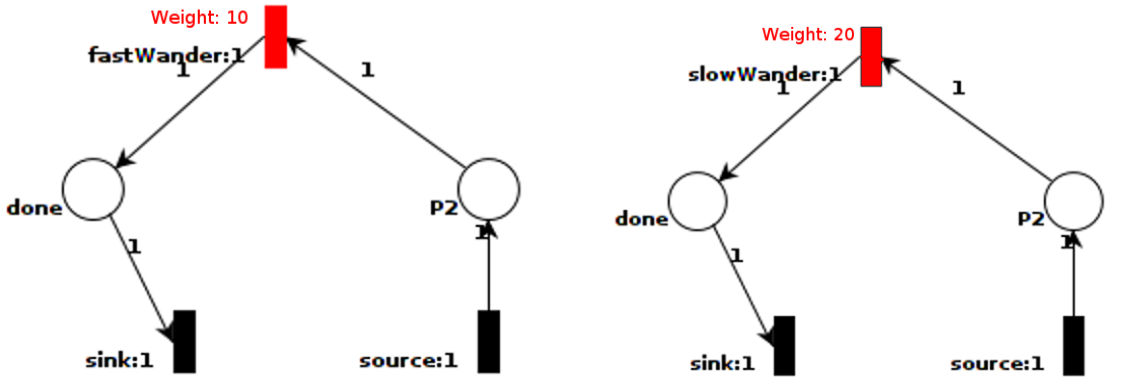
Figure 4.4: Approximate trajectory of a person wandering around.

4.2 Quantitative Experiment

It is quite a challenge to quantitatively establish whether lifelike behavior has been modeled. However, it is possible to check whether the mechanics of time planning work as predicted. If the mechanics work as expected, the simulation should start with a preference of *relaxed* over *hurried* behavior. Then, as time passes and the deadline draws near, the frequency of hurried behavior should increase while relaxed behavior decreases.
In order to test our theory, we run the simulation once with 100 pedestrians and log for every step in time, for every pedestrian, which action he is doing at the moment. This information will be plotted in a graph that shows the frequency of every action at every step in time.
The deadline of going to the goal will be 200 steps in our simulation. We chose this value because our behavior takes up to roughly 40 steps (but most behavior takes 20 steps or less) and this leaves enough room for the pedestrians to decide on multiple behaviors while "relaxed" before time starts running out and they have to start hurrying.
To distinguish between hurried and relaxed behavior we replaced the "wander" behavior of the qualitative experiment by two other behaviors, called *fastwander* and slowwander. We use these two behaviors to try to get a clear distinction between hurried and relaxed, and see how the number of pedestrians doing one or the other changes over time. The petri nets for these behaviors look exactly the same, as can be seen in figure 4.5. The difference between these Petri nets is in the weight of the slowwander:1 and fastwander:1 transitions. These are 20 versus 10 respectively. This weight is used by the Dijkstra algorithm to compute the amount of time that a this behavior will take. The reason that these are 20 and 10 is that we programmed the slowwander behavior in such a way that the pedestrian will walk roughly in one direction for 20 timesteps. Fastwander makes the pedestrian walk faster than slow wander, and for a shorter period of time, namely 10 timesteps. We added slowwander and fastwander situation areas in the simulation that both encompass the entire map.

Because of the structure of our implementation, it was easier to log the actions of the pedestrians instead of the Petri nets that are fired for every pedestrian. Because most Petri-nets only contain one action, it is an *almost* one-on-one mapping from action to Petri net (or behavior), but for clarity, we will list the various actions that can be executed below:

- *fastwander*
- *slowwander*



(a) "Fast wander" behavior. **(b)** "Slow wander" behavior.

Figure 4.5: Petri nets of the new behaviors

- *gototoilet* and *waitforfriend* (these both belong to the gototoilet situation)
- *gotonearestpillar* and *leanagainstpillar* that are both part of the leanagainstpillar situation
- *standstill*
- *idle*

The last action is the default action when nothing else can be executed. It is executed when a fired transition does not have an associated action. Another situation in which this idle behavior can happen that will be important in these experiments, is when there are no transitions left anymore to fire. This can be for example when all transitions have gotten a probability of 0 for firing.

We will try different utility functions for the go-to-goal and non-goal behavior to see how this influences the simulation. We chose our parameters based on which values would give a significant but gradual increase in the course of this timeframe and give the maximum utility on either $t = 0$ or $t = 200$, the first when the function is used as utility measure for go-to-goal behavior, the latter when applied to non-goal behavior. The utility functions will be judged on whether they result in behavior that transitions from relaxed to hurried, and whether they help the pedestrians reach their goal in time.
We have used the combination of functions listed in table 4.1. For more information about the various parameters refer back to section 3.2.3. We will judge these results on two aspects: *deadline drivenness* and the transition from *hurried* to *relaxed*. The results can be found in the next chapter in section 5.2.

4.3 Second Quantitative Experiment

It is possible that the goal and non-goal utility functions interfere with eachother. For example, when the goal utility function causes the pedestrians to go to their goal early in the simulation, they might not have time enough to show a transition from hurried to relaxed behavior. That is why we decided to run a second quantitative experiment. The purpose of this experiment is to follow the the hurried and relaxed behavior of the

	Goal Utility	**Non-goal Utility**
1	Linear: $U_0 = 1, U_d = 0$	Linear: $U_0 = 0, U_d = 1$
2	Linear: $U_0 = 1, U_d = 0$	Sigmoid: $\mu = 100, \beta = 0, \omega = 10, \eta = 1$
3	Linear: $U_0 = 1, U_d = 0$	Gaussian: $\mu = 200, \sigma = 50$
4	Linear: $U_0 = 0.1, U_d = 0$	Linear: $U_0 = 0, U_d = 1$
5	Linear: $U_0 = 0.1, U_d = 0$	Sigmoid: $\mu = 100, \beta = 0, \omega = 10\eta = 1$
6	Linear: $U_0 = 0.1, U_d = 0$	Gaussian: $\mu = 200, \sigma = 50$
7	Sigmoid: $\mu = 100, \beta = 0, \omega = -10, \eta = 1$	Linear: $U_0 = 0, U_d = 1$
8	Sigmoid: $\mu = 100, \beta = 0, \omega = -10, \eta = 1$	Sigmoid: $\mu = 100, \beta = 0, \omega = 10, \eta = 1$
9	Sigmoid: $\mu = 100, \beta = 0, \omega = -10, \eta = 1$	Gaussian: $\mu = 200, \sigma = 50$
10	Gaussian: $\mu = 0, \sigma = 50$	Linear: $U_0 = 0, U_d = 1$
11	Gaussian: $\mu = 0, \sigma = 50$	Sigmoid: $\mu = 100, \beta = 0, \omega = 10, \eta = 1$
12	Gaussian: $\mu = 0, \sigma = 50$	Gaussian: $\mu = 200, \sigma = 50$

Table 4.1: The list of combinations of utility functions used for the first quantitative experiment.

pedestrians with the goal utility being the constant function $U(\tau) = 0$. We varied the non-goal utility functions in the same way that we did in the first quantitative experiment. For clarity, the functions and parameters we will use are the following:

- Linear: $U_0 = 0, U_d = 1$
- Sigmoid: $\mu = 100, \beta = 0, \omega = 10, \eta = 1$
- Gaussian: $\mu = 200, \sigma = 50$

The results will again be judged on its transition from relaxed to hurried. We expect that this transition will be more clear than in the previous experiment.

Chapter 5

Results

5.1 Results of the Qualitative Experiment

We ran a simulation that included the behaviors that we found in real-life, and observed the trajectories that the simulated behaviors made. The resulting trajectories can be found in figure 5.1, 5.2, 5.3 and 5.4. From a qualitative point of view, it seems that the behavior of one person going to the toilet and one person waiting could be mimicked very well. Also leaning against a pillar has a trajectory close to the real-life trajectory. Standing still is such a trivial move that it is hardly surprising that it could be mimicked in our simulation. Lastly, 5.4 shows the trajectory of a simulated agent wandering around. One thing that might stand out is that the trajectory in our simulation swerves around more than the real-life trajectory. Part of this behavior is easily modified by changing the parameters of the wandering behavior, but it also illustrates a more high-level difference between real-life "wandering around" and the behavior in our system. it seems that often, the people tracked in real-life have some kind of determination to go to a certain place. The behavior is not as random as our simulated behavior. This is not a weakness of our system though, but merely indicates that more inquiry needs to be done into peoples' intentions in a airport-type scenario. Unfortunately this is outside the scope of our research.

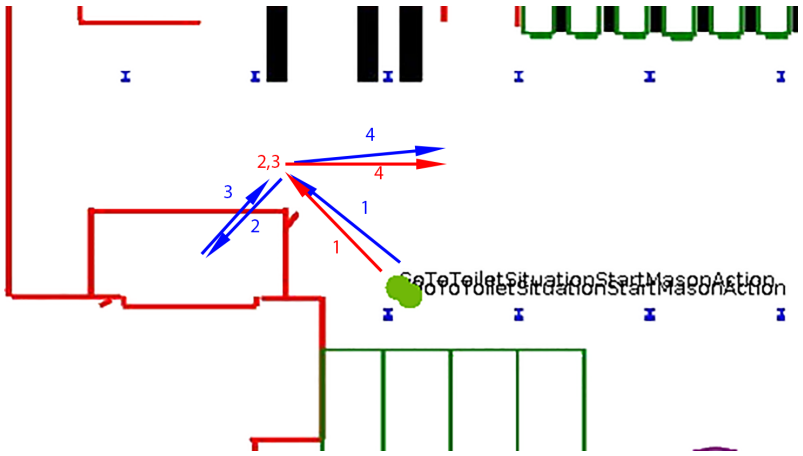


Figure 5.1: Approximate trajectories of one person going to the toilet, other person waiting.

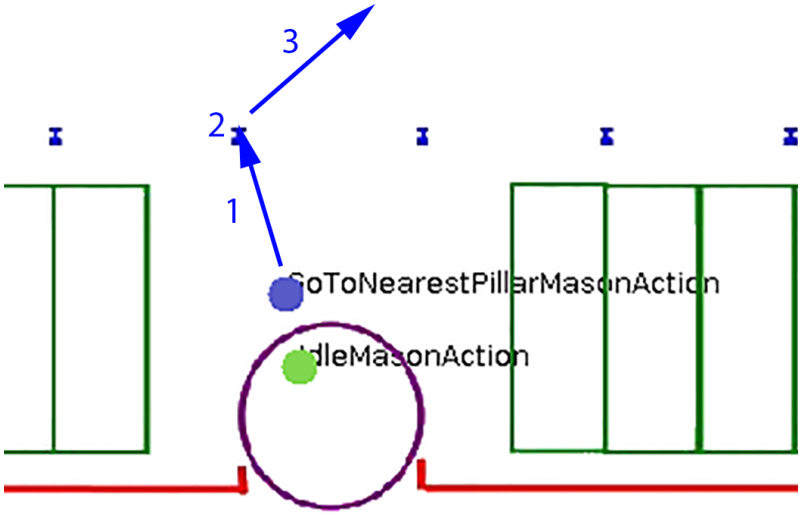


Figure 5.2: Approximate trajectory of a person going to a pillar and standing against it.

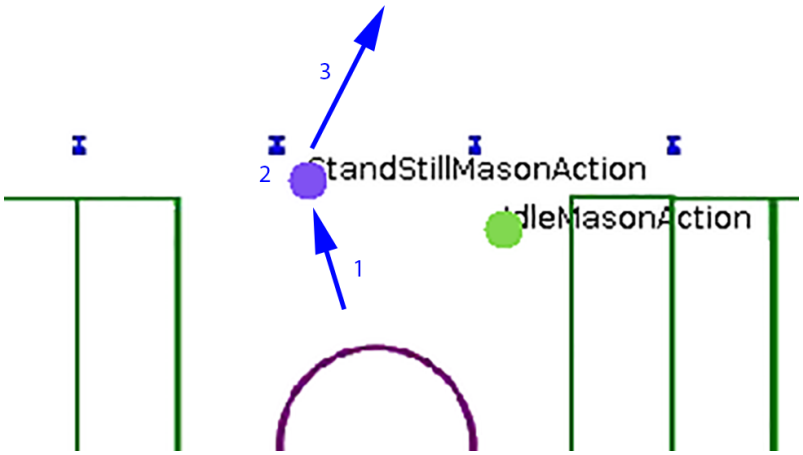


Figure 5.3: Approximate trajectory of a person standing still for a while and then moving on.

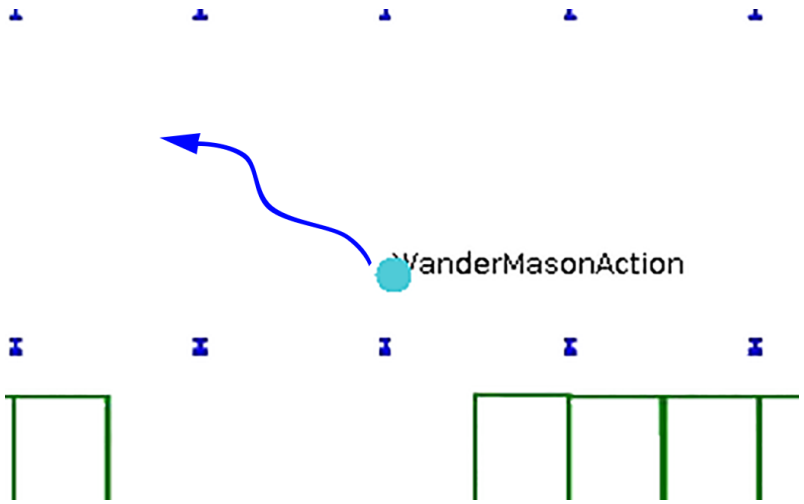


Figure 5.4: Approximate trajectory of a person wandering around.

5.2 Results of the First Quantitative Experiment

Firstly we are going to discuss the results of the first quantitative experiment. One thing that is noticable in all results is that there are periods in which the number of pedestrians, especially in slow and fast wander, is steady, after which the number drops drastically. After a few steps the number goes up again. In sync with this behavior the number of pedestrians remaining idle stays low, and increases when the other behaviors drop. This is easily explained. Slow and fast wander are behaviors that exist of a single action. When this action has been executed, which takes a few steps, the token of the pedestrian moves out of the attached Petri net into its base place. In order to move this token, a slot and sink transition have to be fired. These transitions do not have an associated action. When there is no action available for the pedestrian, it executes the idle behavior. Especially at the start, this effect is very prominent. This is because all pedestrians start the simulation at exactly the same moment. All pedestrians executing the same actions or actions that take the same amount of time stay synchronous. After a few dozens of steps, the behavior has varied more and the pedestrians are not as synchronous any more in going back to the idle "action".

Another thing that we can notice is that there are more or less four actions that dominate simulation, namely *slow wander*, *fast wander*, *go to goal*, and *idle action*. Behavior like going to the toilet and waiting for their friend (which are both part of the same going to the toilet behavior and Petri net) are only done by one or two pedestrians at a time. This is completely as expected. The go-to-toilet situation is shared, which means that one Petri net is attached to multiple pedestrian Petri nets. The more dominant actions belong to situations that are not shared and are instantiated for every pedestrian that enters it. That means that the shared Petri nets are only instantiated once for every situation, while the ones that aren't shared are instantiated many times.

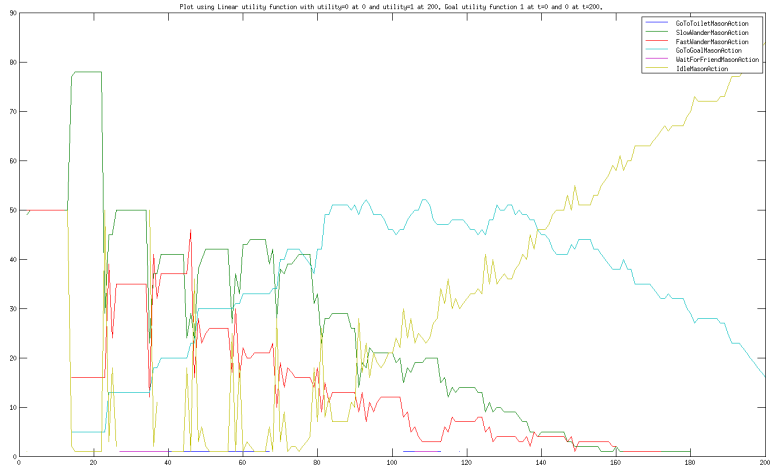


Figure 5.5: Goal utility linear with $U_0 = 1$ and $U_n = 0$. Other behavior utility linear with $U_0 = 0$ and $U_n = 1$.

In figure 5.5, 5.6 and 5.7 we see the results for having a linear goal utility function that decreases from 1 to 0 and the utility function for other behavior varying. Figure 5.5 and 5.6 look very similar. The simulation starts out with a large preference for slow wander behavior, which decreases while fast wander increases. After about 70 steps however, the gotogoal action starts to dominate the simulation, increasing at a fast pace. This makes the other actions decrease rapidly until pedestrians are either idle (because they have already reached their goal) or still going to their goal. When we look at figure 5.7 we see

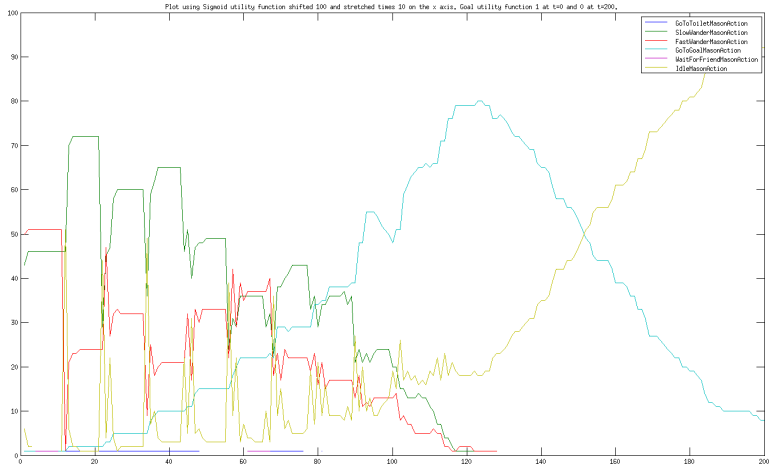


Figure 5.6: Goal utility linear with $U_0 = 1$ and $U_n = 0$. Other behavior utility sigmoid with $t_0 = 100$ $\beta = 0$ $\omega = 10$ $\eta = 1$

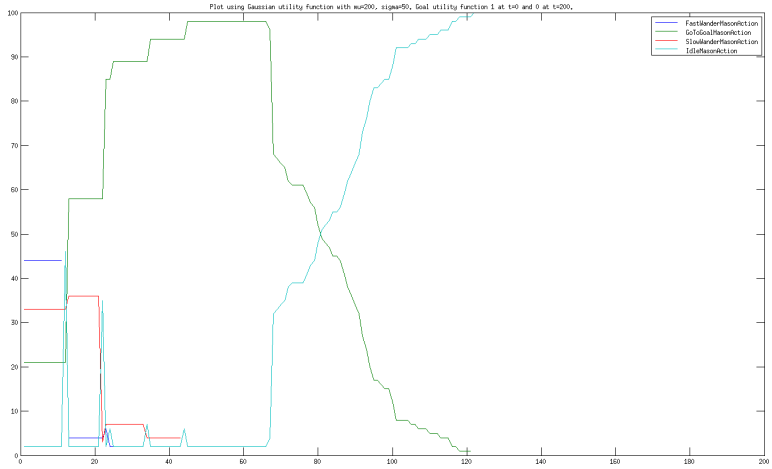


Figure 5.7: Goal utility linear with $U_0 = 1$ and $U_n = 0$. Other behavior utility Gaussian with $\mu = 200$ and $\sigma = 50$.

that almost all pedestrians go to the goal in the first dozen of steps. Fast wander and slow wander become completely overshadowed.

In figures 5.8, 5.9 and 5.10 we see the results for the experiments with the goal utility function with $U_0 = 0.1$ and $U_n = 0$. We varied the utility function of the other behavior the same as before. For some reason, the go-to-toilet and wait-for-friend actions are not present in this simulation. They have been replaced by the stand-still action. We would have expected to see more variation in actions now the utility of the go-to-goal behavior has been lowered. Instead, the time-consuming go-to-toilet behavior has been replaced by the less demanding stand-still behavior.

In figure 5.11, 5.12 and 5.13 are the results for the experiments with a sigmoid goal utility function where $t_0 = 100$, $\beta = 1$, $\omega = -10$ and $\eta = 1$. The effects are more or less the same as for the previous results. Again, the sigmoid goal utility function causes the pedestrians to go to the goal very soon, eliminating the possibility to do other behavior very soon in the simulation.

The results for a Gaussian goal utility function are shown in figure 5.15, 5.14 and 5.16. We see that in general, the pedestrians seem to wait longer before they go to their goals.

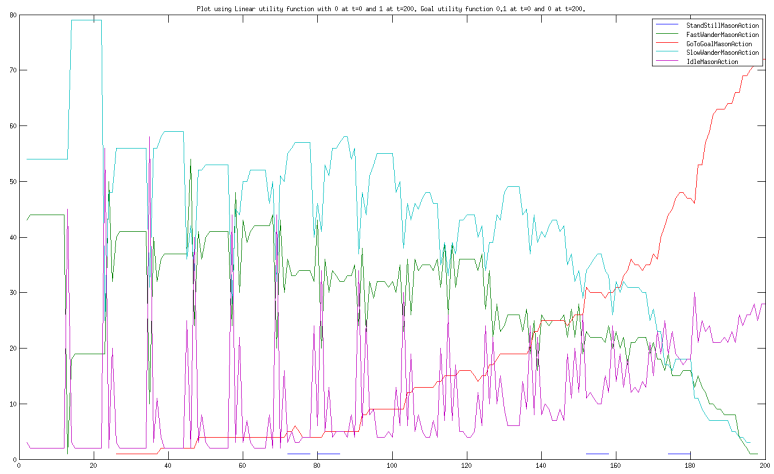


Figure 5.8: Goal utility linear with $U_0 = 0.1$ and $U_n = 0$. Other utility linear with $U_0 = 0$ and $U_n = 1$

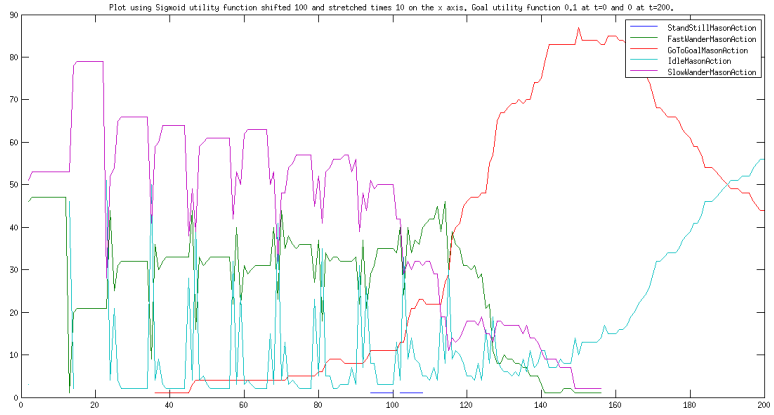


Figure 5.9: Goal utility linear with $U_0 = 0.1$ and $U_n = 0$. Other utility sigmoid with $t_0 = 100$ $\beta = 0$ $\omega = 10$ $\eta = 1$.

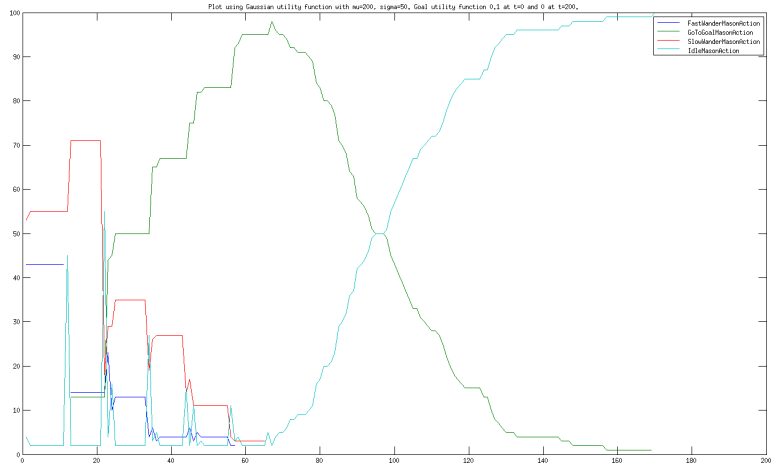


Figure 5.10: Goal utility linear with $U_0 = 0.1$ and $U_n = 0$. Other utility Gaussian with $\mu = 200$ and $\sigma = 50$.

This also causes more pedestrians to be too late, because they still are on their way to the

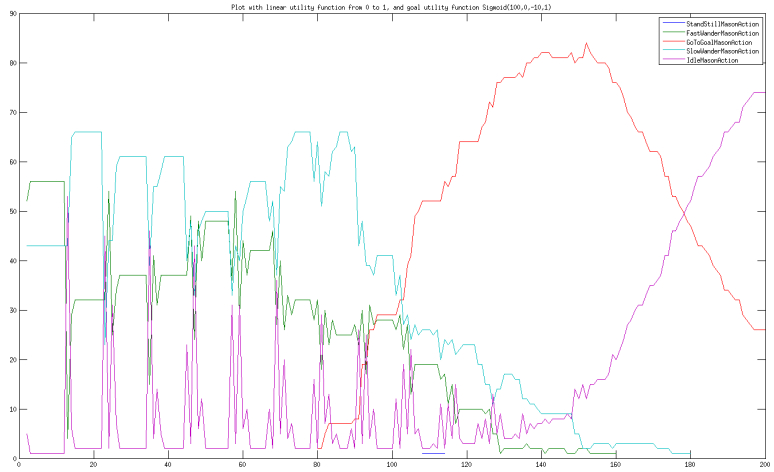


Figure 5.11: Goal utility sigmoid with $t_0 = 100$ $\beta = 0$ $\omega = -10$ $\eta = 1$. Other utility linear with $U_0 = 0$ and $U_n = 1$

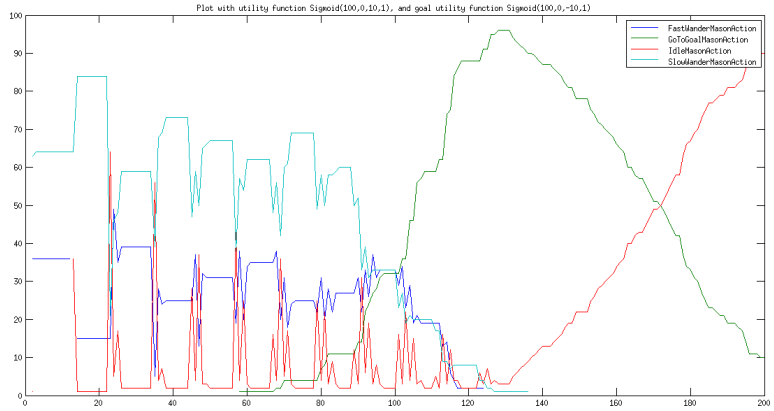


Figure 5.12: Goal utility sigmoid with $t_0 = 100$ $\beta = 0$ $\omega = -10$ $\eta = 1$. Other utility sigmoid with $t_0 = 100$ $\beta = 0$ $\omega = 10$ $\eta = 1$.

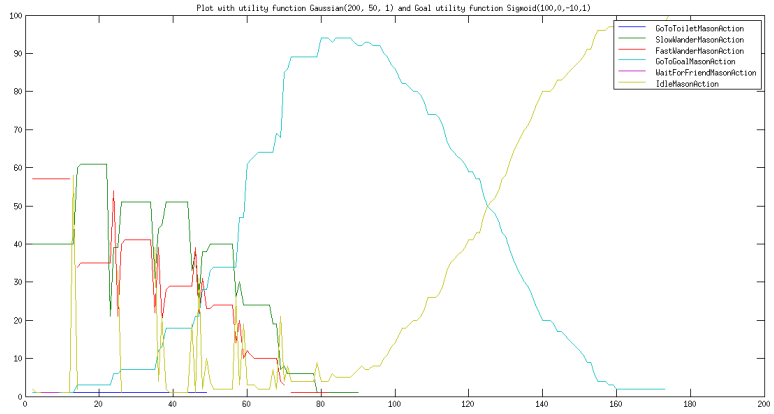


Figure 5.13: Goal utility sigmoid with $t_0 = 100$ $\beta = 0$ $\omega = -10$ $\eta = 1$. Other utility Gaussian with $\mu = 200$ and $\sigma = 50$.

goal when the deadline passes. This is very likely due to the fact that the estimation of going to the goal is too low for some pedestrians, because the area they move in is larger than I took into account when estimating the time needed to go to the goal.

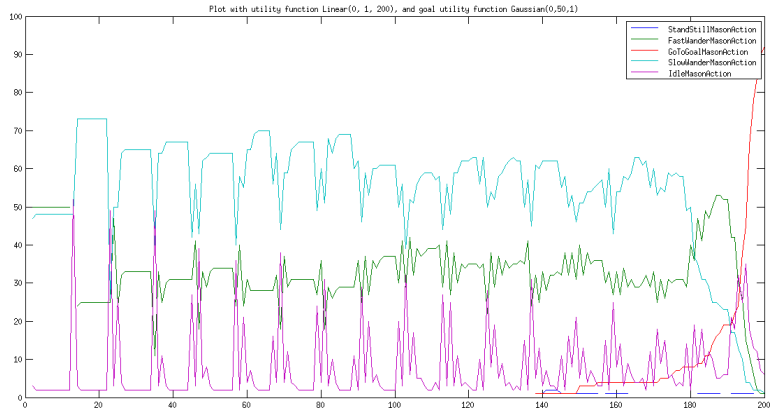


Figure 5.14: Goal utility Gaussian with $\mu = 0$ and $\sigma = 50$. Other utility linear with $U_0 = 0$ and $U_n = 1$

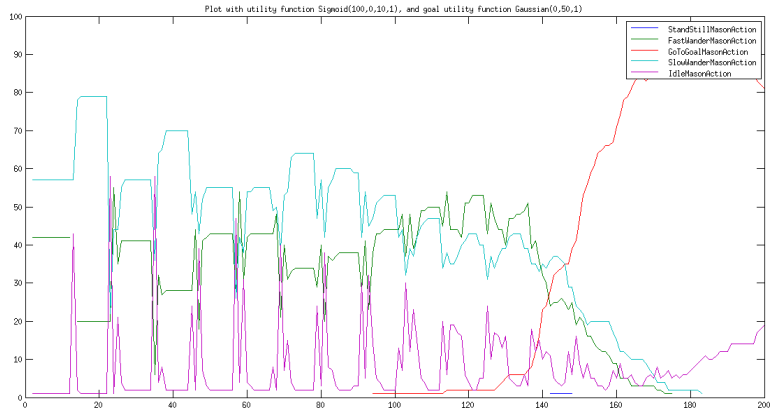


Figure 5.15: Goal utility Gaussian with $\mu = 0$ and $\sigma = 50$. Other utility sigmoid with $t_0 = 100$ $\beta = 0$ $\omega = 10$ $\eta = 1$.

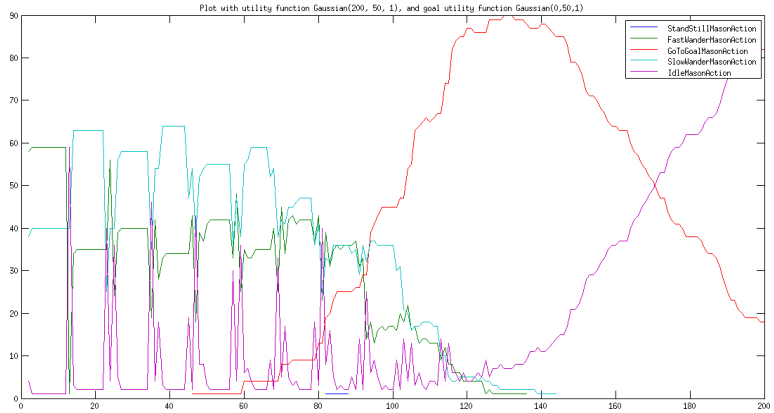


Figure 5.16: Goal utility Gaussian with $\mu = 0$ and $\sigma = 50$. Other utility Gaussian with $\mu = 200$ and $\sigma = 50$.

From these results we can make a table in which we judge the utility functions on two important aspects: change from relaxed to hurried behavior (table 5.1) and deadline-drivenness in table 5.2 (do they reach the goal in time?). We give the graphs a score in the following way: when it scores sufficiently, we give a checkmark (✓). If they score better, we have + and ++, and for worse, – and – –. We also gave a special score in table 5.2, namely >>. It means that the pedestrians reached their goal too early. When we would judge in terms of whether they are on time, we should have given them ++. However, the pedestrians reached their goal so early that they hardly did any other behavior than going to their goal. This is not desirable, so we decided to score them differently.
When looking at the tables, there is only one combination of utility functions that scores sufficiently or better on both categories. A Gaussian goal utility function combined with a sigmoid non-goal utility seems to behave the best in our simulation.

		Goal			
		Linear	Linear to 0.1	Sigmoid	Gaussian
Non-goal	L	–	+	–	✓
	S	–	✓	–	✓
	G	– –	– –	– –	–

Table 5.1: Utility functions judged by change from relaxed to hurried behavior

		Goal			
		L	L to 0.1	S	G
Non-goal	L	+	– –	s	–
	S	++	–	+	++
	G	>>	>>	++	+

Table 5.2: Utility functions judged by if the pedestrians reach the deadline soon enough.

In short, a number of combinations of utility functions perform well on one of the two criteria, and the combination of a Gaussian goal utility function with a sigmoid non-goal utility performs at least sufficiently on both. We do notice that the goal utility function does start to have a large influence after only a few dozen of steps in most simulations. This way it is possible that the goal utility function "sabotages" the development of a nice transition from hurried to relaxed behavior (we will also talk about this in chapter 6). This was actually why we decided to do the second quantitative experiment that we described in the previous chapter (4.3). The results can be found in section 5.3.

5.3 Results of the Second Quantitative Experiment

In the first experiment, the go-to-goal behavior was so dominant that it was difficult judge other emergent behavior. The following results will give a clearer view of whether hurried and relaxed behavior can emerge from our framework.

First of all, we have figure 5.17 where the $U = 1$ for all t. We see the typical phases that we saw in section 5.2 where most pedestrians execute slow- or fast wander for a while, after which the frequency of idle behavior goes up for a few steps. We can also see a few pedestrians doing the go-to-toilet and the accompanying wait-for-friend action. The

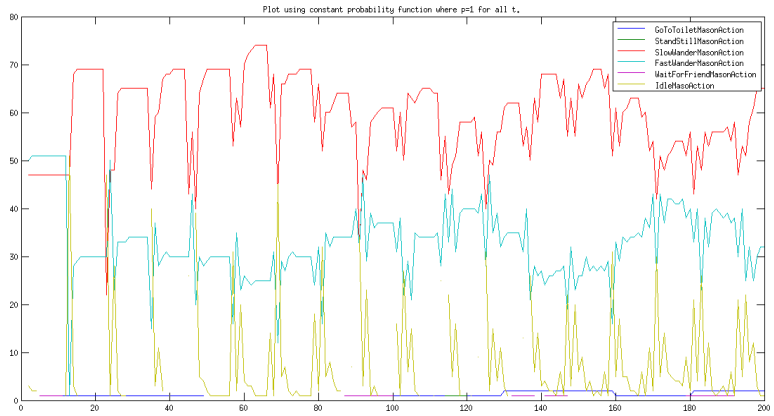


Figure 5.17: Non goal utility is constant with $U = 1$

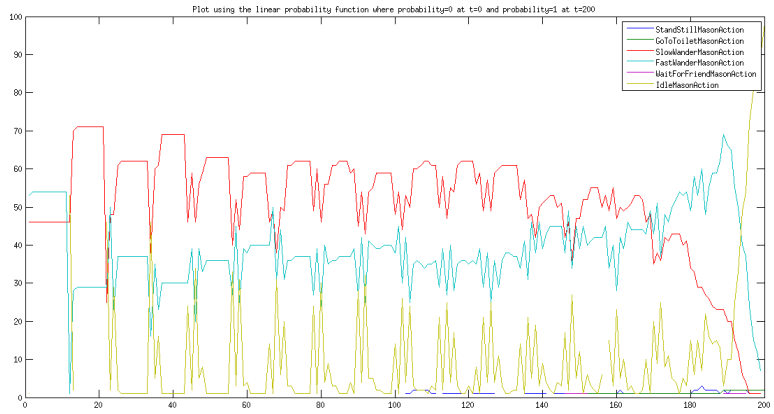


Figure 5.18: Non-goal utility linear with $U_0 = 0$ and $U_n = 1$.

frequencies of the actions deviate more or less around the same value through the whole simulation.

Next, we have the results for the utility functions that do decrease when time runs out. It is very clear that eliminating the goal gives the pedestrians the freedom to do different kinds of behavior. We see that the slow wander action has the preference most of the time, except when time has almost run out. Slow wander then decreases while fast wander increases, until even this less time consuming action takes too much time, and the pedestrians become idle. This preference of relaxed behavior (slow wander) until that takes to long to catch a deadline and transitioning to hurried behavior (fast wander) is exactly what we wanted to show with our framework.

In figure 5.19 we very clearly see the influence of the shape of the sigmoid curve. At around 100 steps the pedestrians exchange their preference of relaxed behavior (slow wander) for hurried behavior (fast wander). With a Gaussian utility function, the resulting graph resembles the results of the linear function again for the first part, but after about 180 steps, it is quite different. We saw that with linear utility functions, the frequency of slow wander would decrease first, followed by fast wander, while idle behavior increases. The Gaussian curve never reaches 0, and the probabilities of executing non-goal behaviors are derived from the relation between the other non-goal behaviors (i.e. the non-goal utilities are normalized). Consequently when the go-to-goal behavior is eliminated, the frequency of non-goal behavior will not decrease with a Gaussian curve.

We have scored the utility functions again on whether they transition from relaxed to

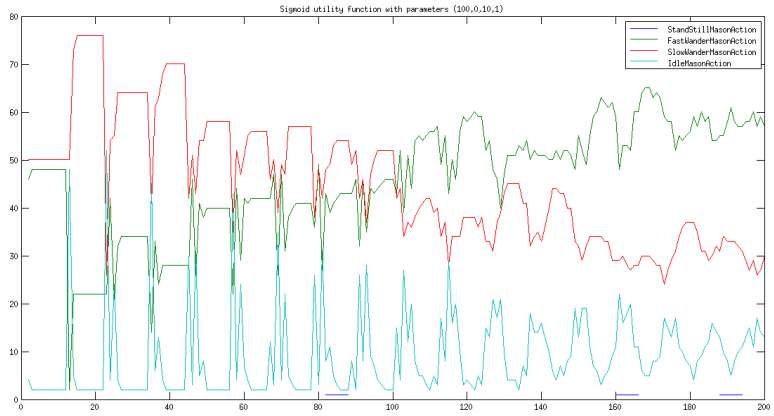


Figure 5.19: Non-goal utility sigmoid with $t_0 = 100$ $\beta = 0$ $\omega = 10$ $\eta = 1$.

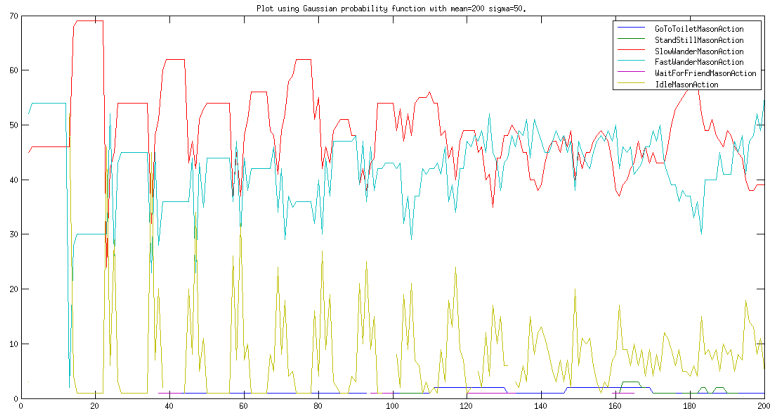


Figure 5.20: Non-goal utility Gaussian with $\mu = 200$ and $\sigma = 50$.

	Relaxed to Hurried
Linear	++
Sigmoid	++
Gaussian	–

Table 5.3: Scoring table of non-goal utility functions when leaving out go-to-goal action and transition

hurried behavior in a realistic manner. The scores can be found in table 5.3. The Gaussian non-goal utility function still results in a poor performance, but with the linear and sigmoid function, the simulation has a better transition from hurried to relaxed behavior than it had when any goal utility function was added.

Chapter 6

Conclusion & Discussion

In this research we managed to create a system with which behavior for an airport-type or deadline-driven scenario can be easily designed with Petri nets and placed in an environment by drawing the areas in which this behavior should take place. In this chapter we will first discuss what answers our research can give to our research question and its subquestions. Secondly we will discuss the limits of our system. Lastly, we will provide a few directions followup research could take.

6.1 Research Questions

In our research we wanted to give an answer on the following question:

How can an intelligent virtual environment for simulated pedestrians be extended to deal with time-restricted destinations?

This will be done by first answering each of the subquestions. We will go through them one by one:

To what extent can pedestrians be simulated realistically?

In our research we compared a number of key behaviors found at Rotterdam Airport with their simulated counterparts in our system. In terms of the trajectories these agents follow, the simulated pedestrians matched up with the trajectories of actual pedestrians.

How do we let the pedestrians make decisions based on the time left to reach the destination?

When deciding on behaviors, the pedestrians are guided by how much time is left after a behavior has been executed (variable τ). This variable is used together with the utility functions to compute the likelihood of a behavior occurring at that moment. From the experiments can be concluded that a Gaussian goal utility function combined with a sigmoid non-goal utility function gives the best result in terms of deadline drivenness and emergent relaxed and hurried behavior.

Is it possible to have emergent behavior based on time restrictions?

As stated in the answer to the previous question, when using the right combination of utility functions, hurried and relaxed behavior does emerge. The slower, more time-consuming behaviors are preferred when the deadline is still far away, while quick behaviors are preferred when more time has passed. From this we can conclude that emergent behavior based on time restrictions is possible with our system.

Is it possible to quickly generate these virtual pedestrians without much tweaking for each environment?

By using Petri nets combined with the situations framework, behaviors can be designed once and used in many different environments. The behavior of the pedestrians will then emerge from the combination of situations defined in the environment and a predefined goal for every pedestrian. In an airport-type scenario, the goal for almost every pedestrian is the same, namely reaching their plane before it leaves. The goal of those pedestrians can therefore be defined once.

6.1.1 The Benefits of Petri nets

In our research we replaced the finite state machines of the situations framework of Sung et al. with Petri nets, but was it necessary to do so? For our purposes it most certainly was. The way in which a sense of deadline was implemented through the use of goal and non-goal utility functions could probably be approximated with finite state machines, but in doing so the potential complexity of implementable behaviors would be limited severely. The usefulness of Petri nets over finite state machines expecially expresses itself when multiple tokens can move around. This property of Petri nets enables the designers to make behaviors where multiple agents are interacting, or where a single agent can keep track of resources. For example, a token could indicate whether an agent has bought a drink, and only when that token would be present, he could actually execute the behavior of drinking.

6.2 Limits of the Deadline Driven Behavior Framework

Modeling pedestrian behavior with our method does have its limits. First of all, the movements of the agents are designed explicitly through linking together basic movements in the Petri nets. As a consequence, interaction with other agents is quite static, and not directly responsive to surrounding agents. Consequently, our method is not particularly suitable for situations in which the pedestrians have to move very close together, such as when a large amount of pedestrians has to move through a narrow space and have to move closer together or form a queue. When pedestrians move very close together, their behavior will become more uniform, and it would be more sensible to look at the group as a whole, and not as individuals. A more suitable approach would then for example be to look at crowds as particles in a liquid, like Moore, Ali, Mehran and Shah have done [16]. Their supposition is that people in crowds seem to move according to the flow, just like particles in a liquid. However, in the type of scenarios our system is made to simulate, it is not a problem that crowd behavior isn't easily designed. When observing the footage, we noticed a lot of individual behavior, or behaviors in pairs, and crowding behavior was virtually absent.

In section 6.1.1 a number of benefits were named to using Petri nets. One of the examples used was the possibility of adding resources to the Petri nets. It is important though to mention the restrictions of this type of use of Petri nets. While it is possible to keep track of the number of items of a specific resource through tokens, it is not possible to have the Petri net react to it in a continuous fashion. Either a place contains the token(s) and a transition can be fired, or the place is empty and the associated behavior is not possible. It is not possible to have an agent react differently depending on how much he has of a resource on a continuous scale.

Add something about dijkstra's algorithm

6.2.1 Future Work

There are a number of directions research regarding our type of framework can take. First of all, the number of behaviors used to simulate the pedestrians on Rotterdam Airport was very limited, so it could be beneficial to add more kinds of behaviors to the scenario. Furthermore, the system could be tested with more complicated Petri net designs. The choice to use Petri nets gives the ability to design very complicated behavior, that takes a lot of factors into account. For example, the fact that Petri nets can contain multiple tokens gives the opportunity to work with resources as mentioned in section 6.1.1.

There are also several ways in which the deadline driven behavior framework can be extended or improved. For example, currently, the probabilities for entering situations are only dependent on how close the agent is to the deadline. So, no matter what time of day it is, the pedestrians will always have the same utility for a certain action. However, in real life, a person's probabilities for certain behavior is also largely dependent on their daily cycle. Ideally, a pedestrian's propensity to execute certain actions, such as eating, should vary dependent on whether the individual has recently executed that action, and their daily cycle. In other words, it would be beneficial to introduce *needs* to the framework. By adding the concept of needs, we would be better able to model the daily flow of people in a typical public area. We can vary the needs according to the time of day and whether this need has been fulfilled recently. It is not definite though whether the system needs to be changed for this to be possible. It might be possible to introduce the concept of needs in a way that it can be designed with our existing system.

Another improvement the system would benefit from would be to change the way the goal utility function is used. In our current setup, the goal utility function sometimes stands in the way of the emergent relaxed and hurried behavior. Cumulatively, the probability of going to the goal becomes high soon in the simulation, unless a goal utility function is used with only very low values. Because there is nothing to be done after the pedestrian reaches its goal, this means they remain idle for a large part of the simulation. It is very likely that this behavior could be improved if the time for a pedestrian to go to its goal is sampled beforehand from a probability distribution function, and only the non-goal behavior is handled by a utility function.

Check all references (figures, papers, etc.)

Bibliography

[2] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[3] O. Burchan Bayazit, Jyh ming Lien, and Nancy M. Amato. Better group behaviors in complex environments using global roadmaps. In *In Artif. Life*, pages 362–370, 2002.

[4] P. Becheiraz and D. Thalmann. A model of nonverbal communication and interpersonal relationship between virtual actors. In *Proceedings of the Computer Animation*, CA '96, pages 58–, Washington, DC, USA, 1996. IEEE Computer Society.

[5] Adriana Braun, Soraia R. Musse, Luiz P. L. de Oliveira, and Bardo E. J. Bodmann. Modeling individual behaviors in crowd simulation. *Computer Animation and Social Agents, International Conference on*, 0:143, 2003.

[6] Jörg Desel and Wolfgang Reisig. Place/transition petri nets. In *Lectures on Petri Nets I: Basic Models*, pages 122–173. Springer, 1998.

[7] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 10.1007/BF01386390.

[8] J. Dijkstra, H. J. P. Timmermans, and A. J. Jessurun. A multi-agent cellular automata system for visualising simulated pedestrian activity. In *in S. Bandini and T. Worsch (Eds.), Theoretical and Practical Issues on Cellular Automata - Proceedings on the 4th International Conference on Cellular Automata for research and Industry*, pages 29–36. Springer Verlag, 2000.

[9] Arthur Gill. Introduction to the theory of finite-state machines. 1962.

[10] T. Hamagami and H. Hirata. Method of crowd simulation by using multiagent on cellular automata. In *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, pages 46 – 52, oct. 2003.

[11] Dirk Helbing, Illes Farkas, and Tamas Vicsek. Simulating Dynamical Features of Escape Panic. *Nature*, 407:487–490, September 2000.

[12] Terry R. Hostetler and Joseph K. Kearney. Strolling down the avenue with a few close friends. In *In Third Irish Workshop on Computer Graphics*, pages 7–14, 2002.

[13] Marcelo Kallmann, Etienne De Sevin, and Daniel Thalmann. Constructing virtual human life simulations. In *Proceedings of the Avatars2000 workshop*, pages 240–247, 2000.

[14] Marcelo Kallmann and Daniel Thalmann. Modeling objects for interaction tasks. In *Proc. Eurographics Workshop on Animation and Simulation*, pages 73–86, 1998.

[15] Michael Köhler, Marcel Martens, and Heiko Rolke. Modelling social behaviour with petri net based multi-agent systems. In *IN: PROCEEDINGS OF THE WORKSHOP MASHO03 AT THE KI*, 2003.

[16] Brian E. Moore, Saad Ali, Ramin Mehran, and Mubarak Shah. Visual crowd surveillance through a hydrodynamics lens. *Commun. ACM*, 54(12):64–73, December 2011.

[17] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 99–108, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[18] Nuria Pelechano and Norman I Badler. Modeling crowd and trained leader behavior during building evacuation. *Departmental Papers (CIS)*, page 272, 2006.

[19] Nuria Pelechano, Kevin O'brien, Barry Silverman, and Norman Badler. Crowd Simulation Incorporating Agent Psychological Models, Roles and Communication. In *1st Int'l Workshop on Crowd Simulation*, pages 21–30, 2005.

[20] Christopher Peters and Cathy Ennis. Modeling groups of plausible virtual pedestrians. *IEEE Computer Graphics and Applications*, 29:54–63, 2009.

[21] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics*, pages 25–34, 1987.

[22] Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 19–28. ACM, 2005.

[23] Barry G Silverman, Michael Johns, Jason Cornwell, and Kevin O'Brien. Human behavior models for agents in simulators and games: part i: enabling science with pmfserv. *Presence: Teleoperators and Virtual Environments*, 15(2):139–162, 2006.

[24] Mankyu Sung, Michael Gleicher, and Stephen Chenney. Scalable behaviors for crowd simulation, 2004.

[25] Shih-I Chang Tsai-Yen Li, Ying-Jiun Jeng. Simulating virtual human crowds with a leader-follower model. IEEE Computer Society, 2001.

[26] Wlodzimierz M Zuberek. Timed petri nets and preliminary performance evaluation. In *Proceedings of the 7th annual symposium on Computer Architecture*, pages 88–96. ACM, 1980.