Deadline Driven Behavior in Intelligent Virtual Environments

Djura S. Smits 5619807

Master's thesis
Credits: 12 EC

Master's in Artificial Intelligence

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Arnoud Visser
Philip Kerbush

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

June 24th, 2010

Change

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 intelligence in machines. The range of methods to attempt this is endless. One of the many subfields 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 learning 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 one would encounter in a public environment.

For the last decade or so, there has been an increased interest in reinforcing security in 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 testing data on the fly. It is beyond the scope of our research to get into what suspicious behavior looks like, so we like to focus on the behavior of

Figure 1.1: People going about their everyday business

"non-suspicious" people. Assuming that so-called "normal" people greatly outnumber terrorists, these suspicious people could be added in later with ease. In other words, we would like to be able to quickly generate crowds of people doing normal, everyday behavior.
Simulation of pedestrians is a widely studied subject. Many models have been created for the various characteristics that arise when multiple agents move around in the same area. Many uses can be thought of for pedestrian simulation. An important purpose for this could be for example to easily create a large amount of "normal" pedestrians (as opposed to "suspicious" pedestrians). Suspicious pedestrians whose behavior is scripted by hand can then be placed in the environment to easily create artificial testing data for suspicious behavior detection algorithms and the like. Furthermore, the addition of simulated virtual pedestrians can add realism to otherwise mostly empty battlefield scenarios.
Another way to look at the behavior of groups of pedestrians is to explicitly specify how they behave when they enter a certain type of situation. In our approach, we would like to be able to specify behavior by drawing situations on an environment. 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.

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 approach results 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.

We validated our model by mimicking certain behaviors found on Rotterdam airport. We chose Rotterdam airport because time restrictions are very prevalent in a departure hall like the one on Rotterdam airport.

So, to summarize, we would like to create a system that enables a user to easily place large amounts of more or less realistically behaving people in an environment. These people should behave in a way appropriate for the environment. Furthermore, the behavior should be guided by the fact that every pedestrian has a fixed deadline for when he has to do his final activity. Ideally, this will lead to a distinction between hurried and relaxed behavior, depending on whether the deadline is near or still far away. In order to facilitate defining the behavior, we will create a tool to easily design these behaviors as Petri nets.

Describe the promise for the utopia of my research

1.1 SIMOBS

SIMOBS is a plugin for the military simulation toolkit VR-forces (`http://www.mak.com/products/vrforces.php` It is a tool for generating and executing battlefield scenarios. SIMOBS is a plugin developed to quickly generate inhabitants of an area by drawing residential areas and plants on the map in VR-forces. The behavior of these inhabitants is determined by *Daily Motion Patterns*, which specify where certain types of inhabitants need to go at specified times. This is the idea that forms the basis for our research.

1.2 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. 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 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?

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. 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. However, this does not limit our search much since it only divides our desired result into two layers; namely the overall structure of different behaviors placed in the environment, and the definition of these different behaviors. When we approach the subject as *designing behavior in an environment*, the search becomes much more narrow and directed.

2.1 Designing the Situation-Dependent Behaviors

In our research, we can divide the subject of "behavior" in two layers. First of all, we have the layer in which individual behaviors are described. However, this information is embedded in a framework that describes how these individual behaviors vary spatially. Let us start with focussing on these individual behaviors. This field knows many approaches. First of all, the focus can lie on the group as a whole. The different members of the group are then often viewed as particles that influence the other group members near them with attracting and repulsive forces. Other approaches view crowds as a group of individuals, and the members are given some kind of simplified psychological model. The motivation behind this simplified model is that

this will lead to complex behavior when many of these simple units are put together in a large crowd. This effect is known as *emergence*.

2.2 Group Interaction

The research of interaction in groups started with Reynolds' boids [**?**] , 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). However, this flocking behavior is more suitable for modelling behavior of 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 large groups by viewing the members as particles.

2.2.1 Global approaches

Many models that have been proposed focus on crowds in panic situations. An example of a model for panic situations is the one proposed by Pelechano et al. [**?**] who divided people in three categories: *trained leaders*, who have complete knowledge about the building, *untrained leaders*, who handle stress well, help others and will explore the building, and *untrained non-leaders*, who might panic.

Maybe sections have to be named differently, or previous reference has to be moved, the approach uses global effects, but also more individual models

Another example of simulating panic situations, can be found in the article of Helbing, Farkas, and Vicsek [**?**]. 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. This model can lead to several effects known to occur in real panic situations.
While these methods give a good insight into the movements in those partic-

ular panic situations, they are less suitable for experiments running over a longer period of time. 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 [**?**] 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 [**?**]. 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. This is the feature that makes this approach stand out from the rest. Because most of the previously mentioned techniques have less sophistication in the path planning of separate portions of the crowds.

However, global roadmaps are not the only way an environment can be divided. Various methods have been developed that use a combination of multi-agent techniques and cellular automata [**?**][**?**]. Here, the behavior stems from a combination of basic multi-agent techniques, 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 $t - 1$.

2.2.2 Smaller Groups and Individual Approaches

The previously mentioned approaches focus on movements of crowds as a whole. 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 few other

pedestrians and very few move around on their own. An important step in this direction has been made by Li, Jeng and Chang [**?**], 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 [**?**]. The walkways have been modeled as ribbons to define the geometry of the surface, and wich 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 [**?**]. 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. 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.
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 [**?**].

2.3 Interaction with Objects

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. This greatly reduces the realism of the simulation. The obvious solution is to extend the knowledge of the pedestrians with instructions about how to interact with these objects. This has been succesfully done for instance by Shao and Terzopoulos [**?**]. 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 invidual is followed for a

long time. The downside to this method is that the behavioral model for the pedestrians has to be specifically crafted for the environment, which will be very 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* [**?**].

2.3.1 Mixed Approaches

The distinction between larger and smaller groups is not black and white, as can be seen in the work of Braun et al. [**?**], who generalized the model of Helbing, and introduced additional features for creating group behaviors, such as family members, dependence level, altruism level, and desired speed of the agent.
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.
Several other methods also have the potential to be extended to incorporate several other methods. For instance, it is very likely that the "situations" framework can be adapted to deal with higher level states, instead of the low-level animation-oriented states that are used now. It will then probably be possible to let situations incorporate certain effects (eg. a repulsive force) instead of simple animations. These effects could be one (or parts) of the previously mentioned other models.
Make sure the text fits better in the rest of the thesis

2.4 Designing the Behavior Structure in the Environment

The approaches that fall under the previously mentioned categories generally focus on the general movement of crowds of pedestrians. However, when we want truly realistic behavior in a regular public environment, these methods do not suffice, because they miss interactions with objects in the environment. There are a few approaches that focus more on environment-

dependent behavior, such as interaction with objects or situation-dependent actions. The obvious solution is to extend the knowledge of the pedestrians with instructions about how to interact with these objects. This has been succesfully done for instance by Shao and Terzopoulos [**?**]. 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 invidual is followed for a long period of time. The downside to this method is that the behavioral model for the pedestrians have to be specifically crafted for the environment, which will be very time consuming.

Fix duplicate text

We will now look at the behavior on the meta-level. A few methods have been developed to enable a user to design environment related behavior. The most obvious solution to specifying the behavior of pedestrians environment-dependently, is to craft a new script for the pedestrians for every new environment. However, this method is very inefficient when we make use of a larger set of environments. Therefore, we would like to be able to place the same pedestrians in different environments without creating a new script every time. One method that deals with this problem uses so-called *smart objects* in order to efficiently create behavior-rich environments. The main feature that makes the smart-objects method so efficient is that the information of how to interact with an object is completely contained within the object itself, and not in the agents. That is why the agents' behavior can easily be enriched by adding new smart objects in the environment.

In the most basic approach, these smart objects take complete control of the agents for a short period of time and let them interact with the object. The big advantage of this method is that the information for interacting with a specific object does not need to be stored in the pedestrians, but is stored in the objects themselves. 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. 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 [**?**]. This model uses five main motivation types: eat, drink, rest, work, and go to toilet. These motivations control the action through a hierarchical decision graph. Information about which objects fulfill these different needs are added to the smart objects.

Another slightly different approach is the use of a situation based control structure (Sung, Gleicher and Chenny [**?**]). 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:

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

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. 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.

2.5 General Behavior Modeling

Finite State Machines

Finite-state machines (FSMs) are behavioral models that are composed of a number of states associated to transitions. A finite state machine moves from state to state by doing sets of actions associated with certain transitions. They are widely used in a variety of applications such as electronic design automation, but also for parsing. Many variations on FSMs exist for a variety of purposes. Finite states can for example be deterministic, or non-deterministic. The latter can also be seen as a representation of a *Markov chain*.

2.5.1 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 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. Basic Petri nets can be described by a five-tuple:

$$PN = (P, T, I, O, M_0) \tag{2.1}$$

which comprises of

- a set of places $P = (p_1, p_2, ..., p_m)$,
- a set of transitions $T = (t_1, t_2, ..., o_m)$,
- a set of input arcs $I \subset P \times T$,
- a set of output arcs $O \subset T \times P$,
- an initial marking $M_0 = (m_{01}.m_{02}, \ldots, m_{0m})$.

Petri nets have been extended in many ways in order to accomodate many different functionalities. The extention that attracts our attention the most is *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:

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

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.
Köhler et al. show us an example of how Petri-nets can be used to model elaborate social situations [**?**]. 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.

Check whether I understood correctly

2.5.2 Advantages of Petri Nets over Finite State Automata

Petri nets hold several advantages over finite state automata. First of all, Petri nets allow for concurrent behavior. This would enable our pedestrians to 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. It would not 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. However, this is not the only advantage Petri nets have over finite state automata. Petri nets come with a standardized way to incorporate a time aspect. For finite state automata, several techniques have been developed, but there is no general consensus over what methods are most suitable. The problem becomes even larger when we would like to incorporate both time and non-determinism. Both the aspects of time and non-determinism are easily incorporated in Petri nets with only small modifications, and can even be mixed with traditional Petri nets. An essential issue which has to be implemented in the method we are going to choose, is non-determinism. Fortunately, this is one of the basic properties of a Petri net. When multiple transitions are enabled at the same time, any of them may fire.
Lastly, availability will not be an issue either, since there are many packages for many languages freely available on the web.

2.5.3 Are FSMs and Petri Nets interchangable?

While finite state automata and Petri nets look very similar, there are some essential differences which make a direct mapping impossible. First of all, Petri nets enable us to work with concurrency. This means that usually, multiple transitions are enabled in one point in time. This could mean that the pedestrian should be able to do multiple activities at once. For example, a token could be created for certain parts of the pedestrian's body, such as for its hands and feet, or upper part and lower part.
Another issue that has to be dealt with is how the nets are going to be attached and detached from eachother. This could be done largely the same as with finite state automata, but Petri nets have the potential to do this in a much more versatile way. For example, situation petrinets could be designed to have different slots for pedestrians so that when multiple pedestrians use the situation at the same time, the pedestrians' Petri nets could be attached to the same situation Petri net at different places, which could be a convenient way to model several kinds of behavior involving multiple pedestrians such as queueing. An example of how this could be modeled can be found in figure **??**. Customer 1 and customer 2 could be slots where a pedestrian's personal Petri net could be attached.
When we take an approach similar to this, there will be an important distinction between the Petri nets belonging to the pedestrians and situations. While there will be many identical pedestrian Petri nets in a simulation, there will be only one Petri net per instance of a situation. This might seem a trivial fact, but it is an important distinction between using finite state automata and Petri nets. When using finite state automata, every pedestrian will get a new instance of a situation's FSM attached to it. However, situation Petri nets will not multiply and will accept several pedestrians in their nets. This will make interaction between pedestrians in a situation a natural property of the system, in contrast to the FSM approach, where it will have to be implemented outside of the situations framework.

2.6 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

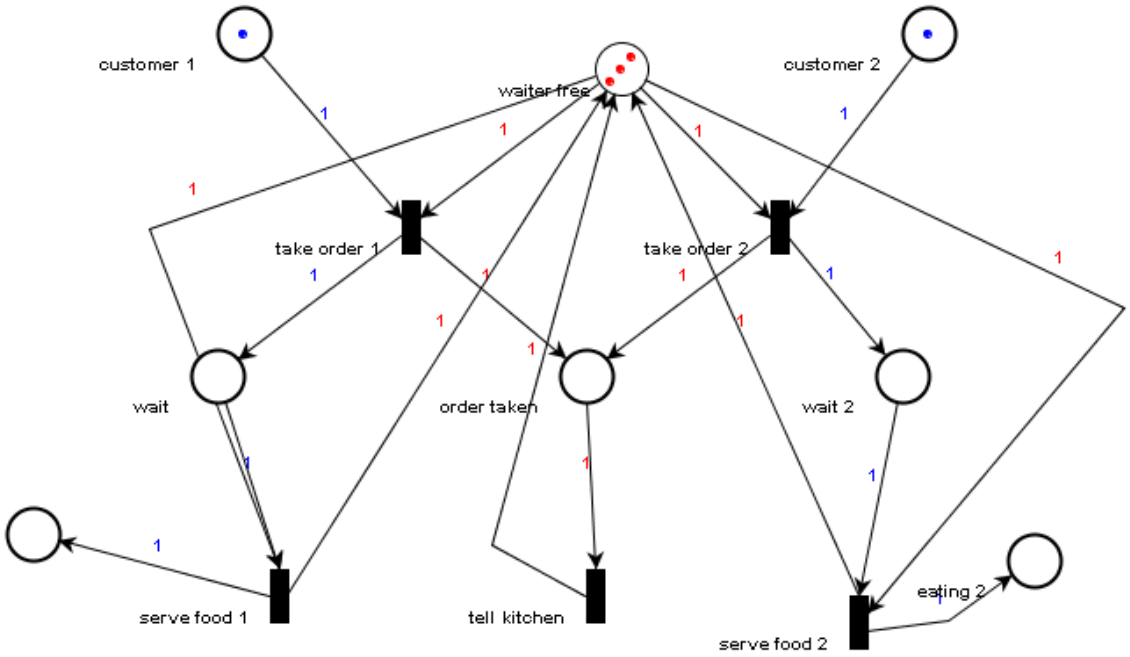


Figure 2.1: An example of a behavioral model of a restaurant expressed in a Petri net

deadline for the goal.

2.7 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. Since most people move in smaller groups, this can lead to realistic effects. 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. 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 implement our system. We decided to base our model around the situations framework. 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. We will do this by replacing the finite state machines with Petri nets. However, we wont work with finite state machines, since these offer too little functionality to enable us to work with time restrictions. That is why we do the behavior modeling a little bit differently in our system. Instead of modeling the system with finite state machines, we describe the system with Petri nets. The toolkit we will use for this is *Platform Independent Petri net Editor 2* (PIPE2). `http://pipe2.sourceforge.net/` In the next section we describe how we decided to use this toolkit.

3.1 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 our system. 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 the modeling language. Unfortunately, at a later time it seemed that the generalized stochastic petrinets did not behave as described in the specifications, so we decided to stick with modified, basic, untimed petri nets. Lastly, the PIPE2 toolkit has a clear, simple gui which can be used to create and modify Petri

nets. This has been the main reason we chose this gui, because we wanted to create tools for easy modeling of behavior. It should require minimal effort to place large amounts of agents in a environment.

Extend introduction.

3.2 Pedestrian Layer

Some drastic changes will be made in order to adapt the situations framework to our needs. Instead of using regular finite state automata, we use Petri nets so that environments in which time constraints are important (e.g. train stations, airports, etc.) can be properly dealt with.

3.3 Assumptions

The method we propose is based on various assumptions which have to be clarified. An important assumption is that the environment the pedestrians have to walk in are designed in such a way that it helps creating realistic behavior. This means that situations have to be defined in such a way that pedestrians will walk into them and act in the appropriate way. This makes our framework probably limited to certain kinds of environments. When an environment does not contain many situation areas or if they are too far apart, it is very likely that the framework does not give a satisfactory result. Another assumption we make is that the pedestrians have to be at a certain place at a certain time. This makes the system more suitable for daily routine type situations rather than cases in which pedestrians are walking around without a proper goal. However, most situations can be described as having a deadline (e.g. eventually, most people have to go to bed), so this assumption is not necessarily very restrictive.
Furthermore, we assume that the behavior of the pedestrians can be described as finite state automata, and that these automata incorporate base states to which transitions loop back, and from which it is always possible to reach the goal state (given enough time is left). We need this assumption in order to create an efficient way of determining which actions can be done before the deadline. Otherwise we would have to search through the finite state automaton in order to find the various ways actions can be tied together.

3.4 Preparations

Because the proposed system needs to run real-time, we refrain from overly complex systems that take too much computing power. However, it is possible to do some computations before running the simulation, and save this for use during the simulation. An important application is the calculation of the distance in time between all the places in a Petri net and the goal place. This distance can easily be computed using the *Dijkstra shortest path algorithm* [**?**]. 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 goal place (the source). This can be very useful when we would like to compute 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. But 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.

The use of Dijkstra's algorithm does limit the use of our Petri nets though. For our simple implementation of Dijkstra's algorithm to be effective, we cannot use the Petri nets' more advanced features such as colored tokens.

Write about problems with dijksta and petri nets, such as multiple tokens/slots, and colored tokens etc.

3.5 Time Planning & Decision Mechanism

The fact that we use Petri nets with timed transitions instead of finite state automata does not necessarily mean our pedestrians are able to deal with time constraints. However, these Petri nets have helped making our problem representable. In order to use these stochastic Petri nets efficiently for making decisions based on time constraints, we have made a number of assumptions.

First of all, we assume there are certain *base places* from which it is always possible to reach the (time constrained) goal. Then, we will compute for every transition that will not take the pedestrian to its goal, how much time it takes to get back to the base state. Then we can check whether the goal place is still accessible from the base state when a certain transition has been taken. We use this information to modify the timed transition rate, so pedestrians are more likely to choose the actions that leave them more time

to reach their goal. 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, since 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 functions for computing the probability of going to the goal, and attempt to assess which function will be most suitable.

About the most important part of the thesis, so has to be extended

Include Pictures

Maybe put the explanation of the difference between actions aan behaviors at an earlier place?

Before we continue it might be necessary to explain what is meant when we talk about behaviors and actions. These two terms are not interchangable. With *actions* we indicate movement of a pedestrian that is described in a single transition in a petrinet. *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.

3.5.1 The Basic Algorithm

Though the probabilities of the various behaviours a pedestrian can have is dependent on what kind of distribution we are going to use, the underlying mechanism will always be as described in algorithm **??**. This algorithm is used to choose one of the various behaviors that are currently available to a pedestrian at a certain timestep. Once it is chosen, this behavior will

be executed be executed in the following timesteps. 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 Petri net rules.
First of all, we subtract the current time from the deadline time of the pedestrian. Before we go any further, it is important to understand how we keep track of the time.

Achterhaal of iedere pedestrian een persoonlijke time heeft of dat er een algemene timer is. Ik ga voorlopig uit van een persoonlijke timer ook beslissen of het volgende detail misschien meer een implementatiedingetje is.

Every pedestrian has its own timer which in our system is implemented to increment by one at every timestep of our simulation engine. When we have computed t_d, we know how much time the pedestrian has left to reach its goal.
Next, the algorithm computes for every situation how much time is left would the pedestrian choose and execute this behavior. This remaining time is then used to compute the probability rate of the pedestrian deciding to choose this behavior. The probability rate is computed with the function ϕ, and can have various distributions. In chapter **??** (*Experiments*) we will try out various options for ϕ to see which will give the most realistic behavior. When the probability rates for all the behaviors have been computed, they are normalized, and based on the resulting probabilities, one of the behaviors is randomly chosen.

Check if it's the right way around (afteller of opteller)

Improve structure

Algorithm 1 The behavior decision mechanism

$t_d \leftarrow$ deadline time - current time
for Every connected situation **do**
    $t_e \leftarrow$ estimated time for situation
    $t_{new} \leftarrow t_d - t_e$
    Probability rate $\leftarrow \phi(t_n ew)$
end for
normalize(probability rates)
select one transition according to the computed probabilities

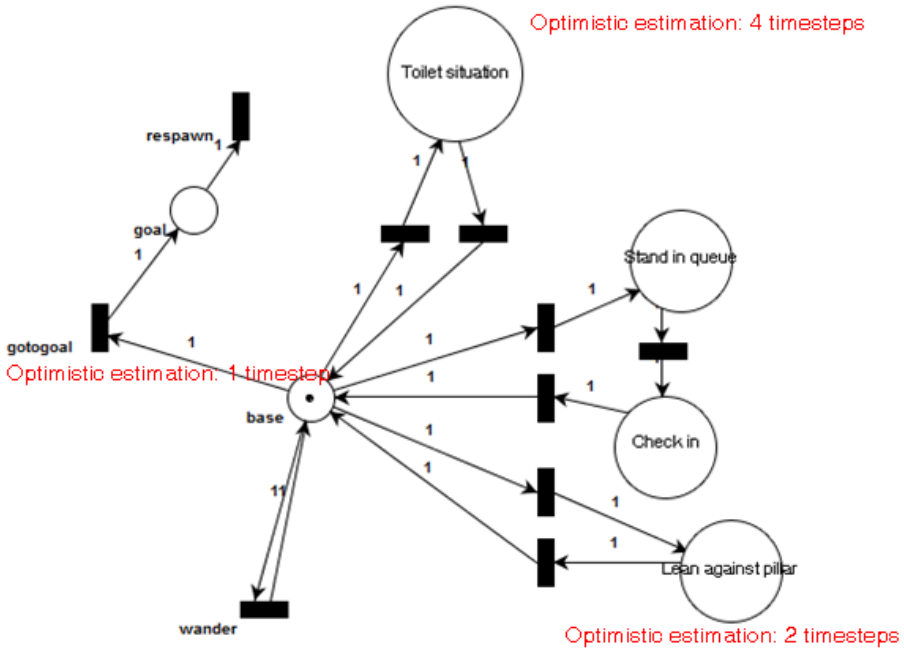


Figure 3.1: Example structure

3.5.2 Examples

Check if the title of this subsection is ok

Figure **??** gives an example of how the pedestrian petrinet is connected to the situations. It also shows the estimation created by the Dijkstra preprocessing algorithm that makes a guess at how much time it takes to complete the situation behavior.

Chapter 4

Experiments

Allthough 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 annotated locations of the visitors. Secondly, we will assess how the the different probability 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.

Specify missing parameters

4.0.3 The Behaviors

Going to the toilet

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. The Petri net used for this can be found in figure **??**.

Checking in

When the pedestrians reach the end of the queue situation, they will enter the check-in situation, in which they will stand in front of the check-in desk for a little while, after which they are free to go again.

Lean Against Pillar

Another recurring behavior we saw is that people lean against the pillars in the hall. This is a type of idle behavior, a variation on the standing still behavior.

Wander

One crucial behavior is missing. How are the pedestrians going to reach the different situation areas? This is only possible if they already have a move to begin with, otherwise they are only going to stay in place. That is why we added the *wander* behavior. This behavior lets the pedestrian turn a random amount of degrees between $-\frac{1}{4}\pi$ and $\frac{1}{4}\pi$ and walk a few steps in that direction.

4.1 Quantitative Experiment

Statistics?

It is very difficult to quantitatively establish whether lifelike behavior has been modeled. However, it is possible to check whether the mechanics of time planning work as predicted. In order to do this, we log the pedestrian's relative time when they arrive at their goal to check how much time they had left until their deadline. If the model works correctly, this time should roughly correlate to how the time probability function has been chosen. We will discuss the various functions we have used to model the probabilities over time.

4.1.1 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:

$$P(t) = \eta \frac{1}{1 + e^{-\omega(t-t_0)}} \tag{4.1}$$

In our experiments, we chose $t_0 = 100$ and $\omega = 100$, so that the middle of the curve lies in the middle of the 200-timestep period which the pedestrians have to reach their deadline. We chose ω to be 100 as well to scale the

function to fit in the 200 timestep interval. For η, we varied this between 0 and 1.

4.1.2 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. We will use this function to check the necessity of the sigmoid function. Our linear function was of the following form:

$$\begin{gathered}\text{If } 0 \leq t \leq n\text{: } P(t) = at + b \\ \text{If } t < 0\text{: } P(t) = P(0) \\ \text{If } t < n\text{: } P(t) = P(n)\end{gathered}$$

However, we do not specify these parameters directly. What matters the most to us, is the value at the start of the linear function($t = 0$, and at the end(t=n). Consequently, we compute a and b as follows:

$$\begin{gathered}a = \frac{endValue - startValue}{endX} \\ b = startValue\end{gathered}$$

4.1.3 Gaussian Function

We also used a Gaussian function to model the behavior. A Gaussian function is defined as follows:

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}} \tag{4.2}$$

Where real constants $a, b, c > 0$. A Gaussian distribution (or normal distribution) is a continuous probability distribution with a bell-shaped probability density function which have a mean and variance as parameters.

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 declines thereafter.

4.2 Qualitative Experiment

Apart from quantitative analysis, we will also qualitatively judge the pedestrians' behavior. We will do this by comparing our modeled behavior with real-life behavior from recordings of Rotterdam airport. We have picked a couple of specific behaviors that we have modeled with our system.

4.2.1 The Dataset

We acquired manually annotated data indicating the tracks of the visitors of Rotterdam airport.

4.3 Results

4.3.1 Results for Sigmoid Function

4.3.2 Results for Linear Function

It turns out a linear function gives significantly different behavior than a sigmoid function. Because there is a constant rise in the probability of walking to the goal, the pedestrians reach the goal very early, since there is a chance to go there every step, and this chance steadily increases. So all pedestrians have reached the goal long before the deadline. This is because the cumulative probability of the pedestrian going to the goal is very high very early.

4.3.3 Results for Gaussian Function

4.3.4 Results

Below you will find the results of the quantitative experiment, where we used a variety of functions to check whether emergent behavior occurs.

In figure **??**, you can see the results of choosing a linear function where the maximum is 0.001, 0.01, 0.02, 0.05 and 1.0 respectively as the goal probability function. We can see here that the pedestrians reach their goal very early, that is, 100 % of the pedestrians reach the goal so early, they could easily have used the remaining time to do a multitude of other activities.

Aggegrate graphs

In figure **??** the same effect can be seen even more exaggerated. Again, we see that the linear probability function causes the pedestrians to arrive at their goal far too early.

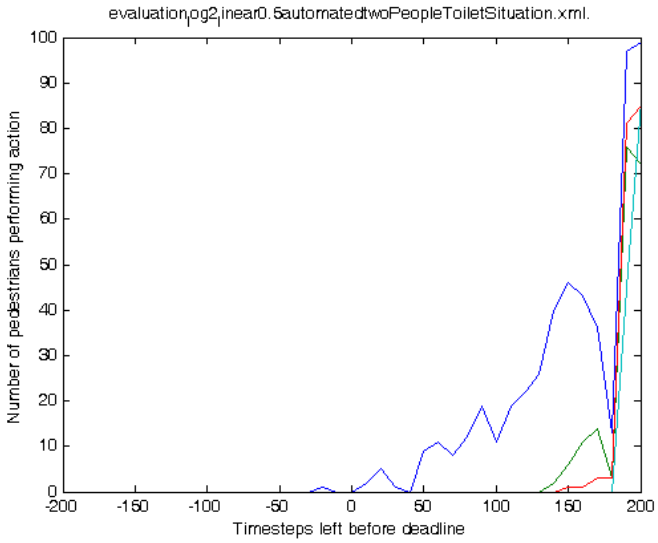


Figure 4.1: Blue: max = 1
green: max. = 0.5
red: max. = 0.2
turqooize: max. = 0.1
purple: max. = 0.01

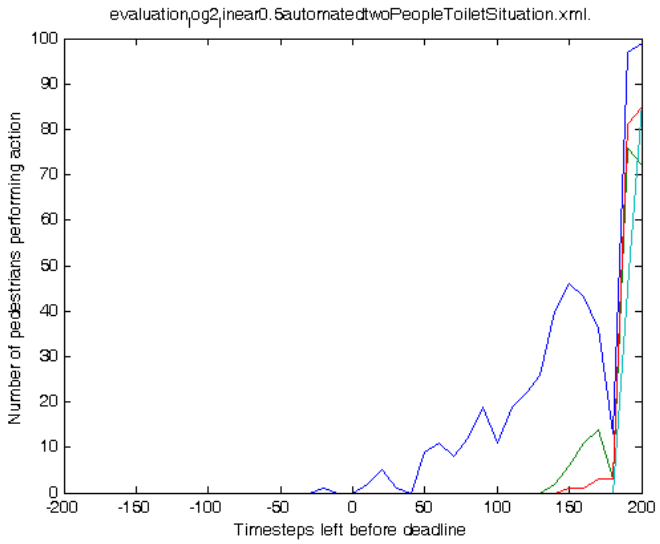


Figure 4.2: Blue: max = 1
green: max. = 0.5
red: max. = 0.2
turqooize: max. = 0.1
purple: max. = 0.01

Aggegrate graphs

In figure **??** we see the results of using a sigmoid for goal probability function with a minimum

find out how to describe, since those values only count in the limit or whatever

of 0 and a maximum of 0.01. Again we see that a maximum probability of 0.01 is low to have the pedestrians reach their goal in time.

We can only say more about the results if the frequencies are divided by the total number of pedestrians in the simulation at that moment. The curve we see in figure **??**, **??**, **??**, and **??** (though to a lesser extent) can be explained by a multitude of causes. First of all, it can be explained by the fact that the number of pedestrians in the simulation diminishes as they finish their goal action. It could also be caused by the probability function increasing, but in this case, that is highly unlikely, because that would result in data quite opposite to the results here. Lastly, it could be that at certain moments, there are more pedestrians already engaged in other multi-step

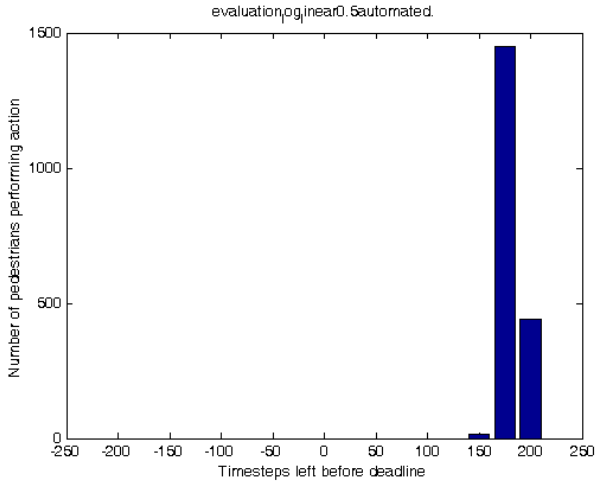
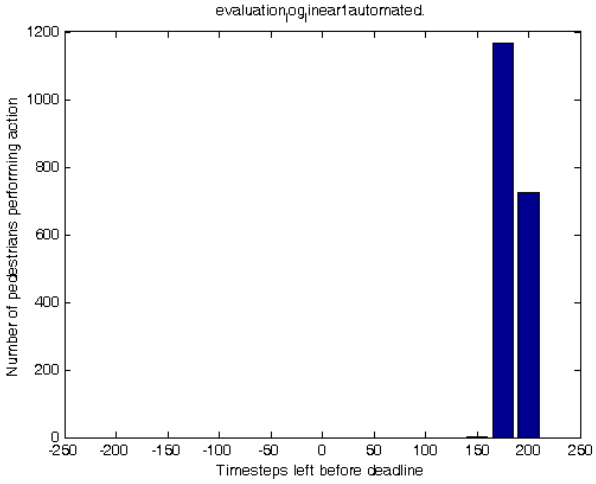


Figure 4.3: evaluation_log_linear0point5automatedpointcsv



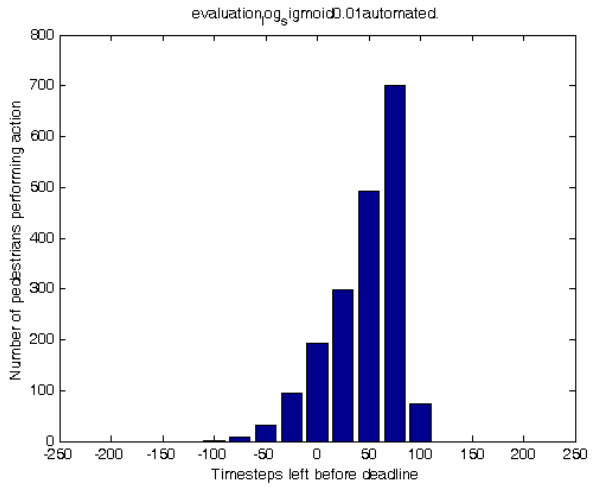
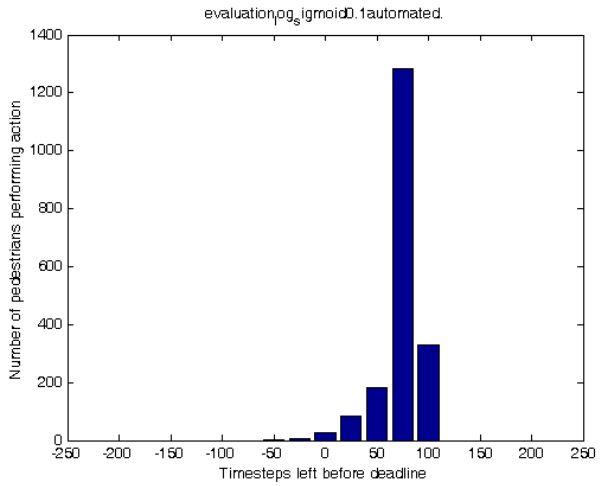


Figure 4.4: evaluation_log_sigmoid0point01automatedpointcsv



captionevaluation_log_sigmoid0point1automatedpointcsv

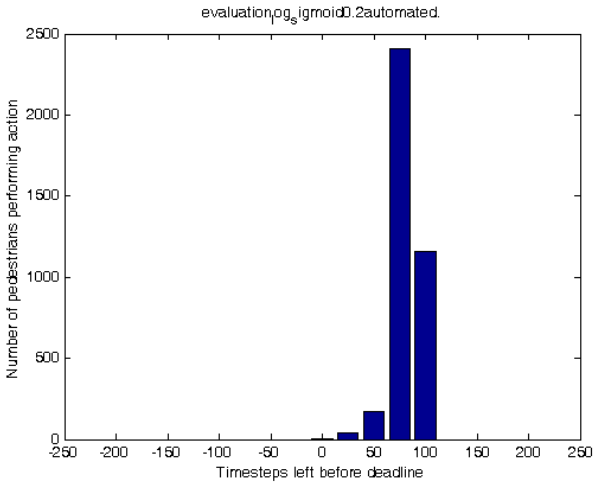


Figure 4.5: evaluation_log_sigmoid0point2automatedpointcsv

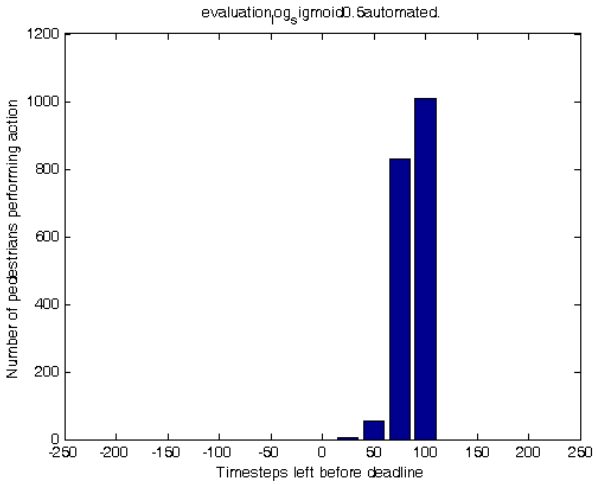
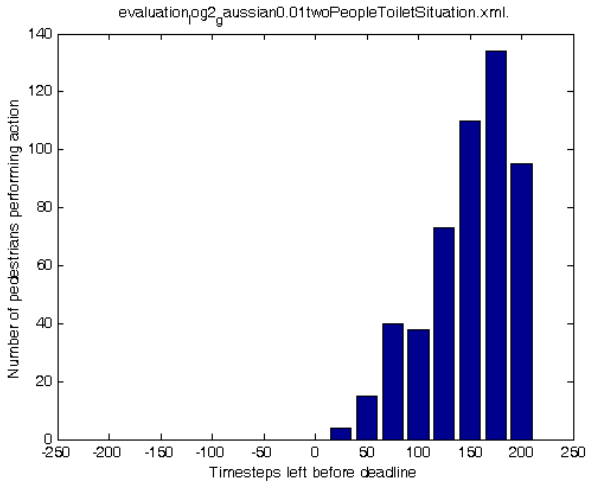


Figure 4.6: evaluation_log_sigmoid0point5automatedpointcsv



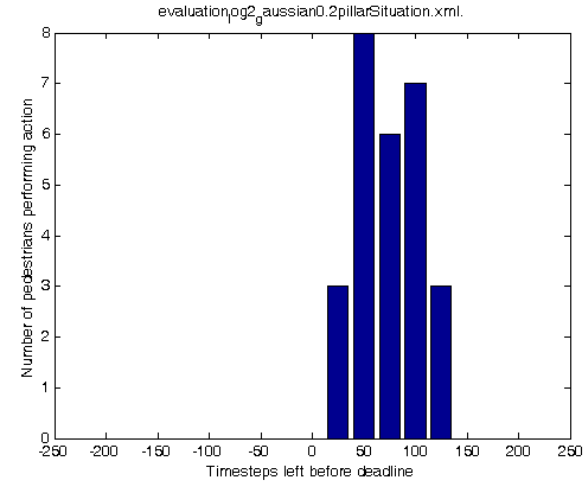
activities.

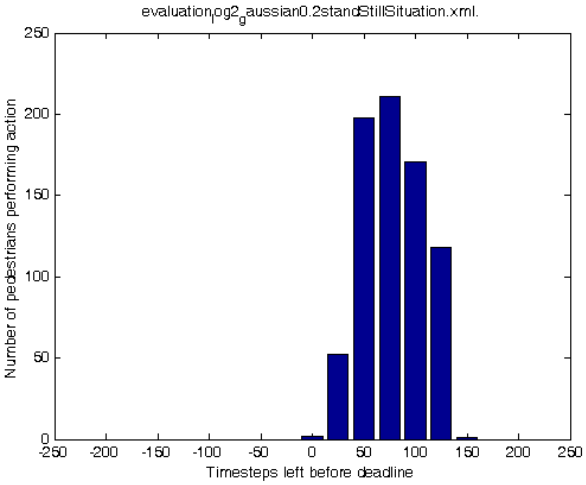
4.3.5 Results 2

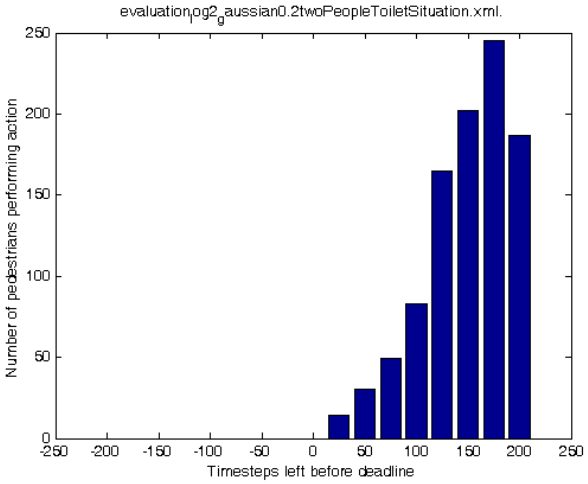
Below follow the results of the other actions, that is, the actions that are not directly connected to the goal probability. The results of these actions have to show that our framework is also capable of generating *emergent* behavior.

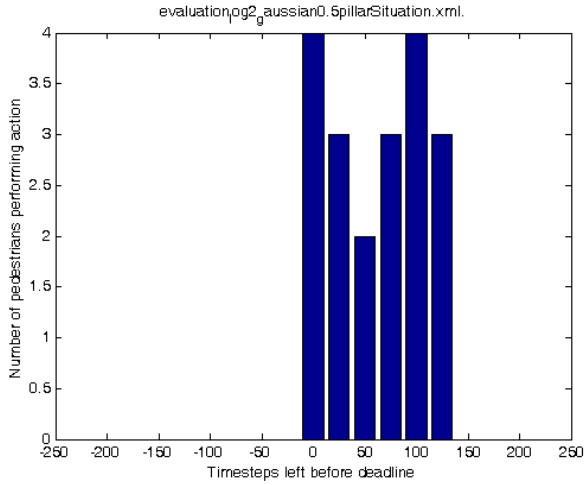
Merge results that were accidentally separated

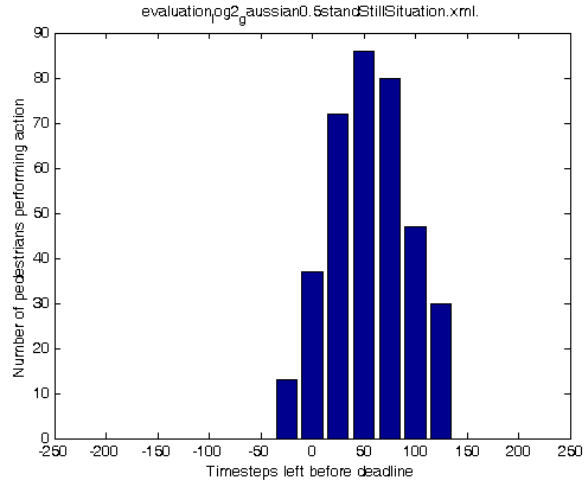
This graph was probably made for the toilet situation that is now saved as twopeopletoiletsituation.xml, but the design of the petrinet is possibly flawed, and could maybe better be replaced by twopeopletoiletsituation2.xml. What also possibly is missing is that the pedestrian could maybe come back after having gone to the toilet.

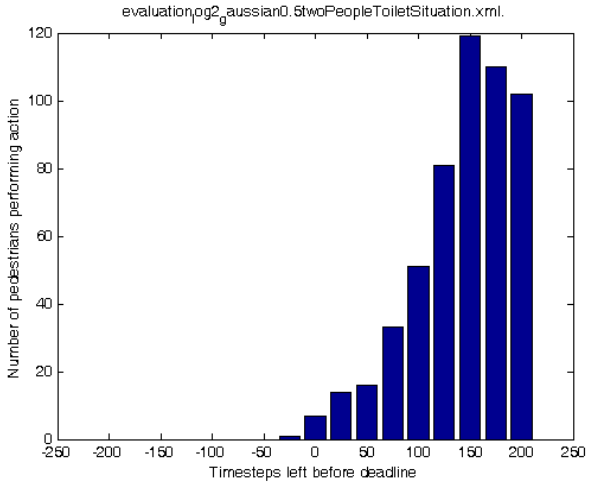

evaluation_log2_gaussian0.2pillarSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
1
2
3
4
5
6
7
8
-250
-200
-150
-100
-50
0
50
100
150
200
250

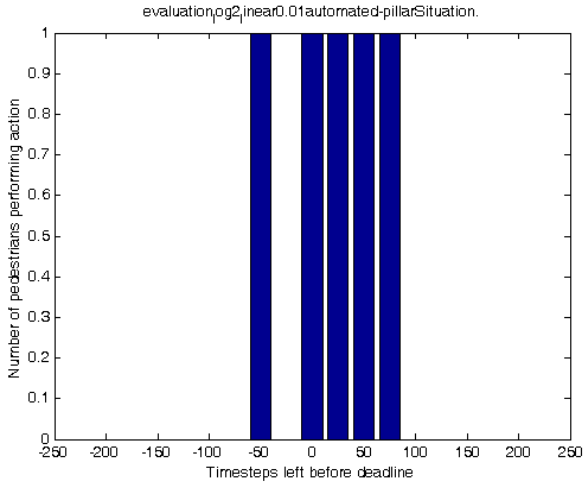

evaluation_log2_gaussian0.2standStillSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
50
100
150
200
250
-250
-200
-150
-100
-50
0
50
100
150
200
250

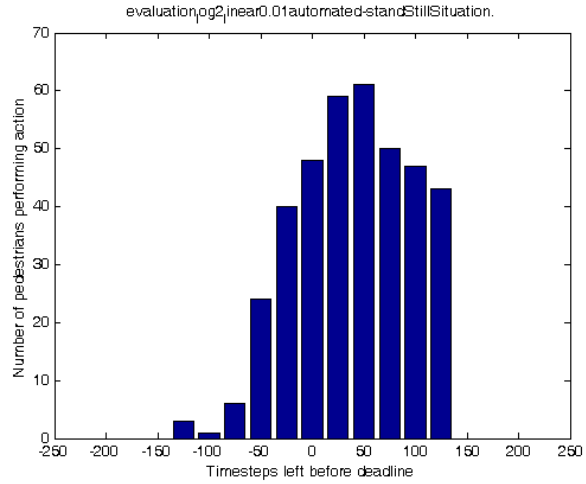

evaluation_log2_gaussian0.2twoPeopleToiletSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
250
200
150
100
50
0
-250
-200
-150
-100
-50
0
50
100
150
200
250

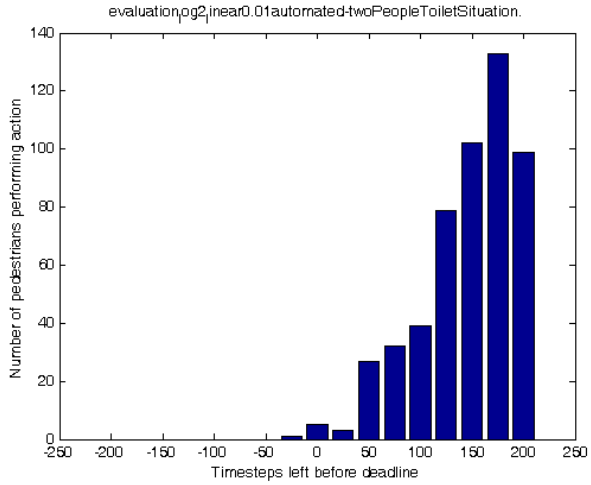

evaluation_log2_gaussian0.5pillarSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
4
3.5
3
2.5
2
1.5
1
0.5
0
-250
-200
-150
-100
-50
0
50
100
150
200
250

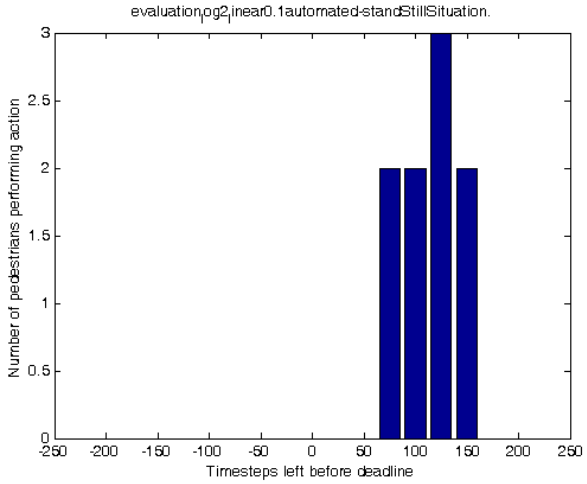

evaluationlog2gaussian0.5standStillSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
10
20
30
40
50
60
70
80
90
-250
-200
-150
-100
-50
0
50
100
150
200
250

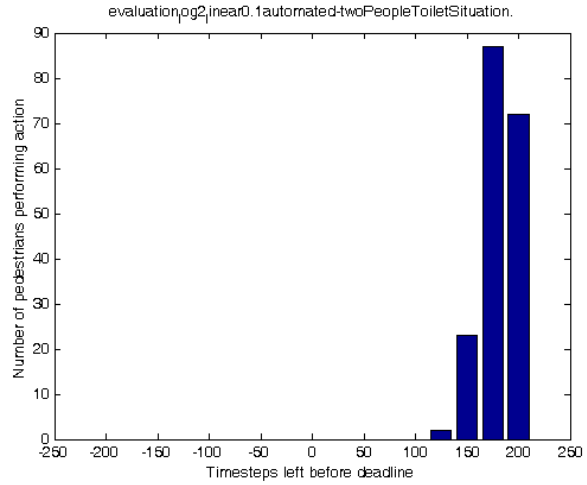

evaluationlog2gaussian0.5twoPeopleToiletSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
20
40
60
80
100
120
-250
-200
-150
-100
-50
0
50
100
150
200
250

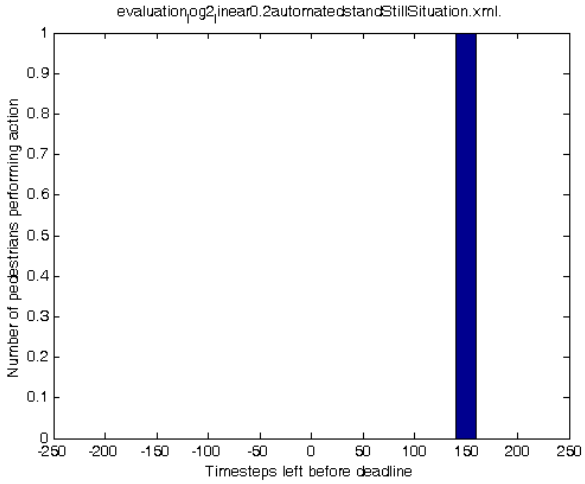

evaluation_log2_linear0.01automated-pillarSituation.
Number of pedestrians performing action
Timesteps left before deadline
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-250
-200
-150
-100
-50
0
50
100
150
200
250

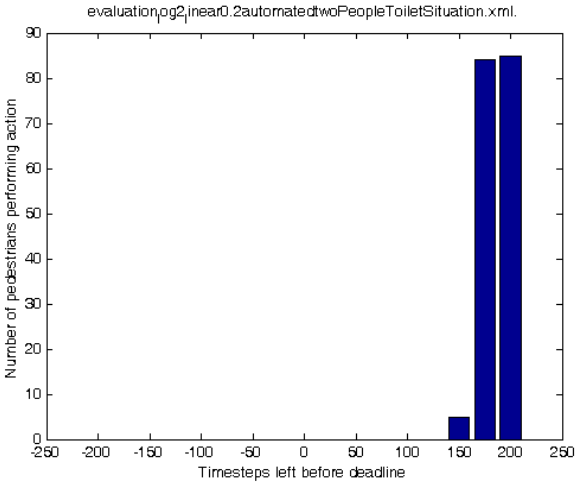

evaluation_og2_inear0.01automated-standStillSituation.
Number of pedestrians performing action
Timesteps left before deadline
0
10
20
30
40
50
60
70
-250
-200
-150
-100
-50
0
50
100
150
200
250

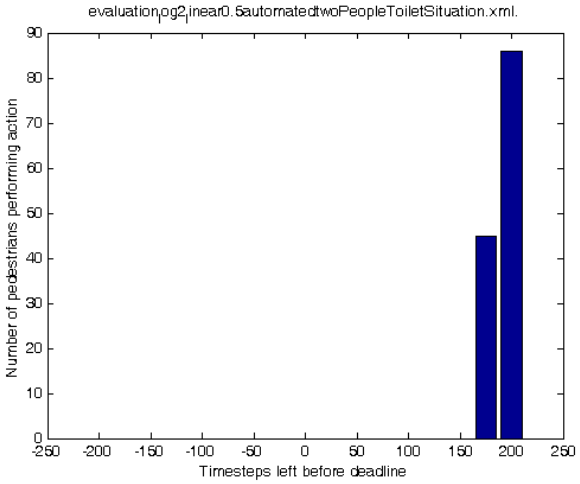

evaluation_og2_inear0.01automated-twoPeopleToiletSituation.
Number of pedestrians performing action
Timesteps left before deadline
0
20
40
60
80
100
120
140
-250
-200
-150
-100
-50
0
50
100
150
200
250

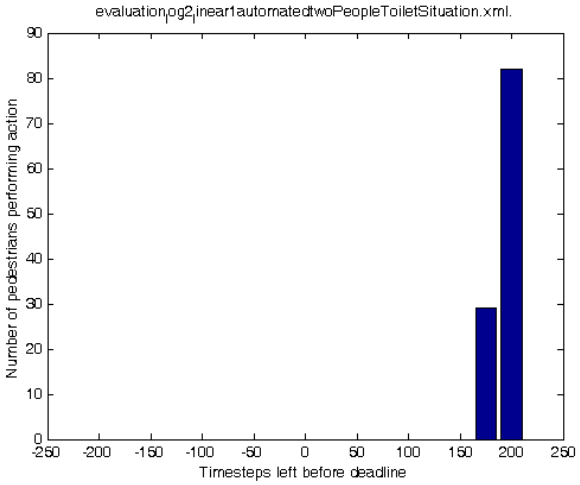

evaluation_log2_linear0.1automated-standStillSituation.
Number of pedestrians performing action
Timesteps left before deadline
0
0.5
1
1.5
2
2.5
3
-250
-200
-150
-100
-50
0
50
100
150
200
250

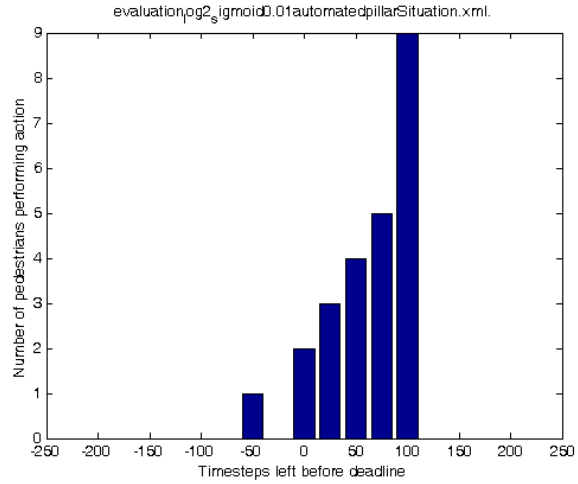

evaluation_log2_linear0.1automated-twoPeopleToiletSituation.
Number of pedestrians performing action
Timesteps left before deadline
0
10
20
30
40
50
60
70
80
90
-250
-200
-150
-100
-50
0
50
100
150
200
250

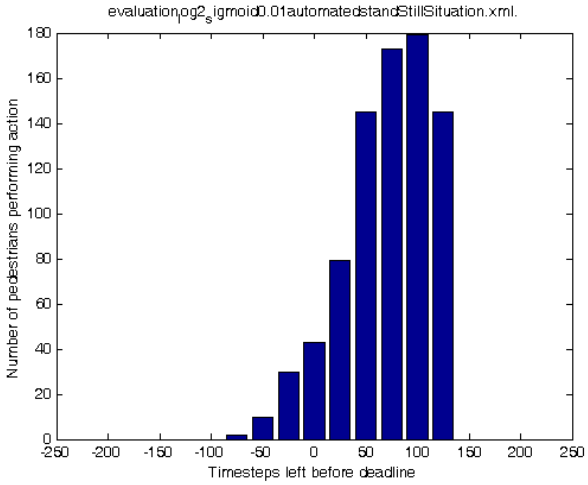

evaluation_log2_linear0.2automatedstandStillSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-250
-200
-150
-100
-50
0
50
100
150
200
250

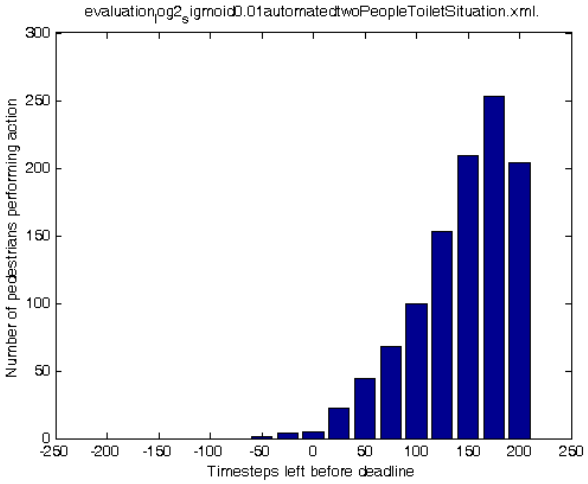

evaluation_log2_linear0.2automatedtwoPeopleToiletSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
90
80
70
60
50
40
30
20
10
0
-250
-200
-150
-100
-50
0
50
100
150
200
250

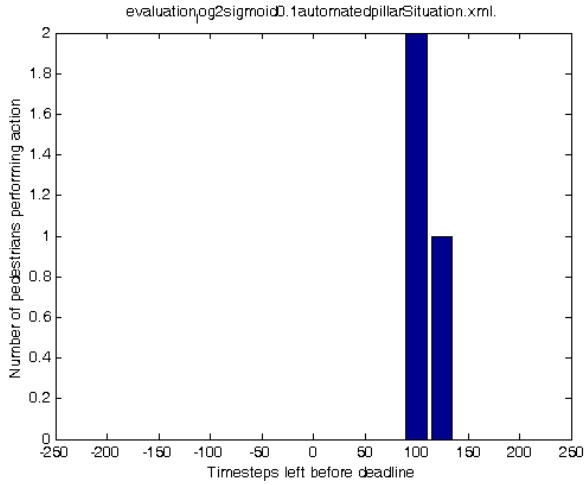

evaluation_og2_inear0.5automatedtwoPeopleToiletSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
10
20
30
40
50
60
70
80
90
-250
-200
-150
-100
-50
0
50
100
150
200
250

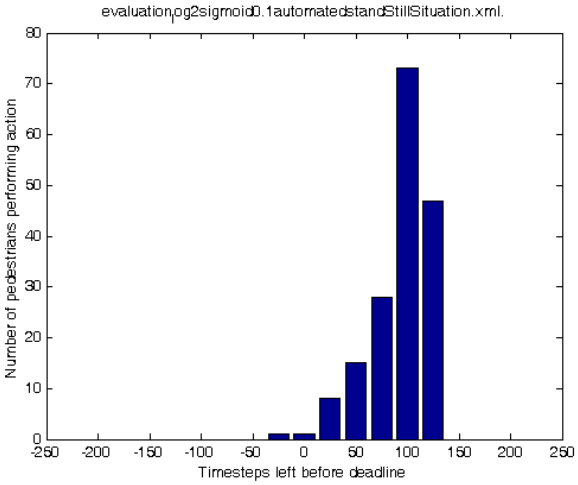

evaluation_og2_inear1automatedtwoPeopleToiletSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
10
20
30
40
50
60
70
80
90
-250
-200
-150
-100
-50
0
50
100
150
200
250

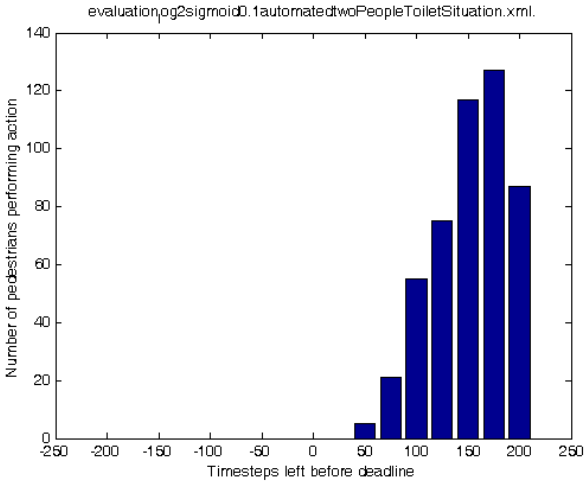

evaluation_log2_sigmoid0.01automatedpillarSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
1
2
3
4
5
6
7
8
9
-250
-200
-150
-100
-50
0
50
100
150
200
250

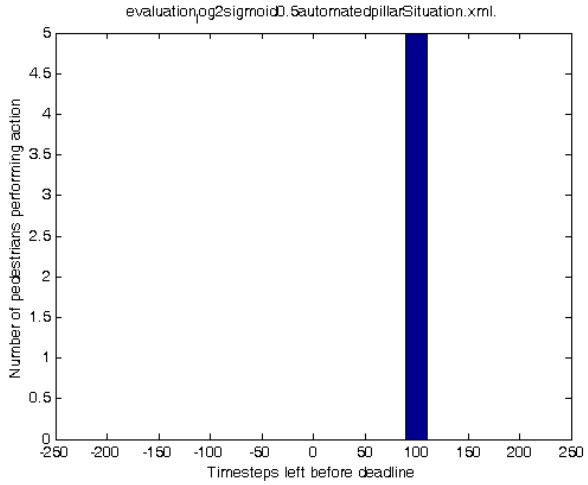

evaluation_log2_sigmoid0.01automatedstandStillSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
20
40
60
80
100
120
140
160
180
-250
-200
-150
-100
-50
0
50
100
150
200
250

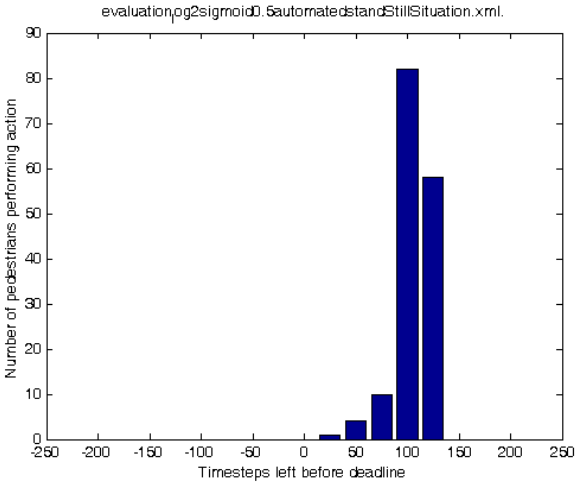

evaluation_log2_sigmoid0.01automatedtwoPeopleToiletSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
300
250
200
150
100
50
0
-250
-200
-150
-100
-50
0
50
100
150
200
250

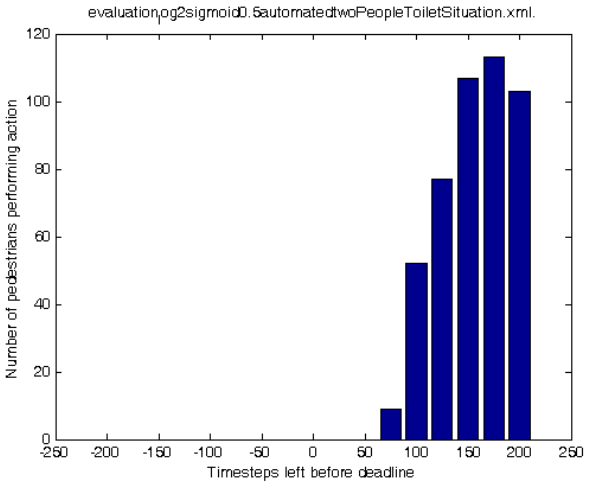

evaluation_log2sigmoid0.1automatedpillarSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-250
-200
-150
-100
-50
0
50
100
150
200
250


evaluation_og2sigmoid0.1automatedstandStillSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
10
20
30
40
50
60
70
80
-250
-200
-150
-100
-50
0
50
100
150
200
250


evaluation_og2sigmoid0.1automatedtwoPeopleToiletSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
20
40
60
80
100
120
140
-250
-200
-150
-100
-50
0
50
100
150
200
250


evaluation_og2sigmoid0.5automatedpillarSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
-250
-200
-150
-100
-50
0
50
100
150
200
250


evaluationlog2sigmoid0.5automatedstandStillSituation.xml.
Number of pedestrians performing action
Timesteps left before deadline
90
80
70
60
50
40
30
20
10
0
-250
-200
-150
-100
-50
0
50
100
150
200
250



4.3.6 What can we derive from these results?

From these results, we can see that not only does the changing of the goal probability function affect the time at which the agents decide when to move to the goal, but also the time at which other actions are executed. This shows that our framework is able to generate emergent behavior. When the probability of going to the goal is high, actions that take longer to finish have a very low frequency.

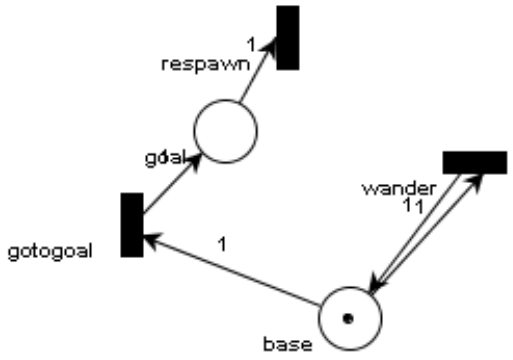


Figure 4.7: The basic petrinet for the pedestrian model for Rotterdam airport

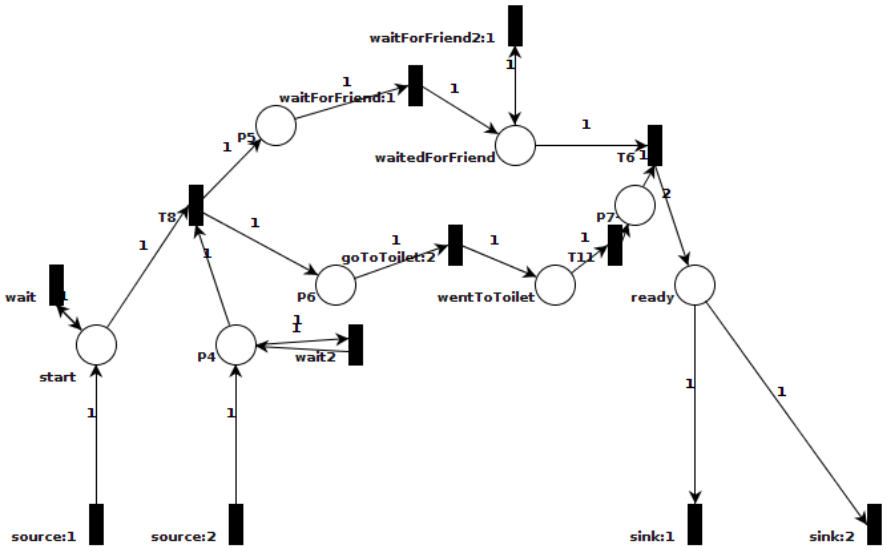


Figure 4.8: Behavior model for the toilet situation

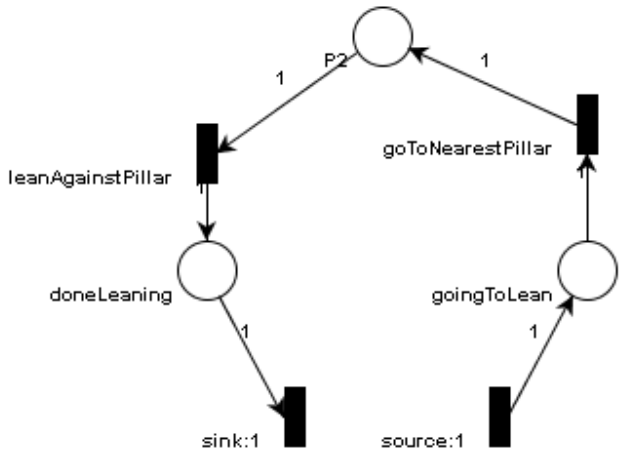


Figure 4.9: Behavior model for the pillar situation

Chapter 5

Conclusion & Discussion

The results are different from what we expected, but logical nonetheless. We expected the sigmoid function to have the ideal curve to reflect the behavior of the pedestrians in Rotterdam airport but

5.0.7 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 [**?**]. Their supposition is that people in crowds seem to move according to the flow, just like particles in a liquid.

5.0.8 Optimization

It is very well possible that the results found in our research are far from the optimal results we could have gotten with our framework. Machine learning could be used to attain the optimal deadline driven behavior. However, this is beyond the scope of our research.

5.0.9 Future Work

There are several ways in which the deadline driven behavior framework can be extended. 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 probability to do a certain action. However, in real life, a person's probabilities for certain behavior is also largely dependent on their daily cycle.

Finding the optimal parameters

Needs

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, we would like to introduce *needs* to the framework. By introducing 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.

Appendix A

Appendix A: Implementation

In the appendix we will go into some of the details of implementation of the deadline driven behavior framework. We will also give some directions about how to use our implementation to simulate your own scenario, or to extend it yourself.

A.1 Implementing the Framework

sectionExtending PIPE2

A.1.1 Sources, Sinks, Slots

In order to facilitate transporting tokens between Petri nets, we created classes of transitions with added functionality to keep track of how these nets are connected. These transitions are called *sinks* and *sources*. A source is a transition that has no incoming connections, only outgoing, with the result that this kind of transition produces new tokens without consuming any, increasing the total amount of tokens in the Petri net. A sink is exactly the opposite: it only consumes tokens and does not produce, effectively decreasing the amount of tokens in the network. Subsequently, we can use these special classes of transitions to transport tokens from one net to another. In order to keep track of how the nets are connected, we pair a sink with every source transition. These pairs are put together in a class called *Slot*.

A.1.2 Additional Modifications

A.2 MASON

A.3 Connecting MASON and the Situations Framework

We deliberately divided our framework into two main modules. Because the situations framework has been implemented with the prospect of interfacing it with other simulations we decided to have the two main units communicate through socket connections. We have the situations side keep track of a thread pool. Every time a pedestrian from the MASON side connects to the situations side, a check is done whether this MASON pedestrian has connected to the Situations before. If this is not the case, another thread will be created to handle the connection to this new pedestrian. Figure **??** shows the global structure. Here you can see how the two interact.

Finish this part about MASON

- Blabla thread pool
- blabla 1 kant is niet multithreaded maar whatever
- ...

Appendix B

Appendix B: Results

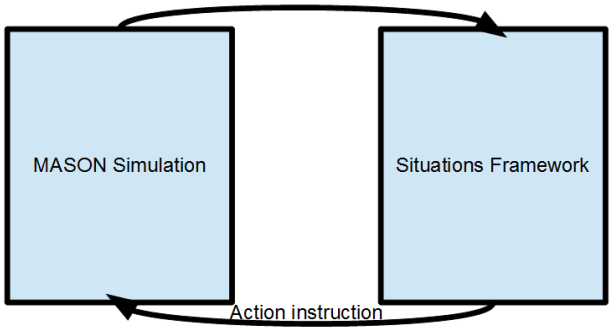


Figure B.1: The global workings of our framework

Bibliography

[1] Wei Shao A and Demetri Terzopoulos B. Autonomous pedestrians.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

[9] 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.

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

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

[12] Michael Kohler, 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.

[13] 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.

[14] 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.

[15] 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.

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

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

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

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

Check all references (figures, papers, etc.)