Assessing the viability of crowd simulation and evolutionary algorithms for solving problems in the field of cognitive psychology

Bram van Dijk, 2012-2013

Universiteit Utrecht
Master toegepaste cognitieve psychologie

Assessing the viability of crowd simulation and evolutionary algorithms for solving problems in the field of cognitive psychology

Bram van Dijk, 3345238
07-04-2013

Naam begeleider vanuit programma
Hinze Hogendoorn

Dagelijkse begeleiding verzorgd door
Gerard Vreeswijk

Naam tweede beoordelaar vanuit programma

Abstract

Cognitive psychology is an interdisciplinary field which uses methods from many other scientific disciplines. In this thesis, the combining of crowd simulations and evolutionary algorithms is assessed as a new method for solving certain problems in the field of cognitive psychology. A proof of concept for this method is given by applying it to a token problem "Determine the optimal configuration of trash cans at De Uithof", where "De Uithof" is part of the Utrecht University campus. The conclusion is that the method returned promising results and that it can be extended to solve many other problems in the field of cognitive psychology.

1. Introduction

The field of cognitive psychology has been around since 1967 (Neisser [17]) and characterises people as dynamic information-processing systems whose mental operations may be described in computational terms. This point of view partially arose from the field of artificial intelligence, where scientists such as Turing sought to create machines which could replicate (human) intelligence. Since Neisser coined the term, cognitive psychologists joined forces with computer scientists and from this collaboration methods evolved with which cognitive psychologists can now make computational representations of humans and use these representations to solve real world problems. Examples range from the lexical database WordNet (WordNet [26]), which models language as a semantic network, to Neo-Piagetian theories (Mora [16]) which seek to explain cognitive development. Conversely, computer scientists have also enlisted the help of cognitive psychologists to help in their field of interest. Clear-cut contemporary examples are the research regarding usability (such as the eye-tracking research of Poole & Ball [20]) and the research which seeks to optimise human-computer interfaces (Encyclopedia of Human Computer Interaction [9]).

One of the more recent area's where computer sciences and cognitive psychology overlap is the study regarding the simulation of human crowds ("crowd simulation"), which will also be the focus of this thesis. Accurately modelling and/or predicting the behaviour of human crowds can be done for many reasons (Epstein [7] provides an overview) but Ballinas-Hernández [1] provide one of the most eloquent justifications for the research regarding crowd simulation: "*Understanding collective human motion is a key issue in designing and developing urban spaces and pedestrian facilities. [..] Crowd models are [..] valuable tools to gain insight into the behavior of human crowds in both everyday and crisis situations."*

Other researchers take a different approach, such as Henein & White [11], who start their introduction by noting *"this year alone over 250 pilgrims were crushed during the Hajj, and over 80 were crushed at the lantern festival in Beijing".* Their research is therefore devoted to a crucial human behaviour in crowds called local pushing. By properly modelling this behaviour, they hope to prevent terrible tragedies like these from happening again. It should be noted that, although the motivations of [1] and [11] were quite different, they were both conducted by computer scientists. Conversely, Kaminka & Friedman [13] provide an excellent example of a study regarding crowd simulation conducted by cognitive psychologists.

Crowd simulation is not only studied by cognitive psychologists and computer scientists. A clear example of this are the methods devised at the theoretical physics department of the University of Cologne (Schadschneider [21]), upon which Burstedde et al. [4] expanded by using crowd simulation as a tool to provide solutions to a cognitive psychology-related problem: compare different floor plans for a room to find which one will be evacuated the quickest in the case of an emergency. To accurately represent such an evacuation, many socio- and psychological factors have to be incorporated in the simulation to ensure its ecological validity. Completing the circle, finding an optimal floor plan from a massive number of possible floor plans is a problem where optimisation algorithms such as evolutionary algorithms can be used, which are common tools for any computer scientist.

This thesis will not focus on the differences between the various approaches regarding crowd simulation. Instead, it will take the point of view of a cognitive psychologist and try to diminish these differences by providing a proof of concept for a combination of both crowd simulation and optimisation algorithms. Due to this point of view, some concepts are explained in more detail than a computer scientist might expect and the problem which this thesis will address is a clear-cut example of a matter a cognitive psychologist would seek to solve: "Devise the optimal configuration of trash cans at De Uithof". Note: "De Uithof" is part of the Utrecht University campus.

Classical methods of solving such a problem would be through trial and error: place the trash cans in a different configuration and see whether it is superior to the old one. Though the new configuration might be constructed through scientifically justifiable heuristics, checking it is still a slow and expensive progress. If a simulation can be constructed which allows for a biologically plausible reproduction of the trash behaviour of humans, this will provide a less intrusive, not to mention less resource exhaustive, alternative for determining whether a given configuration is better than the present one.

However, this research does not seek to either virtually represent De Uithof as accurately as possible, or to provide the best solution to this problem. Instead, it will use this problem to demonstrate various optimisation techniques called evolutionary algorithms. Evolutionary algorithms are widespread throughout the computer sciences, where they are used to provide approximations to problems which have a vast amount of possible solutions and where it is unclear what the consequences of slight changes to a possible solution are. This thesis will show how they can be adapted to help solve problems in cognitive psychology as well.

Before evolutionary algorithms could be applied in this context, a virtual simulation of the real world had to be created. This simulation needed to contain every aspect of the real world which is relevant to the problem being investigated. This meant that the simulation needed a way of representing De Uithof, its inhabitants and their trash. Furthermore, it needed functions which made sure these representations could interact with each other. Though the ones used in this thesis were rather simple, these functions can become highly complex, eventually mimicking pedestrian behaviour and also including factors such as smokers, cyclists and cleaners. Figure 1.1 shows a satellite photo of the part of De Uithof which was under investigation (taken from BingMaps [2]).

Figure 1.1 The area for which the optimal configuration of trash cans is investigated.

NetLogo (see [18]) is a programming language which was able to virtually represent every aspect of De Uithof accurately enough for this proof of concept. Furthermore, it is used in many related scientific researches (Montes [15]; Kaminka & Friedman [13]; Tisue & Wilensky (1) [22]; Tisue & Wilensky (2) [23]) and therefore was therefore chosen to be the language in which the simulations would be created. Many other functions had to be created, such as methods for 'spawning' (creating an object in a virtual world) new pedestrians. Most of these functions are described in further detail in the "Methods" section. When finished, the simulations had to be implemented in the fitness function of evolutionary algorithms (EA's). Many great textbooks have been written about EA's (such as Introduction to Evolutionary Algorithms [27]), and therefore this thesis will only briefly explain the basic concepts.

An evolutionary algorithm works using the principles of evolution. It starts with a digitally represented population of genes, calculates their fitness (see below) using a fitness function and then gives genes with a higher fitness a bigger chance to reproduce, leading to a new population (also known as the next generation). Using these principles, the average fitness of the population increases. In the context of crowd simulation, a gene would be a certain configuration of variables, such as a configuration of trash cans on De Uithof. The fitness function would be a crowd simulator which takes a configuration and calculates how well it solves a problem, measured in 'fitness' (section 3.6 shows various fitness measures). Finally, a reproduction function takes a number of configurations (preferring the ones with a higher fitness) and uses these to create a new population. Usually it does so by combining and mutating existing configurations, but more advanced EA's can use other methods as well.

When writing an evolutionary algorithm three distinct considerations should always be made. First, the type of evolutionary algorithm that is to be used. Second, how the genes are represented (known as the "gene representation"). Finally, it should be determined when 'enough is enough': when is the fitness of a solution good enough to stop the evolutionary algorithm. The first two factors are highly dependent on each other: some EA's work better on certain representations and vice versa. Both of them can differ in programming complexity and space-/time-specifications. The third factor is more of a meta-point: if it is an optimisation problem and the resources are limited, much time should be spent on the theoretical background of the EA used. However, if the problem is merely "improve the present situation by X%", a simpler solution might suffice.

As stated before, EA's are best used when there are a large number of possible solutions to a problem and it is unclear what part of a solution influences its effectiveness. For instance, there are many possible locations for trash cans on De Uithof, but it is hard to clearly define what a 'good' location for a trash can is. This is because the effectiveness of a trash can is not only determined by its location but also by how many trash cans surround it. Finding an optimal configuration of trash cans is therefore a highly complicated task.

Should the EA's prove successful, it is easy to see how this method could be extended to many other problems in the field of cognitive psychology, such as finding optimal configurations of emergency exits, bus stops, food vendors, street lights etc. Therefore, this thesis will not just provide a different application of crowd simulation but an entire new method of approximating the optimal solutions to real world problems: simulate the real world using crowd simulation and then use an evolutionary algorithm to find a (near-)optimal solution.

2. Methods

2.1 Representing De Uithof

Figure 1.1 shows De Uithof. Ideally, a satellite photo like this would be imported straight into crowd simulation software. However, since the simulation software also requires

its agents to interact with the environment, a photo like this needs some processing before it can be used. In figure 2.1 and 2.2, different representations of the same area are shown.

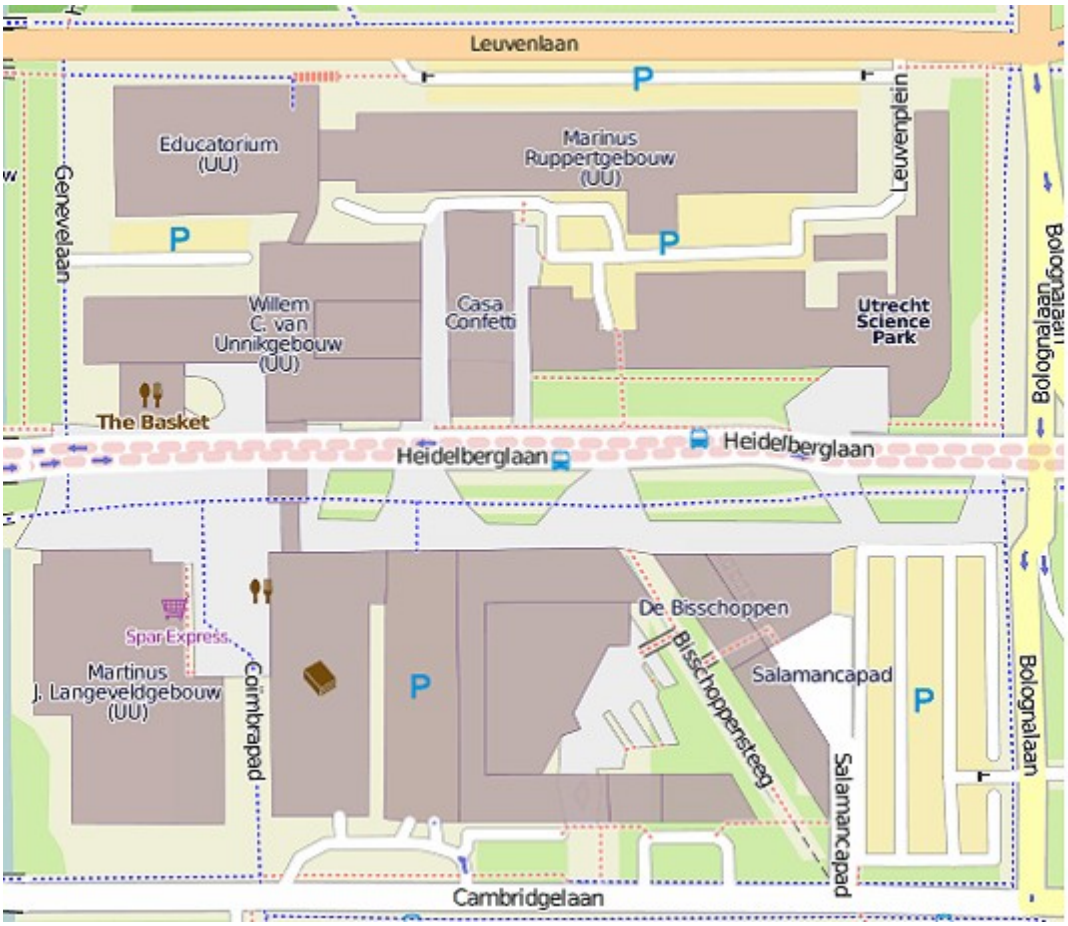

Figure 2.1 Another way of representing De Uithof (taken from openstreetmap.org [19]). It is important to notice that this representation combines an aerial view with ground floor data: this is shown by the Heidelberglaan running 'through' a building. In real life, this is due to a sky way connecting the library to the Willem C. van Unnikgebouw.

When importing geographical data into NetLogo, distinctions between different kinds of terrain can only be made through colour. This is why the representation in figure 2.2 was created. Naturally, more detail can be added (such as adding street lights, or the supports of the sky way) and this should be done for any applied research if they are relevant to the problem being investigated. Since the aim of this thesis was to provide a proof of concept, features such as these were omitted.

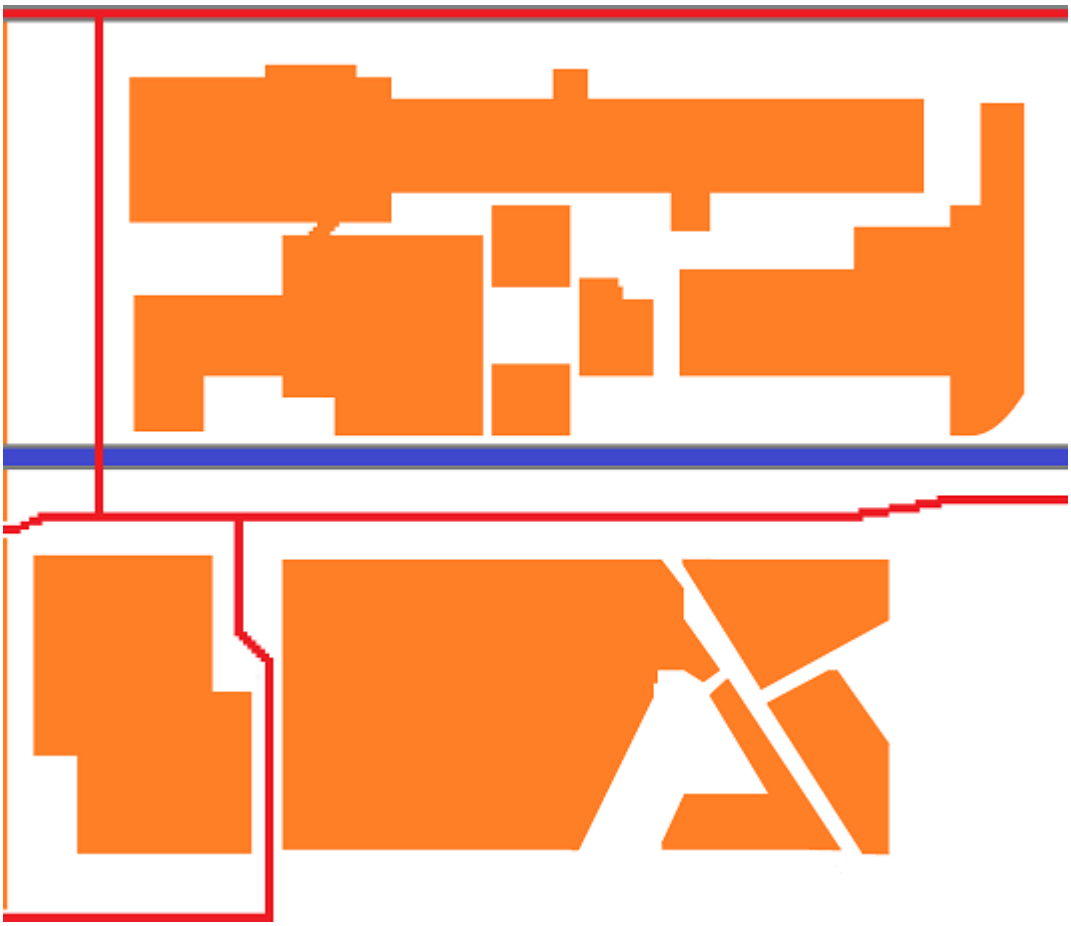

Figure 2.2 The way De Uithof is represented in NetLogo. It only makes a distinction between 'accessible' (non-orange) versus 'non-accessible' (orange) terrain on ground level. Any sky ways are ignored. The bus and bicycle lanes are merely for aesthetic purposes.

The final simulation was on a 3:1 scale. This meant that every patch (the smallest discernible unit of terrain in NetLogo) represented a three by three meters square of the real world. Since the area was 360 by 321 meters, the NetLogo grid was 120 by 107. As seen in figure 1.3, the only distinction made between terrains was 'accessible on ground level' (non-orange) versus 'non-accessible on ground level' (orange). Any sky ways were ignored. The bus and bicycle lanes were merely for aesthetic purposes.

Every patch had a number of variables: a path-table (used for path-planning), an integer which represented the amount of trash on that patch and two booleans which determined whether that patch was a spawn point or a trash can. Recall that "spawning" refers to the procedure for creating objects in the simulation. Therefore, a spawn point is a place where agents can enter the simulation. These agents represented the pedestrians of De Uithof, see section 2.2. Furthermore, there was an extra data structure which determined the amount of agents in each building. A day started at 07:00 in the morning and lasted until 19:00. Every 'tick' (the default time-step in NetLogo) represented one second. This meant that the entire simulation lasted 43200 ticks per day. For a decrease in computing time, a tick could be defined as any preferred period of time. Per tick, the time was updated and newly spawned agents were created at locations specified by a number of functions. In addition, all agents moved one step, either towards their goal or towards a trash can.

2.2 Representing the pedestrians

The pedestrians were represented by agents. These agents needed to have a number of parameters: a goal, a boolean which determined whether they had trash or not, a (trash)timer which determined when the trash boolean would be set to true and an integer which denoted the distance they were willing to walk to a trash can once they had trash. To reduce the time complexity, the agents were represented using the default NetLogo settings: as tuples (the mathematical term for an ordered list of elements) which moved along a separate (independent of patches), continuous grid in the NetLogo world. The default step size was one patch (which would correspond to a speed of 3 m/s in the real world), but this could easily be changed into any speed pre-determined by the researcher.

The agents were spawned through buses and entry points on the edges of the grid, since this is the only way for pedestrians to enter De Uithof in the real world. A method was implemented which read two kinds of files. Each entry point was represented by a pair of the following format:

$$\langle\langle X, Y\rangle, \langle V_{t1}, V_{t2} \quad \dots \quad V_{t42}\rangle\rangle$$

All values were integers. The elements X and Y in the first argument denoted the coordinates and the elements $V_{t1}, V_{t2} \dots V_{t42}$ in the second array argument denoted how many agents would spawn during that time period at the given coordinates (each time period lasted 15 minutes, ergo 42 time periods per 12-hour day). The exact timing within the time period was randomized, to ensure a gradual spawning. The bus data was also read from a file. This file contained pairs of the following format:

$$\langle\langle Hour, Minute\rangle, \langle X, Y, Pass, Line\rangle\rangle$$

Such a pair made sure NetLogo spawned “Pass” passengers, at time “Hour:Minute” on the patch “X Y”. “Line” was not used, but was added for showing how the bus data mirrored the real world: if preferred, the real world bus schedules could easily be implemented and “Line” would denote the line number. Due to time constraints, only two buses per hour were spawned.

Once the agents were spawned, they needed a goal to move to. This goal could depend on a large number of factors: time of day, time of year, starting point, where exactly in the building the agents had to be (due to most buildings having more than one entrance), the weather and whether the agent wanted to dodge busy chokepoints were but some of the factors involved. Because modelling the real world this precisely was beyond the scope of this study, only one factor was considered: the current amount of agents.

Two functions were created: a random destination function and a random “exit” destination function. The first one picked a random goal for an agent. The second one picked a random spawn point which was not an entrance of a building, ensuring the agent would leave the simulated area. This was done to control the complexity of my simulation: because the buildings were programmed to increase the decrease rate of their agent counters near the end of the day, more and more agents were spawned from these buildings as the day progressed. A method was needed to make sure this would not 'drown' the simulation in agents, and therefore the “exit” function was called whenever a new agent spawned and simulation started to lag.

Once the goal was decided, the agent needed to get there using a path-planning algorithm which mimicked pedestrian behaviour. For path-planning a multitude of options should be considered. Some methods simply determine the shortest paths (e.g. A*), some determine the optimal path considering the paths of other agents and crowding effects (van Toll et al. [24]) and some focus on the biological plausibility of the agents' movements (Karamouzas et al. [14], Henein & White [11], Curtis & Manocha [5]). All of these methods have their (dis)advantages. For this research, the ones which focus on biological plausibility may seem

like the best choice. However, this biological plausibility mostly stems from the way agents interact with each other. Because that was not the focus of this thesis, and implementing these methods is rather time-consuming, a different approach was used.

Since the paths did not change in between simulations, and the NetLogo world would be used in an evolutionary algorithm, the paths were determined pre-simulation and then hardcoded into the NetLogo grid, thus reducing the amount of time spent in-simulation looking for paths. Once the paths were determined, the world was exported to an Excel file. After this, the resulting file could be read using NetLogo's 'import-world' command. First, all of the different endpoints of the paths the agents could take were determined. These were the points where agents could enter and exit the simulation. Next, Dijkstra's algorithm was used to determine the shortest paths between these points. The paths were then hard-coded in the NetLogo grid. When the agents wanted to move they searched the surrounding patches (in a "walkradius" currently set to 2, see 4.1) for the highest value of their path, which determined their next step. This is in essence a implementation of a method called uphill-climbing, but then along a predetermined path.

When an agent reached its goal, it was removed from the simulation. If an agent reached a goal which was an entrance/exit of a building, the agent counter of this building went up. This agent counter represented the amount of agents inside a building. The agent counters decreased with a certain half-time factor, which was determined per building per time period of fifteen minutes by another file. The agent counters used the following format:

$$\langle N, \langle V_{t1}, V_{t2} \quad \dots \quad V_{t42} \rangle \rangle$$

In these tuples, N denoted the name of the building and the integers in the second argument denoted half-time factors for their specified time periods. If a factor was 0.9, this meant that in the corresponding fifteen minutes, 10% of the population would leave the building for which this factor held. The exact moment was determined by a randomised half-life function to ensure a gradual decline. Furthermore, a half-life method is independent of the agent counter's current value. When the agent counter of a building dropped, an agent was spawned at one of its entrances/exits, chosen randomly.

2.3 Representing trash

The only way for trash to 'enter' the simulation was through the agents. They had a variable called "trash", which could be true or false. There were two ways for an agent to get trash. 1. Whenever an agent was spawned, it had a 10% chance of having trash. 2. Whenever an agent was spawned, it had a 10% chance of getting a trashtimer which would cause it to have trash within 100 ticks (the exact value was randomised per agent). The default setting for "trash" was false. It should be noted that in applied research, other methods should be devised to determine whether an agent has trash or not. This would depend on factors such as time of day, and from which spawn point the agent entered the simulation. These factors could be determined by questionnaires for the inhabitants of De Uithof, or more by observational approaches such as counting what percentage of the population carries trash on various moments of the day.

A trash can was represented by a boolean per patch called "trash can", which was true if a trash can was present. If a trash can was found within the previously mentioned "walkradius", the underlying patch was set to be the new subgoal of the agent. If the trash can was reached, the "trashcount" of that trash can (technically of the patch with the boolean "trashcan" set to true) was increased by one and the boolean "trash" of the agent was set to false. If no trash can was found in the value denoted in the "walkradius" (set to 2), the agent would dump the trash on the floor. This was represented by colouring the patch to a dark green colour and increasing its "trashcount" by one. The agent's "trash" would be set to false again, and it would return its

normal path-planning algorithm. Since the walkradius was used both for looking for the next step along the paths and for looking for trash cans, agents could never get lost by dumping their trash in a trash can.

It should be noted that this method of finding a trash can was not based on any previous research and as such is unlikely to be realistic. This method was chosen because it seemed to be a reasonable simplification of human behaviour. However, in real life the situation is probably more complex: not only could one expect humans to favour trash cans in front of them over trash cans behind them, humans with previous knowledge of the area might also decide to hold onto trash for a while until they run into a trash can further along their path.

2.4 Evolutionary Algorithm (EA)

For the gene representation the NetLogo code had to be edited to ensure Java could communicate with it, since the EA's were written in Java using JGAP (an extension which enables EA's to be written with relative ease; see JGAP [12]). This was done by adding a method called “insert-trashcans” to the NetLogo simulation. This method took a numeric array as input and added trash cans to the specified locations. For instance, calling “insert-trashcans [[30 50] [78 50]]” added two trash cans to the simulation: at patch 30 50 and at patch 78 50. The array could be of any desired length.

Now, this method only added trash cans to patches which were accessible to the agents. Since running simulations with configurations which were partially non-placed would be inefficient (though more complete), the locations where trash cans could be added were limited to the area denoted in figure 2.3. A default gene was represented as follows: ten integers, which denoted five coordinates in the area denoted in figure 2.3.

After the fitness values of all genes of a population were determined, a subgroup was formed on which two operators were used with which the next generation would be created. Genes with a higher fitness value had a higher chance to be part of this subgroup. The first operator was a recombination operator, which took two configurations and created a new one which combined coordinates from both configurations. The second one was a mutation operator which took a configuration and created a new one by mutating one of the coordinates. Following the principles described in the introduction, these operators ensured the average fitness of the population kept increasing per generation.

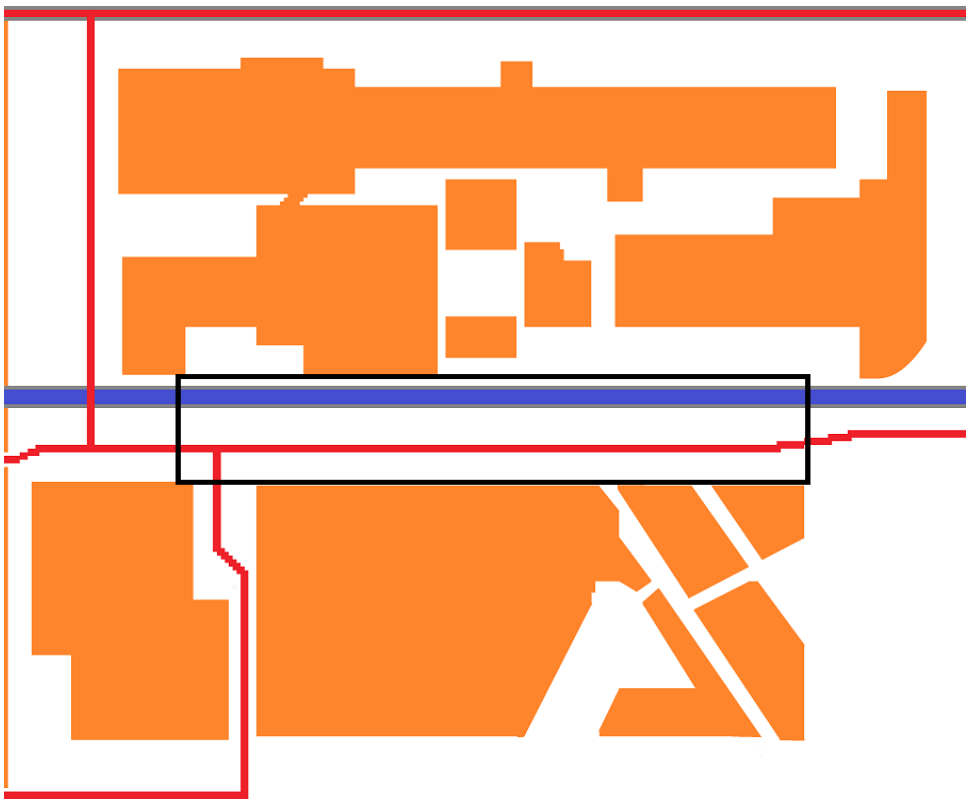

Figure 2.3. The area enclosed by the black square denotes the area in which trash cans could spawn.

Several EA's were created and all of them tried to find the most optimal way of placing the allowed amount of trash cans in the area denoted in figure 2.3. Unless stated otherwise, each EA added five trash cans, the population size was set to ten genes and all evolutionary algorithms were iterated for 25 generations. The subproblem they tried to solve can thus be phrased as follows: “Place the allotted amount of trash cans in the area denoted in figure 2.3 in such a way that the fitness of the resulting configuration is as high as possible.” The fitness of a configuration (and of an EA) can be tested in many ways. Table 3.1 uses some fitness measures to show the differences in results between the EA's used.

All the EA's started with a random population of genes. Recall that a gene was a trash can configuration. To ensure no EA had an advantage over the others, the EA's would only work if no start configuration collected over 3% of the trash. Normally, this measure would not be taken but an unfair start might result in unexpected results. By ensuring this start condition, some general properties of EA's (see [27]) could, hopefully, also be found in the EA's used in this thesis. For instance, the results should show that a decrease in either the population size or the amount of total generations should result in a less effective solution. If these effects could be found, they will provide evidence for a successful combination of crowd simulation and EA's.

3. Results

Every configuration was named after the defining characteristic of the EA which returned it. The one named “Top-25” was the best configuration of the population after 25 generations. Recall that the population size was set to ten and each configuration consisted of five trash cans. The other EA's are variations of this EA, and together they show the characteristics of the various EA's as outlined in section 2.4.

3.1 "Top-25"

Figure 3.1 shows Top-25 in the simulation. With this configuration, 96 pieces of trash ended up in the trash cans versus 276 on the floor. Using just five trash cans on a limited area of the configuration, Top-25 still managed to get 25.8% of all the trash in the trash cans. This is an impressive result, since the configurations of the first generation were set to collect less than 3%. The average fitness of the total population was 96 too, though not all configurations were the same.

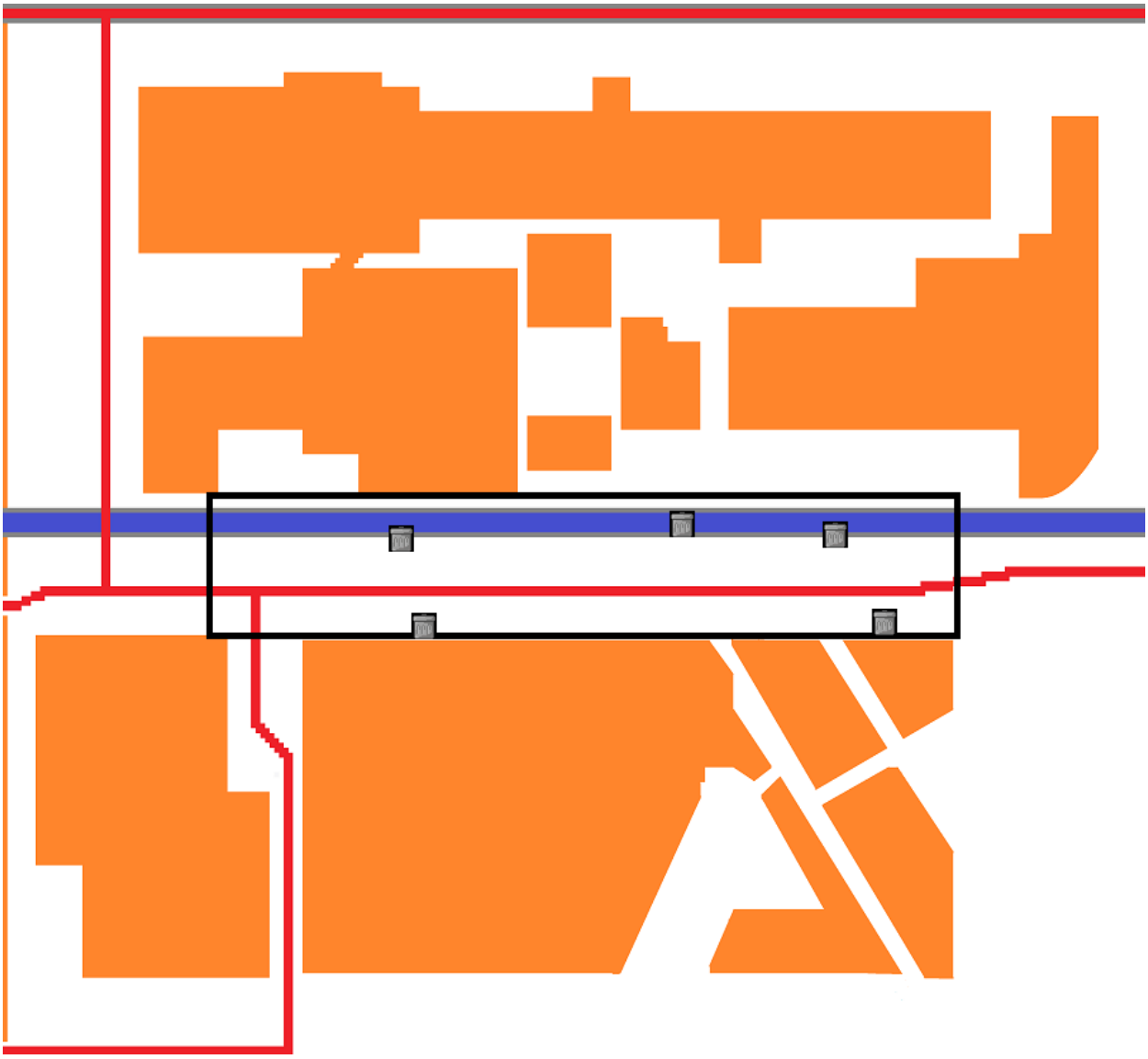

Figure 3.1 This figure shows Top-25, which was the best configuration of the population after 25 generations. The population size was set to ten and each gene (representing a trash can configuration) consisted of five trash cans.

3.2 "Top-10"

Since the amount of generations ran was determined by time constraints, the evolutionary algorithm was also executed another time with the same population size. This time, the algorithm was stopped after just ten generations, and the best configuration of the resulting population was "Top-10". As expected, the average fitness of the final population was a lot lower than after 25 generations (66.67 versus 96). All of the configurations had a fitness

value between 64 and 66, except for one, Top-10, which had a fitness value of 83. This is less than the top fitness value for 25 generations (96). However, Top-10 still managed to get 24.9% of all trash in the trash cans: due to the many random factors the total amount of trash dropped was lower during this run than during the Top-25 run.

3.3 "3-cans"

The EA was also ran using three trash cans instead of five, creating a configuration called "3-cans". After 25 generations, this resulted in a configuration which collected 44 pieces of trash. This corresponded to 15.7% of all the trash spawned. The average fitness of the final population was 43.6. Note that, in this case, every trash can on average collected 5.23% of all the trash. In "Top-25" this was 5.16%. These results seem to imply a linear relation between the amount of trash cans of a configuration and the trash collected on average by a trash can of that configuration, but this is not the case: since the trash cans could only be placed in an enclosed square, no amount of trash cans could possibly collect 100% of all the trash.

3.4 "Pop-5"

To show the effect the population size can have on the final output, another EA was created which had a population size of five (instead of ten): "Pop-5". After 25 generations, this resulted in a configuration which collected 53 pieces of trash. This corresponded to 19.1% of all trash spawned. The average fitness of the final population was only 51.4, clearly showing the effects of a decrease in population size as postulated in section 2.4

3.5 Manually created configuration

To test the results of these EAs, another configuration was determined using only the NetLogo simulation and logical reasoning. Since the problem space was nowhere near as complicated as it would be for a real world application, area's which were suitable for a trash can could be determined. After quite some trial and error this returned the configuration "Manual".

Manual had an absolute fitness value of 138 and collected 37,8% of all trash. This demonstrates the fact that an evolutionary algorithm decreases in usefulness when the problem it is applied to decreases in complexity. This does not imply an evolutionary algorithm could not, eventually, find a configuration with a higher fitness value. However, manually determining this configuration took about four times less time than calculating Top-25 whilst returning a superior result.

3.6 Overview

Table 3.1 shows the final results. It clearly shows the effects of a decrease in generations ran (Top-10), of a decrease in trash cans (3-Cans) and of a decrease in population size (Pop-5). The top three measures apply to the best individual (recall: an individual was a configuration of trash cans) of the final population that the corresponding EA returned. "Absolute Fitness" is the amount of trash collected in trash cans, "Relative Fitness" is the percentage of trash collected in trash cans and "Per Can" denotes what percentage of the total trash each can (on average) collected. The bottom two parameters denote the averages of the final population per EA.

Configuration	Top-25	Top-10	Manual	3-Cans	Pop-5
Absolute Fitness	96	83	138	44	53
Relative Fitness	25,8%	24,9%	37,8%	15,7%	19,1%
Per Can	5,16%	4,98%	7,56%	5,23%	3,82%
Average Absolute Fitness	96	66,67	-	43,6	51,4
Average Relative Fitness	25,8%	20,0%	-	15,5%	18,6%

Table 3.1, showing the final results. Since "Manual" was not created by an EA, no average values could be calculated.

4. Discussion

4.1 Agent parameters

In the crowd simulation literature, there does not seem to be a consensus regarding the actual physical presentation of a human being. Schadschneider [21] and Burstedde et al. [4] used a cellular automaton, but most studies used a representation on a continuous two dimensional plane. Two different methods of representing humans on a continuous plane were used: representing humans as ellipses (Curtis et al. [6]) and as circles (Van Toll et al. [24]), but no method seems clearly superior. Curtis et al. [6] used an ellipse with r1 = 0.24 meters and r2 = 0.15 meters and noted that the surface of this ellipse was 0.11 m^2. This corresponds to a circle with r = 0.19 meters. It is unclear what the effects of these different representations are, and further research should be conducted to determine this. For this thesis, all of these options were implemented but no significant changes were found. However, one might expect this to not be the case when the density of agents increases in a much smaller simulated space.

In the crowd simulation literature, there also does not seem to be a consensus regarding the movement speed of humans. A fixed movement speed for the entire population can be seen in Van Toll et al. [24], but this is but one of the methods used. Some studies include acceleration (Schadschneider [21]), a 'desired speed' versus a maximum speed (Helbing & Monár [10]) or a normally distributed speed attribute, which can differ per agent and agent class (Curtis et al. [6]). Apart from these methods, the actual values for the movement speed differ with margins as wide as 50% between studies. Since no consensus was found, and speed was not a relevant factor for this research, the NetLogo simulations used the method which required the least amount of computations (a fixed top speed). It should be noted that all other options can be implemented as well. Further research should be done in order to determine the correct movement speed of human beings.

As an example of a more problem-specific function, creating one which returns a biologically plausible goal for an agent is one of the many challenges in crowd simulation. It is a crucial component of any simulation, but it can be one of the hardest functions to formalise due to its dependency on many problem-specific factors (see Curtis et al. [6] for a well-documented example). In this study, the starting point was also dependent on the goal of an agent: if an agent wanted to go from building A to B, it would probably exit building A through the exit closest to building B. Before a problem can be solved using crowd simulation, preliminary testing should be done to ensure this function returns realistic results. This could be done by the methods mentioned in section 2.3: either by questionnaires or through more observational methods.

As an example of a more problem-specific variable, the walkradius was an integer which determined how far an agent would look for a trash can (or the next part of its path). In the simulations it was set to 2, which meant the agents checked all patches coloured black in the "r = 2" grid, which is demonstrated in figure 4.1. Both the value for r = 2 and the shape of the

search space were little more than educated guesses. This is because the area in which a human looks for a trash can is a very problem-specific variable and there is no relevant literature available regarding it. This is a prime example of the kind of human behaviour that can be critical to the biological plausibility of any applied study. If it cannot be modelled correctly, there is no point in running any simulations. Still, this should not be seen as a critical blow to the viability of crowd simulation, but as an argument for why computer scientists and psychologists should collaborate and try to create more realistic, computational models of various aspects of human behaviour.

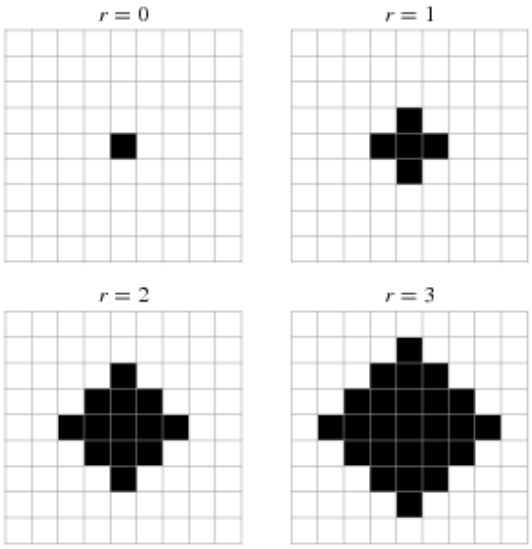

Figure 4.1 This figure shows how many patches were checked for different values of the variable "walkradius" (r). Von Neumann neighbourhoods: [25].

4.2 Simulation parameters

This study sought to provide a proof of concept for combining EA's and crowd simulations and as such provide an (approximation of an) optimal solution to an applied problem. Because this method required the simulation to be ran many times, the entire simulation was built to run as quickly as possible. This limited the biological plausibility of the NetLogo simulation. Especially the geographical accuracy proved to be an unsurpassable bottleneck for this research. This is a clear consequence of the "comparing versus optimising" debate: is the aim of a research to select the best option from some pre-determined configurations, or to provide an entirely new configuration?

A model is, by definition, less complex than the object or situation it seeks to model. Therefore, some properties of the real world will not be present in any simulation. However, if the research is comparative, much more detail can (and should) go into maintaining the biological plausibility of the simulation. For instance, for my thesis I chose to wait for a maximum of 24 hours on a solution from an algorithm. This eventually resulted in a three minute time period to return a fitness per simulation of a certain trash can configuration. However, if the aim of the research would not have been to optimise but to compare, a simulation could last for a time period of 24 hours divided by the amount of pre-determined trash can configurations.

With regards to biological plausibility, it is important to note that, while many researchers set their goal to accurately model crowds as realistically as possible, not all of them check their results in real-life. For instance, Curtis et al. [6] not only provide a framework for

modelling crowds as realistically as possible, they also tested this framework on a case study. This is in sharp contrast to a research such as the one conducted by Van Toll et al. [24], who conclude "*To enhance realism, we used crowd density information to guide a large number of characters along various routes."* Though their results seem visually convincing, they should be compared to actual human crowds before being called 'realistic'. In this thesis, the problem was merely used to illustrate a proof of concept and therefore the trash can configuration was not checked in real life.

4.3 Evolutionary algorithms (EA's)

A discussion point for almost any EA is the fitness function and how fitness is measured. The results section shows some fitness measures. Even though "Manual" is superior in every respect, table 3.1 shows how the definition of 'optimal' is not as clear-cut as one might like. For instance, "Per Can" 3-cans is superior to Top-25, even though Top-25's relative fitness is a lot higher. In applied research, fitness is determined by the (employer of the) researcher. In most cases, this will probably be a financial consideration.

New evolutionary algorithms are still being developed every day. From the classic ones (Introduction to evolutionary algorithms [27]) to state of the art algorithms which combine evolutionary techniques with different approaches (Ghanbari & Amiri [8] & Bosman & Thierens [3]), there's an EA for almost any optimisation problem. The aim of this thesis was to provide a proof of concept for a simple statement: "A subset of problems in the field of applied cognitive psychology can be solved using optimisation techniques derived from the computer sciences". This proof of concept is successfully given, but the evolutionary algorithms were unable to provide a better solution than a manually determined one. This is in part due to the relatively simple problem the EA's tried to solve, but might also be due to the EA's used or the speed of the computer on which they were run. Either way, it is irrelevant since this thesis merely used the trash can problem as an example to show several characteristics of EA's as described in section 2.4.

5. Conclusion

This study sought to provide a proof of concept: show that it is possible to combine crowd simulation and evolutionary algorithms to solve a subset of problems in the field of applied cognitive psychology. First, literature was assessed from both computer sciences and cognitive psychology. Next, a token problem was chosen: "Determine the optimal configuration of trash cans at De Uithof". The evolutionary algorithms showed all of their expected characteristics and returned promising results, leading to the conclusion that crowd simulations were successfully incorporated in multiple evolutionary algorithms. The methods used in this thesis can therefore be extended to solve many other problems in the field of applied cognitive psychology.

References

1. Ballinas-Hernández, A.L., Muñorz-Meléndez, A., Rangel-Huerta, A. (2011). Multiagent System Applied to the Modeling and Simulation of Pedestrian Traffic in Counterflow. *Journal of Artificial Societies and Social Simulation 14 (3) 2.*
2. Bing maps: http://www.bing.com/maps
3. Bosman, P.A.N & Thierens, D. (2011). Optimal Mixing Evolutionary Algorithms. *GECCO'11, July 12-16, 2011, Dublin, Ireland.*
4. Burstedde, C., Kirchner, A., Klauck, K., Schadschneider, A., Zittartz, J. (2001). Cellular Automaton Approach to Pedestrian Dynamics – Applications. http://arxiv.org/abs/cond-mat/0112119v1

5. Curtis, S. & Manocha, D. (2012). Pedestrian Simulation using Geometric Reasoning in Velocity Space. http://gamma.cs.unc.edu/PEDS
6. Curtis, S., Guy, S.J., Zafar, B., Manocha, D. (2011). A Case Study in Simulating the Behavior of Dense, Heterogenous Crowds. http://gamma.cs.unc.edu/PEDS
7. Epstein, J.M. (2008). Why model? *Journal of Artificial Societies and Social Simulation 11 (4) 12.*
8. Ghanbari, R. & Mahdavi-Amiri, N. (2011). Solving bus terminal location problems using evolutionary algorithms. *Applied Soft Computeing 11 (2011) 991-999.*
9. Ghaoui, C. (2006). Encyclopedia of Human Computer Interaction. *Information Science Publishing.*
10. Helbing, D. & Molnár, P. (1998). Social force model for pedestrian dynamics. http://arxiv.org/pdf/cond-mat/9805244.pdf
11. Henein, C.M. & White, T. (2005). Agent-Based Modelling of Forces in Crowds. *Multi-agent and multi-agent based simulation. Lecture Notes in Computer Science v. 3415; p. 173-184.*
12. JGAP: http://jgap.sourceforge.net/
13. Kaminka G.A. & Fridman, N. (2006). A Cognitive Model of Crowd Behavior Based on Social Comparison Theory. http://www.aaai.org/Papers/Workshops/2006/WS-06-02/WS06-02-004.pdf
14. Karamouzas, I., Geraerts, R., Overmars, M. (2009). Indicative Routes for Path Planning and Crowd Simulation. http://www.staff.science.uu.nl/~gerae101/pdf/fdg09.pdf
15. Montes, G. (2012). Using Artificial Societies to Understand the Impact of Teacher Student Match on Academic Performance: The Case of Same Race Effect. *Journal of Artificial Societies and Social Simulation 15 (4) 8.*
16. Mora, S. (2007). Cognitive Development: Neo-Piagetian perspectives. *Londen: Psychology Press.*
17. Neisser, U. (1967). Cognitive Psychology. *New York, NY: Meredith*.
18. NetLogo: http://ccl.northwestern.edu/netlogo/
19. Openstreetmap: http://www.openstreetmap.org
20. Poola, A. & Ball, L.J. (2006). Eye Tracking in HCI and Usability Research. *Encyclopedia of Human Computer Interaction.*
21. Schadschneider, A. (2001). Cellular Automaton Approach to Pedestrian Dynamics – Theory. http://arvix.org/abs/cond-mat/0112117v1
22. Tisue, S. & Wilensky, U. (2004). NetLogo: A Simple Environment for Modeling Complexity. *http://ccl.northwestern.edu/papers/netlogo-icss2004.pdf. Presented at the International Conference on Complex Systems, Boston, May 16-21, 2004.*
23. Tisue, S. & Wilensky, U. (2004). NetLogo: Design and Implementation of a Multi-Agent Modeling Environment. http://ccl.northwestern.edu/papers/netlogo-swarmfest2004.pdf. *Presented at SwarmFest, Ann Arbor, May 9-11, 2004.*
24. van Toll, W.G., Cook IV, A.F., Geraerts, R. (2012). Realistic Crowd Simulation with Density-Based Path Planning. *Presented at ICT.OPEN, Rotterdam, October 22-23, 2012.* http://igitur-archive.library.uu.nl/math/2012-1112-200439/DensityBasedCrowdSimulation_ASCI-2012.pdf
25. von Neumann Neighbourhood: http://mathworld.wolfram.com/vonNeumannNeighborhood.html
26. Wordnet: http://wordnet.princeton.edu
27. Yu, X. & Gen, M. (2010). Introduction to Evolutionary Algorithms. *Springer.*