university of groningen / faculty of science and engineering

Agent Allocation in Fighting Forest Fires

Bachelor's Project Thesis

Roel Rotteveel, s3143112, roel.rotteveel@gmail.com,
Supervisor: Dr M.A. Wiering

Abstract: Forest fires are an increasing problem with respect to human safety and most importantly the ever increasing climate change. In previous research an Evolutionary Neural Network method called ESP was used to stop forest fires in a simulation. For this research the ENN CoSyNE is instead used to establish the smallest amount of forest being burnt. This is done by creating eight waypoints for agents to navigate through, cutting firelines as they go. The impact of the amount of agents and the way of agent allocation is mostly looked into, and it turns out that a Multi Agent System with eight agents using General Intelligent Allocation turns out to produce significantly better outputs than a MAS using Nearest Goal Allocation or Nearest Goal Allocation And Direction-tuning.

1 Introduction

While forest fires were originally relatively harmless natural phenomena with ecological importance (Hutto, 2008), due to human intervention and human causes they are now a great potential threat for human structures and the carbon cycle. As a consequence of the increased dryness caused by the climate change (dryness is one of the three major causes for a forest fires), combined with more human intervention (cigarette buds etc.) there exists a greater chance of the ignition then ever before. In addition due to human intervention in the past century natural fuel beds have build up that cover the forest grounds with a highly flammable carpet. This has resulted in a higher tree mortality from fire (dos Santos and Nelson, 2013), a faster spreading of the fire and made them significantly harder to fight. Globally forest fires also have negative feedback on climate change since more carbon is released in the atmosphere and less oxygen can be produced, which leads to an even dryer climate as a consequence (Gillett, Weaver, Zwiers, and Flannigan, 1998).

A fire can be stopped by removing any of the three elements from the fire triangle: oxygen, temperature or fuel. Removing oxygen is a daunting task with forest fires and perhaps something for in the future, but as of now no realistic option for fighting large-scale forest fires.

Removing temperature is a more realistic task and is being performed on forest fires already by putting dropping water bombs on the fires from special water planes. The problem with this method is that temperatures can reach up to 800 degrees Celcius (Wotton, Gould, WL, Phillip Cheney, and Taylor, 2012) so cooling it down beyond the ignition threshold is almost impossible. This method is therefore mostly used to slow forest fires down. Also while the planes have enormous volume the method itself is only works locally, as proportionally only a small part of the forest fire can be reached via these water bombs.

Lastly removing fuel is the most efficient method for stopping the spreading of the forest fire. Reducing the amount of potential fuel is the main method focused on when fighting forest fires in real life. When fighting a forest fire there is always a crew boss who makes the full plan for fighting the fire. This plan mostly consists of locations to create firelines to stop the spreading. On the ground several ground troops can be distinguished. Starting off with the people who cut the fireline; the 'hot shots type 1'. Using tools like shovels, chainsaws and bulldozers they destroy the natural fuel in the fireline. Their support mainly comes from 'swampers', who move the cut materials away from the fireline and 'hot shots type 2' who assure the fire does not spread over the fireline, as for efficiency the fireline is kept at a minimum width.

In these experiments agents are used who should be seen as bulldozers performing the tasks of both the 'hot shots type 1' and the 'swampers'. To maintain simplicity it is assumed that fire cannot spread over firelines. The 'crew boss' part is played by our self-learning system, which will be discussed below.

These agents will thus be used to cut the fire lines around the fire, with the goal of enclosing the fire completely to reduce further spread. For the navigation of these agents subgoals will strategically be placed around the fire for the agents to traverse through.

How the subgoals will be implemented is discussed later, but the main method that will be used is Reinforcement Learning (RL) (Sutton and Barto, 2011). RL for fighting forest fires has been the subject of earlier papers (Wiering and Dorigo, 1998), but more specifically the method that will be used in this paper is an Evolutionary Neural Network (ENN).

This has been the topic of earlier research (Wiering, Mignogna, and Maassen, 2005), in this paper however the Evolutionary Neural Network method 'Enforced Sub Populations' (ESP)(Gomez and Miikkulainen, 1998, as cited in Wiering et al., 2005) is used. In a later paper multiple ENN's are compared, and there is one particular algorithm that proved to be most successful (in pole balancing). The algorithm meant is Cooperative Synapse NeuroEvolution (CoSyNE) (Gomez, Schmidhuber, and Miikkulainen, 2006). In short CoSyNE is an evolutionary neural network which evolves the networks at the level of weights. CoSyNE will be used to generate the subgoals needed for the traversal of the agents.

The main goal of this project was building a scalable Multi Agent System (MAS), so after the above implementation is discussed further research on scalable MAS and different scalable methods of assigning the agents to the subgoals are evaluated and compared.

2 System Description

This project was started with a group of five, all after the same objective: finding a way to reduce the effects of forest fires with a system using reinforcement learning. To be able to perform the research a simulation was needed, which is discussed below. The simulation was built by all five students, whereas part of CoSyNE was built by two students (including the author), and later expanded upon by the author. All other work belongs solely to the author. An example of an enclosed fire in the simulation can be witnessed in 2.1.

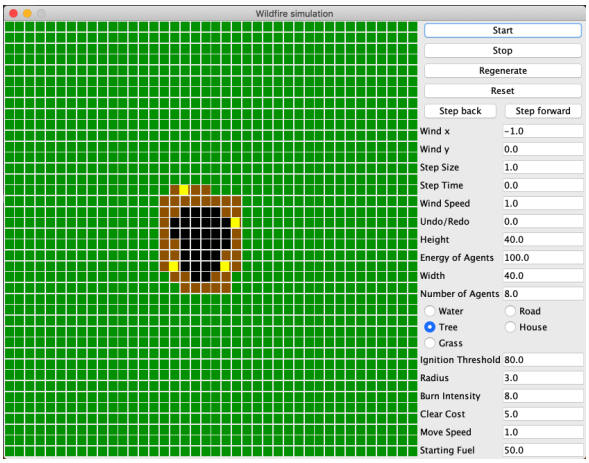

Figure 2.1: A fire enclosed by four agents at the end of the simulation

2.1 The Simulation

To simulate forest fires one first needs a simulation, so a simulation was built by the group who started this project. As advised by the supervisor a cellular automata (Wolfram, 1983) was built in Java, where the size of the field and the content of each tile in the field may be specified.

A tile may contain one element from the following list: Dirt, Grass, House, Road, Tree, Water. An agent can be placed upon each element apart from 'Water' and a burning tile. Within the list of elements the most important distinction is wether a tile is burnable or not. The burnable elements are 'Grass', 'Tree' and 'House', while the other elements are unburnable. An agent itself is not burnable in the sense described below but is able to be affected by the fire and die as a consequence.

Every element has multiple parameters, but since we are discussing forest fires we will mainly discuss the parameters important for the spreading of the fire. Every burnable tile is able to be ignited, burn for several time steps and be burnt afterwards. During the burning the tile will gradually change colour

from yellow to red to show the increase in ignition and will eventually turn black to show that it is burnt.

One of the most important parameters is 'Fuel'. Denser and larger areas (such as areas filled with trees) will burn longer than e.g. areas filled with grass (Nez-Regueira, Rodrguez-An, Proupn-Castieiras, and Nez-Fernndez, 2001). If a fire is contained in a tile it will not spread of course, so tiles can ignite neighbouring cells. They do this via their 'Burn Intensity'. A cell with a higher burn intensity will lead to a higher potentation of neighbouring cells, influenced by distance between the tiles and the wind. Cells must of course also be influenced by this intensity and this is performed in the parameter 'Fire Activity', which measures how much fire activity is radiated from neighbouring cells onto one specific cell. If this number crosses the 'Ignition Threshold' the cell will catch fire. Lighter, less dense materials such as grass have a lower ignition threshold than denser areas such as trees. The spreading of the burn intensity is greatly influenced by the wind, which is summarized in a vector with additional an additional velocity.
Regarding the movement of the agent two additional parameters are very important. The agent is able to move over every tile excluding 'Water', and this is set by the 'Move Speed'. This determines how many time steps it takes for the agent to traverse the tile. For each burnable element the agent is also able to cut it down, which also costs a number of time steps (as for moving) which is set in the parameter 'Clear Cost'. As said above when the agent 'clears' a tile, or cuts it down, it creates a path of dirt. This path of dirt may be made into a fireline if the navigation is chosen wisely.

In the simulation described above a heterogenous enviroment can thus be formed consisting of almost indefinite layouts with the agent being able to traverse through the environment. In the environment the fire is able to spread in all directions, influenced by different ground materials in the map and the wind.

While this was eventually not used in the experiments of this paper the simulation also has a built in generator which is able to randomly generate maps. Each generated map is completely unique and may consists of: meandering rivers, lakes, bridges, roads, houses, clusters of houses (villages), plains of grass, dirt and forests. The setup of the map can also be influenced by tuning the parameters 'Wetlands' and 'Urban' from zero to ten. Doing so will create different maps ranging from dry to wet and from rural to urban (with more roads and houses). An example can be seen in figure 2.2.

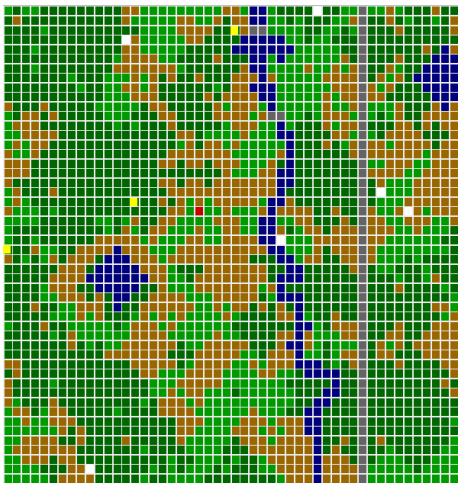

Figure 2.2: A randomly generated map, in this case with a river running through it

To allow for agents to cut a fireline guidance of the direction of the agents is needed. This is done using subgoal allocation with CoSyNE as described below in section 2.2 & 2.3. The most important part of this approach however is that the path of the agent(s) is not specified, but the coordinates of subgoals through which the agent(s) traverse(s). The agent thus does not learn with this method, but the generator of the subgoals does. The subgoal generator thus fulfills the role as a 'crew boss'. As discussed above a fire is unable to spread any further and thus contained when it is encircled by firelines, as can be seen in figure 2.1. The goal is therefore to make sure a forest fire is contained while minimizing the spread of the forest fire.

2.2 Subgoal Allocation

The subgoal generator can be thought of as a compass placed (with its own center) on the center of the fire. From that center eight cardinal axes spread out in the following directions: North, North-East,

East, South-East, South, South-West, West, North-West. On each of these axes a subgoal is placed through which the agents must traverse in order to enclose the fire. The problem is now thus extremely downgraded from 'encircling a forest fire' to finding a correct position on a one dimensional axis for a specific subgoal. If this is done eight times on each cardinal axis an outset for a circle is made which the agents can use to cut their firelines through. A schematic overview of this method is shown in figure 2.3. A neural network is used to determine the offset of the subgoal from the center of the fire but the network also starts with a standard offset, to guide the network in the correct direction.

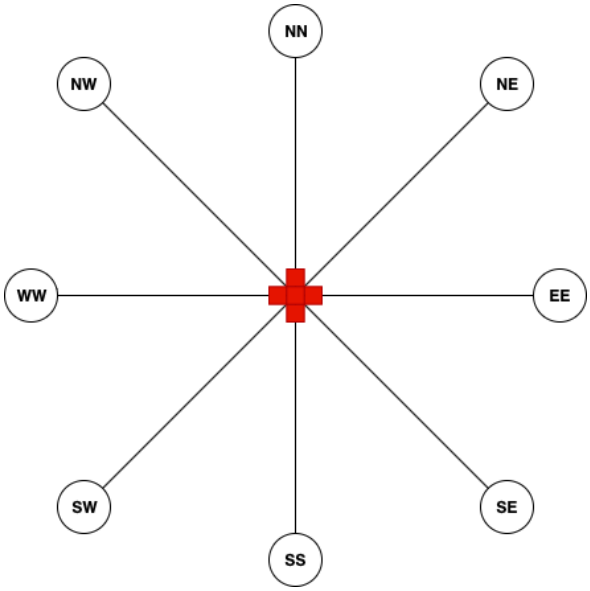

Figure 2.3: Schematic drawing of the eight subgoals on the cardinal axes

2.3 CoSyNE

What sets CoSyNE apart from other genetic algorithms is that it evolves networks at the level of weights, instead of crossing over e.g. entire networks. For each weight in the Neural Network CoSyNE assigns a bag with values to it. For each iteration while testing a value is pulled out of the bag for each individual weight. After evaluation of the networks (by use of the fitness) the top 25 percent are recombined using crossover and mutation. The new values are put back in the new weight bags. To assure the true potency of each weight is evaluated the bags are mixed up such that each weight potentially is part of a different network. A very schematic overview of CoSyNE is shown in figure 2.4.

The reason CoSyNE was used in these experiments is due to the fact that in the paper proposing CoSyNE it is compared to 14 other learning methods, including ESP, and seems to constantly outperform most methods. The methods are compared on four task configurations with increasing difficulty regarding pole-balancing with and without state information. In three out of the four task CoSyNE performed best (on the third it scored second) where ESP is only in the top-three once. Since the method of subgoal allocation using the cardinal axes transforms the daunting task of fighting a forest fire to a one dimensional problem this was chosen as the main method for setting subgoals (borrowed from Wiering, 2005), but for the above reasons CoSyNE was chosen instead of ESP.

In the implementation used for this paper the network is presented a varying number of inputs, dependant on the number of agents present in the simulation. The inputs are as following:

- Distance from the center of the fire to the predetermined offset subgoal
- Distance from center fire to fireline
- Wind relative to the axis
- Distance to agent(s)
  - A_0
  - ...
  - A_n

For the placement of each subgoal each of these four inputs are considered. The last input is always one number for a single agent but adds extra inputs automatically for additonal agents.

For the first input the center of the fire is calculated alongside the shortest path from that center to the coordinates of the predetermined offset. This way the network has information about how far away the subgoal is from the fire.

It also receives the distance from the center of the fire to the fireline (the edge of the fire) at that particular axis. Doing so ensures that the network

has information about how far the fire has spread. All simulations were performed with a fire consisting of several tiles already burning to account for the time needed to spot a fire.

As described above the wind influences the direction of the spreading of the fire. Since our subgoals are placed on a one-dimensional axis it matters enormously where the wind is blowing from relative to the axis. If the wind is blowing straight from East to West this logically influences the spreading of the fire in those directions while it has almost no effect on the spreading of the fire in the North and South direction. It is therefore extremely important to know how the wind blows relative to the axis.

Lastly, for each agent the shortest distance from that agent to current subgoal is calculated. The amount of agents may differ from one to eight.

With these inputs the network produces a single output, which is the distance between the center of the fire and the subgoal. This distance is projected upon the axis. Since the fires are always started in the middle of the map (to allow for maximum radius of the circle around the fire) the maximum distance the subgoal can be placed away from the center of the fire is therefore either half of the width or half of the height of the map. Since the output of the network is a normalised number we multiply this with half of the width/height and thus obtain the normalized offset of the subgoal, starting from the center.

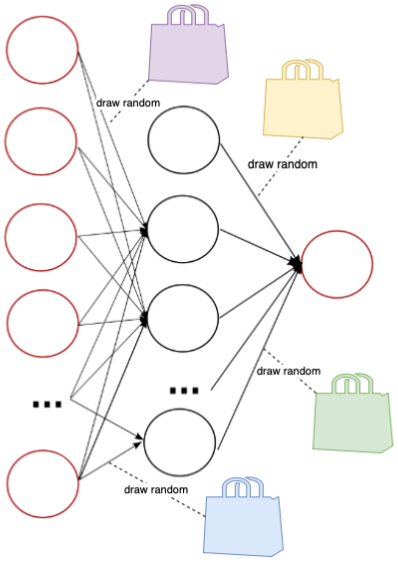

Figure 2.4: Schematic graph of a CoSyNE Neural Network with some weight-bags

2.4 Agent Allocation

This project was originally started as a single-agent problem, but since the applicability and the authenticity of the problem were important to the author of this paper the decision for a MAS was quickly made. If we look at real life examples there are always multiple agents at hand when successfully controlling a fire, and it is unthinkable that a single agent will be invented in the near future who is able to do encircle a forest fire by itself. Several implementations were made to study the efficiency of the division of agents over the subgoals.

In the single agent system the agent is first assigned to its closest subgoal, and traverses to that goal without cutting the tiles it travels over. It does not yet cut firelines to be able to reach its goal quicker. If it reaches the subgoal it consequently travels to the next subgoal moving in anti-clockwise direction. It does so until the fire is encircled or the agent dies. This method is called Nearest Goal Allocation (NGA). This method is extremely efficient for cases with just one agent or cases where multiple agents are already spread out evenly around the fire, e.g. if eight agents are all near a different subgoal. When multiple agents are close to the same subgoal however, there is no benefit of having multiple agents since agents cannot work together in this simulation. This means that the addition of an agent in cutting a fireline has no influence of the speed of cutting, meaning the extra agent is redundant. One can imagine the worst case scenario where eight agent all started near the same subgoal and traverse the same route a single agent could have done in the same time.

Two more implementations were made in assigning agents to subgoals. The second one has the addition that it assigns different directions to agents, namely clockwise and anti-clockwise. We call this one Nearest Goal Allocation And Direction-tuning (NGAAD). This has the advantage that if agents are placed near the same subgoal they at least work in different directions, but for more than two agents this is still not optimal.

For the third implementation the agents are first assigned to their closest subgoal, and then grouped into pairs if they are close enough (maximal two subgoals apart). If a pair is formed their directions are made opposite s.t. they both move away from the subgoal. In this step it is also made sure that there are never more than two agents on one subgoal, since agents still cannot work together. The agents are then spread evenly over the circle, s.t. an even distance is always maintained between agents, given the number of agents. Each agent then thus independently moves to their optimal starting subgoal before following a optimal direction to enclose the fire. We call this method General Intelligent Allocation (GIA). Using this method all agents cut the smallest amount of fireline, since moving around is significantly faster than cutting a fireline. An example with three agents is presented in figure 2.5, where the small black arrows represent the agents moving without cutting, and where the big white arrow represent the agents moving while cutting.

For navigation each agent uses the implementation of the famous Bresenham path (Bresenham, 1965). Other implementation like Dijkstra's shortest path (Dijkstra, 1959) and the A*-star algorithm were also implemented but not used in the end to maintain a low computation time.

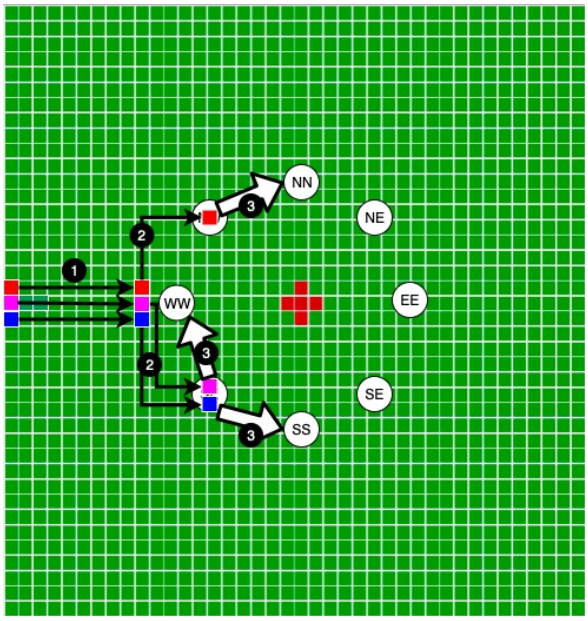

Figure 2.5: Schematic overview of the path of three agents with GIA

3 Results

All results in this experiment were gathered in an incremental fashion: perfecting a simpler variant of the problem before moving on to a more sophisticated version of the problem. The projects foundation is a single agent system using CoSyNE for subgoal allocation and assigning the agent to the nearest subgoal (following an anti-clockwise direction). These results will be presented first, before moving on to a Multi Agent System with the same agent allocation. After this the three methods of agent allocation described above will be presented with tests with three agents (since allocation is hard for three agents) before finally moving on to the results of a MAS with eight agents and the optimal agent allocation.

3.1 Experimental Setup

The task of the system is to enclose the fire with firelines in order to stop the fire from spreading. The simulation stops if all tiles were burned, all agents are burned or if the fire has not spread over a number of time-steps (meaning it is enclosed). The fitness of the system is determined by the amount of fuel that is burnt when the simulation stops. For example, a grass tile has a fuel value of 20, so for a 40x40 grid, the total amount of fuel that can be burnt is 32.000. This means that the lower a fitness is achieved the better, since less area is burnt. In reality the fitness is slightly off, overestimating the fitness by a little, but since this method is so efficient it was kept. Another important aspect of this fitness is that it increases in quadratic manner, as the fire mostly spreads in a circle. A quick encapsulation of the fire is thus enormously rewarding. All tests below were done with homogeneous grids filled with grass which most important parameters are given in table 3.1

To allow for testing with less agents the 'moveSpeed' was sometimes altered to make sure the agent(s) had sufficient speed to encircle the fire. Note that these velocities are in no way realistic but

Burn Intensity	8
Ignition Threshold	30
Ignition Threshold	30
Fuel	20
Clear Cost	1
Move Speed	X

Table 3.1: Parameters for the 'Grass' Tile

needed to ensure testing was possible. The value for this was mostly kept between one and three and will be mentioned at every test. In most experiments no significant difference in the learning curve was established after 100-200 generations, so most results will be with generations in that order.

CoSyNE was used with subpopulations (bag-size) of 30 weights, three hidden layers and mutation rate of 5%. All experiments were run on a 2.3 GHz Intel Core i5.

3.2 Single Agent System using CoSyNE and NGA

To test the performance of CoSyNE in the forest fire simulation a test was first performed with a single agent and NGA.

First tests were performed on a 20x20 grid, but this soon turned out to be too small. The grid was then enlarged to a homogeneous grid of 40x40, filled with grass. Since the ignition threshold of grass is relatively low it spreads relatively fast. In order to be able to encircle the fire the single agent needed quite some speed to traverse around the fire, so the move speed was set to 4. This leads to a situation impossible in real life where the agent is able to move insanely fast, but for this first experiment the main interest was the performance of CoSyNE. The results are summarized in figure 3.1

3.3 Multi Agent System using CoSyNE and NGA

In order to test the efficiency of the increase of agents tests were performed to discover the difference in efficiency between using multiple agents. In the test performed each individual agent was placed next to a different subgoal and they all moved anti-clockwise, e.g. no agents were 'grouped together'. Notice that due to this testing set-up tests with

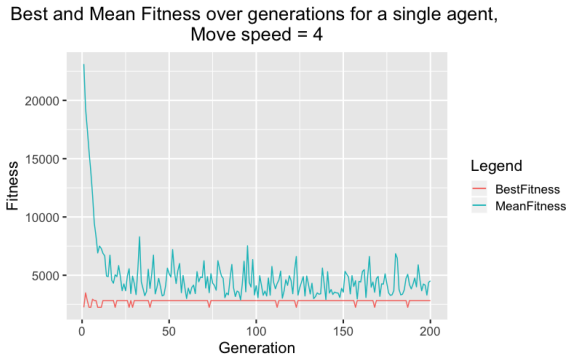

Figure 3.1: Graph of the fitness of a single agent with Movespeed = 4

multiple agents have a considerate advantage over test with lesser agents. The tests were performed with the value one for 'Movespeed' to ensure the results would be as realistic as possible (for higher velocities single agents can still encircle the fire relatively easily). Since in previous tests no significant difference in performance was witnessed after 100 generations the tests for multiple MAS were performed for 100 generations. The results are presented in figure 3.2

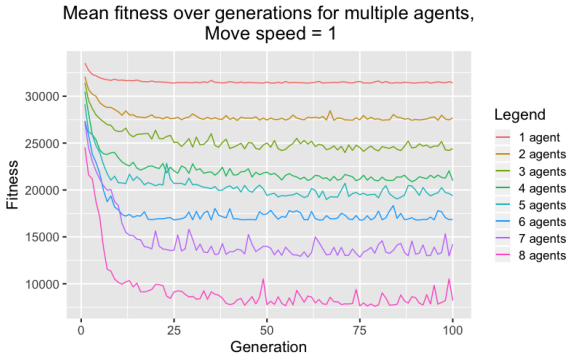

Figure 3.2: Graph of the fitness of multiple MAS with Movespeed = 1

3.4 Agent allocation: NGA vs. NGAAD vs. GIAA

One of the most important additions of this experiment was the manner of agent allocation. In order to maintain scalability all methods are immediately applicable to any number of agents (the maximum

in this experiment was kept at eight however). As described above we have the following methods:

- Nearest Goal Allocation (NGA)
- Nearest Goal Allocation And Direction (NGAAD)
- General Intelligent Allocation (GIA)

The last method can be seen as the final result of the agent allocation, where the first method was the first step and the second method is more of a control method to see what the actual impact is of the third method.

All three methods will be tested with three agents since this is quite hard to allocate optimally while it is still easy to illustrate the importance of agent allocation. To challenge the methods the agents start closely together, as illustrated in figure 3.3

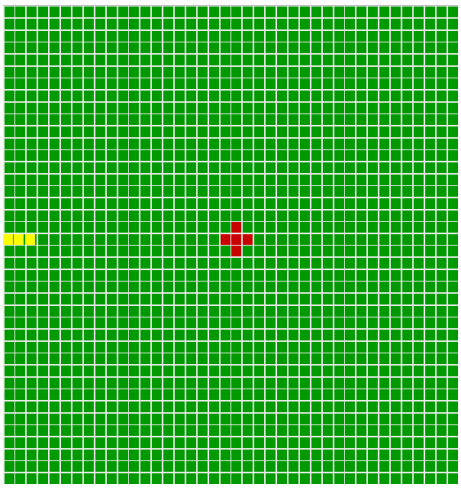

Figure 3.3: Screenshot of the starting position of the three agents

The different allocation methods were tested in the order described above and their results are presented below. For each method the 'Mean Fitness' was recorded and these values are presented in figure 3.4

3.5 CoSyNE, GIA and 8 agents

Based on the results testing was then performed with eight agents and GIA. This MAS was tested on the following environments:

Figure 3.4: Graph of the mean fitness of different agent allocation methods with Movespeed = 2

- The homogeneous grass environment also tested in above
- A homogenous environment with more wind

The first experiment can be seen as a base-case experiment. Based on previous experiment we already know positive results can be obtained in this environment so this is not the most interesting case. In all experiments the eight agents are placed around the fire in a circle. Since the effects are more abundant with a move speed of two the move speed was kept at at two.

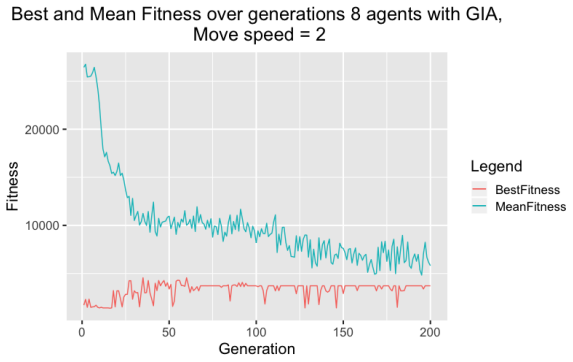

Figure 3.5: Graph of the best and mean fitness of a MAS with 8 agents, GIA and Movespeed = 2

The other case however is more interesting. It represents situations where the wind might have more effect on the spreading of the fire in a certain direction. In real life there is almost always some wind present and especially for larger fires this has

a big impact on the direction of the fire. The input regarding the wind relative to the subgoal was one of the first features to be implemented so was not looked back upon extensively throughout the extra implementations. It was thought that the wind was kept static at 0.1 for all experiments, but when starting the experiments with more wind however, it came to attention that the wind had been kept static at 0.0 , probably due to a different pull from Github early in the process of building this system. Thus the increased wind is compared to the baseline of a wind of 0.0, which should have actually been a wind speed of 0.1. The results of these experiments are presented in figure 3.6

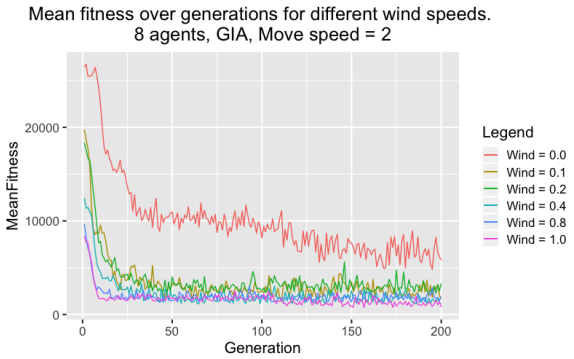

Figure 3.6: Graph of the mean fitness of a MAS with 8 agents, GIA and Movespeed = 2 for differen wind speeds

3.6 Discussion of Results

Since the discussion of some of the above results is too elaborate for the conclusion we will discuss the differences between the different Agent Allocations here, before moving on to the conclusion.

3.6.1 Agent Allocation

When we look at figure 3.4 the results of Nearest Goal Allocation for 3 agents stabilises at 27.000, which is considerably worse than the result with the single agent. This can be explained by the lower move speed and the starting position of the three agents as presented in figure 3.3. Since in this particular experiment NGA was used all 3 agents are allocated to the 'West' subgoal and all move anti-clockwise around the same subgoals. For these 3 agents with this move speed it would thus make no difference if 1 agent would be placed at that location or if 3 were.

When examining the above method the most inefficient part of it is the fact that all agents move in the same direction, namely anti-clockwise. In order to examine the effect this has on the final result NGAAD was also tested. With NGAAD half of the agent moves in clockwise direction, while the other half moves in anti-clockwise direction. The direction is assigned randomly. When we look at the results again a downward trend is witnessed, now stabilising lower than NGA. NGAAD stabilises around 21.000, a decrease in 'Mean Fitness' of 6.000 as compared to NGA. This decrease in tiles burned of course comes from the fact that 2 of the 3 agents travel in different directions through the subgoals, thus meeting on the other side of the circle. This simple implementation allows for almost one fifth of the total map (6.000 fuel out of 32.000) extra to be saved due to the shorter distance both agents have to travel. Notice that although there is a significant reduction in the amount of tiles burnt there is still one agent who is absolutely abundant, as it travels along the same path as one other agent.

The GIA architecture avoids the problem described above of assigning too many agents to the same subgoal. It does so by not just blindly assigning agents to their nearest subgoal or blindly setting different directions for agents but actually assigning each agent a optimal subgoal and a optimal direction. This can also be witnessed from the results. Here the downward learning curve is again witnessed, but this time stabilizing at much lower 'Mean Fitness': around 11.000. Compared to the 'Mean Fitness' described above this a significant reduction; reaching almost half of the tiles burnt as compared to NGAAD. This can easily be explained by looking at the subgoals and directions each of the three agents are allocated to. Starting from their initial positions as presented in figure 3.3 2 of the 3 agents will travel to the nearest goal, namely the subgoal 'West'. As with NGAAD they will both be assigned different directions so they travel in opposite ways. GIA now recognises however that this subgoal is full and moves the (groups of) agents apart, as can all be seen in figure 2.5. Only after being assigned to their optimal

subgoal do they start cutting in the right direction. Through this assignment the system assures that each agent only has to travel one-third of the distance of the total circle drawn around the fire through the subgoals (remember that moving without cutting is way faster, so the third agent reaches its subgoal relatively fast).

The enormous reduction in fitness as compared to NGAAD can be explained by the distance each agent has to travel over the circle. With NGAAD 2 agents both traversed through half of the circle, while the third agent was abundant. With GIA each of the 3 agents traverses through one third of the circle on average. Notably the agents do not travel twice as less distance, but the total fitness is still almost halved. This is due to the fact that the fire spreads in a circle, and thus spreads in quadratic manner. Any reduction of the radius of the forest fire leads thus to a significant decrease in the number of tiles burnt, especially for larger radii.

3.6.2 MAS with GIA: higher 'Best Fitness'

When we look at the 'Best Fitness' we see a result that is puzzling at first, as it slightly increases over the generations. This is due to two factors. First, because of the simplification of the problem and the many iterations per generations, there is great chance of finding a (near) optimal solution in the first generation. That explains why the 'Best Fitness' is so low in the beginning. Secondly, to explain the slight rise it must be realised that the system learns that the risk of pursuing the best fitness is too high to aim for the best fitness all the time. One single miscalculation or different taken path may result in the fire escaping from the circle (which results in the whole map burning down), while if the system aims at a somewhat larger radius, each iteration a little more terrain burns down (the extra addition of the radius on top of the best radius) but there is a significantly smaller chance of the fire escaping, thus resulting in a significantly lower 'Mean Fitness', as can be witnessed in the graph.

3.6.3 Limitations

Before concluding on these results some matters need to be justified. While all effort was put into making this simulation as realistic as possible there are some serious limitations to consider.

Firstly a big limitation is that agents currently cannot work together. This is briefly discussed above but although it simplifies the problem at the moment it is believed to be a crucial matter for a more realistic simulation. Secondly in order to approach some manner of reality it is also believed to be crucial that in further implementations a definite velocity can be set for agents and fire alike. In professional jargon the velocity of forest fires is given by the Forward Rate Of Spread (FROS). If this together with absolute velocities (in km/h) for agents could be altered in the simulation some notion of how realistic the 'solution' to this problem is could be reached.

4 Conclusions

All results will be discussed in the incremental order they were performed and discussed in the Results section.

4.1 Single Agent System using CoSyNE

During building the system the focus was first on building a Single Agent System that could succesfully encircle a forest fire, using CoSyNE as the method for the subgoal allocation. These results are presented in 3.1 and show a very steep learning curve in the first few generations. After around 25 generations the fitness seems to stabilise just below 5000, which is one fifth of the original fitness at the first generation. This shows that using the subgoal allocation method with CoSyNE is quite effective for Single Agent Systems. What is interesting to see is that the 'Best Fitness' is found around the very first generations and stays around this value. The move speed could probably have been a bit lower here to make the encircling harder, since there seems so be little risk involved when pursuing the 'Best Fitness' (otherwise 'Best Fitness' would have increased somewhat).

An important factor for the extremely steep learning curve is due to the simplicity of the subgoal generator. Since this transforms the problem into a one dimensional problem a optimal solution is almost immediately found in the first few gener-

ations and the mean fitness approaches this fitness over the generations.

4.2 Multi Agent Systems

After concluding that CoSyNE is an efficient method for allocating the subgoals, the aim of this project was moved to building a MAS. As discussed in section 3.6 agents cannot work together in cutting down firelines so the absolute maximum of agents that could potentially be working together is 16, meaning that for every subgoal 2 agents can be allocated that dig in different directions. If every subgoal contains 2 agents then agents of neighbouring subgoals can 'meet in the middle' inbetween subgoals. Since this is very cumbersome in the experiments a maximum of 8 agents was used, thus with a maximum of 1 agent per subgoal. The results of using a number of agents varying between 1 and 8 are presented in figure 3.2. As would be expected a considerate drop in area burnt is witnessed with an increase in the number of agents. A significant drop in 'Mean Fitness' can mostly be witnessed in the difference between 7 and 8 agents, meaning it is extremely efficient to use (at least one) agent per subgoal.

4.3 Agent Allocation

The experiments have shown that CoSyNE is indeed an efficient method and that this MAS works most efficient for 8 agents. The latter part of this research however was aimed mostly at allocating the multiple agents to the subgoals generated by CoSyNE.

All three systems show a similar trend where the 'Mean Fitness' slowly approaches a minimum. If we compare the three systems, as can be seen in figure 3.4, It is immediately clear that GIA outperforms the other 2 systems, and by a significant amount. For the reasons described in section 3.6 this makes perfect sense, and it can thus be concluded that GIA is able to save more forest than with regular agent allocation. Although this has not yet been tested GIA is built such that it could easily be scaled up to more agents, as it is built such that it solely takes as input the nearest subgoal for each agent. From this it determines the optimal subgoal for each agent by looking at the distances between agents with regard to the subgoals. For more agents also more subgoals would be needed but this also poses no problem for GIA, that is merely a matter of extending an array.

4.4 MAS with GIA

Finally it is time to evaluate a combination of the above: a MAS with 8 agents using GIA and CoSyNE for subgoal allocation. The results of this experiment are presented in figure 3.5 and contain no surprising results. As before the downward trend of the 'Mean Fitness' where it approaches the 'Best Fitness' can be witnessed. In the end it seems to stabilise at 6.000, which is the lowest fitness as of yet (here the efficiency of GIA is again accentuated once more as with more than twice as much agents (as compared to the 3 agents in the above experiments) the system is still not able to score less than half of the fitness with 3 agents, which was around 11.000). From the 'Best Fitness' it is seen that the system learns to be more careful, what results in a slightly higher 'Best Fitness' throughout the generations. This experiment is more of a summary of the experiments described above and shows that not only all parts of the system work separately, but that they also work together.

4.5 Impact of wind

These results may have been the most surprising results as of yet. Due to a human error the windspeed was accidentally kept at 0 , where it should have been kept at 0.1 . Figure 3.6 shows the same downward trend seen in all experiments before for all wind conditions, but most importantly it shows a significant reduction in mean fitness when the wind is not kept at 0. Although the wind was purposefully kept static throughout all experiments until now the original intention was to keep it static at 0.1, which seems to stabilize around a fitness of 3.500 , as opposed to the fitness of 6.000 where the wind is 0. This can of course be explained due to the fact that one of the inputs of the input vector of the network was the wind relative to the subgoal, multiplied by the wind speed. When the windspeed is 0 this number naturally also becomes 0, so the network always received a 0 as input. When the windspeed is nonzero however, this input starts to have an influence on the offset of the subgoal and

this can be witnessed in the almost halving of the 'Mean Fitness' due to the nonzero windspeed.

Another interesting, although more expected, result is the fact that with increased windspeed the total 'Mean Fitness' seems to decrease. In the graph the lowest mean 'Mean Fitness' is attributed to a windspeed of 1.0 , where the mean 'Mean Fitness' slightly increased with lower wind velocities. The cause for this phenomenon is the big impact the wind has in the simulation. Wind from e.g. the east with a high windspeed spreads mostly towards the west, and less in a circle as with lower wind speeds. Although the wind spreads faster in westward direction it spreads significantly slower in north- and Southside direction, whereas it almost does not spread towards the east. Since with a circular fire the burnt area spreads quadratically due to the expansion of the radius, with wind velocities this high the wind mostly spreads to the west and the total burnt area does thus not spread quadratically anymore. Thus a lower mean 'Mean Fitness' more achievable to obtain as long as the downwind firelines are cut.

4.6 Final Conclusion

This project was started to be able to enclose forest fires with firelines and with this particular implementation a MAS using CoSyNE for subgoal allocation was pursued. First a SAS was achieved where the CoSyNE implementation was developed and tested. These test have all proved successful. It was then proved that in a perfect symmetrical setup using multiple agents was more efficient than using less, where a maximum of eight. agents was established. The next challenge was agent allocation for which GIA was developed, which has been compared to NGA and NGAAD and has proven to be significantly more efficient, which has been shown with an asymmetric setup. The above positive results were combined in one MAS using CoSyNE and GIA which proved to be even more efficient in combination with each other. In the end it was discovered that a paramater for the wind velocity was accidentally kept at zero, where it should have been kept nonzero. This lead to the pleasant discovery that changing this ensured a reduction in 'Mean Fitness' by a factor two (where the aim was to reach a lowest possible fitness). It has thus been proven that the overall system is quite successful in reducing the burnt area after the start of a forest fire, and most importantly that General Intelligent Allocation is able to save more forest than regular agent allocation.

4.7 Future work

The maximum amount of agents evaluated in this experiment was eight, a considerably smaller amount than the 1000's of firefighters that are actually getting deployed for large scale fires. In the future it might be looked into how a similar project could be realistically scaled up. Tools like a 'Markov Chain Model for Evaluating Seasonal Forest Fire Fighter Requirements' could perhaps be used to evaluate how many firefighters are really needed (Boychuk and Martell, 1988).

As was seen in the experiments wind played a huge role in the final outcome, so now that this foundation is finished more research could be done on the effects of wind, and this could also be considered more when deploying a technique to fight the fire.

In the end it might be even better to focus our attention on the prevention and detection of forest fire, as opposed to fighting them. It has for example been looked into how wireless sensors might help for spotting fires(Lloret, Garcia, Bri, and Sendra, 2009), but little attention has been paid to prevention of forest fires. A reason for this could be the costs of huge projects like this, but let us not forget that every forest fires is not only a direct threat to mankind, but by itself also contributes to the disturbance of the carbon cycle.

References

Dennis Boychuk and David L. Martell. A Markov Chain Model for Evaluating Seasonal Forest Fire Fighter Requirements. *Forest Science*, 34(3): 647–661, 09 1988.

Jack E Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1): 25–30, 1965.

Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

A.R. dos Santos and B.W. Nelson. Leaf decomposition and fine fuels in floodplain forests of the rio negro in the brazilian amazon. *Journal of Tropical Ecology*, 29(5):455458, 2013.

N. P. Gillett, A. J. Weaver, F. W. Zwiers, and M. D. Flannigan. Detecting the effect of climate change on canadian forest fires. 1998.

Faustino Gomez and Risto Miikkulainen. 2-d pole balancing with recurrent evolutionary networks. In *International Conference on Artificial Neural Networks*, pages 425–430. Springer, 1998.

Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. Efficient non-linear control through neuroevolution. In *European Conference on Machine Learning*, pages 654–662. Springer, 2006.

Richard L Hutto. The ecological importance of severe wildfires: some like it hot. *Ecological Applications*, 18(8):1827–1834, 2008.

J. Lloret, M. Garcia, D. Bri, and S. Sendra. A wireless sensor network deployment for rural and forest fire detection and verification. *Sensors*, 9: 8722–8747, 2009.

L Nez-Regueira, J.A Rodrguez-An, J Proupn-Castieiras, and O Nez-Fernndez. Calculation of forest biomass indices as a tool to fight forest fires. *Thermochimica Acta*, 378(1):9 – 25, 2001.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

MA Wiering and Marco Dorigo. Learning to control forest fires. Metropolis Verlag, 1998.

MA Wiering, Fillipo Mignogna, and Bernard Maassen. Evolving neural networks for forest fire control. In *Proceedings of the 14th Belgian-Dutch Conference on Machine Learning*, pages 113–120, 2005.

Stephen Wolfram. Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, 55:601–644, Jul 1983.

Mike Wotton, James Gould, McCaw WL, N Phillip Cheney, and S.W. Taylor. Flame temperature and residence time of fires in dry eucalypt forest. *International Journal of Wildland Fire*, 22, 01 2012.