Multi Agent Systems - Final assignment
Simulating embankment building with an autonomous Multi Agent System

David Zomerdijk, Jorg Sanders & Maurits Bleeker

March 27, 2016

1 Introduction

Even though efforts are made to mitigate green house gasses the increase of the sea level seems to be inevitable. For the Dutch this is great news since our reputation on building embankments is unparalleled. However, to stay in the race we need stay on top of our game. Although the TU Delft builds beautiful embankments they take a lot of time and need heavy machinery to build. Therefore, this approach is not suitable when the need for protection is acute. We propose that instead of using machinery that takes weeks to transport, we use a swarm of relatively small robots that work together to build an embankment. Due to their limited size they can be flown in right before a flood is about to happen, potentially saving lives and billions of euros on damages.

Before we commercialize this idea, we intend to build a simple prototype in NetLogo [1]. In this prototype we will visualize the task at hand and implement multiple strategies for completing the tast at hand. In this report we will describe the implementation of our prototype and compare the implemented strategies both qualitatively and quantitatively.

2 Implemented Model

For this project we implemented a few different simulation models (environments with different agents). Each model represents a Multi Agent System in which multiple agents are working together to build an embankment between the sea and the beach as fast as possible. The different models differ in the way how the agents respond to specific states in the world. By means of these models we investigate if the agents will solve the problem different and/or faster by using other communication approaches or ways to deal with the environment.

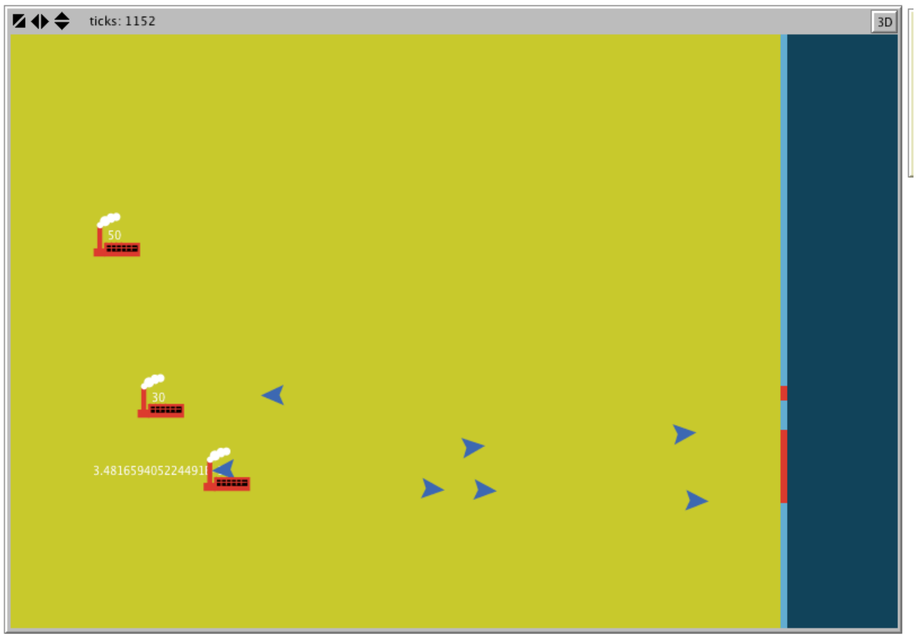
As a starting point we build a 'basis model'. Subsequently, we develop extensions on top of the basis model with some different implementation of the agents. Both, the basis model and the extensions will be discussed.

First we will give a detailed description about the environment. Second, an overview of the agent implementation and the BDI-models we used for every model is described. Third, we will describe the communication system the agents use for result sharing and coordination between the tasks.

2.1 Environment

In figure 1 an overview is presented of the environment we build for this project. This environment is completely static (nothing is changing until an agents does something in the world) and deterministic.

Figure 1: The environment created for this project.



There are two main components in this environment the shoreline and the depots.

Shoreline The dark blue area on the right of the figure 1 is the sea, the light blue line is the separation between the beach and the sea: the shoreline. Every patch on the blue shoreline is a building spot where an agent could possibly build a part of the embankment (embankment part). In case all building spots have been filled with a patch, the construction process has been successfully completed. When starting the simulation every blue patch on the shoreline is an empty building spot. This means that part is still open and that one of the agents needs to build an embankment part there to complete the entire embankment. The red patches on the shoreline are building spots where one of the agent build an embankment part. After an agent build an embankment part it stays there until the simulation is over. This is simplification relative to the real world. In the real world embankment parts could get damaged and should be maintained and repaired. It is assumed that every embankment part that has been built is high and strong enough to hold the sea.

We modeled the shoreline as a straight line for this project. This because a bumpy coastline is much harder to explore for the agents. The cost to build a complete embankment part is 10 resources.

Depots The red buildings that are visualized in figure 1 are the depots. Each depot has a different location and the amount of depots could vary between the simulations. These depots contain resources the agents need for building embankment parts. When an agents arrives at a depot it can pick up a maximum amount of resources, for all of our simulations we set the max amount of resources an agent can pickup at 10. The depots don't have an unlimited amount of resources. At every time point in the simulation a depot can contain at most 50 resources. Every time a agents picks up resources the amount of resources of that concerned depot will decrease with 10 until there are no more resources left. Agents could supplement depots if there are not enough resources left to build embankment parts. This they could to by 'making' new resources at a depot. Every tick (time step) in the simulation they could make 0.2 new resources. Agents can repeat this process until there are 50 resources at a depot again. How this new resources will be made has not been taken into account.

The environment is the same for the ' model' as in the extended models.

2.2 Agent implementation

For this project we made an agent-model. Every agent in the Multi Agent System is using the same agent model. Every agent is self-interested. They act to further their own interests, possibly at expense of others, but they try to communicate as much as possible with each other to avoid conflicts and double labor.

We implemented the agents according to the BDI-model [2], in addition to that we added some more metal states specific for this domain application.

2.2.1 Desires

In this simulation agents could have three different desires. See table 1 for a complete overview.

When the simulation starts the world model of the agents is not complete. At $ticks = 0$ all the agents only know the amount of depots in the environment and the length of the shoreline. What they do not know is the location of the shoreline patches (building spots) and the location of the depots. Therefore they first need to discover the environment. We implemented two different approaches for discovering the environment; one where all agents cooperate to explore the entire world model before they commence building, this one is used in the basic model. In an extended version we used a 'smart approach' where some builders already commence building while other agents finish the exploration. Hence, in the latter approach not every agent has the same desire. The latter model will be discussed in subsection "exploration extension".

Standard model In this approach the agent moves in a straight line through the world until it hits an obstacle (e.g. `patch-ahead 1 is a obstacle`). The heading direction of the agent will be initialized random. In the case that patch-ahead is an obstacle (e.g. depot, the shoreline, another agent,the boundaries) it changes direction to a random direction facing away from the object. While the agent moves through the environment it observes the environment.

Table 1: The possible desires an agent could have during the simulation for all the details.

Desire	Implementation	Description/Explanation
Find depots and shoreline	Implemented as an array with on the first index the desire. This is the default desire	When the simulation starts, the world model of the agents if not complete. They know how many depots there are in the environment but not where they are. This holds for the shoreline as well, they know how long the shoreline is, but not exactly where it is.
Build embankment	Implemented as an array with on the first index the desire	If the whole shoreline has been discovered and the agent have found all the depots. They can get the desire to build the embankment.
Have a beer	Implemented as an array with on the first index the desire	The whole embankment has been build. The agents are finished and can have a beer. If all the agents have the desire to get a beer, the simulation is over.

At every tick the agent observes the patches in a radian of α around itself. If one of these patches in the set of observed patches is a depot or a shoreline patch, then the agent will add this patch to his beliefs about the world. In the case the agents have discovered the entire shoreline and all depots, their desires change to "building embankment". Consecutively, when the entire embankment is build (patches_shoreline = patches_build), the agent's get the desire to drink beer with their buddies and they will find the closest depot to fulfill this desire. If all agents are located at a depot with this desire the simulation ends.

Table 2: The mental state an agent has during the simulation

Mental states	**Implementation**	**Description/Explanation**
Mental State Refill Action	Boolean (refilled_depot)	Mental state that is true when an agent just refilled a depot
Mental State Reconsideration Intentions	Boolean (do_reconsider)	Mental state that is true when an agents needs to reconsider its intentions, this could be if there are no depots left with resources.
Mental State Closest Depot	Patch (closest-depot)	This mental state contains the closest depot which contains resources relative to the current position of the agent. This Mental state will be deliberated from the set of beliefs_depots.
Mental State Observations	List with patches (observations)	This mental state contains all the patches the agent observed during $tick = t$.
Mental State Empty Depots	Boolean (found_empty_depot)	Mental state that is true if an agent observed a empty depot.
Mental State Nearby Buiders	Agent set (builders_nearby)	Mental state about all the other builders that in a range of β patches around an agent. This mental state could be used if an agent want to collaborate with another nearby agent. However, this state will not be used in the final version of the model.
Mental State Found Shoreline	Boolean (just_found_shoreline)	Mental state that is true if an agent just observed a shoreline patch.

2.2.2 Beliefs

While the agents are solving the problem they have beliefs and mental states about the environment and the the state of the depots and the embankment. We made a difference between mental state and beliefs; the main difference between a mental state and a belief is that mental state are properties we needed for our specific Netlogo implementation. For example: the mental state *Refill Action*, is a Boolean value we needed so that the agent will remove this depots from its beliefs in the next tick. In other program languages this could be implemented in other ways. The beliefs are states about the world every agent should have to solve this problem, regardless the implementation or the program language. In table 2 an overview can be found of all the mental states an agent could have during the simulations.

In table 3 and 4 an overview is presented of all the different beliefs an agents has during the simulations. For every belief a short description will be given how this belief arise from observations from the environment or from other beliefs.

Table 3: Beliefs about the environment an agents has during the simulation

Beliefs	**Implementation**	**Description/Explanation**
World Model Beliefs Coastline	List with patches (belief_costline_patches)	Beliefs about the locations of coastline in the environment. This is a set of all the unqiue observed shoreline patches in the world by the agents. Every time an agents observe the world, this belief will be updated from this observations
World Model Beliefs Depots	List with patches (beliefs_depots)	Beliefs about the locations of the depots in the environment. This is a set of all the unqiue observed depot patches in the world by the agents. Every time an agents observe the world, this belief will be updated from this observations
World Model Empty Depots Beliefs	List with patches (beliefs_empty_depots)	Beliefs about the locations of depots that are empty in the environment. This list will be updated every time an agent has found a empty depot. When a agents observe that a depot is empty, it will add this depot to the beliefs of empty depots.
World Model Belief Depots	Boolean (belief_all_depots_found)	Belief if all the depots in the environment has been found. If true, the world model about the depots is complete. This belief becomes true is the length of the list beliefs_depots is as long as the amount of depots in the environment.
World Model Belief Refill Depot	Variable (belief_depot_to_refill)	The agent has the belief that it has to refill this (depot in front of the agent) depot. In the 'basis model' this becomes true if the agents beliefs all depots are empty (e.g. the length of the list with empty depots is as long as the amount of depots in the environment), in one of the extended models this becomes true if the agent just observed a empty depot.

Table 4: Beliefs about the environment an agents has during the simulation

Mental states	**Implementation**	**Description/Explanation**
Agent Model Belief Carrying Resources	Boolean (belief_carrying_resources)	Belief whether or not the agent is carrying resources. Becomes ten if an agent picked up resources to build an embankment part, false if an agent just build an embankment part. When this belief is true the agents moves slower trough the world, because it is carrying heavy resources.
Agent Model Belief Working Alone	Boolean (belief_working_alone)	This one is true if the agent is working alone. This belief initial was meant for the case that agents are working together. However, this belief will not be used in the final version of the model.
Agent Model Belief Carrying Resources	Integer (belief_carrying_resources)	Belief about the amount of resources an agent is carrying to a depot.

2.2.3 Intentions

While the agents are working towards their goal (discover the world and build an embankment) they have different intentions to achieve this goal. To solve the entire problem a lot of different subtasks and activities should be performed. All these subtasks and activities are modeled as different intentions. In table 5 an overview is presented of the intentions an agent could have while it is exploring the world.

In table 5 and 6 an overview is given of all the different intentions an agent could have during the simulations. We created two different sets of intentions; one set of intentions that an agent could havee while it is discovering the world (table 5), and one it could has while it is building embankments (table 5).

2.3 Communication implementation

To make cooperation possible, we implemented a communication system. With this communication system the agents can send messages among each other.

To avoid conflicts in intentions (e.g. that agents will work on the same building spot) and to avoid wrong world models (e.g. a agent has the belief that a depot contains resources will it is empty) all the agents send all their observations and actions to all the other agents. In table 7 an overview can be found of all the communication actions the agents can perform. Using this communication should result in a better, faster and more efficient way to solve the problem.

Table 5: Intentions an agent could has while it is discovering the world

Intentions	**Implementation**	**Description/Explanation**
Move along shoreline		If an agent arrived at the shoreline (e.g. the agent observes an shoreline patch at *patch-headed 1*) he has to explore the whole shoreline. So this means he is going to walk along the shoreline to explore it completely.
Explore world	Standard intention of the agent while the world model of the agent is not complete.	When the world model of the agents is not complete yet, explore the world randomly. When exploring the agents moves and observe the area in radian N around himself.
Move to shoreline		An agent just observed a shoreline patch (*just found shoreline* mental state), now he gets the intention to move to this patch (to the shoreline) to observe the whole shoreline.

Table 6: Intentions an agent could has while it is building the embankment

Intentions	**Implementation**	**Description/Explanation**
Find closest depot	An agent could have one intention per tick, therefor all the intentions are modeled as a string variable	If the agent has the belief he is not at a depot at the moment and he is not carrying any resources, then it needs to find the closest depot to pick up resources.
Pick up resources		If the agent beliefs that it is not carrying resources at the moment, and it observes a depot one patch ahead. Then it needs to pick up the resources from the depot.
Find building spot		If the agent beliefs it is carrying resources it needs to find the closest (empty) building spot to build an embankment part.
Go to building spot		If the agent beliefs it is carrying resourcess and it has a mental state about the closest building spot, it needs to move to that building spot.
Build embankment		If the agent beliefs it is carrying resources and it observes a its building spot one patch ahead. It will build the embankment part.
Refill depot		**Basis model:** If the agent has the belief that all depots are empty, the agent needs to refill the last depot it visited. **Extended model:**When an agent observes a empty depot, it will get the intention the refill this depot directly.

Table 7: Communication actions a agent could preform

Communicated Information	**Implementation**	**Description/Explanation**
Observed Depots	List with patches (msg_out_b_depots_	Every time an agents observes a depot, it will add this depot to its beliefs if it has not observed this depot before. Besides that, the agents will send its observations to all the other agents, so that they can add this depot to their beliefs as well. Hereby every depot only needs to be observed once.
Observed shoreline patches	List with patches (msg_out_b_shoreline)	The same approach holds for the shoreline patches. Every time a agents observes a shoreline patch it will communicate this with the other agents. This will result in a much faster exploration time of the world, because not all the agents need to explore the entire world, only a small part of it.
Selected building spot to work on	Patches (msg_out_b_selected_coastline_part)	When a agent select a building spot to build an embankment part, it will send this spot to all the other agents. Other agents will remove this patch from their beliefs about open building spots (e.g. the list belief coastline patches). This avoids that agents will select the same building spot to work on.
Empty depot found	Patch (msg_out_b_depots_empty)	When an agent found a empty depot, it will remove this depot from its depot belief because it is not possible to pick up resources there anymore. It will send this depot to the other agents as well.
Refilled a depot	Patch (msg_out_b_depots_refilled)	When an agent has refilled a depot, it will send the location (the patch) of that depot to the other agents. The other agents will add this depot to their beliefs of depots again.

2.4 Exploration extension

Although the agents cooperate in our standard exploration strategy, we thought we might be able to speed up building by letting some agents commence building when the other agents finish exploring the environment. In the section `research questions` we will elaborate on when we think this strategy might be more efficient than our standard strategy. The rest of this section will be dedicated to explaining our implementation.

The agent that finds the shoreline first will message the other agents it has found the shoreline. Usually, at least one depot has been found at this point, which means that the agents know te location of the depot and some coastline patches. At this point all agents that didn't find the shoreline first, will change their desires to `build embankment`. The agent that found the shoreline first will continue to explore the entire coastline, after which it will, if necessary, finish exploring until all the depots are found. When everything is found the agent also changes its desire to `building embankment`. In the case that the shoreline is found before a single depot is found, every agent continues exploring until one depot is found.

However, what will happen if multiple agents find the shoreline at the same moment and therefore message about them finding the shoreline during the same tick? when this happens the agents need to negotiate about who found the shoreline first, because we only want one agent to continue the exploration if necessary. We decided that it doesn't matter which agent continues exploring, therefore, we assigned all agents with a unique number during initialization that refers to its place in the hierarchy (it is implemented as a belief). When multiple agents find the shoreline first, the agent highest in hierarchy (lowest number) will be accredited with the discovery. This means that our agents negotiate with arguments of authority.

3 Research Questions & Hypotheses

To test and validate the functionality of the Multi Agent System we propose a number of research questions. The different research questions examine both the performance of the systems and the behavior of the agents. After this we will discuss our hypotheses on these research questions.

3.1 Research Questions

For this research project we propose the following research questions. Each research question is divided in one or more sub-questions.

1. Do agents solve the embankment building problem more efficiently when they refill a depot directly after discovering the depot is empty, or, when the agents wait until all depots are empty before refilling?
    (a) What kind of different behavior will emerge if the agents start refilling the depots directly after they found a new depot instead continue until there is no depot left with resources.
    (b) Will the embankment building problem be solved more efficient when an agent starts refilling directly after an agent found a empty depot?

2. Under which circumstances does the extended exploration approach outperform the standard model?
    (a) Does the number of builders in the model increase the favor-ability of one of the two exploration approaches?
    (b) Does the size of the world increase the favor-ability of one of the two exploration approaches?

3.2 Hypothesis

To answer our research question we have formulated hypotheses for all research questions.

3.2.1 Hypothesis question 1

3.2.2 Hypothesis research question 1a

When the agents refill the depots directly, there will emerge different behavior among the agent then when they continue until there is no depot left with resources. This because agent do not have to look for new depots any more when they found an empty one. And second, there will be a higher supply of new resources; this will result in less waiting time for new resources for the agents.

3.2.3 Hypothesis research question 1b

When the agents refill the depots directly, instead of continue until all the depots are empty, the performance of the system will increase significantly. This because of the fact that in general there will be more resources available in the environment (because now there is a more continue supply of new resources). Therefore the agents can go to the closer depots which decreases time of travel. As a result the total runtime of the model will be significantly lower when the agent refill the depots directly.

3.2.4 Hypothesis question 2

We implemented our extended exploration approach because we thought it will decrease the overall runtime of our simulation. First of all, we thought that especially when there are many builders, observation is not optimal. This because when exploration is done together in a random fashion there are moments that agents observe the same patches. In the case that only one explores and the others build this will not occur anymore. Therefore, we think that $(explorationtime) * (numberofagents)$ is higher for the standard approach. However, because one agent needs more time to find all the depots, it could occur that the other builders use a depot that is at a sub optimal location (further from building patch). We think that the latter is less of an influence on the overall model time, therefore, our hypothesis for research question 2 is: "The extended exploration approach will decrease the time needed for the embankment to complete".

3.2.5 Hypothesis research question 2a

Because, the more agents there are in the model, the more their observations will overlap. This results in decreasing individual-agent-efficiency in the standard approach when the number of agents increase. Which in turn means that the favor-ability for the extended approach increases when the number of agents in the model increase. Therefore, our hypothesis for 2a is: "the differences in runtime between the standard exploration approach and the extended exploration approach increase with the number of agents in the model".

3.2.6 Hypothesis research question 2b

When the size of the model increases, exploration time increases, therefore, our hypothesis for 2b is: "When the model size increases the absolute difference between completion time for the standard exploration approach and the extended exploration approach also increases".

4 Simulation Results

4.0.1 Results research question 1a

During the simulations of research question 1b we observed the behavior of the system to discover different patterns in emerging behavior between the two Multi Agent Systems. With emerging behavior we mean the behavior the system shows when it is solving the problem.

As described in research question 1a we will compare the behavior the system shows when agents continue building embankment parts until all depots are empty, and an approach where the agents start refilling directly after they discovered that a depot is empty. In this section we will describe and explain our observation results for the two simulated refill scenarios.

Wait until all depots are empty

1. The last depot that finally will be empty is always the depot with the greatest distance to the shoreline. This because the agents always go the the closest depot who has enough resources left. Therefore the first depot that will be refilled is (in most cases, the last one that became empty) the one that has the biggest (x-direction) distance to the shoreline (we call this depot *depot empty*). When the *depot empty* is refilled the agents will start to collect resources from this depot again (because the remaining depots are still empty) until this depot is empty again. This process will continue until the whole embankment has been built. This is very inefficient, because from the moment that *depot empty* has been refilled (in almost all cases) , *depot empty* will be the only depot in use for the agents. The other ones will always remain empty. This is not in every simulation the case, sometimes two depots become the *depot empty* at the same time. Then this two depots will be refilled by the agents. However, still most of the depots remain empty after all the resources of this depots are gone.

2. Let's assume an environment were there is one depot left with resources and two agent (or more) are moving to the same depot, lets call these two agents *Agent A* and *Agent B*. When *Agent A* arrives first at the depot, and it discovers that this last depot is empty it will send a message to the other agent that this last depot is empty as well and it starts refilling it. *Agent B* will wait then until *Agent A* has refilled the depot, and then *Agent B* will continue his way to the depot. The reason that *Agent B* will wait until *Agent A* is finished with refilling the depots is because *Agent B* has no belief anymore about depots which contain resources. Only when an agents arrives at a empty depot and all other are empty as well it will decide to refill it. In other cases the agent will look for another depot, or wait until another agents sends a message that it refilled a depot. This behavior is also very inefficient.

3. When two agents *Agent A* and *Agent B* are moving to the same empty depot and *Agent A* arrives first, it will tell the other agents that this depot it empty. If *Agent B* has any beliefs about other depots who does have resources left, it will continue its way to one of that resources. This cooperation/coordination between the agents improves the improvements of the system as unit. Because not every agents has to discover by itself that a depot it empty it could spend more time to gather resources from other depots.

Directly refill the depots when observing one is empty For the other developed system we did observations as well. We looked if the system is responding differently as the situations described in point 1,2 and 3 in the paragraph 'Wait until all depots are empty'. Our observations are described in the list below.

1. Because agents refill depots directly after one discoveres that a depot is empty. Not always the depots that has the greatest distance from the shoreline will be refilled first (emergin behavior from first model), Therefore all depots will be used more equally. Now the one that is the closest to the beach will be empty first (because the agents always have the intention to go to the closest depot). One agents will refill the empty depot, and the other ones continue with building embankment parts. This behavior is much more efficient then the behavior we observed in the other system where agents continue until all depots are empty. This because the agents has to travel less with this approach.

2. Because all depots are refilled directly after an agents has discovered that this depot is empty, the chance that all depots are empty at the same time is almost 0 (unlike then the other approach). Therefore none of the agents have to wait anymore while other agents are refilling the depots. Also this behavior is much more efficient then the behavior we observed in the other system.

We can conclude that when agents are refilling the depots directly much less overhead problems will occur. Agents don't have have to wait until other agents have refilled empty depots (in case all depots are empty). Also more depots will be refilled using this approach and none of them will remain empty the whole simulation. This will result is a shorter traveling distance for the agent which results in better performance for the system.

4.0.2 Results research question 1b

To answer this research question we performed 10 simulations in both models; one model in which the agents continue with building until all depots are empty and then start refilling them. And one model where an agent starts refilling the depots directly after it discovered a empty depot. We measured the amount of ticks from the initialization of the model until the desire changes to "drinking beer with buddies". We took ticks as a quantification of the performance of the system. Another possibility could runtime. However, runtime is very dependent on background processes of the machine where it is running on. Therefore we choose ticks instead, the better the system performs as a unit (the *Coherence of the system*), the less ticks it needs to solve the problem. In table 8 an overview is given of the 10 simulations for each model. We used the following model parameters for both models:

1. number of depots = 3
2. coastline_bumpiness = 0
3. vision_angle = 19
4. amount_of_workers = 19
5. resources_per_depot = 19

Table 8: results

System	Mean amount of ticks to solve the task	Standard deviation
Wait until all depots are empty	8079	368.85
Refill directly when found empty depot	6788.69	560.14

When applying a two-tailed significant test on the results in table 8 with a significance level of 5% we get a p-value of 0.000009. Therefor we can conclude that a Multi Agent System where agents start to refill the depot directly after one of them discovered that a depot is empty is significantly more efficient then when continue until all depots are empty and then start refilling.

We also measured the amount of resources available in the environment during the whole simulation. This results can be found din figure 2 and 3. Both visualisations are made during one entire simulation. The figure could differ for other simulations, but on average they are almost the same every simulation.

You can see that in figure 3 the amount of resources never becomes 0, and that there always are resources available for the agents to build embankment parts. Therefor an agent never has to wait until it can pick up it resources from the depot, which resulted in a faster solving time for the problem. You can also see that the average amount of resources available in the environment is much higher in figure 3 then in figure 2. This resulted in a a faster solving time for the problem as well.

Figure 2: The total amount of resource in the environment when the agents wait with the refill action until all depots are empty.

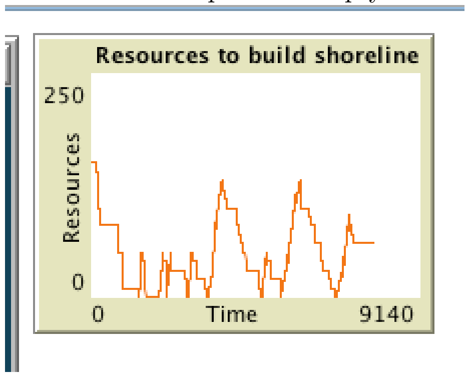
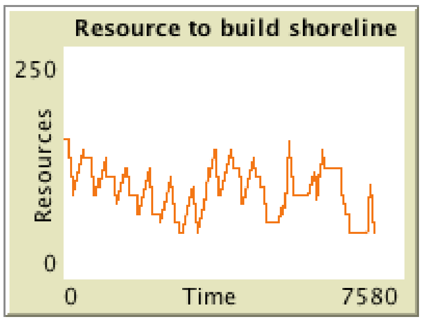


Figure 3: The total amount of resource in the environment when the agents refill an depot directly when they found a empty depot.



4.0.3 Results research question 2

To test our hypotheses we ran the standard exploration approach model and the extended exploration model 50 times for multiple parameters. Some parameters were unchanged:

1. number of depots = 3
2. coastline_bumpiness = 0
3. vision_angle = 18

We counted the number of ticks from the initialization of the model until the desire changes to "drinking beer with buddies". Even though it is no the actual runtime we will refer to this measure as `runtime`.

4.0.4 Results research question 2a

In table 9 the results for research question 2a are depicted. To compare the two models we look at the mean runtime. To compare them we look at the percentual difference from the standard model in runtime. We also look at the standard deviation σ to see how the runtime is distributed. Finally, we look

Number of agents	expl. model	mean	σ	difference	p-value
2 agents	standard	4533	395	-1.0%	0.612
	extended	4489	464		
4 agents	standard	2312	236	-4.3%	0.042
	extended	2213	246		
6 agents	standard	1571	203	-4.4%	0.068
	extended	1502	168		

Table 9: Here we look at the difference in runtime for the two exploration approaches with different numbers of agents. We see that for 2 agents, there is no significant difference. For 4 agents we do see a significant difference of 4.3%. For 6 agents we also see that the mean decreases by 4.4%, however, this result is not significant.

Number of agents	expl. model	mean	σ	difference	p-value
small world	standard	2312	236	-4.3%	0.042
	extended	2213	246		
large world	standard	5089	489	1.2%	0.575
	extended	5152	615		

Table 10: Here we look at the difference in runtime for the two exploration methods for different sizes of the world. For these tests we used 4 builders because we know this yields a significant difference in the small world. We see that the standard deviation increases and the mean decreases when the extended exploration method is used in the large world. However, the difference for the large world is not significant.

whether the difference in runtime is significant. We do this by using a two-tailed t-test. The runs are performed on a world with a grid size of 80 x 80. The first thing we notice is that the extended model is faster for all 3 situations, however, only in the situation with 4 agents the difference is significant. Although the difference between the two approaches seems to increase when the number of agents increases, we cannot provide conclusive proof for this because we lack statistical significane. More extended research with more runs is necessary to conclude whether the difference for 6 agents is significant or not. If this is proofen, we should determine whether the difference between 4 and 6 agents is significant also.

4.0.5 Results research question 2b

In table 10 the results for research question 2b are depicted. To answer the question we used two map size 80 x 80 and 120 x 120. Contrary to our hypothesis the extended model doesn't save more time in a larger world. Where the extended model saves 4.3% time in the small world the extended model is 1.2% slower in the large world. However, the latter difference is nog significant. It seems that for our model we should reject our hypothesis. It is interesting to see how the standard deviation is rather large for the extended model, from this we conclude that the exploration time for the agent that found the shoreline first in the extended model varies a lot which causes the model to perform less than the standard model. Because the map-size is larger, the effect of known depots being at a sub-optimal location is also larger. This could also explain why the extended model performs less on the bigger map.

5 Conclusion

For this project we implemented several Multi Agent Systems. These multi Agent Systems will try to build an embankment as fast as possible in an unknown environment. We implemented the agents according to the BDI-model, besides that we added some extra mental domain specific state. We also implemented a communication system such that agents could share task results with each other to avoid conflicts in beliefs and tasks.

First we started with a 'basis model' where the agents cooperate to semi-randomly discover the world. If the agents have explored the whole world they start with building the embankment. After this we implemented some extensions on top of that model, to see if these extensions result in better system performance. One of the extensions is a a system where not all agents share the same desires in such a way that labor is divided. The second extension is that agents will not continue with building embankment parts until all depots are empty, but directly start to refill a depot when one of the agents observe an empty depot.

The approach that agents start to refill a depot directly after finding an empty depot results in significant performance improvements for the system as a whole. Agents do not have to wait anymore until all depots are refilled. Another positive behavior that emerges from this approach is that all depots are filled more equally, and not only the depot that became empty. Agents now have to travel less trough the environment to get resources and in general there are (on average) more resources available to build embankment parts. This extention resulted in a significantly lower runtime to complete the embankment.

In our second research question where we look at the extension of our model where we let one builder continue exploring while the others begin building. We found that there is a situation with 4 builders in a small world where the extended model significantly improves the performance of the system. We haven't found any situations where the extended performed significantly worse. Regarding hypothesis 2a "the differences in runtime between the standard exploration approach and the extended exploration approach inceaste with the number of agents in the model". We can neither accept or reject the hypothesis. We think further reseach with a higher number of runs is necessary to rule out the possibility of accepting this hypothesis. The last hypothesis, where we said that the absolute difference between the standard and the extended model increases in favor of the extended model, is rejected. This because the simulation didn't find a significant difference between the models for a large grid-size anymore.

6 Discussion

After we finished this project there are some points of discussion left. We will discuss some further development options, and some extensions we tried to make which where not working properly.

6.1 Cooperation

We did a lot of experiments with a system where agents where working together to carry the resources. When agents are carrying resources together they can move a lot faster. We implemented a system where agents could ask for help to nearby agents (e.g. agents in a range of 10 patches). If one of the nearby agents is available (it is not carrying resources) itself, this agents will always help the other one. After a lot of testing we decided to not use this cooperation in the final versions of our model. The first reason was that it was to hard to setup a good communication between the agents to ask for help. We tried a lot of different approaches for this communication but none of them was working properly. Second, the environment seems to be to small to let this cooperation be really efficient. When *Agent A* decides to help *Agent B* it first needs to move to *Agent B* and then they could work together. The environment we used for this system was to small to let this cooperation be really efficient.

6.2 Further development and possible improvements

Due to time limitations we were not able to implement the complete agent system we had designed during the beginning of the project. Also the research we did raised a lot of new questions for which we didn't have the time to investigate. The following points can be done to improve the system:

1. All the embankment parts are the same in this simulations. For a more realistic simulation we would like to use different embankment parts. Those parts require different resources to build, and agents could use more of these parts to build a higher embankment.

2. It should be possible for agents to work together. This means that they can carry resources together so that they can bring more and faster resources with them.

3. Now the agents move randomly through the environment when they are exploring. For a better approach we can define a multi agent system where areas are designated to every agent to explore. Also, moving randomly could be improved by defining a search strategy.

4. The process of refilling the depots could be more realistic. In this systems agents are able to magically create resources. In a more realistic version agents should gather resources from one main depot, or collect them in the environment.

5. The shoreline should be implemented as a bumpy line. In the first prototypes we made a bumpy shoreline but it was too hard to make a system that could efficiently explore this shoreline.

6. The embankment should be wider then one patch, and it should be possible to build a embankment with a different height. Now every embankment part is one patch wide, and all of them have the same hight.

References

[1] Wilensky, U. *Netlogo*, http://ccl.northwestern.edu/netlogo/, 1999.

[2] Rao, Anand S., and Michael P. Georgeff. *BDI agents: From theory to practice.*, ICMAS. Vol. 95. 1995.