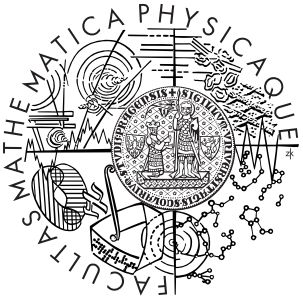
Charles University in Prague
Faculty of Mathematics and Physics

BACHELOR THESIS



Vojtěch Kopal

Komunikace a paměť pro plausibilní agenty
Communication and memory in plausible agents

Department of Theoretical Computer Science and
Mathematical Logic

Supervisor: Mgr. Ondřej Sýkora

Study programme: General Computer Science

2011

Na tomto místě mohou být napsána případná poděkování (vedoucímu práce, konzultantovi, tomu, kdo půjčil software, literaturu, poskytl data apod.).

Prohlašuji, že jsem svou bakalářskou práci napsal(a) samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne Vojtěch Kopal

Contents

Název práce: Komunikace a paměť pro plausibilní agenty
Autor: Vojtěch Kopal
Katedra (ústav): Katedra teoretické informatiky a matematické logiky
Vedoucí bakalářské práce: Mgr. Ondřej Sýkora
e-mail vedoucího: mail@ondrejsykora.com

Abstrakt: V předložené práci jsem se zaměřil na porovnávání různých implementací pamětí pro plausibilní agenty v multi-agentním prostředí. Vytvořil jsem simulaci, ve které se snaží jednotliví agenti naplňovat svou potřebu jíst. Aby uspěli musí se naučit, kde jsou zdroje potravy, k čemuž jim slouží implementovaná prostorová paměť a schopnost komunikovat mezi sebou. Agenti jsou posléze hodnoceni podle úrovně hladu v průběhu simulace.

Klíčová slova: plausibalní agenti, prostorová paměť, growing neural gas

Title: Communication and memory in plausible agents
Author: Vojtěch Kopal
Department: Department of Theoretical Computer Science and Mathematical Logic
Supervisor: Mgr. Ondřej Sýkora
Supervisor's e-mail address: mail@ondrejsykora.com

Abstract: In the present work we I have focused on comparation of different implementation of memory for plausible agents in multi-agent environment. I have created a simulation whereby the agent are strangling to fulfill their need to eat. To succeed they have to learn the locations of food resources using the implemented spatial memory and an ability to communicate with each others. Later the agents are evaluated according to the level of hunger throughout the simulation.

Keywords: plausibal agents, spatial memory, growing neural gas

Introduction

In the modern society, the amount of information is far behind what one can remember or even process. In understanding this, one realizes the integral importance of delegation of thinking and information processing in a group. Decision making in groups and teams has been covered in the literature to some extent. [4] Supposing we have limited memory capacity, information required in decision making has to be distributed and communicated between people efficiently.

Our decisions can be either conscious or subliminal, depending on our needs or drives - whilst the former is connected with human behaviour, the latter is used for plausible agents. Just like in microeconomics, we can use utility as a measure of relative satisfaction [15] and see how one manages to fulfil their needs. In satisfying these needs, the knowledge stored in our memory and updated regularly is a key tool. With infinite memory, problems of information storage would be eliminated and with necessary information available at all times, provided it had once been acquired. Our memory, however, is limited.

What I mean by saying that our memory is limited is that we are not able to remember everything. Certain pieces of information are fading away with time or as one is learning new facts. I set out to understand whether and how intensive communication can substitute insufficient memory space at a constant level of utility. It is obvious that adding the ability of communication improves the agents' chances to survive in the environment.

The goal of my work is to observe efficiency of the agents in their struggle to fulfill their needs using different implementations of spatial memory. Such agents will also be able to communicate with each others and thus possibly improve their chances.

This thesis consists of six parts. First, I will introduce the agent and possible memory implementations based on specific examples *(Chapter 1)*. Then I will explain algorithms to be used in the program, such as growing neural gas *(Chapter 2)*. In *Chapter 3* I will describe the simulation, the agents, their memory and their communication. I will further describe the concrete implementation of the

introduced algorithms in *Chapter 4*. All experiments are about to be presented in *Chapter 5* and the outcomes will be discussed in *Chapter 6*.

Chapter 1

Related work

I will use this chapter to provide an insight into the world of *agents* and spatial memory. I hope you will not be disappointed as you will not find *007* in the following lines.

1.1 Agents

There are several ways to explain what or who an *agent* is. Apart from systems of agents used in philosophy or sociology, we can see a first modern use of agency and agents in economy where economists have substituted a human with a simpler agent. They intended to simplify their economic models in order to carry out general simulations. Buyers and sellers are typical examples of agents used in simplified market model in microeconomics. In this context, agents are entities in the model which can react to a given context.

In the case of artificial intelligence, as Wooldridge *et al.* has clarified there is no exact definition of what an agent is. **??** We can, however, use the definition of an agent found in [12]. It could not be simpler:

Definition 1 Agent *is just something that acts.*

Of course it is as general as it could be and for our purposes this is too simple, so we will use another definition which meets better the context of this work.

Definition 2 Agent *is something that senses the environment and affects it using its actuators.*

Having defined an agent, we can now begin to distinguish specific kinds of agents. In this thesis I will use several slightly different terms in order to refer to agents: *rational, autonomous, plausible and believable.*

A *rational agent* refers back to economics, where we can find a definition of rational behaviour. Even though it is rather a hypothetical model since people are usually irrational in their decisions from the perspective of economics, the definition used in economics suits our needs well. A rational agent is one that acts as if balancing costs against benefits to arrive at action that maximizes personal utility (Milton Friedman (1953), Essays in Positive Economics). In simple terms, the agent does what is or perhaps might be the best for him based on his current knowledge of the world.

Rational behaviour might, however, be understood in a completely different way. *Plausible agents* are agents where the basic approach is to implement human-like internal processes. A well-known example is that of neural networks, which could be used for a simulation of brain processes, although these are usually used in a simplified way. Since it is really difficult to implement a completely plausible agent, there are many research teams focusing on specific parts of the complex human nature.

Believable agents are personality-rich autonomous agents with the powerful properties of characters from the arts [1].

The last type of agents is referred to as autonomous. An *autonomous agents* are able to accomplish a useful task or are effective problem solvers.

One additional term should be defined in reference to agents:

Belief-Desire-Intention (BDI) agency model implements three parts: agent's belief, his desire and his intention. These three parts are combined in reasoning. A BDI agent is a particular part of a **bounded rational agent**, which uses the three parts to separately prepare plans which are later executed. What distinguishes BDI from a simple reactive agent is that a reactive agent creates an immediate decision based on the current state of environment and the inputs of his sensors. The BDI agent, on the hand, uses all three parts:

- **belief** represents the agent's informational state, for example sensory inputs and information in his memory,
- **desire** is the agent's motivational state, what he needs to approach, for example he is hungry and he needs to find appropriate food,
- **intention**, on the other hand, is his immediate decision how he attaing the goal he desires, in other words it is execution of plan, for example next move.

1.2 Spatial resource-bounded memory

A memory is something that changes a reactive agent into an agent with an ability to learn. It can be used for learning consequences of agent's acts, conditional dependencies in the agent's world implementing Bayesian networks [11], or spatial information about the environment. The latter is a kind of memory we used for agents in our simulation.

A spatial memory is used when the agent needs to navigate usually in a two or three dimensional space. In short, it is a component of the agent which tells him where to go when he needs to collect an item that is available only on specific locations, or to perform an action that must be performed at certain locations. There are several different approaches and several examples are going to be covered in this section. I am going to introduce several existing implementations of spatial memory. Mainly I will attempt to address whether and how they have dealt with bounded resources - either due to implementation restrictions, or when approaching plausibility in their models.

1.2.1 Resource-bounded reasoning

Rational agents cannot be expected to be able to compute a load of data in a reasonable time or in a time in which the environment doesn't change much. That is why we have to take into account bounded resources when simulating plausible or real agents. We want to avoid computations of plan that take a long period of time during which the environment changes significantly. As noted by Bratman *et al.*, plan computation can be separated from plan execution, whereby the plan is prepared over several executions. [5] In this case we need either to be able to perfectly predict the future, or base our plan on data, which does not change at all or remains constant for the given period of time.

1.2.2 Short-term and long-term memories

Generally, when we talk about remembering something, two terms need to be defined: a long-term memory (LTM) and a short-term memory (STM). Both of these describe a capacity for holding a certain amount of information in mind. Apart from the varying amount of information stored, the memories differ in the availability of such information and the period of time over which these memories last.

In 1968 Richard Atkinson and Richard Shiffrin suggested a memory model divided into three components [2]:

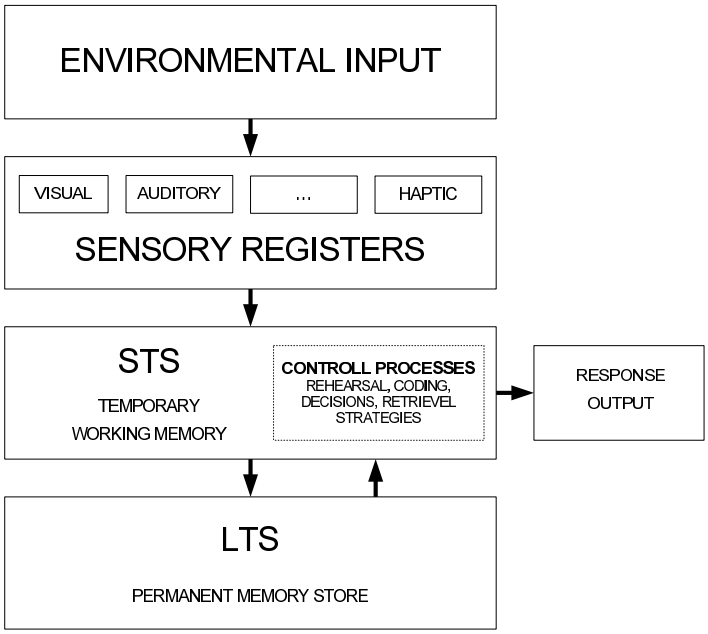


Figure 1.1: Information flow in the memory system, adopted from Atkinson *et al.* [3]

(a) *sensory register*, able to store only a relatively small amount of data for a short period of time,

(b) *short-term store* with the ability to store also a limited amount of data, but for a longer period of time,

(c) *long-term store* with a large capacity for a nearly unlimited amount of time.

We use **short-term memory** to store information for a relatively short period of time. This could be seconds or minutes. A number of entites we are able to hold in the STM was studied by George Miller in 1956. [13] The outcome of his work was the magical number 7 ± 2, which is the number of similarly small things we can hold in STM.

A **long-term memory**, on the other hand, is used to store information we do not think of consciously. The pieces of such information are important for the everyday life. The capacity of LTM memory is unknown, as there is no way how to measure it.

1.2.3 Computational memory architectures

Computational memory architectures for autobiographic agents interacting in complex virtual environment suggested by Ho *et al.* work with both short-term

and long-term autobigraphic memories, where they have studied the agent's ability to survive in comparison to a purely reactive agency model. [8] Moreover, they researched whether the narrative communication amongst agents somehow positively influence those agents. They have separately experimented with three types of agents: purely reactive (PR), short-term memory (STM) and long-term memory (LTM). A purely reactive agent walks randomly around the environment avoiding obstacles and searching for resources in order to to fulfil his needs.

STM agents in further extend the model of purely reactive agents and add a track-back memory system in addition to the reactive behaviour. Each time an agent deals with an event (e.g. collision, or resource object), he confines this information into his memory. They refer to this as an event-based memory entry making mode. These events are kept in a linear list of a finite size and the oldest events are deleted. The memory is used when an internal variable, i.e. his desire, is over threshold. That is the moment when agent searches in his memory for an information about relevant resource object. If he succeeds and finds what helps him to fulfill his desire, he undoes backwards all memorized states leading to the object he has found. Therefore, what a STM agent actually puts into memory is his current state: where he has been and what he has perceived. In order to account imperfection in retrieving information from short-term memory, they introduced noise to alter the values.

LTM is mostly based on psychological autobiographic memory models. There are three parts that are involved in the reasoning process: Event Specific Knowledge (ESK), Event Reconstruction Process (ER) and Event Filtering and Ranking Process.

The conclusion presented by Ho *et al.* were not much suprisising as it confirmed that more complex memory implementation leads to better results. However, there were several interesting outcomes. For example the LTM agents without communication were significantly influenced by the presence of more agents and thereby dynamically changing environment without their endeavour. Such a negative influence is restrained when the agents communicate and share their knowledge. To sum up, the experimental studies has shown that more complex LTM architecture extends the lifespan of an agent compared to pure reactive or STM architecture and also that communication helps the agents to challenge a dynamically changing environment.

1.2.4 How Place and Objects Combine?

Brom *et al.* mainly focused on plausible behaviours while searching for things in structured spaces such as flats.[6] Unlike others who previously researched the area of spatial memory for plausible agents, they suggested a model for an agent which could successfully live in a dynamic environment with objects that could be moved without the agent's involvement.

In this model, the environment consists of abstract and specific areas such as rooms and pieces of furniture. These areas are combined into a tree structure used in the model where, for example, the flat is a root node, the immediate child nodes are rooms and, finally, specific pieces of furniture are child nodes below room nodes. Four different categories have been created for objects to reflect varying probabilities of changing object's location which in fact simulates a presence of another agent in the environment. The observed agent therefore does not leave the flat and it is there alone.

During their experiments, what is interesting for my work is the subsequently observed ability of the model to emerge (**nechapu emerge**) the searching rules from scratch, to relearn the rules in the case a particular setting changes, **and if the merged rules meet with the human behaviour, i.e. they are believable. (nedava smysl)**

1.2.5 Inspirations for my work

The suggested RTM and LTM models of agents as described by Ho et al. are of interest in my simulation. [8] There has to be a couple of minor changes, though, as I am working with a simpler environment compared to the one used in the work described above. The changes will influence the structure of memory records in both RTM and LTM. Also the communication protocol will be different and I am going to introduce it later in this thesis. Although it is not going to be part of this particular thesis, comparing the memory models might prove interesting.

In my project, the environment I want to use does not differentiate the area as the one assumed by Brom *et al.*, that means it is homogeneous. [6] For the purpose of implementing an agent with a similar spatial "What-where" memory model, I will use a different spatial organization, which could be similarly structured into a grid. The suggested model, however, is usable for objects which are moved around the environment and I do not have such objects. Objects in my environment are generated around a distribution place and locations of those places are something that could be learnt using “What-Where” model.

Chapter 2

Used methods and algorithms

In previous chapter I have introduced several kinds of agents, how they can be used and also what a spatial memory is. I have briefly prepaired you for the next chapters, where I will explain my contribution to this area. This chapter is going to cover the used algorithms and computational methods I have studied and implemented in my work.

The first subsection dissects the implementations of agents' memory and in detail describes its fundamental parts. Both the Growing Neural Gas and the Grid are used as memory storage to handle spatial information about the environment with bounded resources.

2.1 Growing Neural Gas

A learning process is something what we have been experiencing every day since we were born. In this process we usually know whether we have mastered a new skill by trying it and being rewarded. Either at school, or at work. Such a procedure, whereby someone is rewarded or punished after he does something, is called a *reinforcement learning* and it expects someone - a supervisor - who gives the rewards. On the other hand, sometimes we need to be able to learn without that supervisor. Growing Neural Gas (GNG) is an example of an unsupervised learning method. GNG is a self-organizing map used to find a simpler data structure representation of its origin.

2.1.1 Topology learning

Processing an enormous amount of spatial data about an environment is computationally demanding when, for example, we want to navigate in such environment. A topology learning or recognition can help us create a representation such as a topological map, which can be viewed as a graph and which makes reasoning in the environment much easier. Rather complex understanding of topology in an indoor space using Bayesian programming has been shown. [14]

Based on competitive Hebbian learning (CHL) method [10] and Neural Gas (NG) [9], Bern Fritzke proposed Growing Neural Gas [7], an unsupervised learning method for finding a topological structure which reflects the topology of the data distribution. Although the combination of both CHL and NG is an effective method for topology learning, there are some flaws in practical application as it requires an initial set-up of number of nodes/centres that are used. This fact prevents the method from adequately describing the topology when a different number of nodes would work more effectively.

Fritzke described an algorithm that uses a set of nodes and edges that connects the nodes. A simplified description of algorithm in the context of two-dimensional space follows [7]:

1. Add two nodes at random position onto canvas.
2. Generate input signal based on the data distribution (its probability density).
3. Find the nearest node n_1 and second nearest node n_2 to the signal.
4. Increment the age of all edges leading from node n_2.
5. Add the squared distance between the input signal and the nearest unit in input space to a local counter variable $\Delta error(n_1)$.
6. Move node n_1 and its topological neighbors towards the signal (according to parametres $epsilon_{winner}$ and $epsilon_{neighbour}$).
7. Remove all edges with an age larger than a_{max}.
8. Generate new nodes (see [7]) using variable *alpha*.
9. Decrease all error variables by multiplying them with a constant *beta*.
10. Go to 2.

```
procedure Score()
  (px, py, pvar) <- GetEstimatedGauss()
  (rx, ry, rvar) <- GetRealGauss()

  sqDistance <- (px - rx)*(px - rx) + (py - ry)*(py - ry)
  sqSize <- (pvar + rvar)*(pvar + rvar)

  score <- sqDistance / sqSize

  return score
end
```

Figure 2.1: The *SCORE* method

For the purpose of this work I want to use this algorithm to learn a topology of data which dynamically changes through time. We have to set-up the variables for this algorithm *alpha*, *beta*, $epsilon_{winner}$, $epsilon_{neighbour}$ and maximal number of nodes. In the following section, I am going to introduce you to the experiments carried out with this algorithm..

2.1.2 Experiments on dynamic data

As I mentioned previously, I had to set-up the variables so as to be able to use Growing Neural Gas method properly. To achieve this, I have made a Java program which tests various combinations of variables' values and finds the best one. It has subsequently run the algorithm for a given number of steps and measured the *score* (see 2.1). Throughout the experimental simulation there are randomly generated GNG inputs following a Gaussian distribution. The value of *score* measures the difference between the genuin distribution and the estimated distribution which is computed based on the GNG data. The result is a combination of the distance between the distributions' centers and ratio of variances.

[2.1.2 proc zrovna takhle ten algoritmus, jak z toho poznam, ze tohle score je takhle ok?]

A total number of possible combinations is equal to 19712. For each of these combinations, I ran 10000 steps of the GNG learning sequence and measured the average score. The entire experiment was computed in a parallel fashion using

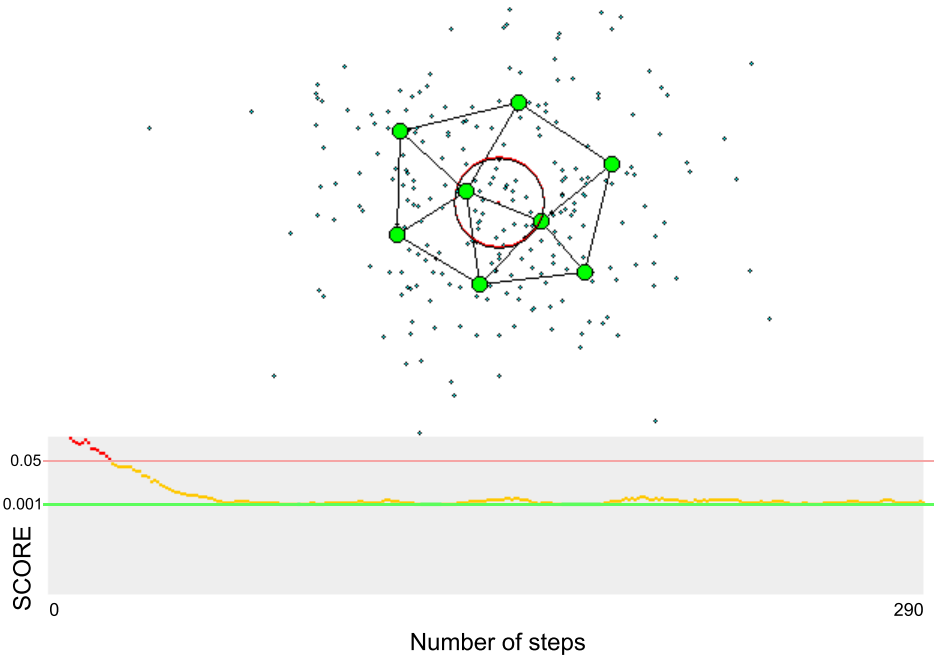


Figure 2.2: Screenshot showing the process of searching optimal variable values. The visualization has been made for testing purposes in the first place, but it nicely shows what had been lately paralelly computed. The bottom part of the picture shows the SCORE value throughout the simulation. (Red color means $SCORE > 0.05$, orange is $SCORE > 0.001$ and green is $SCORE <= 0.001$)

- $alpha \in \{0.0, 0.2, 0.4, 0.5, 0.6, 0.8, 1.0\}$
- $beta \in \{0.0, 0.00001, 0.00005, 0.0001, 0.001, 0.005, 0.01, 0.5, 0.1, 0.5, 1.0\}$
- $epsilon_{winner} \in \{0.0, 0.001, 0.005, 0.01, 0.1, 0.2, 0.5, 1.0\}$
- $epsilon_{neighbour} \in \{0.0, 0.0001, 0.0006, 0.001, 0.005, 0.05, 0.1, 0.2\}$
- $maxNodes \in \{4, 8, 16, 32\}$

Figure 2.3: Domains of variables for the experimental learning of best values.

$alpha$	$beta$	$epsilon_{winner}$	$epsilon_{neighbour}$	$numNodes$	$SCORE$
0.0	1.0	0.0050	0.0	16	$3.8 * 10^{-12}$
0.5	0.0	0.01	1.0E-4	8	$5.1 * 10^{-12}$
0.5	0.0010	0.1	0.0010	8	$8.8 * 10^{-12}$
0.5	1.0	0.0	1.0E-4	8	$3.1 * 10^{-12}$
0.8	1.0E-5	0.0010	1.0E-4	32	$7.4 * 10^{-12}$
0.8	1.0E-5	0.0050	6.0E-4	8	$4.3 * 10^{-12}$

Table 2.1: Variable values with best average SCOREs

30 threads.

The best results with an average $SCORE < 10^{-11}$ are shown in table 2.1.2.

The conclusion is that the values we have been given from this experiment and which are presented in 2.1.2 were proved to be quite unsuccessful in the main simulation. The values could not have been used, because of the bad results the GNG agents had. Therefore we have been slowly altering the values to see the immediate result in the simulation and through this way we come up with following values which works quite well.

$$\alpha = 0.8 f \beta = 1.0E-5 f epsilon = 5.0E-4 f epsilon_2 = 6.0E-4 f numNodes = 5 \tag{2.1}$$

2.2 Grid

The idea for this data structure representing resource-bounded memory is based on Brom *et al.* [6]. What differs in my work from their observed environment is that agents in my simulation act in a homogeneous space which cannot be differentiated in a way the mentioned simulation does. To solve this issue I have simply differentiated the environment into grid of 4x4, where each cell works as the place in 1.2.4.

Each cell is given two variables *positive* and *negative*, both of which are set to zero and increased throughout the simulation. When an agent sees at least half of that area determined by the cell, provided he sees any food, he increases the *positive* variable. If the agent searches for food and he cannot see any, he increases the *negative* variable.

Later when one wants to ascertain from the grid whether there is food at specific cell, it answers according to this method with parameter a to be found:

$$ANSWER = \alpha \times positive - negative \tag{2.2}$$

Similarly, I will use this structure to keep the spatial information about the environment in the simulation.

Chapter 3

Simulation and used memory architectures

In this chapter I will describe the simulation, environment and agent's reasoning and communication how it is used in later experiments.

3.1 Simulation

The *simulation* consists of a set of agents, a set of generators and a set of pieces of food. According to given settings it subsequently processes a number of steps, each of which invokes agents' life step and eventually generating new food.

The simulation can also contain a couple of monitors which observe the environment or agents.

3.2 Environment

The environment is a two-dimensional space which contains agents and food. Agents can move around and eat the food which is randomly distributed using the food generators.

3.3 Agent

As mentioned previously, an agent is an entity in the environment which moves and interact with the world around. The interaction is done through eating

food which is a part of the environment and through communication with other agents. The latter one actually changes agents' beliefs about the environment.

Each agent has his needs which influence his decisions as fulfilling his needs keeps him alive. When his internal variables of needs is higher than **.....?**

There are four types of agents each of which is different in the way they decide about next step. If one is hungry and sees food (i.e. there is a piece of food in the sight distance) then they choose to go after this food. If there is no desired food around they go searching for it and that is when the agents' actions differ.

- *random agent* moves randomly around the environment,
- *pure reactive agent* sees the whole environment, i.e. they always sees a desired piece of food,
- *grid agent* implements a memory based on clustering the space into a grid,
- *GNG agent* implements a memory based on growing neural gas.

3.4 Communication

Apart from what agent sees, there is another way how the agents gather information about the environment. They communicate. It is a quite simple way of sharing information. When suggesting an implementation for communication, I had to create a unified protocol which could have been used throughout all types of agents. I tried to keep this communication protocol as simple as possible.

Moreover, although all agents have a knowledge of a sort about the environment, they are not able to answer easily when asked about a specific food location. Since the food appears in environment according to given normal distribution, it is not clear what should be an answer to such question. Several possible kinds of answers follow.

First and the simplest answer might be the exact X,Y coordinates of the food location as it is stored in agent's mind. Additionally, there would be a noise added to such answer, keeping in mind that the answer should not be perfect and there is always a distortion and imperfection in our answers based on how a person is certain about his answer.

Another way, and perhaps a more plausible one, might be answering with a direction (an angle) with an approximate distance. What both the X, Y answer and the latter information have in common is that the answers are hard

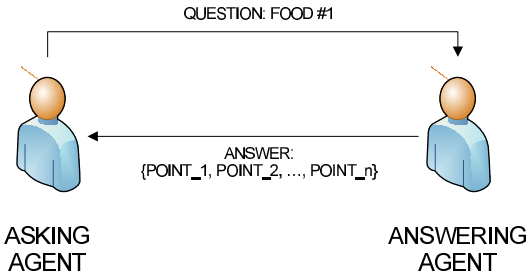


Figure 3.1: Simple communication protocol

to combine with the learning method used in GNG memory. GNG works with samples of data which subsequently influence the neural network. Both kinds of answers could be used if agents would ask more often or the agent's answer would be a sample of points straight away.

So I have implemented a communication model (see 3.2) whereby an answer consists of a small set of points. Each point is generated according to agent's knowledge. When an agent is asked, first he uses his either GNG memory, or grid memory to estimate the Gaussian distribution of the food resource and then randomly generates the points using the estimated distribution.

3.5 Decision making

While searching for food, each type of the agent makes the decision where to go next. This process is either done randomly or following one's knowledge of the environment. A simple diagram of decision making follows.

The diagram 3.2 shows how an agents decides what to do. In fact it is common for all types of agents described in this thesis, although the first step "Put known food location into memory" is omitted in case of *random* and *PR agents*.

3.6 Memories

There are two types of memory which should allow agents to improve their lifespan compared to a random agent. Those are memories based on a growing neural gas and a spatial grid.

The *GNG memory* uses a self-teaching neural network which has been described in 2.1. The neural network allows the agent to learn the approximate

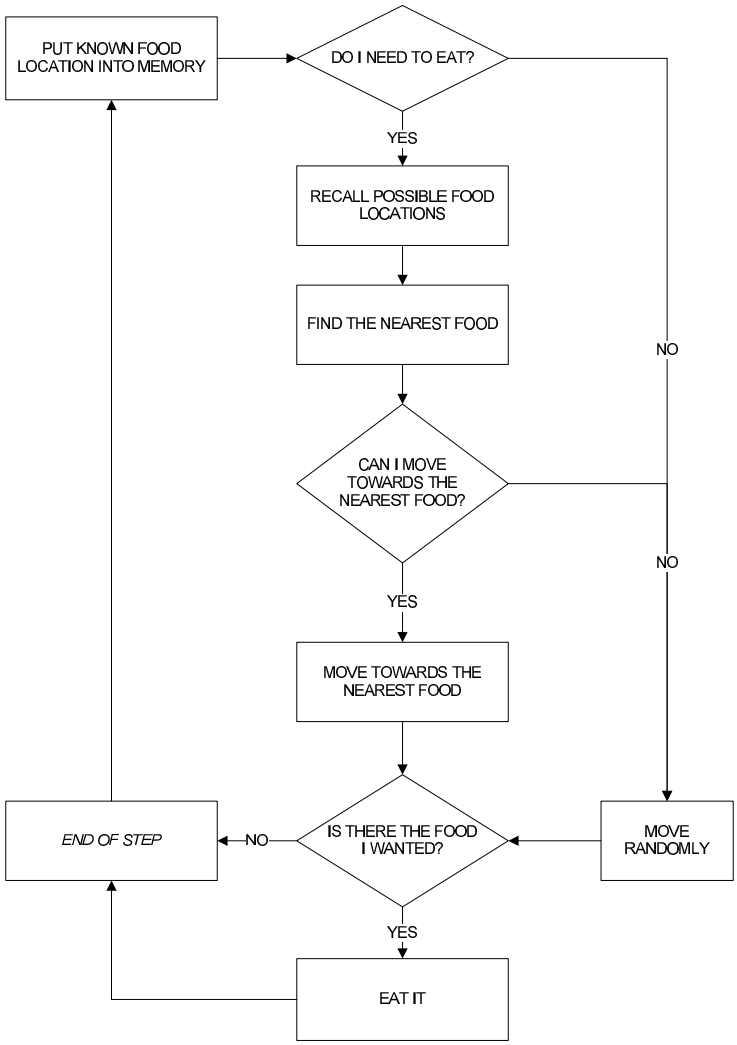


Figure 3.2: How the agent decides what to do next

location where the food is distributed. Each kind of food is given a single neural network which tries to learn the distribution reflecting the data inputs..

The *grid memory* divides the environment into a grid in order to simplify the space and to restrict the total size of the data structure used to describe the space.

Chapter 4

Implementation

4.1 Simulation application

Simulation was implemented as a Java application using JDK 1.6.

4.2 Memories

Each memory implementation implements following IMemory interface:

```
interface IMemory {
        Learn(integer foodKind, Point[] locations);
        Point[] GetSample(integer foodkind);
        (Point, integer) GetExpectedGauss(int foodkind);
}
```

Through the *Learn method* the memory gets new inputs and thus it learns. *GetSample method* returns a sample of food locations according to the agent's believes, it means that based on the information in memory it tries to generate several samples. *GetExpectedGauss method* returns expected Gaussian distribution (x,y location and variance) for the food kind given as a parameter.

The expected Gaussian distribution is not calculated every time the *GetExpectedGauss method* is called, but it precalculated after the new inputs are processed.

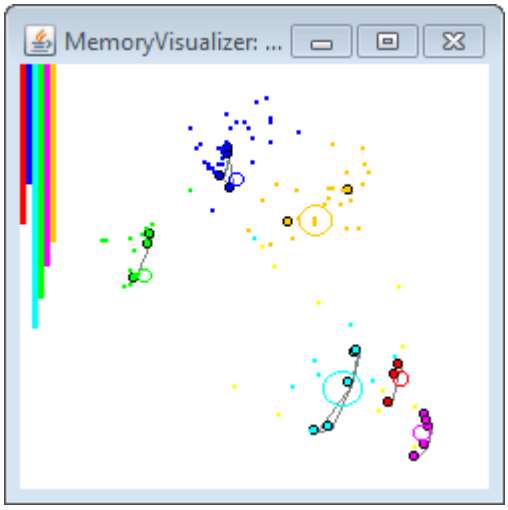


Figure 4.1: A visualization of an GNG agent's memory and the situation in the environment. Connected coloured circles represent growing neural gas for each food kind.

4.3 Growing neural gas as a memory

Growing neural gas has been explained previously in 2.1. In this section I will explain how that algorithm was used to learn positions of food sources. For this purpose the GNG uses five nodes.

For each kind of food there is a separate GNG engine which learns the believed location of such food.

```
procedure Learn(integer foodkind, Points[] locations)
  gngEngine <- GetEngineByFoodKind(foodkind)
  gngEngine.SetDescreteSignals(locations)
end
```

SetDescreteSignals method assigns known food locations (array of x,y points) to the neural network so as to be later processed using the GNG algorithm described in 2.1. At the end of day, i.e. end of each simulation step, the learnt information is processed by the GNG.

When the memory is asked about expected Gauss distribution, it uses the five nodes in the GNG to compute it.

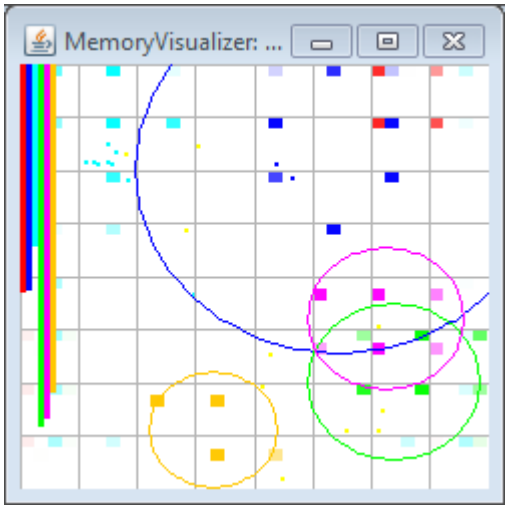


Figure 4.2: A visualization of an grid agent's memory and the situation in the environment. The coloured bars on the left side show current levels of agent's hunger. Coloured squares display grid cell values.

4.4 Grid as a memory

As in GNG memory implementation, the grid memory uses *Learn method* to update current state.

```
procedure Learn(integer foodkind, Points[] locations)
    grid <- GetGridByFoodKind(foodkind)
    hasInput <- CreateEmptyMatrix(cols, rows)

    for (x, y) in locations do
    begin
        (gridX, gridY) <- GetGridCoords(x, y)
        grid[gridX, gridY]++;
    end

    for (i, j) in grid do
    begin
        cell <- grid[i, j]
        if hasInput[i, j] > 0 then
          cell.IncPositive()
```

```
        else
          if node is in sight distance then
            cell.IncNegative()
    end
end
```

An actuall value of the cell is computed using following formula:

$$\begin{aligned} value &= positive - negative/\alpha \\ if(value < 0) value &= 0 \end{aligned} \tag{4.1}$$

α is set to 2.
The gaussian distribution is computed following way:

- x, y position is computed as a weighted mean of cells:

$$\frac{\sum CellValue_i \times (x_i, y_i)}{\sum CellValue_i} \tag{4.2}$$

- variance is computed as normalized sum of weighted distances:

$$\frac{\sum DistanceToCenter_i \times CellValue_i}{\alpha \times NumPositiveCells \times MaxCellValue} \tag{4.3}$$

Chapter 5

Experiments

5.1 Experimental settings and methodology

All following experiments are run using a default setup as it is described in this section. Each of the experiments is run on a quadcore *Intel Core i5 with 2,4 GHz and 6 GB RAM*.

Environment is set to be a square matrix with *64 x 64 dimension*. All agents start in the middle of the environment. There are *six kinds of food* which are randomly positioned in the environment and which generate a piece of food each *50 steps*.

Since an environment contains of six food kinds, an agent has six internal variables for each such food kind (see 5.1). By default they are set to 0 and are increased by *0.001 each step* in simulation. When they are equal to 1 (or higher), the agent dies.

5.2 Homogeneus agent set comparision with communication

In this experiment I will compare avarage life span and efficiency of groups which contains of agents with only single type of memory. Thereby you can see which of the used memory implementation works better in memory homogeneus environment.

What I assume is the *random agents* are about to expire almost immediately as they had no chance to find all the food. While the *PR agents* should approach their goals easily, thereby they will stay alive. Both results of *GNG agent* and

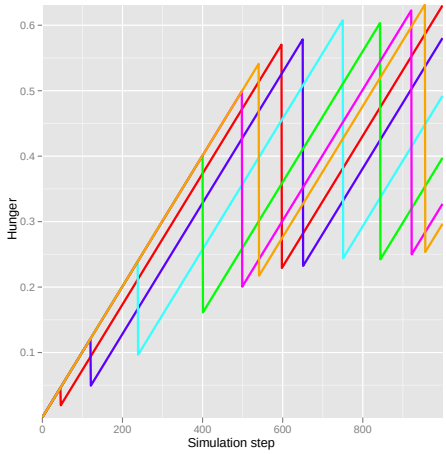


Figure 5.1: An example of agent's first 1000 steps showing the needs for each kind of food.

grid agent are matters of the experiment and I can only expect them not to be worse than *random agent* and not to be better than *PR agent*.

5.2.1 Random agent

As for the random agents I assumed that they will immediately pass away without communication. And as you can see in 5.2(b) it happened to be true.

On the other hand, if I allow them to communicate with each other it might happen they improve their chances. Although they will not last through the entire simulation (see 5.7(a)), the result is that they have managed to slow down [it] (see 5.3).

Agent kind	median	mean	min	max
Random agents with communication	1	0.939	0.55	1
Random agents without communication	1	1	1	1

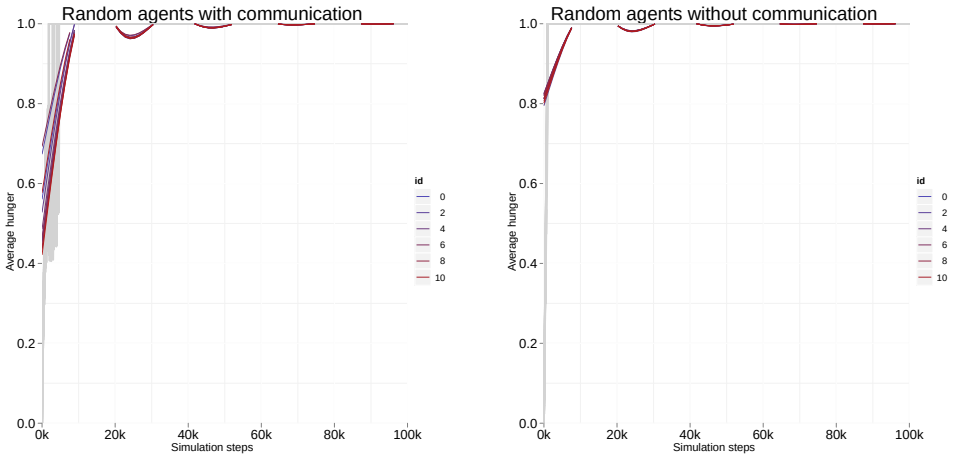


Figure 5.2: Random agent *without communication* fades out quickly.

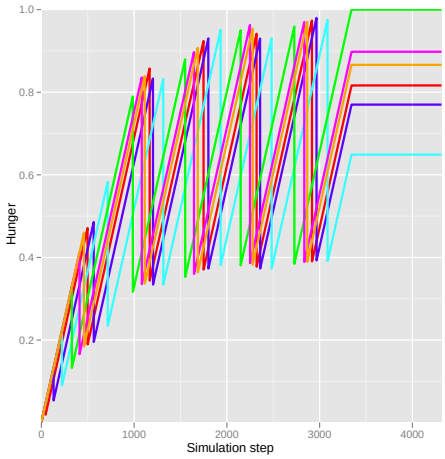


Figure 5.3: A beginning of life of a random agent with communication.

5.2.2 PR agent

In case of the *pure reactive agents* there is not much to compare between simulation with and without the communication, because the communication is not used since the PR agents see all the environment. So both graphs 5.4(a) and 5.4(b) are the same.

Agent kind	median	mean	min	max
PR agents with communication	0.69	0.689	0.54	0.86
PR agents without communication	0.69	0.689	0.54	0.86

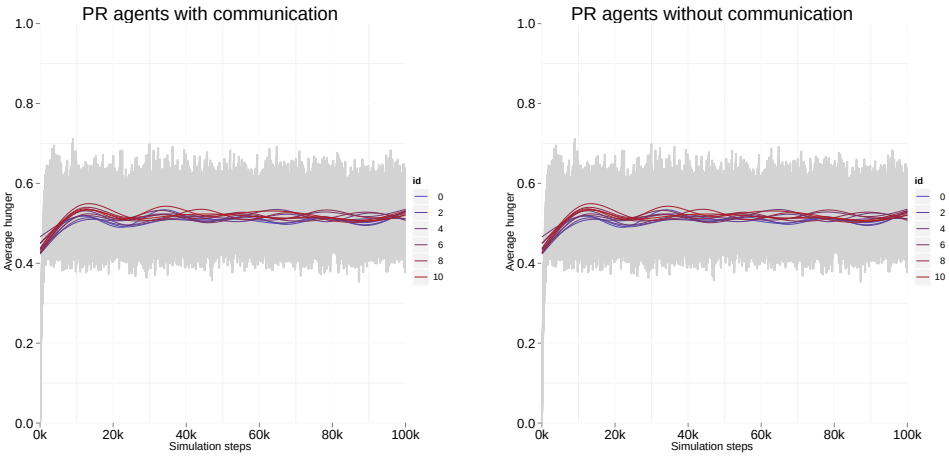


Figure 5.4: PR agent with and without communication.

5.2.3 GNG agent

GNG agents need more information to be able to learn and to survive that is why I do not expect them to deal with it well. In case of a simulation without communication they might end up similarly to random agents, they should do better if they communicate.

Agent kind	median	mean	min	max
GNG agents with communication	0.69	0.683	0.49	0.83
GNG agents without communication	1	1	1	1

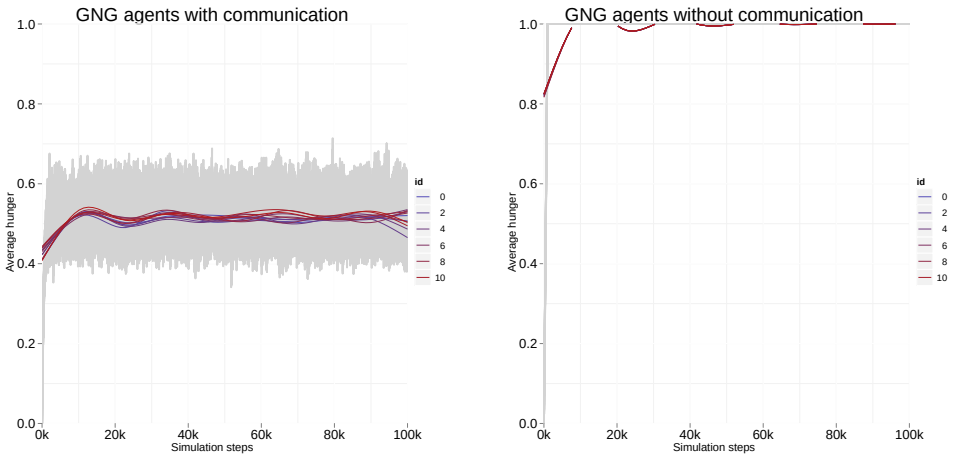


Figure 5.5: GNG agent with and without communication.

5.2.4 Grid agent

I assumed they will do simirarly like GNG agent, i.e. they fail to survive without communication, which assumption ended up to be wrong. Grid agents are able to learn quickly the environment just by moving around randomly and there is the chance that a couple of them survive.

I have run 20 simulations with grid agents to verify the probability of grid agent's learning the environment. As in the standard experiments there were 12 agents and the following values shows the result of the tests:

$$mean=5.56,\ median=5.5,\ min=1,\ max=12$$

Agent kind	median	mean	min	max
Grid agents with communication	0.81	0.826	0.54	1
Grid agents without communication	0.65	0.771	0.44	1

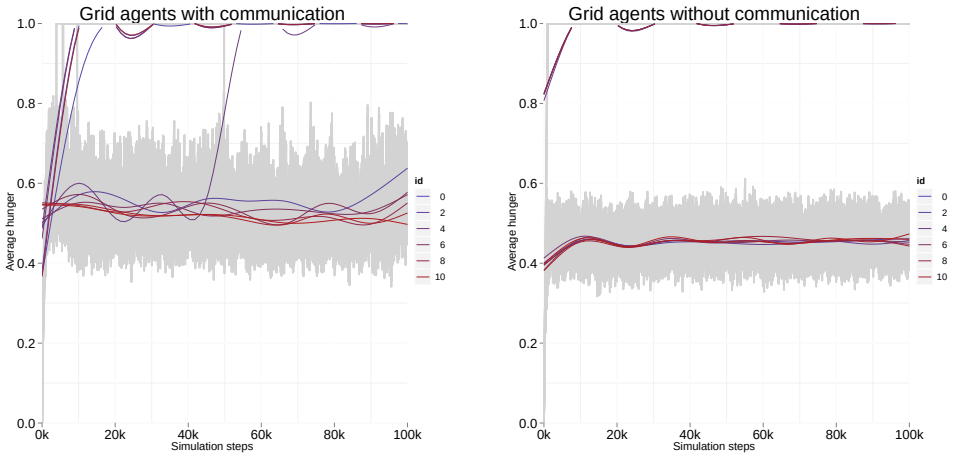


Figure 5.6: Grid agent with and without communication.

5.3 Mixed environment

Second part of the experiments are about environment where different kinds of agents are trying to fulfill their needs. I will observer whether some kind of agents prosper from a presence of other agents or if they are not that successful as they were in a homogeneous environment in previous experiments.

5.3.1 GNG+Grid+PR+Random agents

First I will start with combination of all kinds of agents, whereby since there are those PR agents the others have an advantage of the perfect source of information.

Results of a simulation with communication follow:

Agent kind	median	mean	min	max
GNG agents	0.7	0.699	0.57	0.86
Grid agents	0.7	0.697	0.54	0.83
PR agents	0.69	0.689	0.55	0.81
Random agents	0.7	0.704	0.57	0.87
All agents	0.7	0.698	0.54	0.87

Results of a simulation without communication follow:

Agent kind	median	mean	min	max
GNG agents	1	1	1	1
Grid agents	0.61	0.721	0.47	1
PR agents	0.58	0.576	0.44	0.68
Random agents	1	1	1	1
All agents	1	0.824	0.44	1

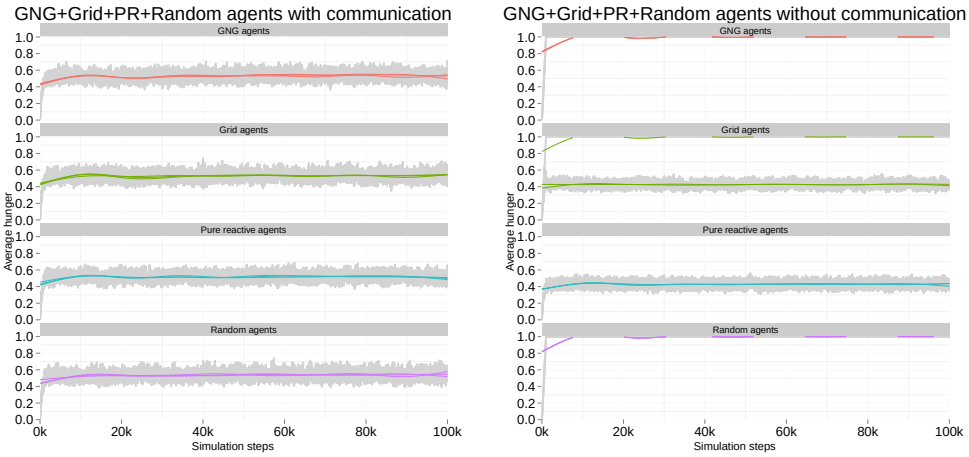


Figure 5.7: GNG, grid, PR and random agents together. Owing to PR agents and the communication they all do well.

5.3.2 GNG+Grid+Random agents

In this experiment I have omitted pure reactive agents and thus left others on their own. For better comparision I have added the difference between values in the current experiment and the previous one (GNG+Grid+PR+Random). I used red colour for a decrease and green colour for an increase, although it is an improvement if they values are lower.

Agent kind	median	mean	min	max
GNG agents	0.69	0.691	0.52	0.86
	-0.01	-0.008	-0.05	
Grid agents	0.7	0.700	0.55	0.87
		+0.003	+0.01	+0.04
Random agents	0.7	0.702	0.55	0.85
		-0.002	-0.02	-0.02
All agents	0.7	0.698	0.52	0.87
			-0.02	

Agent kind	median	mean	min	max
GNG agents	1	1	1	1
Grid agents	1	0.747	0.39	1
	+0.39	0.026	-0.08	
Random agents	1	1	1	1
All agents	1	0.916	0.39	1
		+0.092	-0.05	

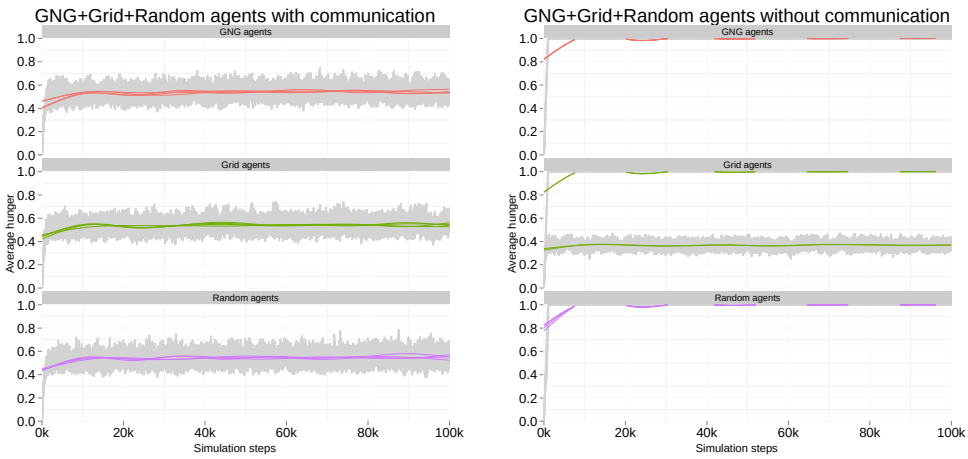


Figure 5.8: GNG, grid, PR agents together with and without the communication.

5.3.3 GNG+Grid agents

In this experiment I have observed a simulation with just memory agents in it. Again you can see highlighted differences between new values from this experiment and the values from the previous one (GNG+Grid+Random).

Agent kind	median	mean	min	max
GNG agents	0.72	0.715	0.5	0.89
	+0.03	+0.024	-0.02	+0.03
Grid agents	0.72	0.721	0.57	0.89
	+0.02	+0.021	+0.02	+0.02
All agents	0.72	0.718	0.5	0.89
	+0.02	+0.020	-0.02	+0.02

Agent kind	median	mean	min	max
GNG agents	1	1	1	1
Grid agents	1	0.755	0.4	1
		+0.008	+0.01	
All agents	1	0.877	0.4	1
		-0.039	0.01	

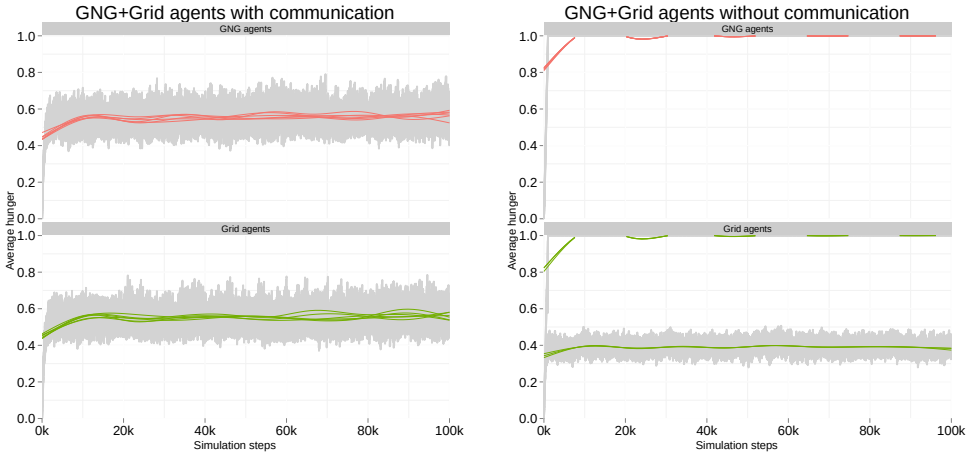


Figure 5.9: GNG and grid agents together.

Chapter 6

Discussion

In previous chapter you have seen experimental runs of the simulation where I have compared the level of overall hunger of agents throughout the simulation's steps. Now I am going to sum up those results and discuss possible outcomes.

Simulation settings (size of the environment, number and distribution of resources) allowed random agents to survive few steps when they could communicate with each others. Thereby the communication was strong tool for agents how to improve their results.

The communication was not always an improvement as you can see in 5.6 where the grid agents were less efficient when they were allowed to share information about the environment. The communication brings more income information and thus increases the ration between positive and negative variables (see 2.2) and disorients agents. Owing to communicatio it is hard to decide which of gng and grid agent was better.

As I have already clarified the grid agent has perfect results without using communication as soon as he survives first thousand steps. On the other hand, he obviously fails to use communication as an improving factor. That is something where the gng agents win.

Second part of experiments is about comparing simulation with more kinds of agents at once. Thereby I could observe how they compete against each others. So I have observed GNG+Grid+PR+Random, GNG+Grid+Random and GNG+Grid. Apart experiments with communication, I have also run simulation whereby the agents were not able to communication, although in such conditions the result could not be different from single-kind setups.

Since there are pure reactive agents in the first experiment (GNG+Grid+PR+Random) the agents have a perfect source of information and thus easily succeeded. Hav-

ing omitted PR agents I could have observed that gng agents have bettered and, on the other hand, the grid agent become worse. Furthermore, when you look at results of GNG+Grid simulation where the random agents are missing, you can see that both grid and gng agents become worse, although the random agents should be much helpfull.

What is different between random agents and memory agents is the random agent answers only correct positions, when he is asked, and the memory agent usually answers using his believes.

There were minor problems I had to deal with, for example I had to setup variables for the GNG algorithm. What I have learnt from that is the algorithm is expected to slowly converge to the learnt topology. On the other hand, I have used it for unncertain dynamicly changing data which were mostly based on other agents' believes. That is probably why much simpler memory structure used for grid agents has better results.

Conclusions

I have created a multi-agent simulation with four different kinds of agents each of which differs in their approach to fulfill their needs. What they had to succeed in was they were put inside a two dimensional environment whereby they had to learn positions of six food resource so as to be able to survive. They were pure reactive agent, random agent, GNG agent and grid agent. Latter two had a memory to learn those positions. GNG agent used implemantion of growing neural gas, an unsupervised neural network, and grid agent used data structure inspired by [6].

In the experiments I have compared those agents in their efficiency. First I have set up environments in which was only a single kind of agent and after that I have combined different kinds of agents together. The results of their efficiency measured by the overall hunger were presented in graphs and statistical variables such as mean, median, maximal and minimal value.

Bibliography

[1] PhD thesis.

[2] R. C. Atkinson and R. M. Shiffrin. Human memory: A proposed system and its control processes. *In K. W. Spence and J. T. Spence (Eds.), The Psychology of learning and motivation: Advances in research and theory (vol. 2).*, pages 89 – 105, 1968.

[3] Richard C Atkinson and Richard M Shiffrin. The control of short-term memory. *Scientific American*, 225(2), 1971.

[4] Duncan Black. On the Rationale of Group Decision-making. *Journal of Political Economy*, 56(1):23–34, 1948.

[5] Michael E. Bratman, David J. Israel, and Martha E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3):349–355, 1988.

[6] Cyril Brom, Tomáš Korenko, and Jiří Lukavský. How do place and objects combine? "what-where" memory for human-like agents. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, IVA '09, pages 42–48, Berlin, Heidelberg, 2009. Springer-Verlag.

[7] Bernd Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

[8] Wan Ching Ho, Kerstin Dautenhahn, and Chrystopher L. Nehaniv. Computational memory architectures for autobiographic agents interacting in a complex virtual environment: a working model. *Connect. Sci*, 20:21–65, March 2008.

[9] T. M. Martinetz and K. J. Shulten. A "neural-gas" network learns topologies". In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397–402. North-Holland, Amsterdam, 1991.

[10] Thomas Martinetz. Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps. In Stan Gielen and Bert Kappen, editors, *Proc. ICANN'93, Int. Conf. on Artificial Neural Networks*, pages 427–434, London, UK, 1993. Springer.

[11] Judea Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine*, pages 329–334, August 1985.

[12] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition)*. Prentice Hall, 3 edition, December 2009.

[13] Robert J. Sternberg. *Cognitive Psychology*. Wadsworth Publishing, August 2002.

[14] A. Tapus, G. Ramel, L. Dobler, and R. Siegwart. Topology learning and recognition using bayesian programming for mobile robot navigation. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3139 – 3144 vol.4, sept.-2 oct. 2004.

[15] Hal R. Varian. *Intermediate microeconomics : a modern approach / Hal R. Varian*. W.W. Norton, New York :, 1987.