Swarm robotics simulator

Bedřich Said, bsaid@seznam.cz

May 21, 2011

Annotation

Today's fields of artifical intelligence, notably neural networks and genetic programming, derive their results based on random permutations of elementary skills. Often they try to mimic the behavior of biological systems. However, we currently lack tools which would, in addition to yielding correct solutions, also define the method that solves a given problem.

This paper describes an algorithm that can, based merely on the success it had solving the problem so far, learn to solve the problem correctly and in addition provide a procedure to solve the problem in the future. The derived procedure can be stored and later applied on more instances of the same problem. For developing the algorithm I made a simulator, which is capable of performing simulations on an arbitrary number of robots placed in a virtual world. The simulator is a multi-platform client-server application. Users can control all simulations via the internet and save the results in text or in 3D animation format.

The simulator simulated a population of robots, which did not know, how they can get required energy. The artificial intelligence algorithm successfully solved this problem in three minutes.

The virtual world helps us to develope new and better programs like this new artificial intelligence. We can idealize conditions in the virtual world and see some unreal situations. The unreal simulations helps us to discover new principles in the real world.

Key words: Swarm intelligence, artificial intelligence, simulator, robot, virtual world, software, client, server, 3D animation

Contents

List of Figures

1 Introduction

Nowadays we know some algorithms like Artificial Neural Networks (ANN), genetic programming, data mining etc. These systems use random processes for solving problems. This paper describes a new artificial intelligence algorithm that do not use any random processes and it can make new program as its output. The new created program can solve the current problem all the time. For developing this new intelligence I made a simulator Yunimin that simulates many robots in virtual world.

The intelligence algorithm runs on the virtual robots on the simulator Yunimin. We can see all simulations in real-time and the simulated virtual robots can communicate with real robots via serial (for example bluetooth) connection. We can compare user written algorithms with artificial intelligence, because the simulator can simulate all programs for robots in language C/C++. The virtual world helps us to develope new and better programs, we can idealize conditions in the virtual world and see some unreal situations. The unreal simulations helps us to discover new principles in the real world.

Swarm robotics algorithm: The goal of this intelligence is Database of experiences where it has all its known functions. The algorithm makes elementary combinations of its experiences and gets a fitness value as an information about its success. User can edit the experiences in the database and he can prepare the intelligence to solve the current problem. The algorithm needs axiomatic experiences to learn new experience as a combination of the axioms. When the algorithm will solve the next problem, it will use new learned experiences as its axioms and it can solve more difficult problems. User can copy the Database of experiences and use it for other version of the intelligence algorithm or he can merge more databases to make one big database that can solve all learned problems.

The algorithm can colaborate with other intelligence algorithms, because the user can add other intelligence algorithms as one experience or communicate with other programs in real-time.

Simulator Yunimin: The simulator is designed for simulating robots moving on the ground in the virtual world. The programs for robots are in C/C++ programming language. The simulator can run on Windows or Linux based operating systems. The server simulates the virtual world with robots. Users can view and control whole simulation via the Internet. Users can pre-set some conditions of the virtual world and simulate real or unreal processes. Users can add new robots to the virtual world during the simulation. The simulator is able to simulate big populations of robots, we can see the behaviour of populations and discover new principles. Users can save the output data in text (CSV) format or render it as 3D animation with our 3D models of robots and virtual world. During the simulation we can see the results as real-time 2D animation or in plain-text format.

The simulator was successfully simulated several scenes of collective cooperation of robots. For example, a simulation of ten robot, whose task was to survive as long as possible. The question was, how can they keep their decreasing energy. The artificial intelligence solved this problem successfully on the simulator.

2 The artificial intelligence algorithm

2.1 Motivation for the basic algorithm of the artificial intelligence

Imagine a game where we get a number (numbers) and our task is to answer a next number (numbers). At the beginning of the game we do not know any number to answer, so we send a random number. We get an information about our success and we get another number. We have to respond again. We repeat the cycle until we find an algorithm which computes the correct numbers. Since the beginning of the game we know only one information, we can compute the answer only from the input numbers.

Suppose now that we have an algorithm that can play this game. Any input (from sensory perception to non-numeric data) can be converted to a list of numbers that we can put to the input of this algorithm. Similarly, we can convert the numerical output to any data format.

For the functionality of the algorithm is sufficient that we will put the input data and success of the algorithm in each step. For example we can represent the success in percent.

We are looking for an algorithm that cyclically receives the input as list of numbers and the current success and after each cycle it sends the output as list of numbers. The scheme of algorithm that can solve this problem is shown in figure 1. The program takes input data and success. The main algorithm calls elementary and then complex combinations of experiences from Database of experiences. Database of experiences is a directed acyclic graph, where the vertexes points to functions and edges means the conectivity between elementary and complex functions. The entire graph is based on the axiomatic experiences and then the algorithm adds new vertexes as new learned experiences. The called combination of experiences from the database has access to:

1. the temporary memory,
2. the global memory for reading general informations (Structure of the informations)
3. the history memory
4. or it can establish communication with other network devices - like sensors.

When the algorithm discovers the correct combination of experiences, this combination is added to the Database of experiences as new experience. The algorithm has fulfilled its task and it can respond correctly to a specified input or it can be terminated. The intelligence algorithm learned how to solve similar problems related with the solution of this problem.

2.2 Structure and description

All parts of the artificial intelligence are described in figure 1.

2.2.1 Input and output

Input and output is a stream of numbers or strings. The algorithm gets input with an information about its success and it computes some outputs. Then it gets informations about success of these outputs with new input. The new input data can be the same as the first input data.

2.2.2 Fitness function

Fitness function gets to the artificial intelligence an information about its success. This function will make the success information from the output data of the intelligence or compute it from other external data. User can use the intelligence algorithm to try to compute the correct success.

2.2.3 Structure of the informations

Structure of the informations is a directed acyclic graph. When the intellignce solves a new problem the Structure of the informations is reseted. Vertices contains data which are processing by the intelligence. We have input vertices and then the algorithm will create some added vertexes. The input vertexes are created from the input data. Each added vertex will appear as an output of an actual called combination of the experiences. Some added vertexes are labeled as the output data.

2.2.4 Experience

Experience is a function which work with Structure of the informations and its own memory. The experience gets some vertices from Structure of the informations as its input. The experience computes output and use its own memory and then sends output as making new vertices in the Structure of the informations. The experience makes directed edges between each input and output vertex. The value of each edge is an id of experience which computes the output.

2.2.5 Database of experiences

The database of experiences contains axiomatic experiences, combinations of the axiomatic experiences and sometimes it contains user added experiences. The axiomatic experiences are basic functions that can operate all common basic operations needs for computing the given problem. Combinations of the axiomatic experiences are new experiences that formes automatically when the algorithm successfully solves any problem. The artificial intelligence algorithm learnes itself. User can add new experiences manually when he wants to help intelligence to solve the given problem. User adds new experience as a function in the programming language. He puts his function to array of user experiences. This function gets input and output in Structure of the informations format.

2.2.6 History, global memory, sleeping

History is a memory where are saved informations about history of work. There are saved informations about called functions and processed data. The experiences and other functions can read this memory. Global memory is a memory with global informations about constants, settings etc. The experiences can only read the global memory, because they can fill up the global memory in a short time. Sleeping is a process after finishing each task which deletes temp memory and Structure of the informations. It can clean and sort other memories and prepare the artificial intelligence for next tasks.

History, global memory and sleeping function are not very important for running.

2.2.7 Description of the artificial intelligence algorithm

In the begining the artificial intelligence algorithm has the Database of axiomatic experiences and learned experiences from previous tasks. Structure of the informations is empty graph and other memories are empty, too. The algorithm gets the first input. It saves the input to the Structure of the informations and then it calls all possible experiences and their basic combinations. When it finds some possible results it will save new experience as the combination which found these results. User can update the input data cyclically. When the intelligence find the correct results it will create a program which solves the given problem all the time.

2.3 Practice

I simulated ten robots in two groups on the simulaotr Yunimin. The task was to still alive as long as possible. The robots have to get an energy to still alive. The value of their energy was a success value for the fitness function. Input were all distances from other robots and each robot drove its two wheels as output.

The robots were in a playground. Each group had a half of the playground. If the robot was in his half of the playground, his energy was lower with time. If the robot was in opposite half, his energy was upper with time. When the distance between two robots was very small, the robots can fight or collaborate. If the robots are from the same group, the robots with more energy put some energy to the second robot. The robots from the same group helped themselves. If the robots is not from the same group, stronger robot got some energy from the robot with less energy.

When the simulation started, the robots went on the ground "randomly". After a few seconds two robots went to opposite half of the playground. They remained on the border and they waited for more energy. After one minute some robots went to the opposite half and they waited for more energy for long time. After three minutes all robots changed their sides and waited for more energy from opposite side of the playground.

The whole simulation was saved in CSV format and it was rendered as 3D animation in AVI format.

3 Simulator Yunimin

Simulator Yunimin is a multi-platform client-server application that simulates behavior of mobile robots in 2D virtual world. The application is primary intended to help debug programs for robots. We cannot idealize the real world and pre-set the conditions for robots. The simulator can indealize the vitual world to set ideal conditions for debugging programs and simulating some unreal situations. It will help us in developing new projects with robots.

The simulator Yunimin simulates lots of robots as client applications. The Server accepts new client connections and simulates the virtual world. Clients asks sever for some questions like data on the sensors and communication. The robots in the virtual world can communicate with real robots in real-time. The simulator use the TCP/IP communication.

The third part of the simulator is viewer. Viewer is a user based application that controls all simulations via the Internet. The server accepts unlimited connections of clients or viewers. One simulation can be controlled by namy users. Each user can add his own robots to the simulation and see the behaviour of all robots in the scene. The simulator gets us the output of the simulation as text/table data or we can render the 3D animation of whole simulation. We can use our own models for rendering the scene. If we want to see the simulation in real time, we can see real-time 2D animations or plain text data.

3.1 Motivation and benefits

When we are developing new mobile robots or programs for them, we need to test and debug their behaviour. Developing is easier when we have a tool which can idealize conditions in the real world.

Simulator Yunimin simulates many robots in the virtual world. We can pre-set conditions of the virtual world. For example, we can set zero weight of robot or absolute accurary of the sensors. We can control all simulations via the internet, the server accepts connections from clients (simulating robots) and viewers (users). The simulating virtual robots can communicate with the real robots in the real world, because the server can simulate real-time communication among real and virtual robots.

3.2 Characteristics of the simulator

The whole application consists of a server, clients and viewers. Server simulates the virtual world like its physical conditions, position of robots, communication etc. Simulated virtual robots (clients) are connecting to the server and give him some questions about the virtual world and their position in this world. Clients gets answers like values of robot's sensors which are computed by server. Each connected client simulates one program for one robot. Users can view and control the simulation via the Viewer. Viewer is a user interface that connects to the server. Viewer sends to server all user equirements and gets back all informations about all simulated robots. User can save the whole simulation in CSV (text) format or render it to 3D animation.

The simulator is designed for robots moving on the ground by wheels. The configuration of next equipment is arbitrary. Accurary of the simulation is the same as accurary of input data. The simulator supports parallel computing.

3.3 Usage

We can use the simulator in our own or we can connect to some server with this simulator. The simulator can help us with debugging programs and deleting software errors. When your program do not run correctly on the simulator, we know, that the error is not in hardware. The simulator can simulate lots of robots, so we can simulate behaviour of big robotics population. We can simulate unreal processes, because we can idealize the conditions in the virtual world. We can watch detailed behaviour of robots, because we get data from the simulator in CSV (text) format. We can edit and process the output data by other programs which are compatible with CSV data format. We can render a 3D animation of the simulation with our 3D models of robots and 3D model of the virtual world. The virtual robots can communicate among them and with the real robots in the real world in real-time.

3.4 Technical characteristics

The simulator is a client-server application. The server simulates the virtual world, clients are connected robots that communicates with server and ask it for the input like sesor data, position etc. The clients (simulated robots) send to server informations about their behaviour. The communication is in text format via TCP/IP protocol. The detailed communication is desribed in documentation for simulator Yunimin.

User can view and control the simulation via the viewer. Viewer is a user interface that receives informations about simulation from the server and display it like 2D animation for user. User can save whole simulation in CSV (text) data format or render it to 3D animation in AVI format. Many users can connect via many viewers to one server and independently manage the simulation and their simulating robots. Viewer can add (or delete) new robots to running simulation on the server. The detailed user manual and communication description is described in documentation for simulator Yunimin.

The server accepts all requests from clients and viewers. The server supports TCP/IP protocol to communication. It supports parallel computing and it can run on the multi-processor computers. The server computes the simulation discreetly, because the robot's CPU changing the input data all the time. The server can run in real-time mode and the simulated robots can communicate with real robots, users and other computers in real-time, too. If the user switches the server to nonreal-time mode (on slow computers), clients are waiting for the server and compute next data after accepts the answer from the server. The detailed description and technical informations are written in the documentation for simulator Yunimin.

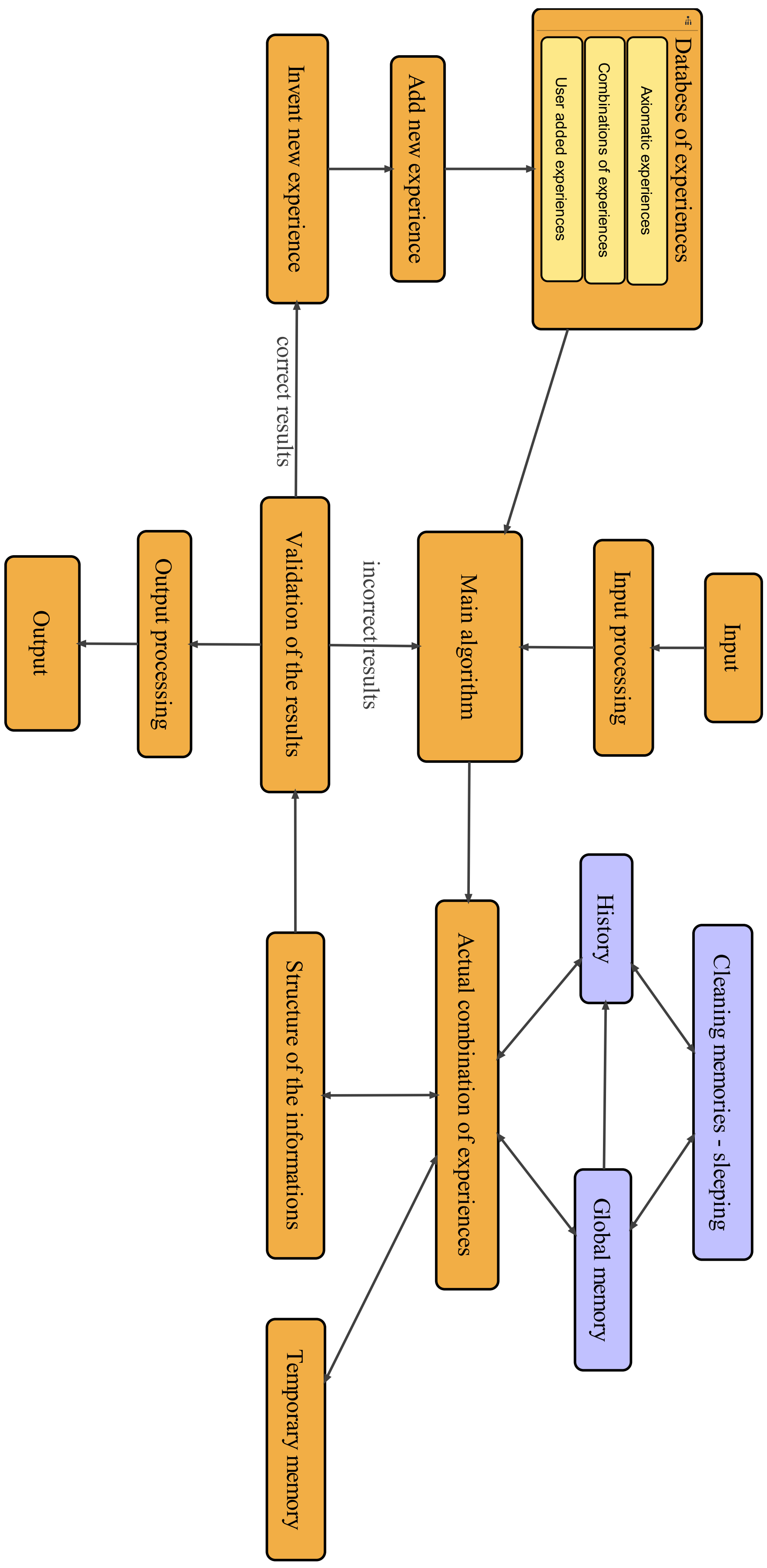


Figure 1: Schema of the artificial intelligence algorithm

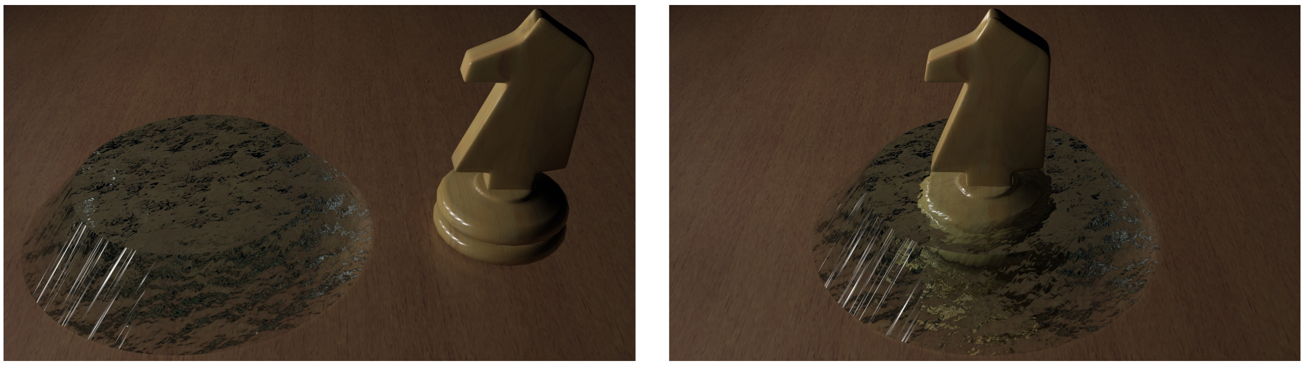


Figure 2: Rendered animation of the artificial intelligence algorithm made by simulator Yunimin

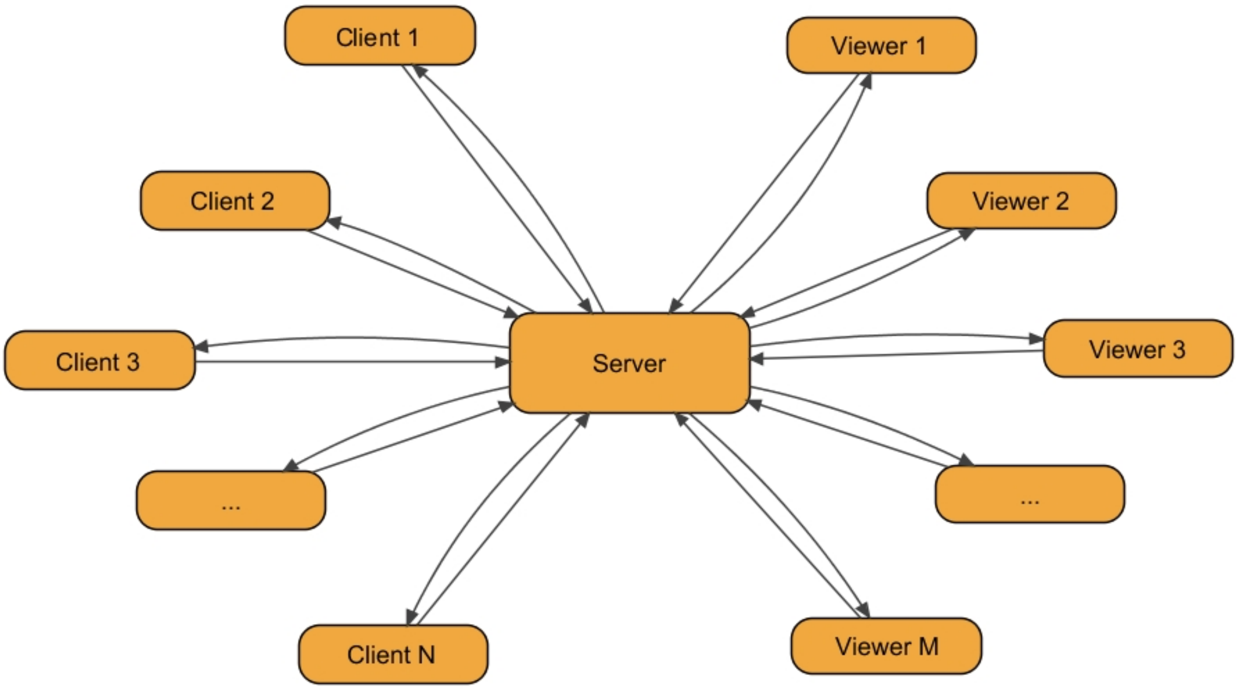


Figure 3: Architecture of the simulator Yunimin

4 Conclusion

The artificial intelligence algorithm was successfully simulated on the simulator Yunimin. Many robots were simulated in one scene, these robots did not know their task. They knew only their actual success value and they tried to solve their problem only with this information. Another task for robots in the virtual world was to still alive as long as possible. The robots did not know how they could give some energy. The artificial intelligence algorithm successfully solved this problem.

In the future we can enlarge the Database of experiences and find all axiomatic experiences. The intelligence can colaborate with other artificial systems like Artificial Neural Networks (ANN), genetic programming etc.

For developing the artificial intelligence and other problems I developed simulator Yunimin. The simulator Yunimin is a multi-platform client-server application that can simulates all programs for robots in language C/C++. Many robots are simulated in the virtual world on the server. All simulations are controlled by user via the Internet.

The simulator correctly simulated real scenes with real robots. The accuracy of the simulator is the same as accuracy of the input data. We can idealize some conditions in the virtual world and we can simulate some unreal processes that will us to discover new principles.

5 The bibliography

References

[1] CorMac Technologies Inc., NeuNet Pro [online]. 2011 [cit. 2011-05-10]. NeuNet Pro 2.3 for Windows. WWW: http://www.cormactech.com/neunet/

[2] Lawrence, Jeanette (1994) Introduction to Neural Networks, California Scientific Software Press. ISBN 1-883157-00-5

[3] Wasserman, Philip (1993) Advanced Methods in Neural Computing, Van Nostrand Reinhold, ISBN 0-442-00461-3

[4] Masters, Timothy (1994) Signal and Image Processing with Neural Networks, John Wiley & Sons, Inc. ISBN 0-471-04963-8

[5] POLI, R.; LANGDON, W. B.; MCPHEE , N. F. A Field Guide to Genetic Programming . Web : Lulu.com, 2008. 252 s. WWW: http://www.lulu.com/browse/search.php?fSearch=genetic+programming. ISBN 9-781409200-73-4

[6] RML Technologies [online]. 2010 [cit. 2011-05-10]. Discipulus^{TM} 5 Genetic Programming Predictive Modelling. WWW: http://www.rmltech.com/.

[7] Banzhaf, W., Nordin, P., Keller, R.E., and Francone, F.D. (1998), Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications, Morgan Kaufmann

[8] Brameier, M. and Banzhaf, W. (2007), Linear Genetic Programming, Springer, New York

[9] Weise, T, Global Optimization Algorithms: Theory and Application, 2008

[10] SourceForge.net [online]. 2011-05-01 [cit. 2011-05-10]. Simbad 3D robot simulator. WWW: http://simbad.sourceforge.net/

[11] eDownload.cz [online]. 2009-06-16 [cit. 2011-05-10]. Robot simulator. WWW: http://www.edownload.cz/sw/robot-simulator/

[12] Manifestation.com [online]. 2006 [cit. 2011-05-10]. Eliza, computer therapist. WWW: http://www.manifestation.com/neurotoys/eliza.php3

[13] SAID, Bedrich. Bitbucket.org [online]. 2011 [cit. 2011-05-10]. Repository. WWW: https://bitbucket.org/bsaid

A Results of the intelligence algorithm in developing

Figures 4, 5 and 6 shows behaviour of the artificial intelligence in developing. All graphs shows record of one problem. The intelligence gets a number as input. It must find correct arithmetic expression to compute an output number. All numbers were only integers and the axiomatic experiences were operations plus, minus, times and division.

X axis is a time in steps, Y axis represents integers. Blue line is a success value of the intelligence in each step. Red line represents input number and yellow line represents the output number.

The graphs:

1. The first graph shows intelligence when it was only able to try all axiomatic experiences.
2. The intelligence priority called the experiences that were successful in previous tasks.
3. The intelligence can make a new experience as a combination of axiomatic experiences.
4. The same task as in graph three. The intelligence remembred new learned experience.
5. The intelligence gets new input when it has high success value.
6. The same task as in graph five, the intelligence remembred it.

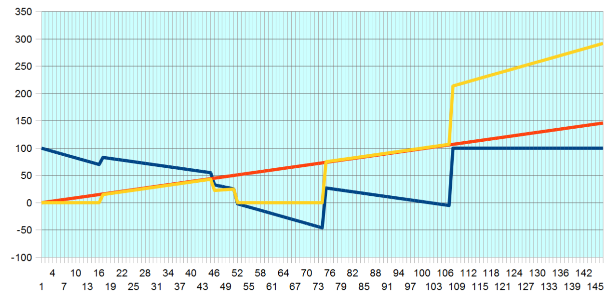
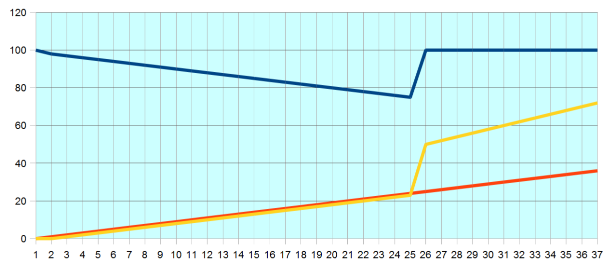



Figure 4: The first and the second graph of the artificial intelligence results

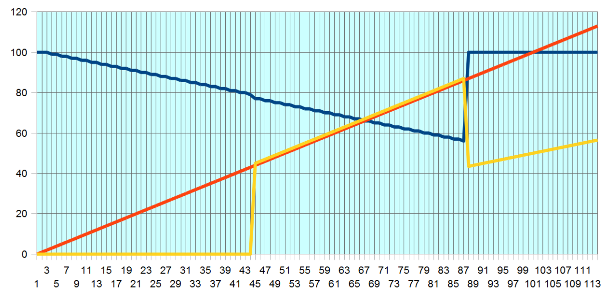
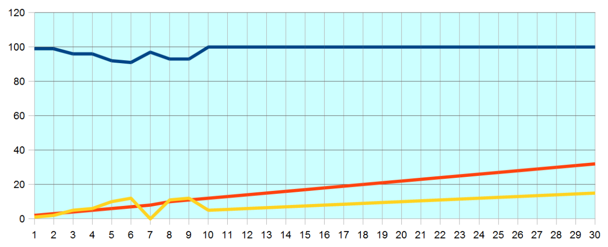



Figure 5: The third and the fourth graph of the artificial intelligence results

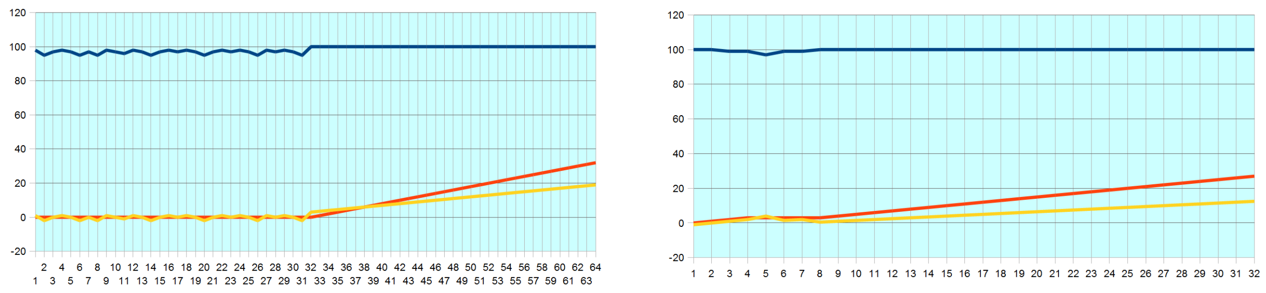


Figure 6: The fifth and the sixth graph of the artificial intelligence results

🟥 Red - the input data

🟨 Yellow - the output data

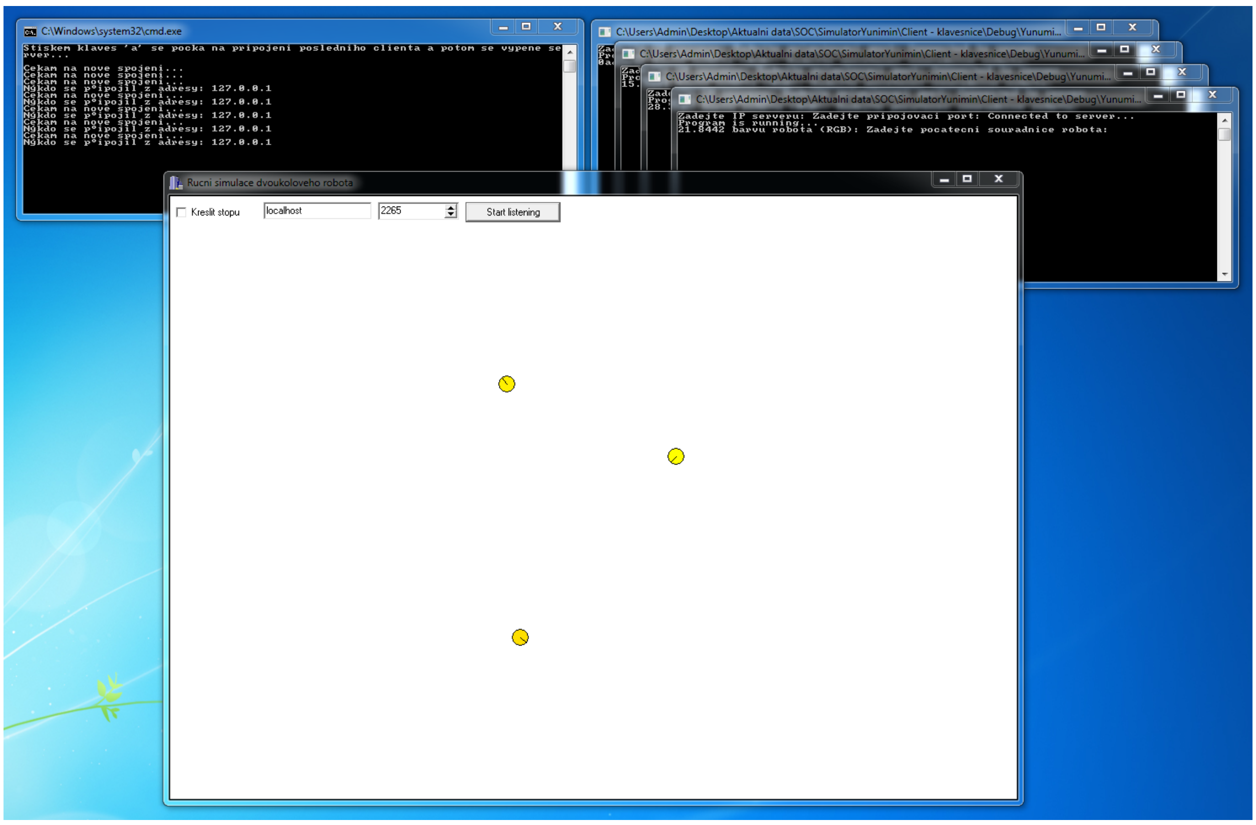
🟦 Blue - the success of the algorithm

Figure 7: Legend for the graphs of the artificial intelligence results

B Simulator Yunimin

B.1 Screenshots

Figures 9 and 8 shows screenshots of the running simulator Yunimin on the server and then on the local computer as localhost.


C:\Windows\system32\cmd.exe

```
Stiskem klaves 'a' se pocka na pripojeni posledniho clienta a potom se vypene se
rver...

Cekam na nove spojeni...
Cekam na nove spojeni...
Cekam na nove spojeni...
Nýkdo se p°ipojil z adresy: 127.0.0.1
Cekam na nove spojeni...
Nýkdo se p°ipojil z adresy: 127.0.0.1
Cekam na nove spojeni...
Nýkdo se p°ipojil z adresy: 127.0.0.1
Cekam na nove spojeni...
Nýkdo se p°ipojil z adresy: 127.0.0.1
Cekam na nove spojeni...
Nýkdo se p°ipojil z adresy: 127.0.0.1
```

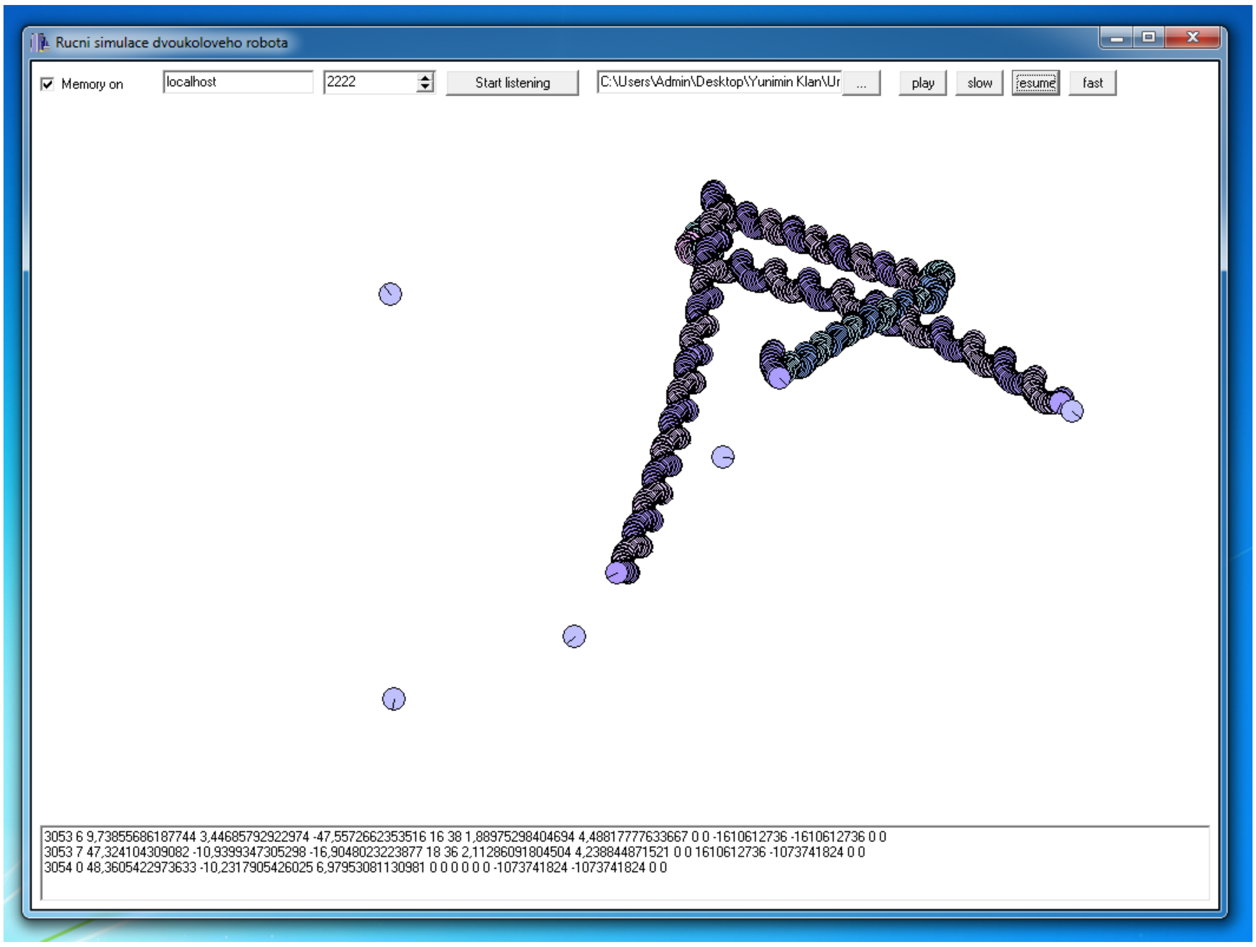
C:\Users\Admin\Desktop\Aktualni data\SOC\SimulatorYunimin\Client - klavesnice\Debug\Yunumi...

```
Zadejte IP serveru: Zadejte pripojovaci port: Connected to server...
Program is running...
21.8442 barvu robota (RGB): Zadejte pocatecni souradnice robota:
```

Rucni simulace dvoukoloveho robota

☐ Kreslit stopu | localhost | 2265 | Start listening

Figure 8: Screenshot of the running simulator on one PC


Rucni simulace dvoukoloveho robota
Memory on
localhost
2222
Start listening
C:\Users\Admin\Desktop\Yunimin Klan\Ur
...
play
slow
esume
fast

```
3053 6 9,73855686187744 3,44685792922974 -47,5572662353516 16 38 1,88975298404694 4,48817777633667 0 0 -1610612736 -1610612736 0 0
3053 7 47,324104309082 -10,9399347305298 -16,9048023223877 18 36 2,11286091804504 4,238844871521 0 0 1610612736 -1073741824 0 0
3054 0 48,3605422973633 -10,2317905426025 6,97953081130981 0 0 0 0 0 0 -1073741824 -1073741824 0 0
```

Figure 9: Screenshot of the real-time simulation of many robots

B.2 Photographs

Figure 10 shows two photos from the competition on the simulator Yunimin. The contestants had to write program for robot which will win the match. The robot wins when it catches labeled robot as the first. The contestants competed with other people and with the artificial intelligence algorithm.

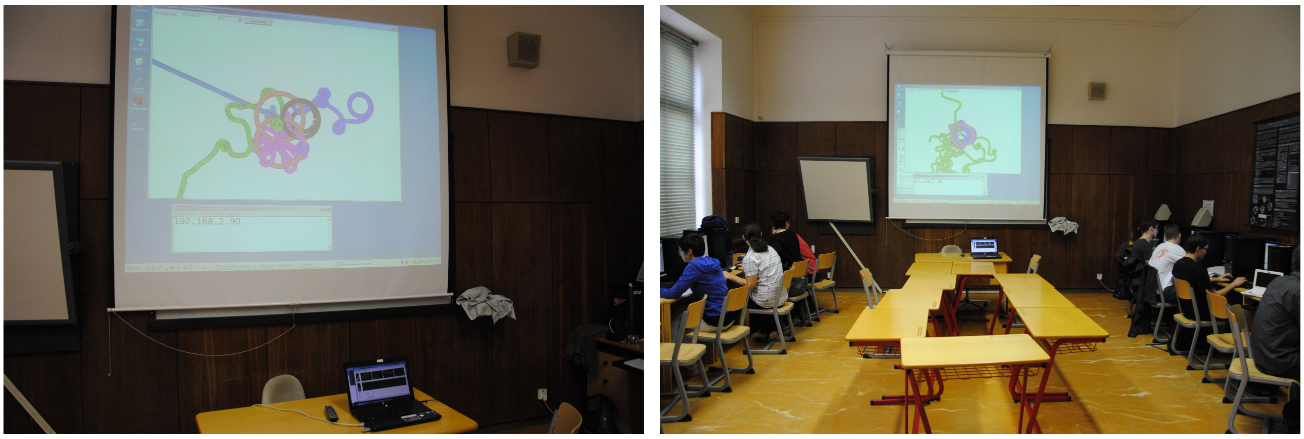


Figure 10: Photos from the competition on the simulator Yunimin

Figure 11 shows the first simulated robot on the simulator Yunimin. In the photo is real photo of the robot and 3D model that was used for rendering 3D animations.

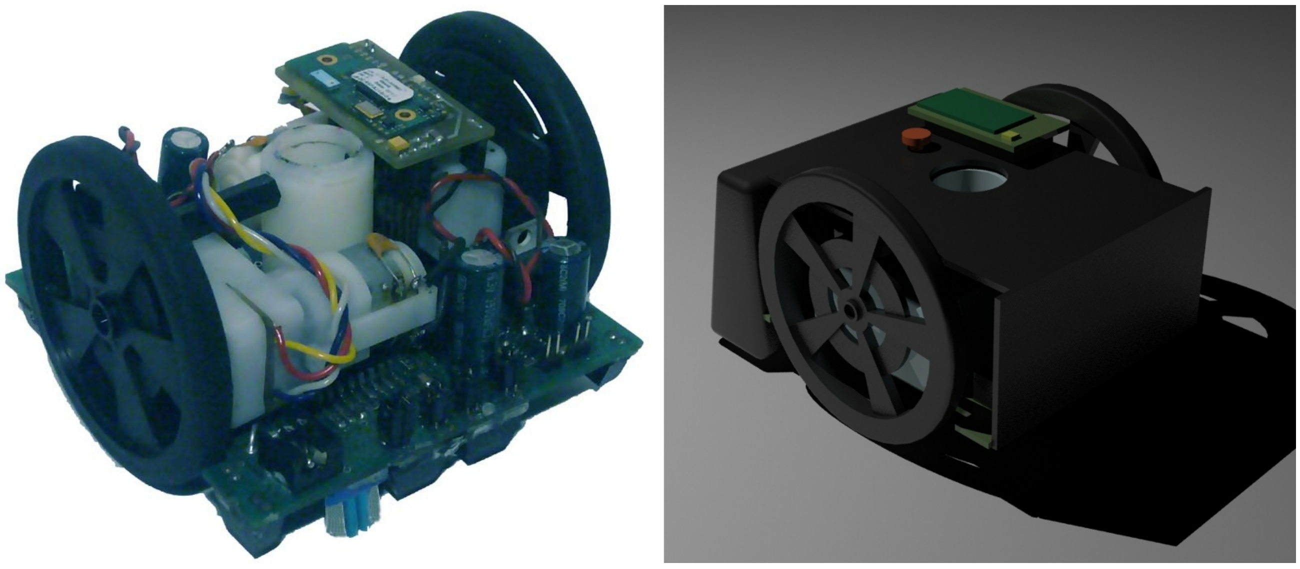


Figure 11: The first real simulated robot on the simulator with his 3D model