18th International Conference
ENGINEERING MECHANICS 2012
Svratka, Czech Republic, May 14 – 17, 2012

pp. 1493–1496
Paper #245

ARTIFICIAL ANT COLONY METHOD FOR STATE-SPACE EXPLORATION

S. Vechet[*], J. Krejsa[**], J. Hrbacek[+]

Abstract: *Finding a way through an unexplored environment belongs to actual problems in many artificial agent systems. Common algorithms as state-space searching or rapidly exploring random trees are used when the map of given environment is known. In this paper we present a simulation experiments with multi agent system which is represented as artificial ant colony.*

Keywords: ***artificial ant colony, path planning, state space exploration.***

1. Introduction

Finding a way through an unexplored environment belongs to actual problems in many artificial agent systems. Common algorithms as state-space searching or rapidly exploring random trees are used when the map of given environment is known. In many real-world applications the agent is faced to problem find path in partially mapped or unknown environment (Krejsa,2011).

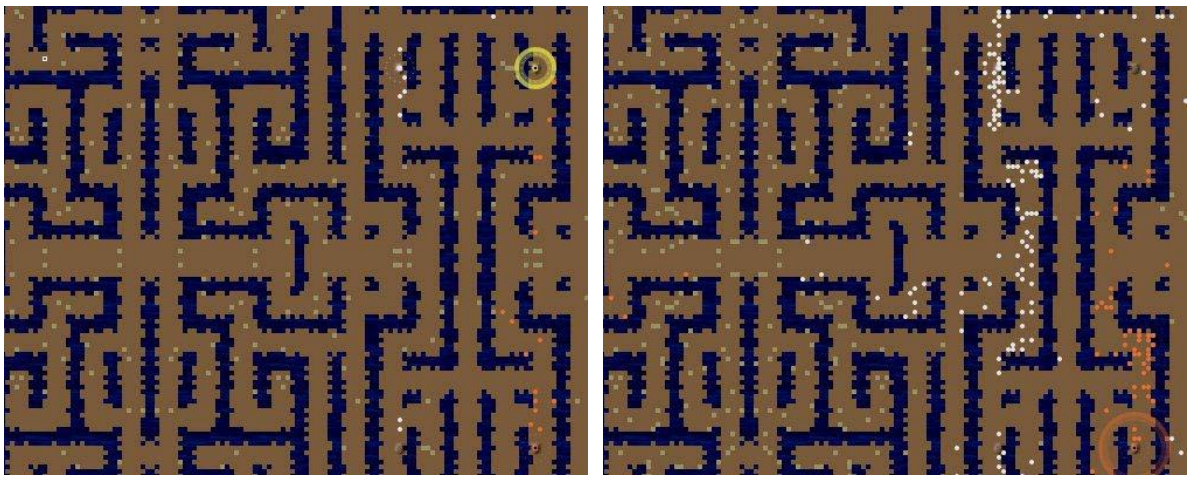

Fig. 1. Simulation framework

Presented algorithm of artificial ant colony is used to finding a path through unexplored environment. The main issue is to explore the biggest space in shortest time. Each ant is represented as simple agent with its own searching strategy (Krejsa,Ondrousek,2011). As each agent is a part of the ant colony, this local strategy can be very simple, but the exploring efficiency of the whole colony can be huge. Each agent has local information only, about near environment based on visible range of agents sensors.

[*+] Ing. Stanislav Vechet, Ph.D., Ing. Jan Hrbacek: Brno University of Technology, Faculty of Mechanical Engineering, Faculty of Mechanical Engineering, Brno University of Technology, Technicka 2896/2, Brno, Czech Rep, e-mail: vechet.s@fme.vutbr.cz, yhrbac03@stud.fme.vutbr.cz

[**] Ing. Jiří Krejsa, PhD.: Institute of Thermomechanics ASCR, v.i.i, Brno department, Technická 2, 616 69, Brno, CZ, email: krejsa@fme.vutbr. cz

2. State-space exploration

Path planning using artificial ant colony method is based on state space searching algorithms. Each artificial ant/agent implements simple local behavior which can results in complex global solution. Each agent is represented by simple rules described in high level programming language. In our case the Python is used. The main advantage of Python is the simple and powerful syntax. In some cases when pseudo-code is needed the *pythonic* syntax is used. In next step, there is no conversion needed for writing a runnable simulation code.

The state space for searching is represented by static environment in two dimensional rectangular grid G. Each cell C in the grid can represents one of possible states S. For better understanding of simulation results each state is also represented by different color. All used states and its colors are described as follows:

S_{free} – Free unoccupied space (brown),

S_{water} – Water represents an obstacle (blue),

S_{food} – Food represents free cell for movement but with food placed on it (light brown),

S_{ant} – Ant represents single agent (another colors), each ant colony has its own color,

$S_{anthill}$ – Anthill is the place where the new ants are born and also the anthill of opponent colony is the goal G for path planning.

Each agent can perform one of four possible actions A:

A_{north} – move to the north,

A_{south} – move to the south,

A_{east} – move to the east,

A_{west} – move to the west,

The simulation starts with single agent in each ant colony. Each food founded by the agent results in new agent in the colony. The colony has to grow rapidly to explore as much space as possible in shortest time which results in quick finding of goal for the path planning.

The need of finding food is in opposite to space exploration, because of those agents which are searching for food cannot perform the space exploration task. There are two main tasks for agents: exploration or searching, each agent has to decide what task to choose in single simulation step. The simulation is discrete, in one simulation step the agent can change its position, gathering food or explore some unexplored space. The space is explored when is in small given range from agent which represents the visible neighborhood.

Tab. 1: Algorithm SRCH1 of simple Agent

```
1.  GetClosestFood()
2.  GoStraight4Food()
3.  if Obstacle(x,y):
4.      Avoid()
5.  if AnotherAntHill(x,y):
6.      GoalReached()
```

However, the algorithm SRCH1 (the main task for agent is searching) looks very simple its very powerful. The main task of this algorithm is just searching nearest food and avoiding obstacles, if necessary. This kind of algorithm has also big ability for state space exploration because of the near food is quickly harvested and is necessary to find a new food in higher distances. On the other hand

the goal is founded accidentally in case that the agent in visible distance to the goal, this make in some cases the algorithm very slow.

Tab. 1: Algorithm of modified Agent, SRCH2

```
1.  Get4OneAntOneFood()
2.  GoStraight4Food()
3.  if Obstacle(x,y):
4.     Avoid()
5.  if AnotherAntHill(x,y):
6.     GoalReached()
7.  if AnotherAntHillLocationIsKnown():
8.     SendAllAnt2AntHill(x,y)
```

The algorithm of smarter agent SRCH2 is similar to the simple algorithm but with one small difference. In this method each agent goes for one different food, in such a case there are lot of agents without task. If the position of goal is known, but the path to goal is still unknown, all remains agent are searching the way to the goal. This simple change makes this algorithm much more efficient.

3. Simulation environment

For simulations the AI challenge framework was used. The native language of AI challenge framework (figure 1) is Python for running the simulation code and Java for showing simulation results. This framework is freeware and open source so anyone can used it for own experiments.

For comparing the efficiency of our algorithms there are number of different environments. Each environment can run simulation at least two artificial ant colonies in duel. The maximum number of colonies in one environment is ten. The main reason for using more colonies in one simulation is the possibility to straight comparison of colonies in performed simulation.

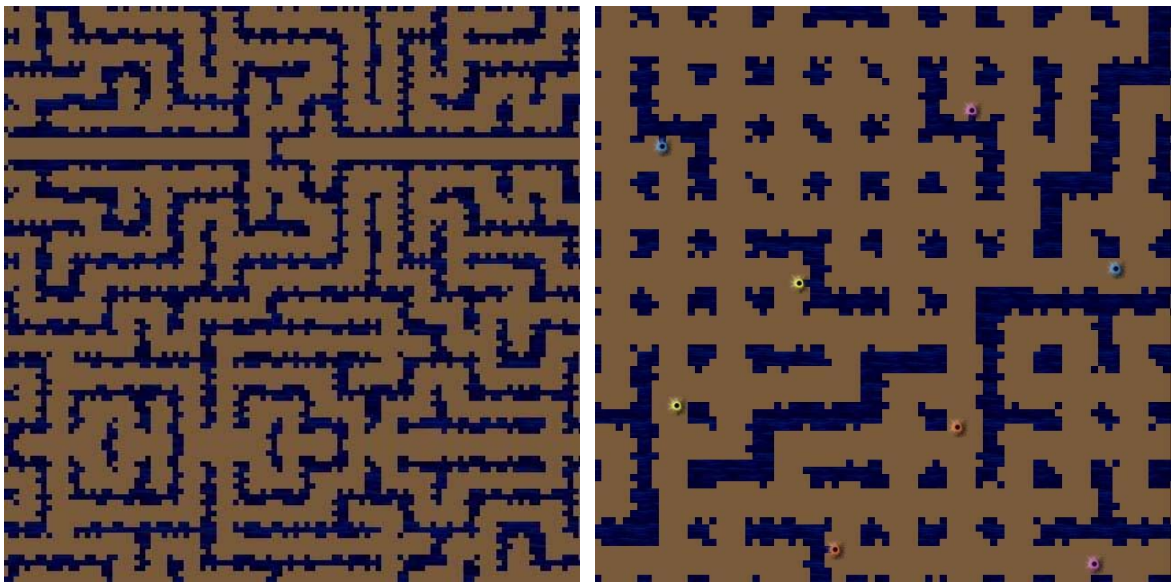

Fig. 2.Typical rectangular mazes with open cycles

Typical rectangular mazes for two artificial ant colonies are shown on fig. 2. The main issue on this kind of environment are the open cycles around the obstacles, which usually results in infinite loops so the ant colony cannot efficiently explore other parts of given map.

4. Simulation results

This chapter presents the simulation results from performed experiments. We prepare number of simulations with two artificial ant colonies for direct comparison and also some experiments with more, usually six, ant colonies (see figure 3). Each colony used in single simulation has different strategy. As the main criterion for the classification of the efficiency of the method, the number of ants in one colony, was used. The exploration strategy is based on efficient algorithm for path planning and also on effective food searching policy. Thus, the final number of ants in one colony, after the maximal number of simulation steps, was used as the main criterion for the classification.

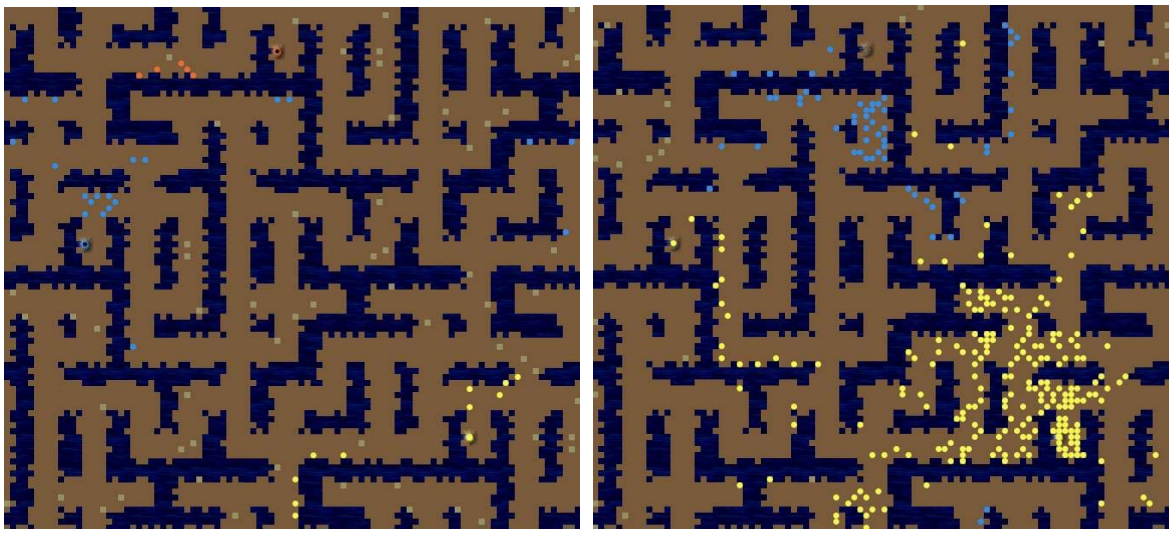

Fig. 3.Comparison of algorithms SRCH1 and SRCH2

5. Summary

Methods for finding path through unknown environment via artificial ant colony algorithms were presented in this paper. In simulation experiments were successfully tested two main approaches to artificial ant colony behavior: searching and exploration. The searching method is very easily implemented and is computationally fast, on the other hand the exploration method is more complicated for implementation, more time consuming but very efficient.

6. Acknowledgement

Presented results were obtained with the support of the Academy of Sciences of the Czech Republic under the research plan AV0Z20760514 and Brno University of Technology under project FSI-S-11-15.

References

Krejsa, J., Reduction of face detection false positives in mobile robot interaction using proximity sensors, in *Proceedings of Mendel 2011, 17th International Conference on Soft Computing*, pp. 540-545, 2011

Krejsa, J., Ondrousek, V.: The design of human-machine interface module of presentation robot Advee, in *Mechatronics, recent technological and scientific advances*, Springer, pp. 423-428, 2011