Movement Optimisation of Cooperating Ant Colony: A Study in Agent–based Social Simulation

Machová Kristína[1], **Peter Illiáš**[1]

[1]Department of Cybernetics and Artificial Intelligence, Technical University, Letná 9, 042 00 Košice, Slovakia, Kristina.Machova@tuke.sk,

Abstract: The article is dedicated to the field of ant algorithms. It focuses on the problem of searching the shortest path in a graph using ant algorithms in combination with some artificial intelligence methods: evolutionary algorithms and multi-agent systems. The article presents a simulation program, which is able to search the shortest path in a graph. The process of searching the solution is simulated also in a graphical way – in a visual form.
Agents, which simulate ants in our work, need only a limited memory. Thus, the use of the presented implementation is not restricted by the complexity of the solved problem. It is an advantage of our simulation. The size of population is controlled by natural selection. On the other hand, agents need more time to get lost in bigger environments. Therefore, the population size increases more quickly in greater environments. The implementation is adaptive as well. If some edge – graph route – is deleted while the program is running or a new one is created, the system is able to adapt to the changed environment.
The designed simulation program can be used for solving various problems related to the following domains: electronic market, computer maps, traffic planning, computer games, labyrinth search by a robot, connection-oriented network routing and connection-less network routing.

Key words: ant algorithms, evolutionary algorithms, multi agent systems, graph, the shortest path

1. Introduction

The nature inspires researchers in various ways. Airplanes were designed according to bird wings. Robot movements were copied from movements of insect. Many resistant materials were made by the same technology which spiders employ to produce their nets. After many million years of evolution, the nature offers many solutions of various problems not only in the field of the research and technique.
Algorithms based on the behaviour of ants – ant algorithms – were used at first by Marco Dorigo[1] and his co-workers under the name 'Ant Colony Optimization' (ACO) algorithms. ACO were realised using on multi-agent systems in order to solve hard combinatorial optimisation problems [2], [3]. The most known problems from them were Travelling Salesman Problem (TSP) and Quadratic Assignment Problem (QAP). New approaches and applications of the ant algorithms appear during the short history of these methods. The latest applications deal with traffic movement, graph colouring, routing in communication nets, and so on. The idea of the ACO, like many other ideas in the field of research and technique, is based on principles, which are present everywhere in the nature. ACO algorithms, according to the name, are based on the principles which hold in the world of natural ant colonies. Ants are "social" kind of insect, like bees. The basic principle of this "social system" is the following: colony is always preferred to individual ants. The ability of the ants to find a source of food and the shortest route to the food is very interesting. A secret of their skill is in releasing pheromone in their surrounding and thus in creating a pheromone

[1] Marco Dorigo – professor at Universit´e Libre de Bruxelles. He is also the founder of the association IRIDIA, which associates people interested in theory and applications of ant algorithms.

trail. The ants can percept this pheromone trail and move along it. In case of trail crossing, the ants make a decision where to go – according to the intensity of the pheromone trails. They usually follow the route with the most intensive pheromone trail. The trail also enables them to go back to their nest. In this way, several routes between the nest and the food source are created. The ants are gradually able to find the shortest route to the food source and then the majority of ants will use this route in future.

2. Collective intelligence

Scientists from many research fields have taken an interest in "social" behaviour of insect species mainly due to ingenious structure of their communities. Some kinds of insect, including ants, have the ability of self-organising in the form of a super-organism. In reality, ants are able to find the shortest path from a source of food to the nest without using eyes. Also, they are able to adapt their behaviour to environmental changes. For example, they can find the new shortest path in the case, when the old one is not usable any more because of barricading it by an obstacle. (Figure 1) illustrates the situation, when an obstacle appears in the middle of the path connecting the nest and the source of food.

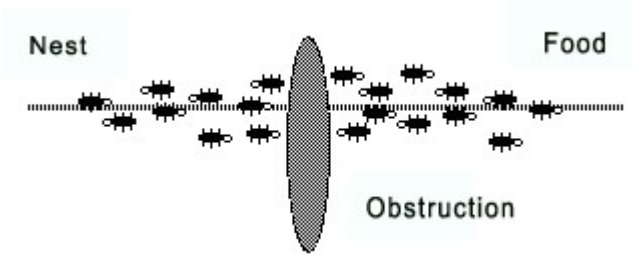

Figure 1: An obstruction of the shortest path

In the moment the obstacle appears on the path, those ants, which find oneself in front of the obstacle, cannot follow the pheromone trail. They have to decide to which side (left or right) they turn. To take an unbiased position, let us suppose that the half of ants turn left and the other half of ants turn right (Figure 2).

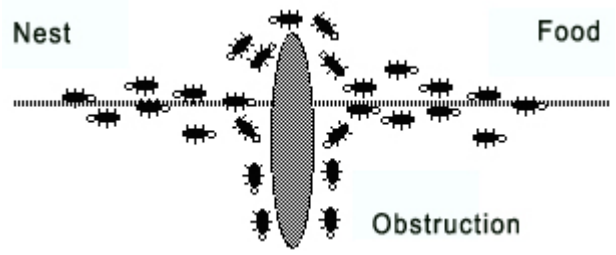

Figure 2: The primary uniform ant distribution

The ants, which select the shorter path around the obstacle, are able to reconstruct the pheromone trail on this new path more quickly than the ants, which select the longer path around the same obstacle. Pheromone evaporation is also an important factor to come into play. Without pheromone evaporation, quantity of pheromone on both paths would be in balance after returning the slower ant group. As a result, an ant would have to select between two equally strong trails on the crossroad. Because of pheromone evaporation, more quantity

of pheromone is evaporated from the longer trail. Thus, the shorter path becomes stronger and stronger and, subsequently, more and more ants select this shorter path. Finally, all ants move along this shorter path (Figure 3).

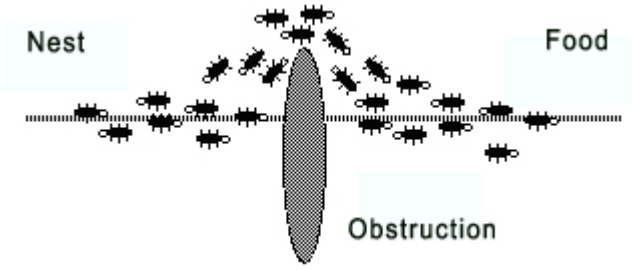

Figure 3: The final selection of the shortest path around the obstacle

Because of this positive auto-catalytic process, incoming ants can easily make a decision to select shorter path. Auto-catalysis is a very strong property. This property is also used by evolutionary algorithms within selective and reproductive mechanisms. It is about preferring better organisms (results) – this preference determines the direction of subsequent search. A distributed optimisation mechanism can be based on this principle, where pheromone trails represent a memory. This memory can be read from and wrote into by all ants in a colony. The memory serves as a communication channel enabling communication between individual ants. It is interesting, that an individual ant can randomly find the shortest path to food as well. But only ant colony – a group of ants can find the optimal solution - the shortest path to food. Thus, the shortest path finding is an emergent behaviour of the ant colony.
Ants find the path using on indirect communication – using on changing the environment they operate in. The phenomenon of stigmergy relates narrowly with the activity of social insect. The term (stigmergy) was introduced by French biologist Pierre-Paul Grassé in 1959 to refer to termite behaviour [6]. He defined it as: "Stimulation of workers by the performance they have achieved."The essence of this phenomenon is the following fact: the colony coordination does not require any direct interactions between colony members (agents). Colony members communicate between each other only using on stimuli, which remain in the environment as results of their previous activities. In case of active stigmergy, results of their previous activities are characteristic marks, for example pheromone trails of ants. In case of passive stigmergy, results of their previous activity are local changes of the structure or state of the environment.

3. Used methods

We used basic ideas of ant algorithms in our experiments. The most known ant algorithms are Ant System [4] and ACO (Ant Colony Optimisation) [3]. These algorithms simulate the behaviour of an ant colony using on “artificial ants”. These artificial ants (similarly to real ants) represent a colony of cooperating individuals, which use pheromone marks for both activity control and communication. They search a sequence of local steps in search space to find the shortest path. They also perform random decisions without any procedure defined beforehand. All their decisions are local in space and also in time.

Artificial ants differ from real ants in the following:

- Artificial ants live in a discrete world and their movements in space consist of transitions from one discrete place to another.
- Artificial ant has an internal state representing memory of his previous actions.
- Artificial ants conserve the values of a pheromone trail, which leads to "solution of good quality".
- To improve results of ant algorithms and systems based on them, the algorithms can be enriched by some methods (e.g. look-ahead, local optimisation, backtracking, etc.) which cannot be found in real ant activities.

We decided to base our own simulation of ant algorithms only on using basic ideas of ant algorithms without using complicated mathematical models. Our aim was to show a simple simulation of ant colony behaviour using on artificial intelligence methods. Particularly, we used techniques of evolutionary algorithms and multi-agent systems. Multi-agent systems were used to create an ant population searching given graphs. Evolutionary algorithms were used to refine obtained results and their optimisation.
We used a modified version of the Ant Colony Optimisation algorithm, which can by used in distributed environment:

```
procedure ACO_Meta_heuristic()
        while (stop_condition = false)
                select_activity:
                        generate_ant();
                        evaporation_pheromone ();
                        daemon_action();
                end select activity
        end while
end procedure

procedure generate_ant()
        initialize_ant();
        M = actualize_memory();
        while (actual_state ≠ goal_state)
                A = local_route_table();
                P = count_movement_probability(A,M,restrictions);
                next_step = select_step(P,restrictions);
                go_to_next_state(next_step);
                if (online_pheromone_inserting = true)
                        set_pheromone();
                        correct_route_table();
                 end if
                M = modify_internal_state();
        end while
        if (online_pheromone_inserting = true)
                evaluate_solving ();
                save_pheromone();
                correct_route_table();
         end if
        die();
end procedure
```

3.1 Evolutionary algorithms

Evolutionary algorithms are based on the observation of properties of biological evolution, which was described by Ch. Darwin in 1859. Biological evolution represents the development of the genetic information in individual organisms during generation replacement. Evolutionary process contains mainly the following important components:

- natural selection
- random mutation
- reproduction process

The natural selection enables a stronger individual to succeed better in an environment. This fact means its greater chance for reproduction. The reproduction process is an important component of evolution because of the opportunity to transfer genetic information from parents to children. During reproduction by crossing, the part of information in genetic code is replaced. Moreover, this information can be randomly mutated. The random mutation is a random change of a certain sector of the genetic information. The most important part of evolution process is just the reproduction process. Some number of descendants (usually two) is selected according to their strength. The stronger descendant (agent), the higher probability to be selected. The whole process is realized within several phases:

- selection
- crossing
- returning of new individuals to population

There are two strategies for adding new individuals to population during evolution process. Using the first strategy, a completely new population is generated. In the second strategy, several new agents are generated. These new agents consequently replace the same number the weakest agents from the old (original) population. The evolutionary algorithms, serving as an optimization method, exist nearly thirty years. Many new techniques were developed during this time [8], [9].

3.2 Multi-agent systems

The most frequently cited definition of agent is the definition by Wooldridge and Jennings [10], which differs between strong agents and weak agents. A weak agent is defined as a hardware system or (more often) a software system, which fulfils the following requirements:

- Autonomy: The agent acts without any direct human or another intervention. It has control over its actions and internal states.
- Social ability: Agents communicate with other agents (occasionally with humans) using on some communication language.
- Reactivity: Agents perceive their environment, which can represent physical world, user, other agents, Internet or a combination of above mentioned possibilities. In addition, agents react on changes in the environment in real time.
- Pro-activity: Agents do not perform only simple reactions according to the state of the environment. Their acting can be goal-oriented with taking over initiative.

The definition of a strong agent adds to above given requirements some other concepts, which can characterize mental and emotional states for example cognisance, belief, attention and so on. This conception was reformulated in the work by H. S. Nwan, in which the following definition of strong agent is introduced: the agent has to have at least two from the following three abilities:

- Autonomy: Agent is able to operate in an environment without human intervention even if the world is not explicitly described. The key element of the autonomy is pro-activity – the ability to act in a goal oriented way and initiatively.

- Cooperation: A set of social abilities, which enable agent to communicate with other agents or human using on some communication language. Cooperation is one of very important abilities of multi-agents systems.
- Learning: In order to enable an agent to react within a dynamic and non deterministic environment and perform some actions in it, learning must be present in the form of an interaction with the environment. It is expected that the quality of agent behaviour in the environment can be improved using on learning.

Several agents are grouped together to reach some goal in multi-agent systems and particular relations between agents of the group are defined. In such system, each agent has some means (it can dispose with) and limitations (it has to act within these limitations to reach its goal). No agent in the system is unusable or redundant.
Each agent tries to perform a given task in its best way which is reflected in the final solution of the problem. In this "agent colony", agents coordinate their abilities, knowledge, goals and plans in order to solve a given problem collectively. These agents can work collectively on the same problem or each agent can work on a separate task while all tasks are related to one another.
There are several reasons, why the use of multi-agent systems provides an advantage:

- Parallel system – the possibility of using more agents to speed up system working using on parallel methods.
- Robust system – if control and responsibility are suitably distributed among particular agents, then the system can tolerate failure of one or more agents.

An ant as an individual has no global knowledge about task it performs. Its decisions are local and unpredictable. The simplicity of ants provides ability to model an ant colony as a multi-agent system.

4. Implementation – simulation of graph search

A comparison of a graph search and ‘food search’ performed by ants provides some similarities between these two problems. Ants try to find the best (the shortest) path between two places (nest-food) in a given environment. Scientists try to find the best (the shortest) route joining two nodes in a given graph.
The main idea of our designed system is to provide ants with the possibility to move in a given graph. In this context, the intensity of pheromone marks on separate graph edges is studied. The graph nodes represent various places in the environment, where ants can stop. These places can be called cities. The edges, which join the nodes of the graph, represent routes between city pairs in the environment. This ‘virtual environment’ will be inhabited by a population of agents, which represent separate ants.

4.1 Simulation of an ant colony using on the multi-agent system

Agents move in a given graph from one city to another one. They also collect food when they reach a food source. And they ‘drop’ the food when they come back to the nest. An agent, moving along a route in the graph, marks it by its pheromone to update pheromone trail. Finally, agents have to be able to decide, which route they should follow, according to the intensity of pheromone on edges. Agents can perform only certain actions according to their position, as illustrated in Table 1. All actions listed in Table 1 need only local information. The memory of each agent is of low capacity, but the capacity of this memory is sufficient for each ant to know the path back to the nest. The easiest way how to design a multi-agent simulation of the environment is to use a turn-based system. This kind of system is like a game – each player can realize only one step in each cycle of this game. This idea forms a

base of the simulation in which each agent can perform only one action from Table 1 per cycle.

Table 1: Actions, which can be performed by agents according to the place they are located in.

Agent position	Performed action
In the city	Next route selection and marking this route by pheromone
On the route	Moving along
In the nest carrying food	Putting down the food
In the food source	Taking the food and coming back to the nest

All elements of the environment are simulated using on the following classes (in C++ programming language):

a) **Ant class** represents an individual agent – an ant
b) **Route class** represents edges of the graph – paths between cities
c) **City class** represents nodes of the graph – individual cities
d) **Civilization class** represents the environment in which ants move (live)

City class represents X and Y coordinates of a position in the environment. This class is essential for calculating the distance between two cities.
Route class needs two pointers to city class in order to define the cities it connects. Another property of this class is the length of the route between the cities. The longer the route, the greater number of steps is needed for an ant to walk along this route. In addition, the class represents the intensity of pheromone located on the given edge. Very important aspect, which has to be simulated, is the evaporation of the pheromone trail. Thus, the class has to contain also this information.

Class Route – a sample of class definition

```
class Route
{
 private:
 float Length;          // route length
 int Pheromone;       // pheromone intensity on the route
 City *FirstCity;       // cities, which are connected by the route
 City *SecondCity;

 public:
 void EvaporatePheromone();      // simulation of the pheromone evaporation
    // constructors, boundaries, assistant methods and other parameters
    // . . .
};
```

The most important characteristic of each ant in this context is its individual and unpredictable tendency to select one of accessible edges. Each instance of the **ant class** must represent an individual agent with individual and unique properties, which can be realized using a mathematical function based on the level of pheromone intensity. The pheromone intensity has an integer representation. A tendency for the selection of a route based on the pheromone intensity is evaluated for individual agents.

Good flexibility of the behaviour of agents can be guaranteed by a function:

$$T(PL) = \alpha * \sin(\beta * PL + \gamma) \qquad (1)$$

Function T(PL) derives it's name T from the word "tendency" of the selection. It evaluates the tendency for selecting a route with PL (pheromone level). PL is the level of the pheromone intensity on the given route. The route with the greatest value of function T(PL) is selected. α, β and γ are parameters of the ant class initialised as random float numbers from the interval [-5, 5]. These parameters guarantee the presence of different individuals in the population of agents, since agents with the same way of making decisions and acting are not required. These parameters play some task in a evolutionary algorithm (will be described in the next section).
We decided to choose the function T(PL) in the above given form because of using evolutionary algorithms in the role of a tool used for solving the problem: of finding the shortest path between two places (nest-food).

Class Ant – a sample of class definition

```
class Ant
{
 private:
 float α;     // indicates sensibility of ants on pheromone
 float β;
 float γ;
 bool HaveFood;  // indicates ant, when carrying the food

 public:
 float GetTendency(int PheroLevel);
// the tendency of the route selection in relation with the pheromone intensity
 void PickFood();                  // taking the food while in the food source
 void LeaveFood();                 // dropping food in the nest
 void PutPheromone();              // changing the pheromone credit of a route
 void Walk();                      // making another step
// constructors, boundaries, assistant methods and other parameters
// . . .
};
```

Civilization class represents a mean to control the whole environment and simulation process. This class is responsible for evolutionary processes as well. The environment is represented by a graph. This graph has to be created by user using on interface provided to users. The other possibility is to read a graph from the set of graphs (environments) created and stored before. Two nodes of the selected graph have to be specified as nest and food sources. When simulation starts, a random number of ants are created in the nest. Each movement of an ant depends on its actual position. During simulation run, the most frequently used route increase their pheromone credits. After some time, a solution of the given problem is found as a collective decision of the virtual ant colony.
Although this system provides good results, the random number of agents with random characteristics can cause troubles. An agent population, which is able to find the shortest path in a graph, may not be able to find a solution in a complete heterogeneous environment

Class Civilization – a sample of class definition

```
class Civilization
{
 private:
 City *FoodSourceCity;    // food source of the given civilisation in the environment
 City *Nest;              // nest of the given civilisation in the environment
 TList *Routes;           // all routes in the given environment
 TList *Cities;           // all cities in the given environment
 TList *Ants;             // all ants in the given environment
 int NaturalSelection;    // remaining steps before next selection

public:
 void NextTurn();       // realisation of one step in simulation
 // constructors, boundaries, assistant methods and other parameters
 // . . .
};
```

There are some questions, which should be answered: How to find the best agents for solving a given problem? How many agents are needed to perform the search of graphs with various complexities? How to counterbalance various properties of various agents to create a good population? Answers can be produced using evolutionary algorithm techniques.

4.2 Optimisation of the ant colony using on evolutionary algorithms

Evolutionary algorithms provide adaptive optimisation techniques based on the evolutionary process as seen in the nature. A population of individuals, which represent possible solution candidates for a given problem, is controlled by Darwin's principle of "surviving the most successful" in next generations. The most successful individuals can transfer their properties on their descendants – the transfer is mediated by their genes. Gradual population development and its adaptation to the environment accomplished by mutation and natural selection enable to obtain better results. Because of random generation of agents with random characteristics (parameters), a very long time may be needed to obtain a successful solution. Or maybe no successful solution will be found at all. The efficiency of the system can be improved by some evolutionary algorithm techniques like selection, crossover and mutation.

Selection

Selection represents the selection of the best individuals for mating. In our case, the most successful individuals are those, which are able to collect greater amount of food (workers) or which are able to find the shortest alternative paths to food sources (explorers). During the food collection, an agent increases the pheromone credit of a good (frequently used) route, and in such way it influences other ants from the colony. Natural selection enables the best individuals to spread their good properties by mating. Like in the nature, the evolution in the system enables the new generation to be better than the parent generation (the previous generation). Agents, which lost their way in the environment, cannot help to solve the problem. For example a "traveller", which travels only between two nodes and moves along only one edge, is not usable. These individuals are removed from the population.
Our application implements the selection in the following way. The most successful worker is an individual, which collects the greatest amount of food. A counter is implemented in the **Ant class.** It is incremented whenever the agent brings food into the nest. Agents with higher values of the counter will be considered as more successful than others during each selection process. The purpose of another counter is to count how many times an agent used the same

route. Low values of this counter indicate a successful explorer. Higher values of the counter indicate the fact that an agent has lost his way in the environment.

Crossover

New descendants are based on characteristics and properties of successful individuals. Since there are two kinds of individuals in the colony: workers and explorers, two new individuals are created in each evolutionary cycle. One new individual is a descendant of two the most successful workers and the other new individual is a descendant of two the most successful explorers. Genes representing parents' properties are combined in order to create a new chromosome on which the new individual is based. This technique has borrowed its pattern from the nature, particularly from the biological crossover. Each property of the new individual is inherited randomly from one parent.

In our application, crossover is implemented in the following way. In the **Ant class** there is a constructor containing two references (from parents) to an object Ant in the form of a set of three parameters α, β and γ defined in the ant class. A new individual is created by combining characteristics (parameters) of the parents as illustrated on (Figure 4). A chromosome consists of three genes, which represent properties of an individual. A new chromosome is created from "zeros" and "ones" with the same probability. "Zero" means that a property was inherited from one parent (mother) and "one" means that the property was inherited from the other parent (father).

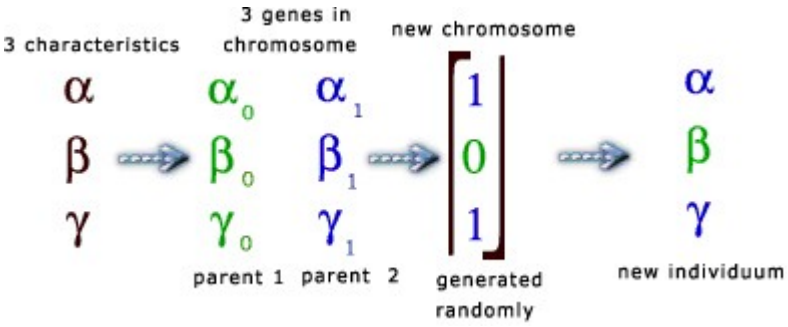

Figure 4: Crossover in action.

Mutation

There is also some low probability of mutation after crossover. The mutation promotes the diversity of the population because one from the three parameters is randomly changed during mutation.

Migration

A completely new individual is inserted randomly into the population during the migration step. The effect of migration is similar to the effect of mutation. Diversity of the environment is increased. The implementation of migration in our application is following. A completely new individual is created by assigning random values to parameters α, β and γ. This implementation is done using on the default constructor of the **Ant class**.

The Ant class has the following form after described modifications:

Class Ant – a sample of class definition after modification

```
class Ant
{
 private:
 float α;        // indicates sensibility of ants on pheromone
 float β;
 float γ;
 bool HaveFood;      // indicates whether ant is carrying food
 int FoodCollected;  // amount of food collected by ant

 public:
 Ant();                          // Default constructor: complete new individual
 Ant(Ant *Father, Ant *Mother);       // crossover constructor
 float GetTendency(int PheroLevel); // tendency of the route selection evaluation
 void PickFood();                  // picking up the food in a food source
 void LeaveFood();                    // dropping food in the nest
 void PutPheromone();                 // increase of the pheromone credit of a route
 void Walk();                        // making another step
 void Mutation();                    // mutation operator

  // constructors, boundaries, assistant methods and other parameters
 // . . .
};
```

4.3 Interface description of the simulation

The main part of our application is a window representing an interface for defining graphs. User can design and draw a new graph, read an old graph used before, or to save the newly designed graph. In each graph there is the necessity to define a nest and a food source. The nest is represented by the blue colour while the food source is marked in red. The application contains a window, showing the route with the highest actual pheromone concentration, as illustrated in (Figure 5).

The window of the simulation interface provides information about in which actual cycle (loop) the simulation is, how many ants reside within population and how much food has been collected. The simulation runs until a termination condition is met. The stop condition can be defined in the form of a maximum number of cycles (loops) or maximum amount of food collected within a given loop.

In order to illustrate the application, a sample of simulation run can be presented. Let us consider the following initial state of simulation in (Figure 6): eight cities and nine edges constitute a test graphical environment to search while nodes 0 and 7 represent the nest and the food source. There is a number assigned to each edge, which represents the pheromone intensity and route length associated with the given edge. The information is expressed in the following form: ‘pheromone | length’.

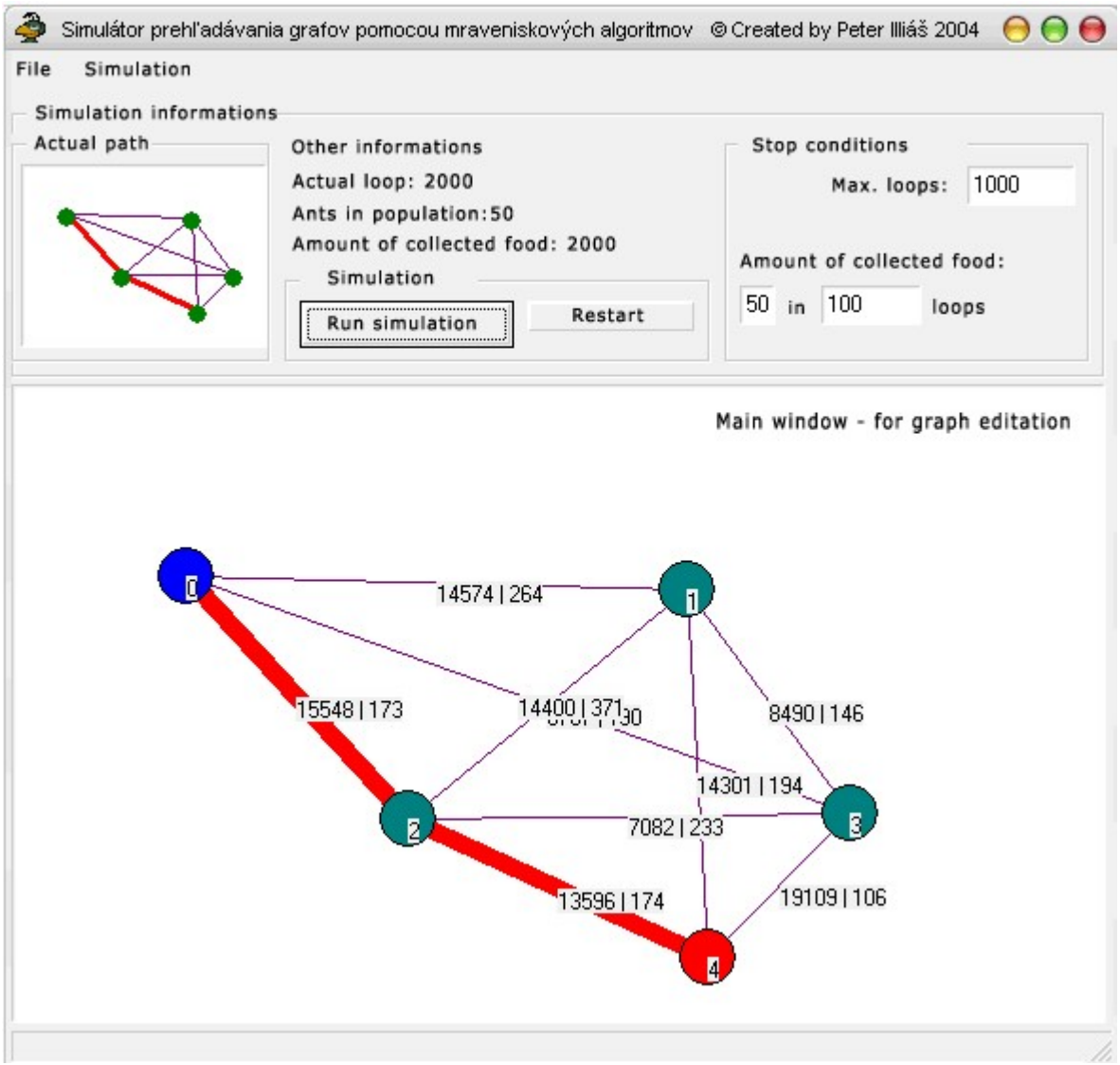

Figure 5: User interface of used simulation program

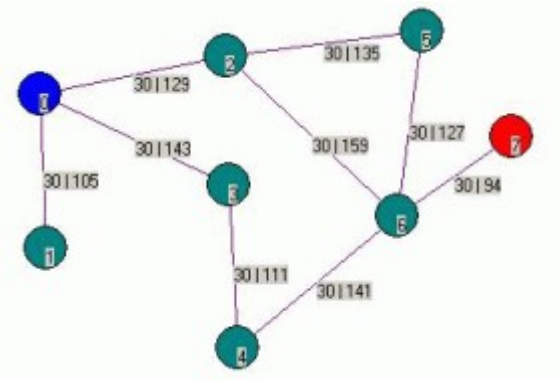

Figure 6: The initial state of simulation

After 115 simulation cycles (Figure 7), the route 0-3-4-6-7 is preferred.

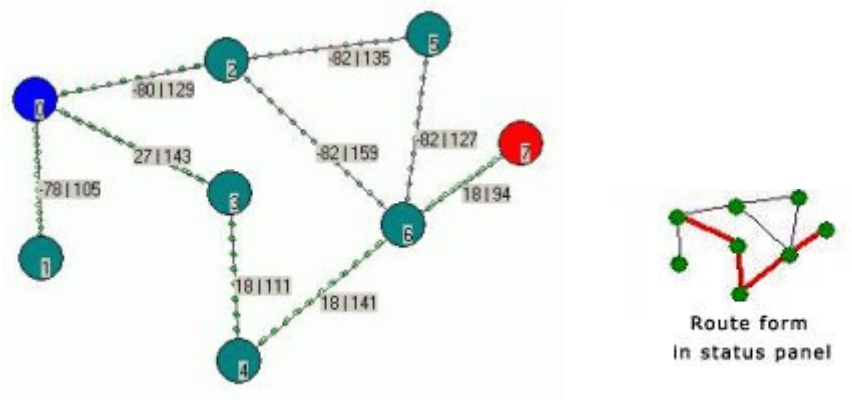

Figure 7: The preferred route after 115 simulation cycles

After 254 cycles of the run, a change in route selection was detected (Figure 8). Now, the route 0-2-6-7 seems to be preferred. Because this path is shorter than the previous one, the pheromone intensity starts to increase along this path. The previous route is evaporated.

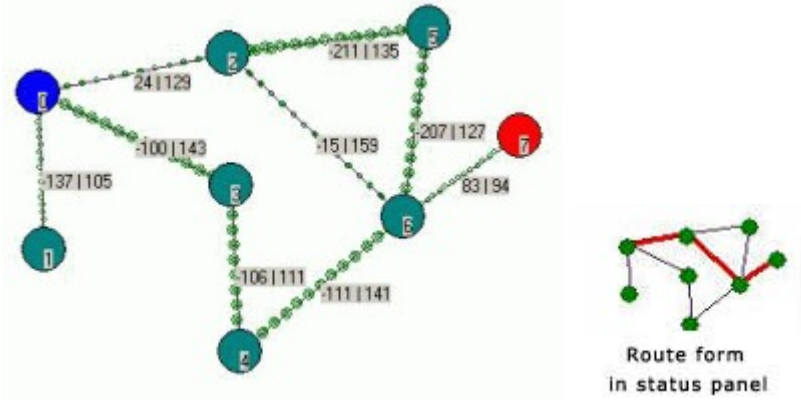

Figure 8: The preferred route after 254 simulation cycles

After simulation completes 316 cycles from the beginning of the simulation run, the termination criterion regarding the maximum number of cycles comes to effect. The result is definitely the route 0-2-6-7, which is marked in thick line in (Figure 9).

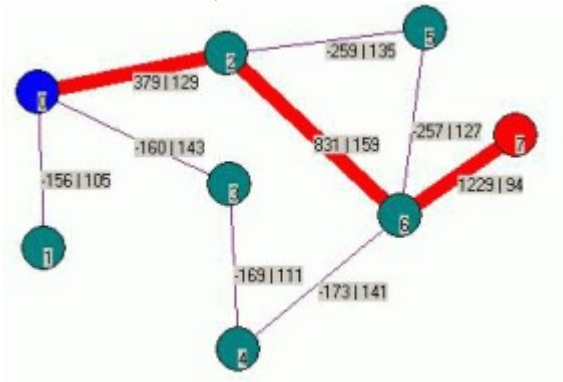

Figure 9: The resulted route of simulation

Many experiments were completed with good results. Efficiency of the system has been increased after adding some improvements. For example, releasing stronger pheromone marks by ants while returning back to the nest with food. It enables for the most successful agent to increase its influence on the other agents.

5 Discussion

Many search algorithms rapidly increase their memory requirements when trying to solve difficult problems. Agents, which simulate ants in our work, need only a limited memory. Thus, the use of the presented implementation is not restricted by the complexity of the solved problem. It is an advantage of our simulation.
Another advantage is the fact, that the size of population is controlled by natural selection. On the other hand, agents need more time to get lost in bigger environments with greater number of alternative paths. As a result, unsuccessful individuals are eliminated from the population later than while searching simpler environments. Therefore, the population size increases more quickly in greater environments.
There is no central system to perform decision making in the presented implementation. Information is distributed among agents and the environment. If one agent is lost in the environment, then the system continues in problem solving and the lost agent cannot influence the result.
The implementation is adaptive as well. If some edge – graph route – is deleted while the program is running or some new one is created, the system is able to adapt to the changed environment.

6 Conclusions

The aim of our work was to design a simulation program, which would be able to present competencies of ant algorithms in a graph searching task. In frame of this simulation we used techniques of artificial intelligence: evolutionary algorithms, multi-agent systems and ant algorithms. The presented implementation is not restricted to only graph search. It is applicable also in tasks with unknown search space.
The designed simulation program can be used for solving various problems related to the following domains: electronic market, computer maps, traffic planning, computer games, labyrinth search by a robot, connection-oriented network routing and connection-less network routing. It would be interesting to use our access based on ant colony simulation on simulation of financial markets [5] or retail markets [7]. Nowadays, computer maps start to be widely used in car industry. The systems built in cars enable drivers to see all possible routes or they navigate drivers to use the shortest route from a city A to B. Traffic planning requires solving very difficult optimisation tasks. Path finding is one of the very frequent problems and applications of artificial intelligence in computer games. The most suitable route is searched for between two nodes in two-dimensional plane (2D games – e.g. strategies) or in three-dimensional space (3D games – e.g. plane simulator).
Our simulation program has proved that ant algorithms represent a suitable method for solving problems, which can be reduced on a graph search. For example, the particular mapping of a web sub-graph be used in the role of new technology, can be used to increase the educational flexibility of a modern laboratory for teaching computer systems [1]. We believe in the future of the ant algorithm theory. Nowadays, systems based on the theory of ant algorithms start to be used in the commerce sphere.

Acknowledgements

The work presented in the paper was supported by the Slovak Grant Agency of Ministry of Education and Academy of Science of the Slovak Republic within the 1/4074/07 project ”Methods for annotation, search, creation, and accessing knowledge employing metadata for semantic description of knowledge”.

References:

1. BABIUCH, M., **The Usage of the New Technologies at the Education at the Department of Control Systems and Instrumentation**. Sborník vědeckých prací, Technical University Ostrava, No. 2, Vol. L II., Ostrava, 2006, 7-12, ISBN 80-248-1211-8, ISSN 1210-0471.

2. COLORNI, A., DORIGO, M. and MANIEZZO, V., **Distributed optimization by ant colonies.** Proceedings of ECAL91 - European Conference on Articial Life, Elsevier Publishing, 1991.

3. DORIGO, M., Di CARO, G. and GAMBARDELLA, L. **Ant Algorithms for Discrete Optimization.** Technical Report IRIDIA/98-10, Universite Libre de Bruxelles, Belgium. To appear in Artificial Life, 1998, http://citeseer.ist.psu.edu/dorigo98ant.html.

4. DORIGO, M., **The Ant system: Optimization by a colony of cooperating agents.** IEEE Transactions on Systems, Man, and Cybernetics--Part B , vol. 26, No. 2, pp. 29--41, 1996, http://citeseer.ist.psu.edu/dorigo96ant.html.

5. GOU, Ch., **The Simulation of Financial Markets Using an Agent-Based Mix-Game Model.** Journal of Artificial Societies and Social Simulation 9(3), 2006 http://jasss.soc.surrey.ac.uk/9/1/15.html.

6. GRASSE, P.P., **La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. La theorie de la stigmergie: essai d'interpretation du comportement des termites constructeurs.** Insectes Sociaux, 6, 1959, 41–81

7. HEPPENSTALL, A., EVANS, A. and BIRKIN, M., **Using Hybrid Agent-Based Systems to Model Spatially-Influenced Retail Markets.** Journal of Artificial Societies and Social Simulation, 9(3), 2006, http://jasss.soc.surrey.ac.uk/9/1/15.html.

8. KVASNICKA, V., POSPÍCHAL, J. and TIŇO, P., **Evolutionary algorithms.** Vydavateľstvo STU, Bratislava, 2000, http://math.chtf.stuba.sk/evol/Prednaska.htm

9. OLEJ, V., **Economic Processes Modeling on the Base of Computational Intelligence.** Miloš Vognar-M&V, Hradec Králové, Česká republika, 2003, 160s., ISBN 80-903024-9-1.

10. WOOLDRIGE, M., **An Introduction to Multiagent Systems**. John Wiley & Sons, Chichester, England, 2002, ISBN 047149691X. http://www.csc.liv.ac.uk/~mjw/pubs/imas/