Image Processing & Communication, vol. 17, no. 1-2, pp. 7-18
DOI: 10.2478/v10248-012-0011-5

SHORTEST PATH PROBLEM SOLVING BASED ON ANT COLONY OPTIMIZATION METAHEURISTIC

MARIUSZ GŁĄBOWSKI, BARTOSZ MUSZNICKI, PRZEMYSŁAW NOWAK, PIOTR ZWIERZYKOWSKI

Poznań University of Technology, Faculty of Electronics and Telecommunications,
Chair of Communications and Computer Networks, Polanka 3, 60-965 Poznań, Poland
bartosz@musznicki.com, przemyslaw.nowak@inbox.com

Abstract. The Ant Colony Optimization (ACO) metaheuristic is a versatile algorithmic optimization approach based on the observation of the behaviour of ants. As a result of numerous analyses, ACO has been applied to solving various combinatorial problems. The ant colony metaheuristic proves itself to be efficient in solving $\mathcal{NP}$-hard problems, often generating the best solution in the shortest amount of time. However, not enough attention has been paid to ACO as a means of solving problems that have optimal solutions which can be found using other methods.

The shortest path problem is undoubtedly one of the aspects of great significance to navigation and telecommunications. It is used, amongst others, for determining the shortest route between two geographical locations, for routing in packet networks, and to balance and optimize network utilization. Thus, this article introduces ShortestPathACO, an Ant Colony Optimization based algorithm designed to find the shortest path in a graph. The algorithm consists of several subproblems that are presented successively. Each subproblem is discussed from many points of view to enable researchers to find the most suitable solutions to the problems they investigate.

1 Introduction

This article focuses on a presentation of possibilities and the usability of the application of the Ant Colony Optimization metaheuristic for solving the Shortest Path problem. The following subsections present basic background and introduce basic notions and parameters. Further sections discuss individual questions related to the algorithm proposed by the authors. The article is concluded with a summary of basic facts and conclusions.

1.1 Ant Colony Optimization algorithms

The ant colony algorithms were initially proposed by Marco Dorigo, who in his doctoral dissertation of 1992 [1] presented his first algorithm based on the behaviour of

ants, called the Ant System (AS), in the application for the classical Travelling Salesman problem. Arguably, the most commonly used and the most successful research line related to the ant colony algorithms concerns their applications to combinatorial optimization problems and is called Ant Colony Optimization (ACO) [2]. The inspiration for this algorithm was provided by the observation of a colony of the Argentine ant *Linepithema humile* that are capable of finding, subject to certain conditions being satisfied, the shortest path from among a number of alternative routes between the nest and a source of food. Ants are able to find food thanks to the phenomenon of stigmergy, i.e. the exchange of information indirectly via the environment by depositing pheromones, while the information exchanged has a local scope; only an ant located where the pheromones were left has a notion of them. During their walk ants leave the so-called pheromone trail on the ground that, though volatile and evaporating over time and thus reducing its attractive strength, makes other ants, including the ant that actually left the trail, follow the trail to find the best way to the target place — food, or the way back to the nest. The more "marked" the way is (i.e., has a higher concentration of pheromones), the more likely it is to be chosen by an ant running its errands. At the same time, the trail is increasingly enhanced when a greater number of ants chooses a given path, while in the case of finding an even better way, the deposited pheromones begin to evaporate. If, however, an ant does not find any trail on its way, its choice of further path is purely random — unless one of the paths of choice includes obstacles. This mechanism can be described as a positive feedback while on shorter routes and a negative feedback on longer paths. The volatile nature of pheromones encourages exploration of new paths following a decrease in the intensity of pheromone trails and, in this way, biases the choice process of a given route. This skill of a colony to find out and mark the best routes can be viewed as a process of collective learning of ants.

Recently, ant algorithms have been gaining popularity in such areas as combinatorial problems solving where they can be applied in finding appropriate paths in a graph. They constitute one of the numerical methods that have been inspired by biology and have found application in robotics and telecommunications, among others.

1.2 Problem of the shortest path

For the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where V is the set of vertices and E is the set of edges we assign the cost a_{ij} to each of its edges $(i, j) \in \mathcal{E}$ (alternatively, this cost can be also called the length). For the resulting path $(n_1, n_2, \ldots, n_k)$, its length can be expressed by Formula (1).

$$a_{ij} = \sum_{i=1}^{k-1} a_{n_i n_{i+1}} \tag{1}$$

A path is called the shortest path if it has the shortest length from among all paths that begin and terminate in given vertices. The shortest path problem involves finding paths with shortest lengths between selected pairs of vertices. The initial vertex will be designated as s, while the end vertex as t.

A number of basic variants of the shortest path problem can be distinguished:

- Finding the shortest path between a pair of vertices.
- Finding the shortest paths with one initial vertex.
- Finding the shortest paths with one end vertex.
- Finding the shortest paths between all pairs of vertices.

This problem finds its application in a number of areas such as routing in telecommunications networks, dynamic programming or project management.

2 ShortestPathACO Algorithm

As a result of our study we present in this paper ShortestPathACO — a proposition of an algorithm to solve the shortest path problem that is based on the metaheuristic of ant colony. It is worthwhile to remark that in the case of a problem that is so frequently studied and considered, such as the shortest path problem, it is difficult to develop an algorithm that would outperform the already existing solutions. An additional impediment is the fact that the metaheuristic of ant colony has been developed for solving $\mathcal{NP}$-hard problems, whereby the results obtained following its application provide approximations of optimum results. With the above difficulties in mind, the article aims at presenting an algorithm that would offer, at least to a certain degree, a possibility of obtaining characteristics (properties) comparable or better to those obtained with the algorithms used hitherto that are capable of calculating results in an accurate way.

The ShortestPathACO algorithm includes a number of subproblems that will be discussed separately for convenience in the next sections. Each subproblem is approached from several different perspectives to allow researchers to choose and adjust a method for a solution of a specific subproblem. This kind of an approach has the advantage of creating a procedure that would be best suited for a solution of the whole problem in its entity. The procedures presented in the following sections relate to type of the widely adopted representation of ACO algorithms [2].

3 Initiation of the algorithm

The ShortestPathACO algorithm uses the following parameters:

m — the number of ants

α — the parameter that defines the influence of pheromones on the choice of the next vertex

β — parameter that determines the influence of remaining data on the choice of the next vertex

ρ — parameter that determines the speed at which evaporation of the pheromone trail occurs; takes on values from the interval $[0, 1]$

τ_0 — initial level of pheromones on the edges

τ_{min}, τ_{max} — the minimum and maximum acceptable level of pheromones on edges

s — initial vertex

t — end vertex (required when the shortest path between a pair of vertices is to be calculated)

The number of ants m influences considerably the accuracy of a solution obtained as a result of the operation of the algorithm. At the same time, it increases time of its operation. Hence, it is necessary to match their number (the number of agents) with a corresponding and specific case under investigation. This will make it possible to negotiate a compromise between the accuracy of the algorithm and the duration of its operation. The parameters α and β allow us to modify a method for a selection of the next vertex, which in turn influences considerably the quality of the solution. The parameter ρ decides on the speed at which the pheromone trail evaporates on individual paths. The quicker it is evaporated, the more possibilities arise to increase exploration-oriented initiatives on the part of ants and to find new solutions, whereas its slower evaporation makes ant use solutions that have already been found. Parameters related to the pheromone level enable to have it regulated (adjusted) accordingly, which is then translated into a change in the characteristics of the operation of ants.

Depending on a variant of the shortest path problem, the initial vertex s and/or the end vertex t may be also required. Depending on a chosen method for finding paths, discussed in the following section, a list storing vertex occupation time, i.e the amount of time needed by a given ant to reach the vertex, and the current value of time may turn out to be necessary as well. Whereas certain methods

Function Initialize($\mathcal{G}, s$) of ShortestPathACO

Data: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, initial vertex s

```
C ← 0
foreach i ∈ V do
    foreach j ∈ V do
        τ_ij ← τ_0
        vi_edges_ij ← false // Set (i,j) edge to not visited
        if a_ij > C then C ← a_ij
    end
    vi_nodes_i ← false          // Set i node to not visited

end
length ← C(V − 1)
time ← 0
for k ← 1 to m do
    Reset(G, k) // Erase data gathered by ant k
    SetRun(k, 0)     // Set the counter of ant's routes k to 0
    SetNode(k, s)  // Set the current vertex of an ant k
    SetVisited(k, s) // Set vertex s as visited for ant k
    Add(list, time, k)      // Add ant k to list with time 0
end
convergence ← 0
last_length ← +∞
```

of choice of the next vertex require storing the vertices and/or edges that have already been visited.

Function 3 shows a method to initiate the algorithm that uses all the mentioned parameters. Additionally, while determining the convergence, the auxiliary parameters have been initiated.

4 Methods for finding paths

One of the more important elements of any algorithm that is based on the use of the ant colony system paradigm are the rules for the process of establishing paths leading from the initial point to the end point. Subsections 4.1 and 4.2 show two methods for finding paths developed for the ShortestPathACO algorithm. Item 4.3 presents the conclusions of the comparison of the main characteristics of the two methods.

4.1 Finding paths for each ant one by one

In this method a choice of a subsequent vertex is made iteratively for each ant one after another. In each iteration the pheromone trail is evaporated. After reaching the target vertex on a path that led the ant to this vertex, the pheromone trail is reinforced by $\Delta\tau$. A further modification of this method is also possible. The modification involves finding, consecutively by each ant, the whole path in each iteration, and then updating the pheromone trail on these paths.

4.2 Finding paths according to the time list

Here, a time list is used for the purpose of recording times (durations) in which an ant reaches a given vertex. This list, in addition to putting order to ants grouping, has also additional remarkable property — an ant that reaches the target vertex earlier will be able to lay down pheromones on the same path on which it arrived quicker. In this way it is possible to increase the pheromone trail on the edges of this path earlier than on the edges of longer paths, which makes remaining ants more inclined in the following iterations to choose this path. Evaporation of pheromones occurs with every transition to the next ant on the list. Updating of the pheromone trail proceeds in turn in steps. After reaching the end vertex, each ant recedes on the path it came to reach the vertex and on each of its edges leaves $\Delta\tau$ pheromones. This strategy makes it possible to modify the process of choice of the next vertex far more earlier also for the ants that have not yet reached the end vertex. This method introduces, however, certain redundancy resulting from the use of the structure that sorts out ants according to times (durations) of reaching the next vertex.

4.3 Observations related to the method

Depending on the character of a scenario to be solved and its anticipated characteristics, it is necessary to choose an appropriate method for finding paths and the parameters

that would make it possible to seek the best solution more effectively and with better accuracy. After a comparison of the presented methods the following conclusions should be highlighted:

- In search for paths step by step we favour paths that consist of a lower number of edges because such paths will be found after a lower number of iterations, and what follows will enhance the pheromone trail on them quicker. In such a case it is required to apply an additional rule that concerns pheromone trails left on the paths that have been found, e.g. making this trail dependent on the ratio between the length of a path and the length of the shortest path found so far, or leaving pheromones only on selected paths.

- The use of the time list induces that laying pheromones requires as many iterations as the number of edges on the path. In the case of the fist method, updating of pheromones takes place in the same iteration as the reaching of the target vertex. While comparing these two methods it may seem that in the case of the time list iterations are unnecessarily dedicated to ants that get marched over the path in opposite direction.

- The redundancy resulting from the usage of the time list can be compensated by a stronger reinforcement of short paths and better adjustment, with regard to certain respects, of the state of pheromones to the currently held information on the paths.

5 Choice of the next vertex

During the choice process of the next vertex the formula presented in [3] can be used. The formula defines the transition probability of ant k that is in vertex i in its transition to vertex j. Its general form is presented in (2).

$$p_{ij}^{k} = \frac{q_{ij}}{\sum_{\{l:(i,l)\in\mathcal{E}\}} q_{il}} \tag{2}$$

5.1 Parameters influencing the choice

The choice of the next vertex is considerably influenced by the parameters α and β that regulate the reinforcement of pheromones and additional data at the calculation of the coefficient of a given edge q_{ij}. Additional data is understood to be all the information that is either available a priori or is gathered in the course of the performance of the algorithm regardless on the pheromone trail. These can be weights of edges or a marking whether a given edge has been already visited. The edge coefficient can be calculated using many methods. Some exemplary methods are presented in Equation (3).

$$q_{ij} = \tau_{ij}^{\alpha}\eta^{\beta} \tag{3a}$$

$$q_{ij} = \tau_{ij}\alpha + (1-\alpha)\eta \tag{3b}$$

$$q_{ij} = \begin{cases} \tau_{ij}^{\alpha}(1+\beta), & \text{for } vi\_nodes_j = false \\ \tau_{ij}^{\alpha}, & \text{elsewhere} \end{cases} \tag{3c}$$

$$q_{ij} = \begin{cases} \tau_{ij}^{\alpha}(1+\beta), & \text{for } vi\_edges_{ij} = false \\ \tau_{ij}^{\alpha}, & \text{elsewhere} \end{cases} \tag{3d}$$

$$\mathrm{q_{ij}} = \begin{cases} \tau_{ij}^{\alpha}(1+\beta)^2 & \text{for } \begin{cases} vi\_nodes_j = false \\ vi\_edges_{ij} = false \end{cases} \\ \tau_{ij}^{\alpha}(1+\beta) & \text{for } \begin{cases} vi\_nodes_j = true \\ vi\_edges_{ij} = false \end{cases} \\ \tau_{ij}^{\alpha}(1+\beta) & \text{for } \begin{cases} vi\_nodes_j = false \\ vi\_edges_{ij} = true \end{cases} \\ \tau_{ij}^{\alpha} & \text{elsewhere} \end{cases} \tag{3e}$$

The choice of a method for the calculation of the edge coefficient is absolutely vital for the operation of the whole algorithm. Depending on the applied formula, we can reach a quicker convergence of the algorithm (not necessarily for the most optimal solution), an increase in the

Function SelectNextEdge(i, k) of ShortestPathACO

Data: current vertex i, ant k
Result: selected edge (i, j)

```
if {j : (i, j) ∈ E} = ∅ then
    return NULL
else
    best ← −1
    result ← NULL
    foreach (i, j) ∈ E do
        // Has vertex j been visited by ant k
        if GetVisited(k, j) = false then
            // Calculate the edge coefficient (i, j) for ant k
            current ← ComputeCoefficient((i, j), k)
            if current > best then
                best ← current
                result ← (i, j)
                // Generate the real number from within the interval from 0 to 1
            else if current = best and RandReal(0, 1) > 0.5 then
                result ← (i, j)
            end
        end
    end
    return result
end
```

Function ComputeCoefficient((i, j), k) of ShortestPathACO

Data: edge under consideration (i, j), ant k
Result: edge coefficient q_{ij}

```
if τ_ij = 0 then
    return q'_ij
else
    if GetRun(k) < 3 then    // Retrieve the route number for ant k
        return 0
    else
        return q_ij
    end
end
```

exploration features of ants, or assertion as to the check of all edges and vertices. It is also worthwhile to stress that the coefficient should be concordant with and adjusted to a given problem, i.e., an equation that would secure good properties of the algorithm for one problem can be completely misleading for another one.

5.2 Optimization of the process of choice

One of the equations (3) is not enough to successfully select the next vertex for the ant. In order to increase the chances of finding better paths for ants, a pseudo-random generator can be used that will introduce randomness to the process of edge selection. The pseudo-random generator is also necessary in case uniform (equal) probabilities for a higher number of edges appear. When this is the case, an edge to be followed by the ant should be selected randomly, which will make it possible to avoid a situation where the algorithm, when activated, chooses the same edges again and again. Randomness of operations of ants is very frequently a factor that favourably influences the obtained results. Another idea worth mentioning here is to apply an individual method for the selection of edge as early as the initial stage of the operation of the algorithm. This will make it possible to adjust the performance of the algorithm to a specific problem and successfully prevents the algorithm from being, for example, greedy, or to additionally reinforce randomness of ants in the choice of an edge. The basic version of the method for selection of the next vertex is presented with the help of the function 5.1. This method uses the edge coefficient q_{ij} only, because calculations of the probability p_{ij} in each of the paths is not necessary in the process of the selection of the best edge and requires additional computation.

5.3 Calculation of the edge coefficient

The method for calculation of the coefficient q_{ij} is depicted by the function 5.2. The coefficient is calculated in one of three ways depending on whether τ_{ij} of the edge under consideration is equal to 0 and whether ant k has already found 3 paths. If τ_{ij} of a given path has not been yet increased, we use one of the formulas from Equation (4). Interestingly enough, this equation changes the meaning of the parameter α adopted earlier — it is a purposeful operation that aims at limiting the number of parameters of the algorithm. Instead of introducing a new parameter, an already available parameter is used, though incompatibly with its original purpose. The parameter α is used here as

the power of the length of edge or its inverse, while the parameter β retains its original application and influences additional data — in this particular case it is the marking whether a given vertex or edge have been already visited.

$$q'_{ij} = a_{ij}^{\alpha} \tag{4a}$$

$$q'_{ij} = \left(\frac{1}{a_{ij}}\right)^{\alpha} \tag{4b}$$

$$q'_{ij} = \begin{cases} a_{ij}^{\alpha}(1+\beta)^2 & \text{for } \begin{cases} vi\_nodes_j = false \\ vi\_edges_{ij} = false \end{cases} \\ a_{ij}^{\alpha}(1+\beta) & \text{for } \begin{cases} vi\_nodes_j = true \\ vi\_edges_{ij} = false \end{cases} \\ a_{ij}^{\alpha}(1+\beta) & \text{for } \begin{cases} vi\_nodes_j = false \\ vi\_edges_{ij} = true \end{cases} \\ a_{ij}^{\alpha} & \text{elsewhere} \end{cases} \tag{4c}$$

$$q'_{ij} = \begin{cases} \left(\frac{1}{a_{ij}}\right)^{\alpha}(1+\beta)^2 & \text{for } \begin{cases} vi\_nodes_j = false \\ vi\_edges_{ij} = false \end{cases} \\ \left(\frac{1}{a_{ij}}\right)^{\alpha}(1+\beta) & \text{for } \begin{cases} vi\_nodes_j = true \\ vi\_edges_{ij} = false \end{cases} \\ \left(\frac{1}{a_{ij}}\right)^{\alpha}(1+\beta) & \text{for } \begin{cases} vi\_nodes_j = false \\ vi\_edges_{ij} = true \end{cases} \\ \left(\frac{1}{a_{ij}}\right)^{\alpha} & \text{elsewhere} \end{cases} \tag{4d}$$

If the pheromone trail has already been deposited on the edge under consideration, then again we have a choice of two options. In the case when an ant for which we choose the next edge has not found yet three paths, we return 0. This causes the algorithm to perform randomly in the second stage of its operation, the first stage being understood as the selection of edges on the basis of their lengths or inverses of these lengths. This operation is necessary if we are to solve a highly complex graph (i.e., one that has many edges) or when the number of ants is relatively small as compared to the size of a graph. Otherwise, this stage can be omitted. It is only for an ant that is trying to find the fourth or any successive number of path that we use one of the basic formulas for the edge coefficient presented in Equation (3).

6 Updating pheromone trail

The way of updating the pheromone trail is somehow related to a method for finding paths, but it is possible to combine characteristic features of different methods to obtain new strategies. These ways can be grouped into the following categories:

in steps and progressive – pheromone trail is reinforced with every passage of the ant from one vertex to another during its search of a path leading to the end vertex

in steps and backward (reverse) – reinforcement is introduced during a change of a vertex by the ant, but following a way back from the end vertex to the initial vertex (in an analogy with real life ants, it is tantamount to a return of an ant to the nest)

overall – reinforcement is done in one iteration (the whole path followed by an ant is reinforced in one go)

selective – only a certain subset of the paths or one particular and defined path is reinforced, e.g. paths with a weight not exceeding a given pre-defined threshold, a path that is currently the best or the best path in a given iteration

Depositing pheromones during the process of finding the end vertex, i.e., concurrent enhancement of the vertex during the relocation of ants along the edges of a graph, is the most intuitive way of its upgrading. This process is the one that resembles most the real behaviour of ants. Such a strategy may turn out, however, to be of little effect as a path has has not been found yet is reinforced, while it may turn out that the path is far from being optimal. Due to the fact that the pheromone trail deposited by ants influences considerably decisions made by other ants on the move, in this particular example the algorithm may terminate its operation, while the first encountered path may be yielded

as the result. This path, on account of the character of the strategy for pheromone trail reinforcement, will be chosen in successive iterations by an increasing number of ants until the algorithm reaches convergence.

A good method to counteract risk of the occurrence of the above situation is to reinforce the pheromone trail as late as the moment at which the whole of a path is generated by an ant. At this point, we have precise and accurate information on the quality of the path and we are in position to appropriately apportion the amount of pheromones to be deposited. Moreover, if we have knowledge on the currently best solution we can make the pheromone trail dependable on it, additionally increasing its amount deposited by an ant on its way back to the initial vertex. The characteristic feature of this strategy is making the pheromone trail dependant on the length of the path. This is, however, done at the expense of the time (iteration) spent on updating.

It may be so that the information on the state of pheromones on individual edges of a graph has to be updated very quickly and the earlier strategy is too slow for this case. To retain some of its advantages, such as the knowledge on the quality of the generated solution (the weight of the path found by an ant), we can, after reaching the end vertex, immediately update the whole path instead of doing it in a series of steps. The state of pheromones is quicker adjusted to the information currently held, whereas iterations related to a reconstruction of the ant's path is omitted. Such an approach, however, levels off additional reinforcing of good solutions because with the application of the time list the reaching time of the end vertex and the return to the initial vertex with short paths is decidedly shorter than with long paths, and a quicker updating of the pheromone trail means a higher probability of selection of given edges by other ants.

As it happens, however, all the above strategies may bring no satisfactory results. When this is the case, it is worthwhile then to consider reinforcing only a given subset of paths obtained by ants. For example, we can reinforce only $\frac{m}{2}$ best paths in a given iteration, choose the best path in this iteration, or reinforce only those paths that are better than the best path at the initial stage of the operation of the algorithm. The latter method, if not safeguarded by appropriate conditions, can eventually lead to a too early convergence of the algorithm. Another thing worth mentioning here is that only in the case of finding whole paths for all ants one by one in each iteration, we are in position to somehow compare them. With the application of the time list of a path for all ants, they will be available as late as when an ant that has chosen the worst path finally reaches the end vertex, and at that time the remaining ants will be on their way back to the initial vertex or even involved in the next route. At this moment it is already too late to update trails on the edges because ants that are embarking on their successive routes have information identical to that available at the beginning of the operation of the algorithm or just few iterations earlier. On the other hand, for those ants that have found their paths to wait until the remaining ants find their routes may turn out to be disadvantageous in view of the very idea of the operation of the algorithm — ants should operate independently and communicate only with the help of pheromones. An iterative choice of the edge for each ant one after another is followed by similar consequences, with the difference that paths with a lower number of edges are in this case more advantageous from the point of view of the operation of the algorithm.

The above considerations show how complicated is the task of adjusting (matching) a method for finding paths to an appropriate way of updating edges. It is undoubtedly necessary to apportion the required amount of time for experiments that will illustrate which combinations perform better or best for a problem under consideration. Obviously, it may turn out that finding a good combination is not viable and the ant colony metaheuristic will not prove to be of use in solving a given problem.

Updating of the pheromone trail can be also dependent on the length of the performance of the algorithm and, along with its increase, can undergo changes. A use of a variant reinforcing policy in initial iterations may prove to be a good solution with the case of a situation in which the convergence of the algorithm ensues too quickly or the algorithm generates sub-optimal solutions. At the same time, it is also important to control transition periods and do not let them be too long as this can lead to unnecessary prolongation of the operating time of the algorithm or to a situation in which the algorithm remains in unchanged state.

$$\Delta\tau = const \tag{5a}$$

$$\Delta\tau = \frac{1}{a_P} \tag{5b}$$

$$\Delta\tau = \frac{C}{a_P} \tag{5c}$$

$$\Delta\tau = \frac{a_{P_{best}}}{a_P} \tag{5d}$$

where:

$$C = \max_{(i,j)\in\mathcal{E}} a_{ij} \qquad a_P = \sum_{(i,j)\in P} a_{ij}$$

Equations (5) illustrate possible ways of the calculation of the number of pheromones that are to be deposited on a found path. C is the maximum cost a_{ij} of the edge $(i,j) \in \mathcal{E}$, P denotes the path found by the ant, a_P its weight, and $a_{P_{best}}$ is the weight of the best path found so far.

The choice of $\Delta\tau$ is also very important and makes it possible to control the operation of ants because it directly influences decisions made by them. The better selected $\Delta\tau$, the quicker the algorithm reaches convergence and the higher probability of avoiding invalid results.

6.1 Additional reinforcement of currently optimal paths

If the algorithm returns non-optimal results it is worthwhile to attempt to improve its performance by reinforcing additionally good solutions. One of the applicable ways is to add a given number of pheromones to $\Delta\tau$ that has to be added to the edge of a path that has been found by the ant. The number itself, just like $\Delta\tau$, can be fixed or may depend on the length of a given path or on the length of the best path hitherto found. Another way is to multiply $\Delta\tau$ by the number of ants m, which effects in the edges of a given path to have a higher number of pheromones, regardless of the number of paths found by other ants. This can, however, lead to a situation in which a reinforced path will not be optimal, which in turn can result in a premature termination of the algorithm and the algorithm returning invalid result. In general, just like in the case of other methods, caution and appropriate experience are advisable.

Function Evaporate of ShortestPathACO

Data: all edges (i,j), pheromone level so far τ_{ij}, parameter ρ, minimum acceptable pheromone level τ_{min}
Result: new level of pheromone τ_{ij}

```
foreach (i,j) ∈ E do
    τ_ij ← τ_ij(1 − ρ)
    if τ_ij < τ_min then τ_ij ← τ_min
end
```

6.2 Evaporation of pheromones

The mechanism for evaporation of pheromones itself is easy and intuitive — the value of pheromones on each of the edges is multiplied by an appropriate parameter. This is presented with the help of the function 6.2. In addition, pheromone values on edges cannot be lower than the level determined by the parameter τ_{min}. If it happens, it is set on the value of this parameter. Such an approach is drawn from $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [4, 5]. A more difficult problem, however, is the selection of frequency, the moment of evaporation of pheromones and the value of the parameter ρ.

It is adopted that the most appropriate moment for evaporation of pheromones is most frequently at the end

of a current iteration, i.e., the moment when all remaining actions have been completed. However, it should be considered if this process should proceed after all these actions have been completed by a single ant or the whole of the colony. Combined with frequency of pheromone evaporation, the following instances can be established:

1. evaporation after the step of an ant
2. evaporation after the step of all ants
3. evaporation after finding a path by ant
4. evaporation after finding paths by all ants
5. evaporation after a time change
6. evaporation per time unit

As it is easy observable, some instances depend on a method for finding paths and can be applied only with the application of one of them. The two earlier presented cases require the application of the time list, but this method also includes applications of the remaining instances.

Evaporation after each step of an ant can be applied with both methods for finding paths presented in Section 4, it may, however, turn out to be too frequent with these modes of operation. The second and the forth case prove better with finding paths for each ants one by one because it is easier then to determine this particular moment without additional calculations. On the other hand, the fifth instance can be followed by evaporation that is too low, which is in turn levelled off in the sixth instance.

The value of the parameter ρ makes it possible to control the speed at which pheromones evaporate. The higher its value, the quicker pheromones are evaporated, while for $\rho = 0$ evaporation does not take place at all. In the case of some of the problems, a choice of this parameter is absolutely vital for the operation of this algorithm. Its too high value can result in a convergence to non-optimal solutions, whereas its too low value to a complete lack of convergence of the algorithm.

7 Termination of the algorithm

While taking into consideration the fact that the discussed algorithm is an optimization algorithm and does not guarantee finding a solution that would be optimal, its termination can be made dependent on various and diverse factors.

The best condition for termination is its convergence, which occurs when all ants from the colony find the same solution. This situation takes place when pheromones that indicate this solution have so high value that the successive iterations of the algorithm bring no further changes. In the case of a great number of ants, reaching convergence can take much time, hence a termination of the operation of the algorithm is possible if a pre-defined fraction of ants consecutively yields identical solution. This provides a chance to shorten the operation of the algorithm at the expense of an increase in getting a wrong (invalid) solution. Having this in mind, to terminate the operation of the algorithm we have to select the percentage of ants that has to reach the same solution appropriately. In this way, the end task condition will be adjusted to the solution under consideration.

In the case of improperly selected parameters of the algorithm or when the algorithm is not suitable for a given problem, it is worthwhile to introduce additional task end conditions. Most frequently, it is enough to introduce a limit/boundary to the iteration that the algorithm is to perform or to introduce a time limit for its performance. Both methods provide an efficient mechanism to avoid unnecessary infinite repetition of the procedure that, in consequence, may translate into lowering of the quality of obtained solutions.

In practice, the best way is to attempt to apply skilfully all of the methods discussed above to guarantee an acceptable running time of the algorithm, with given required credibility level of results. A selection of task end conditions for the algorithm will make it possible to evaluate

its usefulness for a given problem and to proceed with optimization initiatives or to make a decision on finding a necessary alternative method.

8 Conclusions

Algorithms based on the metaheuristic of ant colony do not guarantee finding an optimal solution in all possible cases. Accordingly, experimentation is particularly important to find and select parameters dedicated to each of the problems under consideration. Individual elements of the procedures applied in the process should be also analysed with regard to their usability and purposefulness of application. The construction of the presented ShortestPathACO algorithm directly reflects this particular approach through the proposal and the discussion of various variants of the execution of each of the elements of the procedure. In this way, it is possible to improve the method for the solution of the shortest path problem to approach or reach optimal solutions. An evaluation of the duration time and the quality of returned solutions will provide information for making a decision on the implementation of a given scheme as being of optimum quality or an alternative to more time-consuming procedures or procedures with higher computational cost.

References

[1] M. Dorigo, Optimization, Learning and Natural Algorithms, Ph.D. Thesis, Politecnico di Milano, 1992.

[2] M. Dorigo and T. Stützle, Ant Colony Optimization, The MIT Press, Cambridge, 2004.

[3] M. Dorigo, V. Maniezzo and A. Colorni, The Ant System: Optimization by a colony of cooperating agents, in *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1):29–41, 1996.

[4] T. Stützle and H. H. Hoos, The $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and Local Search for the Traveling Salesman Problem, in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 309–314, 1997.

[5] T. Stützle and H. H. Hoos, $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System, in *Future Generation Computer Systems*, 16(8):889–914, 2000.