DOI: 10.2298/CSIS110927038J

Ant Colony Optimization Algorithm with Pheromone Correction Strategy for the Minimum Connected Dominating Set Problem

Raka Jovanovic[1] and Milan Tuba[2]

[1] Texas AM University at Qatar
PO Box 23874, Doha, Qatar
rakabog@yahoo.com
[2] Megatrend University Belgrade, Faculty of Computer Science
Bulevar umetnosti 29, N. Belgrade, Serbia
tuba@ieee.org

Abstract. In this paper an ant colony optimization (ACO) algorithm for the minimum connected dominating set problem (MCDSP) is presented. The MCDSP become increasingly important in recent years due to its applicability to the mobile ad hoc networks (MANETs) and sensor grids. We have implemented a one-step ACO algorithm based on a known simple greedy algorithm that has a significant drawback of being easily trapped in local optima. We have shown that by adding a pheromone correction strategy and dedicating special attention to the initial condition of the ACO algorithm this negative effect can be avoided. Using this approach it is possible to achieve good results without using the complex two-step ACO algorithm previously developed. We have tested our method on standard benchmark data and shown that it is competitive to the existing algorithms.

Keywords: Ant colony optimization (ACO), Minimum connected dominating set problem, Swarm intelligence, Optimization metaheuristics

1. Introduction

A dominating set for a graph $G(V, E)$ is a subset of vertexes $D \subseteq V$ that has a property that every vertex in G either belongs to D or is adjacent to a vertex in D. Finding the dominating set with the smallest possible cardinality among all dominating sets for a graph is one of the standard NP-complete problems. A very important variation of the minimum dominating set problem is its connected version. We call a dominating set connected if it has the property that any node $n \in D$ can reach any other node $m \in D$ by a path that stays entirely within D. That is, D induces a connected subgraph of G. The minimum connected dominating set is the one with the minimum number of vertexes. The minimum connected dominating set problem (MCDSP) is also NP-complete.

This research was supported by Ministry of education and science of Republic of Serbia, Grant III-44006.

The MCDSP has gained popularity due to its close connection to the mobile ad hoc networks (MANETs) and sensor grids. In practical problems that can be transformed to the MCDSP there is usually no need to get the optimal solution, near-optimal solutions are sufficient in most cases.

In this paper we introduce an improved ACO algorithm for the MCDSP. The rest of the paper is organized as follows. In the next section we present different approaches to the MCDSP. In the third section a greedy algorithm for solving the MCDSP is introduced. In the fourth section we present the implementation of the ACO for the MCDSP. In the fifth section we explain our approach to avoid stagnation in ACO using a pheromone correction strategy and our method of selecting the initial vertexes. In the last section, we analyze and compare the use of pure ACO and its combination with pheromone correction on standard benchmark problems and generated examples for the MCDSP.

2. Minimum Connected Dominating Set Problem (MCDSP)

Different methods have been developed to find near-optimal solutions for the MCDSP. There are two main directions in developing algorithms for solving this problem: centralized and distributed, each of them closely connected with the type of application they are used for. In this article we focus on centralized algorithms.

Several heuristics and appropriate greedy algorithms have been developed for the MCDSP. Some of them are one-step [25] or two-step [6], [22], [12] growing techniques, or pruning-based greedy algorithms [4], [5]. A multi-step collaborative cover heuristic approach has been presented in [23]. The MSDSP has also been solved using a combination of simulated annealing and taboo search [24], neural networks [13] and parameterized approximation [10].

The ant colony optimization (ACO) is a meta heuristic that has been developed by Doringo for the traveling salesman problem [9]. ACO and other evolutionary algorithms have been proven to be effective on a wide range of combinatorial and continuous optimization problems [1], [19], [3], [2], [27]. Previously, ACO has been applied to the MDSP with great success [14], also on its weighted version [17]. For implementation of a network cluster presented as a MCDSP, a two step ACO approach was used [31]. As the first step a dominating set is created and next, as the second step, new vertexes are added to make it connected. The effectiveness of the ACO has been improved by use of different types of hybridization, like combining ACO with GA [20], [18] or differential evolution (DE) [32].

In this article we present an implementation of the ACO algorithm for the MCDSP. In our ACO implementation, we use a one-step approach applying the heuristic proposed by Guha and Khuller [12]. This approach was avoided in article [31] because of fear of early trapping in local optima and a more complex one was chosen. We propose to overcome this problem by introducing a method for avoiding early stagnation. We use a pheromone correction strategy (PCS), similar to the one used in our article [15], to direct the ant colony to

areas were good solutions are more likely. The idea of this approach is to update the pheromone trail used in ACO based on a heuristic that determines the desirability of vertexes in the solution, depending on the properties of the currently best found solution. We further improve the effectiveness of this method by implementing a good procedure for setting initial conditions of the algorithm. In our tests we show that our method is a good choice compared to existing methods and that the use of ACO combined with pheromone correction strategy has significantly better performance than the standard max-min ant system (MMAS) [26] version of ACO for this problem.

3. Greedy Algorithms for the MCDSP

There are two possible approaches to create a greedy algorithm for the MCDSP. The first one is to use a one-step approach in which the solution is constructed using only one heuristic. Another approach is represented by two-step greedy algorithms. In that case an intermediate problem like the MDSP or the maximal independent set is solved first, and at the second stage obtained solution is converted to the solution for the MCDSP as in articles [12], [6], [22]. Two-step methods usually give better results, but at the cost of being more complex for implementation. The improved results are a consequence of less constrained selection of new vertexes at the first stage. Using this type of algorithm as a base for ACO does not come natural since a separate ACO has to be developed for each stage of the algorithm.

Because of the problems mentioned, we propose a one-step greedy algorithm as a base for our ACO implementation. We have chosen to use the first greedy algorithm given by Guha and Khuller [12]. The idea of this approach is the following: we start with an initial vertex $v_0 \in V$ with the highest degree. The degree of a vertex v is the number of edges that v is incident to. Now, v_0 is the root of the tree T. At each step we pick a vertex w, which is a neighbor of some vertex v in T, that covers the highest number of uncovered vertexes. We call a vertex v covered if $v \in T$, or there exists vertex $w \in T$ for which $(v, w) \in E$. We repeat this process until all vertexes in G are covered.

To implement this greedy algorithm we need to be able to easily distinguish between neighboring, covered and uncovered vertexes. We accomplish this by using the following process. Initially all vertexes are colored white. When a new vertex is added to T it is colored black. We mark all its neighbors that are not already in T with the gray color. In the next step we select a gray colored vertex that is connected to the highest number of white vertexes. The algorithm is finished when all of the vertexes have been colored. An illustration of this algorithm is given in Fig. 1.

As noticed by Guha and Khuller [12], this type of heuristic for the greedy algorithm is easily trapped in local optimal solutions due to its short-sightedness. Because of this, more complicated algorithms have been created. Guha and Khuller have used the same approach, but instead of using single vertexes, they used pairs of them. In article [25] a heuristic that tracks the number of

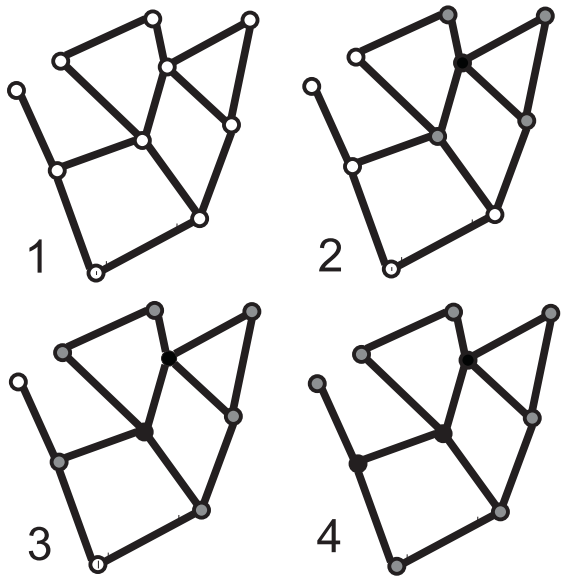

Fig. 1. Example of creating a connected dominating set using the greedy heuristic: 1) Input graph, 2) Initial step 3,4) Further steps in the algorithm

black and gray vertexes, and the number of separate black sections is used in the greedy algorithm. In [4] and [5] a greedy pruning-based approach is used where the least important vertex is removed from the dominating set. All these algorithms have a more complicated and slower implementation. We show that shortcomings of the mentioned first simple heuristic are greatly reduced when it is combined with ACO and our improvements.

4. Implementation of ACO for the MCDSP

In the ACO implementation for the MCDSP there are significant differences compared to its implementation for the traveling salesman problem (TSP). In the case of TSP the solution is a permutation of the set of all the cities; contrary to this for the MCDSP the solution is a subset of the set of graph vertexes where the order is unimportant. The heuristic function for the TSP is static because it represents the distance between cities. For the MCDSP the heuristic function is the number of white neighbors (not yet covered), which is dynamic because more vertexes are marked black or gray as new vertexes are added to the solution subset. Finally, in the case of TSP at each step all the non visited vertexes are potentially selected, while in the case of MCDSP only the vertexes marked gray are considered. These three differences affect the basic algorithm in the following way: the ants leave the pheromone on vertexes instead of edges, the heuristic function is dynamically updated and potential candidates have to be

tracked. Such variant of ACO with dynamic heuristic and a solution that consists of a subset instead of a permutation have also been used for solving the set partitioning [7], minimum vertex cover [15], set covering [21] and maximum clique [11] problems.

When implementing ACO, we first need to represent the problem in a way that makes simple the dynamic calculation of the heuristic function. This can be done in the following way. Initially, for each vertex i the value of the heuristic function η_i^0 is it's degree, or in other words, the number of connections that it has. Three sets are then created: white W^0 that initially holds all the vertexes and two empty sets, B^0 for black and Gr^0 for gray vertexes. As mentioned before, the heuristic is dynamic and it has to be updated as new vertexes are added to the result set. If at step j vertex v is added, all it's neighbors have their degree decreased by one giving the new heuristic function η^j. At this step we also move vertex v from the Gr^j to B^j, and all its neighbors from W^j to Gr^j.

To define ACO algorithm for a problem, three parts need to be defined: ant transition rule, global update rule and local update rule. We start by defining the transition rule using heuristic function η^i in the following equation:

$$p_j^k = \begin{cases} 0 & , j \notin Gr_k \\ prob_j^k & , j \in Gr_k \end{cases} \tag{1}$$

$$prob_j^k = \begin{cases} 1 & , q > q_0 \ \ \& \ \ j = \arg\max\limits_{i \in Gr_k} \tau_i \eta_i^k \\ 0 & , q > q_0 \ \ \& \ \ j \neq \arg\max\limits_{i \in Gr_k} \tau_i \eta_i^k \\ \frac{\tau_j \eta_j^k}{\sum_{i \in Gr^k} \tau_j \eta_i^k} & , q \leq q_0 \end{cases} \tag{2}$$

In Equation (2) parameter q_0 is used to define exploitation/exploration rate. Connected to it, q is a random variable upon which the next selection depends. Unlike the TSP transition rule, the selection does not depend on which vertex was added last to the current solution, but only on the current state of the graph. That is why τ_i is used instead of τ_{ij} for pheromone trail, and η_i^k instead of η_{ij} for the heuristic function. To fully specify the ACO algorithm, it remains to define the global (when ants finish their paths) and the local (when an ant chooses a new vertex) update rules.

$$\Delta\tau_i = \begin{cases} 0 & , i \notin V' \\ \frac{1}{|V'|} & , i \in V' \end{cases} \tag{3}$$

In Equation (3) $\Delta\tau_i$ is quality measure of the best global solution subset V' that contains vertex i ($|V'|$ is the number of vertexes in V'). It is used when the global update rule in Equation (4) is defined. Parameter p is used to set the influence of a newly found solution on the pheromone trail.

$$\tau_i = (1 - p)\tau_i + \Delta\tau_i \tag{4}$$

We wish to emphasis that $\Delta\tau_i$ is equal to zero for most of the vertexes, which means that the pheromone will be falling to zero for points that are not part of the global best solution.

The formula for the local update rule has the standard form

$$\tau_i = (1 - \varphi)\tau_i + \varphi\tau_0 \tag{5}$$

The quality measure of the solution acquired by the greedy algorithm (where the vertex with the best ratio of vertex degree and weight is selected) is taken for the value of τ_0. Parameter φ is used to specify the strength of the local update rule.

5. Avoiding Stagnation in ACO for the MCDSP

When ACO algorithm with the heuristic approach given by Guha and Khuller [12] is used for the MCDSP, there is a strong possibility of getting trapped in local optima. There are two main reasons for this. The first one is that this is a standard problem with ACO due to the way the pheromone matrix is created. The second one is induced by the way Guha and Khuller's greedy algorithm, which is a base for ACO implementation, works where the initially selected vertex has a very strong influence on the final result.

5.1. Pheromone Correction Strategy

We first focus on a way to avoid the problems caused by updating of the pheromone matrix. The basic approach to avoid stagnation in ACO is to use the MMAS version of ACO, in which an extra constraint is added which requires that all pheromone values are bounded, $\tau_i \in [\tau_{\min}, \tau_{\max}]$. In our case this is very important because our update rule can lower the minimum value of the pheromone very close to zero and inflicted vertexes will practically never be selected. The problem with MMAS is that for keeping the search greedy enough $\tau_{\min}$ has to be very small but the search will never be intensified after the pheromone for a vertex has reached $\tau_{\min}$.

Another interesting approach is combing ACO with the minimum pheromone threshold strategy (MPTS) as proposed in article [29]. The idea of the MPTS is to intensify search around vertexes that have been rarely selected. This is done by adding a minimum threshold value τ_{mt} that is bounded $\tau_{min} < \tau_{mt} < \tau_{max}$. In the beginning τ_{mt} is set to some initial value and then adjusted during the search, depending on the performance. Threshold τ_{mt} is used for updating the pheromone trail. When the search is conducted, values in the pheromone trail τ_i are compared to the τ_{mt} and if τ_i is lower than τ_{mt}, than τ_i is changed to some significantly higher value. In our experiments this approach proved to be efficient for small graphs, but for larger problems the search would not be greedy enough and would give results that are of lower quality than ones acquired by the MMAS version of ACO.

To improve the performance of ACO we implemented a pheromone correction strategy similar to the one used for minimum weight vertex cover problem (MWVCP) [15]. The idea of this approach is to change the pheromone

matrix by analyzing some of the properties of the best found solution. More precisely, when the search for a better solution becomes stagnant we update the pheromone matrix. We do this by using a simple heuristic function that describes the desirability of a vertex in the solution. For example, a vertex that is part of the solution and does not cover any vertexes solely by it self is not very desirable. For an undesirable vertex in the solution we greatly decrease the value of the pheromone and as a consequence, that vertex is not often chosen as a part of the solution in the following steps of the algorithm.

We have adapted this approach for the MCDSP. First, let us define $\eta(v, V')$ as the number of vertexes that vertex v, which is part of the best found solution V', solely covers.

$$Sus = \frac{1}{1 + \eta(v, V')} \tag{6}$$

In Equation (6) we have defined Sus as the undesirability of a vertex in the solution. The next step in the pheromone correction strategy is to select a random number RK of vertexes which solely cover the smallest number of vertexes. For each vertex i in the solution the probability of it being selected for pheromone correction is:

$$p_i(selected) = \frac{RK - RankSus(i, V')}{RK} \tag{7}$$

In Equation (7) instead of using the value of Sus for vertexes, we used $RankSus$ which represents their rank by undesirability . RK is the maximum number of vertexes that are considered for correction. The final step is to lower the pheromone trail for the selected vertexes:

$$\begin{array}{l} \forall i \in Selected \\ \tau_i = \delta\tau_i \end{array} \tag{8}$$

The use of $Sus(v, V')$ as a measure of desirability is not fully effective because the same group of vertexes would be repetitively selected until a better solution set was found. Because of this we introduce an improved desirability criterion:

$$CorSus(i, V') = \ Sus(i, V') \ * ExSusepect(i) \tag{9}$$

The improvement consists of tracking which vertexes have already been selected and preferring the selection of new vertexes. To do this, a new array $ExSuspect$ is introduced with elements initially set to 1. If vertex i is selected, the following correction is done:

$$\begin{array}{l} 0 < \lambda < 1 \\ ExSuspect(i) = ExSuspect(i) * \lambda \end{array} \tag{10}$$

This type of approach in which the pheromone value has been greatly decreased for some vertexes that are part of the best solution has been applied

to the MWVCP with good results [15]. The ant colony in the following steps of the algorithm avoids using these vertexes when creating new solutions. This approach however, does not give good results when extended to graph covers that also need to be connected. The problem is that when a vertex is removed, it is highly likely that it will leave the remaining vertex set disconnected. In the following steps it is hard for the ants to create a new good solution avoiding the removed vertexes due to the connectivity problem. Because of this a new type of correction is added, which is used to make it easier for new solutions to be constructed. This is done by increasing the pheromone values at vertexes that are not a part of the best found solution but are highly likely to appear in new good solution. We will consider a vertex that is not part of the solution, but covers many of the vertexes in the best solution, desirable to appear in good solutions.

Now we define a method for pheromone correction for vertexes that are not part of the best solution. First, let us define $Des(v, V')$ as the number of vertexes that are a part of the best found solution that $v \notin V'$ is connected to. The next step in the pheromone correction strategy is to select a random number RK' of vertexes which cover the greatest number of vertexes that are in the best found solution or in other words, have the greatest value of Des. For each vertex i not in the solution the probability of being selected for pheromone correction is:

$$p_i(selected) = \frac{RK' - RankDes(i, V')}{RK'} \tag{11}$$

In Equation (11) instead of using the value of Des for vertexes, we used $RankDes$ which represents their rank by desirability. RK' is the maximum number of vertexes that are considered for correction. The final step is to correct the pheromone value pheromone for the selected vertexes by increasing the value of pheromone:

$$\begin{matrix} \forall i \in Selected \\ \tau_i = \frac{\tau_{max} + \tau_{min}}{2} \end{matrix} \tag{12}$$

For vertexes for which the pheromone values will be increased we also track how often they are selected with the array $ExSuspect$ and use a new corrected desirability function $CorDes$ in the same way as for vertexes that are a part of the solution.

Finally, we need to define a stagnation criteria for recognizing if the search has been trapped in a local minimum. The criterion used is that there has been no improvement in the solution in n iterations of the ant colony. In our implementation we use separate values n_1 and n_2 for the two pheromone correction methods.

5.2. Initial Vertex Selection

When the starting point for creating an ACO algorithm for the MCDSP is Guha and Khuller's greedy algorithm, the performance is extremely influenced by the

vertex that is initially selected. This is because the solution set slowly grows from the initial vertex through its neighbors. As previously mentioned, the heuristic function is dynamic, so the previously selected vertexes not only affect the potential candidates but also which one of them will be selected. This way the initially selected vertexes have a snowball effect on the final solution. In the case of the TSP this problem is also present but it is less severe and can be solved by selecting the first vertex at random, out of all the vertexes in the graph since they all participate in the best solution. In our case this is not a good approach because only a relatively small number of vertexes are part of the best solution so the search becomes too wide. In the case of MWVCP [15] where the solution is also a small subset of V, we selected a random vertex of the best solution. However, if we choose the initial vertex for MCDSP in this way the search becomes too narrow. This is because in the case of MWVCP the previous steps only affect the heuristic function but in the case of MCDSP the candidate list is also affected.

We try to balance these two approaches in the following way:

$$InitVertex = \begin{cases} Random(V') & , s < s_0 \\ Random(V, \tau) & , s \geqslant s_0 \end{cases} \tag{13}$$

In Equation (13) s is a random variable on which the type of selection depends, s_0 is a fixed parameter that defines how often the initial vertex will be selected from the global best solution V' or from all the vertexes in V. In case it is selected from V, the probability distribution is only dependent on the pheromone trail corresponding to vertexes.

5.3. Our Improved ACO Algorithm for the MCDSP

The recapitulation of the key elements of our improved ACO algorithm for the MCDSP is:

- ACO algorithm for the MCDSP is implemented with necessary adjustments considering that for the MCDSP solution is a subset of the set of graph vertexes where the order is unimportant and that the heuristic function is dynamic. That affects the basic algorithm in a way that the ants leave the pheromone on vertexes instead of edges, the heuristic function is dynamically updated and potential candidates have to be tracked.

- The mentioned ACO algorithm is based on the first greedy algorithm given by Guha and Khuller [12]. It starts with an initial vertex $v_0 \in V$ with the highest degree as the root of the tree T. At each step a vertex w is picked, which is a neighbor of some vertex v in T, that covers the highest number of uncovered vertexes. This process is repeated until all vertexes in G are covered.

– ACO algorithm for the MCDSP based on Guha and Khuller's greedy algorithm is strongly influenced by the vertex that is initially selected because the solution set slowly grows from the initial vertex through its neighbors. We introduce modification that narrows the selection to vertexes that belong to the global best solution, but not always, according to Equation (13).

– When stagnation is detected, search has to move to, at that moment, less promising areas. Rather than using more standard method of increasing the pheromone level for vertexes that currently do not belong to the best found solution, we decrease the pheromone level for, by defined criteria, undesirable vertexes in the best found solution. This novel approach improves leaving local optima in the directions that lead to better solutions.

– The previous step, very successful for some other problems [15], creates some problems when unmodified applied to graph covers that also need to be connected. The problem is that when a vertex is removed, it is highly likely that it will leave the remaining vertex set disconnected. Because of this a new type of correction is added, which is used to make it easier for new solutions to be constructed. This is done by increasing the pheromone values at vertexes that are not a part of the best found solution but are, by defined criteria, highly likely to appear in new good solution.

The program for our experiments was written in C#, using the framework from article [16]. The program implements the following pseudo code

```
Reset Graph Info
Reset Solution for all Ants
Select Initial Vertex for all ants

 while  (! AllAntsFinished)
   for All Ants
     if(AntNotFinished)
       begin
         add new vertex A to solution based on probability
         correct ant's cover graph data
         calculate new set of candidates
         calculate new heuristic
         local update rule for A
       end
     end for
   end while

 Compute DeltaTauI
 Compute TauI
```

```
If(Iteration_NoChange % n1)
  Use CorSus for Pheromone Correction
If(Iteration_NoChange % n2)
  Use CorDes for Pheromone Correction
```

6. Test and Results

We have conducted two types of tests. In the first type we analyze the effectiveness of our method on benchmark data sets with existing solutions. In the second group of tests we generate problem instances as proposed in article [14] that correspond to ad hoc network clustering problems.

The ACO algorithm is implemented in its MMAS version. For both, ACO with and without pheromone correction, we conducted ten separate colony simulations and compared average solutions and standard deviations. All the colonies had the following parameters: $q_0 = 0.9$ specifies the exploitation/exploration rate, $p = 0.1$ and $\varphi = 0.1$ specify the global and local update rules. These are the standard values used by most authors and after some testing we decided that there is no need to change them. The value of the parameter that defines initial vertex selection is $s_0 = 0.2$. This parameter is specific for our method and was determined empirically after significant number of tests. The parameters used for our pheromone correction had the following values: coefficient for the pheromone correction $\delta = 0.0001$, the maximum number of selected vertexes $RK = \frac{|V|}{s}$ where s is a random number from the interval [2,10] and $\lambda = 0.9$. We determined these parameters in [15] and after some testing determined that the same values are appropriate for this problem. The stagnation parameters had the following values: $n_1 = 20$ and $n_2 = 40$. These values were empirically proven to balance two corrections specific for our method. In our tests we used 10 colonies for both, ACO and ACO with pheromone correction strategy. In both cases we used random seeds with values from 0 to 9.

We have tested our method on benchmark data sets that have been used on the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09) [28]. The maximum number of iterations for a colony was 350 which means that 3500 solutions have been created. We compare the quality of solutions achieved by our ACO combined with a pheromone correction strategy to standard MMAS ACO using the basic version of Guha and Khuller's heuristic, to pure greedy algorithm and to known best benchmark results from the LPNMR'09. These results are in Table 1.

In Table 1 we only give the results for problem instances that have had a satisfactory solution (the solution is known) given in the LPNMR benchmark. The best found solution for the ant colonies, which is commonly shown, does not appear in the table due to the fact that both ACO algorithms have achieved the best given solution in all the benchmark examples. We first notice that the basic greedy algorithm of Guha and Khuller performs poorly and gives the average error of 126% compared to the best solution. The MMAS variation of ACO gives results that on average have 8.5% error. This shows that the use of

Table 1. Comparison of LPNMR, Greedy 1, simple ACO and ACO combined with MPTS

Problem	LPNMR	Greedy	ACO MMAS			ACO with PCS		
Dimensions	Result		Average	St.Dev.	t	Average	St.Dev	t
40*200	5	10	5.8	0.60	3.2	5.3	0.45	4.1
45*250	5	15	5.8	0.40	3.5	5.5	0.50	4.3
50*250(1)	8	15	8.1	0.54	4.8	8.0	0.00	6.1
50*250(2)	7	17	7.5	0.50	5.0	7.1	0.30	6.5
55*250	8	20	8.8	0.98	5.6	8.3	0.45	7.3
60*400	7	15	7.0	0.00	6.1	7.0	0.00	9.1
70*250	13	32	14.2	0.74	11.0	13.9	1.04	13.5
80*500	9	20	10.0	0.44	12.1	9.8	0.40	16.9
90*600	10	19	10.9	0.83	14.0	10.6	1.01	17.3
Average	8.00	18.11	8.68			8.34		

ACO, with careful selection of the initial vertex, manages to overcome the shortsightedness of the underlying greedy method. Finally, the results that have been archived by adding the pheromone correction strategy to ACO manages to improve the results even further to have an average error of 4.2%. Standard deviation is also improved in most cases. Columns marked with t report computational times in seconds for ten runs. They should be used only for coarse comparison since they include hard disk time, no optimization of the algorithm was attempted and it was written in C#.

As an illustration of the effectiveness of this method we give a comparison with results for the problem viewed as decision problem achieved by Answer Set Programming (ASP), Propositional Satisfiability (SAT) and Constraint Programming (CP) that are given on the LPNMR'09 web site. The benchmark test set consists of 21 problems of different sizes, and for each it is requested to answer if a solution of a certain number of vertexes exists. For each of the test examples we have conducted two colony runs with a fixed number of iterations (350), and we check if any of the colonies has found a solution with the requested number of vertexes; if it has the problem is satisfied, otherwise it is not. The test have been done on similar hardware (ours slightly better): at LPNMR'09 Dell OptiPlex 745, 1 CPU with 2 cores: GenuineIntel Intel(R) Core(TM)2 CPU 6600 @ 2.40GHz 4 GB RAM, and in our case Dell OptiPlex 755, 1 CPU with 2 cores: GenuineIntel Intel(R) Core(TM)2 CPU E8500 @ 3.16GHz 3 GB RAM. The software used at LPNMR'09 was created in C++ and our application was made using C# which gives them a speed advantage. Our method had successfully solved all the problem instances and for that it needed 52 seconds. In comparison to this, the best method from LPNMR'09 has solved all the problems in 36 seconds, and the following ones needed 128, 169, 316, 465 and 535 seconds. Although the comparison is not fully accurate it still shows that our method is very competitive.

In our second group of tests we generated graphs in the same way as proposed by Chen [14]. The graphs are generated in the following way. In some fixed area $N * N$ a random number of points are selected with a uniform distribution which represent the nodes of our graph. If the distance between two nodes i and j is smaller that some value R than edge (i, j) is a part of our graph. We have generated problems with different number of nodes and different edge densities and used them to compare ACO and ACO with a pheromone correction strategy. We use the same parameters for ACO as before, except for the maximum number of iterations for a colony which is now 5000 due to the increased size of the problems. We can see the results in Table 2.

For each of the 41 test instances we compared the best found solution and the average solution for ACO and ACO combined with pheromone correction. We first wish to point out that the basic greedy algorithm performs poorly for larger problem instances. Both ACO approaches improve the minimal solution 2-3 times compared to the greedy algorithm. ACO combined with a pheromone correction strategy improved the best found solution in 18 cases and decreased its quality in only 3 cases. When the average solution is observed the addition of a pheromone correction strategy improved the result quality in 33 cases and decreased its quality in 6 cases. The advantages of using the PCS are greater in the case of small and medium problem instances. We explain this by the fact that the PCS parameter values have been chosen from analyzing the behavior of the algorithm for small problem instances. We believe that the same level of improvement can be archived with a better choice of parameters.

7. Conclusion

In this paper we have presented an ant colony optimization algorithm for the minimum connected dominating set problem. Our implementation is fast and simple one-step ACO method based on a greedy heuristic where our pheromone correction strategy and special attention to the initial condition of the ACO overcome shortcomings of that heuristic. The tests on standard benchmark data as well as on standard generated examples have shown that our algorithm generates good solutions compared to other state of the art algorithms. Moreover, the execution time is favorable compared to the results obtained on 10^th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09) benchmark data sets. This is important since such solutions are usually used in MANETs and the speed of execution is more important than optimality. We used successfully the similar strategy to improve ACO for the MWVCP and another version for the TSP so we can consider that our pheromone correction strategy is a rather general method of improving ACO. Future research may include additional tuning for larger examples and use of different pruning-based greedy heuristics as in [4], [5]. They are more complex for implementation but natural for the ACO since these are one-step algorithms that much less depend on the initial vertex selection. Some recent improvements in greedy algorithms [30], [8] can also be included in the future research.

Table 2. Comparison of ACO and ACO combined with pheromone correction on different MCDSP instances

Area(N*N) Nodes	R	Greedy	ACO		ACO + PCS	
			Min	Avg	Min	Avg
400	60	48	20.0	21.6	19	**21.2**
80	70	33	16	17.0	**15**	**16.2**
	80	35	12	14.0	12	**13.1**
	90	41	11	11.8	11	**11.6**
	100	23	8	9.0	8	**8.9**
	110	25	8	8.5	8	8.5
	120	17	7	7.5	7	**7.2**
600	80	38	23	24.7	**22**	**23.6**
100	90	40	22	23.8	**21**	**23.6**
	100	38	17	20.0	17	**19.0**
	110	35	15	17.2	15	**16.8**
	120	36	15	16.2	**14**	**15.5**
700	70	96	46	50.7	46	**49.6**
200	80	89	41	**43.7**	41	43.9
	90	84	34	36.0	**33**	**35.7**
	100	75	28	**30.8**	28	31.0
	110	70	23	27.4	**22**	**26.4**
	120	68	21	23.6	21	**23.4**
1000	100	96	46	50.7	46	**49.6**
200	110	92	43	44.9	**42**	**44.8**
	120	82	37	39.9	37	**39.8**
	130	91	32	34.7	32	**34.9**
	140	76	30	31.3	**29**	31.3
	150	83	28	29.6	**26**	**28.8**
	160	86	**24**	26.6	25	**26.5**
1500	130	158	60	64.5	60	**64.3**
250	140	144	53	57.2	**52**	**57**
	150	170	51	54.9	51	**54.4**
	160	151	47	50.5	**45**	**49.8**
2000	200	178	55	**58.6**	**52**	58.8
300	210	151	51	53.5	**50**	**52.8**
	220	140	47	48.9	**45**	**48.4**
	230	166	44	47.5	44	**46.9**
2500	200	198	79	82.0	79	**81.5**
350	210	185	75	79.1	**74**	**78.2**
	220	205	**68**	**72.6**	69	73.8
	230	193	66	69.2	66	**68.9**
3000	210	259	99	**101.6**	**98**	104.0
400	220	225	**88**	**95.4**	91	97.6
	230	205	86	91.4	86	**90.3**
	240	210	82	85.8	**80**	**84.1**

Acknowledgment. Authors thank anonymous reviewers for useful comments that helped improve the quality of this paper.

References

1. Abbaspour, R.A., Samadzadegan, F.: An evolutionary solution for multimodal shortest path problem in metropolises. Computer Science and Information Systems 7(4), 789–811 (2010)
2. Bacanin, N., Tuba, M.: Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators. Studies in Informatics and Control 21(2), 137–146 (2012)
3. Brajevic, I., Tuba, M.: An upgraded artificial bee colony algorithm (ABC) for constrained optimization problems. Journal of Intelligent Manufacturing (published Online First), DOI: 10.1007/s10845–011–0621–6 (2012)
4. Butenko, S., Cheng, X., Oliveira, C., Pardalos, P.: A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks. In: Recent Developments in Cooperative Control and Optimization. pp. 61–73. Kluwer Academic Publishers (2004)
5. Butenko, S., Oliveira, C., Pardalos, P.: A new algorithm for the minimum connected dominating set problem on ad hoc wireless networks. In: CCCT'03. pp. 39–44. International Institute of Informatics and Systematics (IIIS) (2003)
6. Cheng, X., Ding, M., Chen, D.: An approximation algorithm for connected dominating set in ad hoc networks. In: Proc. of International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks (TAWN) (2004)
7. Crawford, B., Castro, C.: Ant colonies using arc consistency techniques for the set partitioning problem. In: Professional Practice in Artificial Intelligence. pp. 295–301. Springer, Boston (2006)
8. Das, A., Mandal, C., Reade, C., Aasawat, M.: An improved greedy construction of minimum connected dominating sets in wireless networks. In: 2011 IEEE Wireless Communications and Networking Conference (WCNC). pp. 790–795. IEEE (2011)
9. Dorigo, M., Gambardella, L.M.: Ant colonies for the travelling salesman problem. Biosystems 43(2), 73–81 (July 1997)
10. Downey, R.G., Fellows, M.R., McCartin, C., Rosamond, F.A.: Parameterized approximation of dominating set problems. Information Processing Letters 109(1), 68–70 (2008)
11. Fenet, S., Solnon, C.: Searching for maximum cliques with ant colony optimization. In: Applications of Evolutionary Computing. pp. 291–302. Springer-Verlag, Berlin/Heidelberg (2003)
12. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmica 20(4), 374–387 (1998)
13. He, H., Zhu, Z., Makinen, E.: A neural network model to minimize the connected dominating set for self-configuration of wireless sensor networks. IEEE Transactions on Neural Networks 20(6), 973–982 (June 2009)
14. Ho, C.K., Singh, Y.P., Ewe, H.T.: An enhanced ant colony optimization metaheuristic for the minimum dominating set problem. Applied Artificial Intelligence 20(10), 881–903 (2006)
15. Jovanovic, R., Tuba, M.: An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. Applied Soft Computing 11(8), 5360–5366 (December 2011)

16. Jovanovic, R., Tuba, M., Simian, D.: An object-oriented framework with corresponding graphical user interface for developing ant colony optimization based algorithms. WSEAS Transactions on Computers 7(12), 1948–1957 (2008)
17. Jovanovic, R., Tuba, M., Simian, D.: Ant colony optimization applied to minimum weight dominating set problem. In: Proceedings of the 12th International conference on Automatic control, modelling and simulation. pp. 322–326. ACMOS'10, World Scientific and Engineering Academy and Society, Stevens Point, Wisconsin, USA (2010)
18. Jun-Qing Li, Q.K.P., Xie, S.X.: A hybrid variable neighborhood search algorithm for solving multi-objective flexible job shop problems. Computer Science and Information Systems 7(4), 907–930 (2010)
19. Kratica, J., Kostic, T., Tosic, D., Dugosija, D., Filipovic, V.: A genetic algorithm for the routing and carrier selection problem. Computer Science and Information Systems 9(1), 49–62 (2012)
20. Lee, Z.J., Su, S.F., Chuang, C.C., Liu, K.H.: Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment. Applied Soft Computing 8(1), 55–78 (2008)
21. Lessing, L., Dumitrescu, I., Sttzle, T.: A comparison between ACO algorithms for the set covering problem. In: LNCS 3172, Springer. pp. 1–12 (2004)
22. Min, M., Du, H., Jia, X., Huang, C.X., Huang, S.C.H., Wu, W.: Improving construction for connected dominating set with steiner tree in wireless sensor networks. Journal of Global Optimization 35(1), 111–119 (2006)
23. Misra, R., Mandal, C.: Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks. IEEE Transactions on Parallel and Distributed Systems 21(3), 292–302 (June 2010)
24. Morgan, M., Grout, V.: Metaheuristics for wireless network optimisation. In: The Third Advanced International Conference on Telecommunications, AICT 2007. p. 15 (May 2007)
25. Ruan, L., Du, H., Jia, X., Wu, W., Li, Y., Ko, K.I.: A greedy approximation for minimum connected dominating sets. Theor. Comput. Sci. 329(1-3), 325–330 (2004)
26. Stützle, T., Hoos, H.H.: Max-min ant system. Future Gener. Comput. Syst. 16(9), 889–914 (June 2000)
27. Tuba, M., Brajevic, I., Jovanovic, R.: Hybrid Seeker Optimization Algorithm for Global Optimization. Applied Mathematics and Information Sciences, 7(3), 867–875 (2013)
28. URL: The second answer set programming (asp) competition: Submitted benchmarks (2009), http://dtai.cs.kuleuven.be/events/ASP-competition/encodings.shtml
29. Wong, K.Y., See, P.C.: A new minimum pheromone threshold strategy (MPTS) for max-min ant system. Applied Soft Computing 9(3), 882–888 (June 2009)
30. Yang, D., Wang, X.: Greedy Algorithms for Minimum Connected Dominating Set Problems. In: Proceeding of the 10th International Conference on Intelligent Technologies. pp. 643–646 , Guangxi Normal Univ., Guilin, Peoples Republic of China, (December 2009)
31. Zhang, C., Xu, Q.: Clustering approach for wireless sensor networks using spatial data correlation and ant-colony optimization. In: Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing - Volume 01. pp. 538–541. IEEE Computer Society, Washington, DC, USA (2009)
32. Zhang, X., Duan, H., Jin, J.: Deaco: Hybrid ant colony optimization with differential evolution. In: IEEE Congress on Evolutionary Computation. pp. 921–927. IEEE Computer Society (2008)

Raka Jovanovic is a Ph. D. candidate at the University of Belgrade, Faculty of Mathematics where he also received B.S. and M.S. degrees in Computer Science. He worked as a research assistant/associate at the Institute of Physics, University of Belgrade and was employed as a research associate at Texas AM University at Qatar. His research interests include Optimization problems, Data compression, Image processing, Numeric simulation and Fractal imaging.

Milan Tuba is Professor of Computer Science and Provost for mathematical, natural and technical sciences at Megatrend University of Belgrade. Before that he was associate professor at Faculty of Mathematics, University of Belgrade and assistant professor of Electrical Engineering at Cooper Union, New York. He received B. S. in Mathematics, M. S. in Mathematics, M. S. in Computer Science, M. Ph. in Computer Science, Ph. D. in Computer Science from University of Belgrade and New York University. His research interest includes mathematical, queuing theory and heuristic optimizations applied to computer networks, image processing and combinatorial problems. Professor Tuba is the author of more than 100 scientific papers. He is coeditor or member of the editorial board or scientific committee of number of scientific journals and conferences. Member of the ACM since 1983, IEEE 1984, New York Academy of Sciences 1987, AMS 1995, SIAM 2009, IFNA 2012.

Received: September 22, 2011; Accepted: October 10, 2012.