An Analysis of Different Variations of Ant Colony Optimization to the Minimum Weight Vertex Cover Problem

MILAN TUBA
Faculty of Computer Science
Megatrend University Belgrade
Bulevar umetnosti 29, N. Belgrade
SERBIA
tubamilan@ptt.rs

RAKA JOVANOVIC
Institute of Physics
Belgrade
Pregrevica 118, Zemun
SERBIA
rakabog@yahoo.com

Abstract: - Ant colony optimization (ACO) has previously been applied to the Minimum Weight Vertex Cover Problem with very good results. The performance of the ACO algorithm can be improved with the use of different variations of the basic Ant Colony System algorithm, like the use of Elitism, Rank based approach and the MinMax system. In this paper, we have made an analysis of effectiveness of these variations applied to the Minimum Weight Vertex Covering Problem for different problem cases. This analysis is done by the observation of several properties of acquired solutions by these algorithms like best found solution, average solution quality, dispersion and distribution of solutions.

Key-Words: - Ant Colony, Minimum Weight Vertex Cover, Optimization Problems, Population Based Algorithms

1 Introduction

One of the classical graph theory problems is the Minimal Vertex Cover Problem. The problem is defined for an undirected graph $G = (V, E)$. V is the set of vertexes and E is a set of edges. A vertex covering of a graph is set of vertexes $V' \subset V$ that has the property that for every edge $e(v_1,v_2) \in E$ at least one of v_1,v_2 is an element of V'. A minimal vertex cover is a vertex covering that has a minimal number of vertexes. In this paper we devote our attention to an extension of this problem named the Minimum Weight Vertex Cover Problem (MWVCP) in which weights are added to the vertexes. The solution is not the vertex covering with a minimal number of vertexes but with a minimal sum of weights.

A large number of real life problems could be converted to this form. An example is the optimal positioning of garbage disposal facilities. Every area needs to be covered by a garbage disposal facility, but not all potential positions are equal. It has been shown that this problem is NP-complete even when it is restricted to a unit-weighted planar graph with the maximum vertex degree of three [1]. The vertex cover is one of the core NP-complete problems that are frequently used for proof of NP-hardness of newly established ones. In the same way as for many other NP-complete problems, finding the optimal solutions is very time consuming and, in larger problem cases, even impossible in realistic time. Because of this, varieties of different methods have been presented for calculating near optimal solutions. The first method is a greedy heuristic approach of collecting the vertex with the smallest ratio between its weight and degree [2], [3]. Degree of vertex is the number of edges that have it as a member. This problem has also been solved by the use of more complex method genetic algorithms [4].

Ant colony optimization is a meta-heuristic inspired by the behavior of ants used for solving optimization problems. Ant colonies are able to find the shortest possible path between their nest and a food source, this is done by the cooperation of the ants in the colony. Each ant starts from the nest and walks toward food. It moves until an intersection where it decides which path it will take. In the beginning it seems as a random choice but after some time the majority of ants are using the optimal path. This is achieved by using *pheromone.* Each ant deposits pheromone while walking which marks the rout taken. The amount of pheromone indicates the usage of a certain route. Pheromone trail evaporates as time passes. Due to this a shorter path will have more of it because it will have less time to evaporate before it is deposited again. The colony behaves intelligently because each ant chooses paths that have more pheromone.

Dorigo first used the simulations of ant colony for on the Traveling Salesman Problem and defined the new meta heuristic [5]. This method has been

successfully used on a wide variety of problems since then. The use of ant colony optimization (ACO) gave very good results when applied to the MWVCP. The results acquired by ACO where better that the ones calculated by genetic algorithms and local search methods like tabu search, and simulated annealing [6].

One of the methods of improving the efficiency of ACO is the use of different variations of the basic algorithm. It has been shown that for some problems variations of ACO gave results of different quality [7]. Because of this, we have decided to analyze the implementation details and effect of different Ant colony optimization algorithms like Ant Colony system, the use of Elitism, Rank Based Ant colony systems and the MinMax approach to MWVCP. In this paper we will compare several different aspects of performance of these variations.

This paper is organized as follows. In Section 2 we give an mathematical definition of the MWVCP. In Section 3 we present the implementation of ACO for this problem. In Section 4 we show the implementation details of ACO algorithm variations. In Section 5, we analyze and compare experimental results of the use of different variations of ACO on several problem instances.

2 Mathematical Definition of MWVCP

In this section we define the MWVCP in strict mathematical terms, through its integer linear programming form. Linear programming is a procedure for finding the maximum or minimum of a linear function where the arguments are subject to linear constraints. This type of formulation is a very good guideline for software implementation of problems. For the MVCP we accept the formulation that was presented in article [8]:

Let $G = (V, E)$ be an undirected graph, where

$$V = \{1, 2, .., n\}$$
$$E \subseteq \{(i, j) \mid i, j \in V\} \quad (1)$$

V is the set of vertexes and E is the set of edges. Let $m = |E|$ and $n = |V|$ be the number of edges and vertexes respectively. Let $A = n \times m$ be the vertex-edge incidence matrix of G. The cells of matrix A are defined in the following way:

$$a_{ij} = \begin{cases} 1 & \text{, if edge } j \text{ is incident to vertex } i \\ 0 & \text{, otherwise} \end{cases} \quad (2)$$

In addition, we introduce a cost function to the vertex set, $w: V \to N$, which corresponds to the weights of the initial graph G. With the given definitions, the MWVCP can now be described by the following integer linear programming formulation:

Minimize	$\sum_{i=1}^{n} w(i) x_i$
Subject to	$\sum_{i=1}^{n} a_{ij} x_i \geq 1 \ , j = 1, .., m$
	$x_i \in \{0,1\} \quad , i = 1, .., n$

3 ACO for the MWVCP

The use of ACO has proven to be effective on various types of problems from Economic Load Dispatch [9], Scheduling problems [10], even image processing[11], and its use has been also very efficient on MWVCP. The MWVCP is in two main aspects different than most of the problems solved using ACO. To illustrate this, for comparison we will use the Traveling Salesman Problem (TSP). In the TSP our solution is an array of all the cities appearing in the problem, or in other words the solution is the permutation of the set of cities. In the case of MWVCP the solution is a subset of the graph vertexes set, in which the order is unimportant. In TSP, our heuristic function is static in the sense that it represents the distance between cities and does not change during the calculation of the path. Opposite to this, for the MWVCP the heuristic function is the ratio between the weight and the degree of a vertex, which is dynamic. The degree of a vertex changes as we add new vertexes to the solution set because more edges become covered. We wish to point out that ACO with a dynamic heuristic and a solution that consists of a subset instead of a permutation have been also used for solving the set partitioning [12], maximum independent set [13] and maximum clique [14] problems.

These two differences affect the basic algorithm in two directions. First, ants leave the pheromone on vertexes instead of on edges and second, we dynamically update the graph, and with it, the heuristic function. The first step in solving these problems is representing the problem in a way that makes dynamic calculation of the heuristic function simple.

Since ants in their search can move from a vertex to any other vertex, it is natural to use a fully connected graph $G_c(V,E_c)$ derived from G. In the articles [6], [5] it is proposed to add weights to

edges in the new graph G_c. If an edge exists in G it is given the weight 1, or 0 if it does not exist in the original graph. We have adopted this approach, which is illustrated by Fig.1.

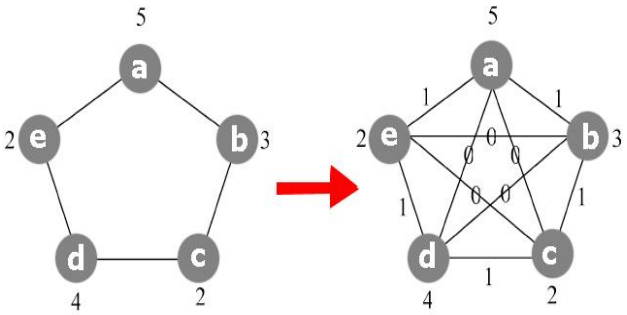

Fig. 1 Expansion to the fully connected graph

As we mentioned before we also have to update this graph as we add new vertexes to the result set. This is done using the following rule: when we add vertex *a*, weights of all edges in G_c that are connected to *a*, are set to 0. This is illustrated by Fig. 2.

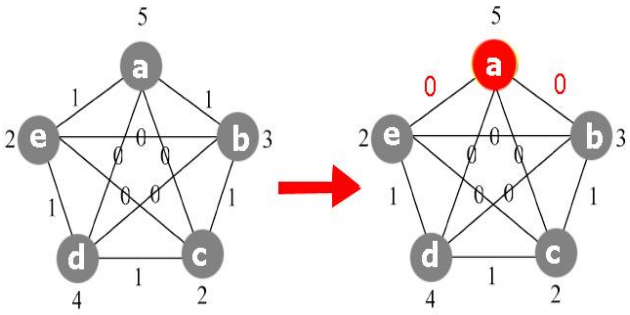

Fig. 2 Adding a vertex to the solution set

Now we can define G_k (V,E_{ck}) as the state of the graph after k vertexes have been added to the solution set, and a corresponding functions in Equation 3:

$$\psi_k(i,j) = Value(E_{ck}(i,j)) \qquad (3)$$

This update rule has two roles. First, we can dynamically evaluate the preference of vertexes with function ψ_κ. The second role is that it gives us the information when all edges have been covered, or more precisely, if the total sum of edge weights in G_k is 0 then all edges are covered. Now we can define a dynamic heuristic

$$\eta_{jk} = \frac{\sum_{(i,j)\in E_c} \psi_k(i,j)}{w(j)} \qquad (4)$$

In Equation 4 $w(j)$ is the weight of a vertex. Using the heuristic defined with η_{jk} in Equation 4 we can setup the state transition rule for ants:

$$p_j^k = \begin{cases} 1 & , q > q_0 \ \& \ j = \arg\max_{i\in A_k} \tau_i \eta_{ik}^\alpha \\ 0 & , q > q_0 \ \& \ j \neq \arg\max_{i\in A_k} \tau_i \eta_{ik}^\alpha \\ \dfrac{\tau_j \eta_{jk}^\alpha}{\sum_{i\in A_k} \tau_j \eta_{ik}^\alpha} & , q \le q_0 \end{cases} \qquad (5)$$

In Equation 5 q_0 is the standard parameter that specifies the *exploitation/exploration* rate of individual ant searches. that appears in ACO. q is a random variable that decides the type of selection on each step. A_k is a list of available vertexes. We point out that opposite to the TSP transition rule it does not depend on the last selected vertex that is why we have τ_i instead of τ_{ij}.

To fully specify an Ant Colony System we still have to define a global (when an ant finishes its path) and a local (when an ant chooses a new vertex) update rules. The role of the global update rule is to make paths creating better solutions to become more desirable, or in other words, it intensifies exploitation.

$$\triangle\tau_i = \frac{1}{\sum_{j\in V'} w(j)} \quad , \forall i \in V' \qquad (6)$$

$$\tau_i = (1-p)\tau_i + \triangle\tau_i \qquad (7)$$

Equation 6 defines the global update rule. In it $\Delta\tau_i$ is a quality measure of solution subset V' that contains vertex i, and with it we define an global update rule in Equation 5. This measure is inverse proportional to the weight of a solution. Parameter p is used to set the influence of newly found solution on the pheromone trail.

The local update rule purpose is to shuffle solutions and to prevent all ants from using very strong vertexes. The idea is to make vertexes less desirable as more ants visit them. In this way, exploration is supported. The formula for the local update rule has the standard form

$$\tau_i = (1-\varphi)\tau_i + \varphi\tau_0 \qquad (8)$$

For the value of t_0 we take the quality measure of the solution acquired with the greedy algorithm when we select the vertex with the best ration of vertex

degree and weight. Parameter φ is used to specify the strength of the local update rule.

4 Variations of ACO for MWVCP

There are different methods of improving ACO like certain types of hybridizations. Standard hybridizations are the combination of the basic algorithm with a local search [15] or some genetic algorithm [16]. These hybridizations are effective in increasing the efficiency of ACO but are often complicated for implementation. The complexity of their implementation is due to the fact that to separate algorithms need to be developed one for ACO and another for the local search or for the genetic algorithm. The other method of improving the performance of ACO is the use of different variations of the basic algorithm. On TSP different variations of ACO gave different quality of results, and no variation can be considered the best [7]. That is why we have decided to performed a comparative assessment of standard variations of ACO on MWVCP. The variations mostly differ in the global update rule. We present several variations of ACO by giving their global update rules.

Ant System (AS), is the most basic implementation of ACO, in this version of the algorithm all ants are equal and leave pheromone. It is defined with Equations 9 and 7. In Equation 7 *AntS* is the set of all the solutions created by ants in the current step of the algorithm.

$$\forall i \in \bigcup_{V_i' \in Ants} V_i'$$
$$\Delta\tau_i = \sum_{V_k' \in AntS} \frac{1}{\sum_{j \in V_k'} w(j)} \tag{9}$$

Reinforced Ant System (RAS), which is the same as Ant system, except that the global best solution is reinforced each iteration.

$$\forall i \in V'_{gb} \cup \bigcup_{V_i' \in Ants} V_i'$$
$$\Delta\tau_i = \frac{1}{\sum_{j \in V'_{gb}} w(j)} + \sum_{V_k' \in AntS} \frac{1}{\sum_{j \in V_k'} w(j)} \tag{10}$$

In some variations of this method the iteration best solution is also reinforced each iteration. With this approach the basic AS is made to be slightly greedier. It is defined with Equations 10 and 7.

Elitist Ant System (EAS), In this version of the algorithm, individual ants do not automatically leave pheromone. In each iterations step of the colony, or in other words when all the ant complete their solutions, only the global best solution will be used to update the pheromone trail. In this way, the search is even more centralized around the global best solution. It is defined with Equations 11 and 7.

$$\Delta\tau_i = \frac{1}{\sum_{j \in V'_{gb}} w(j)}, \forall i \in V'_{gb} \tag{11}$$

MinMax Ant System (MMAS) is same as the Elitist Ant colony System, but with an extra constraint that all pheromone values are bounded, $\tau_i \in [\tau_{\min}, \tau_{\max}]$. We adopt the formulas presented in article [17] in which τ_{max} is calculated dynamically as new best solutions are found by Equation 12, and τ_{min} is calculated at the beginning of calculations with Equation 13. a*vg* is the average number of vertexes that are possible to be chosen, p_{best} is the possibility of the best overall solution being found and τ_0 is the initial value of the pheromone trail gotten as the quality measure of the greedy algorithm solution.

$$\tau_{\max} = \frac{1}{(1-p)} \Delta\tau_{gb} \tag{12}$$

$$\tau_{\min} = \frac{\tau_0 (1 - \sqrt[n]{p_{best}})}{(avg - 1)\sqrt[n]{p_{best}}} \tag{13}$$

This variation has two effects that improve the effectiveness of EA. First, the pheromone trial will not become very strong on some good vertexes and making them a part of almost all newly created solutions. By giving a lower bound to the pheromone trail the potential problem of certain parts of the solution being totally excluded from the search due to very weak values of pheromone is avoided.

Rank Based Ant Colony System (RANKAS) [18] in which except the quality we also use the rank (*R*) of found solutions. Rank is defined by the quality of the solution compared to solutions found by other ants in the same iteration. It is defined with Equations 7 and 14:

$$BRank = \{V \mid (R(V) < RK) \wedge (V \in AntS)\}$$
$$\forall i \in V'_{gb} \cup \bigcup_{V'_i \in BRank} V'_i$$
$$\Delta\tau_i = \frac{1}{\sum_{j \in V'_{gb}} w(j)} + \sum_{V_k' \in BRank} \frac{(RK - R(V))}{RK} \frac{1}{\sum_{j \in V'_k} w(j)} \quad (14)$$

In the implementation of this algorithm, it is important how many best solutions will be taken into account when updating the pheromone trail. In Equation 14 parameter *RK* is used to define the number of best ranked ants who will affect the trail. This parameter is user defined. This parameter is very important for the effectiveness of this algorithm. In its extreme cases when RK is equal to 0 it is equivalent to EAS.

5 Application and Results

In this section, we present the comparative assessment of different variations of ACO. In our test, the implementation of an iteration step has been done by the use of the following pseudo code.

```
Reset Graph Info
Reset Solution for all Ants

while  (! AllAntsFinished)
  for All Ants
    If(Ant Not Finished)
      begin

        add new vertex A to solution
        based on   probability

        correct ants covering graph data

        calculate  new heuristic

        local update rule for A

      End If
    End for
End while

Compute Δτi for variation
Compute  τi
```

The program for our experiments was written in C#, using the framework from article [19]. This framework is dot net based and is designed for creating windows applications. It is implemented as a plug-in system so similar research on the effect of different ACO variations can be conducted on other problems that could be solved by this method just by creating the basic ACO algorithm. We have created a plug-in for this system and used existing features to conduct our tests. The executable alpha version of this software (Fig. 3) and accompanying Microsoft Visual Studio project can be downloaded from *http://mail.phy.bg.ac.yu/~rakaj/home/*. All of our test have been performed on an Intel(R) Core(TM)2Duo CPU E8500 @ 3.16 GHz with 4GB of RAM with Microsoft Windows Vista Ultimate x64 Edition Version 2007 Service Pack 1.

In the tables (1,2,3,4,5,6,7,8,9,10) EAS (Elitist Ant Colony System) corresponds to the algorithm presented in article [6], in which the efficiency of using ant colony optimization on this problem was shown.

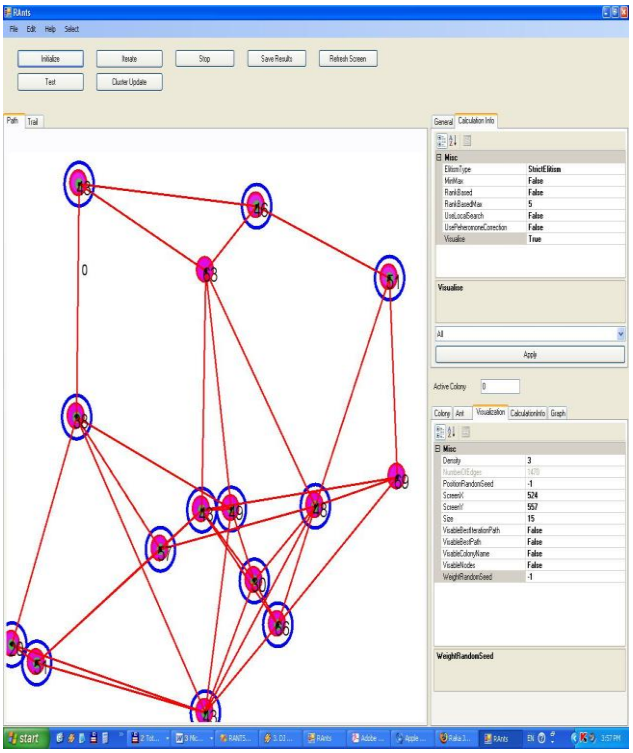

Fig. 3. Graf-Ant software with the plug-in for Minimal Weighted Vertex Covering Problem

We have tested several graphs containing different number of edges and vertexes. In each test, we have used colonies consisting of 10 ants. The exploration rate was q_0=*0.1*, and the influence factor of heuristics was α= *1*, evaporation rates where φ=*0.1*, *p=0.1*. In RANKAS we used *RK = 5*. For the

initial value of the pheromone trail, and τ_0 was calculated from the solution gained using the greedy algorithm presented in article [3].In MMAS for the value of p_{best} =0.05.

For each variation, we conducted 10 separate runs. In each test, we set a maximum number of possible iterations and compared results obtained up to that number of steps. The analysis is done by observing the best-found solution, the average solution value, dispersion and distribution of solutions. The calculation time of each variation of ACO is very similar, so we excluded it from the analysis instead we use the number of iterations.

We generated random problem instances. In which weights where randomly selected for vertexes from the interval [20, 70]. We used graphs of 25, 50, 150, 250, 500 vertexes and for each of these sizes, we tested two different sets of edges. In the algorithm for edge set creation, we would generate *n* edges from each vertex to random vertexes. *n* was a random number between [1, 4] in Tables 1, 3, 5 and [1, 10] in Tables 2, 4, 6, 7 and [10-20] in Tables 8, 10.

We first observe the behavior of these methods in small problem cases (Tables 1, 2). All ACO variations gave the optimal solution, except the two most basic AS and RAS. The optimal solution was found using a brute force method testing the whole solution space.

Table 1. Number of nodes 25, Number of edges 71, greedy algorithm solution value 1088, Maximum number of iterations 1250

Variation	Best Value	Best Value Iteration	Average
AS	839	696	871.6
EAS	779	89	834.7
RAS	787	606	856.3
RANKAS	779	1120	**827.1**
MMAS	779	129	830.6

Table 2. Number of nodes 25, Number of edges 131, greedy algorithm solution value 1135, Maximum number of iterations 1250

Variation	Best Value	Best Value Iteration	Average
AS	952	1064	986.7
EAS	952	21	985.3
RAS	952	78	994.1
RANKAS	952	45	**957.6**
MMAS	952	34	983.6

For finding the optimal solution, we used a recursive method that implements the following pseudo code

```
OptCover ( startIndex, Connections, Covered,
           Sum)
begin
  if(startIndex >= SizeOfGraph) return;
  if(Sum > BestValue) return;
  if(Covered = SizeOfGraph)
    begin
      if (Sum < BestValue) BestValue = Sum;
      return;
    end
  Connections1 = Connections;
  Covered1 = Covered;
  OptCover( startIndex+1, Connections1,
            Covered1, Sum );
  Connections2 = Connections;
  Covered2 = Covered;
  UpdateCovering( Connections2,
                  startIndex, Covered2);
  Sum1 = Sum + NodeValues[startIndex];
  OptCover( startIndex + 1,Connections2,
            Covered2, Sum1);
end
```

In larger problem cases, we did not calculate the optimal solution due to very long execution time. The quality of solution acquired by AS and RAS variations was bad in larger problem sizes. In small problems in average RANKAS gave the best quality of results, but the best solution was found at a higher number of iterations than MMAS and EAS.

In the medium (Tables 3, 4) and large (Tables 5, 6, 7, 8, 9, 10) problem cases MMAS and EAS gave the best results, with EAS being slightly better.

Table 3. Number of nodes 50, Number of edges 172, greedy algorithm solution value 2238, Maximum number of iterations 2000

Variation	Best Value	Best Value Iteration	Average
AS	1736	4	1740.1
EAS	**1554**	1767	1597.4
RAS	1716	517	1744.3
RANKAS	1650	1620	1694.7
MMAS	1556	235	**1589.3**

In medium problems RANK preformed slightly worse than these variations, but in large cases this difference would greatly increase. For problems of this size RANK performance was similar to AS and

RAS. This could be explained by the very large solution space and because of this a need for a more focused search. EAS and MMAS have this property and because of this give better results. We also wish to point out that it is highly possible that if significantly longer calculation time were used RANKS would have improved its results. We believe this is because this variations has a slow convergence to optimal solutions but has a lower possibility of getting trapped in local optima.

Table 4. Number of nodes 50, Number of edges 374, greedy algorithm solution value 2238, Maximum number of iterations 2000

Variation	Best Value	Best Value Iteration	Average
AS	1861	1329	1918.6
EAS	1833	298	1876.3
RAS	1861	127	1885.8
RANKAS	1833	1583	**1872.2**
MMAS	1833	295	1882.2

Table 5. Number of nodes 150, Number of edges 562, greedy algorithm solution value 6782, Maximum number of iterations 2000

Variation	Best Value	Best Value Iteration	Average
AS	5827	993	5951.2
EAS	**4920**	1688	**5117.9**
RAS	5760	1476	5912.1
RANKAS	5694	999	5802.2
MMAS	5002	1952	5169.2

Table 6. Number of nodes 150, Number of edges 1470, greedy algorithm solution value 6834, Maximum number of iterations 2000

Variation	Best Value	Best Value Iteration	Average
AS	6303	1606	6354.7
EAS	**5688**	1932	**5872.5**
RAS	6284	402	6185.8
RANKAS	6156	624	6230.7
MMAS	5756	1701	5889.6

Table 7. Number of nodes 250, Number of edges 970, greedy algorithm solution value 10877, Maximum number of iterations 500

Variation	Best Value	Best Value Iteration	Average
AS	9814	370	9961.2
EAS	**8962**	458	**9259**
RAS	9958	247	10006.4
RANKAS	9686	111	9761
MMAS	9106	480	9269.2

Table 8. Number of nodes 250, Number of edges 8399, greedy algorithm solution value 11239, Maximum number of iterations 500

Variation	Best Value	Best Value Iteration	Average
AS	10900	364	10968.4
EAS	**10683**	441	10747.4
RAS	10915	3	10957.6
RANKAS	10882	362	10940.6
MMAS	10698	328	**10763.2**

Table 9. Number of nodes 500, Number of edges 18581, greedy algorithm solution value 22395, Maximum number of iterations 500

Variation	Best Value	Best Value Iteration	Average
AS	21900	271	21997.6
EAS	21499	494	**21558**
RAS	22014	411	22057.2
RANKAS	21963	149	22005.4
MMAS	**21462**	457	21624

Table 10. Number of nodes 500, Number of edges 50973, greedy algorithm solution value 22626, Maximum number of iterations 500

Variation	Best Value	Best Value Iteration	Average
AS	22421	264	22458.8
EAS	**22344**	341	**22440.4**
RAS	22406	434	22436
RANKAS	22430	453	22463.6
MMAS	22410	63	22448.8

To better qualify the performance of different variations of ACO we also observed the dispersion and distribution of solutions obtained by them. As a measure of dispersion, we used the standard deviation given by the Equation 15

$$s = \sqrt{\frac{\sum (\bar{W} - W_i)^2}{n-1}} \qquad (15)$$

The results for these properties where very similar for all problem cases, because of this we only present the data for problems from tables 3 and 6. We show the standard deviation in table 11.

Table 11. Standard deviation of solutions

Variation	Stand. Deviation for Table 3	Stand. Deviation for Table 6
AS	20.7354	22.608
EAS	47.2912	135.312
RAS	38.6891	84.3487
RANKAS	58.5454	58.5454
MMAS	27.198	37.4019

We first notice that AS and RAS have a small deviance, which can be explained by the fact that they are trapped in local optima in early steps of the algorithm. The better performing variations have noticeable greater deviances. The importance of this property is better understood if we observe the distribution of the solutions in Fig. 4 and 5.

Now we can see the difference in behavior of AS and RAS. These two methods give the average solution and best solutions of similar quality, but the distribution is very different. AS gives us a much smaller variety of solutions, than RAS.

It is surprising that by reinforcing of the best solution the dispersion grows which is opposite to the first impression of the effect of focusing the search near good solutions. The effect of no or weak reinforcement of best iteration solutions and global best solution is that, the ants search a wider area near the best solutions but when a better solution is found, the pheromone trail will not change sufficiently. Because of this, the majority of the ants will not move to the area near the newly found best solution. This explains the fact that RANKS converges to good solutions slower that EAS and MMAS but can find better ones. In RANKS the reinforcement of the global best is weaker than in these two methods, because of this areas around each solution are more detailed searched, but it takes longer to move to regions around new best solutions.

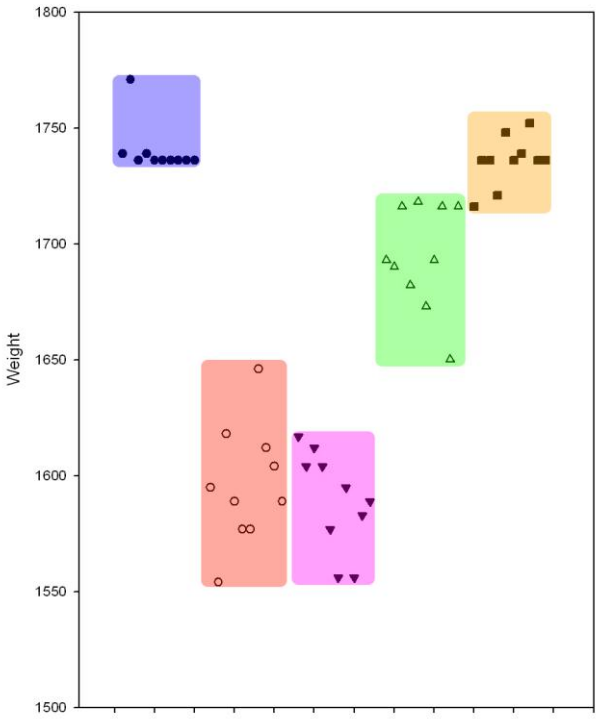

Fig. 4. Distribution of solutions for problem from Table 3 AS-filled circles, EAS – empty circles, MMAS – filled triangles, RANKAS-empty triangles, RAS-filled squares

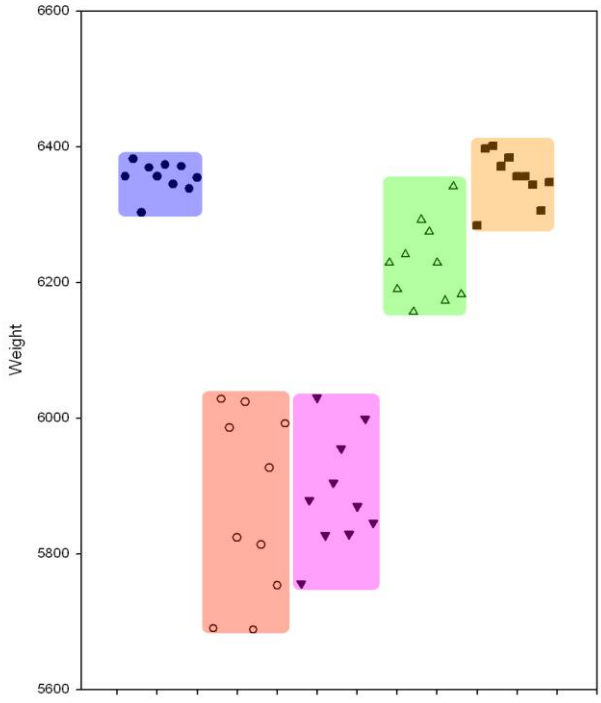

Fig 5. Distribution of solutions of problem from Table 6 AS-filled circles, EAS – empty circles, MMAS – filled triangles, RANKAS-empty triangles, RAS-filled squares

The last conclusion that we make from the analysis of dispersion and distributions is that MMAS is a more reliable for acquiring good results than EAS. EAS has a greater dispersion than MMAS. This means that even if there is a higher possibility of getting the best overall solution with EAS it is more like to get a good solution with MMAS.

In practical use, this means if we are going to perform a larger number of separate runs to find the best solution it is better to use EAS, but in the case of a small number of attempts MMAS is a better choice.

6 Conclusion

In this paper, we have done an extensive analysis of the standard variations of ACO, which are Ant System, Reinforced Ant system, Elitist Ant system, Min Max Ant System and Rank based Ant System on the MWVC problem. To do this we have transformed the standard variation formulas to a form that could be applied on this problem. We used our previously developed framework [19] to create software for conducting tests. Our first observation was that the difference in calculation time for all the variations was neglect able. We have analyzed different aspects of solutions calculated for these variations like the best-found solution, average solution quality, dispersion and distribution of solutions. From our analysis, we came to the following conclusions.

The two basic algorithms AS and RAS gave significantly worst results in all tested cases than the other methods and are not a good choice for this problem. An overall best variation did not exist but it depended on the size of the problem, and the available resources. In small problem cases EAS, MMAS and RANKS gave good results. RANKS gave the best quality of results but it had found them in a higher of number iterations than the other two methods. In our tests, it has been shown that EAS and MMAS gave the best results in large problem cases with EAS having slightly better results when best solutions and average solution quality were compared. When we took into account the dispersion and distribution of solutions, we concluded that EAS could not be seen as the better method than MMAS. This is due to the fact that MMAS solutions where less dispersed and because of that the method can be considered to be more reliable. The choice of which of these two methods we shall use depends on our resources if plan to conduct a large number of separate runs it is better to use EAS, and if a small number if tests is planed MMAS is the better choice. Although RANKS preformed worse than EAS and MMAS in our test it should not be discarded. By analyzing the tendencies in our experiments, we have observed that RANKS converges slower than the other two methods but can come to better solutions if the calculation time is sufficiently long.

In further research, we wish to adopt and implement the suspicion hybridization for ACO used on the TSP[19]. Parallelization of ACO proved to be very effective and in some cases even give super linear improvement, because of this we wish to implement parallel version of ACO on this problem and compare the effectiveness of different parallel topologies.

References:

[1] Karp, R.M.. Reducibility Among Combinatorial Problems. In R.E. Miller and J.W. Theater, *Complexity of Computer Computations*, New York: Plenum Press, 1972

[2] Chvatal, V., A Greedy-Heuristic for the Set Cover Problem. *Mathematics of Operations Research*, Vol.4, 1979, pp. 233–235.

[3] Clarkson, K.L., A Modification of the Greedy Algorithm for Vertex Cover, *Information Processing Letters*, Vol. 16, 1983, pp. 23–25.

[4] Ashok Kumar Gupta, Alok Singh, A Hybrid Heuristic for the Minimum Weight Vertex Cover Problem, *Asia-Pacific Journal of Operational Research*, 2006, vol. 23, No 2, pp 273-285

[5] Dorigo M, Maniezzo V: Ant Colony system: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man and Cybernetics* - Part B Vol. 26, No.1, 1996, pp. 29-41

[6] Shyong Jian Shyu, Peng-Yeng Yin, Bertrand M.T. Lin, An Ant Colony Optimization Algorithm for the Minimum Weight Vertex Cover Problem, *Annals of Operations Research*, Vol.131, 2004, pp. 283–304,

[7] D.Asmar ,A. Elshamli, S. Areibi, A Comparative Assessment of ACO Algorithms Within a TSP Environment. *In DCDIS: 4th International Conference on Engineering Applications and Computational Algorithms*, Guelph, Ontario, Canada , July 2005.

[8] R. Garey and D. Johnson, Computers and Intractability, *A Guide to the Theory of NP-Completeness* W.H. Freeman and Company, San Francisco, 1979.

[9] Vlachos Aristidis, An Ant Colony Optimization (ACO) algorithm solution to Economic Load Dispatch (ELD) problem. *WSEAS Transactions On Systems,* Vol 5, No 8, pp. 1763 – 1771, 2006

[10] Kolahan, F., Abachizadeh, M., Soheili, S, A comparison between Ant colony and Tabu search algorithms for job shop scheduling with sequence-dependent setups, *WSEAS Transactions on Systems, Vol. 12*, 2006, pp. 2819-2824

[11] Mastorakis, N.E., Zhuang, X, Image processing with the artificial swarm intelligence, *WSEAS Transactions on Computers*, Vol 4, No. 4, 2005, pp. 333-341

[12] Broderick Crawford, Carlos Castro, Ant Colonies using Arc Consistency Techniques for the Set Partitioning Problem, *Professional Practice in Artificial Intelligence*, Springer Boston, 2006, pp. 295-301

[13] Youmei Li, Zongben Xul, An ant colony optimization heuristic for solving maximum independent set problems, *Proceedings of the 5th International Conference on Computational Intelligence and Multimedia Applications*, 2003 pp: 206 – 211

[14] Serge Fenet, Christine Solnon, Searching for Maximum Cliques with Ant Colony Optimization, *Applications of Evolutionary Computing*, Springer Berlin / Heidelberg, 2003, pp. 291-302

[15] Y.J Feng, Z.R. Feng, Ant colony system hybridization with simulated annealing for flow-shop scheduling problems. *WSEAS Transaction on Business and Economics*, Vol. 1, No. 1, 2004, p. 133-138

[16] Hong-hao Zuo, Fan-lun Xiong, Time Ant Colony Algorithm with Genetic Algorithms, Information Acquisition, *IEEE International Conference on* Volume , Issue , 20-23 Aug. 2006, pp. 1057 - 1061

[17] T. Stützle et H.H. Hoos, MAX MIN Ant System, *Future Generation Computer Systems*, Vol. 16, 2000, pp. 889-914

[18] B. Bullnheimer, R. F. Hartl, and C Strauss. A new rank-based version of the ant system: a computational study, *Central European Journal for Operations Research and Economics*, Vol.7 No.1, 1999, p. 25-38,

[19] Raka Jovanovic, Milan Tuba, Dana Simian, An Object-Oriented Framework with Corresponding Graphical User Interface for Developing Ant Colony Optimization Based Algorithms, *WSEAS Transactions on Computers*, Vol. 7, No. 12, 2008, pp. 1948 – 1957

Acknowledgment: The research was supported by the Ministry of Science, Republic of Serbia, Projects no. 144007 and 141031