A Comparative Assessment of Ant Colony Optimization Algorithms for the Minimum Weight Vertex Cover Problem

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

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

Abstract: - In this paper we analyze the application of the Ant Colony Optimization to the Minimum Weight Vertex Covering Problem. We use the software system that we developed and implemented different standard ACO algorithms to this problem: Ant Colony System, the use of Elitism, Rank based approach and the MinMax system. We have made a comparative assessment of the effectiveness of these algorithms to the Minimum Weight Vertex Covering Problem in different problem cases and shown that Elitist Ant System and MinMax Ant System give better solutions for larger test cases while run-times for all algorithms were similar.

Key-Words: - Ant Colony Optimization, Minimum Weight Vertex Cover, Evolutionary Computation, Swarm Intelligence

1 Introduction

The Minimum Weight Vertex Cover Problem (MWVCP) is considered for an undirected graph $G=(V, E)$, with weights assigned to each vertex in the graph. A vertex cover 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 cover that has the minimal sum of weights of vertexes that are members of that cover.

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] (degree of a vertex is the number of edges that have that vertex as a member). A large number of real life problems could be converted to this form. An example could be the optimal positioning of garbage disposal facilities.

In the same way as for many other NP–complete problems, finding the optimal solution is very time consuming and, in larger problem cases, even impossible in realistic time. Because of this a variety 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]. This problem has also been solved by use of genetic algorithms [4].

The use of Ant Colony Optimization (ACO) gave very good results when used on the MWVCP. The solutions were better than those acquired by genetic algorithms and local search methods like tabu search and simulated annealing [5].

It has been shown that for some problems variations of ACO give results of different quality [6]. 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 the MWVCP.

This paper is organized as follows. In section 2 we present the implementation of ACO for the MWVCP. In Section 3 we show the algorithm variations for this problem. In Section 4 we analyze and compare experimental results of using different variations of the ACO.

2 ACO for MWVCP

The use of ACO has proven to be effective on various types of problems from Economic Load Dispatch [7], Scheduling problems [8], even Image Processing [9] and its use has been also proven powerful on the MWVCP. The MWVCP is however, different from most of the problems solved using ACO in two main aspects. To illustrate this, for comparison, we will use the Traveling Salesman Problem (TSP).

For the TSP the solution is an array of all the cities appearing in the problem, or in other words,

the solution is a 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 the TSP the 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 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.

These two differences affect the basic algorithm in the direction that the ants leave the pheromone on vertexes instead on edges and that 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 one vertex to any other vertex, it is natural to use a fully connected graph $G_c(V,E_c)$ derived from G. In [5] it is proposed to add weights of 1 if the edge exists in G or 0 if it does not exist in the original graph. We have adopted this approach, which is illustrated by Fig.1.

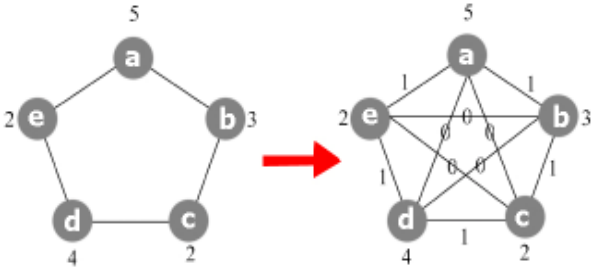

Fig. 1 Expansion to the fully connected graph

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

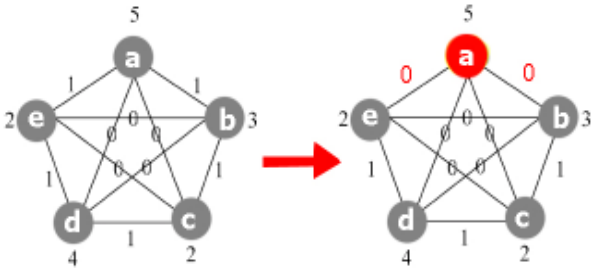

Fig. 2 Adding a vertex to the solution set

Let us define $G_k\ (V,E_{ck})$ as the state of the graph after k vertexes have been added to the solution, and a corresponding function:

$$\psi_k(i,j) = Value(E_{ck}(i,j)) \qquad (1)$$

Now we can define a dynamic heuristic

$$\eta_{jk} = \frac{\sum_{(i,j)\in E_c} \psi_k(i,j)}{w(j)} \qquad (2)$$

In Equation 2 $w(j)$ is the weight of a vertex. Using the heuristic defined with η_{jk} in Equation 2 we can setup the state transition rule for ants:

$$p_j^k = \begin{cases} 1 & , q > q_0 \ \&\ j = \arg\max_{i\in A_k} \tau_i \eta_{ik}^{\alpha} \\ 0 & , q > q_0 \ \&\ j \neq \arg\max_{i\in A_k} \tau_i \eta_{ik}^{\alpha} \\ \dfrac{\tau_j \eta_{jk}^{\alpha}}{\sum_{i\in A_k} \tau_j \eta_{ik}^{\alpha}} & , q \le q_0 \end{cases} \qquad (3)$$

In Equation 3 q_0 is the standard parameter that specifies the *exploitation / exploration* rate, and 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 and that is why we have τ_i instead of τ_{ij}.

To fully specify an Ant Colony System we still have to define the global (when an ant finishes its path) and local (when an ant chooses a new vertex) update rule:

$$\Delta\tau_i = \frac{1}{\sum_{j\in V'} w(j)} \quad , \forall i \in V' \qquad (4)$$

$$\tau_i = (1-p)\tau_i + \Delta\tau_i \qquad (5)$$

In Equation 4 $\Delta\tau_i$ is a quality measure of solution subset V' that contains vertex i, and with it we define a global update rule in Equation 5. Parameter p is used to set the influence of newly found solution on the pheromone trail. The formula for the local update rule has the standard form

$$\tau_i = (1-\varphi)\tau_i + \varphi\tau_0 \qquad (6)$$

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.

3 Variations of ACO for MWVCP

On the TSP different variations of ACO gave different quality of results [6]. That is why we have performed a comparative assessment of standard variations of ACO on the MWVCP. The variations mostly differ in the global update rule. We have used the following variations.

Ant System (AS) in which all ants leave pheromone. It is defined with Equations 7 and 5. In Equation 7 *AntS* is the set of all the solutions created by ants in the current step of the algorithm.

$$\forall i \in \bigcup_{V_i' \in Ants} V_i'$$
$$\Delta\tau_i = \sum_{V_k' \in AntS} \frac{1}{\sum_{j \in V_k'} w(j)} \tag{7}$$

Reinforced Ant System (RAS), which is the same as Ant colony system except the global best solution, is reinforced in each iteration. It is defined with Equations 8 and 5.

$$\forall i \in V'_{gb} \cup \bigcup_{V_i' \in Ants} V_i'$$
$$\Delta\tau_i = \frac{1}{\sum_{j \in V'_{gb}} w(j)} + \sum_{V_k' \in AntS} \frac{1}{\sum_{j \in V_k'} w(j)} \tag{8}$$

Elitist Ant System (EAS) in which only the global best solution leaves pheromone at each iteration. It is defined with Equations 9 and 5.

$$\Delta\tau_i = \frac{1}{\sum_{j \in V'_{gb}} w(j)}, \forall i \in V'_{gb} \tag{9}$$

MinMax Ant System (MMAS) is same as the Elitist Ant colony system, but with an extra constraint that all pheromone values $\tau_i \in [\tau_{\min}, \tau_{\max}]$. We adopt the formulas presented in [10] in which $\tau_{\max}$ is calculated dynamically as new best solutions are found by Equation 10, and $\tau_{\min}$ is calculated at the beginning of calculations with Equation 11. 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.

$$\tau_{\max} = \frac{1}{(1-p)} \Delta\tau_{gb} \tag{10}$$

$$\tau_{\min} = \frac{\tau_0 (1 - \sqrt[n]{p_{best}})}{(avg - 1)\sqrt[n]{p_{best}}} \tag{11}$$

Rank Based Ant Colony System (RANKAS) in which besides the quality we also use the rank (*R*) of found solutions. The rank is quality of the solutions compared to solutions found by other ants in the same iteration. It is defined with Equations 5 and 12.

$$BRank = \{V \mid (R(V) < RK) \wedge (V \in AntS)\}$$
$$\forall i \in V'_{gb} \cup \bigcup_{V_i' \in BRank} V_i'$$
$$\Delta\tau_i = \frac{1}{\sum_{j \in V'_{gb}} w(j)} + \sum_{V_k' \in BRank} \frac{(RK - R(V))}{RK} \frac{1}{\sum_{j \in V_k'} w(j)} \tag{12}$$

In this variation we select *RK* best ranked solutions and, depending on their quality and rank, we correct the pheromone trail.

4 Application and Results

In this section we present the comparative assessment of different variations of ACO. The implementation of an iteration step is shown in the pseudo-code in Fig. 3.

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

We have tested for 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*. The initial value of the pheromone trail and τ_0 where calculated from the solution gained using the greedy algorithm presented in [3]. In MMAS the value of $p_{best}=0.05$.

```
Reset Graph Info
Reset Solution for all Ants
while  (! AllAntsFinished)
  For All Ants
   If(AntNotFinished)
     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 Δτ_i for variation
Compute  τ_i
```

Fig. 3 Pseudo-code for an iteration step

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 and the average solution value. The calculation time of each variation of ACO is very similar, so we excluded it from the analysis and instead we used the number of iterations. The program for our experiments was written in C#, using the framework from [11].

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 vertexes and for each of this sizes, we tested two different sets of edges. In the algorithm for edge set creation, we generated *n* edges from each vertex to random vertexes where *n* was a random number between [1,4] in Tables 1,3,5 and between [1,10] in Tables 2,4,6.

In small problem cases (Tables 1 and 2) all ACO variations gave good results, except the two most basic AS and RAS. The quality of solution acquired by these two variations was also bad in larger problem sizes. For average values RANKAS gave the best quality of results, but the best solution was found at a higher number of iterations than MMAS and EAS.

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

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

In medium size 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.

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**

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

5 Conclusion

In this paper, we performed a comparative assessment of the standard variations of ACO on the MWVC problem. To do this we have transformed variation formulas to a form that could be applied to this problem. We used our previously developed framework [11] to create software for conducting tests. The difference in calculation time for all the variations was neglectable. An overall best variation did not exist but it depended on the size of the problem. The basic use of ACO gave significantly worse results in all tested cases compared to MMAS, EAS and RANKAS. Through analysis it has been shown that EAS and MMAS gave the best results in large problem cases with EAS slightly better. Rank Based Ant Systems gave the best results in small cases, and slightly worse in medium problem instances, but in large cases it is not a good approach.

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

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] 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,

[6] 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.

[7] 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

[8] 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*, pp. 2819-2824, 2006

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

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

[11] 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