Meta-Heuristic Optimization techniques in power systems

Vlachos Aristidis
Department of Informatics
University of Piraeus
80, Karaoli & Dimitriou str. 18534 , Pireus
GREECE

Abstract: Ant Colony Optimization (ACO) algorithm is a modern Optimization technique solving problems of power system field. It is inspired by the behavior of ants in finding paths of the ant colonies to food. In this paper, the Max-Min Ant System, an Ant Colony Optimization algorithm, is used for solving the Generator Maintenance Scheduling problem.

Key-words: Ant Colony Optimization, Ant System, MAX-MIN Ant System, metaheuristic algorithms, Generator Maintenance Scheduling, Power Systems.

1. *Introduction*

Planning tasks are vital for maintaining safe and economical operations in modern power systems. Planning tasks are categorized into the long-term system planning tasks and the short-term operational planning tasks. System planning tasks concern long term investments in power systems and may include activities like generation expansion through new installations of generating units or increase transmission capacity through building new power transmission lines. On the other hand operational planning tasks concern activities like generator maintenance scheduling, unit commitment or economic dispatch. Preventing maintenance scheduling of generating units plays a key role and affects both system and operational planning decisions over the planning horizon. The importance of maintenance scheduling is based on the high cost of the maintenance equipment and the need to ensure high system reliability. So, the objectives of generator maintenance scheduling (GMS) could be to assure system reliability or minimize the maintenance and operational cost with respect to both generating units and power system constraints. The goals concerning GMS can be divided into two categories, those which are based on reliability and those that rely on economic cost. A common reliability criterion is to maintain a certain level of reserve generation during the operational planning period. This is exactly our concern in this paper which can be achieved by minimising the sum of squares of reserved generation [1]. However, the computation of a maintenance schedule involves several other factors – constraints, which should be met in order to ensure system security and reliability. Typical constraints of a GMS problem may include according to [1]:

- Maintenance window constraints, which define the possible times and the duration of maintenance for each generating unit.

- Resource constraints, which specify the limits on the resources needed for maintenance at each period.
- Crew constraints, which consider the manpower availability for maintenance work.
- Exclusion constraints, which prevent the simultaneous maintenance of a set of units.
- Sequence constraints, which restrict the initiation of maintenance of some units after a period of maintenance of some other units.
- Reliability constraints, which consider the risk level of a given maintenance schedule.
- Load constraints, which consider the demand on the power system during the scheduling period.
- Transmission capacity constraints, which specify the limit of transmission capacity in an interconnected power system.

Geographical constraints, which limit the number of generators under maintenance in a region.

GMS problem is a hard combinatorial optimization problem and is classified as a deterministic cost-minimization problem. Many traditional optimization methods – dynamic programming, integer programming and branch-and-bound – have been proposed to solve the problem. All of them function well for relatively small problem instances, but as the problem size increases the size of the search space increases exponentially and the computational time of these methods, as well. These limitations make the above methods inappropriate to apply in a real world GMS problem.

However, the drawbacks that traditional approaches suffer from can be overcome using modern optimization techniques [10]. These alternative methodologies include genetic algorithms (GAs) [1, 11, 12], simulated annealing (SA) [12, 9], tabu search (TS) [12, 9] and neural networks [13]. These methods have been applied successfully on their own, but hybrid approaches have also been reported in literature [12, 9].

This paper deals with maintenance scheduling of generating units and relies upon the work of Aldridge, Dahal and McDonald presented in the book Modern Optimization Techniques in Power Systems [1]. We provide a different approach using a nature inspired algorithm, the *MAX-MIN* Ant System (MMAS), which handles well hard combinatorial optimization problems regardless of the problem size.

The rest of this paper describes the GMS problem and how MMAS is applied to a GMS problem instance. In section 2 the problem formulation is given. A brief introduction to Ant Colony Optimization (ACO) and MMAS is given in section 3. Section 4 describes the design of MMAS algorithm, focusing on the key aspects of representation and heuristics as well as the experimental algorithm itself. Section 5 gives the computational results and section 6 presents some useful conclusions.

2. Generator Maintenance Scheduling problem

2.1 Problem formulation

As it is already mentioned, the problem we delve into is based on the problem described in [1]. The goal of this GMS problem is to ensure system reliability by minimizing the sum of squares of reserve generation in every period. GMS problem requires maintaining each unit – without interruption – for a specified duration within a specific window while the available manpower for maintenance work is limited. The available crew each week is bounded to 20 people and the peak load of the system is 4739 MW. Below, Table 1 presents the capacities, allowed periods, duration of maintenance and the manpower required for each period of maintenance for every unit of the system.

Table 1: Data for the test system

Unit	Capacity (MW)	Allowed period	Outage (weeks)	Manpower required for each week
1	555	1-26	7	10+10+5+5+5+5+3
2	555	27-52	5	10+10+10+5+5
3	180	1-26	2	15+15
4	180	1-26	1	20
5	640	27-52	5	10+10+10+10+10
6	640	1-26	3	15+15+15
7	640	1-26	3	15+15+15
8	555	27-52	6	10+10+10+5+5+5
9	276	1-26	10	3+2+2+2+2+2+2+2+2+3
10	140	1-26	4	10+10+5+5
11	90	1-26	1	20
12	76	27-52	3	10+15+15
13	76	1-26	2	15+15
14	94	1-26	4	10+10+10+10
15	39	1-26	2	15+15
16	188	1-26	2	15+15
17	58	27-52	1	20
18	48	27-52	2	15+15
19	137	27-52	1	15
20	469	27-52	4	10+10+10+10
21	52	1-26	3	10+10+10

In order to formulate the GMS problem, the following variables are introduced:

i index of generating units

I set of generating units indices

N total number of generating units

t index of periods

T set of indices of periods in planning horizon

e_i earliest period for maintenance unit I to begin

l_i latest period for maintenance of unit i to end

d_i duration of maintenance for unit i

P_{it} generating capacity of unit i in period t

L_t anticipated load demand for period t

M_{it} manpower needed by unit i at period t

AM_t available manpower at period t

We have no reason to change the problem formulation presented in [1], in order the reader to follow easier our alternative to GAs for the GMS problem. For the completeness and the clarity of the approach in [1] some additional sets have to be defined.

So, let $T_i \subset T$ be the set of periods when maintenance of unit i may start, so that for each unit i is $T_i = \{t \in T : e_i \le t \le l_i - d_i + 1\}$. Additionally,

$$X_{it} = \begin{cases} 1 & \text{if unit i starts maintenance in period t} \\ 0 & \text{otherwise} \end{cases}$$

is defined, to be the maintenance start indicator for unit $i \in I$ in period $t \in T_i$. Furthermore, $S_{it} = \{k \in T_i : t - d_i + 1 \le k \le t\}$ is defined, where S_{it} is the set of start time periods k such that if the maintenance of unit i starts at period k that unit will be in maintenance at period t.

Finally, $I_t = \{i : t \in T_i\}$ is the set of units which are allowed to be in maintenance in period t.

The objective function, which is to minimize the sum of squares of reserve generation, as mentioned above, as well as the GMS problem constraints are given below.

GMS objective function

$$\underset{X_{it}}{\mathrm{Min}} \left\{ \sum_{t \in T} \left(\sum_{i \in I} P_{it} - \sum_{i \in I_t} \sum_{k \in S_{it}} X_{ik} P_{ik} - L_t \right)^2 \right\} \quad (1)$$

maintenance window constraint

$$\sum_{t \in T_i} X_{it} = 1 \text{ for all } i \in I \quad (2)$$

manpower constraint

$$\sum_{i \in I_t} \sum_{k \in S_{it}} X_{ik} M_{ik} \le AM_t \text{ for all } t \in T \quad (3)$$

load constraint

$$\sum_{i \in I} P_{it} - \sum_{i \in I_t} \sum_{k \in S_{it}} X_{ik} P_{ik} \ge L_t \text{ for all } t \in T \quad (4)$$

3. ACO algorithms

Generally, ACO is a family of algorithms which have been inspired by the foraging behaviour of real ant colonies. They have recently developed and have been applied to common hard combinatorial problems with encouraging results [4].

The ACO algorithms are using agents, called *ants,* which iteratively construct candidate solutions for the optimization problem under consideration. Ants build a solution probabilistically by iteratively adding solution components to their partial solutions influenced by heuristic information of the problem solved, if any, and (artificial) pheromone trails which change dynamically during the construction procedure to reflect ants' acquired search experience. In theory, ACO algorithms can be applied to any combinatorial optimization problem, by simply defining the solution components of the problem which will be used by the ants to build a candidate solution with probable pheromone deposit (more details can be found in [4]. Every ant starts with an empty solution and iteratively adds a solution component without backtracking until a complete candidate solution is obtained. After the completion of the solution construction procedure, ants deposit pheromone to every solution component they have used to build their candidate solution. Those solution components which are part of the better solutions or simply are used most by the ants receive greater pheromone amounts, and thus are made more preferable to ants in future iterations of the algorithm. In order to avoid algorithm stagnation and achieve greater exploration of the search space, pheromone evaporation on each pheromone trail by a factor ρ takes place before pheromone deposit.

The first ACO algorithms – such as Ant System – might construct solutions which either were not optimal or the algorithm performance was not competitive compared to state-of-art algorithms of the problem domain. These drawbacks were addressed by later variations of the algorithm, which basically aimed to a stronger exploitation of the search history in order to guide ants' search process. MAX-Min Ant System, which is used in this paper for tackling GMS problem, achieve this objective by allowing only the ants found the best solutions to deposit pheromone during pheromone trail update. Also, MMAS may easily and effectively extended by local search algorithms, which may further improve solutions' quality. In this hybrid algorithm model ACO algorithms are used to produce an initial set of solutions and then let local search algorithm to operate on this.

In general all ACO algorithms for static combinatorial problems are following the following algorithmic scheme:

ACO procedure for static combinatorial problems

Define parameters, initialize pheromone trails

while termination condition not met

construct solutions

apply local search algorithm //optional

update pheromone trails

end

end

After the definition of some parameters and the initialization of pheromone trails, a main loop takes place until a termination condition is met. Termination condition could be a specific cpu time limit or a number of constructed solutions. In the while loop, ants construct effective solutions, which optionally can be improved by a local search algorithm. Finally, pheromone trail update takes place.

Every ACO algorithm has initially been tested on Travelling Salesman Problem (TSP) [5, 6, 8] and Quadratic Assignment Problem (QAP) [7, 8] which are classical, well known *NP*-hard combinatorial optimization problems with encouraging results in terms of both computation time and solution quality (constructed solutions approach optimal). As already mentioned, the research on ACO algorithms has shown that shorter execution times can be achieved by a stronger exploitation of best solutions found. However, such strategies may guide in a premature convergence of algorithm in a local optimum. So the key for ACO algorithms in order to achieve shorter computation times is the exploitation of best solutions found, while an effective mechanism prevent algorithm from premature stagnation. These are some key features taken into account in MMAS design in contrast to Ant System. To summarize, there are three key points, which differentiate MMAS from Ant System [8]:

- To exploit the best solutions found during an iteration or during the run of the algorithm, after each iteration only one single ant adds pheromone. This ant may be the one which found the best solution in the current iteration (iteration-best ant) or the one which found the best solution from the beginning of the trial (global-best ant).
- To avoid stagnation of the search the range of possible pheromone trails on each solution component is limited to an interval [τ_{min}, τ_{max}].
- Additionally, we deliberately initialize the pheromone trails to τw*max*, achieving in this way a higher exploration of solutions at the start of the algorithm.

4. Algorithm description

We need to transform the mathematical problem formulation described in section 2.1 in ACO terms and design an effective and efficient MMAS algorithm to tackle GMS problem. Firstly, the assignment problem, that is the assignment of a unit to be maintained to a period, should be tuned into an optimal path searching problem, which is a problem that MMAS algorithm can solve. The first step towards this goal is to create a graph representation of GMS problem, which will be utilized by the MMAS algorithm to construct solutions for the problem. Next we should decide for the appropriate representation of pheromone, which ants will deposit on graph arcs and the problem dependant heuristic information, which affects every step of solution construction process.

4.1 Graph construction

A fundamental principle of ACO metaheuristic algorithms is the representation of the problem under consideration as a graph [2]. Every node of this graph should correspond to a solution component and every path to a solution of the problem. So, for the GMS problem every unit $i \in I$ should be mapped to a period $t \in T_i$. This mapping indicates that unit $i \in I$ starts maintenance at period $t \in T_i$. This graph representation is given by, and we have to decide if ants will traverse the set of units and choose an appropriate period for mapping or ants will traverse the $I \times T$ set of periods and choose a favorable unit for mapping. Both approaches have their pros and cons. In our case the second alternative has been selected, because the fulfillment of the window maintenance constraint – described by (1) – becomes easier which results in fewer evaluations and improved performance. Figure 1 depicts the constructed graph. But as the number of periods |T| is much greater than the number of units |N|, we need to introduce a new set of virtual periods $T' = \{t'_1, t'_2, ..., t'_{|N|}\}$, since the total number of mappings (unit to start maintenance period) is equal to the number of units |N|. Every virtual period maps to an actual period. To overcome the complexities of using a function to designate how a virtual is related to an actual period $f : T' \to T$, the *null unit* is introduced, to correspond with periods that do not have mapped with any actual unit $i \in I$ (Figure 2). In contrast to actual units, *null unit* can be mapped to periods many times and cannot be part of a solution or be involved in pheromone update process.

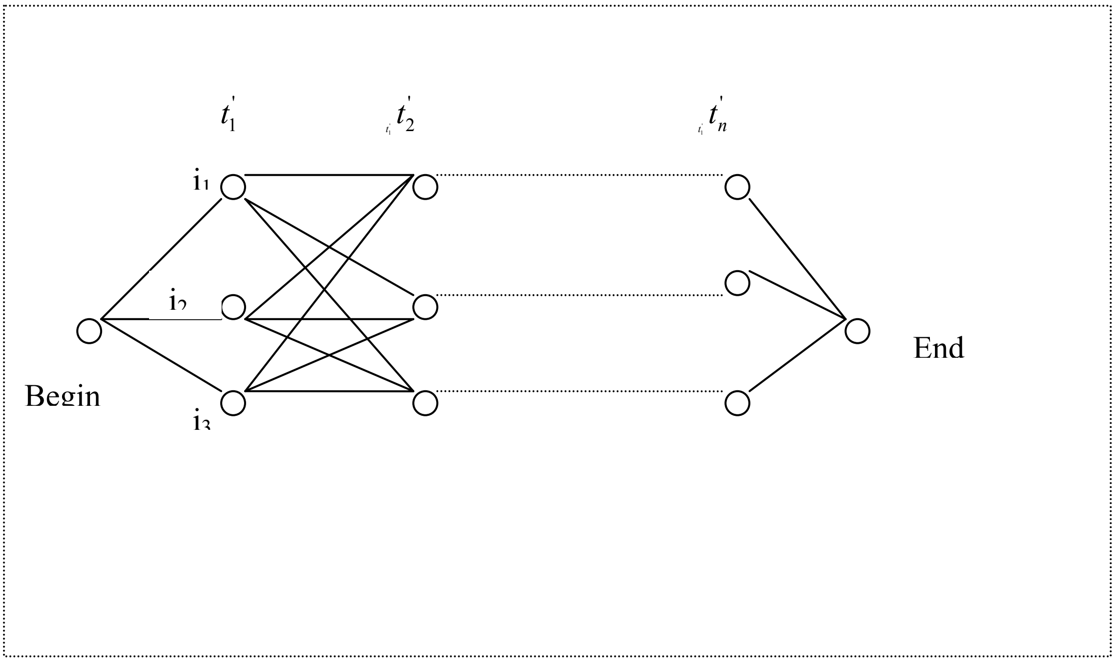

Figure 1: Every ant traverses the set of virtual periods and for each one of them chooses the unit to start maintenance for.

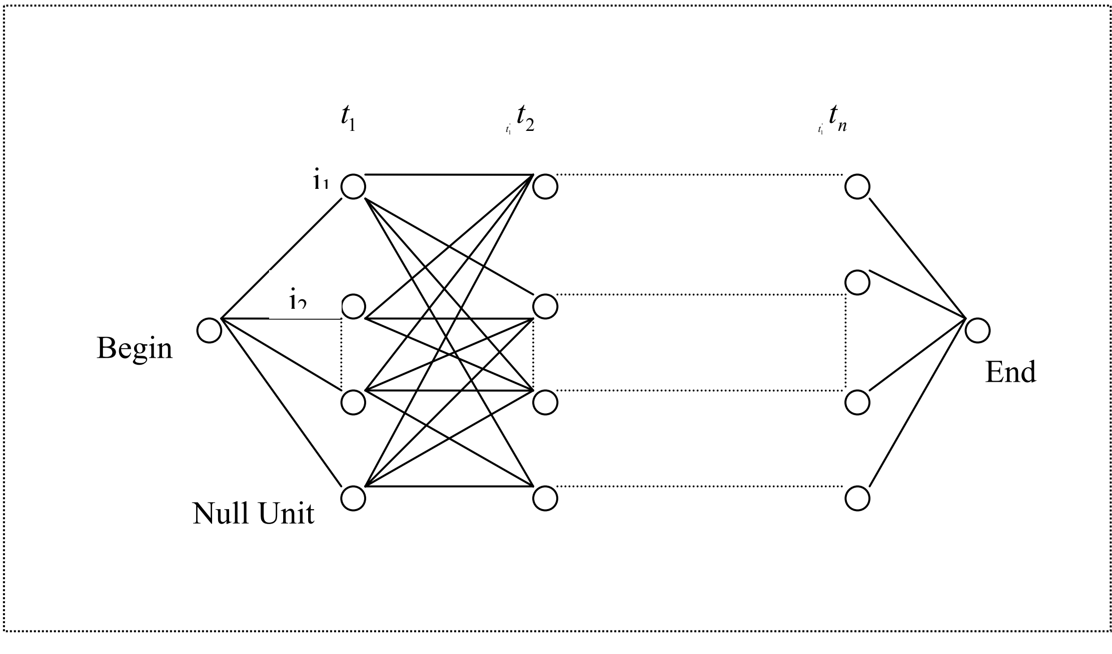

Figure 2: Every ant traverses the set of periods and for each one of them chooses an actual unit or even the null unit to start maintenance for.

Every ant traverses the above graph and creates assignments (mappings) of periods to units, which comprise part of problem solution (solution components) that the ant will iteratively create. In each step every ant chooses the next unit for mapping for the current period. This choice is based on a probabilistic decision which is biased by the pheromone trail amount τ_{it} and the locally available heuristic information η_{it} and is given by equation (5):

$$p_{it}(M_{i-1}) = \frac{(\tau_{it}(M_{i-1}))^{a} \cdot (\eta_{it}(M_{i-1}))^{\beta}}{\sum_{i \in I}(\tau_{it}(M_{i-1}))^{a} \cdot (\eta_{it}(M_{i-1}))^{\beta}} \quad (5)$$

where *α, β* are two parameters which determine the relative importance of pheromone trail and heuristic information and M_{i-1} corresponds to the mapping (assignment) of current period to unit *i-1*. So equation (5) calculates the probability an ant to choose unit *i* while traversing the set of periods *T* and create a new partial mapping M_i. This partial mapping is added to previous partial mapping M_{i-1} in order to create a complete mapping which actually forms a GMS problems' solution. Each ant stores the units chosen in each step to a list (just as the tabu list in Tabu Search algorithm). The units in the list should not be used by an ant until the completion of solution construction and are used to form ant's feasible neighbour in each construction step.

4.2 Pheromone trails

Pheromone is deposited on the arcs of the graph and guides each ant's choice in every construction step. It is necessary to make clear that that arcs that contain the *null unit* do not receive any pheromone at all. It was introduced in our algorithm description just as an auxiliary notation in order to complete the graph representation and does not affect solution construction by any means. In MMAS not all the ants deposit pheromone on the arcs, but the one found the best solution in the trial (global-best solution). The pheromone update rule is given by the equation

$$\tau_{it}(A_{i+1}) = \rho \cdot \tau_{it}(A_i) + \Delta\tau_{it}^{best} \quad (6)$$

where $\Delta\tau_{it}^{best} = 1/f(s^{best})$ and $f(s^{best})$ is the cost of global-best solution and (1-ρ) is a parameter that models pheromone evaporation.

4.3 Heuristic information

A very simplistic method utilized in this paper for heuristic information computation is given by the following rule:

$$\eta_{it}(A_i) = \frac{1}{1 + PCV_{it}(M_i)} \quad (7)$$

where $PCV_{it}(M_i)$ is a function that counts the total number problem constraint violations. It is possible to bias each problem constraint using weights which correspond to the relative importance of each constraint (or assign weights based on constraint categories, that is hard and soft constraints). In case of GMS problem constraint (3) is biased by a five times greater weight than constraint (2).

4.4 The algorithm

The MAX-MIN algorithm developed for GMS problem is

Algorithm

Compute T_i

Compute I_t

while termination criteria not met

for each ant do

for each period $t \in T$ do

choose unit $i \in I$ for maintenance

(probabilistic decision)

(unit choice creates a partial mapping of unit i to period t, Mit)

add partial mapping to ant's constructed solution

(constructed solution CS)

end

evaluation of solution (CS) constructed by the ant

(evaluate solution using objective function)

$S_{iteration\text{-}best} = \min \{S_{iteration\text{-}best}, CS\}$

end

$S_{global\text{-}best} = \min\{S_{global\text{-}best}, S_{iteration\text{-}best}\}$

Global update of pheromone trails using $S_{global\text{-}best}$, τ_{max}, τ_{min}.

end

In every iteration each ant constructs a complete solution for the GMS problem, which is a complete mapping of units to periods. Every ant for each period selects one of the available periods (feasible neighbour), which are elements of the set T_i given that it is not already selected (not in tabu list). The T_i is calculated before the execution of the actual algorithm (pre-calculated set). The selection of unit to start maintenance follows the probabilistic rule given in equation (5). Finally, each ant constructs a solution (CS) and the best one is assigned to $S_{iteration\text{-}best}$. After all ants have completed their tours the iteration is considered to be finished and the best solution found among the iterations so far is assigned to $S_{global\text{-}best}$. The components of this solution will be used to update the corresponding pheromone trail values according to equation (6). If any of the pheromone values exceed the limits of τ_{max} and τ_{min}, they are adjusted accordingly. The algorithm continues with other trials, until a specified number of iterations or time limit is reached.

5. Results

The algorithm described in the previous section was created and executed in a PC with Mobile Intel Pentium 4-M CPU 2.00 GHz using IBM WebSphere Studio Application Developer 5.1 with Sun's jre 1.4.2. As a pseudo-random number generator was used the Sun's java.util.Random class [14].

In order to develop a feasible MMAS algorithm for any optimization problem it is essential the appropriate selection of algorithm parameters. Many different combinations of values have been tested on our GMS problem instance and the values produced the best results are provided in Table 2.

Table 2. MMAS algorithm parameter values for GMS problem

	Parameter	Value
ρ	Pheromone evaporation factor	0.2
$\tau_{max} = 1/\rho$	Pheromone value upper limit	5
τ_{min}	Pheromone value lower limit	0.0010
α	Relative importance of pheromone trails	1.0
β	Relative importance of heuristic information	1.0
m	Number of ants	10

The minimization of objective function value for GMS problem – given by equation (1) – during algorithm execution is represented in Figure 3. The algorithm converges at 55th iteration, nevertheless a slight decrease of algorithms' objective function value is observed also in 85th iteration.

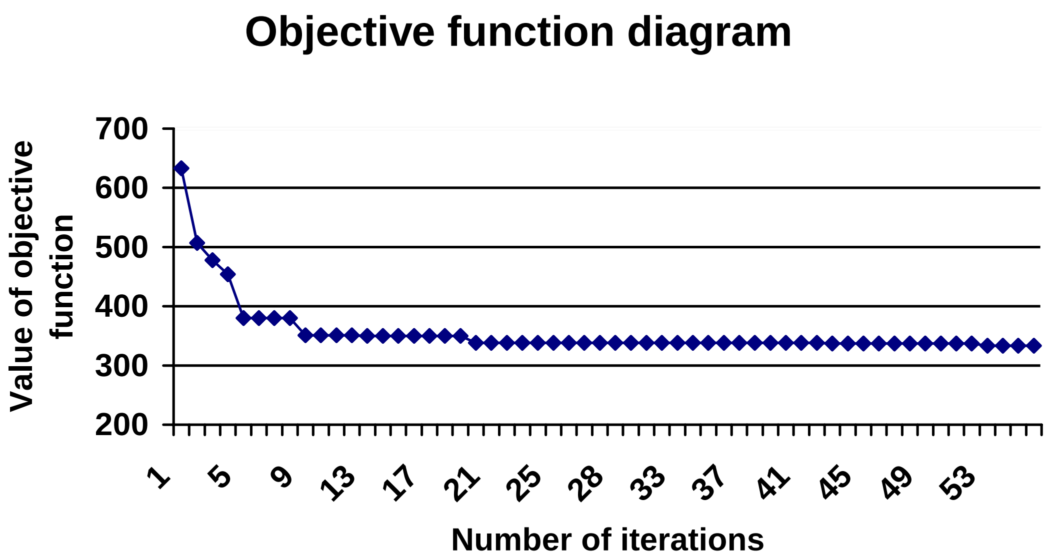

Figure 3. Depiction of objective function. After the 55th iteration the objective function receives the lowest value.

Finally, the pheromone matrix after 101 iterations of the algorithm is presented.

Unit	Period	Pheromone Trail
1	13	4.999999999223381
2	41	4.999999996929478
3	2	5.0
4	11	4.999999956360151
5	29	4.999928655455962
6	5	4.85929769601676
7	15	4.999999999223381
8	34	4.999928655455962
9	1	5.0
10	15	4.999999999223381
11	9	4.999999996929478
12	27	5.0
13	8	4.999999996929478
14	12	4.999999957877433
15	14	4.999999999223381
16	3	4.85929769601676

17	32	4.999992339437917
18	31	4.999999958132063
19	28	4.999992374082709
20	40	4.999999996929478
21	10	4.999999996929478

Table 3. Pheromone matrix after 101 iterations

6. Conclusions

It is fundamental for developing not only MMAS but any ACO algorithm for an optimization problem to convert – represent it as a graph. That was the first step in tackling Generator Maintenance Problem in our case. Next, the solution components (the mapping of a unit and a period that the maintenance of that unit may start) were defined. The pheromone matrix representation and a simple yet convenient method for heuristic information computation was the next step in our approach.

The results above seem to be encouraging for ACO algorithms utilization for solving Generator Maintenance Problem. The next step could be the exploitation of a local search algorithm, such as 2-opt, 3-opt, Tabu Search, Simulated Annealing, within the MMAS in order to obtain even better results.

References:

[1] Yong, Huasong: Modern Optimization Techniques in Power Systems, 1999 Kluwer *Academic Publishers* pp 77-86.

[2] Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Transaction on Systems, Man and Cybernetics 26(1996) 29-41.

[3] M. Dorigo, Optimization, learning and natural algorithms. Ph.D.Thesis, DEI, Politecnico di Milano, Italy, 1992.

[4] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for distributed discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[5] Dorigo, M., Gambardella LM: Ant Colonies for Traveling Salesman Problem. Biosystem 1997: 43(1): 73-81

[6] M.Dorigo and L.M. Gambardella. Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[7] V. Maniezzo and A. Colorni: The Ant System: Applied to Quadratic Assignment Problems. IEEE Transactions on Knowledge and Data Engineering, Vol.11, 769-778, 1999.

[8] T. Stutzle, H. H. Hoos: *MAX-MIN* Ant System. *Elsevier Science*, 2000.

[9] Edmund K. Burke, John A. Clark, Alistair J. Smith, "Four methods for maintenance scheduling", In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, pp264–269.

[10] Dahal K.P., and McDonald J.R. (1997). A review of generator maintenance scheduling using artificial intelligence techniques, *Proceedings of 32nd Universities Power Engineering Conference* (UPEC'97), 787-790.

[11] Dahal K.P., and McDonald J.R. (1997). Generator maintenance scheduling of electric power using genetic algorithms with integer representation, *Proceedings of 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA' 97),* IEE Conference Publication 446,456-461

[12] Kim H., Hayashi Y., Nara K. (1997). An algorithm for thermal unit maintenance scheduling through combined use of GA, SA and TS, *IEEE Transactions on Power Systems* 12, 329-335.

[13] Niebur, D. and Dillon, T.S., "Artificial Neural Networks for Power Systems: State of the Art", *Neural Networks Applications in Power Systems*, T. S. Dillon and D. Niebur, eds., CRL Publishing Ltd., London, 1996, pp. 37-110.

[14] Overview (Java 2 Platform SE v1.4.2), http://java.sun.com/j2se/1.4.2/docs/api