Optimizing Large- Scale Combinatorial Problems using Max-Min Ant System Algorithm

ARISTIDIS VLACHOS
Department of Informatics
University of Piraeus
80 Karaoli and Dimitriou Street, 18534, Piraeus
GREECE
AVlachos@unipi.gr

Abstract: - The maintenance scheduling of thermal generators is a large-scale combinatorial optimization problem with constraints. In this paper we introduce the Max-Min Ant System based version of the Ant System. This algorithm reinforces local search in neighborhood of the best solution found in each iteration while implementing methods to slow convergence and facilitate exploration.Max-Min Ant System (MMAS) algorithm has been proved to be very effective in finding optimum solution to hard combinational optimization problems. To show its efficiency and effectiveness, the proposed Max-Min Ant System is applied to a real-scale system, and further experimenting leads to results that are commented.

*Keywords:*Thermal Generator Maintenance Scheduling Problem, Ant Colony Optimization, Max-Min Ant System.

1. Introduction

The Thermal Generator Maintenance Scheduling Problem is a complex multivariable problem that is necessary for the reliability and right operation of a generator system, given that the whole production cost is dependent on the maintenance and operation cost. Thus, the maintenance procedure has to be scheduled and complied with the best possible way, minimizing these two costs and at the same time, covering the energy demands, so as every constraint of the problem is satisfied.

The problem has been studied in the past with a variety of modeling methods. The initial formulation was made by Gruhl [1], [2]. He presented an umbrella of scheduling problems, one of which was the generator maintenance scheduling problem, with a linear approach.

Two years later, Dopazo and Merill [3] developed a model which was claimed to have the ability of finding the best solution, but this approach was lacking in real-scale problems application, something that Zurn and Quintana [4] later achieved to do using computational methods.

In 1983, Yamayee and Sidenblad [5] improved the cost function that was used till then, with great improvements in execution time.

In 1991, Satoh and Nara [6] applied for the first time a stochastic method, called Simulated Annealing with very good results in large-scale systems as well, that were impossible to be solved with linear methods in the past. They also investigated the problem with genetic algorithms [7] and tabu-list methods [8] with similar results, but with the ability to solve real-scale problems, too.

In 1993, Charest and Ferland [9] tried to modify the linear method with successful results in execution time, while Dahal and McDonald [10] applied a genetic algorithm in Boolean

representation [11] which had also some good results. In 1997, Burke and Smith [12] tried to create a hybrid model of the simulated annealing and the tabu-list method without success, following another attempt to make another hybrid model with memetic and tabu-list methods three years later, which resulted in better results, but with a small increase in execution time.

In 2010 Y.Yare., G.K. Venayagamoorthy (13) using multiple swarms-MDPSO framework with good results for Optimal maintenance scheduling of generators problem.

In 2011, Saraiva, Pereiva, Mendes, and Sousa (14) solved the generator maintenance scheduling problem using a simulated annealing algorithm.

In this paper, we introduce Max-Min Ant System Algorithm [15], an imported version of basic Ant System [16] of the family algorithms: Ant Colony Optimization (ACO) [17], which was inspired by the observation of ant colonies.

This paper is composed of the following sections: Section 2, describes general Ant Colony Optimization,Ant System(AS) and Max-Min Ant System(MMAS) algorithms. In section 3, we represent the formulation of the Thermal Generator Maintenance Scheduling problem. In section 4, we represent the implementation of MMAS for the problem and the algorithm used. The paper ends with case studies on a real system in section 5 and conclusions in section 6.

2. ANT COLONY OPTIMIZATION

2.1. Generally Analogy

The Ant Colony Optimization (ACO) [18] is a metaheuristic to solve combinatorial optimization problems, is motivated by the behavior of real ant colonies. When ants attempt to find short paths between their nest and food sources, they communicate indirectly by using pheromone (pheromone trail) to mark the decisions they made when building their respective paths. Within ACO algorithms, the optimization problem is represented as a complete weighted graph G = (N,A) with N being the set of nodes and A the set of edges fully connecting the nodes N. In the Travelling Salesman Problem (TSP) application, edges have a cost associated (e.g. their length) and the problem is to find a minimal-length closed tour that visits all the nodes once and only once. In order to solve the problem, random walks of a fixed number of ants through the graph take palce. The transition probabilities of each ant are governed by two parameters associated to the edges of the graph: the pheromone values (or pheromone trail) τ_{ij}, representing the learned desirability of choosing node j when in node i. inverse of the distance between two nodes i and j: $n_{ij} = \frac{1}{d_{ij}}$ where d_{ij} is the distance between these two nodes.

The more distinctive feature of ACO is the management of pheromone trails that are used, in conjunction with the objective function, to construct new solutions. Informally, the pheromone trails are used for exploration and exploitation. Exploration representing the probabilistic choice of the components used to construct a solution. A higher probability is given to elements with a strong pheromone trail. Exploitation is based on the choice of the component that maximizes ablend of pheromone-trail values and partial objective function evaluations. The mathematical formulations of the ACO algorithms presented in this paper named Ant System (AS) and Max-Min Ant System (MMAS), are given in the following sections.

2.2. Ant System

Ant System (AS) [19] is the original and most simplistic ACO algorithm. The decision policy used within AS is as follows: The probability with which ant k, currently at node i, chooses to go to node j is given [16] by:

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [n_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}(t)]^{\alpha} \cdot [n_{il}]^{\beta}} \quad (1)$$

If j ϵJ_i^k and 0 if $j \notin J_i^k$

J_i^k : is the feasible neighborhood of ant k, that is, the set of nodes which ant k has not yet visited.

$\tau_{ij}(t)$: is the concentration of pheromone associated with edge (i,j) in iteration t.

n_{ij} : is the inverse of the length of the edge known as visibility

α and β: are parameters that control the relative importance of pheromone intensity versus visibility

Upon conclusion of an iteration (i.e. each ant has generated a solution) the pheromone on each edge is updated, according to the following formula:

$$\tau_{ij}(t) = \rho \tau_{ij}(t) + \Delta \tau_{ij}(t) \quad (2)$$

Where ρ is the coefficient representing pheromone persistence ($0 \le \rho < 1$), and $\Delta\tau_{ij}$, is a function of the solutions found at iteration t, given by:

$$\Delta \tau_{ij} = \sum_{k=1}^{n} \Delta \tau_{ij}^{k}(t) \quad (3)$$

n: number of ants

$\Delta\tau_{ij}^{k}$:is the quantity per unit of length of pheromone addition laid on edge (i,j) by the k[th] ant at the end of iteration t, is given by:

$$\Delta \tau_{ij}^{k}(t) = \begin{cases} Q/L^{k}(t), & \text{if } (i,j) \in T^{k}(t) \\ 0, & \text{if } (i,j) \notin T^{k}(t) \end{cases} \quad (4)$$

Where $T^k(t)$ is the tour done by ant k at iteration t, $L^k(t)$, is its length and Q is a constant parameter, used for defining to be of high quality solutions with low cost.

2.3. Max-Min Ant System

Max-Min Ant System (MMAS) [15], is a direct improvement over AS. The solutions in MMAS are constructed in exactly the same way as in AS, that is, the selection probabilities are calculated as in Equation (1).

The main modifications by MMAS with respect to AS are the following:

(i) To exploit the best solutions found, after each iteration only one single ant is allowed to add pheromone

(ii) To avoid search stagnation, the allowed range of the pheromone trail strengths is limited to interval $[\tau_{\min}(t), \tau_{\max}(t)]$, that is $\tau_{\min}(t) \le \tau_{ij}(t) \le \tau_{\max}(t)$

(iii) The pheromone trails are initialized to the upper trail limit, which causes a higher exploitation at the start of the algorithm. The upper bound, $\tau_{\max}(t)$, is given by:

$$\tau_{\max}(t) = \frac{1}{(1-\rho) Cost_{opt}(t)} \quad (5)$$

where $Cost_{opt}(t)$, is the optimal solution value for a specific problem.

The lower bound $\tau_{\min}(t)$, is given by:

$$\tau_{\min}(t) = \frac{\tau_{\max}(t)(1-\sqrt[n]{p_{best}})}{(\lambda - 1)\sqrt[n]{p_{best}}} \quad (6)$$

Where p_{best} is the probability of creating the global-best solution. This parameter is defined from the user. If p_{best}=1 then $\tau_{\min}(t)$=0. Also if p_{best} is very small there is a probability to use $\tau_{\min}(t) > \tau_{\max}(t)$. In the case we set $\tau_{\min}(t) = \tau_{\max}(t)$ and this algorithm uses only this heuristic information for solving the problem, n is the number of decision points and λ is the average number of edges at each decision point.

3. Formulation of the Problem

The objective of the Thermal Generator Maintenance Scheduling Problem is the maintenance of the energy production units of a system in a given horizon, with the lowest possible cost.

The list of symbols that describe the problem is as follows[20,21]:

i : Number of generator
I : Number of generators
j : Number of week
x_i : Maintenance start period; $x_i \in \{1,2, \ldots,J\}$
J : Horizon in weeks
X_i : Set of proposed maintenance start periods in weeks
M_i : Maintenance length in weeks
Y_{ij} : State variable:

$$Y_{ij} = \begin{cases} 1, \textit{if unit} - i \textit{is in maintenance at period } j \\ 0, \textit{otherwise} \end{cases} \quad (7)$$

p_{ij} : power output of unit-I at period-j
f_i : fuel cost coefficient (linear cost function)
$c_i(x_i)$: maintenance cost of unit-I when the maintenance is committed at period x_i
P_i : capacity of unit i
D_j : anticipated power demand at period-j
R_j : required power reserve at period-j

The generator maintenance scheduling problem is formulated as shown below:

Objective function

The objective is to minimize the objective function which is the sum of the following two terms:

$$Min \left\{ \sum_{i=1}^{I} \sum_{j=1}^{J} f_i \cdot p_{ij} + \sum_{i=1}^{I} c_i(x_i) \right\} \quad (8)$$

Where the first term is the production cost and the second is the maintenance cost.

Constraints

1) The nominal starting period of maintenance is pre-specified for each generating unit:

$$x_i \in X_i \subseteq \{1,2,...,J\} \quad (9)$$

2) Once the maintenance of unit-I starts, the unit must be in the maintenance state for just M_i periods:

$$Yij = \begin{cases} 0, \ j = 1,2,..., x-1 \\ 1, \ j = x_i,..., x_i + M_i - 1 \\ 0, \ j = x_i + M_i,..., J \end{cases} \quad (10)$$

3) If unit-i_1 and unit-i_2 cannot be maintained in a given week because of the crew constraint, the following constraint is imposed:

$$Y_{(i1)j} + Y_{(i2)j} \le 1, \ j = 1,2,...,J \quad (11)$$

4) If the maintenance of unit-i_1 must be finished prior to the starting of that of unit-i_2, the following constraint is added:

$$x_{i1} + M_{i1} \le x_{i2} \quad (12)$$

5) The generator output must be less than its upper limit; and the output of the

generator in maintenance must be equal to zero. Such an operation constraint is expressed by:

$$0 \le p_{ij} \le P_i \cdot (1 - y_{ij}), i = 1,...I, j = 1,...J \quad (13)$$

6) The demand constraint must be met:

$$\sum_{i=1}^{I} p_{ij} = D_j, j = 1,2,...,J \quad (14)$$

7) In order to ensure that the total available power is greater than the demand D_j even when a unit random outage occurs, the reserve constraints are imposed. That is, the total available power from units which are not committed must be greater than the demand plus reserve:

$$\sum_{i=1}^{I} P_i \cdot (1 - y_{ij}) \ge D_j + R_j, j = 1,2,...,J \quad (15)$$

Penalty Function

In the generator maintenance scheduling problem, the constraints are classified into two groups; "easy" constraints and "difficult" constraints. The easy constraints are equations (9), (10), (12), (13), the difficult constraints are equations (11), (14), (15). Since the set X_i is given, the value of x_i can be selected as a member of X_i so that equations (9), (12) are satisfied. Then the value of y_{ij} is directly defined by equation (10), and equation (13) becomes a simple bound on p_{ij}. On the other hand, it is very difficult to find a feasible solution which satisfies equations(11), (14) and (15). So, the artificial variables z_i, u_i, and v_i are introduced corresponding to equations (11), (14) and (15), with associated positive penalty parameters α, β, and γ. Then the problem is re-formulated as follows:

$$Min \left\{ \begin{array}{l} \sum_{i=1}^{I} \sum_{j=1}^{J} f_i \cdot p_{ij} + \sum_{i=1}^{I} c_i(x_i) \\ + \alpha \cdot \sum_{j=1}^{J} z_i + \beta \cdot \sum_{j=1}^{J} u_i + \gamma \cdot \sum_{j=1}^{J} v_i \end{array} \right\} \quad (16)$$

$$x_i \in X_i \subseteq \{1,2,...,J\} \quad (17)$$

$$\mathrm{Y}ij = \begin{cases} 0, & j = 1,2,..., x - 1 \\ 1, & j = x_i ,..., x_i + M_i - 1 \\ 0, & j = x_i + M_i ,..., J \end{cases} \quad (18)$$

$$Y_{(i1)j} + Y_{(i2)j} - z_j \le 1, \; j = 1,2,...,J \quad (19)$$

$$x_{i1} + M_{i1} \le x_{i2} \quad (20)$$

$$0 \le p_{ij} \le P_i \cdot (1 - y_{ij}), i = 1,...I, j = 1,...J \quad (21)$$

$$\sum_{i=1}^{I} p_{ij} + u_j = D_j, j = 1,2,...,J \quad (22)$$

$$\sum_{i=1}^{I} P_i \cdot (1 - y_{ij}) + v_j \ge D_j + R_j, j = 1,2,...,J \quad (23)$$

$$z_n \in \{0,1\} \quad (24)$$

$$u_j, v_j \ge 0_j, j = 1,2,...,J \quad (25)$$

By using the above formulation, once the value of x_i is determined, the value of y_{ij} is directly defined, and the value of p_{ij} is calculated through the equal incremental method for the economic dispatch problem [22]. Therefore, the value of the objective function can be efficiently evaluated if the value of x_i is specified.

3. Implementation of MMAS for the Generator Maintenance Scheduling Problem

a. Expression approach

For the implementation of the problem, we used a sort of graph. Every node of the

graph represents a feasible solution, and more specifically a feasible week that the maintenance of every generator can be started. In this way, every ant traverses one by one the generator units, choosing one of the feasible maintenance starting periods and, in the end, constructing a complete solution. When all ants complete their tours, the iteration is completed and a new one takes place.

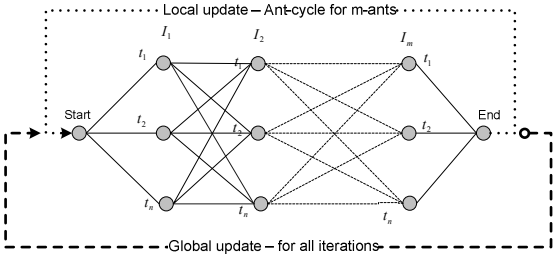

Figure 1 – Representation of the problem using graph

So, on every step, all units are selected, and the total cost of the solution is calculated, summing up the total maintenance cost, plus the total generator operation cost needed for every week (plus the penalty of erroneous solutions, if any).

When the k[th] ant is on town i, the probability to move to town j, is given by the equation:

$$p^k_{ij}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{j\in j_i^k(t)}[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}, & if\ j\in j_i^k(t)\\ 0, & if\ j\notin j_i^k(t)\end{cases} \quad (26)$$

where j_i^k are the towns that are not yet included on the agent's tabu list. As visibility η_{ij} between towns, we will use the equation $\eta_{ij}(t)=\dfrac{1}{1+PCV_{ij}(t)}$ where $PCV_{ij}(t)$ is a method counting the total number of the Problem Constraint Violations. We will bias each constraint violation using weights which correspond to the relative importance of each constraint. Thus, each ant will be guided as to not choose the towns that violate the problem's constraints.

Pheromone Update Rule: $\tau_{ij}(t)$ is the pheromone quantity that is found in edge that connects every generator (except the reference generator) with power level. Because the algorithm that was used is the MMAS some notifications must be made concerning the pheromone calculation:

(i) Renewal of pheromone takes place from every ant in every iteration. Either from the one that has found the global best solution (global best ant) or from the one that has found the best solution in an iteration (iteration best ant). These two mechanisms can be combined.

(ii) First of all $\tau_{ij}(t)$ is the quantity of pheromone on the edge that connects machine i with its power level j. At the beginning this quantity should be equal to $\tau_{\max}$, but since this has not been calculated yet we have to use a large value τ_0 and after the first iteration we must set all pheromone trails equal to $\tau_{\max}$. The resulting pheromone update rule is:

$$\tau_{ij}(t+1)=\rho\tau_{ij}(t)+\Delta\tau_{ij}(t) \quad (27)$$

Where

$$\Delta\tau_{ij}(t)=\begin{cases}1/Cost^*(t), & \text{if } (i,j)\in T^*(t)\\ 0, & \text{if } (i,j)\notin T^*(t)\end{cases}$$

$Cost^*(t)$ is either the global best solution so far ($Cost_{best}(t)$), or the best solution during the current iteration ($Cost_{iter}(t)$) and $T^*(t)$ is the list that keeps track for the best solution.

ρ with $0\leq\rho\leq1$ is the evaporation coefficient

After this step it is checked if the pheromone trails are within the limits $\tau_{\min}$, $\tau_{\max}$ and finally the pheromone is updated according to the following relationship:

$$\tau_{ij} \leftarrow \begin{cases} \tau_{\min} & \text{if } \tau_{ij} < \tau_{\min} \\ \tau_{\max} & \text{if } \tau_{ij} > \tau_{\max} \\ \tau_{ij} & \text{otherwise} \end{cases} \quad (28)$$

The upper bound $\tau_{\max}(t)$ and the lower bound $\tau_{\min}(t)$ are given by Equation (5) and Equation (6) respectively.

(iii) Smoothing of pheromone (optional step): When the algorithm converges, the following mechanism can be activated, that increases pheromone levels depending on the difference from $\tau_{\max}$, so that the selection possibility of trails with low pheromone levels is minimized. This can take place as following:

$$\tau_{ij}^{*}(t) = \tau_{ij} + \delta(\tau_{\max}(t) - \tau_{ij}(t)) \quad (29)$$

where $\tau_{ij}^{*}(t)$ and $\tau_{ij}(t)$ are the pheromone trails before and after smoothing,

δ is a parameter set by the user and $0 \leq \delta \leq 1$. For δ=1 there is a reinitialization of pheromone levels and for δ=0, this mechanism becomes inactivate.

b. The algorithm

1. Define problem parameters for each agent and generator.
2. Calculation of a first solution for each unit.
3. Evaluation of every solution constructed by each ant

$Cost_{best} = \min \{ Cost_{best} , Cost_{iteration\text{-}best} \}$

4. Renew pheromone using the pheromone update rule, and also $\tau_{\max} - \tau_{\min}$ levels.
5. . Repeat algorithm from Step 3
6. (Optional Step).When the algorithm seems to converge,smoothing of pheromone trail can take place.

4. Case study on a real-scale system of generators

The algorithm just described, was implemented on a real scale system of generator units [23] with 22 power generator units that have to be maintained within a 52-week horizon.

The following table shows the parameters that describe every generator unit's operation and maintenance:

Table 1 – System Parameters

I	R_i	E_i	L_i	M_i	a	b	c	f_i	**Crew constraint for every maintenance week**
1	100	1	47	6	70	8.00	0.00585	0.25	10+10+10+5+5+5
2	100	1	50	3	70	8.00	0.00580	0.20	15+15+15
3	100	1	50	3	70	8.00	0.00580	0.20	10+15+15
4	100	1	50	3	70	8.00	0.00580	0.20	10+10+10
5	90	1	47	6	60	8.00	0.00610	0.35	10+10+10+5+5+5
6	90	1	49	4	60	8.00	0.00610	0.30	10+10+10+10
7	95	1	50	3	68	8.00	0.00579	0.20	10+10+10
8	100	1	49	4	72	8.00	0.00565	0.20	10+10+5+5
9	650	27	48	5	525	7.00	0.00120	0.52	10+10+10+5+5
10	610	6	11	12	510	7.20.	0.00142	0.50	3+2+2+2+2+2+2+2+2+2+2+3
11	91	1	49	4	62	8.25	0.00600	0.20	10+10+10+10
12	100	1	45	8	74	8.15	0.00578	0.30	10+10+5+5+5+5+5+3
13	100	1	50	3	70	8.00	0.00580	0.20	15+15+15
14	100	1	47	6	70	8.00	0.00585	0.25	10+10+10+5+5+5
15	220	1	48	5	85	7.90	0.00460	0.25	10+10+10+10+10
16	220	1	47	6	87	7.95	0.00464	0.25	10+10+10+5+5+5
17	100	1	48	5	69	8.18	0.00570	0.20	10+10+10+10+10
18	100	1	48	5	69	8.17	0,00572	0.25	10+10+10+5+5
19	220	1	50	3	81	7.90	0.00463	0.25	10+10+10
20	220	1	50	3	82	7.95	0.00462	0.25	10+15+15
21	240	1	50	3	82	7.40	0.00410	0.30	15+15+15
22	240	1	48	5	80	7.42	0.00415	0.30	10+10+10+5+5

Table 2 - Weekly demand

j	Demand D_j	j	Demand D_j
1	1694	27	1737
2	1714	28	1927
3	1844	29	2137
4	1694	30	1927
5	1684	31	1907
6	1763	32	1888
7	1663	33	1818
8	1583	34	1848
9	1543	35	2118
10	1586	36	1879
11	1690	37	2089
12	1496	38	1989
13	1456	39	1999
14	1396	40	1982
15	1443	41	1672
16	1273	42	1782
17	1263	43	1772
18	1655	44	1556
19	1695	45	1706
20	1675	46	1806
21	1805	47	1826
22	1705	48	1906
23	1766	49	1999
24	1946	50	2109
25	2116	51	2209
26	1916	52	1779

where:

R_i the highest level of energy can be produced.

E_i and L_i the earliest and latest period that the maintenance can start.

M_i the maintenance period length (in weeks).

a, b, c, the cost parameters for the operation of the generators.

f_i the fuel cost coefficient (linear function).

and, finally, the maintenance crew needed for every maintenance week.

The minimum level of energy that can be produced from each generator is zero.

Table 2, also, represents the anticipated demand of the system for every week within the horizon.

It is worth mentioning that the required reserve for each week of the horizon can be defined using one of the following approaches:

i. As a constant percentage of the energy demand, D_j.

ii. As equal to the size of the largest generating unit.

iii. In dependence of other necessary criteria.

Here, we applied the first approach, with a 20% percentage on demand D_j.

That is: $R_j = 20\% \cdot D_j, j = 1,2,...,J$

It is important to define some determinant parameters for the solution of the problem from the beginning. The following executions of the problem are looking into the following matters:

- As we are working on a real system, it is easy to appreciate the fact that we need a maintenance crew constraint that will be:
  - "Flexible", concerning the solutions that can be produced, without confining them.
  - Big enough to produce solutions without violations-penalties and, respectively, non-feasible.
  - Small enough so as to minimize the existence of not needed crew.

For all these reasons, after close study of the problem constraint table and the solutions produced, the crew number was set to 30.

- As we can see, the objective function describes the constraint violations as

extra cost added to the production cost. As these solutions are not feasible, we have to define the size of the parameters α, β and γ to be analogous with the solution cost of the problem, so as to be added an extra cost feasible to reject them. After experimental executions, we found that a solution without violations is in the order of hundreds millions cost units (10^8). So, forasmuch as each violation can occasionally occur, the parameters were defined as follows:

- α = 100
- β = 100
- γ = 20

The definition of the weight parameter α in regard to the weight parameter β, is also essential. The following values were tested with β=1: represent the results given by experimental executions.

Table 3 – α versus Best solution Best Cost

α	Best solution
0.01	3.34650000
0.02	3.34320000
0.03	3.34230000
0.05	3.33560000
0.1	**3.32540000**
0.2	3.34120000
0.3	3.34480000
0.4	3.34670000
0.6	3.34820000
0.8	3.34510000
1	3.35100000

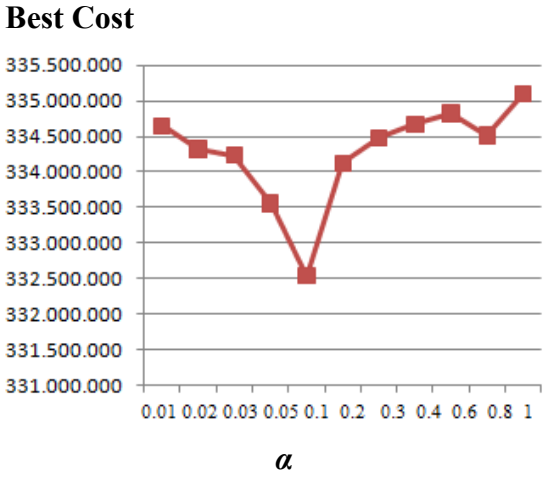

Figure 2 - α versus Best Cost

A graphical representation of pbest versus Best Cost is shown in the Figure 3.

Table 4 – pbest versus Best cost Best Cost

pbest	Best cost
0.2	3.34470000
0.4	3.34100000
0.6	**3.32540000**
0.8	3.34210000
1	3.34520000

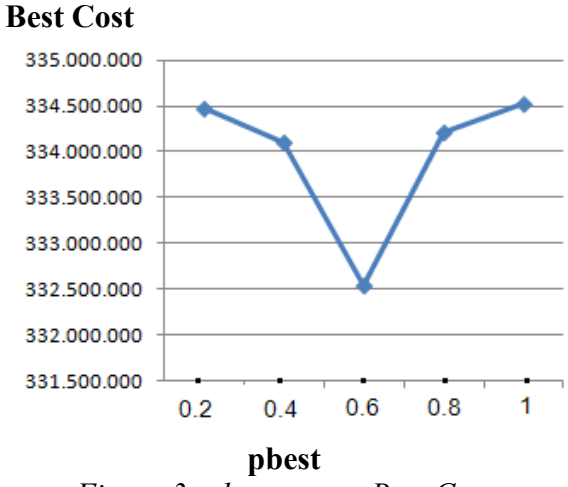

Figure 3- pbest versus Best Cost

The following parameters were chosen The following parameters ,for MMAS algorithm,were chosen for the resolution of the problem:

- $\alpha = 0.2$
- $\beta = 1$
- $\rho=0.3$
- pbest = 0.7

The execution of the proposed method was run on an AMD Athlon 3000+ 1.79 GHz processor giving the following results:

- The best solution found is

[1, 16, 12, 21, 43, 3, 28, 6, 33, 7, 17, 10, 44, 36, 47, 27, 29, 38, 33, 28, 7, 40]

That is the period that maintenance for every unit of the system can start .

The best solution cost found is 3.32540000.

- The maintenance periods are represented in detail on the following diagram:

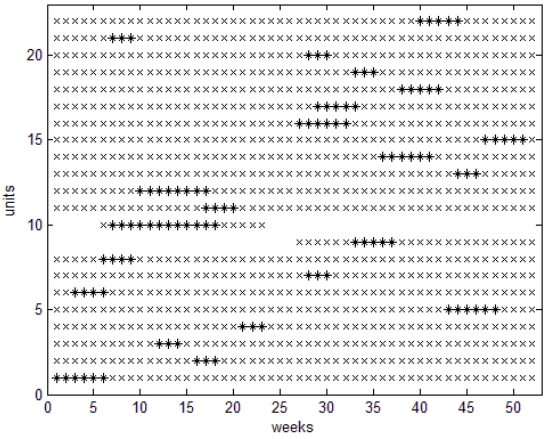

Figure 4 – Maintenance periods of Best solution

where the feasible maintenance periods are represented with "x" and the selected maintenance periods with "*" according to the best solution.

- The best solution was found on iteration number 72.
- The best solution progress versus iterations is represented to the following figure:

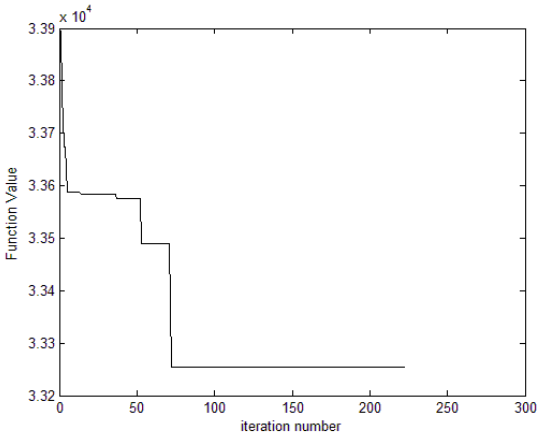

Figure 5 – Best solution cost versus Iterations

- The total execution time is 0h:2m:36s.

5. Conclusion

This paper looked into the Thermal Generator Maintenance Scheduling Problem of a real-scale system of energy production units. The problem has been studied with many mathematical and heuristic approaches in the past. In this project, we seek better results using the Max-Min Ant System algorithm which belongs to the Ant Colony Optimization algorithms.

The results produced prove that the algorithm can be applied successfully to the problem so as the optimum solutions can be found, even in real energy production systems where the complexity raises significantly, because of the number of generating units, but also due to the number of feasible solutions that have to be produced in a reasonable time interval.

References

[1] J. Gruhl, "*Electric generation production scheduling using a quasioptimal sequential technique*," Research Note MIT-EL 73-003, MIT Energy Lab, April 1973.

[2] J. Gruhl, "*Electric power unit commitment scheduling using a dynamically evolving* mixed *integer program*," Research Note MIT-EL 73-007, MIT
Energy Lab, January 1973.

[3] J. F. Dopazo and H. M. Merrill, "*Optimal generator maintenance scheduling using integer programming*," IEEE Transactions on Power Apparatus and Systems, vol. PAS-94, no. 5, pp. 1537–1545, 1975.

[4] H. H. Z¨urn and V. H. Quintana, "*Generator maintenance scheduling via successive approximations dynamic programming*," IEEE Transactions on Power Apparatus and Systems, vol. 94, no. 2, pp. 665–671, 1975.

[5] Z. A. Yamayee, K. Sidenblad, and M. Yoshimura, "*A computationally efficient optimal maintenance scheduling method*," IEEE Transactions on Power Apparatus and Systems, vol. 102, no. 2, pp. 330–338, 1983.

[6] T. Satoh and K. Nara, "*Maintenance scheduling by using the simulated annealing method*," IEEE Transactions on Power Systems, vol. 6, pp. 850–857, 1991.

[7] H. Kim and K. Nara, "*A method for maintenance scheduling using GA combined with SA*," Trans. IEE Japan, vol. 115-B, no. 11, 1995.

[8] H. Kim, Y. Hayashi, and K. Nara, "*An algorithm for thermal unit maintenance sched*uling *through combined use of GA, SA and TS*," IEEE Transactions on Power Systems, vol. 12, no. 1, pp. 329–335, 1997.

[9] M. Charest and J. A. Ferland, "*Preventative maintenance scheduling of power generation units*," Annals of Operations Research, vol. 41, pp. 185–206, 1993.

[10] K. P. Dahal and J. R. McDonald, "*Generational and steady state genetic algorithms for generator maintenance scheduling problems*," Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, pp. 260–264, 1997.

[11] K. P. Dahal and J. R. McDonald, "*Generator maintenance scheduling of electric power systems using genetic algorithms with integer representation*," Submitted to GALESIA'97, 1997.

[12] E. K. Burke, J. A. Clark, and A. J. Smith, "*Four methods for maintenance scheduling*," in Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms. 1997, pp. 264–269, Springer.

[13] Y.Yare.,G.K.Venayagamoorthy."*Optimal maintenance scheduling of generators using multiple swarms-MDPSO framework*".Egnineering Applications of Artificial Intelligence.23(2010).pp.895-910.

[14] J.T.Saraiva,M.L.Pereira,V.T.Mendes,J.C.Sousa."*A simulated Annealing based approach to solve the generator maintenance scheduling problem*.Electric Power Systems Research.81(2011).pp.1283-1291.

[15] Stutzle T., and Hoos H.H, Max-Min Ant System, Future Generation Computer Systems, (16), (2001), 889-914.

[16] Bonabenn E, Dorigo M., Theraulaz G., Swarm Intelligence from natural to Artificial systems, Sante Fc Institute studies in the Sciences of complexity, (1999), Oxford University Press

[17]. Dorigo M., Di Caro G., The Ant Colony Optimization Meta-Heuristic, In D.Come, M.Dorigo, and F.Glover, editors, New Ideas in optimization, Mc Graw-Hill.

[18] Dorigo M., Gambardella L.M, Ant Algorithms for Discrete Optimization, Artificial Life, volume 5, no.2, (1992), 137-172 .

[19] Dorigo M., Maniezzo V., and Colorni, The ant system: optimization by a colony of cooperating agents, IEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics, 26(1), (1996), 29-44.

[20] Burke E. K., Smith A. J., *Hybrid Evolutionary Techniques for the Maintenance Scheduling Problem*, IEEE Transactions on Power Systems, 2000, 15(1), pp.1-2.

[21] Satoh T., Nara K., *Maintenance scheduling by using the simulated annealing method*, IEEE Transactions on Power Systems, 1991, 6, p. 850-857.

[22] A. J. Wood and B. F. Wood and B. F. Woolenberg, *Power Generation, Operation, and Control*, John Wiley, 1984.

[23] Ibrahim El-Amin, Salif Duffuaa, Mohammed Abbas, 1998-1999, *A tabu search algorithm for maintenance scheduling of generating units*, King Fahd University of Petroleum and Minerals