Leonardo Journal of Sciences
ISSN 1583-0233

Issue 12, January-June 2008
p. 143-164

Maintenance Scheduling by using the Bi-Criterion Algorithm of Preferential Anti-Pheromone

Triantafyllos MYTAKIDIS and Aristidis VLACHOS

Department of Informatics, University of Piraeus, 80, Karaoli & Dimitriou Str., 18534 Piraeus, Greece
E-mails: T.Mytakidis@gmail.com, AVlachos@unipi.gr

Abstract

This paper presents the solution for the Thermal Generator Maintenance Scheduling Problem, using a bi-criterion Ant Colony Optimization Algorithm called Preferential Anti-pheromone (PAP). This method allows the "agents" of an ant colony to deposit a small amount of pheromone trail to every path that has been used, to construct the potential solutions, but also to give extra emphasis to the best solution found at the end of an iteration of the algorithm. In the same time that good solutions are being investigated from the agents, bad solutions are examined too, with the aim to avoid short-term poor solutions and lead to long-term good and, respectively, global best solutions. In this way, through the iterations of the algorithm, we end up to the final solutions. The algorithm is applied to a real-scale problem, and further investigation is being made so as to find the best possible solution.

Keywords

Thermal Generator; Maintenance Scheduling; Ant Colony Optimization; Ant Colony System; Preferential Anti-Pheromone.

Introduction

The Thermal Generator Maintenance Scheduling Problem is a complex 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 methods and algorithms [1]. The initial formulation was made by Gruhl [2, 3] in 1973. He presented an umbrella of scheduling problems, one of which was the maintenance scheduling problem, with a linear approach.

During 1975, Dopazo and Merill [4] 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 [5] later achieved to do.

In 1983, Yamayee and Sidenblad [6] improved the cost function that was used till then, with great improvements in execution time.

Eight years later, Satoh and Nara [7] 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 [8] and tabu-list methods [9] with similar results, but with the ability to solve real-scale problems, too.

In 1993 Charest and Ferland [10] tried to modify the linear method with successful results in execution time, while Dahal and Donald [11] applied a genetic algorithm in Boolean representation [12] which had also some good results. In 1997, Burke and Smith [13] 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.

This paper studies the results of the Anti-pheromone method, an ACS-like algorithm based on the behavior of true ants. It is organized as follows: In section 2, we review the principal framework of the ACO, Ant Colony System and Prefferential Anti-Pheromone methods. In section 3, we represent the formulation of the Maintenance Scheduling problem. In section 4, we represent the implementation of PAP 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.

Ant Colony Optimization (ACO)

Overview

Ant Colony Optimization is a pack of Artificial Intelligence algorithms that rely on the imitation of the social insects' behavior, and especially ants'. These algorithms use agents, that we call "ants", for the investigation of the best solution of a problem, for example the shortest path between some places that might be food for the colony, just like happening with the true ant colonies.

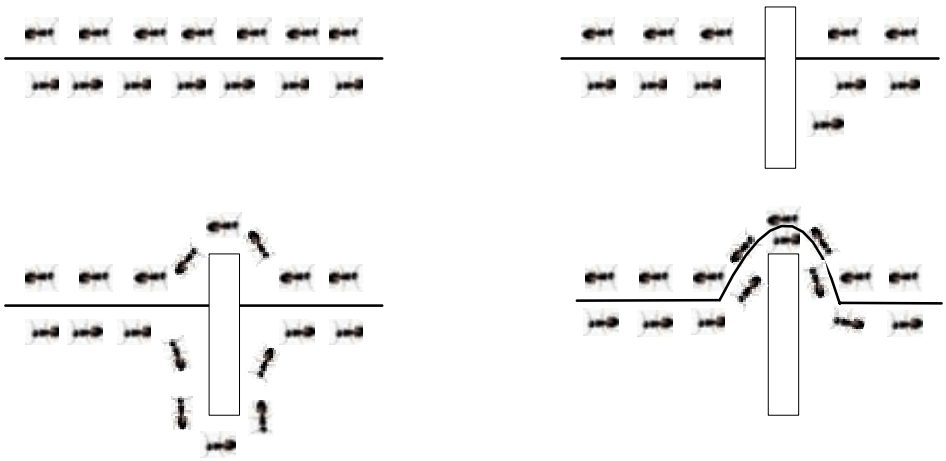

Figure 1. Real ants after a while tend to choose the shortest path between nest and food

These agents, are constructing through iteration the solutions of the problem. The probability for an ant to visit a town is affected from the amount of pheromone that every agent detects during its exploration. Pheromone is a substance that ants produce and deposit along the paths that have traversed, making them more attractive for the next ones that might pass from the same point, while the already existent pheromone is vaporizing as time passes. During the progress of the algorithm, the artificial pheromone is placed after the construction of a complete tour-solution on each and every town that was chosen and visited for the construction of it. In this way, the amount of pheromone is the heuristic information at a given timing point, reflecting the experience of the colony about the feasible solutions of the problem.

If we consider the possible solutions as a graph with the ants moving from town sr to

sr+1, then the pheromone level that exists after the passing of an agent from a feasible solution is given from equation:

$$\tau(s_r, s_{r+1}) = (1-\rho)\cdot\tau(s_r, s_{r+1}) + \Delta\tau(s_r, s_{r+1}) \quad (1)$$

where:

ρ is the pheromone trail evaporation factor

$$\Delta\tau(s_r, s_{r+1}) = \sum_{k=1}^{n} \Delta\tau^k(s_r, s_{r+1})$$

n the number of ants

k the number of each ant.

The probability of the k^{th} ant to follow the path from town sr to sr+1, is:

$$T^k(s_r, a_r, s_{r+1}) = \frac{\tau(s_r, s_{r+1})}{\sum_{i \in N_r^k} \tau(s_r, s_i)} \quad (2)$$

where

N_r^k is a feasible solution when ant k is on node r, and $a_r$the action required for the ant to move from node s_{r+1} tos_r.

Ant Colony System (ACS)

The Ant Colony System (ACS) [14] meta-heuristic algorithm belongs to the umbrella of ACO algorithms. On each step of the algorithm (specifically for the Traveling Salesman Problem - TSP, but respectively for other problems), the transition of an ant from town i to town j, depends on:If town has already been visited. For each ant, a tabu list is being saved, which grows each time it reaches a new town, end empties whenever a full solution for the problem is accomplished. In this way, towns are never visited more than once.

The inverse proportion of the distance between two towns

$$\eta_{ij} = \frac{1}{d_{ij}}$$

that is called visibility, and describes the heuristic preference of an ant between two towns.

- The amount of pheromone τ_{ij} that exists between two towns i and j, on which the experience of the colony is dependant, and describes the preference for town j, according to the experience from previous iterations.

  When the k^{th} ant is on town i, the probability to move to town j, is given from equation:

$$j = \begin{cases} \arg\max_{u \in N^k} \{\tau_{iu} [\eta_{iu}]^\beta\}, \ if \ \ q \le q_0 \\ J \qquad\qquad , \ otherwise \end{cases} \tag{3}$$

where:

÷ q is a random variable uniformly distributed between [0,1]

÷ q_0 is a determining parameter between [0,1]

÷ N^k the number of nodes that are not in tabu list of ant k yet.

J is a town randomly chosen under equation:

$$p^k_{ij} = \begin{cases} \dfrac{\tau_{ij} \ [\eta_{ij}]^\beta}{\sum_{u \in N_i^k} \tau_{iu} \ [\eta_{iu}]^\beta}, & j \in N^k \\ 0 \qquad\qquad , & j \notin N^k \end{cases} \tag{4}$$

where:

÷ β the visibility variable for the transition rule.

÷ N^k the number of nodes that are not in tabu list of ant k yet.

In the end of every iteration the pheromone trail is updated under equation:

$$\tau_{ij}(t+1) \leftarrow (1-\rho)\tau_{ij}(t) + \rho \cdot \tau_0 \tag{5}$$

where:

÷ t the number of iteration taking place

÷ ρ the evaporation trail factor with 0< ρ < 1

÷ τ_0 the initial pheromone amount that is placed on every town.

After completing an iteration (when all ants have created their won feasible solutions) the global update rule is being applied:

$$\tau_{ij}(t+1) \leftarrow (1-\gamma)\tau_{ij}(t) + \gamma \cdot \Delta\tau_{ij}(t) \tag{6}$$

whereγ the global pheromone evaporation factor

$$\Delta\tau_{ij}(t) = \begin{cases} Q \big/ L^t \ , & Town \in BestSolution \\ 0 \ , & otherwise \end{cases}$$

where:

÷ Q the amount of pheromone added

÷ L^t the best solution found till iteration t

In this way, the equation above is applied only to towns that belong to the best solution during the execution (all iterations) and this enables them with a bigger amount of pheromone.

Preferential Anti-pheromone

During the ant exploration in the search space, they gain knowledge concerning the most desirable nodes. This often leads to short-term solutions that are not the best ones that can be found. Randal and Montgomery [15] are taking into account this fact with the "Accumulated Experience Ant Colony" - AEAC using a method that can find which solutions can lead to long-term good solutions. Anti-pheromone is a substance that can have similar effects and is placed in nodes that belong to the worst solutions, information that is lost with other methods.

Iradi, Merkle and Middendorf [16] suggest a method called "Preferential Anti-pheromone" (PAP) that uses pheromone, as much as anti-pheromone for the exploration of the search space. The use of this bi-criterion exploration is improved during the iteration process of the algorithm. This improvement is feasible due to the difference of ants concerning their preference between these two kinds of pheromone. For this reason, we use a parameter λ, which is given for the k^{th} ant by equation (k-1)/(m-1), where k=[1,m]. In this way, we give some ants the opportunity to explore the best, but also to some others the worst solutions.

In this way, the transition rule of an ant from town I to town j that we discussed on classical ACO is:

$$j = \begin{cases} \text{argmax}_{u \in N^k}\{[\lambda\tau_{iu} + (1-\lambda)\tau'_{iu}]\cdot[\eta_{iu}]^{\beta}\}, \; if \;\; q \le q_0 \\ J \qquad\qquad , \quad otherwise \end{cases} \tag{7}$$

where J is a town chosen by equation:

$$p_{ij} = \begin{cases} \dfrac{[\lambda\tau_{il} + (1-\lambda)\tau'_{il}]\cdot[\eta_{il}]^{\beta}}{\sum_{u \in N_i^k}[\lambda\tau_{iu} + (1-\lambda)\tau'_{iu}]\cdot[\eta_{iu}]^{\beta}}, & j \in N^k \\ 0 \qquad\qquad\qquad , & j \notin N^k \end{cases} \tag{8}$$

Pheromone and anti-pheromone are updated in the same way when ants traverse the problem's nodes to form the solutions, as local updating ignores the cost of solutions produced. So, in addition with the local updating rule of classical ACO for pheromone, anti-pheromone is updated according to the following rule:

$$\tau'_{ij}(t+1) \leftarrow (1-\rho)\tau'_{ij}(t) + \rho \cdot \tau_0 \tag{9}$$

After completing a tour and each ant has chosen a feasible solution, pheromone is placed on towns that belong to the best solution according the global updating rule of classic ACS, and pheromone to the worst ones, according to the following rule:

$$\tau'_{ij}(t+1) \leftarrow (1-\gamma)\tau'_{ij}(t) + \gamma \cdot \Delta\tau'_{ij}(t) \qquad (10)$$

where:

$$\Delta\tau'_{ij}(t) = \begin{cases} Q/L^t w, & Town \in WorstSolution \\ 0, & otherwise \end{cases}$$

and $L^t w$ the worst length (cost) solution found on t^{th} iteration.

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, usually in weeks, with the lowest possible cost.

The list of symbols that describe the problem is as follows [1, 7]:

i : Generator number

I : Number of generators

j : Number of week

J : Horizon in number of weeks

x_i: Maintenance start period; $x_i \in \{1, 2, \ldots, J\}$

X_i: Set of preferred maintenance start periods; $X_i: \in \{1, 2, \ldots, J\}$

M_i: : Maintenance length in weeks

Y_{ij}: : State variable;

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

p_{ij} : generator 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 demand at period-j

R_j : required reserve at period-j

The generator maintenance scheduling problem is formulated as shown below:

Objective function

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

$$Min\left\{\sum_{i=1}^{I}\sum_{j=1}^{J} f_i \cdot p_{ij} + \sum_{i=1}^{I} c_i(x_i)\right\} \tag{11}$$

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:

$$x_i \in X_i \subseteq \{1,2,...,J\} \tag{12}$$

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

$$Yij = \begin{cases} 0, & j = 1,2,...,x-1 \\ 1, & j = x_i,...,x_i + M_i - 1 \\ 0, & j = x_i + M_i,...,J \end{cases} \tag{13}$$

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

$$Y_{(i1)j} + Y_{(i2)j} \le 1,\ j = 1,2,...,J \tag{14}$$

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

$$x_{i1} + M_{i1} \le x_{i2} \tag{15}$$

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:

$$0 \le p_{ij} \le P_i \cdot (1 - y_{ij}),\ i = 1,...,I,\ j = 1,...,J \tag{16}$$

6) The demand constraint must be met:

$$\sum_{i=1}^{I} p_{ij} = D_j,\ j = 1,2,...,J \tag{17}$$

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:

$$\sum_{i=1}^{I} P_i \cdot (1 - y_{ij}) \geq D_j + R_j,\ j = 1,2,...,J \quad (18)$$

Penalty Function

In the maintenance scheduling problem, the constraints are classified into two groups; "easy" constraints and "difficult" constraints. The easy constraints are eq (2), (3), (5), (6), the difficult constraints are eq (4), (7), (8). Since the set X_i is given, the value of x_i can be selected as a member of X_i so that eq (2), (5) are satisfied. Then the value of y_{ij} is directly defined by eq (3), and eq (6) becomes a simple bound on p_{ij}. On the other hand, it is very difficult to find a feasible solution which satisfies eq (4), (7) and (8). So, the artificial variables z_i, u_i, and v_i are introduced corresponding to eq. (4), (7) and (8), with associated positive penalty parameters α, β, and γ. Then the problem is re-formulated as follows:

$$Min \left\{ \begin{array}{l} \sum_{i=1}^{I}\sum_{j=1}^{J} f_i \cdot p_{ij} + \sum_{i=1}^{I} c_i(x_i) \\ + \alpha \cdot \sum_{j=1}^{J} z_i + \beta \cdot \sum_{j=1}^{J} u_i + \gamma \cdot \sum_{j=1}^{J} v_i \end{array} \right\} \quad (19)$$

$$x_i \in X_i \subseteq \{1,2,...,J\} \quad (20)$$

$$Yij = \begin{cases} 0, & j = 1,2,...,x-1 \\ 1, & j = x_i,...,x_i + M_i - 1 \\ 0, & j = x_i + M_i,...,J \end{cases} \quad (21)$$

$$Y_{(i1)j} + Y_{(i2)j} - z_j \leq 1,\ j = 1,2,...,J \quad (22)$$

$$x_{i1} + M_{i1} \leq x_{i2} \quad (23)$$

$$0 \leq p_{ij} \leq P_i \cdot (1 - y_{ij}),\ i = 1,...I,\ j = 1,...J \quad (24)$$

$$\sum_{i=1}^{I} p_{ij} + u_j = D_j,\ j = 1,2,...,J \quad (25)$$

$$\sum_{i=1}^{I} P_i \cdot (1 - y_{ij}) + v_j \geq D_j + R_j,\ j = 1,2,...,J \quad (26)$$

$$z_n \in \{0,1\} \quad (27)$$

$$u_j,\ v_j \geq 0_j,\ j = 1,2,...,J \quad (28)$$

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 (equal method) for the economic dispatch problem [1]. Therefore, the value of the objective function can be efficiently evaluated if the value of xi is specified.

Implementation of PAP for the Maintenance Scheduling Problem

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.

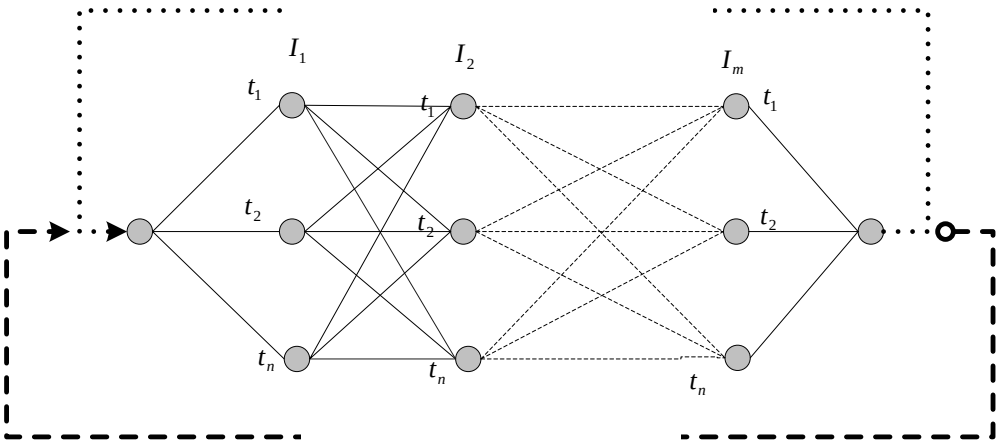

Figure 2. Representation of the problem using graph

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).

The probability for an ant-k that is on town-i to move to town-j, is given by the random-proportional rule of PAP:

$$j = \begin{cases} \text{argmax}_{u \in N^k}\{[\lambda \tau_{iu} + (1-\lambda)\tau'_{iu}] \cdot [\eta_{iu}]^{\beta}\}, \textit{ if } \; q \le q_0 \\ J \qquad\qquad\qquad , \textit{ otherwise} \end{cases} \tag{29}$$

where J is a town chosen by equation:

$$p_{ij} = \begin{cases} \dfrac{[\lambda\tau_{il} + (1-\lambda)\tau'_{il}]\cdot[\eta_{il}]^{\beta}}{\sum_{u\in N_i^k}[\lambda\tau_{iu} + (1-\lambda)\tau'_{iu}]\cdot[\eta_{iu}]^{\beta}}, & j \in N^k \\ 0, & j \notin N^k \end{cases} \quad (30)$$

where N^k are the towns that are not yet included on the k^{th} agent's tabu list. As visibility η_{ij} between towns, we will use the equation

$$\eta_{ij}(t) = \frac{1}{1 + PCV_{ij}(t)}$$

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, the nodes that cause violations will be less desirable by ants.

The pheromone trail is updated every time an ant traverses a node, according to the equation:

$$\tau_{ij}(t+1) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\cdot\tau_0 \quad (31)$$

where:

t the number of iteration taking place

ρ the pheromone evaporation factor, $0< \rho < 1$

τ_0 the initial amount of pheromone added on each node

while the anti-pheromone trail is updated according to the rule:

$$\tau'_{ij}(t+1) \leftarrow (1-\rho)\tau'_{ij}(t) + \rho\cdot\tau_0 \quad (32)$$

Finally, the nodes' pheromone trails that belong to the best solution found are updated according to equation:

$$\tau_{ij}(t+1) \leftarrow (1-\gamma)\tau_{ij}(t) + \gamma\cdot\Delta\tau_{ij}(t) \quad (33)$$

where

$$\Delta\tau_{ij}(t) = \begin{cases} Q/Cost_{best}, & if\ Town \in BestSolution \\ 0, & otherwise \end{cases}$$

and the nodes' pheromone trails that belong to the worst solution, according to equation:

$$\tau'_{ij}(t+1) \leftarrow (1-\gamma)\tau'_{ij}(t) + \gamma\cdot\Delta\tau'_{ij}(t)$$

where

$$\Delta\tau'_{ij}(t) = \begin{cases} Q/Cost_{worst}, & if\ Town \in Worst\ Solution \\ 0, & otherwise \end{cases}$$

where:

γ the global pheromone evaporation factor

Q the amount of pheromone added

$Cost^{t}_{best}$ the best (lowest) cost found till iteration t

$Cost^{t}_{worst}$ the worst (highest) cost found on iteration t

The algorithm

1. Define problem parameters for each agent and generator.
2. For every ant and for every generator select a power level randomly.
3. Evaluation of every solution constructed by the ants

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

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

4. Update the amount of pheromone and anti-pheromone trails for every traversed solution (local)
5. Update the amount of pheromone for the best solution path. (global)
6. Update the amount of anti-pheromone for the worst solution path. (global)
7. For every ant and for every generator select a power level based on the random-proportional rule.
8. Repeat algorithm from Step 3

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

The algorithm just described, was implemented on a real scale system of generator units [17] with 22 power generator units that have to be maintained within a 52-week horizon.

The table 1 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 maint. 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 highest level of energy that can be produced from all generators is zero.

The right table, also, represents the anticipated demand of the system for every week within the horizon.

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 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: $R_j = 20\%, D_j , j = 1,2…J$

It is important to define some determinant parameters for the solution of the problem. 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. These solutions are not feasible, thus we have to define the positive penalty weight 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:
    - $\alpha = 100$
    - $\beta = 100$
    - $\gamma = 20$
- The following variables were defined empirically:
    - $\tau_0 = 0{,}00001$
    - $Q = 1$

- Maximum number of iterations=2000, so as execution is not stopped early, and consequently without a good solution.
- Maximum number of iterations without a better solution=150, number big enough for a decent result.

- The definition of the heuristic information weight parameter β, the local and global pheromone trail evaporation factor ρ and γ, as well as the q_0 parameter, is also essential. The following tables represent the results given by experimental executions. All solution costs are indicative and expressed in 10^8 cost units.

Table 3. β, ρ, q_0 and γ versus Best solution

β	**Best Cost**	**ρ**	**Best Cost**	**q_0**	**Best Cost**	**γ**	**Best Cost**
0.1	3.3637	0.1	3.3477	**0.1**	**3.3475**	0.1	3.3486
0.2	3.3474	0.2	3.3475	0.2	3.3478	0.2	3.3479
0.3	3.3474	0.3	3.3473	0.3	3.3577	**0.3**	**3.3468**
0.4	3.3485	0.4	3.3479	0.4	3.3591	0.4	3.3473
0.5	**3.3463**	**0.5**	**3.3465**	0.5	3.3565	0.5	3.3476
0.6	3.3468	0.6	3.3482	0.6	3.3489	0.6	3.3475
0.7	3.3473	0.7	3.3534	0.7	3.3592	0.7	3.3488
0.8	3.3495	0.8	3.3483	0.8	3.3675	0.8	3.3494
0.9	3.3582	0.9	3.348	0.9	3.3576	0.9	3.3479
1	3.3581						
2	3.3591						
3	3.3498						
4	3.3504						
5	3.3485						
6	3.3477						
7	3.3479						
8	3.3479						
9	3.3473						
10	3.3476						
15	3.3486						
20	3.3485						

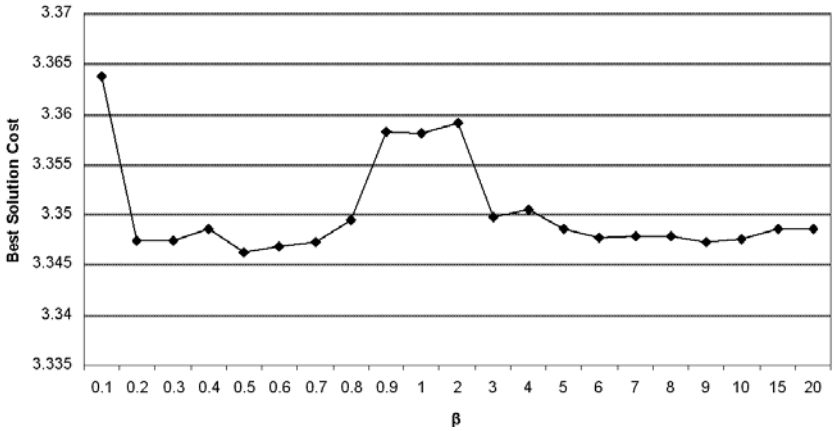

Figure 3. β versus Best solution

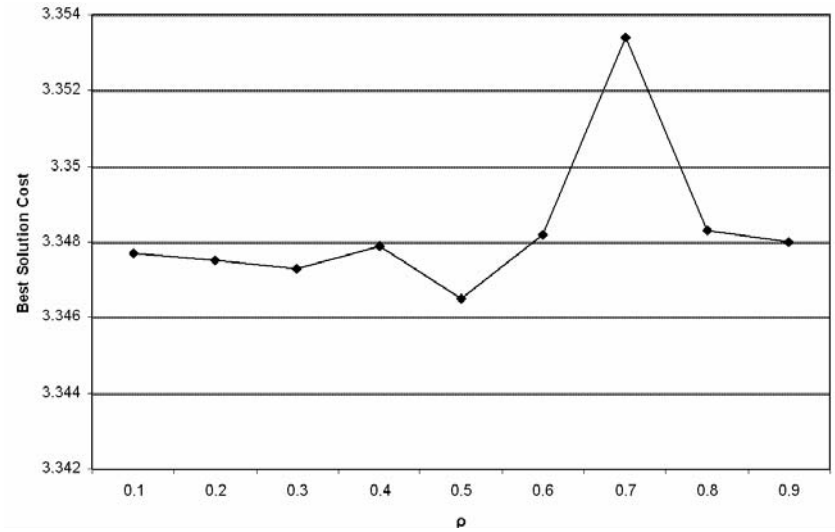

Figure 4 - ρ versus Best solution

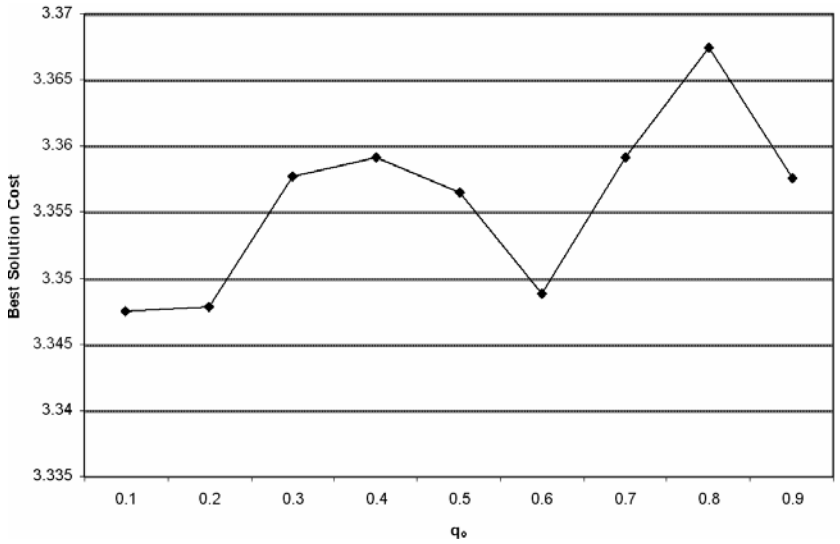

Figure 5. q_0 versus Best solution

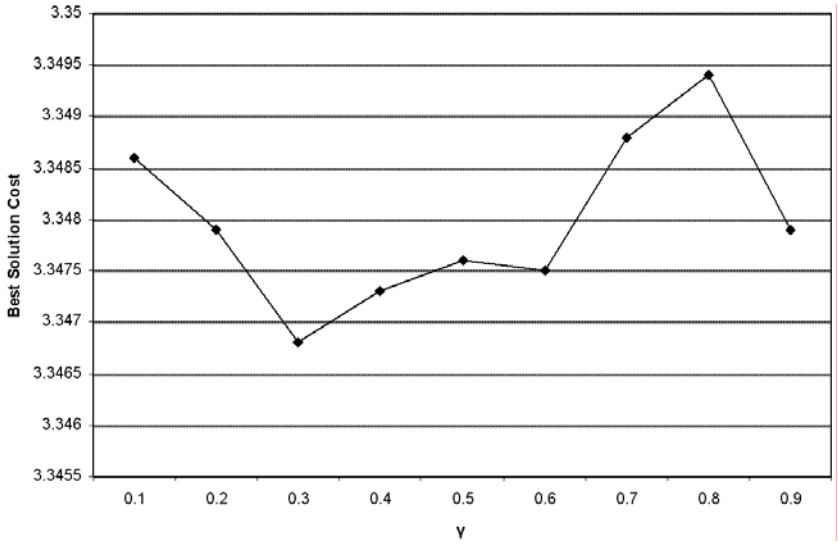

Figure 6. γ versus Best solution

We have to emphasize that the above results are not unique, as randomness is taken into account, although effort was made to minimize this factor.

The following parameters were chosen for the resolution of the problem:

- β=0,5
- ρ=1
- q_0=0,6
- γ=0,3

The program of the proposed method is written in Matlab 6.5 and it is run on an AMD Athlon 3000+ 1.79 GHz processor giving the following results:

- The best solution found is

[35,5,36,34,30,19,29,14,46,6,40,7,10,16,25,24,21,26,31,18,48,41]

That is the period that maintenance for every unit of the system can start (For unit 1, maintenance starts at week 35, for unit 2 on week 5, and so on).

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

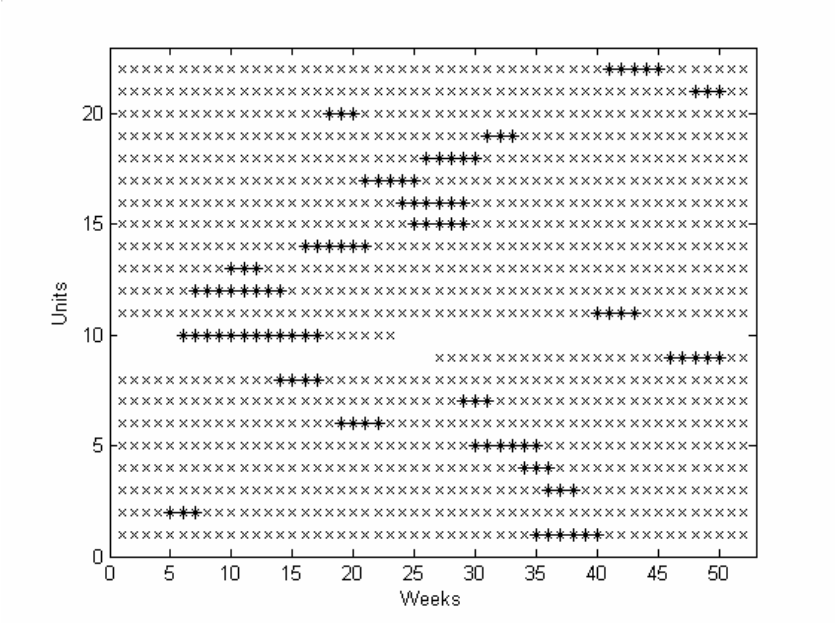

Figure 7. Maintenance periods of Best solution

where the feasible periods are represented with "x" and the selected periods with "*"

- The best solution cost found is 330560000.
- The best solution was found on iteration number 203.
- The best solution progress versus iterations is represented to the following figure:

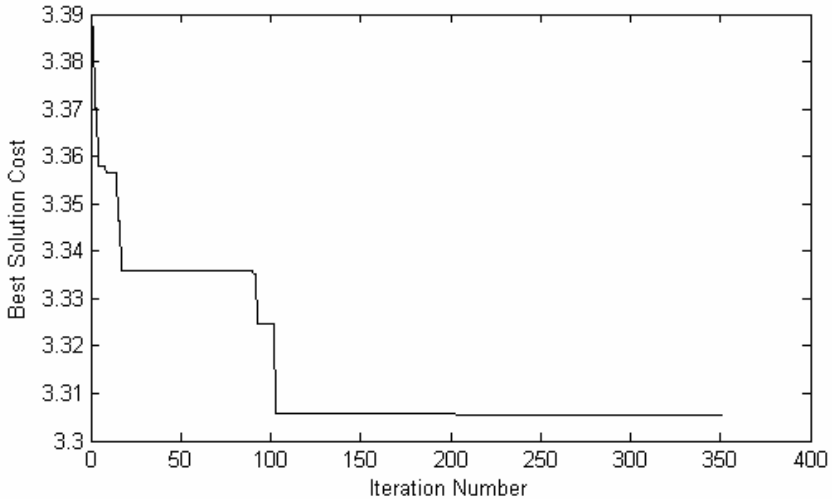

Figure 8. Best solution cost versus Iterations

The execution time is 0:13:35.812.

As we can see in Figure 8, the PAP algorithm has produced a very good solution with only a small number of iterations. From the first steps, it continuously finds better solutions

with lower costs, until iteration 203 that no better solution is found for the next 150 iterations, so the algorithm is terminated. It is remarkable that the algorithm produced much better solutions step by step, without being "trapped" on local best solutions due to pheromone trails (a problem that characterizes most ant-based algorithms).

Conclusions

This paper looked into the Thermal Generator Maintenance Scheduling Problem of a real-scale system. The problem has been studied with many mathematical and heuristic approaches in the past. In this project, the preferential Anti-pheromone approach was used, which is a variation of the ACS algorithm based on the behavior of real ant colonies.

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 that 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. 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.
2. Gruhl J., *Electric generation production scheduling using a quasioptimal sequential technique*, Research Note MIT-EL 73-003, MIT Energy Lab, April 1973.
3. Gruhl J., *Electric power unit commitment scheduling using a dynamically evolving mixed integer program*, Research Note MIT-EL 73-007, MIT Energy Lab, January 1973.
4. Dopazo J. F., Merrill H. M., *Optimal generator maintenance scheduling using integer programming*, IEEE Transactions on Power Apparatus and Systems, 1975, PAS-94(5), p. 1537-1545.
5. Zurn H. H., Quintana V. H., *Generator maintenance scheduling via successive approximations dynamic programming*, IEEE Transactions on Power Apparatus and

Systems, 1975, 94(2), p. 665-671.

6. Yamayee Z. A., Sidenblad K., Yoshimura M., *A computationally efficient optimal maintenance scheduling method*, IEEE Transactions on Power Apparatus and Systems, 1983, 102(2), p. 330-338.
7. Satoh T., Nara K., *Maintenance scheduling by using the simulated annealing method*, IEEE Transactions on Power Systems, 1991, 6, p. 850-857.
8. Kim H., Nara K., *A method for maintenance scheduling using GA combined with SA*, Selected papers from the 16th annual conference on Computers and industrial engineering, 1994, p. 477-480.
9. Kim H., Hayashi Y., Nara K., *An algorithm for thermal unit maintenance scheduling through combined use of GA, SA and TS*, IEEE Transactions on Power Systems, 1997, 12(1), p. 329-335.
10. Charest M., Ferland J. A., *Preventative maintenance scheduling of power generation units*, Annals of Operations Research, 1993, 41, p. 185-206.
11. Dahal K. P., McDonald J. R., *Generational and steady state genetic algorithms for generator maintenance scheduling problems*, Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, 1997, pp. 260-264.
12. Dahal K. P., McDonald J. R., *Generator maintenance scheduling of electric power systems using genetic algorithms with integer representation*, Submitted to GALESIA'97, 1997, p. 456-461.
13. Burke E. K., Clark J. A., Smith A. J., *Four methods for maintenance scheduling*, In Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, 1997, pp. 264-269.
14. Dorigo M., Gambardella L.M., *Ant Colonies for the travellong salesman problem*, BioSystems, 1997, 43, p. 73-81.
15. Randall M., Montgomery J. Q., *The Accumulated Experience Ant Colony for the Travelling Salesman Problem*, Proceedings of Inaugural Workshop on Artificial Life, Adelaide, Australia, 2001, pp. 79-87.
16. Iredi S., Merkle D., Middendorf M.Q., *Bi-Criterion Optimization with Multi Colony Ant*

Algorithms. Evolutionary Multi-Criterion Optimization, First International Conference (EMO'01), Zuric, 2001, pp. 359-372.

17. Ibrahim El-Amin, Salif Duffuaa, Mohammed Abbas, *A tabu search algorithm for maintenance scheduling of generating units*, King Fahd University of Petroleum and Minerals, 1998-1999, Electric Power Systems Research 54, 2000, pp. 96.
18. Wood J., Wood B. F., Woolenberg B. F., *Power Generation, Operation, and Control*, John Wiley, 1984.