SCHEDULING MAINTENANCE ACTIVITIES OF ELECTRIC POWER TRANSMISSION NETWORKS USING AN HYBRID CONSTRAINT METHOD

Nuno Gomes, Raul Pinheiro, Zita Vale, Carlos Ramos
GECAD – Knowledge Engineering and Decision Support Group, Institute of Engineering Polytechnic Institute of Porto, Portugal
ngomes, raul, zav@dee.isep.ipp.pt, csr@dei.isep.ipp.pt

Abstract – In this paper we present a Constraint Logic Programming (CLP) based model, and hybrid solving method for the Scheduling of Maintenance Activities in the Power Transmission Network. The model distinguishes from others not only because of its completeness but also by the way it models and solves the Electric Constraints. Specifically we present an efficient filtering algorithm for the Electrical Constraints. Furthermore, the solving method improves the pure CLP methods efficiency by integrating a type of Local Search technique with CLP. To test the approach we compare the method results with another method using a 24 bus network, which considers 42 tasks and 24 maintenance periods.

Keywords: Power Transmission Networks, Maintenance Scheduling problem, Hybrid Constraint Methods, Constraint Programming

1 INTRODUCTION

Electric markets are changing and imposing additional competition and growing complexity in power systems. Nowadays there is a strong need of high service reliability and low production cost which imposes a constant improvement for the different production areas and processes. Among them, the scheduling of maintenance activities over generation, transmission and related equipment is one of the important ones. Reviewing the literature, we can find several approaches; either for the problem formulation as for the problem solving. In terms of formulation, despite being related, usually Generation is treated separately from the Transmission associated Maintenance Scheduling problem. Regarding problem solving, we can also find different methods and techniques, from traditional methods based on Mixed Integer Linear Programming (MILP) [1], or Dynamic Programming [2], to more recent methods based on Constraints Logic Programming (CLP) [3], or Metaheuristics [4]. Each method has its strengths, but also its weaknesses. From a real world point of view, one of the most appreciated strengths is the flexibility. Sometimes scheduling Engineers have difficulties in enumerating all the problem constraints during the application development phase, so they frequently need to update the problem model. Other times, they just have the feeling, without knowing the reason, so they want to build their plans in a semi-automatic way. CLP is a feasible approach for developing this type of “solving/decision support” tools. With CLP the model of the problem and the solving method can easily be developed, as well as the corresponding computer program. The latter ones are usually small and easy to maintain. Evidence of these strengths is the real world CLP based tool, described in [5], and used by the Spanish Electric company ENHER.

In this paper we present a CLP based model and hybrid solving method for the Scheduling of Maintenance Activities Problem in the Power Transmission Network (SMA_PTN). The CLP model includes all the usual real types of constraints associated with the maintenance of transmission network and also the most important constraints associated with the maintenance of generating units. Furthermore, the solving method improves the pure CLP methods efficiency by integrating a type of Local Search technique with CLP according to the Reduce and Assign Framework [6]. To substantiate the approach we present the results for a 24 bus network, which considers 42 tasks and 24 maintenance periods. Our work can be distinguished from others by the following issues:

- It deals with the maintenance scheduling problem associated with the Power Transmission Network, despite also considering some constraints associated with the Generation Units Maintenance. Most of the references deal only with the simpler problem associated to Generation, or with a simplification of the SMA_PTN.
- It presents a different and complete problem model with a new and efficient way of modeling and solving the electrical constraints. Note that these constraints are the most complex to model and solve.
- It presents an improvement to the basic CLP solving methods based on a hybridization framework which proved to overcome some CLP efficiency problems.

2 OTHER APPROACHES TO THE PROBLEM

One of the characteristics of the SMA_PTN is the large number of different types of constraints. Some of them like, Resources, Temporal, Demand, or Precedence are common to other scheduling types of problems. Other, usually grouped as Electric Constraints, like Overload, Tension Drops, Isolation, or Continuity, are specific to the SMA_PTN. Considering a recent study reported in [7], which classifies almost 90 papers related with Power Systems Maintenance Scheduling Problems, published in the last 30 years in power systems related Journals, Reviews and Conferences, we can say that almost 90% of the work was done over the Generators Maintenance Scheduling Problem (GMS). Only the remaining 10% consider Transmission Network Constraints, and even then, just a few model the problem from the Transmission Network point of view.

Regarding GMS with Transmission Network Constraints, significant contributions have made by Silva et al [8], and Marwali and Shahidehpour [1]. Both use a simi-
lar approach to the GMS problem. Specifically, they divide the GSM in a Master Mixed Integer Linear problem, which includes normal scheduling constraints (time, resource, demand, etc.) and looks for a global scheduling plan, and in several Linear Sub-Problems, which verify for each scheduling plan period if the electric constraints are respected. Whenever the resolution of a Sub-Problem detects infeasibilities associated with trial plan, further constraints (Benders Cuts) are generated and added to the Master Problem which is then solved again. This interactive process continues until a good solution is found. Note that in fact they solve a GSM problem in spite the consideration of some transmission network electric constraints, like the load lines limits, and also the peak load balance [8]. Besides that, the solving process can be inefficient for real world problems. On one hand, it needs to deal with a large number of variables, and constraints. On the other hand, as the electric constraints are verified at posteriori the number of needed iterations can be very large, particularly when the network load reserve is low.

Regarding the SMA_PTN, significant contributions have been made by Creemers et al [5], and Langdon [4]. The problem model used in Langdon work is simpler and less realistic than all the other models presented in this section, as it does not includes any of the usual scheduling constraints like time, resources, or precedence. In fact, the scheduling plan is generated avoiding only line overloads, and node isolation. The solving method is based on Genetic Algorithms. More complete is the model used in [5] as it includes all the usual constraints. Like us, they solve the problem using CLP. However, the electric constraints are treated independently and some of them, like Voltage Drops, neglected during the plan generation phase. The electrical constraints are implemented using two types of domain variables, namely, topological and electric. The topological domain variables represent the states of all network switches in each time slot. Every branch (and switch) of the network needs electric domain variables representing the currents that flow on them. The importance of the current variables lies on their lower and upper limits which will propagate changes to the domains of other current variables, as well as to the domains of topological and temporal variables. This modeling approach requires a lot of memory and penalizes the application performance. As an example in [5] the authors refer that for a power-distribution network of about 1200 nodes and 400 operable switches, considering 15 maintenance jobs to be scheduled, the system creates about 22000 domain variables. In this paper we will present a new complete model and solving method for the SMA_PTN problem that tries to overcome the above referred drawbacks.

3 THE SMA_PTN CLP MODEL

Since the beginning of the 1980s a lot of work has been done concerning constraint based scheduling. An historical perspective is outlined in [9]. CLP problems are modeled as a set of variables with domains (e.g. if \( D(Var) = \{1,2,3\} \) then Var can only take the values \( \{1,2,3\} \)) and a set of constraints restricting the possible combinations of the variables values. The aim is to find an instantiation of the variables satisfying all the constraints or, if an objective function exists, to find a feasible instantiation minimizing or maximizing the objective function. Modeling like this is natural, because the constraints can capture arbitrary relations. Contrary to frameworks like linear and integer programming, the constraints are not restricted to linear equalities and inequalities. The constraint can express arbitrary mathematical or logical formula, like \((x \leq y \lor x = y)\). The CLP “secret” is the combination of Constraint Propagation and solving with sophisticated search techniques. Constraint propagation is based on the idea of using constraints actively to prune the search space. Each constraint has assigned a filtering algorithm that can reduce domains of variables involved in the constraint by removing the values that cannot take part in any feasible solution. Each constraint may have its own filtering algorithm, provided by the CLP system, or developed by the application developer.

The maintenance scheduling problem on electrical transmission networks is an optimization problem that tries to find the starting time for each maintenance task, which minimizes a certain cost function without violating any of the problem constraints. The cost function can be related to energy loss, line overloads, maintenance costs, etc or a combination of these. The constraints or restrictions are of different types. In this work we will consider the following:

a) Time Constraints – These constraints impose, for each unit to be maintained, the valid maintenance periods. For example in some periods of the winter some elements should not be maintained due to the possible bad weather conditions;

b) Resource Constraints – The available amount of resources must be respected at all times. Some of the resources are: maintenance teams, vehicles, tools, etc;

c) Precedence Constraints – Some tasks should be performed before or after others;

d) Continuity constraints - There must be an electrical path from all nodes to all nodes (no islands);

e) Capacity Constraints – The production capacity of each production unit can never be exceeded;

f) Demand Constraints – The network should meet the demand of all consumers;

g) Overload constraints – Line currents must respect line physical limits;

h) Bus voltage constraints - Bus voltages must remain in a tolerance interval around their nominal values.

Among the above, the constraints d) to h) are the most difficult to model. These constraints, specific to SMA_PTN problem, can be grouped under the class of Electric Constraints. As a first step, we will present the CLP model for the general constraints.

3.1 General Problem Variables and Constraints

The main objective of the SMA_PTN problem is to find the starting maintenance period for each of the tasks. Within CLP, this can be represented by the domain variable (1).

\[ S_i \]  \hspace{1cm} (1)
The initial domain of the \( S_i \) variable could be set to the available maintenance periods. Furthermore, some constraints can be completely solved when imposed, reducing immediately the search space (e.g. \( S_i > 4 \) removes all the values lower than 5 from the \( S_i \) variable domain). The performance of CLP applications strongly depends on the filtering algorithms used for each type of constraint. Frequently, the performance varies with the way one constraint is imposed. Considering this, we included in our model the state variable (2) which takes value 1 if task \( i \) is in maintenance in period \( t \) and 0 otherwise. Note that, the variables (1) and (2) must be strictly connected (whenever a variable \( S_i \) is changed the corresponding \( x_{it} \) should be updated according, or vice versa).

Considering the above variables, general scheduling constraints a) to c) can be represented by the following expressions:

\[
\begin{align*}
e_i & \leq S_i \leq \ell_i & \forall i: i \in [1, ..., N] \quad (3) \\
S_i + r_i & \leq S_j & \forall i, j: i \text{ precedes } j \quad (4) \\
x_{ij} & \neq x_{ji} & \forall i, j, t: i \text{ precedes } j, \text{ non overlap} \quad (5)
\end{align*}
\]

\[
\sum_{i=1}^{N} (x_{ip} * rc_{ip}) \leq ar_p & \forall i, p: i \in [1, ..., N] \quad (6)
\]

where \( N \) is the number of tasks, \( rc_{ip} \) the requirements of resource \( p \) by task \( i \) and \( ar_p \) the limit of resource \( p \) in period \( t \). Expression (3) concerns the Time Constraints, and restricts the task execution to a certain time window. A non execution window can also be modeled restricting the value of the corresponding \( x_{it} \) to 0. Expressions (4) and (5) are respectively the precedence and non-overlapping constraints. Precedence constraints establish order among tasks. An extension to this can be the non-overlapping constraint when two constraints cannot be executed simultaneously. Resources Constraints (6) are know to be hard to solve, however they are essential in scheduling problems. This way several specific filtering algorithms were developed for this type of constraints and included in CLP systems as a global constraint [10].

In the CLP system ECLiPSe [11] used in this work, as in other CLP systems, the resources constraint is modeled as a “cumulative” constraint. Depending on the filtering algorithm different levels of domains reduction can be accomplished. However, more reduction means more computation time, so a good trade-off between time to solve the constraint and search tree reduction should be achieved.

3.2 Electrical Constraints

Electrical constraints d) to h) guarantee that, whenever a maintenance task needs the outage of one, or more, network branches, the service quality to the customers and network limits are kept. In order to develop the filtering algorithms for the electrical constraints we use a graph structure to represent the network topology for each scheduling period. The structure keeps all the network branches in the form \( \text{branch}(\text{node}_1, \text{node}_2, x_{it}) \), where \( \text{node}_1 \) is the source node, \( \text{node}_2 \) is the sink node, and \( x_{it} \) is the state variable associated with the task that needs the branch outage.

3.2.1 Continuity Constraints

Considering the network graph we can use some algorithms from Graph Theory to develop the filtering algorithm to the Continuity Constraint d). Continuity constraint is responsible for ensuring that all customers are always supplied and for avoiding “islands”. In terms of graph theory, this means to find “critical branches” through the discovery of “articulation points”. An articulation point of a connected graph is a vertex whose removal disconnects the graph. So, if an articulation point has only one branch connected to it, the branch is critical. In our model we have implemented a Depth Search based algorithm to find critical branches. Whenever a branch is removed from service, the Continuity constraints filtering algorithm determines the critical branches and sets the corresponding \( x_{it} \) variable to 0. This means that critical branches cannot be schedule for maintenance.

3.2.2 Capacity, Demand, Overload and Bus Voltage Constraints

The outage of a branch for scheduling is only possible if none of the network limits is exceeded and the demand is fulfilled, this means constraints e) to h) cannot be violated. The filtering algorithm for all these constraints is called whenever a certain line (branch) is removed from the network and the network topology is changed. The filtering algorithm starts from a certain network state, where it is known that there are some lines that cannot be removed, to determine which of the remaining lines can be removed without violating any of the electrical constraints. Only these lines and corresponding maintenance tasks can be considered for maintenance in the given period. At first sight, one has to perform a load flow for each possible line removal to determine which removals violate electrical limits. It is at this particular point that we propose a function that allows to determine which of the "possible to remove" lines can actually be removed respecting electrical constraints and performing just one load flow, instead of one per "possible to remove" line.

Contingency analysis

When a line is switched on, or off, through the action of circuit breakers, line currents are redistributed throughout the network and bus voltages change. The new steady-state bus voltages and line currents can be calculated using what is called contingency analysis. Contingency analysis uses DC approximations to evaluate the impact of an element or combination of elements outage on the remaining network. Distribution factors and compensating currents are important elements in contingency analysis that allow this evaluation to be done without having to compute load flows or even to modify the network's Z matrix.
### Distribution Factors

The line-outage distribution factor $L_{pq,mn}$ relates the variation of current on line $p-q$ due to an outage on line $m-n$ as shown in expression (7) and it can only be computed from the elements of $Z$ and the $p-q$ and $m-n$ line impedances.

$$L_{pq,mn} = \Delta I_{pq} / I_{mn} \tag{7}$$

Based on the line-outage distribution factor $L$ one can compute which lines one can be put out of service for maintenance purposes, without overloading the remaining ones.

### Compensating currents

When considering line additions or removals from an existing system it is not always necessary to build a new $Z$ matrix, especially if the only interest is to establish the impact of the changes on the existing bus voltages and line flows. An alternative procedure is to consider the injection of compensating currents to account for the effects of the line changes. On our case, one just needs to study the single line case. By means of compensating currents one can determine the new bus voltages resulting from a line removal.

### Current Constraint

Let us assume that our filtering algorithm receives, besides network data, a vector (size = number of lines) where each element is associated with an electrical line and can assume one of three values, indicating the state of a given line state: already removed from the network; and can assume one of three values, indicating the state of a given line state: already removed from the network; or removed for maintenance purposes without overloading the remaining network.

### Bus voltage constraints

To deal with the bus voltage constraints, one has to compute all the new bus voltages resulting from the removal of each one of the possible removable lines. The compensating current approach is suitable to do so. If for a given line removal, the voltage in any of the buses violates the accepted tolerance, then this line cannot be removed.

### Optimization Function

In real world SMA_PTN problems it is usually desirable, not only to find a solution, but to find the best solution according to some optimization function. In the literature we can find different optimization functions. Usually, in recent works related with the GMS problem, the optimization function corresponds to the minimization of maintenance and/or production costs. This is the case in [1]. In some works, particularly older ones, it is also usual to use reliability based optimization functions. One common example attempt to achieve the minimum cost of load probability for a whole year to evaluate the risk of a certain system (see [12] for other reliability functions).

In this work we will try to minimize the maintenance costs and at the same time to increase network reliability. The maintenance costs depend directly on the tasks execution costs, like labor costs, equipment costs, etc. These costs can vary during the year. Reliability costs depend on the network failure risk. As we have discussed in previous section, depending on network topology, we can have a different number of critical branches. The higher this number is higher is the risk of network failure. The solution cost is defined by expression (10).

$$\min \sum_{i=1}^{N} \sum_{t=1}^{T} (x_i \cdot mc_i) + K \cdot \sum_{i=1}^{N} N_{cb}$$ \tag{10}$$

Where $mc_i$ is the maintenance cost for task $i$ in period $t$, $T$ the number of periods, $N_{cb}$ is the number of critical branches and $k$ a normalization factor to scale both the costs terms.

### 4 THE BASIC SEARCH ALGORITHM

Constraint solving is not a complete method. In fact, after propagating all the constraints we just have the guarantee that all the removed domain values are consistent with at least one constraint. In order to find a solution we need, for example, a Repair-based method or, a Refinement-based method. In the last type, which is often used, each of the variables is assigned a value incrementally until a complete solution is found or a constraint is violated. If a constraint is violated, the last assignment is undone and an alternative value is chosen ("enumeration"). If no value assignment is consistent, the search backtracks to a previously assigned variable, and so on. The result is a depth-first tree search. In the rest of this paper we will call this algorithm Basic Backtrack Search (BBS). The following is the corresponding pseudo-code.

set $X = (x_1, ..., x_n)$
while there are unassigned variables in $X$
    select an unassigned variable $x$ from $X$;
    select a value $v$ from the domain of $x$;
    assign $x = v$;
    backtrack if a constraint is violated;

This algorithm contains primarily two choices, namely variable selection and value selection. The order in which
variables and values are selected may have a significant impact on search efficiency, and various heuristics exist for these choices that try to minimize the need for backtracking. One example is a heuristic that selects first the variable with the smallest current domain. This heuristic is usually named First Fail. While such domain-independent heuristics can be effective, domain-specific heuristics could be even better. This is of interest in the context of maintenance scheduling problems where prior knowledge about the problem can be included in a heuristic. One example is a heuristic that selects first the $S_i$ variable, with a value (period) in its domain, which leads to the smallest maintenance cost. This heuristic, named Smallest Maintenance Cost, has produced good results in GMS problems [13].

Finally, if the constraint problem is an optimization problem, a refinement-based search can be augmented easily with a mechanism that adds a new constraint (lower or upper bound) over the total cost variable (result of the objective function) every time a new solution is found. This forces subsequent solutions to have increasingly better objective values and can be very effective in removing parts of the search tree. At the end, the last-found solution is returned as the optimal solution. This kind of optimizing search shares the same idea of the branch-and-bound optimization technique used within Linear Programming.

In section 6.1 we will present the results obtained with the BBS method augmented with the CLP Branch and Bound version, for an SMA_PTN problem instance. We also compare the two sets of variable and value selection heuristics. The first set includes the Smallest Maintenance Costs (SMC) variable selection heuristic presented above and the Lower Value (LV) value selection heuristic. The SMC heuristic simply chooses the lower value of the variable domain. With this heuristic set, choices are made over the Maintenance Cost ($M_{ci}$) variable. Each value of the $M_{ci}$ variable domain consists of the total maintenance cost for task $i$ when schedule in a certain period. Naturally, there is a connection between $M_{ci}$ and $S_i$, which instantiates one variable with the correspondent value, whenever the other variable is instantiated (e.g. being $D[M_{ci}]=[50, 60, 80]$ and $S_i=[1, 2, 3]$; if $M_{ci}$ takes value 50 then $S_i$ is instantiated with value 1, the inverse instantiation order is also true). The idea of this heuristic set is to schedule the tasks in a period which corresponds to the smallest maintenance costs. This heuristic set has some drawbacks. One of the important ones is that it neglects the number of critical branches, and the difficulty of scheduling each of the tasks. In order to evaluate this problem we also present the results for a second heuristic set, where choices are made over the $S_i$ variables. This set includes the general First Fail (FF) heuristic presented above, for variable selection and a value selection heuristic named H1. The H1 heuristic selects the $S_i$ value (maintenance period) for which the maintenance cost is lower. The idea behind this heuristic set is to schedule critical tasks first (there are few available periods to schedule), because this can cause failures. Then the tasks could be scheduled in periods with less constrained networks.

5 GUIDED CONSTRAINT SEARCH

With basic CLP methods like BBS and even with very efficient filtering algorithms which can prune significantly the search space, it is impractical to explore all the search tree of a real world SMA_PTN problem. Ideally, the search method should discard all the unfruitful search space areas and look only on the promising ones. The Guided Constraint Search (GCS) method is an instance of the Reduce and Assign Framework [6] that tries to do that. This method was already tested with success with the GMS problem [3].

The idea consist of using a fitness function to choose, in each iteration, only the $z$ most promising values of each variable domain. The resulting sub-problem is then solved using, for example the BBS method, in order to find a local optimum. If the fitness function is adequate it is possible that the local optimum is also the global one, and a lot of search time is saved. Naturally, if the fitness function is not adequate the results can be worse than if no division was made. This idea is illustrated in Figure 1.

In order to choose the sub-space we need to define an inutility (utility) function. For each pair variable/value (X,Y) the inutility function returns a quantity that is then used to know if the corresponding value should be included, or not, in the variable domain for the next iteration. The inutility function is defined by (11):

$$I_y = C0_y + Bc_y \times p_y \quad (11)$$

Where $I$ is the inutility value; $C0$ is an initial heuristic cost; $Bc$ is the best solution cost that includes the pair and
$p$ is the penalty parameter. Initially $p$ and $Bc$ are initialized, for all pairs, respectively to 1 and to the problem upper bound cost. This means that initially the value of the utility function depends only on $C0$. Consequently $C0$ can be used to express some heuristics that give the possible best initial values. In the end of each iteration, the utility function parameters of the used pairs are updated, depending if they belong to a new best solution. Specifically, the penalty parameter is incremented by one unit for the pairs that do not belong to the best solution. For the other pairs, the penalty remains the same with the $Bc$ parameter updated to the new best cost. If no best solution is found all the pairs see their penalty increased. With this update procedure we accomplish two objectives. First, the probability of certain (possible best) pairs being chosen is progressively increased as they belong to good solutions (convergence of the search). Second, the search is diversified because the penalty of the pairs that do not belong to new best solutions is increased. Note that, as search evolves, the weight of $C0$ becomes negligible. The key to the effectiveness of GCS is the equilibrium between penalizing “bad” pairs variable/values and do not penalizing “good” ones. Considering the above, we can define the GCS algorithm as:

1. Develop the problem’s model in terms of variables and constraints (as a CSP)
2. Initialize for all the pair’s variable/value the inutility function parameters ($C0$, $Bc$ and $p$).
3. Repeat until a termination condition (e.g. a maximum number of iterations or time limit) is reached
   - Select for the domain of each problem’s variable the $z$ values with the smaller inutility value ($\forall i$)
   - Call the CLP module to perform the search and return a new best solution (or not)
   - Update all the parameters, including the inutility value, for all the used pairs according to the returned solution
   - If the solution is a new best solution, update $Bc$ for all the pairs that belongs to the solution and increment $p$ for the other pairs
   - Else increment $p$ for all the used pairs

As we can see, the algorithm is independent of the CLP module, however, for large problem instances, the time the module needs to find the local optimum solution can be very large. This means that it is necessary to limit the time the CLP module has for searching, in order to avoid losses of time exploring bad search space areas. Regarding this, besides the $z$ parameter the GCS algorithm has another parameter to be set, namely the $st$ parameter which corresponds to the searching time of the CLP module.

### 6 TEST AND RESULTS

In order to test the proposed methods we used the IEEE-Reliability Test System [14]. We have added some missing information in order to test all the model capabilities. The resulting network has 24 buses and 38 transmission lines. For the scheduling problem we considerer 42 tasks to be schedule over 24 periods. 30 of the 42 tasks require the outage of a network line. The time window of each task is in average of 15 periods, with a few around 10 and 20 periods. All the tasks need a maintenance team to be undertaken. Some of them can only be undertaken by a particular maintenance team. Each maintenance team is classified as an independent resource. In total we considered 20 different resources. At the most each task needs 3 different resources. Maintenance costs can vary at most 20% depending on the maintenance period. Considering this variation we set the $k$ value of the objective function in order that the weight (in the objective function) of a critical line to be similar to the average of all the 20% maintenance costs. The tests were made using a 1,8 MHz Pentium PC, with 512Mb of memory, running Windows XP and the ECLiPSe CLP system. As all the methods are deterministic only one run was made for each method.

#### 6.1 Augmented BBS results

The augmented BBS method consists of the BBS algorithm augmented with the CLP version of the Branch and Bound. For this method we tested the two heuristics sets presented in section 4, namely SMC+LV and FF+H1. The results are shown in Table 1.

<table>
<thead>
<tr>
<th>Heuristic Set</th>
<th>Best Cost</th>
<th>Best Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMC+LV</td>
<td>143500</td>
<td>4587</td>
</tr>
<tr>
<td>FF+H1</td>
<td>141100</td>
<td>3445</td>
</tr>
</tbody>
</table>

Table 1 -Best Cost (limited time) and Best Time (limited cost) of the augmented BBS method for heuristic sets

Where “Best Cost” is the best solution cost found by the strategy in less than 6000 cpu seconds, and “Best Time” is the time needed to find a solution with a cost lesser or equal than 143500. As we can see, the FF+H1 heuristic set is better than the SMC+LV, at least for our data. As we pointed out before the SMC+LV set only considers the maintenance cost, neglecting the influence of critical branches. This strategy can lead the search to schedule first non critical tasks (tasks with a large number of scheduling periods) instead of the critical ones. This can delay failures in the search process and increase the number of backtracks.

Another important fact is that the search algorithm spends almost 15% of the search time solving the electric constraints, which shows the importance of our filtering algorithm.

#### 6.2 GCS Results

For our tests with the GCS method we used the augmented BBS with the FF+H1 as the CLP module. In this way the choice is made over the $S_i$ variable. The utility function defines, in each search iteration, an inutility value for each pair ($S_i$, $v_j$). For the inutility function $CO$ parameter we used the maintenance cost of the corresponding period ($v_j$ value). The underlying idea is that the best periods to schedule a task probably are that for which the maintenance costs are lower. As pointed before as the search evolves the weight of the $CO$ parameter becomes negligible.

Besides the $CO$ parameter we need to define the number of values of the domain variables ($z$ parameter) and the search time given to the CLP module ($st$ parameter).
Theoretically, $z$ can vary from 1 to $n$. In practice, $z$ values near 1, or near $n$ usually imply bad algorithm’s performance. The reason for this is obvious. If $z$ is close to $n$ then the size of the sub-search space is almost equal to the size of the global one, so the effort that the constraint module must dispense in order to find the local optimum is almost the same it spends to find the global one. If $z$ is close to 1, the significance of the constraint module is almost null, so the overhead associated with it does not bring any advantage. The right $z$ value is expected to be somewhere in the middle range. Based on experiments of [3] we chose for our problem instance $z=12$.

Similarly the $st$ parameter should also be chosen carefully. If $st$ is too small then it is possible that the CP module does not have the time to find the optimal local solution. If $st$ is too high then the CP module spends a lot of time in each iteration, exploring bad search space areas. Based on experiments of [3] we chose $st=70s$. The results obtained with these parameters are shown in Table 2. In order to facilitate the comparison we included the results of the BBS+FF+H1 method.

<table>
<thead>
<tr>
<th>Heuristic Set</th>
<th>Best Cost</th>
<th>Best Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF+H1</td>
<td>141100</td>
<td>4945</td>
</tr>
<tr>
<td>GCS</td>
<td>141100</td>
<td>2523</td>
</tr>
</tbody>
</table>

Table 2 - Best Cost (limited time) and Best Time (limited cost) of the augmented BBS and GCS method

As we can see the GCS method found the same solution of the BBS but in almost half of the computation time. This confirms our expectations and also the results of [3].

7 CONCLUSION AND FUTURE WORK

This paper deals with the Scheduling of Maintenance Activities Problem in the Power Transmission Network. In order to solve the problem we present a CLP based model that includes all the main real constraints. Specifically, the model includes a new approach to represent the electrical constraints and also the respective filtering algorithm. The algorithm showed to be very accurate and efficient without consuming too much computation time or memory.

Besides the model we also presented a Basic Backtrack search algorithm augmented by a type of Branch-and-Bound. The performance of this type of algorithms strongly depends on the Variable and Value selection heuristics. We tested two heuristic sets that include either general CLP search heuristics as also specific SMA_PTN problem ideas. Tests show that the FF+H1 heuristic set produces better results than the SMC+LV.

Finally, we presented a new search algorithm, named Guided Constraint Search that can improve the BBS by selecting and exploring only promising search space areas. With this algorithm we could reduce the computation time needed to find the best solution by almost 50% when compared with the BBS algorithm.

In spite of the good results achieved we think that it is possible to improve them even more by better tuning the GCS parameters. This idea should be confirmed by more tests with different problem instances.

ACKNOWLEDGEMENTS

The authors would like to acknowledge FCT, FEDER, POCTI, POSI, POCI and POSC for their support to R&D Projects and GECAD Unit.

REFERENCES

[7] Gomes, N., Uma Abordagem ao Problema de Escalonamento de Tarefas de Manutenção em Unidades de Produção utilizando Restrições, Work for the Public Proofs for Assistant Professor at ISEP, 2004. (in Portuguese)