A multipopulation cultural algorithm for the electrical generator scheduling problem

Jason G. Digalakis*, Konstantinos G. Margaritis
Parallel Distributed Processing Laboratory, University of Macedonia, 156 Egnatias Street, 54006 Thessaloniki, Greece

Abstract

The electrical generator maintenance scheduling problem has been tackled by a variety of traditional optimisation techniques over the years. This paper proposes a method to solve the maintenance scheduling problem, called the parallel co-operating cultural algorithm (PARCA). In the proposed model, a variety of selection mechanisms, operators, communication methods, and local search procedures are applied to each solution generated by genetic operators and parameters as explained in the sequel. Our cultural algorithm framework combines the weak search method with the knowledge representation scheme for collecting and reasoning knowledge about individual experience.

Keywords: Electrical generator scheduling problem; Parallel cultural algorithms

1. Introduction

The electrical generator maintenance scheduling problem is a complex combinatorial optimisation problem [1,2,7]. This problem has been studied widely in the past. The problem considered here involves a general category of the resource constrained project scheduling problem in which the activities may have more than one execution mode, and renewable as well as non-renewable resource constraints exist.

This problem is relevant in flexible manufacturing systems and other scheduling environments where alternate routings are permitted for jobs, and different machines have different machining characteristics and tool and material handling requirements. A more detailed description of the formulation is given in [1,2,4].

Traditional optimisation-based techniques such as integer programming [16], dynamic programming [6,17] and branch-and-bound [18] have been proposed to solve this problem. For small problems these methods give an exact optimal solution.

Several modern heuristic methods have been applied to the problem. Examples are simulated annealing [10,11], stochastic evolution [12], genetic algorithms (GAs) [12,13] and tabu search [14]. Further, hybrid

* Corresponding author.
E-mail addresses: jason@uom.gr (J.G. Digalakis), kmarg@uom.gr (K.G. Margaritis).
GAs (HGAs) [3,4] using neighbourhood search algorithms (e.g. local search, simulated annealing and tabu search) were proposed to improve the search ability of GAs, and their high performance was reported in the literature.

Cultural algorithms (CAs) are a class of models derived from the cultural evolution process [23]. These algorithms support the basic mechanisms for the cultural change described in [24].

In this paper we propose a guided local search (GLS)-based parallel cultural algorithm which is based on the ideas of Mendes et al. [9], and it is a hybrid algorithm of a GA and a GLS procedure [15].

The algorithm is implemented using the message-passing interface (MPI) standard. MPI is a specification of a message-passing library for parallel computers and workstation networks [19]. Results are performed on the following systems: a SGI Origin 200 and 6 Pentium (P5/100 MHz) cluster where the interconnection is realised through a fast ethernet (100 MB/s).

2. Problem description

Consider $i$ generating units producing output over a planning horizon of $J$ periods. Each unit in $1 \leq j \leq J$ must be maintained for $M_i$ continuous periods in the horizon. However the starting period, denoted by $x_i$, for each unit $i$, is unconstrained even in the case that $x_i = J$ and $M_i > 1$ for some $i$. Some of us are considering a rolling plan, in which the maintenance period would wrap around to the start of our planning horizon. The operating capacity of each unit is denoted by $C_i$. Under no circumstances is it possible for a unit to exceed this limit. In order to avoid random factors in the problem such as unit random outages, a reserve capacity variable proportional to the demand is incorporated into the problem description. This problem is classified as a deterministic cost-minimisation problem and can be solved using an optimisation-based technique.

Therefore in the period $j$ where $1 \leq j \leq J$, the anticipated demand for the system as a whole will be denoted by $D_j$ and the reserve capacity required by $R_j$. Fuel costs can also be estimated for each period as a constant $f_j$ per unit output.

Finally, let $p_{ij}$ represent the generator output of unit $i$ at period $j$, $c_{i(j)}$ be the maintenance cost of unit $i$ if committed at period $j$ and let $y_{ij}$ be a state variable equal to 1 if unit $i$ is being maintained in period $j$ and zero otherwise.

The objective of the problem is to minimise the sum of the overall fuel cost and the overall cost of maintenance:

$$\text{Minimise } \sum_{i=1}^{I} \left( f_j \sum_{i=1}^{I} p_{ij} \right) + \sum_{i=1}^{I} c_{i(x_i)}$$

Once the maintenance of unit $i$ starts, the unit must be in the maintenance state for $M_i$ contiguous periods.

$$y_{ij} = \begin{cases} 0 & \text{if } j = 1, 2, \ldots, x_i - 1 \\ 1 & \text{if } j = x_i, \ldots, x_i + M_i - 1 \\ 0 & \text{if } j = x_i + M_i, \ldots, J \end{cases}$$

If $x_i + M_i > J$, then maintenance wraps around to the next repetition of the planning horizon. This formulation captures the notion of continual maintenance. It would be possible to arrange matters so that overlap never happens; the difference is minor.
The generator output must not exceed the upper limit; the output of the generator is set to zero during the maintenance

\[ 0 \leq p_{ij} \leq C_i(1 - y_{ij}) \]

The total output must equal the demand in each period,

\[ \sum_{i=1}^{I} p_{ij} = D_j \quad \text{where} \quad j = 1, 2, \ldots, J \]

and the total capacity must not be less than the required reserve

\[ \sum_{i=1}^{I} (1 - y_{ij})C_i \geq (D_j + R_j) \]

Our formulation of the problem is based closely on that of [7].

To simplify the operation of the algorithm, all solutions in the solution space are considered to be valid. A solution could be infeasible if the demand and reserve constraints cannot be met. In that case, the solution is penalised by the addition of a penalty function:

\[ \alpha \sum_{j=1}^{J} u_j + \beta \sum_{j=1}^{J} \upsilon_j \]

where \( \alpha \) and \( \beta \) are tunable parameters and \( u_j \) and \( \upsilon_j \) are derived from the shortfall in the output,

\[ \left( \sum_{i=1}^{I} p_{ij} \right) + u_j = D_j \]

and the shortfall in the capacity

\[ \left( \sum_{i=1}^{I} C_i(1 - y_{ij}) \right) + u_j = D_j + R_j \]

\( u_j \) and \( \upsilon_j \) are not permitted to be negative, thus a feasible solution incurs no penalty function.

Thus any initial solution can be chosen and the optimisation algorithm will be directed towards the feasible solutions through the choice of sufficiently high \( \alpha \) and \( \beta \).

3. Proposed model

Parallel co-operating cultural algorithm (PARCA) is a parallelisation strategy for CAs where parallelism is obtained concurrently by several search programs. In this paper we proposed an innovative design for co-operating CAs for maintenance scheduling problem.

PARCA is implemented by using the MPI [19,20]. As depicted in Fig. 2, in a network of workstations, a master processor is in charge of creating the initial population, managing the population, performing selection, recombination and mutation. When the solutions need to be evaluated, they are dispatched to
the slave processors which manage their own executions. Once each slave processor has carried out its execution experiments, the results are returned to the master processor. Therefore PARCA is useful in exploring large space and accumulating global knowledge about the problem space. Individual experiences are collected, merged, generalised and specialised in the belief space.

In PARCA, the population is divided into several subpopulations which can be managed by their own local CA (Fig. 1). The local subpopulations can be relatively large and kept isolated from each other, so that such an approach can be well adapted to distributed computers.

In our proposed design, each subpopulation works with a different data set. The exchange of information between the populations allows them to co-operate and explore the promising areas of the search space found by the other populations, and also to reintroduce the previously lost genetic material in the population.

Different data set depends on:

a. general parameters such as population size, string size, etc.
b. operators such as mutation and crossover along with their variations, inversion, hill-climbing, or other problem-specific devised operators.
c. operators’ parameters such as crossover probabilities and mutation rates.
d. local improvement after the initialisation and after the application of the genetic operators (tabu search, simulated annealing, GLS).
e. selection mechanisms such as proportionate selection, tournament selection and steady state selection.

In the specific implementation of PARCA used in this paper, we chose 350 individuals for each population. Two crossover operators used were single and multi-point crossovers. In addition, an operator called the “phenotype mutation” tries to find a better point in the neighbourhood of the best genotype only.

During the mutation stage, two distinct operators are applied to each individual. A light mutation operator probably moves the start period to a random location. Light mutation is followed by the heavy mutation in which periods in the plan which incur a high penalty are targeted. In such periods, units which are being maintained have their maintenance rescheduled to a random starting period.

We also introduce two other operators that handle weights (genes) and not the whole strings. They both use single point crossover for each gene separating them into two parts. One of them creates the offspring taking the most significant part from the better parent and the least significant part from the worse parent.
This technique gives new weights with small perturbations in their values in the proximity of the weight value of the better parent. The other operator is the reverse of the previous one. It creates the offspring with the least significant part of the best parent and the most significant part of the worse, allowing small perturbations in the weight values in the proximity of the weight value of the worse parent. Different populations use different operators.

The populations are allowed to communicate and exchange their best individuals, each population receives only the best individual from all the other populations.

A crossover rate between the values 0.75 and 0.95 was observed [8]. The crossover probability is high when the population’s make-span range tends to get stuck at a local optimum.

When a sufficiently small probability is chosen, this has the effect of increasing the diversity while still retaining the potentially beneficial effects of the crossover operator. Too small probability causes a little diversity, while too high probability slows the next stage of the algorithm since a larger number of local search steps might be necessary before a local optima is reached.

A migration operator is used to exchange individuals among populations in order to propagate good solutions and to help unsuccessful local CAs. The communications among the local CAs are managed by MPI. The local CAs work according to the general principles of the algorithm which is modified to make it capable of sending and receiving immigrants. Migrations occur when a local cultural algorithm has performed a given number of generations without improving its performance and is therefore suspected of degenerating. The migration rate was set to 0.1 and the number of generations without improvement to 3, because we noticed (as stated for example in [22]) that few migrations that occurred are often proved to be best strategies.

Each individual in the PARCA population has both a vector of parameters that control the inference method (which we call “genes”) and a set of feature definitions, constructed as compounds of primitive features originally used to describe the problem (which we call “memes”).

In this paper three local optimisers (GLS [21] or simulated annealing or tabu search) are added to a CA, and applied to every child before it is inserted into the population (including the initial population). Recombination and mutation will usually produce solutions that are outside this space of local optima but a local optimiser can then repair such solutions to produce final children that lie within this subspace, yielding a CA. Therefore, in essence, different genetic behaviour is the result of different local optimisers for each population.

The two components interact through a communication protocol. The protocol determines the set of acceptable individuals that are able to update the belief space. Likewise the protocol determines how the updated beliefs are able to impact the adaptation of the population component. A component level description of the PARCA is given in Fig. 2.

This is the basic framework that will be used to support the definition of CAs within the test-bed. The knowledge in the belief space will correspond to that information needed to reason about the problem constraints. That knowledge is expressed in the form of constrained networks as employed in the constraint satisfaction literature.

4. Implementation and experimental results

Results of a genetic algorithm for application to this problem are relatively poor. Burke and Alistair [3] compare the same problems with simulated annealing, a genetic algorithm, tabu search and a hybrid
algorithm composed of elements of simulated annealing and tabu search. In the paper by Burke and Smith [4], three problems of varying complexity are given and a comparison is made among CA (TA), CA (SA) and CA (HC).

Here comparison is made among four versions of PARCA for these problems. The value of the objective function is known as the fitness of the individual and is defined in [1] and [5]. A more detailed description of these problems is given in [7].

Table 1 shows the results obtained by using the three methods, where CA1 uses PARCA with only tabu searching, CA2 uses PARCA with only simulated annealing, CA3 uses PARCA with only GLS and CA4 uses PARCA with different local optimisers. Times are shown as minutes:seconds.

Six populations with different behaviours were used. For convenience of reference, the populations were numbered from 1 to 6.

Their different characteristics are described below:

Population 1: Adaptive crossover probability and mutation rate (GLS for CA4).
Population 3: Adaptive crossover probability and mutation rate for two-point crossover (simulated annealing for CA4).

Table 1
Best results and times

<table>
<thead>
<tr>
<th>Method</th>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA1</td>
<td>209210 (1:25)</td>
<td>554342 (6.20)</td>
<td>1201282 (26:00)</td>
<td></td>
</tr>
<tr>
<td>CA2</td>
<td>212022 (8.11)</td>
<td>554793 (53:08)</td>
<td>1201513 (270:34)</td>
<td></td>
</tr>
<tr>
<td>CA3</td>
<td>209210 (1:00)</td>
<td>554330 (3:16)</td>
<td>1201279 (17:50)</td>
<td></td>
</tr>
<tr>
<td>CA4</td>
<td>204205 (0:38)</td>
<td>550100 (2:10)</td>
<td>1197332 (10:38)</td>
<td></td>
</tr>
</tbody>
</table>
Population 4: Adaptive mutation rate and probability of uniform crossover (GLS for CA4).

Population 5: Like population 2, but with adaptive parameter probabilities in favour of mutation.

Population 6: Constant smaller crossover and larger mutation rates (GLS for CA4).

For the test and evaluation purposes, a set comprising of 200 experiments, that is 20 experiments for 10 different number of generations, to exchange individuals was used.

The utility of the GLS can also be seen in the CAs where the addition improves the quality of the result at no cost in terms of the execution time.

Within the CAs, four local search techniques were used. The fastest local optimiser was a GLS, followed by simulated annealing and tabu search in that order.

CA3 presents intrinsically good characteristics to be paralleled. When considering the parallel approach as a whole, one might also query the effectiveness of the GLS algorithm segment of the algorithm. Due to the high computing cost generally needed to evaluate a solution in the electrical generator maintenance scheduling problem, we are interested in sharing the evaluation of different points of the search space among several processors.

In the sequential implementation, these mechanisms increase their effectiveness as the search progresses. In our scheme, one can easily see how the sharing of the memories associated with such genetic evaluation mechanisms increases their effectiveness.

Table 2 gives the number of generations necessary to obtain the optimum when one, two, four, five and six processors were used, respectively. Results show the global reduction of the number of generations needed to obtain the best value achieved, thanks to the distribution of the parallelisation of the GLS process. When more processors are used, there are fewer chances to be trapped in a local optimum.

The success rate of CA4 was doubled in the worst case (Fig. 3), that is, for the generations to exchange individuals equal to 20, the success percentage for CA4 was 90%, which means that the number of successful experiments was \(0.9 \times 20 = 18\), whereas for CA1 it was 15. For CA2 the number of successful experiments was 10 and for CA3 it was 7. This means that the number of successful experiments were 75, 50 and 35%, respectively.

Apart from the improvement in the success rate, we have also observed that the percentage of the experiments that were stuck at local optima was decreased when using CA4. Thus we can assume that CA4 improves the exploration of the search space (Fig. 4).

<table>
<thead>
<tr>
<th>Processors</th>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>940</td>
<td>1058</td>
<td></td>
<td>2024</td>
</tr>
<tr>
<td>2</td>
<td>842</td>
<td>875</td>
<td></td>
<td>1250</td>
</tr>
<tr>
<td>4</td>
<td>507</td>
<td>680</td>
<td></td>
<td>845</td>
</tr>
<tr>
<td>5</td>
<td>389</td>
<td>405</td>
<td></td>
<td>398</td>
</tr>
<tr>
<td>6</td>
<td>214</td>
<td>312</td>
<td></td>
<td>305</td>
</tr>
</tbody>
</table>
5. Conclusions

In conclusion, the experiments show that the proposed approach (PARCA) produces results which indicate that it can be usefully applied to real problems. In the proposed algorithm, the number of populations co-evolve on different processors, each one running a CA with different behaviour and using local search utilities. The exchange of information among the populations allows them to co-operate
and explore promising areas of the search space found by the other populations, and also to reintroduce previously lost cultural material in the population.

This approach offers better results at the cost of slightly increased or at the same execution time. Clearly, our approach, using different evolution behaviour and different local optimiser, produces the best results in our tests. It would be interesting to see if the technique can produce good results for the time-table problem of our University.

References