A multipopulation memetic model for the maintenance scheduling problem

Jason G. Digalakis, Konstantinos G. Margaritis

Abstract—The thermal 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 parallel co-operating memetic algorithm (PARME). In the proposed model used a variety of selection mechanism, operators, communication methods, local search procedures are applied to each solution generated by genetic operators, and parameters as it explained in the sequel. The PARME alone have been found to produce good quality results. High performance of our approach is demonstrated by applying it to maintenance scheduling problem.

Index terms—maintenance scheduling problem, parallel memetic algorithms, guided local search, simulated annealing, tabu search.

I. INTRODUCTION

The thermal 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 activities may have more than one execution mode and renewable as well as nonrenewable 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 [1,4,2].

Traditional optimization based techniques such as integer programming [17], dynamic programming [6,18] and branch and bound [19] 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 [11,12], stochastic evolution [13], genetic algorithms [13,14] and tabu search [15]. Further, hybrid GAs (HGA’s) [3,4] using neighborhood search algorithms (e.g. local search, simulated annealing and tabu search) were proposed to improve the search ability of GA’s, and their high performance was reported in the literature. In Burke et al [1,3,4], it was clearly shown that the performance of GA’s for maintenance scheduling problem was significantly improved by combining neighborhood search algorithms.

Hybrid GAs are generally able to find good solutions in reasonable amount of time, but as they are applied to harder and bigger problems there is an increase in the time required to find adequate solutions. As a consequence, there have been multiple efforts to make HGA’s faster, and one of the most promising choices is to use parallel implementations.

In this paper a parallel HGA model is proposed which uses a variety of selection mechanism, operators, communication methods, local search procedures are applied to each solution generated by genetic operators, and parameters as it explained in the sequel.

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[21]. The communicating through network consists of a SGI Origin 200 and 10 Pentium (P5/100MHz) a FastEthernet (100 MBs/s).

II. PGA AND OTHER SEARCH APPROACHES

Parallel GA and local search

PGAs are not only an extension of the traditional GA sequential model but they represent a new class of algorithms in that they make a different search work. Many different models have been proposed for the rest of PGAs (see [20-21]). Some methods can exploit massively parallel computer architectures, while others are better suited to multicomputers with fewer and more powerful processing elements. The algorithms run for a prespecified number of generations and convey the best objective function value, namely, the minimal project duration, obtained up to that generation.

The role of local search in the context of genetic algorithms and the wider field of evolutionary computing has been much discussed. In the 80’s, a new class of "knowledge-augmented GAs", sometimes called "hybrid GAs", started to appear in the literature. Moscato and
Norman [14] liken this thinking to local refinement, and therefore promote the term "memetic algorithm" to describe genetic algorithms that use local search heavily.

They have shown that they are orders of magnitude faster than traditional GAs for some problem domains. A key difference exists between genes and memes: before a meme is passed on, it is typically adapted by the person who transmits it as that person thinks, understands and processes the meme, whereas genes get passed on whole. Basically, they combine local search heuristics with crossover operators. For this reason, some researchers have viewed them as HGA. However, combinations with constructive heuristics or exact methods may also belong to this class of metaheuristics. Since they are most suitable for MIMD parallel computers and distributed computing systems (including heterogeneous systems) as those composed by networks of workstations, they have also received the dubious denomination of Parallel Genetic Algorithms.

Next we show a pseudo-code representation of a local search-based memetic algorithm.

```
Procedure Memetic Algorithm;
    begin
        initializePopulation Pop using FirstPop();
        for Each individual i in Pop do i = Local Search(i);
        repeat
            for i = 1 to #recombinations do
                Select parents from Pop;
                offspring = Recombine(parents);
                if (selectToMutate(offspring))
                    offspring = Local Search(offspring);
                Evaluatefitness(offspring);
                add offspring to Pop;
            endfor;
            until (termination condition = True);
    end;
```

Fig. 1. Pseudo-code of local-search-based genetic algorithm.

A. Tabu Search and Genetic Algorithms

Burke and Smith [4] have shown that both tabu search and a memetic algorithm can produce very good results on this problem. The tabu search algorithm has been described in detail in [2].

Burke et al [1] have implemented two different methods of tabu classification. The first method compared state of the trial solution to that of the last N solutions, thus the algorithm cannot revisit a previous solution for N iterations. The second method compared the move that took the current solution to the trial solution with those made in the last N iterations, thus the algorithm cannot perform a move more frequently than once in N iterations. N is called the tabu list size. This is referred to as move tabu.

B. Simulated Annealing and Genetic Algorithms

The simulated annealing approach proposed by Kirkpatrick et al [11] as an effective method for finding the global minima of a combinatorial problem. Its basic feature is the possibility of exploring the solution space of the optimisation problem by allowing non-improving moves and is based on the analogy of the physical process of annealing, a process for reducing the temperature of a material to obtain a state with minimum energy[4].

The only difference between this algorithm and the simulated annealing algorithm described from Burke et al [1] is that once a neighbourhood solution is generated, it only becomes a candidate to be accepted if it does not appear in the list of recently accepted solutions.

D. Hill climbing and Genetic Algorithms

Hill climbing heuristics attempt to improve a solution by moving to a better neighbor solution. Whenever the neighboring solution is better than the current solution, it replaces the current solution. Genetic algorithms and hill-climbing heuristics have complementary strong and weak points.

Hill climbing heuristics, on the other hand, are good at fine-tuning, but lack a global perspective. Practice has shown that a hybrid algorithm that combines GAs with hill-climbing heuristics often results in an algorithm that can outperform either one individually.

Burke and Smith [4] to determine the effectiveness of tabu search approach, a steepest-first hill climbing algorithm was also tested. This algorithm was derived from the tabu search approach by setting the size of the tabu list to zero, and removing the multistage components of the algorithm.

E. Guided Local search and Genetic Algorithms

For local improvement after initialization and after application of the genetic operators, we use the GLS based on the ideas of Moscato and D. Holstein [24]. The GLS procedure is applied to the new solution after the mutation [10].

GLS [16] introduces constraints that modify the landscape and guide local search out of local minima. Constraints are based of information gathered during the search process. For example, if local search reaches a local minimum then a guess can be made that the global minimum is unlikely to reside in the surrounding area. Constraints could be then introduced that exclude this area from being searched in future iterations. These constraints are essentially soft because we can not be sure that local search thoroughly searches the space around the local minimum solution.

If we want to effeciently utilize the global search ability of GAs in our approach, we have reduce the computation time spent by the local search. This can be realized by restricting the number of neighborhood solutions examined by the local search procedure.

In our approach after the genetic operators, the current population is replaced with improved population by the guided local search, tabu search or simulated annealing. It has been noted in a past algorithm comparison that GLS has been particularly effective with this problem, thus it was a natural starting choice for the local search element of the PARME, however as a comparison, the next best algorithm,
simulated annealing, was also considered, along with a basic hill-climbing algorithm.

V. PROPOSED MODEL

PARME is a parallelization strategy for memetic algorithms where parallelism is obtained by concurrently several search programs. In this paper we proposed an innovative design for co-operating memetic algorithms for maintenance scheduling problem.

PARME is implemented by using the message-passing interface (MPI) [21,22]. As depicted in figure 2 on a network of workstations, a master processor is in charge of creating the initial population, managing the population, performing selection, recombination, mutation. When solutions need to be evaluated, they are dispatched to 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.

In PARME the population is divided in several sub populations, which can be managed by their own local memetic algorithm. The local subpopulations can be relative large and kept relatively isolated from each other, so that such an approach is well adapted to distributed memory 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 exploit promising areas of the search space, found by the other populations, and also reintroduce in the population previously lost genetic material.

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, mutation rates.

d. Local improvement after initialization and after application of the genetic operators (tabu search, simulated annealing, guided local search).

e. Selection mechanisms such as proportionate selection, tournament selection and steady state selection.

In the specific implementation of PARME used in this paper we chose 400 individuals for each population. Two crossover operator are used single and multi point crossover. In addition an operator called “phenotype mutation” tries to find a better point in the neighborhood of the best genotype only.

During the mutation stage, two distinct operators are applied to each individual. A light mutation operator probabilistically moves a start period to a random location. Light mutation is followed by heavy mutation in which periods in the plan which incur a high penalty are targetted.

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 (memes) and not whole strings. They both use single point crossover for each gene separating it 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. This technique gives new weights with small perturbations of 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 for small perturbations of the weight values in the proximity of the weight value of the worse parent. Different populations use different operators.

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

![Fig 2 : PARME Model.](image)

A crossover rate between the values 0.75 and 0.95 [8]. The crossover probability is increase when the population’s makespan range tends to get stuck at a local optimum.

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

A migration operator is used to exchange individuals between populations, in order to propagate good solutions and to help unsuccessful local memetic algorithms. The communication between local memetic algorithms are managed by MPI. The local MAs work according to the general principles of the algorithm modified in order to be capable of sending and receiving immigrants. Migrations occur when a local memetic algorithm has performed a given
number of generation without improving its performances and is therefore suspected of degenerating. The migration rate was set to 0.1 and the number of generations without improvement to three, because we noticed (as stated for example in [25]) that few migrations occurring often seems to be the best strategy.

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

In this paper three local optimizers (guided local search or simulated annealing or tabu search) are added to a memetic algorithm, 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 optimizer can then repair such solutions to produce final children that lie within this subspace, yielding a memetic algorithm. Therefore in essence different genetic behaviour is the result of different local optimizers for each population.

VI. IMPLEMENTATION AND EXPERIMENTAL RESULTS

Results for the application of a genetic algorithm to this problem are relatively poor. Burke et all [3] compares the same problems with simulated annealing, a genetic algorithm, tabu search and a hybrid algorithm composed with elements of simulated annealing and tabu search.

In the paper Burke and Smith [4] three problems of varying complexity are given and a comparison performed between MA(TA), the MA(SA), and MA(HC).

Here comparison is performed between four versions of PARME for theses problems. The value of the objective function is known as the fitness of the individual and is defined as the sum of [1] and [5] in Section III. A more detailed description of these problems is given in [7].

Table 1 shows the results obtained by using the three methods, where MA1 uses PARME with only tabu searching, MA2 uses PARME with only simulated annealing, MA3 uses PARME with only GLS and MA4 uses PARME with different local optimisers. Times are shown in minutes:seconds.

Six populations with different behaviours were used. For convenience of reference, the populations are numbered from 1 to 10.

Their different characteristics are described below:

Population 1: Adaptive crossover probability and mutation rate. (Guided Local Search for MA4).


Population 4: Adaptive mutation rate and probability and uniform crossover. (Guided Local Search for MA4).

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

Population 6: Constant smaller crossover and larger mutation rates. (Guided Local Search for MA4)

Population 7: Like population 3, but with uniform crossover. (Guided Local Search for MA4).

Population 8: Adaptive mutation rate and adaptive crossover probability, (Tabu Search for MA4)

Population 9: Like population 2, but with adaptive parameter probabilities in favour of mutation.

Population 10: Adaptive mutation rate and adaptive crossover probability, (Guided Local Search for MA4)

For test and evaluation purposes using a set comprises 200 experiments that is 20 experiments for 10 different values of number of generations to exchange individuals.

Table 1: Best results and times

<table>
<thead>
<tr>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA1</td>
<td>504920</td>
<td>554342</td>
<td>1201282</td>
</tr>
<tr>
<td>MA2</td>
<td>512202</td>
<td>547933</td>
<td>1201513</td>
</tr>
<tr>
<td>MA3</td>
<td>504920</td>
<td>543310</td>
<td>1201279</td>
</tr>
<tr>
<td>MA4</td>
<td>501145</td>
<td>510253</td>
<td>1045341</td>
</tr>
</tbody>
</table>

The utility of the guided local search can also be seen in the memetic algorithms where the addition an improvement in the quality of the result at no cost in terms of the execution time.

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

MA3 present intrinsically good characteristics to be parallelized. 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 costs generally needed to evaluate a solution in thermal generator maintenance scheduling problem, we are interested in sharing the evaluation of the different points of the search space among several processors.

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

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

The success rate of MA4 was, in the worst case, doubled (fig. 3) that is for number of generation to exchange
individuals equal to 20 the success percentage for MA4 was 90% which means that the number of successful experiments was 0.9x20=18, whereas for MA1 it was 15. For MA2 the number of successful experiments was 10 and for MA3 it was 7. The means that the number of successful experiments was 75%, 50%, 35% respectively.

Table 2: Generations for three problems using 1-10 processors

| Problem | Processors | |
|---------|------------|--
|         | 1          | 2     | 3     |
| 1       | 998        | 1000  | 1000  |
| 2       | 850        | 868   | 863   |
| 4       | 602        | 645   | 700   |
| 5       | 580        | 568   | 670   |
| 6       | 470        | 513   | 587   |
| 7       | 368        | 483   | 498   |
| 8       | 284        | 346   | 370   |
| 9       | 198        | 213   | 238   |
| 10      | 145        | 180   | 210   |

Fig 3. Success percentage

VII. CONCLUSIONS

In conclusion, the experiments show that the proposed approach (PARME) produces results that 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 MA with different behaviour and local search utilities. The exchange of information between the populations allows them to cooperate and exploit promising areas of the search space, found by the other populations, and also reintroduce in the population previously lost memetic material.

An issue we examine in this paper is to what degree the performance of the different evolution behaviour and local search utilities employed effects the performance of the MA.

GLS drove the MA to its best value in fewer generations. The performance of different behaviour in PARME approach significantly improved and when using PARME the percentage of the experiments that were stuck at local optima was decreased, and it kept decreasing as the generation value was increasing.

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

REFERENCES