MEMETIC ALGORITHMS TO MINIMIZE TARDINESS ON A SINGLE MACHINE WITH SEQUENCE-DEPENDENT SETUP TIMES

Paulo França, Alexandre Mendes and Pablo Moscato

Departamento de Engenharia de Sistemas - DENSIS
Faculdade de Engenharia Elétrica e de Computação - FEEC
Universidade Estadual de Campinas - UNICAMP
e-mail:{franca, smendes, moscato}@densis.fee.unicamp.br

ABSTRACT

The Single Machine Scheduling (SMS) problem is one of the most representative problems in the scheduling area. In this paper, we explore the SMS with time constraints (setup times and due-dates) and the objective function is the minimization of total tardiness. The chosen method is based on a *hybrid-genetic algorithm*, which belongs to the *memetic algorithms* class. Results are compared against *genetic algorithms*, *multiple start*, the constructive heuristic ATCS proposed by Lee *et al.* (1997) and an *EDD Insertion* procedure.

1. INTRODUCTION

Under the generic denomination of *single machine scheduling* we understand one of the first studied class of problems in the scheduling area as the survey paper by Graves [3] shows. The SMS problem with sequence-dependent setup times is present in many industrial manufacturing systems, as pointed out in [6]. In this problem, the processing of a job requires a setup time which depends on the predecessor job.

The SMS problem can be described as:
Input: Let n be the number of jobs, indexed 1, 2, ..., n, to be processed in one machine. Consider that all jobs are available for processing at time zero. Given a list $\{p_1, ..., p_n\}$ of processing times and a list $\{d_1, ..., d_n\}$ of due dates. Given a matrix $\{s_{ij}\}$ of setup times where s_{ij} is the time required to set up job j after job i has just finished. It is assumed that s_{ij} need not to be equal to s_{ji}.
Task: Find a permutation that minimizes the total tardiness of the schedule, which is calculated as:

$$T = \sum_{k=1}^{n} \max[\,0, c_k - d_k\,]$$

where c_k is the completion time of job k.

We should note that the problem of sequencing jobs in one machine without setup times is already NP-hard. Ragatz [7] proposed a *branch-and-bound* (B&B) method but only small instances can be solved to optimality. The paper of Lee *et al.* [4] uses a *dispatching rule* based on the calculation of priority index to build an approximate schedule, which is then improved by the application of a local search phase. Two papers using metaheuristics have been proposed so far. Rubin and Ragatz [8] developed a new crossover operator and applied a *genetic algorithm* (GA) to a set of test problems. The results obtained by the GA approach were compared with the ones from a B&B and with a *multiple start* (MS) and they concluded that MS outperformed B&B and GA in many instances considering processing time and quality of solutions as performance measures. That is the reason why Tan and Narasimhan [9] chose MS technique as a benchmark for conducting comparisons with the *simulated annealing* (SA) approach they proposed. They concluded that SA outperformed MS in all but three instances with percentage improvements not greater than 6%.
In this paper we propose a *memetic algorithm* (MA) for solving the SMS with sequence-dependent setup times. MAs are a class of population-based algorithms that generalizes the well-known *hybrid genetic algorithms*, which combines the strength of the GAs and local search.

2. NEIGHBORHOOD CONCEPTS

The first neighborhood implemented was the *All-Pairs Interchange*. It consists in swapping all pairs of jobs in a given solution. A '*hill-climbing*' algorithm can be defined by reference to this neighborhood; *i.e.* starting with an initial permutation of all jobs, every time a proposed interchange reduces the makespan it is confirmed and another cycle of interchanges takes place, until no improvement is obtained anymore. The second neighborhood implemented was the Insertion neighborhood. It consists of removing a job from one position and inserting it into another one. The '*hill-climbing*' procedure is the same of the All-Pairs scheme.

Neighborhood reduction

As the neighborhood schemes are O(n^2), we tried several reduction policies. The best one turned out to be a hybrid scheme, with reduced All-Pairs and Insertion neighborhoods. These reduction schemes are based on s_{ij} values only and do not take into account the due-dates. Nevertheless the results showed considerable improvements over non-reduced neighborhoods.

3. MULTIPLE START AND EDD INSERTION

The *multiple start* algorithm implemented in this work consists in generating an initial random solution and making it converge to a local minimum by applying a local search procedure. The process is iterated many times until a stop criterion (a time limit criterion was used) is satisfied and the best solution ever found is reported.

The *EDD Insertion* is a constructive procedure. It consists of inserting the jobs according to the EDD sequence and always in the sequence position that leads to the smallest partial total tardiness.

4. MEMETIC AND GENETIC ALGORITHMS

GAs got recognition in the mid 70´s after John Holland's book entitled '*Adaptation in Natural and Artificial Systems*' was published. Since then, due to its simplicity and efficiency, GAs became widespread, and in the 80´s, a new class of '*knowledge-augmented GAs*', sometimes called '*hybrid GAs*', started to appear in the literature. Recognizing important differences and similarities with other population-based approaches, some of them were categorized as memetic algorithms (MAs) in 1989 [5].

Population structure

A population structure approach based on a ternary tree was chosen. In contrast with a non-structured population it divides the individuals in clusters and restricts crossover possibilities.

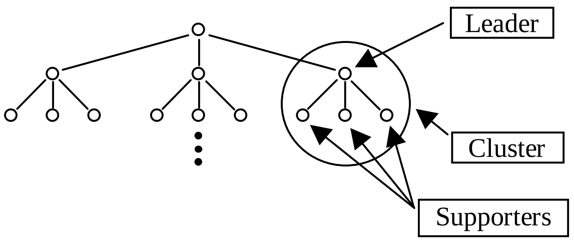

Figure 1. Population structure

The structure consists of several clusters and each cluster consists of a leader and three supporter solutions. The leader is chosen as the best individual of the cluster. The number of individuals in the population is defined by the number of nodes in the ternary tree, *i.e.*, it is necessary 13 individuals to make a ternary tree with 3 levels, 40 individuals to 4 levels and so on.

Representation

For the SMS problem the representation we have chosen is quite intuitive, with a solution being represented as a chromosome with the alleles assuming different integer values in the [1, n] interval, where n is the number of jobs.

Crossover

Two different crossover operators were implemented. The first is the well known *Order Crossover* (OX). After choosing two parents, a fragment of the chromosome from one of them is randomly selected and copied into the offspring. In the second phase, the offspring's empty positions are sequentially filled according to the chromosome of the other parent.

The second crossover we implemented was proposed by Rubin and Ragatz for exactly this problem [8] and as they didn't name it we chose to refer to it as RRX. According to it, the recombination of any pair of parents always generates 8 new individuals so it is necessary to select which of them must be incorporated in the population. They made comparisons with several insertion policies and all performed similar. Eventually, they suggest the random choice of one of the offspring.

Mutation

In our implementation a traditional mutation strategy based on jobs swapping was implemented. According to it, two positions are randomly selected and the alleles in these positions swap their values.

Fitness function

In our problem the objective is to minimize the total tardiness and the fitness function was chosen as:

$$f_i = (T_i + 1)^{-1}$$

where T_i is the total tardiness of solution i. The inversion is due to the fact that a large total tardiness means a low fitness. The total tardiness was increased by one to prevent a division by zero when $T_i = 0$. A fitness function similar to this was also used in [1] and [8].

Offspring insertion in population

Once the leader and one supporter are selected, the recombination, mutation and local search take place and an offspring is generated. If the fitness of the offspring is better than the leader, the new individual takes its place. Otherwise it takes the place of the supporter that took part in the recombination. If the new individual is already present in the population, it is not inserted. We adopted a policy of not allowing duplicated individuals to reduce loss of diversity. After all individuals were inserted, the population is restructured. The fitness of the leader of a group must be lower than the fitness of the leader of the group just above it. Following this policy, the higher subgroups will have leaders with better fitness than the lower groups and the best solution will be the leader of the root subgroup. The adjustment is made by comparing the leader of each subgroup with the leader of the subgroup just above. If the leader in the level below turns out to be better, they swap their places.

5. COMPUTATIONAL RESULTS

Instances are characterized by three parameters: size, due date range factor (R) and tardiness factor (τ). The factor R controls the range of the due date distribution while τ provides an indication of the average tightness of the due dates. The generation of processing times and setup times followed a discrete uniform distribution: DU(0, 100). The definitions for the due date factor and tardiness factor adopted in our experiments are the same used in [6]. The due date's interval Δd and mean d_m is given by the equations:

$$\Delta d = R.n.p_m \qquad d_m = (1 - \tau).n.p_m$$

where p_m is the mean processing time.

We chose 9 combinations for (τ, R) pairs and CPU time was fixed in 2 minutes for MS, GAs and MAs. The software was implemented using Sun Java JDK 1.2 and run using a PC-Compatible PENTIUM II 266 MHz.

Each value in table one is the mean result for a set of 5 runs for each size of instance – 20, 40, 60, 80 and 100 jobs. The performance was calculated as the percentage improvement over the MS solution using *All-pairs Interchange* local search without neighborhood reduction.

Table 1: Mean results for MA, GA, ATCS and EDD Insertion

	Genetic Alg.		**Memetic Alg.**			
(τ, R)	**RRX**	**OX**	**RRX**	**OX**	**ATCS**	**EDD_{ins}**
(0.2, 0.2)	-58.6	1.9	14.0	32.8	-22.8	-31.7
(0.2, 0.6)	-71.9	-13.6	26.9	59.2	-49.7	-93.3
(0.2, 1.0)	-129.2	-74.1	37.9	71.1	-71.8	-161.4
(0.6, 0.2)	-17.5	-3.5	4.0	11.7	-5.0	-10.8
(0.6, 0.6)	-16.7	-5.5	4.0	13.6	-7.1	-5.4
(0.6, 1.0)	-23.3	-5.6	2.9	13.7	-8.7	-10.7
(1.0, 0.2)	-11.4	-2.9	1.4	5.3	-0.5	-5.0
(1.0, 0.6)	-8.9	-3.3	1.6	5.2	-1.2	-4.6
(1.0, 1.0)	-7.8	-2.9	2.2	5.6	-0.2	-3.2
Mean	**-31.8**	**-12.1**	**10.5**	**24.2**	**-18.6**	**-32.7**

6. CONCLUSIONS

Results on GA and MA indicate the OX as the best crossover operator. The RRX crossover performed poorly, probably because of the quick loss of diversity in this crossover. GA with OX lost for the MS in almost all cases, although the loss was close, rarely by more than 10%. On the other way, the MA performance was much better, always outperforming MS. For $\tau = 0.2$ the percentage results were extremely high. That is because in these cases, the optimal total tardiness values zero or lies near zero. Therefore, any deviation will return very high percentage values. The ATCS dispatching rule did well, beating the EDD Insertion strategy in all but one set of data. Its performance gets better as the number of jobs increases. For more than 80 jobs the ATCS can rival MS and beat GA in almost all parameter configurations.

The overall performance of the different methods was: MA was the best one, followed by the GA with OX operator; ATCS came right next followed by the other GA; the worst performance was obtained by EDD Insertion rule.

Acknowledgements

This work would have not been possible without the support of *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq) and *Fundação de Amparo à Pesquisa do Estado de São Paulo* (FAPESP).

7. REFERENCES

[1] Cheng, R. and Gen, M., Parallel Machine Scheduling Problems Using Memetic Algorithms, *Computers & Ind. Eng.* (33), n. 3-4, pp. 761-764 (1997).

[2] Du, J. and Leung, J. Y. T., Minimizing Total Tardiness on One Machine is NP-hard. *Math. Opns. Res.* (15), pp. 483-495 (1990).

[3] Graves, S. C., A Review of Production Scheduling, *Operations Research* (29), pp. 646-675 (1981).

[4] Lee, Y. K., Bhaskaran, K. and Pinedo, M., A Heuristic to Minimize the Total Weighted Tardiness with Sequence-dependent Setups, *IIE Transactions* (29), pp. 45-52, (1997).

[5] Moscato, P., On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, *Caltech Concurrent Computation Program, C3P Report 826* (1989).

[6] Ow, P. S. and Morton, T. E., The Single Machine Early/Tardy Problem, *Management Science* (35), n. 2, pp. 177-191, (1989).

[7] Ragatz, G. L., A Branch-and-Bound Method for Minimum Tardiness Sequencing on a Single Processor with Sequence Dependent Setup Times. *In Proceedings: Twenty-fourth Annual Meeting of the Decison Sciences Institute*, pp. 1375-1377, (1993).

[8] Rubin, P. A. and Ragatz, G. L., Scheduling in a Sequence Dependent Setup Environment with Genetic Search, *Computers & Ops. Res.* (22), n. 1, pp. 85-99 (1995).

[9] Tan, K. C. and Narasimhan, R., Minimizing Tardiness on a Single Processor with Sequence-dependent Setup Times: a Simulated Annealing Approach, *OMEGA* (25), n. 6, pp. 619-634 (1997).