30

A GENETIC ALGORITHM FOR THE DYNAMIC SINGLE MACHINE SCHEDULING PROBLEM

Ana Madureira[1], Carlos Ramos[1], Sílvio do Carmo Silva[2]
[1]*Instituto Superior de Engenharia do Porto, Dept. de Engenharia Informática*
Rua de São Tomé, 4200 Porto-Portugal, {anamadur,csr}@dei.isep.ipp.pt
[2]*Universidade do Minho, Dept. de Produção e Sistemas*
4710-057, Braga - Portugal, scarmo@ci.uminho.pt

This paper starts by studying the performance of two interrelated genetic algorithms (GA) for the static Single Machine Scheduling Problem (SMSP). One is a single start GA, the other, called MetaGA, is a multi-start version GA. The performance is evaluated, for total weighted tardiness, on the basis of the quality of scheduling solutions obtained for a limit on computation time. Then, a scheduling system, based on Genetic Algorithms is proposed, for the resolution of the dynamic version of the same problem. The approach used adapts the resolution of the static problem to the dynamic one in which changes may occur continually. This takes into account dynamic occurrences in a system and adapts the current population to a new regenerated population,

1. INTRODUCTION

The SMSP is a class of scheduling problems that deals with sequencing a set of jobs on a single processor or machine.

The study of SMSP is important by itself and also, among other reasons, because it can provide help and insight into the resolution, understanding, managing and modelling more complex multi-processor problems. In fact, quite often, the single-machine problem appears as a component in larger scheduling problems (Baker, 1974). Sometimes SMSP are independently solved and results incorporated into larger and more complex problems. For example, in multistage multiple machine problems there are often critical machines, i.e. bottlenecks, whose processing capacity are lower than the necessary. The analysis and treatment of bottlenecks as single-machines may determine the properties of the entire schedules for complex systems (Goldratt, 1986), (Adams, 1988), (Brownie, 1988) This treatment can be applied to manufacturing systems comprehending a network of distributed manufacturing units, or virtual enterprises (Camarinha-Matos, 1999). Here too, a

critical unit may be identified in which the work schedule can restrict the properties of the work schedule in the entire network of units.

Manufacturing activity is frequently subject to several sorts of random occurrences and perturbations, such as random job releases, machine breakdowns, jobs cancellation and due date and time processing changes. Thus, the scheduling of manufacturing operations in practice is, essentially, a dynamic problem.

Due to their dynamic nature, real scheduling problems have an additional complexity in relation to static ones. In many situations these problems, even for apparently simple situations, are hard to solve, i.e. the time required to compute an optimal solution increases exponentially with the size of the problem (Morton, 1993).

In the *static* SMSP all jobs are known before processing starts. In the *dynamic* SMSP job release times are not fixed at a single point in time, i.e. jobs arrive to the system at different times. Scheduling problems can also be classified as *deterministic,* when processing times and all other parameters are known and fixed, and *stochastic,* when some or all parameters are uncertain (French, 1982). Here, we deal with these two cases. The adequacy and efficiency of the Genetic Algorithms (GA), for a particular version of this class of problems, is studied.

Some previous GA studies for the static case of the SMSP are referred in (Davis, 1991) (Madureira, 1999) (Michalewicz, 1992) (Morton, 1993).

2. TOTAL WEIGHTED TARDINESS SMSP

First we start with the static SMSP for minimising total Weighted Tardiness. The following assumptions are considered: a set of n independent jobs ($j=1,...,n$) is available for processing at time zero and job attributes are known in advance; a machine is continuously available and is never kept idle if working is waiting; a machine can handle one job at a time; job set-up times are independent of job sequence and included in job processing times; jobs are processed to completion without pre-emption.

For each job j, let p_j be its processing time, d_j its due date and w_j the penalty incurred for each unit of time late. The processing of the first job begins at time $t=0$. The tardiness of a job is given by $T_j = Max\ \{t_j+p_j-d_j,\ 0\}$ where t_j is the start time of job j. The objective is:

$$Min\ \Sigma w_j T_j,\ with\ T_j = Max\ \{t_j+p_j-d_j,\ 0\} \qquad (1)$$

This problem is a combinatorial optimisation NP-complete problem (Lawer, 1977). Optimal algorithms for this problem would require a computational time that increases exponentially with the problem size (Baker, 1974), (Rinnooy Kan, 1975), (Abdul-Razaq, 1990), (Potts, 1991).

3. A GENETIC ALGORITHM FOR THE SMSP

Genetic Algorithms (GA) were originally proposed by Holland (Holland, 1992). In discovering good solutions to difficult problems, these algorithms mimic the biological evolution process.

A GA is based on populations of solutions. Initially a population is selected by some mechanism. Then, the GA generates other solutions, which tend to be better, by combining chromosomes, i.e. solutions, through the use of genetic operators for selection, crossover and mutation.

The interest of GA is that although guarantees of optimal solutions to problems cannot be given, good solutions are likely to be obtained within the time available to get one when using GA (Crauwels, 1998). So, GA can be both robust and flexible in solving hard problems such as the one here considered.

In developing a genetic algorithm we must have in mind that its performance depends largely on careful design and set-up of the algorithm components, mechanisms and parameters. This includes genetic encoding of solutions, initial population of solutions, evaluation of the fitness of solutions, genetic operators for the generation of new solutions and parameters such as population size, probabilities of crossover and mutation, replacement scheme and number of generations.

3.1 Solution Encoding

In this study, solutions are encoded by the natural representation (Bagchi, 1991). In this representation each gene represents a job index. The gene position in a chromosome represents the job position in a scheduling solution, i.e. in a sequence, defining, therefore, the job processing order or priority. The number of genes in the chromosome represents the number of jobs in a solution.

3.2 Initial Population Generation

An initial solution is generated by a procedure, which we call *Randomized Earliest Due Date* (REDD). This generates solutions by exchanging pairs of jobs, represented by their indexes, randomly chosen from a sequence of jobs established by the Earliest Due Date rule.

Thus, we expect to generate a good initial solution from which an initial population will be developed. This is done using a neighbourhood generating mechanism which *exchanges jobs not apart more than af positions*. The distance *af* is dependent on problem size and is given by $af = p^*n$, where p is a percentage and n the number of jobs. This generating mechanism produces a population size of N different individuals or solutions, with:

$$N = (n\text{-}1) + \sum_{i=n-af}^{n-2} i \qquad (2)$$

Thus, in a solution with n jobs, new solutions are successively generated by exchanging the *first* job with the next *af* jobs, then the *second* with the next *af* jobs, and so on, until the last solution is obtained by exchanging the *(n-1)*th job with job

*n*th. For example, if the initial solution is x_1 = [5 4 2 3 1], and p =60% than the population $N(x_1)$ is generated by exchanging 2 jobs not apart more than 3 positions, i.e. af = 60%**n*, with n=5. Therefore the following 9 solutions are obtained: [4 5 2 3 1], [2 4 5 3 1], [3 4 2 5 1], [5 2 4 3 1], [5 3 2 4 1], [5 1 2 3 4], [5 4 3 2 1], [5 4 1 3 2], [5 4 2 1 3].

In GA the quality of results tends to increase with population size. However, the time required for the search procedure also increases. Therefore, when using GA the size of the population can inhibit obtaining satisfactory solutions within an acceptable time frame. Having this in mind, a sub-population with size N' is used in this work, which is a quarter of the population size N above referred. The N' solutions are randomly chosen from the whole population. The parameter definition presented in this section was based on previous work (Madureira, 1999).

3.3 Genetic Operators

Selection

Individuals, i.e. solutions, are selected from the population and combined to produce descendants.

We start by comparing two selection methods, namely the Roulette Wheel Selection (RWS) (Davis, 1991) and Random Selection (RS).

The computational tests involved some instances of the Weighted Tardiness problem with 40 and 50 jobs extracted from OR-Library (in website1).

From the tests carried out the advantages of the RWS in relation to RS, were not clear. In fact, there were minor differences between both selection mechanisms. Moreover, both obtained the best solutions on the same number of instances. We noticed, however, that the RS method was faster and simpler to implement than RWS. For this reason we used this method.

Crossover

Depending on the problems to solve and their encoding, several crossover operators may be used namely one point, two points, uniform and order crossover (Davis, 1991).

Here, we use the order crossover operator. This chooses randomly two points in a chromosome between which the characters, i.e. the genes, and their relative positions are kept unchanged in the offspring, (Pirlot, 1992). The order crossover operator will be applied to M pairs of solutions randomly chosen, with M=N/2, where N is the size of the population.

Mutation

As mutation operator we use the inversion mechanism to prevent the lost of diversity. Thus, two points in a chromosome are randomly selected between which the order of genes, i.e. the processing order of the jobs, is reversed.

In this work some computational tests were carried out for establishing the value of the mutation probability, which determines the mutation rate.

The tests were performed involving several instances of the Weighted Tardiness problem for 40 and 50 jobs as previously referred. The values of the probabilities tested were 0.1, 0.01 and 0.001.

Probability of 0.1 proved to be better for the tests carried out. This may be explained by the larger diversity of individuals generation due to the larger probability.

3.4 Replacement Scheme

When creating a new population by crossover and mutation we must avoid loosing the best chromosomes or individuals. To achieve this, the replacement of the less fit individuals of the current population by offspring is based on elitism (Davis, 1991) (Michalewicz, 1992). Thus, the best individuals, i.e. solutions, will survive into the next generation. However, duplicated solutions may occur. To minimise this, the inversion operator is applied to all duplicated solutions.

4. COMPUTATIONAL RESULTS

This section presents the results of the GA on the resolution of a set of instances of the Weighted Tardiness problem.

A software tool was developed to perform the computational study. The GA was coded in C language and the computational tests were carried out on a PC Pentium Pro 180Mhz

The behaviour of the different parameters of the GA was evaluated through a set of preliminary computational tests and their values were tuned according results. The computational tests involved 10 instances of the Weighted Tardiness problem with 40 and 50 jobs. As stopping criteria for the GA search we use a limit on the number of generations. 300 and 450 were used respectively for the 40 and 50 job problems.

In order to analyse and discuss the performance of the GA, the best value of the Weighted Tardiness and its deviation from the optimal were compared, Table 1.

Table 1 – Computational results from GA

Problem	wt40a	wt40b	wt40c	wt40d	wt40e	wt40f	wt40g	wt40h	wt40i	wt40j	wt50a	wt50b	wt50c	wt50d	wt50e	wt50f	wt50g	wt50h	wt50I	wt50j
Optimal	913	1225	537	2094	990	6955	6324	6865	16225	9737	2134	1996	2583	2691	1518	26276	11403	8499	9884	10655
Best Value	1012	1477	537	2094	990	7158	7005	7133	18298	11557	2447	2009	2619	2878	1640	26904	11431	9071	10556	10964
Dev.BestValue	10.8%	20.6%	0.0%	0.0%	0.0%	2.9%	10.8%	3.9%	12.8%	18.7%	14.7%	0.7%	1.4%	6.9%	8.0%	2.4%	0.2%	6.7%	6.8%	2.9%

The results show that the algorithm achieved good performance for most instances of the problem. Optimal solutions were even obtained for instances wt40c, wt40d and wt40e. The GA achieved an average deviation of 7% from the best solution to the optimal. The computation times were 7 and 24 minutes respectively for problems with 40 and 50 jobs.

Large variations can be noticed on the results. This may be due to the strong randomness introduced in the algorithm and premature convergence. With a higher mutation probability, to introduce more diversity and avoid premature convergence, the performance of the algorithm might improve.

4.1 The MetaGA Algorithm

In an attempt to improve the quality of the GA a multi-start version of the GA, called MetaGA, was tested. With this version we run independently the GA, several times, with different initial populations. In this case, we define three generations. From the obtained results, two further measures were collected, i.e. the average value and its deviation from the optimal.

An important improvement in the results was obtained with the MetaGA, for the same problem instances, Table 2. Deviations were on average less then 3% from the best to the optimal solution and 7 optimal solutions were found.

Table 2 – Computational results from MetaGA

Problem	Wt40a	wt40b	wt40c	wt40d	wt40e	wt40f	wt40g	wt40h	Wt40i	wt40j	wt50a	Wt50b	wt50c	wt50d	wt50e	wt50f	wt50g	Wt50h	wt50i	wt50j
Optimal	913	1225	537	2094	990	6955	6324	6865	16225	9737	2134	1996	2583	2691	1518	26276	11403	8499	9884	10655
Best Value	913	1373	537	2094	990	7158	6603	7046	17003	10621	2356	2009	2583	2691	1518	26441	11431	9071	9983	10964
Dev. Best Value	0.0%	12.1%	0.0%	0.0%	0.0%	2.9%	4.4%	2.6%	4.8%	9.1%	10.4%	0.7%	0.0%	0.0%	0.0%	0.6%	0.2%	6.7%	1.0%	2.9%
Average	960	1466	552	2479	1178	7359	7073	7099	17748	10979	2389	2049	2658	2798	1578	26883	12202	9200	10198	12168
Dev.Average	5.1%	19.7%	2.8%	18.4%	19.0%	5.8%	11.8%	3.4%	9.4%	12.8%	11.9%	2.7%	2.9%	4.0%	4.0%	2.3%	7.0%	8.2%	3.2%	14.2%

As we can see from the results presented in Table 2, the obtained average values were better than the values obtained from single start GA, presented on Table 1, in most of the instances. The deviations from the average value to the optimal were on average less then 8%. This may be considered a relatively good performance considering the extreme behaviours of the algorithm.

The set of tests carried out suggests that applying this MetaGA approach to genetic algorithms may improve their performance. This tends to offer progressive improvements of solutions with moderate additional computational effort. In this way, good balance between computational effort and quality of solutions can be achieved.

5. DYNAMIC SINGLE MACHINE SCHEDULING PROBLEM

The static SMSP refers to the situation in which all jobs are simultaneously available for processing. The complexity increases when the dynamic problem is to be solved.

Considering different job release times, the completion time C_j of a job j in a sequence can be given by:

$$C_j = Max\{C_{j-1}, r_j\} + p_j \qquad (3)$$

where r_j is the release time of j.

Some machine idle time may result from unavailability of the next job in the sequence. The machine idleness between job j and $j-1$ can be given by:

$$Idleness\ (j, j-1) = Max\{(r_j - C_{j-1}), 0\} \qquad (4)$$

We can see the static SMSP as a relaxation of the dynamic SMPS by considering release times equal to zero, i.e. r_j=0, for all j.

5.1 Scheduling Dynamic SMSP

In a dynamic environment frequent rescheduling of work is necessary due to variations on working conditions and requirements over time. This is due to many random events or disturbances as previously referred.

In this work, a scheduling system based on a GA is designed to react to such random events. These could be classified in two categories:

- **Partial events**, which imply changes in job attributes, such as job processing times, due dates and release times;
- **Total events**, which imply changes in population structure, such as new job arrivals and jobs cancellation.

While *partial* events only require a modification procedure to redefine job attributes and a re-evaluation of the fitness function of solutions, *total* events require a modification on chromosome structure and size, by inserting or deleting genes, as well as the re-evaluation the fitness function. Therefore, under a *total* event the modification of the solutions population is imperative. In this work, this is carried out by the mechanisms described in section 3.2.

Rescheduling from the beginning is normally to be avoided, considering the processing times involved and the frequency of the rescheduling. However if work has not yet started and time is available, then an obvious and simple approach to rescheduling would be to restart the scheduling from scratch with the new modified population resulting from a disturbance, i. e. a new job arrival. When there is not enough time to reschedule from scratch or job processing has already started, a re-use approach must be considered which builds on the current schedule.

These views are imbedded in our proposed approach embodied into a dynamic scheduling system structured in three modules, namely pre-processing, scheduling and rescheduling modules, Figure 1.

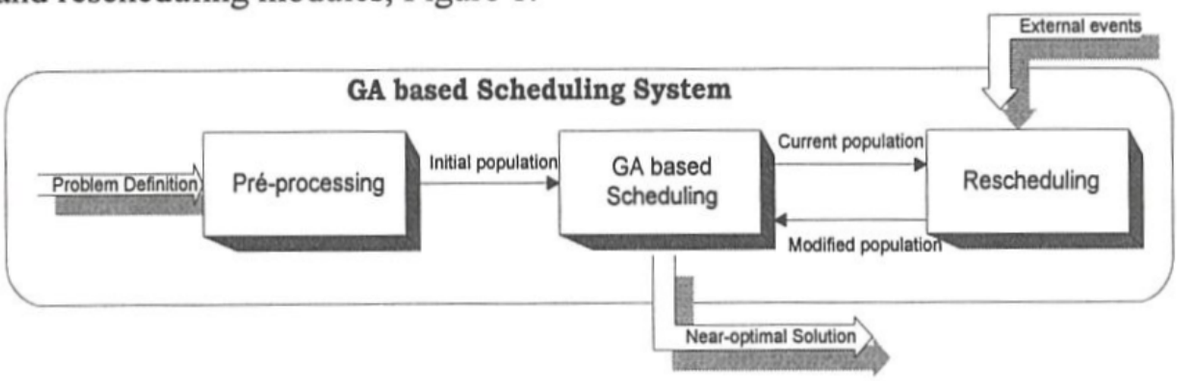

Figure 1 – Dynamic Scheduling System based on GA

Pre-processing module

The pre-processing module deals with processing input information, namely problem definition and instantiation of algorithm components and parameters. With the input information the pre-processing module generates the initial individual and the initial population, following the procedures described in section 3.2.

Scheduling module

The scheduling module is concerned with the application of the genetic algorithm for finding a near-optimal solution to a deterministic problem, where all job attributes are known in advance.

The algorithm described in section 3 and its MetaGA version, can be applied after some adaptations to take into account job release times. One of these has to do with the fitness function.

Thus, whenever new events occur, deterministic problems are generated by the rescheduling module and then solved by the scheduling module.

Rescheduling module

Rescheduling due to new events can be based on two approaches. The first discards the current population and generates a new population from the beginning whenever a new event, changing the problem conditions, occurs. This is achieved by restarting the scheduling module with the new problem. In real scheduling problems, continually subject to several changes over time, on occasions this approach may not perform well, because it implies many successive restarts, lost of inherit genetic characteristics from current population of individuals and is likely to be time inefficient.

With the other approach, a new population is created through modification of the existing one to take into account changes that have occurred on individuals. This modification of the current population is carried out between successive random events. This means modifying the current individuals and population size taking into account problem changes due to external events, allowing the schedule module to continue the search process based on the new modified or regenerated population. This procedure, leads to inheritance of good characteristics of schedules and is likely to be more efficient.

When the random event is a new job arrival then it must be inserted into the sequence of existing jobs. To carry this out two procedures can be implemented:

- Randomly select one position to insert the new gene into the chromosome, i.e. randomly schedule a new job among the existing jobs;
- Use some intelligent mechanism to insert the new job into the sequence.

When a job is cancelled, the correspondent gene has to be deleted from every chromosome.

After the insertion or deletion of genes is carried out, population updating is done by updating the size of the population and ensuring a structure identical to the existing one. Thus, either some further chromosomes have to be generated through some procedure, i.e. REED rule, or some existing chromosomes have to be deleted. In this case either the choice could be random or fall on the less adapted chromosomes. After this is done, the scheduling module can apply the search process with the new updated population.

6. CONCLUSIONS

In this paper two interrelated genetic algorithms for the minimisation of the static weighted tardiness SMSP are proposed. One is a single start GA, the other, a multi-start version GA, called MetaGA. The obtained results show that these GA perform well for the cases studied, being possible to find good solutions in a short time, i.e. a few minutes of CPU time. Substantial performance improvements with the MetaGA were obtained in relation to single start GA.

We also propose a scheduling system based on genetic algorithms to solve the dynamic version of the problem. A population regenerating mechanism is put forward. This adapts the population to a new population, which increases or decreases according to new job arrivals or cancellations.

In studying the dynamic problem we are particularly concerned with evaluating the adequacy of the GA approach to real world scheduling problems. Thus, our work is being further developed to consider other relevant objective functions. Having in mind that finishing a job early, i.e. when tardiness is zero, does not necessarily means good performance, one such function, which seems important, is the accuracy of achieving due dates.

These developments will permit to construct multi-criteria analysis procedures that may be used in Decision Support Systems for scheduling.

7. REFERENCES

1. Abdul-Razaq, T.S., C.N. Potts and L.N. Van Wassenhove, *A survey for the Single-Machine Scheduling Total WT Scheduling Problem*, Discrete Applied Mathematics, nº26, pp. 235-253, 1990
2. Adams, J., Balas, E. and Zawack, D., *The Shifting Bottleneck Procedure for Job Shop Scheduling*, Management Science, vol. 34, nº3, pp. 391-401, 1988
3. Bagchi, Sugato et al., *Exploring Problem-Specific Recombination Operators for Job-Shop Scheduling*, Proc. Fourth Int. Conf. On Genetic Algorithms, Morgan Kaufman, pp. 10-17, 1991.
4. Baker, K.R, *Introduction to Sequencing and Scheduling*, Wiley, New York., 1974.
5. Brownie, J., Harhen, J., Shivnan, J., *Production Management Systems*, Adddison-Wesley Publishing Co, 1988
6. Camarinha-Matos, L. M , Afsrmanesh, A., *The Virtual Enterprise Concept*, In IFIC TC5/PRODNET Working Conf. Infraestructures for Virtual Enterprises, Kluwer Academic Publishers, 1999
7. Crau□ells, H.A.J., C.N.Potts and L.N.V. Wassenhove, *Local Search Heuristics for the single machine total WT scheduling problem*, Informs Journal on Computing, vol.10, nº3, pp. 341-350, 1998.
8. Davis, Lawrence, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.
9. French, S., *Sequencing and Scheduling-An Introduction to the Mathematics of Job-Shop*, John Wiley and Sons, 1982
10. Goldtratt, E., Fox, R., *The Race*, North River Press, 1986
11. Holland, J.H., *Adaptation in Natural and Artificial Systems*, The MIT Press, 1992
12. Lawer, E.L., *A pseudopolinomial algorithm for Sequencing Jobs to Minimize Total*

Tardiness, Annals of Discrete Mathematics, pp. 331-342, 1977.
13. Madureira, Ana M., *Meta-heuristics for the Single-Machine Scheduling Total Weighted Tardiness Problem*, International Symposium on Assembly and Task Planning (ISATP'99), Portugal, 1999.
14. Michalewicz, Zbigniew, *Genetic Algorithms + Data Structures = Evolution Programs*, Third Revised and Extended Edition, Springer - Verlag Berlin Heidelberg, NewYork, 1992.
15. Morton, E. Thomas and David W. Pentico, *Heuristic Scheduling Systems*, John Wiley & Sons, 1993.
16. Potts, C.N. and L.N. Van Wassenhove, *Single Machine Tardiness Sequencing Heuristics*, IIE Transactions, vol. 23, nº 4, pp. 346-354, 1991
17. Rinnooy Kan, A.H.G., B.J. Lageweg and J.K. Lenstra, *Minimising Total Costs in One-Machine Scheduling*, Operations Research, Vol 23, nº 5, pp. 908-927, 1975.
18. Website1, http://mscmga.ms.ic.ac.uk/jeb/orlib