Available online at www.sciencedirect.com
SCIENCE DIRECT®
Applied Mathematical Modelling 30 (2006) 147–154
APPLIED MATHEMATICAL MODELLING
www.elsevier.com/locate/apm

Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance

A. Sadegheih

Department of Industrial Engineering, University of Yazd, P.O. Box 89195-741, Yazd, Iran

Received 1 April 2003; received in revised form 1 February 2005; accepted 15 March 2005
Available online 17 May 2005

Abstract

Genetics algorithms have been designed as general purpose optimisation methods. Simulated annealing simulates the cooling process of solid materials-known as annealing. However this analogy is limited to the physical movement of the molecules without involving complex thermodynamic systems. Physical annealing refers to the process of cooling a solid material so that it reaches a low energy state. Initially the solid is heated up to the melting point. Then it is cooled very slowly, allowing it is to come to thermal equilibrium at each temperature. This process of slow cooling is called annealing. The goal is to find the best arrangement of molecules that minimises the energy of the system, which is referred to as the ground state of the solid material. If the cooling process is fast, the solid will not attain the ground state, but a locally optimal structure. In this paper presents a general purpose schedule optimiser for manufacturing shop scheduling using genetic algorithms and simulated annealing. Then, the 'uniform order-based' crossover and mutation operators and novel general effects of parameter values on minimised objective value are presented.

Keywords: Job shop scheduling; Genetic algorithms; Flow shop scheduling; Heuristic algorithms; Branch and bound programming; Dynamic programming; Simulated annealing

E-mail address: sadegheih@yazduni.ac.ir

0307-904X/$ - see front matter
doi:10.1016/j.apm.2005.03.017

1. Introduction

Throughout the second half of this century there has been much research into algorithms that aim to optimise production schedules for "job shop" and "flow line" type environments. This paper continues to maintain momentum because the perfect solution has not been found yet. The desire to seek this solution is driven by the high dependence of a manufacturer's performance and profitability on schedule optimisation.

Since the late 1980s there has been a growing interest in genetic algorithms—optimisation algorithms based on principles of natural evolution. As they are based on evolutionary learning they come under the heading of Artificial Intelligence. They have been used widely for parameter optimisation, classification and learning in a wide range of applications. More recently, production scheduling has emerged as an application. Whilst it is well documented how GAs can be applied to simple job sequencing problems, this paper shows how they can be implemented to sequence job/operations for manufacturing shops with precedence constraints among manufacturing tasks. The methods presented are very easy to implement and therefore support the conjecture that GAs can provide a highly flexible and user-friendly solution to the general shop scheduling problem. Being suitable for large search spaces is a useful advantage when dealing with schedules of increasing size since the solution space will grow very rapidly, especially when this is compounded by such features as alternative machines. It is important that these large search spaces are traversed as rapidly as possible to enable the practical and useful implementation of automated schedule optimisation. If the optimisation is done quickly then production managers can try out 'what-if' scenarios and detailed sensitivity analysis as well as being able to react to 'crises' as soon as possible. For a simple n jobs and m machines schedule the upper bound on the number of solutions is $(n!)^m$. Traditional approaches to schedule optimisation such as mathematical programming and branch and bound are computationally very slow in such a massive search space [1–7].

2. Data entered into the model

A model of a schedule can have the following inputs: (i) job and operation identification; (ii) machines required by operations; (iii) processing time of each operation; (iv) precedence constraints; (v) job weights in respect of weighted objective function.

The units of time (minutes, hours, days etc.) used in the data are incidental to optimising the schedule, provided the same units are used throughout.

The assumptions made in the examples presented in paper are: (i) there is no randomness, all data is deterministic; (ii) there is only one machines of each type; (iii) no pre-emption of a job; (iv) no cancellation of a job; (v) the set-up times for the operations are independent of operation sequence and are included in the processing times; (vi) after a job is processed on a machine it is transported to the next machine immediately and the transportation time is negligible; (vii) all jobs are simultaneously available at time zero.

3. Single machine scheduling

The single machine example is taken from Kunnathur and Gupta [8]. They consider the minimisation of makespan in single machine environment where the processing time of a job consists of a fixed and a variable part. The variable part depends on the start time of the job. Hence the processing time of job i is given by

$$y_i = p_i + b_i, \tag{1}$$

where p_i is the fixed part and b_i is the variable part. b_i is the penalty for job i. Its value depends on s_i the start time of the job. b_i is given by

$$b_i = \max\{0, v_i(s_i - \mathrm{LST}_i)\}, \tag{2}$$

where LST_i is the latest desired start time of job i. If job i dose not start by LST_i then its processing time increases at the rate of v_i per unit of time that s_i exceeds LST_i, $v_i > 0$.

The objective is the minimisation of makespan.

4. The sequence optimisation genetic algorithm

First, an initial population of randomly generated sequences of the tasks in the schedule is created. These individual schedules form chromosomes which are to be subjected to a form of Darwinian evolution. The size of the population is user-defined and the fitness of each individual schedule in the population is calculated according to a user defined fitness function such as: total makespan, mean tardiness, maximum tardiness, number of tardy job. The schedules are then ranked according to the value of their fitness function. Once an initial population has been formed, selection, crossover and mutation operations are repeatedly performed until the fittest member of the evolving population converges to an optimal fitness value. Alternatively, the GA may run for a user defined number of iterations. The selection process consists of selecting a pair of parents from the current population using a rank based mechanism to control the probability of selection. These parents then mate to produce a child by applying the uniform order based crossover operator. This is based on using a randomly generated bit string template to determine for each parent which genes are carried forward into their child. It is defined as follows:

Given parent-1 and parent-2, create child in this way:

a. Generate bit string that is the same length as the parents.
b. Fill in some of the positions on the child by copying them from parent-1 wherever the bit string contains "1".
c. Make a list of the genes from parent-1 associated with a "0" in the bit string.
d. Permute these genes so that they appear in the same order they appear in parent-2.
e. Fill these permuted genes in the gaps in child in the order generated in iv.

The operator is explained by the example in Table 1. Here, the bit string template contains a one at positions $\{2, 3, 5, 6\}$. The genes in these positions in parent-1 are carried straight across into

Table 1
Schedule chromosomes and the crossover operator

Position in chromosome	1	2	3	4	5	6	7	8
Parent-1	1	2	3	4	5	6	7	8
Binary template	0	1	1	0	1	1	0	0
Parent-2	8	6	4	2	7	5	3	1
Child	8	2	3	4	5	6	7	1

Table 2
Order-based mutation

Parent-1	1	2	3	4	5	6	7	8
Selected		*				*		
Mutated chromosome	1	6	3	4	5	2	7	8

the child. The template contains zeros at positions $\{1,4,7,8\}$. In these position parent-1 has genes $\{1,4,7,8\}$. These genes are copied into the child in the same order as they appear in parent-2 $\{8,4,7,1\}$. The proportion of the jobs in the child coming from the first parent is defined by the crossover rate which has a between 0 and 1. Following crossover, the resultant child may be mutated.

The particular mutation operation used here is order-based mutation. In this operation two jobs change positions in the chromosome. The probability that a job is mutated is defined by the user defined mutation rate which lies in the range 0–1. The purpose of mutation is to ensure that diversity is maintained within the population. It gives random movement about the search space thus preventing the GA becoming trapped in blind corners or local optima. Finally, the new child schedule replaces the weakest schedule in the current pool or population of schedules. In this type of mutation two genes are randomly selected and their positions are interchanged as shown in Table 2.

The order-based crossover and mutation generally work better than one and two-point. They are more effective at combining schemata than either one or two point crossover. Empirically, order-based crossover and mutation are shown to be more effective on a variety of function optimisation problems. In this paper one child is formed by taking a mixture of bits from its two parents according to a random bit string. The proportion of bits coming from the best parent is defined by the user-defined crossover rate in the range zero to one.

5. Job data for Kunnathur and Gupta's

The data given in Table 3 is the job data in the spreadsheet model. The scheduler/evaluator portion of the model is constructed by using the spreadsheet's build in functions.

Kunnathur and Gupta proposed two optimising algorithms, one based on dynamic programming and the other on branch and bound. Both methods found an optimal makespan of 103.91 with a job sequence of 3–6–5–2–4–1. The GA found the same solution.

Table 3
Job data for Kunnathur and Gupta's example

Job	Process time	LST_i	v_i
1	20	38	0.44
2	16	38	0.61
3	14	7	0.74
4	3	68	0.25
5	13	34	0.86
6	19	33	0.77

6. Flow shop scheduling using genetic algorithm

Table 4 gives the job data for this example and the objective is to minimise the makespan for the schedule. When the GA is applied to this the best sequence is 3–6–4–7–2–8–1–5 which yields a makespan of 584 units of time. The result of applying the network scheduling technique to this example was reported and the best sequence found was 3–6–4–7–8–2–1–5 which produces a makespan of 595 units of time [9]. This was also the best result reported when using a heuristic method [10].

The normal sequence optimising GA will rearrange freely the jobs in a schedule. However, in most production scheduling situations there will be precedence constraints on the order of operations within a particular job. For example, holes must be drilled before they are tapped. When the GA creates a child the precedence constraints is checked. If any precedence constraints are broken then tasks which must be performed earlier in the sequence are moved up the schedule.

7. Flow shop scheduling using simulated annealing

Simulated annealing is an intelligent approach designed to give a good though not necessarily optimal solution, within a reasonable computation time. The motivation for simulated annealing comes from an analogy between the physical annealing of solid materials and optimisation problem. The analogy between physical annealing and simulated annealing can be summarised

Table 4
Job data for 8-job, 7-machine flow line and processing times for machine

Job list	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5	Machine 6	Machine 7
1	13	79	23	71	60	27	2
2	31	13	14	94	60	61	57
3	17	1	–	23	36	8	86
4	19	28	10	4	58	73	40
5	94	75	–	58	–	68	46
6	8	24	3	32	4	94	89
7	10	57	13	1	92	75	29
8	80	17	38	40	66	25	88

Table 5
Makespan and iterations

Cooling rate	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.95
Makespan	595	590	589	584	584	584	596	595	587
Iterations	370	430	550	700	750	650	1000	1000	1000

as follows: (i) The physical configurations or states of the molecules correspond to optimisation solution; (ii) the energy of molecules corresponds to the objective function or cost function; (iii) a low energy sate corresponds to an optimal solution; (iv) the cooling rate corresponds to the control parameter which will affect the acceptance probability. The algorithm consists of four main components: (i) configurations; (ii) re-configuration technique; (iii) cost function and (iv) cooling schedule [11,12]. Similarly to simulated annealing, genetic algorithms are stochastic search methods, and they aim to find an acceptable solution where it is impractical to find the best one with other techniques.

The simulated annealing algorithm was employed to solve the problems of combinatorial optimisation of flow shop scheduling problem. The same 8-bit binary string representation scheme applied in the case of the GA was implemented for simulate annealing because of its flexibility and ease of computation. The objective is to minimise the makespan for the schedule. The annealing process stared at a high temperature, $T = 1500$ units, so most of the moves were accepted. The cooling schedule was represented by

$$\boldsymbol{T}_{t+1} = \mu \boldsymbol{T}_t, \tag{3}$$

where μ is the cooling rate parameter, which was determined experimentally.

The algorithm was implemented in Turbo C++. The initial stopping criterion was set at a makespan of optimal solution found by the GA. Experiments were conducted again with a lowered stopping criterion. However no improvement was found even after 15 hours computation time. Nine cooling rates were used (0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.95). The final cost function is showed in Table 5 that the maximum number of iterations was set at 1000.

Therefore the solution found by the GA was accepted as the optimal solution the final flow shop scheduling is shown in Table 5. Overall simulated annealing needed longer computation times compared to the genetic algorithm.

8. Effects of mutation and crossover rate on minimised objective value

The application of the GA to the schedule in example above is repeated here for each of the combinations of the following parameter level: population size: 20, 60, 100; mutation rate: 0.001, 0.005, 0.01, 0.02 and crossover rate: 0.10, 0.20, 0.60, 0.8, 0.90. For each combination of the parameters the GA is run for 15 different random initial populations. These 15 populations are different for each combination. Thus, in total, the GA is run 900 times. The values chosen for the population size are representative of the range of values typically seen in the literature [1–7], with 0.02 being included to highlight the effect of a relatively large mutation rate. The crossover rates chosen are representative of the entire range. The optimal solution known in priori for this schedule is a makespan of 584 units of time. Fig. 1 shows that a general trend in performance

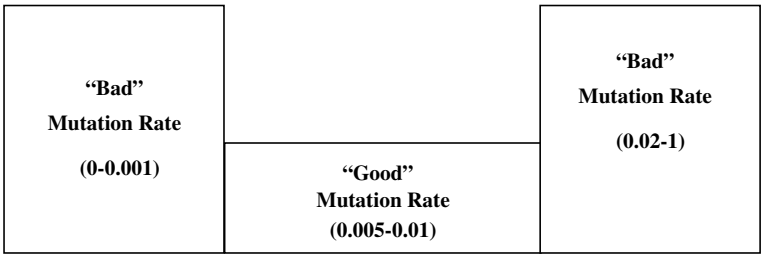

Fig. 1. Effect of mutation rate on objective function value.

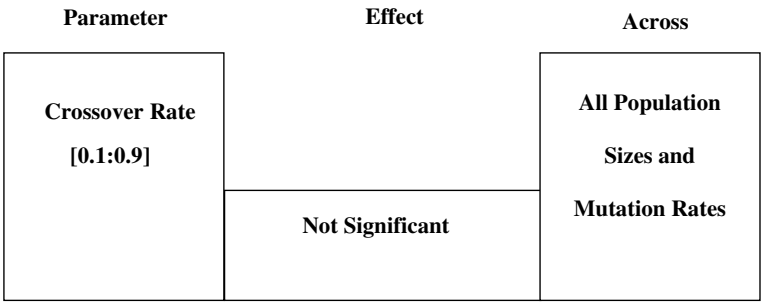

Fig. 2. Effect of crossover rate on objective function.

of mutation rate. That is, it forms a curve of the form illustrated in Fig. 1 which shows that performance degrades either side of a range of 'good' mutation values. This relatively flat bottomed curve gives a degree of tolerance to the choice of mutation rate.

Fig. 2 shows that crossover rate across the rage tested [0.1:0.9] does not have a significant effect on the speed of convergence of GA when working with any of the mutation rates. It shows the same insignificant result across the range of population sizes.

9. Conclusions

The use of genetic algorithms to optimise production schedules has been demonstrated. The key advantage of GA's portrayed here is that they provide a general purpose solution to the scheduling problem, with the peculiarities of any particular scenario being accounted for in fitness function without disturbing the logic of the standard optimisation routine. The GA can be combined with a rule set to eliminate undesirable schedules by capturing the expertise of the human scheduler. The results of the experiment have confirmed that the cooling rate determines the quality of the solutions. If the cooling rate is too low, the configuration can not achieve the optimal solution before it reaches the maximum number of iterations. If the cooling rate is too high, the process could become stuck at a local optimum. Overall simulated annealing needed longer computation times compared to the genetic algorithm. Finally, the performance is insensitive to mutation rate within a good range of values, however, outside the good range the GA performance deteriorates.

In crossover rate, the performance is insensitive to crossover rate in range [0.1:0.9] but operator is necessary.

References

[1] A. Sadegheih, Optimization in spreadsheet model for network problem, Far East J. Math. Sci. 4 (1) (2002) 100–115.
[2] A. Sadegheih, P.R. Drake, Network planning using iterative improvement and heuristic method, Int. J. Eng. 15 (1) (2002) 63–74.
[3] A. Sadegheih, in: C.E. D'Attellis, V.V. Kluev, N. Mastorakis (Eds.), Mathematics and Simulation with Biological, Economical and Musicoacoustical Applications, World Scientific and Engineering Society press, 2001, pp. 11–16.
[4] A. Sadegheih, P.R. Drake, Network optimization using linear programming and genetic algorithm, Neural Network World, Int. J. Non-Stand. Comput. Artif. Intell. 11 (3) (2001) 223–233.
[5] A. Sadegheih, P.R. Drake, in: N. Mastorakis (Ed.), Recent Advances in Applied and Theoretical Mathematics, World Scientific and Engineering Society press, 2000, pp. 150–155.
[6] A. Sadegheih, P.R. Drake, in: N. Mastorakis, V. Mladenov, B. Suter, L.J. Wang (Eds.), Advances in Scientific Computing Computational Intelligence and Applications, World Scientific and Engineering Society press, 2001, pp. 215–221.
[7] A. Sadegheih, Iterative improvement method and mixed-integer programming in system planning, 16th International Power System Conference, 22–24 October 2001, Tehran, Iran.
[8] A.S. Kunnathur, S.K. Gupta, Minimising the makespan with late start penalties added to processing times in a single facility scheduling problem, Eur. J. Oper. Res. 47 (1990) 56–64.
[9] E.O.P. Akpan, Job-shop sequencing problems via network scheduling technique, Int. J. Oper. Prod. Manage. 16 (1996) 76–86.
[10] H.G. Campbell, M.L. Smith, R.A. Dudek, A heuristic algorithm for the N jobs, machine sequencing problem, Manage. Sci. 16 (1970) 630–637.
[11] P.J.M. van Laarhoven, E.H.L. Aarts, Simulated Annealing: Theory and Applications, Reidel, Dordrecht, Holland, 1987.
[12] P. Tian, Z. Yang, An improved simulated annealing algorithm with genetic characteristics and travelling salesman problem, J. Inform. Optim. Sci. 14 (3) (1993) 241–255.