Genetic Algorithms and Machine Scheduling With Class Setups

Nasir Hayat
Department of Mechanical Engineering
University of Engineering and Technology Lahore, Pakistan

Dr. Andrew Wirth
Department of Mechanical and Manufacturing Engineering
University of Melbourne, Australia

Abstract

This paper examines the application of genetic algorithms to a scheduling problem in which n jobs are to be scheduled on a single and parallel identical machines with an objective to minimize the total flow time. The jobs are divided into mutually exclusive classes and a setup task is required when processing switches from a job of one class to a job of another class. The setup times are assumed to be sequence independent.

The range of applications of genetic algorithms has grown rapidly in the recent years and genetic algorithms are being used in industrial and financial fields. For a small and medium scale machine shop, it is difficult to hire experts to design and build customized software. Keeping in view this application oriented aspect, it was decided to use the Evolver software package, an addition to the Excel spreadsheet to implement the genetic algorithm.

Keywords: Genetic Algorithms, Machine Scheduling, Identical Machines, Class Setups

Introduction:

Practical scheduling problems often involve processing several classes of related jobs on common facilities where a setup time is incurred whenever there is a switch of processing from a job in one class to a job of another class. If the setup times are sequence dependent, the problem is N-P-complete. Kanet and Sridhar (1991) and Rubin and Ragatz (1995) have applied genetic algorithm to problems with sequence dependent setup times for single machines. Kanct and Sridhar implemented a simple problem with 16 jobs and in their conclusion they reported they would extend their research to 2,3 and 4 non-identical parallel machines. Robin and Ragatz implemented a set of problems with 15, 30 and 45 jobs for minimizing total tardiness as the criterion. Usually sequence independent setup times problem is considered to be very simple (Rubin and Ragatz, 1995) as the setup may be added to the processing time, but this may not be true when jobs are divided into mutually exclusive classes with setup required when we change from one class to another and the splitting of classes can take place. Various scheduling problems in manufacturing and service organizations can be formulated as single facility problems with job classes (Mason and Anderson, 1991). For example, the scheduling of jobs for a flexible manufacturing system can produce several different types of product on the same machine but requires a setup to rearrange or retool work stations when there is a switch in product type. Other examples are the scheduling of source code programs which are to be executed by a computer using the different compilers and the allocation of a worker who must switch from machine to machine. Mason and Anderson (1991) obtained necessary conditions for an optimum solution. They then use these to prune the search tree. Williams and Wirth (1996) generated a heuristic which satisfied the Mason and Anderson conditions and appeared to perform well when compared with some standard heuristics. Williams

(1993) has extended this work to paral lel machines.

Recently a number of successful attempts have been made to demonstrate the applicability of genetic algorithms to machine scheduling problems. Some examples are given below:

Problem Type	**Reference**
JSSP	Bagchi et al., 1991
JSSP	Biegel amd Davern, 1990
JSSP	Cleveland and Smith, 1989
JSSP	Davis, 1985
JSSP	Dorndorf and Pesch, 1995
JSSP	Falkenauer and Bouffouix , 1991
JSSP	Fang et al., 1993
JSSP	Federico et al., 1995
JSSP	Fox and Mamahon, 1990
JSSP	Nakano and Yamade , 1991
JSSP	Syswerda , 1991
JSSP	Whitely et al ., 1989
FSSP	Chen et al., 1995
FSSP	Mulkens, 1994
FSSP	Reeves, 1995
Single and Multiple Machines	Bean , 1994
Unrelated Parallet Machines	Glass et al., 1994
Single Machine	Gupta et al., 1993
Single Machine	Kanet and Sridharn, 1991
Single Machine	Lee and Choi , 1995
Single Machine	Rubin and Ragatz, 1995
Family and Job Scheduling in a flowine based Manufacturing Cell	Sridhar and Rajendran, 1994

Apart from this list much work has been done to design crossover operators which would yield valid offspring. Some examples are as follows:

Crossover Operator	**Reference**
Order Crossover	Davis, 1985
Union Crossover	Fox and McMahon, 1991
Intersection Crossover	Fox and McMahon, 1991
Partially Matched Crossover	Goldberg, 1989
Cycle Crossover	Oliver et al.. 1987
Tie Breaking Crossover #1	Poon and Carter, 1995
Tie Breaking Crossover #2	Poon and Carter, 1995
Order Crossover #2	Syswerda, 1991
Position based Crossover	Syswerda, 1991

Enhanced Edge Recombination	Starkweather et al., 1991
Edge Recombination	Whitley et al., 1989

Some Characteristics of EVOLVER:

Evolver uses a steady state approach. This means that only one organism rather than an entire population is replaced at a time. The number of generations can be found by dividing the trials by the size of the population. The following parameters must be set:

Population Size:

The population size specifies how many organisms should be stored in memory at any given time.

Selection:

In Evolver, parents are chosen using a rank-based mechanism. This procedure begins by rank ordering the population by fitness. Next an assignment function gives each individual a probability of inclusion. The assignment function can be linear or nonlinear. A roulette wheel is then built with the slots determined by the assignment function. The next generation of an n-sized population is built by giving the wheel n spins. This procedure guides selection towards the better performing members of the population but does not force any particular individuals into the next generation.

Crossover:

We used the order solving option available in Evolver. It applies the order crossover operator (Davis, 19,85). Order Crossover (OX) creates children which preserve the order and the position of alleles in a subsequence of one parent while preserving the relative order of the remaining alleles from the other parent. It is implemented by selecting two random cross points which define the boundaries for a series of copying operations. Alleles from the first parent that fall between the two crosspoint are copied into the same position of the offspring. The remaining allele order is determined by the second parent. Non duplicative alleles are copied from the second parent to the offspring beginning at the position following the second crosspoint. Both the second parent and the offspring are traversed circularly from that point. A copy of the parent's next nonduplicative allele is placed in the next available child position. An example for which the random crosspoints are 3 and 6 is as follows:

Parent1: **A B C G H B D E**
Parent2: **H D E F G C A D**
Child: **E F C G H B A D**

Crossover rate can be varied from 0-1.

Miltation:

The order solving method referred to above performs mutations by swapping the positions of some variables in the organism. The number of swaps performed is increased or decreased proportionately to the increase and decrease in the mutation rate setting (from 0 to 1).

Replacement:

The worst performing organism is replaced with the new organism that is created by selection, crossover and mutation.

Some Results:

The Excel spreadsheet was used to make a model of the problem under study and Evolver was then used to find the solution to the model. In the absence of any standard benchmark problems for this type of scheduling problems (especially when the optimal solution is not known), it was decided to compare the performance of GAs with the heuristics developed by Williams (1993), Williams and Wirth (1996) and Dunstall. We compared the performance of the genetic algorithms with the then best available heuristics and lower bounds.

Test problems such as the ones in the table below, were generated with setup times and processing times (both integers) selected from uniform distributions [0, 1001 and [1, 100] respectively. The number of jobs in each class was also randomly selected with the restriction that N jobs were required in total. The results of the implementation of two sets of problems, one for single machines and other for two identical parallel machines with five classes and upto 80 jobs are shown below:

Job-Classes-Machines	Lower Bound	Best Heuristic Solution	GA Solution	Best	Daviation of GA From Best Heuristic (%)
20- 5- 1	10833	12142	12142		0.000
30- 5- 1	18880	20988	20988		0.000
40- 5- 1	35547	42313	40773	GA	-3.639
60- 5- 1	67678	77183	76456	GA	-0.942
80- 5- 1	120769	145019	139368	GA	-3.897
20- 5- 2	4750	5438	5438		0.000
30- 5- 2	10539	11314	11296	GA	-0.000016
40- 5- 2	20326	23665	22543	GA	-4.741
50- 5- 2	25937	29662	29662		0.000
60- 5- 2	28215	32460	32460		0.000

We can similarly model and apply genetic algorithms to problems with more than two parallel machines.

Parameter Settings:

The most difficult and time consuming issue in the successful implementation of genetic algorithms is to find good parameter settings. Here we point out the trade of'fs that arise:

- Increasing the crossover probability increases the recombination of building,, blocks, but it also increases the disruption of good strings.
- Increasing the mutation probability tends to transform the genetic search into a random search, but it also helps to reintroduce lost genetic material.
- Increasing the population size increases the diversity and reduces the probability that the genetic algorithm will prematurely converge to a local optimum, but it also increases the time required for the population to converge.

Two distinct parameter sets have emerged: one has a small population size and relatively large mutation and crossover probabilities (Grefenstette, 1986) while the other has a larger population size, but much smaller crossover and mutation probabilities (De Jong, as reported in Liepins and Hillard, 1989). Schafer et al. (1 989) have concluded that parameter settings vary from problem to problem.

Statistical Analysis:

Based on De Jong's and Grefenstette's work, we defined a particular genetic algorithm design by indicating the respective values of the parameters. For example: GAI (20, 30, 0.60, 0.0007) implies that in the design we consider a problem in which number of job is 20, the population size is 30, the crossover rate is 0.60 and the mutation rate is 0.0007.

Experimental Design Factor:

Parameter	Level
Population Size	30, 50, 60, 100, 150
Crossover Rate	0.6, 0.75
Mutation Rate	0.000, 0.0007, 0.0008, 0.0009, 0.001, 0.002, 0.005
Problem Size	20, 60

Analysis of Variance for % devtn

Source	DF	Seq SS	Adj SS	Adj MS	F	p
prob	1	2209.613	65.306	65.306	7.37	0.012
populatn	4	505.12	55.053	13.763	1.55	0.219
Xover	1	238.56	102.59	102.59	11.58	0.002
Mutatn	6	1369.148	956.117	159.353	17.99	0.000
Prob*populatn	4	39.727	127.766	31.941	3.61	0.019
prob@Xover	1	214.515	9.368	9.368	1.06	0.314
Prob*Mutatn	6	1531.775	797.787	132.965	15.01	0.000
populatn*Xover	4	88.020	70.579	17.645	1.99	0.128
populatn*Mutatn	24	1174.411	834.687	34.779	3.93	0.001
Xover*Mutatn	6	110.239	110.239	18.373	2.07	0.094
prob*populatn*Xover	4	55.706	55.706	13.926	1.57	0.214
prob*populatn*mutatn	24	843.098	843.098	35.129	3.97	0.001
Prob*Xover*Mutatn	6	97.165	97.165	16.194	1.83	0.136
populatn*Xover*Mutatn	24	287.491	287.491	11.979	1.35	0.233
Error	24	212.592	212.592	8.585		
Total	139	8977.18				

Findings of the ANOVA:

First we checked the adequacy of the model and then tested a hypothesis concerning the different levels of factors and their effect on the quality of the solution.

$$\text{F value for model} = \frac{\text{model MS}}{\text{error MS}}$$

$$= \frac{\frac{8765}{115}}{\frac{212.6}{24}} = 8.6$$

$$\text{Now } (F_{115,24}) = 2.97 \text{ so } P < 0.001$$

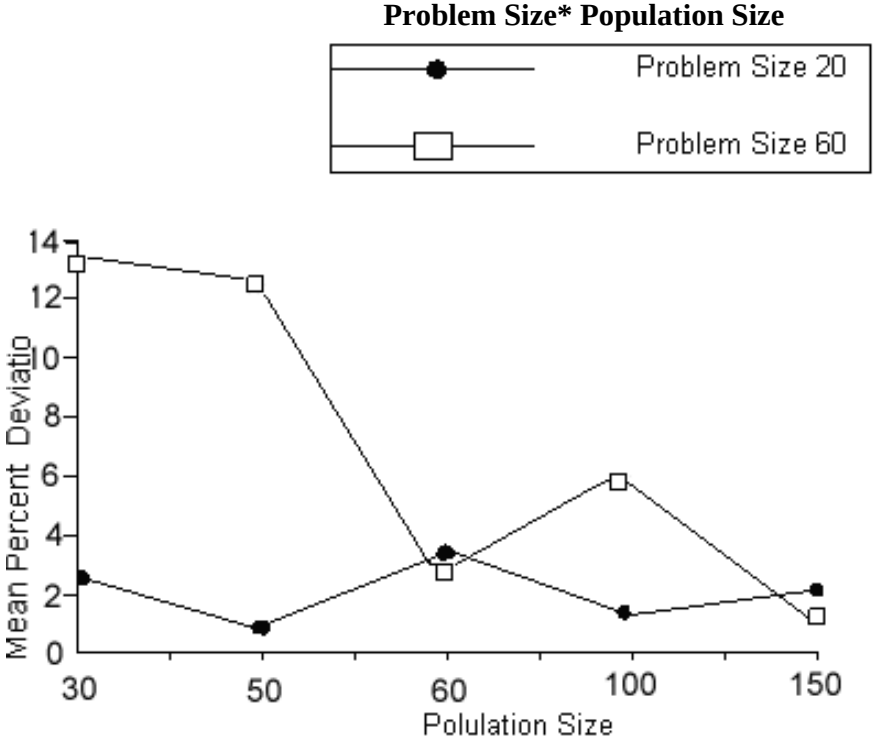
Therefore there is no reason to doubt adequacy of fit of the model. Thus it can be said that overall model is significant at a = 0.05 level.
Now we consider the
Null Hypothesis: Ho: There is no difference in the quality of solution for different levels of factors and
the Alternative Hypothesis: HI: There is a difference in the quality of solution for different levels of factors

An examination of the F-statistics and the corresponding P-values in the ANOVA table suggests that all the main factors except population size are significant at a = 0.05, therefore there is no reason to reject the alternative hypothesis. Higher F-statistics values for mutation rate, crossover rate and problem size than the F-statistics value for the population size indicate that variations in the level of the former factors have a greater impact on the quality of solution than the latter. This suggests that the use of larger populations is not always advantageous and it can increase the computation time and the time to converge. The differences within the different levels can be seen from the means for percentage deviations for the main factors. These are:

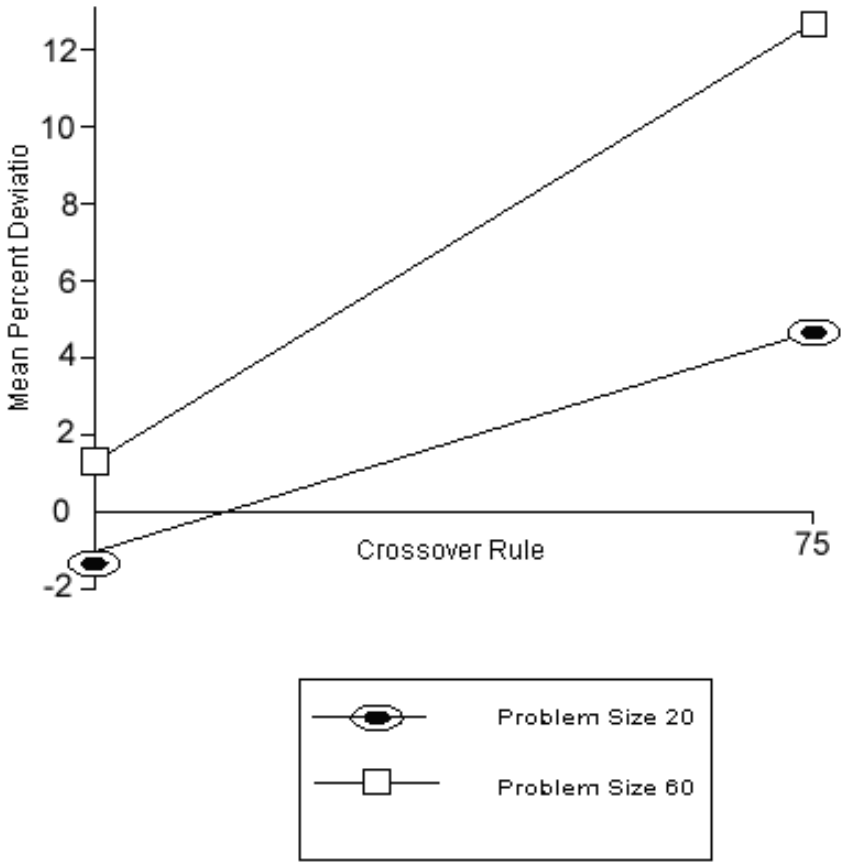
- Two levels of problem size are significantly different from each other and most of the time GA gave the better result for the smaller problem size.
- Although population size was found to be the least significant factor, there were differences within population size, at different levels and the mean for percent deviation was smallest for the highest population size. Here it is interesting to note that in their study of parameter settings Gupta et al. (1993) found that population size was the most significant factor in their algorithm.
- Means for percentage deviation for crossover rate indicate that the two crossover rates are significantly different from each other. The crossover rate of 0.6 seems to increase recombination of building blocks and at the same time is least likely to destroy good strings. Gupta et al. (1993) found crossover rate to be insignificant in their algorithm.
- Different mean percent deviations for different levels of mutation rates indicate that they are significantly different from each other and it is preferable to use a small rather than a zero mutation rate.

Now we will examine the **interaction effects.**

i. Although population size was less significant than the other main factors, there was a very significant interaction between problem size and population size The interaction plot indicates that at smaller populations the mean percent deviation for a larger problem size is greater than for a smaller problem size but at larger populations the mean percent deviation is almost the same for both problem sizes.

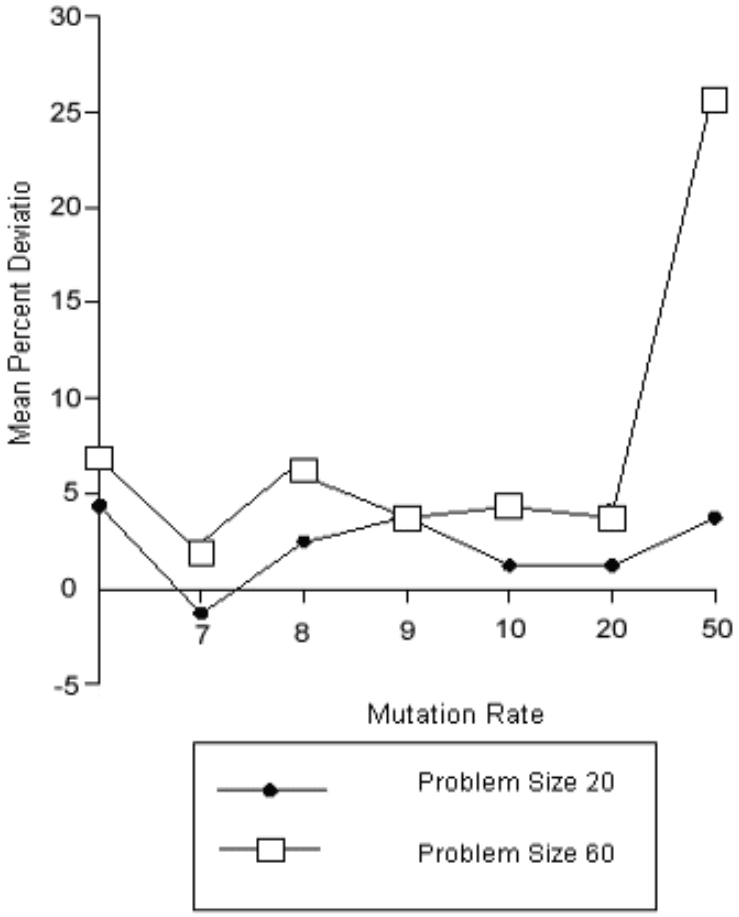


ii. The interaction between problem size and crossover rate is not significant, however the interaction plot indicates that the crossover rate 0.6 is good for both problem sizes. Both problem sizes follow almost the same pattern for the two crossover rates.



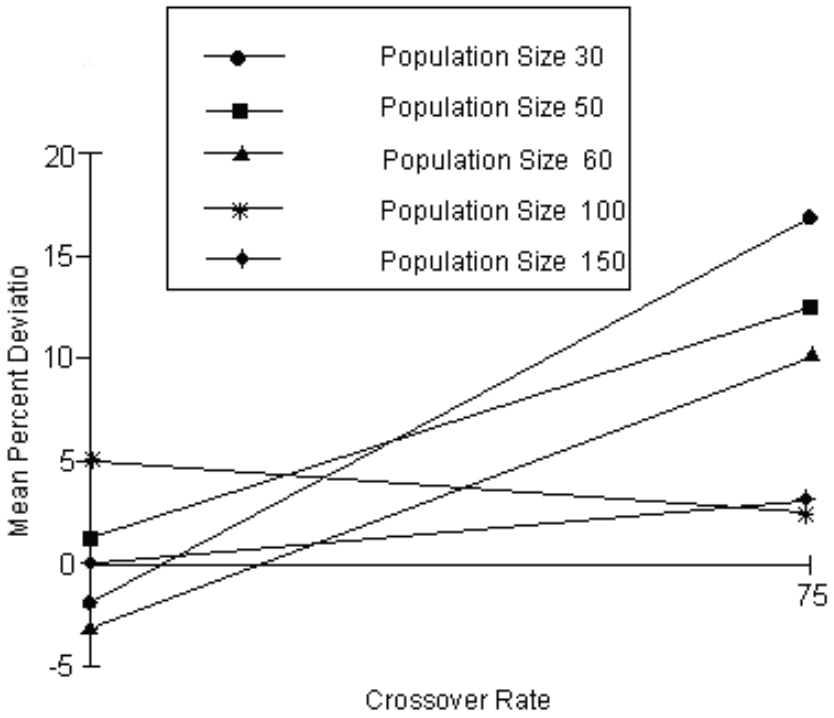
iii. There is an extremely significant interaction between problem size and the mutation rate. The interaction plot indicates that at lower mutation rates both problem sizes follow almost the same pattern and at the mutation rate of 0.0009 the behaviour for both problem sizes is the same. As the mutation rate increases the mean percent deviation for the larger problem size rises suddenly. So for a larger problem size the higher mutation rate is not suitable. Gupta et al. (1993) report a significant interaction between these two factors but in their case increasing the mutation rate gives good results.

Problem Size* Mutation Rate



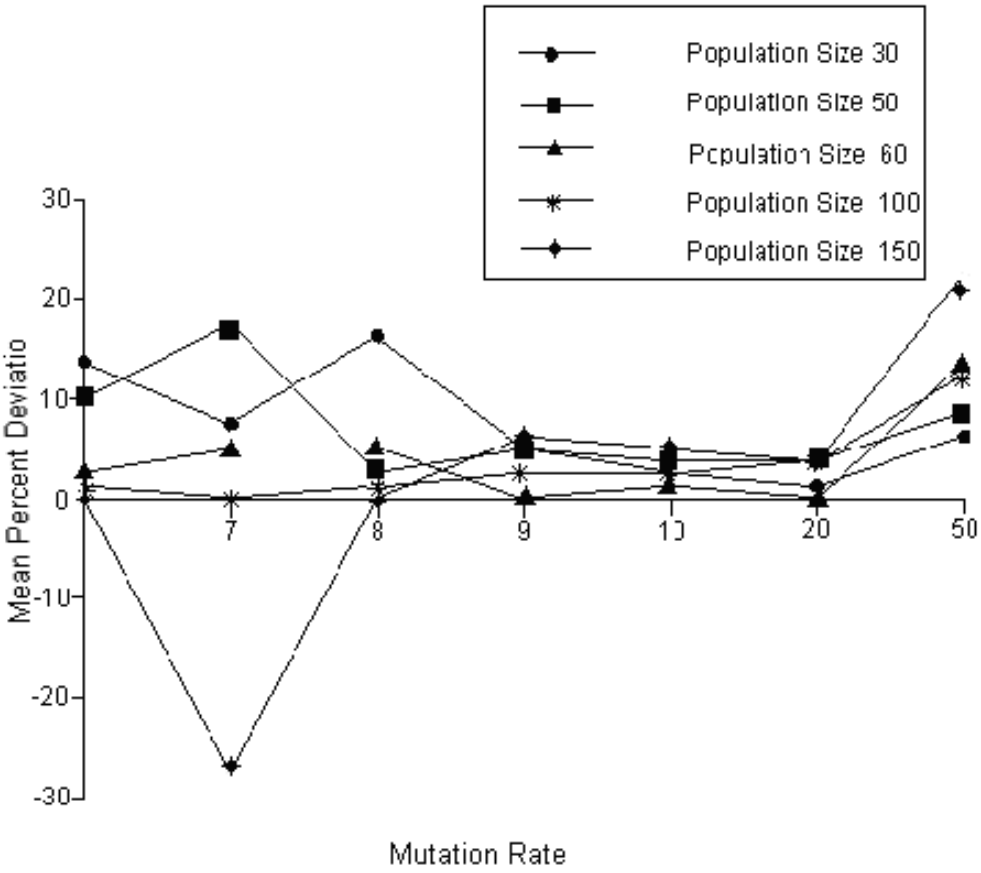
iv. The interaction between population size and crossover rate is not significant. The interaction plot indicates that for a lower population size the mean percent deviation increases with an increase in crossover rate. Hence for a low population size the disruptive effect of crossover dominates the recombination. With the population size of 100 the mean percent deviation decreased with an increase in crossover rate but increased for a population size of 150.

Population Size* Crossover Rate



v. There is a very significant interaction between population size and mutation rate The interaction plot indicates that the highest mutation rate has the same effect (increase in mean percent deviation) for all the population sizes but for a large population size, a lower mutation rate and for a smaller population size a higher mutation performs better. Thus there is some sort of inverse relationship between population size and mutation rate. These two can be traded off for a given level of the quality of the solution The interaction plot indicates that the mutation rate of 0.001 is equally good for all population sizes.

Population Size* Mutation Rate



vi. The interaction between crossover rate and mutation rate is not significant. The interaction plot indicates that there is an inverse relationship between mutation and crossover rates. At the lower mutation rate the lower crossover rate performs better up to a mutation rate of 0.02. However the higher crossover rate is good at the highest mutation rate. Again here our findings differ from Gupta et al. (1993) who found that the interaction between these two factors was significant.

vii. It is difficult to interpret the three factors interaction but it is clear from the ANOVA table that there is a significant interaction at the three factors level in the case of problem size, population size and mutation rate.

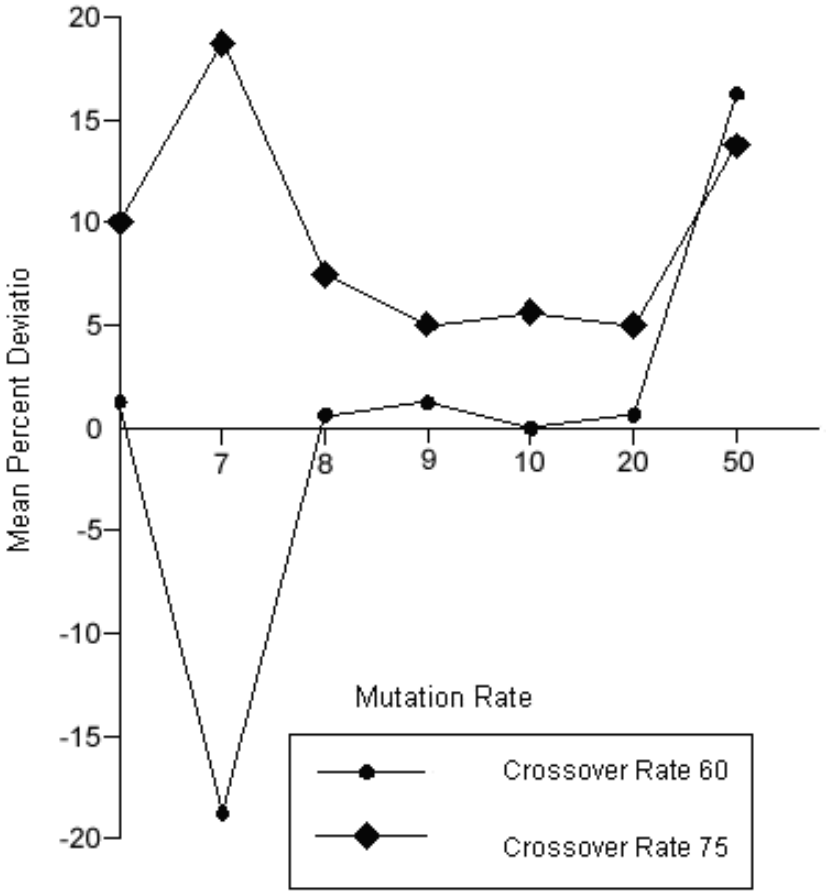
Conclusions:

In this paper the problem of scheduling identical parallel machines with sequence independent class setup times, to minimize the total flow time was approached using genetic algorithm. A literature review of GAs in the field of machine scheduling revealed that most of the early work was with reference to the traveling salesperson problem. Much of the earlier work was of a theoretical nature and concentrated on the genetic representation of the problem and the design of a suitable crossover operator which would yield legal schedules. Recently results of the implementation of GAs to different scheduling problems have been reported. This indicates that GAs are now mature enough for practical applications. The successful implementation of GA to the problem under study was demonstrated by addressing problems of different sizes and comparing the results with the then best available heuristics for this kind of problem. The results show that for sequence independent class setup times, better solution can generally be obtained by the splitting of classes.

Since the successful implementation of GAs depends upon the careful selection of parameter values, an analysis of variance test was carried out to investigate the interaction between different parameters. The results of the ANOVA show that the mutation rate, crossover rate and problem size

have significant impact on the quality of the solution while the population size was found to be insignificant. Furthermore, the interactions between these parameters especially between problem size and mutation rate and population size and mutation rate were also significant, which implied that a specific set of parameters should be selected carefully for the successful implementation of GAs.

Crossover rate* Multation Rate



Future Research:

Although GAs can give better results than other available heuristics for the problem under study, they proved to be slower when compared to other heuristics During the study we came across some important issues which could be the basis for future research. These are as follows:

- Incorporating problem specific knowledge

This can help in improving the speed of the genetic algorithm (Glass et al., 1994), As in Reeves (1995) an initial population can be seeded with a solution generated by a standard heuristic effect on the quality of solution if crossover and/or mutation rates are varied over time when the genetic algorithm is running.

The changes (increases or decreases) could be random or in equal incremental steps.

- Sensitivity analysis of different parameters so that parameters can be set quickly and accurately.

Parameter values giving good solutions for both problem sizes could be chosen and then a

sensitivity analysis could be performed to check the range within which these parameters could be varied while maintaining the level of solution quality.

References

Bagchi, S., Uckun, S, Miyabe, Y and Kawamura, K, 1991, "Exploring Problem Specific Recombination Operators for Job Shop Scheduling", In Belew, R K and Booker, L B (editors). Proceeding of 4th International Conference on Genetic Algorithms, Morgan Kaufmann, pp 10-17

Baker, K.R, 1974, Introduction to Sequencing and Scheduling, John Wiley & Sons, New York

Baufr, R J , 1994, Genetic Algorithms and Investment Strategies, Wiley, New York

Bean, J. C , 1994, "Genetic Algorithms and Random Keys for Sequencing and Optimization", ORSA Journal on Computing, Vol. 6, No, 2, pp, 154-160

Beasley, D, Heitkotter, J., 1994, The Hitch-Hiker's Guide to Evolutionary Computation, FAQ in Camp. ai genetic, Internet newsgroup,

Biegel, J E. and Davern, J. J., 1990, "Genetic Algorithms and Job shop Scheduling" Computers and Industrial Engineering , Vol. 19, Nos. 1-4, pp 81-91

Chen, C.L., Vempati, V. S. and Aljaber, N., 1995, "An Application of Genetic Algorithms for Flowshop Problem", European Journal of Operational Research, Vol. 80, pp 389-396

Conway, R. W., Maxwell, W. L. and Miller, L.W., 1967, Theory of scheduling, Addison-Wesley Publishing Co , Massachusetts.

Croce, F D., Tadei, R and Volta, G, 1995, "A Genetic Algorithm for the Job Shop Problem", Computers and Operation Research, Vol. 22, No. 1, pp. 15-24

Davis, L., 1985, "Job Shop Scheduling with Genetic Algorithms" In Grefenstette, J. J (editor), Proceeding of an International Conference on Genetic Algorithms and their Applications, Erlbaum, pp.136-140.

Davis, L, 1987, Genetic Algorithms and Simulated Annealing, Morgan Kaufmann, San Mateo.

Davis, L , 1989, "Adapting Operator Probabilities in Genetic Algorithms" In Schaffer, J.D. (editor), Proceedings of 3rd International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 61-67.

Davis, L , 1991, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York

Dorndorf, U and Pesch, E, 1995, "Evolution based learning in a Job Shop Scheduling Environment", Computers and Operations Research, Vol. 22, No.1, pp. 25-40.

Dunstall, S., 1996, Personal Communication,

Fang, H L., Ross, P. and Corne, D., 1993, "A Promising Genetic Algorithm Approach to Job Shop Scheduling, Rescheduling and Open Shop Scheduling Problems", In Schaffer, J. D (editor),

Proceedings of 3rd International Conference on Genetic Algorithms, pp. 375-382.

Falkenauer, E, and Bouffouix, S., 1991 "A genetic algorithm for job shop scheduling", Proceedings of 1991 IEEE International Conference on Robotics and Automation, pp. 824-829,

Federicio, D. C., Tadei, R and Volta, G., 1995, "A Genetic Algorithm for the Job Shop Problem", Computers and Operations Research, Vol. 22, No.1, pp. 15-24

Fox, B. R. and McMahon, M. B, 1991, "Genetic Operators for Sequencing Problems" In Rawlins, G. J. E. (editor), Proceedings of Fundamentals of genetic Algorithms, Morgan Kaufmann, pp. 284-300,

Glass, C A, Potts, C. N. and Shade, P., 1994, "Unrelated Parallel Machine Scheduling using Local Search", Mathematical and Computational Modelling, Vol. 20, No. 2, pp 41-52.

Goldberg, D. E.,1989, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Reading, Mass.

Grefenstette, J J , 1986, "Optimization of Control Parameters for Genetic Algorithms", IEEE Transactions on System, Man and Cybernetics, Vol, 16, No 1, pp. 122-128

Gupta, M. C., Gupta,Y. P. and Kumar, A, 1993, "Minimizing Flow Time Variance in a Single Machine using Genetic Algorithms", European Journal of Operational Research Vol. 70, pp. 289-303,

Kanet, J J. and Sridharn, V., 1991, "PROGENITOR: A Genetic Algorithm for Production Scheduling", Wirtschaftsinformatik, pp. 332-336 Lee, C, Y and Choi, J, Y., 1995, "A Genetic Algorithms for Job Sequencing Problems with Distinct Due Dates and General Early-Tardy Penalty Weights", Computers and Operations Research, Vol. 22, No. 8, pp. 857-869

Lee, C. Y., 1995, "Genetic Algorithms for Single Machine Job Scheduling with Common Due Date and Symmetric Penalties", Journal of Operations Research Sociaty of Japan, Vol. 37, No. 2, pp. 83-95

Liepins, G. E. and Hillard, M..R., 1989, "Genetic algorithms: Foundations and Applications", Annals of Operations Research, Vol. 21, pp 31-58,

Mason, A J. and Anderson, E. J., 1991, "Minimizing Flow Time on a Single Machine with Job Classes and Setup Times", Naval Research Logistics, Vol. 38, No. 7, pp. 333-350

Morikawa, K., Furuhashi, T. and UCHIKAWA, Y., 1993, "Single Populated Genetic Algorithm and its Application to Job Shop Scheduling", Transactions of the SICE, Vol. 27, No. 5, pp. 1014-1019.

Mulkens, H., 1994, "Revisiting the Johnson Algorithm for Flowshop Scheduling with Genetic Algorithms", Journals of Evolutionary Camputation, pp. 69-80

Nakano, R. and Yamada, T., 1991, "Conventional Genetic Algorithm for Job Shop Problem", In Belew, R. K. and Booker, L.B. (editors), Proceedings of 4th International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 474-479.

Oliver, I. M., Smith, D. J. and Holland, J. R. C., 1982, "A study of Permutation Crossover

Operators on the Travelling Salesman Problem", In Grefenstette, J. J (editor), Proceedings of 2nd International Conference on Genetic Algorithms and their Applications, Erlbaum, pp 224-230,

Poon, P. W. and Carter, J N., 1995, "Genetic Algorithms Crossover Operators for Ordering Application", Computers and Operations Research, Vol. 22, No.1, pp. 135-147.

Reeves, C. R.. , 1995, "A Genetic Algorithm for Flowshop Sequencing", Computers and Operations Research, Vol. 22, No. 1, pp. 5-13.

Rubin, P. A and Ragatz., G. L., 1995, "Scheduling in a Sequence Dependent Setup Environment with Genetic Search", Computers and Operations Research, Vol. 22, No.1, pp 85-99

Schaffer, J. D., Caruana, R. A., Eshelman, L. J., Das, R., 1989, "A Study of Control Parameters affecting Online Performance of Genetic Algorithms for Function Optimization", In Schaffer, J. D. (editor), Procceding of 3rd International Confeence on Genetic Algorithms, Morgan Kaufmann, pp 51-60

Sridhar, J and Rajendran, C., 1994, "A Genetic Algorithm for Family and Job Scheduling in a Flowline based Manufacturing Cell", Computers and Industrial Engineering, Vol. 27, Nos. 1-4, pp. 469-472

Starkweather, T., McDaniel, S., Mathias, K. and Whitley, D., 1991, "A Comparison of Genetic Sequencing Operators", In Belew, R. K. and Booker, L. B. (Editors), Proceedings of 4th International Conference on Genetic algorithms, Morgan Kaufmann, pp. 69-76.

Starkweather, T., Whitley, D., Mathias, K, and McDaniel, S,, 1992, "Sequence Scheduling with Genetic Algorithms", In Fandel, G and Jones, J (editors), New Directions for Operations Research in Manufacturing, Springer-Verlag, pp. 129-148.

Syswerda, G., 1991, "Schedule optimization using Genetic Algorithms" In Davis, L. (editor), A Handbook of Genetic Algorithms, Van Nostrand reinhold, pp 332-349,

Whitley, D, Starkweather, T. and Fuquay, D, 1989, "Scheduling Problems and Travelling Salesman: The Genetic Edge Recombination Operator" In Davis, L (editor), International Conference on Genetic Algorithms, pp. 133-140

Williams, D, N., 1993, Machine Scheduling Problems with Setup Times, Unpublished M.Eng.Sc Thesis, The University of Melbourne.

Williams, D N. and Wirth, A., 1996, "A new heuristic for a Single Machine Scheduling Problem with Setup Times", Journal of Operational Research Society,Vol. 47, pp 175-180

Webmaster Address: itrssm@au.ac.th