EVOLUTIONARY ALGORITHM PARAMETER FITNESS: AN EXPLORATORY STUDY

Andrew Manikas, University of Wisconsin Oshkosh
Michael Godfrey, University of Wisconsin Oshkosh

ABSTRACT

Genetic algorithms were the first evolutionary algorithm designed. These algorithms simulate natural selection to produce good solutions quickly for complex problems. Job shop scheduling with sequence dependent setup times is an NP-hard problem – any algorithm to optimize this problem has an exponential time. We explore which parameters for a genetic algorithm allow it to solve these job shop problems in limited time trials. Prior literature assumes the use of these parameters is beneficial, and the parameter values are selected either based on prior research values or from design of experiments on a limited range of parameter values. Multiple linear regression is used to determine which parameters can significantly improve solutions. Our results show that a proportional 50/50 crossover parameter and a large population size are the two parameters to obtain good solutions in a constrained time environment.

JEL: C20, C60

KEYWORDS: Evolutionary Algorithms, Genetic Algorithm, Regression, Job Shop Scheduling

INTRODUCTION

Sequence dependent job shop scheduling (SDJSS) is a complex problem because it is NP-hard – any algorithm to optimize this problem has an exponential time – as the size of the problem increases, the computation time increases exponentially. Because the setup time of the next job on a machine is dependent on what was run on the machine previously, there is no linear method to optimize this class of problems. In shop floor scheduling, the most important objective is to produce a feasible schedule quickly. In the space of all feasible solutions, company defined goals drive which feasible solutions are better, and even which one is optimal. However, as the size of the problem becomes larger, and therefore nontrivial, the calculation time to find the optimal solution increases exponentially. Pure heuristics are quick to produce feasible solutions, but when more constraints are added to the model of the manufacturing environment, they cannot handle this added realism so their solutions may be far from optimal. We will test the ability of a genetic algorithm (GA) to achieve a good solution for complex job shop scheduling problems with sequence dependent setup times, staggered job release times (not all material is available for all jobs at time 0) and recirculation (a job can visit a machine multiple times). Genetic algorithms are a class of algorithms that can handle much complexity, and by design will converge toward an optimal solution as the number of iterations of the routine increases. This research is important because finding the best solution in a limited time is highly relevant for business scenarios. Therefore, finding which parameters contribute to solution goodness, and which are not statistically relevant is of importance for finding better solutions via evolutionary algorithms. Further, any parameters that do not contribute to solution goodness do not have to be programmed, thus reducing the complexity of the computer program and parameter value selections.

This paper is organized into the following sections; first is a literature review of job shop scheduling, sequence-dependent setups in scheduling, and GAs as an optimization tool. This is followed by sections containing a description of the problem we are trying to solve, the mechanics behind how a GA works in general and specifics on our genetic algorithm. The next section contains the results of our regression tests. The final section is the summary and discussion of the implications of the results along with directions for future research.

LITERATURE REVIEW

By their nature, job shop scheduling problems are complex. Job shops provide a unique scheduling problem because the routings are based upon the jobs that need to be processed; therefore the resource requirements are not based upon the quantity as in a flow shop, but rather on the routings for the products being produced. Heuristics have been developed to find optimal nondelay schedules in a job shop environment (Hutchinson and Chang, 1990). The heuristic in their paper minimizes makespan in an environment that does not contain sequence-dependent setup times. Heuristics are often problem size independent, but provide good solutions according to single criteria objective functions only. This research uses more real-world multiple criteria objective functions.

Sequence dependent setup times are explored in the literature for flow shops (Vakharia and Chang, 1990). Single machine scheduling with sequence dependent setup times is also explored in the literature. This problem is scheduled using a hybrid genetic algorithm (Miller, Chen, Matson and Liu, 1999). Manikas and Chang (2009) use a genetic algorithm to solve sequence dependent setup time job shop scheduling problems. However, their GA parameters and settings were based on limited experimental trials. Rubin and Ragatz (1995) schedule *n* jobs on a single machine with sequence dependent setup times using a Genetic Algorithm. A common solution to problems with sequence dependent setup times is to batch together like jobs to minimize setup times. This has been done on single machine problems (Cheng et al., 2001).

Genetic algorithms are a viable approach to solving optimization problems. The principles of GAs proposed by Holland (1975) are the foundation of evolutionary algorithms. Genetic algorithms simulate evolution via natural selection. The idea is to evolve a population of candidate solutions using operators inspired by natural genetic variation and natural selection (Mitchell, 1996).

Classical job shop scheduling problems with a set of *n* jobs to be processed on *m* machines have been solved using a GA. Candido (1998) adds realistic constraints and uses multiple objectives for their GA. The GA is used to find the initial solutions and refine locally improved solutions. However, this routine cannot handle sequence-dependent setup times. The ability of GAs to handle complex constraints is further shown by allowing dual-resource constraints in a scheduling problem (ElMaraghy, Patel, and Abdallah, 2000). They achieve better solutions by forcing feasible solutions from birth rather than allowing infeasible solutions to exist in the population. They choose to use a random initial population rather than heuristic based criteria to create an initial population that was intelligently designed. The paper also shows that linear order crossover (LOX) performs better than partially matched crossover for their particular problem. The LOX crossover method preserves the relative position between genes (Falkenauer and Bouffouix, 1991).

A GA is used to solve a job-sequencing problem (Zhao and Wu, 2001). They show that a feasible, good solution can be found in a reasonable time for this problem. Ombuki and Ventresca (2004) introduce crossover and mutation operators as well as an encoding scheme that ensure the GA schedule remains deadlock free. We chose to always have only feasible solutions in the population because scoring an infeasible solution regarding lateness does not make sense – an "on time" solution that was achieved by having operations scheduled not following the routing cannot be scored as a solution. We encode machine sequences and use mutation and crossover operators that ensure those sequences are always feasible.

Tuning parameters such as population size and number of generations can have a positive effect, but Mattfeld and Bierwirth (2004) state that if the parameters are within useful bounds, the expected improvement usually does not justify the costs of finding an even more appropriate fine tuning. Using domain specific knowledge, Miller et al. (1993) are able to improve the running speed of their GA for a multiple fault diagnosis problem. Najafi et al. (2009) tune population size and mutation probability real-time for a resource investment problem. A GA has certain parameters (e.g., population size, number of generations, etc.) that have to be specified. Finding the correct combination of settings is time consuming, so a self adapting genetic algorithm is discussed that seeds the parameters with random levels and has mechanisms to adjust them during the run (Sawai and Kizu, 1998). However, real time tuning takes precious CPU time away from the mechanisms in the GA that converge the solution toward optimal. Some research selects parameter values based solely on prior used values in research. Ojha et al. (2009) uses a set value for number of generations and population size.

Nagano et al. (2008) use ANOVA with six levels for population size, and after 68 combinations, find that their largest population size does best. Ruiz et al. (2006) test five levels for mutation probability and four levels for population size for flowshop problems. They find the best population size was the largest one. In our research, we analyze different parameter levels via regression to determine the parameter settings up front. Multiple objectives are handled easily with GAs. Having a single objective to solve for does not give the production planner much control to differentiate among many competing requirements or constraints. Richter (2002) argues that the use of a multiple objective fitness function has a positive effect on convergence speed. Our job shop scheduling problem involves earliness and tardiness as its two criteria, which are relatively comparable and competing. A single simple genetic algorithm is flexible, but cannot be efficient for all problems (Leonhardi et al., 1998). The encoding scheme, operators (crossover, mutate, elite, etc.) and specific parameters all need to be specified for a particular class of problems.

Problem Description

In many manufacturing environments, the sequence of jobs run on a particular machine affects the setup time. Job shop environments can have significant differences in setup times depending on the sequence of jobs run through the resources. For example, products with significant color differences are likely run from lighter to darker colors with minimal cleaning, but going from a black color product to a white color would require much more time to prepare the machine to avoid black bleeding through and creating a grayish batch.

In manufacturing environments a feasible solution is a must. An optimal solution to a scheduling model is a goal that often cannot be obtained given data and time constraints. However, a schedule certainly would benefit from having a schedule closer to optimal rather than just merely feasible. Heuristics can quickly create feasible solutions, but as problem complexity increases, the solutions may be far from optimal. Commonly used heuristics are Shortest Processing Time (SPT) and Earliest Due Date (EDD) that use a single regular measure as the criterion. We find the appropriate parameter settings for our GA to generate the best solutions quickly for multiple criteria.

To make our job shop scheduling problems more realistic, the environment contains staggered release dates and recirculation in addition to sequence setup times. Staggered release dates means that we do not assume all jobs are ready to start at the same time. Recirculation is the ability of a job routing to visit the same machine more than once. The job shop scheduling problems used in this paper are non-delay, meaning that a machine must process a job if it is available.

The setup times on a machine vary according to the job family of the previously processed job on that machine. A schedule is evaluated by looking at the total tardiness and total earliness. A multiple criteria objective function allows the solution result to better fit a company's needs. In this case, using the single criteria of tardiness may produce a result that has several jobs finishing very early. Having resources committed to finish jobs early reduces flexibility because that capacity could be left free for future orders that really do need to begin processing earlier to meet their due dates. Therefore, adding a second criteria to the objective function to minimize earliness gives additional weight to solutions that are closer to just in time. The weight for tardiness is set to 1000 times that of earliness. A large number was chosen because tardiness is, in general, much less acceptable in production than being early. However, among the set of solutions with equivalent tardiness, solutions that are less early are preferred. This rewards a schedule for not being tardy but also gives minor reward for being more just in time. Adjusting the weights can affect the final solution generated by the genetic algorithm. This gives flexibility for a production planner to adjust the weights on multiple objectives to force the genetic algorithm to find better solutions according to his/her criteria. Each job's routing is known a priori, but the operations sequences on the machines are unknown and have to be determined.

The objective function is: 1000 * tardiness + 1*earliness. We use just two weighted criteria (tardiness and earliness) here, but having many criteria has little effect on the speed of calculation because only after the mechanisms produce solutions are they evaluated one time according to the objective function. The lower the score according to the objective function, the better the schedule solution. For easier comparison between the different problems solved in this paper as well as between the different methods used, scores presented in the results section are expressed as a percentage of optimal.

<u>Mechanics of Genetic Algorithms</u>

The basic structure of the Genetic Algorithm is:

```
Initialize
Evaluate
Loop for Reproduction
    Clone Elite
    Mutate
    Breed Crossover
    Replace Population with Children
    Evaluate
Until Termination Criteria (e.g., one minute time limit)
```

Finding the optimal values for the GA parameters is important for making the heuristic converge to an optimal solution more quickly. From our literature review, we decided to focus on seven specific parameters; Elite %, Mutate %, Population Size, Number of pairwise inversions for mutation, Crossover method, Allow crossover, and Allow mutation.

Elite percent is the top percentage of the population that survives unchanged to the next generation. This ensures that the best solution or solutions are always passed to the next generation. This forces the algorithm to have non-decreasing solutions from generation to generation. The larger the population size, the larger the number of solutions that would be considered elite.

Mutate percent allows the population to achieve random differences from those that come about from crossbreeding alone. Mutate percent is different from Elite percent in that an Elite percent of 1% only looks at the top 1% of solutions and duplicates them, while a Mutate percent of 1% looks at every solution and each one has a 1% chance of being mutated. If selected, the solution is copied to the next generation and then mutated according to inversion discussed next.

Population size is how many solutions are in each population. Intuitively, the more solutions in a population, the more likely the best one will be closer to optimal. Number of pairwise inversions for mutation specifies the probability that a solution is selected to have pairs of alleles switched. One pair may be swapped for mutation, or multiple pairs for a mutated solution.

Crossover method may be a one point method that takes the head of one parent and the tail of the other parent to create a child solution. The other common crossover method is to intermix alleles. We examine one point crossover that is at the half way point in the solution sequence, and one point chosen at a random distance within the sequence. For the intermixing of alleles methods, the crossover gives each parent allele a 50% chance of being selected, thus forming the child solution in sequence. The other intermixed method gives a higher percentage chance of selecting the more fit parent. For example, if parent 1 had a score of 100 (where lower scores are better), and parent 2 had a score of 200, rather than a 50% chance of each parent's allele being chosen, parent 1 would have a 67% chance and parent 2 would have a 33% chance of its allele chosen at any point in the sequence.

Crossover yes/no is a parameter that turns crossover breeding on or off. This is used to explore the possibility that mutation only can produce good solutions. If this parameter is set to off, then mutation is forced on. Mutation yes/no is a parameter that allows mutation to occur or not. This parameter is used to explore if mutation is needed to maintain diversity, or it can be done solely via crossover. If mutation is not allowed, then crossover is forced to be on.

TEST SPECIFICATION AND RESULTS

Regression Model for Tests

Values for the each of the seven parameters fell into certain ranges: 1) Elite Percent was in the range [1%, 50%], and was a ceiling function. We must have at least the single best scoring solution survive to the next generation; 2) Mutation Percent was in the range [1%, 50%]: 3) Population Size was in the range [10, 1000] increments of 10; 4) Number of Pairwise Inversions was in the range [1, 10]; 5) Crossover Method was equally likely to select one of four methods: One point crossover at a random location, one point crossover at the halfway point of the sequence, 50% likelihood of each chromosome's allele being chosen when constructing a child sequence (50/50 method), and likelihood of a chromosome's allele being chosen proportional to the fitness of each parent; 6) Crossover allowed (Yes/No). If no, mutation must be allowed and no parent breeding will occur and 7) Mutation allowed (Yes/No). If no, crossover must be allowed and mutation percent is set to 0% and the number of pairwise inversions is set to 0.
1000 runs of the algorithm were made with random combinations of parameters above in their respective given ranges. With seven predictor variables, the regression model is a hyper plane:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_3 X_{i3} + \beta_4 X_{i4} + \beta_5 X_{i5} + \beta_6 X_{i6} + \beta_7 X_{i7}$$

Interaction is assumed to not exist between these parameters, and therefore the first-order model above is for additive effects on mean response.

Results

The results of the multiple linear regression are shown in Table 1 where *** indicates p-values with significance of less than .01.

Table 1: Multiple Linear Regression Results

Adjusted R Square	*0.1278*	
	Coefficients	
Intercept	2641539	***
Elite %	(1825)	***
Mutate %	(1144)	
Pairwise Int	26333	***
Pop Size	(207)	***
Cross Meth	(28811)	***
MutateYN	(205636)	***
CrossYN	(50913)	

*Examining all parameters shows that mutate percentage and turning off crossover are not statistically significant. These seven parameters account for 12.78% of the variance in solution goodness. ***, ** and * indicate significance at the 1, 5 and 10 percent levels respectively.*

R-Squared was 0.1339. Adding more parameters to this model can only increase R-Squared and never reduce it. Therefore, the adjusted coefficient of multiple determination adjusts R-Squared by dividing each sum of squares by its associated degrees of freedom. In this case, adjusted R-Squared is 0.1278 as shown in Table 1.

Due to the randomness inherent in the genetic algorithm, there is a wide range of values for scores. However, the coefficients for the four predictor variables and their associated p-values make it clear that there is a difference in what level each of the parameters is set to when running the genetic algorithm to solve this particular problem. Using SPSS v14.0, the following results in Table 2 were obtained for stepwise linear regression.

Table 2: Stepwise Regression

Model Summary

Model	**R**	**R Square**	**Adjusted R Square**	**R Square Change**	**F Change**	**Sig. F Change**
1[a]	0.237	0.056	0.055	0.056	59.528	0.000***
2[b]	0.317	0.100	0.098	0.044	48.749	0.000***
3[c]	0.336	0.113	0.110	0.012	13.996	0.000***
4[d]	0.351	0.123	0.120	0.010	11.686	0.001***
5[e]	0.363	0.132	0.128	0.009	10.258	0.001***

Five models adding in the most statistically influential parameters one at a time in a stepwise manner shows how much additional variation in solution goodness is modeled via the added parameters. [a.] Predictors: (Constant), Mutate?, [b.] Predictors: (Constant), Mutate?, Pop Size, [c.] Predictors: (Constant), Mutate?, Pop Size, Crossover Meth, [d] Predictors: (Constant), Mutate?, Pop Size, Crossover Meth, Pairwise Int., [e] Predictors: (Constant), Mutate?, Pop Size, Crossover Meth, Pairwise Int. Elite%. ***, ** *and * indicate significance at the 1, 5 and 10 percent levels respectively.*

The results above confirm the regression done in Excel 2007. Using mutation is statistically significant, but what percentage chance a candidate solution had of being mutated was not significant. Therefore, we set mutation percent to halfway within our range of values, i.e. 25%. The number of pairwise inversions per mutation was significant; specifically, the more inversions, the worse the final solution. Therefore, the number of pairwise inversions was set to one. The next tests focused on crossover method and population size only. The results of these tests are shown in Table 3 where *** indicates p-values with significance of less than .01.

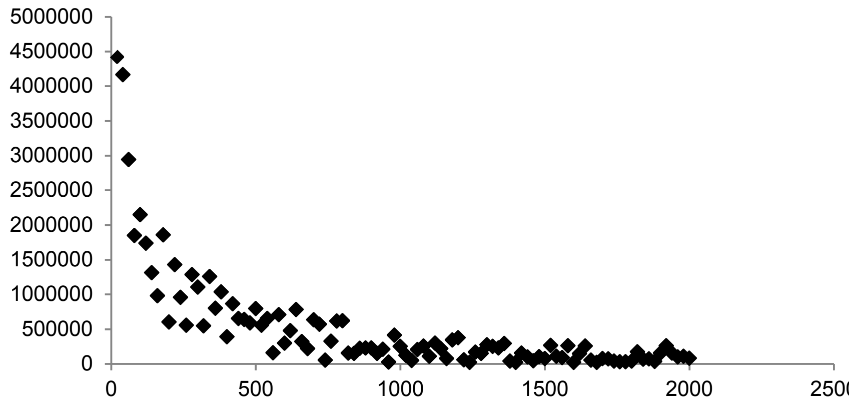
Table 3: Regression for Population Size and Crossover Only

Adjusted R Square	0.5054	
	Coefficients	
Intercept	4,285,189	***
Pop Size	(2.401)	***
Cross Meth	(680.07)	***

Population size and the crossover method chosen account for 50.54% of the variation in solutions. Furthermore, the crossover method for breeding has a very large, statistically significant coefficient (p-value <.01).

The results shown above indicate that we can simplify the model and worry only about two parameter ranges (population size and the number of generations). The programming complexity thus is reduced, and speed of computation improved. We also know that increasing either of these parameter values has a positive effect on the Y (score reduction). The significance is high so it is unlikely that the results seen are due to random variation of the data, but rather due to the change in predictor value. Furthermore, this analysis is helpful because the crossover method (Cross Meth) has a coefficient of -680.065, and population size (Pop Size) has a coefficient of -2.401. We can see that we get the most reduction in solution score by using a different crossover method for parent solution breeding. The 50/50 crossover method achieved the best results in the limited time. Population size is statistically significant in reducing solution scores to lower, better values. However, the low coefficient is counterintuitive because larger populations in prior research where time is unconstrained improve the solution greatly. Given the results of the regression, further tests were done to determine the impact of population size given the time limit. Without a time limit, increasing population size can produce better solutions. However, given limited time, having a larger breeding population may sacrifice computation time that could be used to produce more generations of offspring with a smaller population size. For population size, we tested values of 20 to 2000 in increments of 20. For each population size, 10 replications were done as shown in Figure 1.

Figure 1: Score per Population Size



The graph illustrates diminishing returns in solution goodness as population size increases. Populations increasing from 20 to 1400 show score improvement, while population sizes greater than 1400 have statistically insignificant improvements.

There is a statistically significant difference taking into account population size 20 through 1400 (p-value = .0000), however, from 1400 through 2000, the p-value is .98, meaning that score improvements are likely random versus correlated with the population size increase. This may be due partly to the increased population size limiting the number of generations where the breeding mechanisms of the GA can improve the solution in the one minute time limit.

IMPLICATIONS AND CONCLUSIONS

Common heuristics and Genetic Algorithms can find quick, feasible solutions to job shop scheduling problems that involve complexities such as sequence dependent setup times, spaced release dates and multiple objectives. Common heuristics run quickly, but do not converge to optimal like a GA. Not all parameters for the Genetic Algorithm affect the output goodness equally. Population size up to 1400 improved solution goodness and using a 50/50 crossover method significantly outperformed the other three crossover methods explored here.

Because of the speed with which the Genetic Algorithm runs, changes to customer release and due dates, routing changes, and order deletions and additions can be reflected in a new schedule quickly. The results of this research demonstrate that good solutions can be achieved without worrying about mutation percentage. It is possible to produce good results via mutation only, but superior solutions were obtained when 50/50 crossover was used to breed parent solutions.

Future research using genetic algorithms to solve sequence dependent job shop problems likely does not require researchers to use Design of Experiments (DOE) or regression to determine which parameters and settings to use – they may use our findings directly.

A limitation of this research is that it focused on Genetic Algorithms only. A future study could examine the parameter effects of hybrid GAs or Scatter Search (Manikas and Chang, 2008). Because some of the parameters are common amongst evolutionary algorithms, we would predict similar significant parameters as we found here. A further limitation of our research is that it was used to solve sequence dependent job shop scheduling problems only. Other studies that investigate parameters for other NP-hard problems might yield different parameter settings as optimal.

This Genetic Algorithm can be enhanced by adding more weighted objectives. Resource calendars, alternate machines with different costs or scrap rates, learning curves on machines and other complexities can be added to this algorithm with relatively little additional computing power required.

REFERENCES

Candido, M. A. B. (1998). A genetic algorithm based procedure for more realistic job shop scheduling problems. International Journal of Production Research, 36(12), 3437-3457

Cheng, T. C. E., Janiak, A., and Kovalyov, M. Y. (2001). Single machine batch scheduling with resource dependent setup and processing times. European Journal of Operational Research, 135, 177-183

ElMaraghy, H., Patel, V., and Abdallah, I. B. (2000). Scheduling of manufacturing systems under dual-resource constraints using genetic algorithms. Journal of Manufacturing Systems, 19(3), 186-201

Falkenauer, E. and Bouffouix, S. (1991). A Genetic Algorithm for Job Shop. IEEE International Conference on Robotics and Automation, Sacramento, CA.

Holland, J. (1975). Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI.

Hutchinson, J. and Chang, Y. L. (1990). Optimal nondelay job shop schedules. International Journal of Production Resources, 28(2), 245-257.

Lee, S. and Asllani, A. (2004). Job scheduling with dual criteria and sequence-dependent setups: mathematical versus genetic programming. Omega: The International Journal of Management Science, 32, 145-153

Leonhardi, A., Reissenberger, W., Schmelmer, T., Weicker, K., and Weicker, N. (1998). Development of problem-specific Evolutionary Algorithms, Universitat Stuttgart, Fakultat Informatik, Germany

Manikas, A. and Chang, Y. L. (2008). A scatter search approach to sequence dependent job shop scheduling. International journal of Production Research, 47(18), 5217-5236

Manikas, A. and Chang, Y. L. (2009). Multi-criteria sequence dependent job shop scheduling using genetic algorithms. Computers & Industrial Engineering, 56 (1), 179-185

Mattfeld, D. and Bierwirth, C. (2004). An efficient genetic algorithm for job shop scheduling with tardiness objectives. European Journal of Operational Research, 155, 616-630

Miller, D. M., Chen, H.C., Matson, J., and Liu, Q. (1999). A hybrid genetic algorithm for the single machine scheduling problem. Journal of Heuristics, 5(4), 437-454

Miller, J. A., Potter, W. D. , Gandham, R. V., and Lapena, C. N. (1993). An evaluation of local improvement operators for genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics, 23(5), 1340-1351

Mitchell, M. (1996). An Introduction to Genetic Algorithms, The MIT Press, Cambridge, MA.

Nagano, M. S., Ruiz, R., and Lorena, L. A. N. (2008). A constructive genetic algorithm for permutation flowshop scheduling. Computers & Industrial Engineering, 55, 195-207

Najafi, A. A., Niaki, S. T. A., and Shahsavar, M. (2009). A parameter-tuned genetic algorithm for the resource investment problem with discounted cash flows and generalized precedence relations. Computers & Operations Research, 36, 2994-3001

Ojha, D. K., Dixit, U. S., and Davim, J. P. (2009). A soft computing based optimization of multi-pass turning processes. International Journal of Materials and Product Technology, 35(1/2), 145-166

Ombuki, B. and Ventresca, M. (2004). Local search genetic algorithms for the job shop scheduling problem. Applied Intelligence, 21, 99-109.

Richter, H. (2002). An evolutionary algorithm for controlling chaos: The use of multi-objective fitness functions, Fraunhofer-Institut fur Produktionstechnik und Automatisierung: 308-317

Rubin, P. A. and Ragatz, G. L. (1995). Scheduling in a sequence dependent setup environment with genetic search. Computers and Operations Resources, 22(1), 85-99

Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research, 169, 781-800

Sawai, H. and Kizu, S. (1998). Parameter-free Genetic Algorithm Inspired by Disparity Theory of Evolution, Kansai Advanced Research Center, Kobe, Japan

Vakharia, A. and Chang, Y. L. (1990). A simulated annealing approach to scheduling a manufacturing cell. Naval Research Logistics, 37, 559-577

Zhao, V., and Wu, Z. (2001). A genetic algorithm approach to the scheduling of FMSs with multiple routes. International Journal of Flexible Manufacturing Systems, 13(1), 71-88

BIOGRAPHY

Dr. Manikas earned his B.S. in Computer Science and M.B.A. in Materials and Logistics Management from Michigan State University, and his Ph.D. from The Georgia Institute of Technology. Prior, he worked as an instructor for i2/JDA and as a management consultant for KPMG Peat Marwick and Deloitte Consulting. Email: manikasa@uwosh.edu

Dr. Godfrey earned his B.S. in Operations Management and M.S. in Management Information Systems from Northern Illinois University, and his Ph.D. in Production & Operations Management from the University of Nebraska - Lincoln. He is department chair of the Supply Chain & Operations Management department at UW Oshkosh. He is a CFPIM, CIRM, and CSCP through APICS and a CPSM through ISM. Email: godfrey@uwosh.edu