Refining Scheduling Policies with Genetic Algorithms

Emin Ogur
Department of Computer Science & Technology
University of Bedfordshire
Luton, Beds, LU1 3JU,UK
emin.ogur@beds.ac.uk

Mehmet E. Aydin
Department of Computer Science & Technology
University of Bedfordshire
Luton, Beds, LU1 3JU, UK
mehmet.aydin@beds.ac.uk

ABSTRACT

Genetic Algorithms (GAs) are popular approaches in solving various complex real-world problems. However, it is required that a careful attention is to be paid to the contextual knowledge as well as the implementation of genetic material and operators. On the other hand, the job-shop scheduling (JSS) problem remains as challenging NP-hard combinatorial problem, which attracts researchers since it is invented. The dynamic version of job-shop is even more challenging due to its dynamically changing characteristics. Similar to other metaheuristic approaches, GA has not been so successful in solving this sort of problems due to instant decision making process needed in solving this type of problems. Heuristic procedures such as those so called Priority Rule or Dispatching Rules are more useful for this purpose, but, depending on the properties and purpose of use of each, the same performance is not expected from these instant decision making operators. In this paper, a policy refinement approach is proposed to optimise a sequence of Dispatching Rules (DRs) for a time-window of scheduling process in which a GA algorithm evolves the sequences towards an optimum configuration. The preliminary results provided in this paper seem very encouraging.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: – Nonnumerical Algorithms and Problems – *Sequencing and scheduling*

G.1.6 [**Mathematics of Computing**]: Optimization – *Global Optimization*

G.3 [**Mathematics of Computing**]: Experimental Design –

I.2.8 [**Artificial Intelligence**]: – Problem Solving, Control Methods, and Search – *Scheduling*

General Terms

Algorithms, Management, Performance, Design, Experimentation.

Keywords

Policy refinement, genetic manipulation, semi-dynamic

1. INTRODUCTION

Job-shop scheduling (JSS) is one of well-known NP-hard combinatorial optimization problems, where N jobs are to be processed on M machines [26]. The complexity of such environment is variable with respect to different performance measures such as makespan, mean-tardiness subject to the constraints to be satisfied. Moreover, the term complexity becomes more meaningful with real-world dynamic scheduling. In a static context, all constraints are deterministically known in advance while in a dynamic version, the constraints may be either partially or completely unknown in advance [16, 25]. Due to the complexity of real-world problems the majority of research done in this area is based on static JSS [29]. Examples include [8, 19]. However there are many recent researches for dynamic JSS see for example [1, 4, 10, 16, 24, 28–31].

In this research, a semi-dynamic JSS case is considered with which the only stochastic item is job due-dates assigned following a probabilistic approach; total work content (TWC). The idea is to set our approach for a typical time-window of dynamic scheduling process, where it is aimed to be conducted through optimised time-windows.

The importance of solving scheduling problems in manufacturing systems lies behind issues in production systems. Due to its severe complexity, many researchers (especially from 1950s' onwards) have tested their problem solving approaches using JSS problems. Examples including Tabu Search [27], Simulated Annealing [15], and other heuristics such as Giffler and Thomson [21]. In the other hand, many researchers have followed Davis [14] whom proposed the first Genetic Algorithm (GA) for solving scheduling problems.

Holland [9] had devised GA to be used as a search heuristic for solving adaptation problems inspiring of natural evolution systems. Since then, it has become one of the most popular algorithms for solving scheduling problems besides other soft computing techniques such as Bee Colony Algorithm [11], Ant Colony Optimisation [22]. Since JSS is NP hard, the aim is switched from finding the optimum to a near-optimum since none of the heuristic approach can guarantee the optimum [18]. Since we know that GA reflects imitation to biological processes and developed by [9] for adaptation problems, it is sensible to use this approach in designing adaptable algorithms for solving dynamic problems [17, 25]. However, GA has its own drawbacks in use such as representation of the problem in the algorithm [7, 8, 25].

Although GA is an appropriate approach to search for solutions but modifying it to suit the problem domain by using real-world knowledge is also vital [13]. Dynamic job-shop scheduling problem is even more challenging than the classical one due to its dynamically changing characteristics. Similar to other metaheuristic approaches, GA has not been so successful in solving this sort of problems due to the instant decision making process needed in solving dynamic problems such as dynamic JSS. Heuristic procedures so called Priority Rule or Dispatching Rules are more useful for this purpose, but, depending on the

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.

technical features and the purpose of use of each rule, the same performance cannot be expected from these instant decision making operators. In this paper, a policy refinement approach adopted and proposed to set up a sequence of Dispatching Rules (DRs) for a typical time-window of scheduling process with which a GA algorithm evolves the sequences towards an optimum configuration. Strings of digits are adopted to represent policies with which it is followed to develop a complete schedule. It is well-known that a metaheuristic such as GA on its own is not advantageous for devising the sequence of tasks within a dynamic context, where instant decision making is needed for dynamic scheduling, while a search with GA will last much longer that makes it out of consideration. However priority-based operators (DRs) which we make use of in this research has an advantage of online search for responsive and instant decision making. An optimised combination of this sort of context-aware operators is believed to provide a better solution.

Dorndorf and Pesch [5] have used a similar approach to ours but the significant difference of our approach is that they choose only one rule [8] in chromosome evaluation process while we propose one rule per operation. More recently [23] have used dispatching rules with Genetic Programming (GP) for designing effective rules and Kawai and Fujimoto [12] have combined dispatching rules for generating a more effective approach.

In addition, it is believed that more sophisticated and successful algorithms can be constructed with assistance of experimental design methods, where the impact of each gene on the genome is calculated to bias the evolution process for better efficiency. The next part of our approach is to further investigate impact of each gene on the whole performance and calculate a weighting on this basis for better incorporation with evolution process.

The rest of the paper is organised as follows. In Section 2 we introduce the policy refinement approach with GA. In Section 3 some recent experimental results of this approach on makespan and mean tardiness objectives as well as cross testing of objective specific evolved fittest strings are presented. Some comparison with results from literature and details of future work based on experimental designs and its aim are also presented in this section. Finally, Section 4 presents conclusions.

2. POLICY REFINEMENT APPROACH WITH GENETIC ALGORITHMS

It is argued by Koza [3] that problem representation is a key issue in GA. More interestingly, the first researchers worked with GA to solve JSS problems [14] had also tried to solve representation issue in early attempts. The main drawback of binary string and some other representations have a high possibility of ending up in unfeasible schedules after genetic manipulations. A possible solution for this situation enforces to produce a repairing mechanism such as in [8, 17], which can cause extra computation time.

In order to avoid the problems introduced, local search and/or priority-based operators (dispatching rules) are used in this research are stored/represented in genes of chromosomes. These genes then play a decision making role prior to the rule it holds between conflict sets of jobs during the scheduling process. The chromosome which holds these genes are considered as policies. The role of GA in this approach is to orchestrate the iteration of these policies through evolution with genetic manipulation and elitism prior to each policy's fitness. The manipulation process is managed by use of basic operators of GA; crossover and mutation configuration. A series of generational GA algorithms have been devised and tested, where 3 out of 18 cases are found better performing. The successful cases are C8, C9 and C12 as tabulated in Table 1 with parametric configurations. For further details, see [25]. The best result of the evolution process is adopted as the fittest policy which is the best performing one among all other individuals appearing in all populations.

Table 1: Best GA configurations

Test Case	Crossover probability (%)	Crossover rate (%)	Mutation rate (%)
C8	60	20	5
C9	40	20	5
C12	60	40	5

The chromosomes, which will be called policies now on, include a sequence of priority rules, where each will be applied to assign a particular task/operation on corresponding machine. For representation of each rule in policies we use an integer-based scheme to ease invoking and applying the corresponding rule for making decision on which operation/job to be processed next. Through this approach, the manipulation process will not affect the feasibility of schedule since we only manipulate the decision mechanism; not the order of jobs or tasks.

Our approach in solving JSS problems lies behind automatic design of adaptable scheduling policies through iteration /evolution as part of GA process. Rather than evolving the task order or editing job process order, similar to [5, 25] we use DRs for decision making process. We first define two sets as follows:

$$\boldsymbol{s} = \{\Gamma_{\boldsymbol{k}}(\boldsymbol{o}_{\boldsymbol{i},\boldsymbol{j}}) \mid \boldsymbol{i} \in \boldsymbol{N}, \boldsymbol{j} \in \boldsymbol{M}\} \text{ and}$$

$$\Gamma_{\boldsymbol{k}} \in \{\boldsymbol{SPT}, \boldsymbol{SDR}, \boldsymbol{FIFO}, \boldsymbol{EDD}, \boldsymbol{Slack} / \boldsymbol{RPT}\}$$

where s denotes a complete schedule, $o_{i,j}$ denotes i^{th} operation of job j to be scheduled with policy consisting of rules, $\Gamma_{\boldsymbol{k}}$, which are given as the function of $o_{i,j}$. Since the whole scheduling process is managed through this policy, a new schedule can be denoted as $\hat{s} = \{\Gamma_k \mid k = 1,2,..\}$ where $|\hat{s}| = M \times N$. This brings in a probabilistic approach to task assignment to resources [5, 25] because $o_{i,j}$ is assigned with an undetermined Γ_k in an unknown time within $\hat{s}$. As explained before, the decision making mechanism in this context is very quick and hence the computation time is reduced where a great computation time is needed when using other heuristics such as breadth-first and depth-first search [5]. Through this approach, the aim is to find local optimum solutions using adaptable sequences of instant-decision-making rules rather than searching for the global optimal. Thus, an adaptable algorithm for dynamic environments can be generated. More details on how to develop adaptable algorithms can be found in [16].

3. EXPERIMENTAL RESULTS

The policy refinement approach with GA is applied to 10 JSS problems of 10x10 (MxN) size from OR library (Table 2) to optimise the schedules with respect to makespan and mean tardiness objectives. The main reason for using the problems of this size is that they are mid-size benchmark problems [5] to test an algorithm. After testing the performance of the algorithm, it

can be made scalable with incorporation of time-windowing approaches.

Table 2: Ten 10x10 benchmark problems from OR library with their optimal makespan

Problem Name	Optimal
abz5	1234
abz6	943
ft10	930
la16	945
la17	784
la18	848
la19	842
la20	902
orb01	1059
orb02	888

3.1 Test results with fittest policies

The results in this section are produced fittest policies refined by applying the GA approaches, which have evolved the strings consisting policies to find the fittest string over all ten problems (i.e. one fittest string for all problems). The chromosome evaluation for makespan objective is calculated as $f_{sum} = \sum_{i=1}^{K} C_{\max}$ and $\overline{T} = \frac{1}{K}\sum_{i=1}^{K} \max(0, d_i - C_i)$ for mean tardiness, where C_{max} is the makespan, K is the number of benchmark problems, d_i is the due-date of job i, C_i is completion time of job i respectively. Since we are seeking to find a global optimum for all problems (tested), we use this approach for evaluation. However problem specific evaluations will be discussed in following sections for comparison purposes.

3.1.1 Makespan

Since we know the optimal makespan results for benchmark problems, we compare our results as per their relative error (RE) which is calculated as $RE = (X - Y)/Y$ where X is the test result of the benchmark found and Y is its optimal solution.

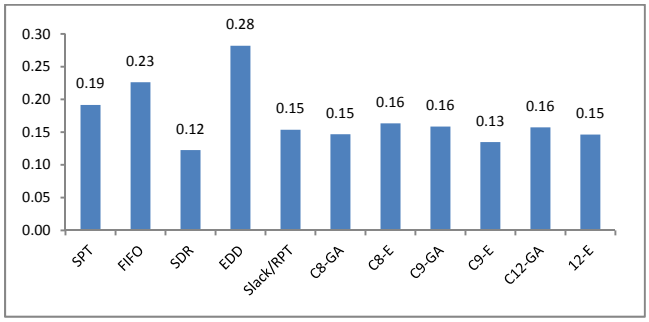

Figure 1: Average makespan of ten problems with optimised policy using GA

The test results by using individual rules show that SDR (shortest distance rule) outperforms other four DRs (Figure 1) where its average RE of makespan over ten problems is 0.12, while C9 provides 0.13, which keeps it competitive to SDR. On the other hand, one of the two due-date dependant rules (EDD) provides the worst result where second best performance comes from another (Slack/RPT) that our C8 and C12 configurations compete with. It is observed that, the most competitive results amongst evolutionary approach come from the fittest string of test C9 (C9-E) with an average of 0.13 RE.

3.1.2 Mean Tardiness

Unlike the test results with makespan, the best performing rule becomes EDD among all other rules and very near to the best evolutionary test case (C9-E). It is also observed that another due-date-based rule Slack/RPT becomes the second best performing rule (Figure 3). However the fittest string found by test case 9, C9-E, provides slightly better result than the best performing rule, EDD.

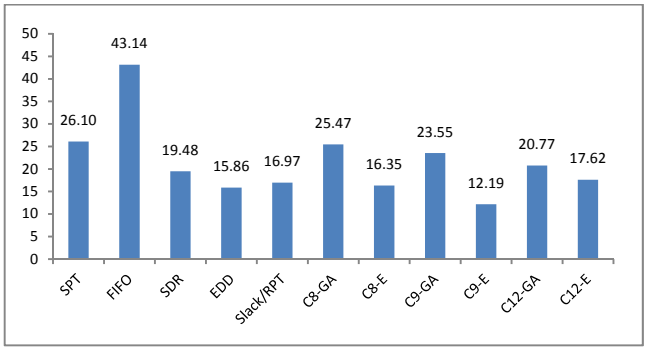

Figure 2: Average Mean Tardiness of ten problems with optimized policy using GA

It is important to note that we could not find any data for optimal solution of the benchmark problems in the literature for mean tardiness objective and therefore not using RE figures.

3.2 Cross Testing of Makespan and Mean Tardiness Fittest Policies

Figures 3 and 4 compare results of cross testing fittest strings (i.e. experimenting makespan (MS) objective with mean tardiness (MT) fittest and vice versa) with DRs. Here we optimised 2 policies; one with makspan and the other with mean tardiness. Then, both fittest policies are cross tested accordingly.

3.2.1 Makespan with Mean Tardiness Fittest Policy

As discussed previously, the SDR performs better than all rules and although competing, slightly better than evolutionary test cases optimized for makespan objective.

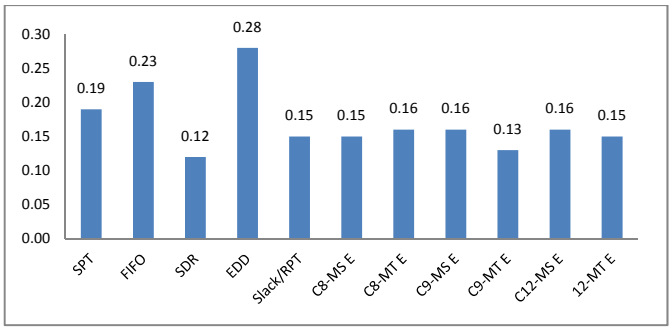

Figure 3: Average makespan of ten problems with mean tardiness fittest string

Besides that, the fittest policy of mean tardiness objective of each test case provides close results to makespan fittest results as well as performing better than FIFO, EDD and Slack/RPT and

competing with SPT. This means that the fittest strings evolved for mean tardiness objective can also be efficient for makespan.

3.2.2 Mean Tardiness with Makespan Fittest Policy

The results shows a similar picture to our previous approach (testing MS with MT fittest) in such way that there is a minor difference between the performance of all genetic approach test cases (Figure 4). Hence, it is understandable that makespan fittest string is not outperforming but remains competitive whilst used for mean tardiness.

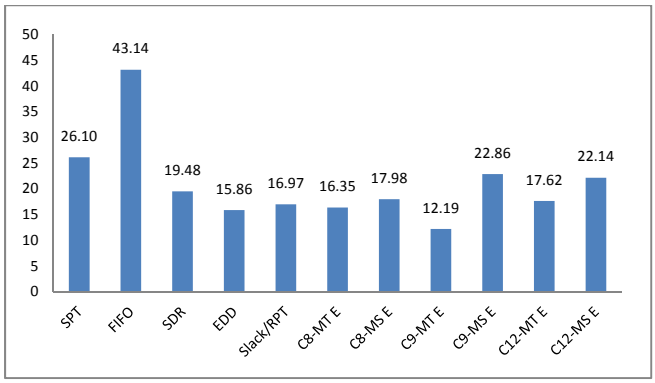

Figure 4: Average mean tardiness of ten problems with both objective fittest strings

3.3 Problem specific evolution

As mentioned previously, we have also tested the algorithm on each of the selected ten problems in order to compare with the results taken from [5] which their approach in solving the problem is closest to ours. In other words, we have evaluated the policy fitness as per C_{max} of each job specifically rather than an aggregate evaluation for ten problems to be in-line with the approach resented in [5].

Table 3: Policy refinement and P-GA makespan comparison

Problem Name	Policy Refinement	P-GA	Optimal
abz5	1266	—	1234
abz6	977	—	943
ft10	964	—	930
la16	985	1008	945
la17	793	809	784
la18	880	916	848
la19	875	880	842
la20	938	928	902
orb01	1110	—	1059
orb02	912	—	888

Table 3 presents the makespan result comparison of our policy refinement approach with results from [5]. The results given are the best out of 30 test cases that were made based on test case C9-E configurations, which performed better than others. It is clearly seen that our approach outperforms 4 (la16, la17, la18 and la19) out of 5 problems tested by [5] where their approach performs better than ours only for one (la20) problem. The data indicated with "—" states that no test results were found in their research.

3.4 Statistical Analysis

In this section, we report the results of statistical analysis to reveal the significance of the results. All experimental results presented in this paper are averaged over 30 repetitions. Few t-tests have been conducted to measure the significance of the differences between the compared results. The main purpose is to investigate how significant the differences between comparative results are, where we compare the results produced by evolved fittest-policies using GA-based algorithms and the best performing individual DRs with respect to both objectives; makespan (MS) and mean-tardiness (MT).

The results presented in Figure 1 and Figure 3 suggest that the best performing DR is SDR with respect to MS objective while the worst performance is delivered by EDD rule in this respect. On the other hand, EDD becomes the best performing rule among all others with respect to MT objective (see Figure 2 and 4). Meanwhile, the best configuration of our GA approach is found to be C9-E, therefore, we decided to apply t-test to reveal the significance of the differences between the results by SDR and EDD versus C9-E in terms of both objectives.

Table 4: t-Test results for statistical significance

Objective	SDR vs C9-E	EDD vs C9-E
Makespan (MS)	0.1789	5.8E-05
Mean-Tardiness (MT)	0.0977	0.1170

Table 4 presents the t-test results conducted to test the significance of differences across the results of the fittest-policy evolved by C9-E, SDR and EDD priority rules with respect to both objectives; MS and MT. The second column of the table shows the t-test results for significance level of the comparison between the performance of SDR rule and C9-E with respect to both objectives while the third column presents the same kind of statistics for EDD versus C9-E.

As discussed previously, SDR rule provides better results than any other approach including C9-E as seen in Table 4, where the difference between the results is insignificant even in 90% confidence level. This means that C9-E remains competitive with respect to MS although it does not outperform the best performing rule, SDR. However in the case of MT objective, C9-E significantly outperforms the SDR rule as well as all other rules including EDD, while EDD remains competitive with C9-E as the difference between the results are not significant in 95% confidence level, but, others are significantly outperformed in 99% confidence level. It is clearly concluded that optimisation with respect to MT using fittest-policies can be improved outperforming the individual rules, while performing well with respect to MS and vice versa. That is, a policy evolved for one objective provides close results when applied on another.

3.5 Experimental Designs and Main Effects

As per the proposed research, our aim is to design a set of handy experiments that could help optimise the solutions further. Orthogonal arrays (OA) have been used for designing experiments in the literature. For example [32] uses OA in designing crossover operator in GA to solve multimedia multicast routing. As they support the idea that use of such sophisticated experimental design provides a better performance, it is believed that the use of such statistics will lead to development of a probabilistic model such that evolved fittest policies can be converted into such probabilistic model to find chose the most fitting DR subject to

imposed circumstances. This is expected to facilitate for better scalability.

In factorial experimental design, a full factorial approach would provide all possible combinations of policies in our case. However, since we have 5 rules (levels) and 100 operations (factors), which mean a problem size of 5^{100}, it is almost impossible to conduct such set of experiments with available computation resources. In fact, OA helps not to test all combinations with design of sufficient number of policies and appropriate combinations which represents possible combinations [6, 20, 32] . In this way, a more reasonable experiment is achieved.

By use of experimental designs, we aim to analyse the main effects of each gene within a policy that would guide us to modification of such gene for optimisation. Hence, a better algorithm will be generated and a brighter picture of solution space will be seen.

4. CONCLUSION

An adaptable algorithm for instant decision making mechanism within dynamic job shop scheduling process to make selection among conflict set of operations is generated and tested successfully. Due to responsiveness and time sensitivity, dynamic job shop scheduling cannot afford long-lasting search process, but, can only use instant decision making procedures. In this paper, a GA –based evolutionary algorithm is proposed to refine the policies consisting sequences of priority rules for this purpose. The evolution is applied with respect to two objectives; makespan and mean-tardiness. The preliminary results suggest that the fittest-policies evolved with GA significantly outperform the individual priority rules with respect to mean-tardiness while remain competitive to the best rules with respect to makespan. We have concluded that optimisation with respect to MT using fittest-policies can be improved outperforming the individual rules, while performing well with respect to MS, too. Similarly, optimisation with MS also provides competitive results to DRs for MT objective respectively.

The test results are compared with a similar approach from literature and provide better results. However, our aim is to take this a step further with design of experiments and this will be the basis of our future work because we believe that understanding the affects of genes on schedule will provide a better picture of the solution space and hence, guide us to a more sophisticated algorithm.

In addition, although a semi-dynamic version of JSS is tested in this paper, one of the core aims of our research is to test the algorithm in a fully dynamic environment where job arrivals, machine break downs and other perturbations occur. However we are keen to continue using GA which we believe is suitable for adaptation in such situation.

5. REFERENCES

[1] Aydin, M.E. and Öztemel, E. 2000. Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*. 33, 2–3 (2000), 169–178.

[2] Congram, RK. Potts, C., Velde, S.L.V. 1998. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on ….* (1998).

[3] Dept, S.U.C.S. and Koza, J.R. 1990. *Genetic Programming: a Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems*.

[4] Dimitrov, T. and Baumann, M. 2011. Genetic algorithm with genetic engineering technology for multi-objective dynamic job shop scheduling problems. *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation - GECCO '11*. (2011), 833.

[5] Dorndorf, U. and Pesch, E. 1995. Evolution based learning in a job shop scheduling environment. *Computers & Operations Research*. 22, I (1995), 25–40.

[6] Fang, K.T. and Wang, Y. 1994. *Number-Theoretic Methods in Statistics*. Chapman & Hall.

[7] Hart, E. and Ross, P. 1998. A heuristic combination method for solving job-shop scheduling problems. *Parallel Problem Solving from Nature—PPSN V*. (1998).

[8] Hasan, S.K. 2008. GA with priority rules for solving job-shop scheduling problems. *Proc. of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008)*. (2008), 1913–1920.

[9] Holland, J.H. 1992. *Adaptation in Natural and Artificial Systems*. MIT Press.

[10] Iioitorrit, D.J. and Lull, B. 1993. Scheduling the Dynamic Job Shop *. (1993), 71–76.

[11] Karaboga, D. and Basturk, B. 2008. On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*. 8, 1 (2008), 687–697.

[12] Kawai, T. and Fujimoto, Y. 2005. An efficient combination of dispatch rules for job-shop scheduling problem. *Industrial Informatics, 2005. INDIN '05. 2005 3rd IEEE International Conference on* (2005), 484–488.

[13] L. Davis 1989. Adapting Operator Probabilities in Genetic Algorithm. *Proceedings of the Third International Conference on Genetic Algorithms*. (1989), 61–69.

[14] L. Davis 1985. Job Shop Scheduling with Genetic Algorithms. *Proceedings of the First International Conference on Genetic Algorithms*. (1985), 136–140.

[15] Van Laarhoven, P. J. M., Aarts, E. H. L., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Oper. Res.*, *40*(1), 113–125.

[16] Madureira, A., Ramos, C., & Silva, S. (2001). A Genetic Approach for Dynamic Job-Shop. *Proceedings of MIC'2001 - 4th Metaheuristics International Conference 41*, 41–46.

[17] Madureira, A., Ramos, C., & Silva, S. do C. (2003). Using Genetic Algorithms for Dynamic Scheduling. *Proceedings of Production and Operations Management 2003 Conference (POMS'2003), Georgia (EUA)*.

[18] Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs* (Third, Rev. and Ext. Ed.). Springer.

[19] Miyashita, K. 2000. Job-shop scheduling with genetic programming. *Proc. of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. 505–512.

[20] Montgomery, D., C. (1991). *Design and Analysis of Experiments* (3rd ed.). New York: Wiley.

[21] Moonen, M. and Janssens, G. 2007. A Giffler-Thompson focused genetic algorithm for the static job shop scheduling problem. *The Journal of Information and Computational Science*. 4, 2 (2007).

[22] Neto, R. F. T., & Filho, M. G. (2011). An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed. *Computers & Operations Research*, *38*(9), 1286–1293.

[23] Nguyen, S., Zhang, M., Johnston, M., & Tan, K. (2012). A Computational Study of Representations in Genetic Programming to Evolve Dispatching Rules for the Job Shop Scheduling Problem. *Evolutionary Computation, IEEE Transactions on*.

[24] Nie, L., Gao, L., Li, P., & Zhang, L. (2011). Application of gene expression programming on dynamic job shop scheduling problem. *Proceedings of the 2011 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 291–295.

[25] Ogur, E., Aydin, M. E., & Ayhan M. B. (2012). Policy Refinement with Genetic Algorithms for Job Shop Scheduling. *Proc. of 8th International Symposium on Intelligent and Manufacturing Systems, 27-28 September 2012, Adrasan/Antalya/Turkey.*

[26] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. Springer, 2008

[27] Ponnambalam, S. G., Aravindan, P., & Rajesh, S. V. (2000). A Tabu Search Algorithm for Job Shop Scheduling. *The International Journal of Advanced Manufacturing Technology*, *16*(10), 765–771.

[28] Proth, J.-M. 1998. An improvement of the Lagrangean relaxation approach for job shop scheduling: a dynamic programming method. *IEEE Transactions on Robotics and Automation*. 14, 5 (1998), 786–795.

[29] Vazquez, M., & Whitley, L. (2000). A Comparison of Genetic Algorithms for the Dynamic Job Shop Scheduling Problem.

[30] Wei, Y. 2004. Composite rules selection using reinforcement learning for dynamic job-shop scheduling. *IEEE Conference on Robotics, Automation and Mechatronics, 2004.* 2, (2004), 1083–1088.

[31] Zhang, B., Yi, L., & Xiao, S. (2005). Study of stochastic job shop dynamic scheduling. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, August 2005*, (August), 18–21.

[32] Zhang, Q., Leung, Y., & Member, S. (1999). An Orthogonal Genetic Algorithm for Multimedia Multicast Routing. *IEEE Transactions On Evolutionary Computation*, *3*(1), 53–62.