ICIC Express Letters
Volume **10**, Number **4**, April **2016**

 ISSN 1881-803X
pp. **923–928**

ADAPTIVE DISPATCHING RULE FOR JOB SHOP SCHEDULING PROBLEM VIA GENE EXPRESSION PROGRAMMING

Liping Zhang, Qiuhua Tang and Peng Zheng

College of Machinery and Automation
Wuhan University of Science and Technology
No. 947, Heping Avenue, Qingshan District, Wuhan 430081, P. R. China
{ zhangliping; tangqiuhua }@wust.edu.cn; tbhc9045@yeah.net

Received October 2015; accepted January 2016

Abstract. *This study redesigns the gene expression programming (GEP) approach to generate dispatching rules for the job shop scheduling problems (JSP). Four parameters about JSP are encoded as the terminal set to boost the self-adaptive capability of the obtained dispatching rules, including the processing time of operations, the number of remaining unscheduled operations, the sum of jobs' remaining processing time and a random numerical constant. Five arithmetic operators are selected as the function set to formulate the nonlinearity and complexity among these parameters. Utilizing the genetic operators of GEP containing selection, mutation, transposition and recombination, an optimal functional representation representing dispatching rule is obtained. Meanwhile, an iterative re-start mechanism is combined to escape from the local optimum. Experimental results with 23 training cases and 7 testing cases show that the obtained dispatching rule via GEP possesses superiority in searching better solutions, robustness and flexible adaption to the given problem.*

Keywords: Adaptive dispatching rule, Job shop scheduling problem, Gene expression programming

1. **Introduction.** Scheduling is one of the most critical issues in the planning and managing of manufacturing processes. One of the most difficult problems in this area is the job shop scheduling problem (JSP), which has been proved to be an NP-hard problem [1]. JSP has been investigated via exact method, dispatching rules or meta-heuristic algorithms [2,3]. The exact methods can guarantee global convergence, but it requires exponentially increasing computing times as the size of problem increases. Hence, the dispatching rules, also called heuristic approaches, and meta-heuristic algorithms attract growing attention of researchers. With respect to meta-heuristic algorithms, it has been computationally proven that they can find high-quality solutions within reasonable computational time [4]. However, meta-heuristic algorithms are not convenient to be applied by the manager in the real enterprise because the solutions from meta-heuristic algorithms must be converted into corresponding orders. On the other hand, the dispatching rules have many advantages like the ease of implementation, satisfactory performance, low computational requirement, and flexibility to incorporate domain knowledge and expertise, and hence are being used more frequently by the industry. Unfortunately, as demonstrated by Huang and Suer [5], the quality of the solutions by dispatching rules is low due to the lack of flexibility and needs to be improved in matching the specific environments. Gene expression programming (GEP) is a genetic algorithm in which the individuals are encoded as linear strings of fixed length which are afterwards expressed as nonlinear entities of different sizes and shapes [6].

This paper proposes a GEP-based adaptive dispatching rule to solve the JSP with makespan as the objective function. Basically, the dispatching rules generated by GEP algorithm are heavily dependent on the problem characteristics and parameters, and vise

verse result in the self-adaptive capability of the proposed rules to the given problem. The adopted parameters characterizing the JSP problem include the processing time, the number of remaining unscheduled operations, and the sum of jobs' remaining time. In addition, 30 benchmarks are employed to evaluate the performance of the proposed approach.

The remainder of the paper is organized as follows. Section 2 describes the JSP problem. Section 3 details the methodology of producing dispatching rules via gene expression programming algorithm. Section 4 reports and analyzes corresponding results. In Section 5, a general conclusion is drawn.

2. **Problem Statement.** There are a set of machines, a set of n $(i = 1, 2, \ldots, n)$ jobs with each job having a set of $|J_i|$ $(j = 1, 2, \ldots, |J_i|)$ operations. Solving JSP problem means to seek for an optimal or near-optimal sequence $S = \{s|s = 1, 2, \ldots, K\}$ of these K $(K = \sum_i |J_i|)$ operations so as to minimize the completion time of all operations $C_{\max}$. Note that, in the following equations, p_{ij} is the processing time of operation O_{ij} on the predefined machine, m_{ij} is the machine predefined to perform O_{ij}, M is a big number, and C_{ij} represents the completion time of O_{ij}. Y_{ijs} equals 1 if O_{ij} is the sth operation in the sequence and 0 otherwise.

$$\min C_{\max},\ C_{\max} = \max_{1\le i\le I, 1\le j\le |J_i|} \{C_{ij}\} \tag{1}$$

$$\text{s.t.} \quad \sum_s Y_{ijs} = 1,\ \forall O_{ij} \tag{2}$$

$$\sum_i \sum_{j|j\le |J_i|} Y_{ijs} = 1,\ \forall s \in S \tag{3}$$

$$\sum_s sY_{ijs} < \sum_s sY_{i,j+1,s},\ \forall O_{ij}, O_{i,j+1}, j < |J_i| \tag{4}$$

$$C_{ij} + p_{i'j'} \le C_{i'j'} + M \cdot (2 - Y_{ijs} - Y_{i'j's'}),\ \forall O_{ij},\ O_{i'j'},\ s < s',\ m_{ij} = m_{i'j'} \tag{5}$$

$$C_{ij} - P_{ij} \le C_{i'j'} - p_{i'j'} + M \cdot (2 - Y_{ijs} - Y_{i'j',s+1}), \forall O_{ij}, O_{i'j'}, s < K \tag{6}$$

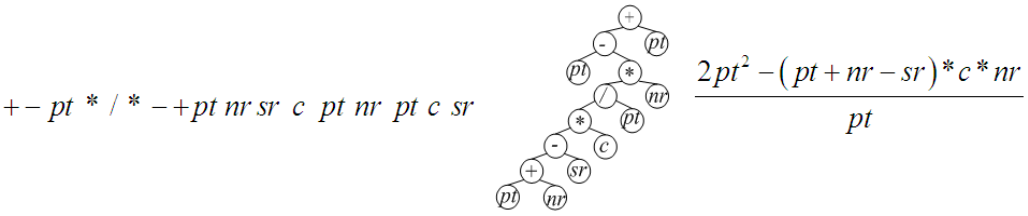
The objective function described in Equation (1) is to minimize makespan. For the constraints, each operation should be performed exactly once and only one task exists in each position of the sequence in Equations (2) and (3). Equation (4) defines the precedence relations constraints that each operation must be executed after its precedent operations have been finished. Equation (5) concerns time sequence constraint that one operation can start after the earlier one being assigned to the same machine has been finished. Equation (6) means that the immediate following operation can be started only after the immediate preceding operation in the sequence has been started.

3. **Methodology.** Gene expression programming (GEP) is a new technique of evolutionary algorithms for data analysis [6]. It has been successfully applied in symbolic regression, time series prediction, classification, optimization, etc. [7]. A novel iterative re-start mechanism is addressed to ensure the global optimum of GEP. The core steps of the algorithm contain parameter setting, population initialization, selection, mutation, transposition, recombination, evaluation, and conditional termination.

3.1. **Representation.** Each GEP chromosome is generated randomly with the gene from a function set (FS) and a terminal set (TS) at the beginning of the search process. It is composed of a list of symbols with a fixed length, which can be divided into two parts, a head and a tail. The head can be any symbol from FS and TS, while the tail contains only terminals. The choice of proper elements for FS and TS is a crucial step in the implementation of the learning process and has a significant effect on the learning ability of GEP [8]. In our GEP system, we define four properties including processing time

(pt), the number of remaining unscheduled operations (nr), the sum of jobs' remaining processing time (sr), and a constant (c). These four properties $\{pt, nr, sr, c\}$ consist of the TS.

One of the improvements of GEP over GP and GA is to separate phenotype from genotype. The genotype and phenotype are mutually dependent and can be transformed into each other directly [7]. For example, from the K-expression in Figure 1(a), we can obtain the expression tree and math expression illustrated in Figure 1 with depth-first search mode.



(a) K-expression (b) Expression Tree (c) Math Expression

FIGURE 1. Expression tree and math expression for K-expression (a)

3.2. **Genetic operators.** Genetic operators contain selection, mutation, transposition and recombination. Roulette-wheel sampling with elitism is selected to generate the next population. One point mutation and flip mutation are selected. The insertion sequence transposition and root insertion sequence transposition are applied for each chromosome [9]. We choose one-point and two-point recombination for two chromosomes.

3.3. **Fitness function.** If and only if the fitness function is clearly and correctly designed, the population can be evolved in the predetermined direction [7] and the given problem can be solved successfully. In our study, a near optimum is expected for the problem. A mean square error (MSE) in Equation (7) is adopted to evaluate the chromosome of GEP.

$$f = \sqrt{\frac{\sum_{r=1}^{s} (p_r - t_r)^2}{s - 1}} \tag{7}$$

where, f denotes the fitness of the chromosome, and p_r and t_r represent the real makespan and the target makespan of case r respectively. Here, the target makespan is the lower bound or the current best solution. We calculate the priority values for each candidate operation with the math expression, and then schedule the operation with the highest priority first until the candidate operation set is empty. Thus, the real makespan p_r for each case is obtained.

3.4. **The random numerical constant.** An extra terminal set is defined to handle the numerical constant. This set will be chosen to process the genetic operators according to the given probability. Note that, each element in this extra terminal set is defined specifically according to the given problem. For example, we select the constant array $\{0.1, 2.5\}$, and then define the extra terminal set as $\{b, c\}$. That means $b = 0.1$, $c = 2.5$.

3.5. **The iterative re-start mechanism.** The traditional GEP is easy to converge prematurely and be trapped into the local optimum. In our GEP system, an iterative re-start mechanism is applied when the current best solution has not been updated after several generations. When the number of iterations without improvement reaches to the maximum number of the pre-set, the iterative re-start mechanism is triggered. The chromosomes are selected with the given probability and reconstructed randomly from scratch.

4. **Results and Discussions.** We apply 30 benchmarks about JSP (as shown in Table 1) to finding the dispatching rules by the redesigned GEP. Researchers have paid attention to these famous benchmarks and obtained the current best solutions or the lower bound (represented by MS) in Table 1. These MS are employed as target makespan in our approach.

Table 1. The experimental data

SN	P	J	M	MS	SN	P	J	M	MS	SN	P	J	M	MS	SN	P	J	M	MS
1	ABZ5	10	10	1234	9	LA05	10	5	593	17	LA14	20	5	1292	25	LA24	15	10	935
2	ABZ6	10	10	943	10	LA06	15	5	926	18	LA16	10	10	945	26	LA25	15	10	977
3	FT06	6	6	55	11	LA08	15	5	863	19	LA17	10	10	784	27	LA26	20	10	1218
4	FT10	10	10	930	12	LA09	15	5	951	20	LA18	10	10	848	28	LA28	20	10	1216
5	LA01	10	5	666	13	LA10	15	5	958	21	LA19	10	10	842	29	LA30	20	10	1355
6	LA02	10	5	655	14	LA11	20	5	1222	22	LA20	10	10	902	30	LA32	30	10	1850
7	LA03	10	5	597	15	LA12	20	5	1039	23	LA22	15	10	927					
8	LA04	10	5	590	16	LA13	20	5	1150	24	LA23	15	10	1032					

Note: SN, P, M, J, and MS are the abbreviation of serial number, the problem, the total number of machines, the total number of jobs, and makespan.

4.1. **Parameters tuning.** It is worth mentioning that some parameters must be predetermined before the redesigned GEP are implemented in the JSP problem. After a serial of preliminary trials, these parameters are confirmed. The population size, the number of iterations, the head size, the tail size and the constant equal 10, 200, 8, 9 and 55 respectively. One point mutation rate, flip mutation rate, one-point recombination rate, two-point recombination rate, insertion sequence transposition, root insertion sequence transposition, the pre-set iterations and the pre-set probability for the iterative re-start mechanism equal 0.05, 0.05, 0.2, 0.2, 0.15, 0.1, 10 and 0.3 respectively.

4.2. **Experimental result.** The proposed approach has been programmed in C++ language and run on a PC with Intel Core 2 Duo CPU 2.20 GHz processor and 2.00 GB RAM memory. From Table 2, there is a total number of 30 simulation experiment sets, which are divided into 2 groups with 23 and 7 cases respectively. All data in Group 1 are used as a training set to seek for the function expression. And the function expression of the obtained best individual is applied to the problems in Group 2 so as to check the validation of the dispatching rules. Each simulation experiment has run 10 times. The best result over the 10 different runs is $\left\{+// - pt/ * \sqrt{\ } \; sr \; nr \; m \; sr \; pt \; nr \; pt \; m \; sr\right\}$. Note that, m is the constant and equals 55. The dispatching rule obtained by the GEP is shown in Equation (8).

$$F(pt, nr, sr) = \left(pt - \left(nr * \sqrt{sr}\right)/55\right)/(pt * sr) + nr \tag{8}$$

To test the performance of the obtained dispatching rule, we apply this dispatching rule to calculating the makespan of 30 benchmarks shown in Table 2. Meanwhile, the results from 8 classical widely-used dispatching rules for these problems are also listed for comparison. Detailed discussion of these selected rules can be found in reference [10].

We first measure and compare the makespan. As illustrated in Table 2, 24 cases calculated by GEP are superior to other 8 dispatching rules. Specially, 3 of them, LA09, LA10 and LA14, have reached the lower bound. Meanwhile, we can see that LRM and MWKR also play a good performance at several cases including FT10, LA01, LA03, and MWKR finds the lower bound of LA14. In summary, the obtained dispatching rule via GEP possesses powerful ability in searching better solutions.

The robustness of solutions has a dominant impact especially when the production parameters are changed such as the number of jobs or operations. Table 2 shows that

Table 2. The makespan/deviation results by different dispatching rules

Problem	LB	SPT	LPT	SSO	LSO	SRM	LRM	MWKR	SWKR	GEP
ABZ5	1234	2509/1.03	2730/1.21	2829/1.29	3186/1.58	3629/1.94	1547/0.25	1451/0.18	3318/1.69	**1370**/0.11
ABZ6	943	2039/1.16	2557/1.71	2434/1.58	2497/1.65	2617/1.77	1101/0.16	1096/0.16	2418/1.56	**1091**/0.15
FT06	55	78/0.41	104/0.89	72/0.31	74/0.34	103/0.87	63/0.14	67/0.21	103/0.87	**60**/0.09
FT10	930	1850/0.99	1916/1.06	1714/0.84	1886/1.02	2021/1.17	**1136**/0.22	1194/0.28	1790/0.92	1151/0.23
LA01	666	1097/0.64	1200/0.80	1227/0.84	982/0.47	1376/1.06	778/0.16	735/0.10	1489/1.23	775/0.16
LA02	655	1218/0.86	1399/1.13	1143/0.75	1086/0.66	1257/0.92	851/0.30	902/0.38	1217/0.85	**901**/0.37
LA03	597	915/0.53	1039/0.74	908/0.52	858/0.43	1056/0.77	748/0.25	731/0.22	1066/0.78	791/0.32
LA04	590	1049/0.77	1103/0.87	1115/0.89	1082/0.83	1137/0.92	846/0.43	817/0.38	1333/1.26	**716**/0.21
LA05	593	931/0.57	977/0.64	892/0.50	899/0.52	994/0.67	630/0.06	612/0.03	1186/1.00	614/0.03
LA06	926	1388/0.50	1618/0.74	1531/0.65	1457/0.57	1496/0.62	974/0.05	1021/0.10	1941/1.09	**952**/0.02
LA08	863	1263/0.46	1804/1.09	1505/0.74	1414/0.64	1878/1.17	1111/0.28	1101/0.27	1739/1.01	**935**/0.08
LA09	951	1665/0.75	1828/0.92	1792/0.88	1454/0.53	2063/1.17	1073/0.12	1132/0.19	2012/1.11	**951**/0.00
LA10	958	1542/0.61	1539/0.60	2144/1.23	1188/0.24	1785/0.86	1049/0.09	1064/0.11	1795/0.87	**958**/0.00
LA11	1222	2027/0.66	2036/0.66	1888/0.54	1774/0.45	2446/1.00	1387/0.13	1459/0.19	2593/1.12	**1229**/0.01
LA12	1039	1687/0.62	1634/0.57	1817/0.75	1524/0.46	2144/1.06	1251/0.20	1176/0.13	2089/1.01	**1103**/0.06
LA13	1150	1872/0.63	2505/1.17	1919/0.67	1797/0.56	2546/1.21	1306/0.13	1277/0.11	2388/1.07	**1167**/0.01
LA14	1292	2132/0.65	1889/0.46	2157/0.67	1637/0.27	2305/0.78	1345/0.04	1292/0.00	2400/0.85	**1292**/0.00
LA16	945	1921/1.03	1945/1.05	1919/1.03	1980/1.09	2557/1.70	1263/0.33	1254/0.32	2530/1.67	**1144**/0.21
LA17	784	1608/1.05	1759/1.24	1811/1.31	1635/1.08	2450/2.12	960/0.22	1004/0.28	2246/1.86	**898**/0.14
LA18	848	1867/1.20	2305/1.71	1789/1.11	1379/0.62	2479/1.92	1095/0.29	990/0.16	2386/1.81	**990**/0.16
LA19	842	2554/2.03	2296/1.72	2279/1.71	1934/1.29	2807/2.33	1093/0.29	1089/0.29	2680/2.18	**1048**/0.24
LA20	902	1975/1.19	2272/1.52	2163/1.39	1619/0.79	2569/1.84	1170/0.29	1117/0.23	2814/2.12	**974**/0.08
LA22	927	2254/1.43	2695/1.91	2162/1.33	2059/1.22	3475/2.75	1261/0.36	1230/0.32	3399/2.66	**1202**/0.29
LA23	1032	2660/1.58	3223/2.12	3066/1.97	2866/1.77	3804/2.68	1232/0.19	1226/0.18	3811/2.69	**1169**/0.13
LA24	935	2674/1.86	2950/2.15	2550/1.72	2435/1.60	3799/3.06	1199/0.28	1254/0.34	3621/2.87	**1180**/0.26
LA25	977	2727/1.79	3545/2.63	2460/1.52	2574/1.63	3269/2.34	1264/0.29	1305/0.33	3793/2.88	**1240**/0.27
LA26	1218	3956/2.25	3423/1.81	3720/2.05	3326/1.73	4807/2.94	**1485**/0.21	1571/0.29	4874/3.00	1508/0.23
LA28	1216	4027/2.31	3386/1.78	3130/1.57	3276/1.69	4597/2.78	1730/0.42	1724/0.41	4758/2.91	**1537**/0.26
LA30	1355	3323/1.45	3703/1.73	3420/1.52	3081/1.27	4991/2.68	1672/0.23	**1570**/0.16	4616/2.40	1751/0.29
LA32	1850	4799/1.59	6204/2.35	5774/2.12	4661/1.52	7313/2.95	2449/0.32	2224/0.20	7858/3.24	**2160**/0.16

the deviation value for each case via GEP is lower than 0.32 and the total deviation via GEP is 4.57. Both of them are the minimum among all dispatching rules. Especially, the average deviation (0.15) via GEP shows that the prediction value is closer to the lower bound. We also notice that the total deviation and the average deviation of LRM and MWKR take the second place, but the deviation by these two dispatching rules is extremely large sometimes, such as the LA04, LA28. The result of deviation demonstrates that the obtained dispatching rules by GEP is robust.

To measure the self-adaptive capability of the obtained dispatching rule by GEP, we discuss the parameters utilized by all dispatching rules. SPT and LPT only consider the processing time of operations. SSO, LSO, SWKR and MWKR focus on the number of jobs with remaining work. SRM and LRM take account of the jobs with remaining processing time. Different from these dispatching rules, the obtained dispatching rule by GEP comprises not only a variable numerical constant but also the above three factors: the processing time of operations, the number of jobs with remaining work and the jobs with remaining processing time. What is more, it states the functional relationship among these factors. Employing multiple parameters and functional representation enables the obtained dispatching rule by GEP to be more adaptive to the given problem and thus has the potential of finding the best solutions constantly and steadily.

5. **Conclusions.** In this paper, we used the GEP to find the dispatching rule for solving the JSP with the makespan criteria. Firstly, we define four properties including processing time, the number of remaining unscheduled operations, the sum of jobs' remaining processing time and a constant, to state the properties of JSP. Then, the terminal set of GEP takes these four properties as the elements. We also select five basic arithmetic operators as the function set. Secondly, four genetic operators, including selection, mutation, transposition and recombination, are utilized to evolve the individual. Especially, the

random numerical constant and the iterative re-start mechanism are applied to enhancing the search space and the population diversity. Finally, 30 well-known benchmarks are selected to test the performance of the dispatching rule via GEP. The main conclusions are as follows.

(1) The obtained dispatching rule via GEP possesses powerful ability in searching better solutions since 24 cases are superior to other 8 classical dispatching rules.

(2) The obtained dispatching rule by GEP is robust since the deviation is low.

(3) The obtained dispatching rule by GEP owns self-adaptive capability since it comprises a variable numerical constant and three factors about the JSP problem.

Acknowledgment. This research is supported by the National Natural Science Foundation of China under Grant No. 51305311 and No. 51275366, the Specialized Research Fund for the Doctoral Program of Higher Education of China No. 20134219110002 and China Postdoctoral Science Foundation No. 2013M542073.

REFERENCES

[1] Q. Ren and Y. P. Wang, A new hybrid genetic algorithm for job shop scheduling problem, *Computers & Operations Research*, vol.39, pp.2291-2299, 2012.

[2] R. Zhang and C. Wu, Bottleneck machine identification based on optimization for the job shop scheduling problem, *ICIC Express Letters*, vol.2, no.2, pp.175-180, 2008.

[3] J. Li and T. Munehisa, Genetic algorithm using the inhomogeneous Markov chain for job shop scheduling problem, *ICIC Express Letters*, vol.9, no.2, pp.501-509, 2015.

[4] T. C. Chiang and L. C. Fu, Using dispatching rules for job shop scheduling with due date-based objectives, *International Journal of Production Research*, vol.45, no.14, pp.3245-3262, 2007.

[5] J. Huang and G. A. Suer, A dispatching rule-based genetic algorithm for multi-objective job shop scheduling using fuzzy satisfaction levels, *Computers & Industrial Engineering*, vol.86, pp.29-42, 2015.

[6] C. Ferreira, Gene expression programming: A new adaptive algorithm for solving problems, *Complex Systems*, vol.13, no.2, pp.87-129, 2001.

[7] Y. Yang, X. Li, L. Gao and X. Shao, Modeling and impact factors analyzing of energy consumption in CNC face milling using GRASP gene expression programming, *The International Journal of Advanced Manufacturing Technology*, 2015.

[8] L. Nie, L. Gao, P. Li and X. Li, A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates, *Journal of Intelligence Manufacturing*, vol.24, pp.763-774, 2013.

[9] C. Ferreira, Gene expression programming in problem solving, *The 6th Online World Conference on Soft Computing in Industrial Applications*, pp.10-24, 2001.

[10] S. Nguyen, M. Zhang, M. Johnston and K. C. Tan, A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem, *IEEE Trans. Evolutionary Computation*, vol.17, no.5, pp.621-639, 2013.