Evolving Dispatching Rules Using Genetic Programming for Multi-objective Dynamic Job Shop Scheduling with Machine Breakdowns

Salama, Shady Amgad Ahmed Ahmed
Kaihara, Toshiya
Fujii, Nobutada
Kokuryo, Daisuke

(Citation)
Procedia CIRP,104:411-416

(Issue Date)
2021-11-26

(Resource Type)
journal article

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/0100479042

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 104 (2021) 411–416

www.elsevier.com/locate/procedia

54th CIRP Conference on Manufacturing Systems

Evolving Dispatching Rules Using Genetic Programming for Multi-objective Dynamic Job Shop Scheduling with Machine Breakdowns

Salama Shady[a,*], Toshiya Kaihara[a], Nobutada Fujii[a], Daisuke Kokuryo[a]

[a]*Graduate School of System Informatics, Kobe University, Kobe, Hyogo 6578501, Japan*

* Corresponding author. Tel.: +81-78-803-6250; fax: +81-78-803-6391. *E-mail address:* shady.salama@kaede.cs.kobe-u.ac.jp

Abstract

Dynamic Job Shop Scheduling Problem (DJSSP) is an NP-hard problem that has a great impact on production performance in practice. The design of Dispatching Rules (DRs) is very challenging because many shop attributes need to be investigated. Therefore, this paper proposes a Genetic Programming (GP) approach to generate DRs automatically for multi-objective DJSSP considering machine breakdowns. Computational experiments are conducted to compare the GP rule performance with 12 literature rules. The results indicate the superiority of the GP rule in minimizing mean flow time and makespan simultaneously. Finally, the best evolved rule is analyzed, and the significant attributes are extracted.

Keywords: Multi-objective dynamic job shop scheduling; machine breakdowns; genetic programming; dispatching rules.

1. Introduction

The Job Shop Scheduling Problem (JSSP) is one of the most popular manufacturing optimization models used in the industry and has attracted many researchers and practitioners due to its wide applicability and inherent difficulty [1]. Typically, JJSP is categorized into static and dynamic conditions. Different from JSSP under static assumptions where all jobs are ready to process at time zero and their information is always available, Dynamic JSSP (DJSSP) deals with situations where jobs arrive at any time and the processing attributes are not known before their arrival [2]. In the single-objective problems, the superiority of a solution over the others can be easily determined by comparing their objective function values. Whereas in the multi-objective optimization problems, the solution quality is determined by the dominance relationship. Therefore, the main goal of Multi Objective-DJSSP (MO-DJSSP) is to obtain a large number of diverse non-dominated solutions covering the Pareto front of many objectives. However, practical scheduling problems always involve multiple objectives that experience some conflict, and it is inevitable to avoid the stochastic nature of the job shops. Most of the current studies focus on static single-objective JSSPs or aggregate many objectives into a single function and treat the problem as a single-objective optimization [3].

Numerous techniques are proposed in the literature for solving JSSP. These approaches include exact methods to obtain optimal solutions for small instances, such as dynamic programming, and branch-and-bound. In addition, approximate methods are used to achieve near-optimal solutions for both medium and large problem instances, including meta-heuristics, and dispatching rules [4]. Dispatching Rules (DRs) are the most frequent approach used to solve DJSSP due to their flexibility in incorporating domain knowledge, ease of implementation, limited computational budget, scalability, and rapid response to dynamic events [5]. Manually designing DRs is not an easy task due to three main reasons. It is challenging to identify the important factors from a large set of job and machine attributes and to understand their impact on the rule's overall performance. Also, the performance of the developed rules is assessed through extensive simulation runs to choose the best DRs for specific job shop settings. This trial-and-error

10.1016/j.procir.2021.11.069

cycle requires a considerable amount of time, experience, and code effort. Also, most of the literature rules experience a myopic nature by only considering local features (the information available in the current machine) and neglecting global ones (information available in subsequent machines) that leads to poor results compared to global optimization methods. Lastly, the literature rules are developed to address specific job shop environment using a certain objective and usually perform poorly under other system settings or objectives [6].

With the advances in computing power, several Artificial Intelligence (AI) methods are proposed in order to select or generate heuristics to efficiently solve optimization problems called Hyper-Heuristics (HH). HH refers to a high-level automated search methodology that explores a search space of low-level heuristics rather than the search space of the underlying problem [7]. Recently, Genetic Programming (GP) has emerged as a promising hyper-heuristic approach for the automated design of dispatching rules in the DJSSP. GP offers several advantages over other AI methods, including a powerful search mechanism that explores both the structure and parameters of evolved heuristics, the use of flexible representation using a variable-length encoding scheme, and the evolved heuristics can be partially interpreted and executed directly [8]. Tay and Ho [1] is the first work that aimed to solve flexible MO-DJSSP with respect to minimum makespan, mean tardiness, and mean flow time objectives. In this study, the objective function is constructed by combining the objectives into a linear weighted sum in which all the objectives have the same priority. As mentioned in [9], the major drawback of this approach is that it requires domain knowledge about the search space to adjust the weight for each objective, which is not achievable in most cases. The authors in [10] proposed a cooperative coevolution GP method to evolve non-dominated dispatching rules and due date assignment rules for MO-DJJSP. The experimental results showed that the Pareto front rules outperform various combinations of literature rules under different dynamic scenarios regarding three objective functions. In a later study [3], the authors developed a MO-GPHH technique to generate DRs for the DJSSP while neglecting the impact of machine malfunctions. The harmonic distance-based multi-objective evolutionary algorithm is adopted to explore the Pareto front of evolved non-dominated rules. Five objectives are investigated including mean flowtime, maximum flowtime, percentage of tardy jobs, mean tardiness, and maximum tardiness to assess the performance of the rules. The same objectives are adopted in [6] where a MO-GPHH approach is suggested that integrates GP with NSGA-III to evolve trade-off dispatching rules under static conditions. The proposed algorithm showed superior performance by examining its performance using the Taillard static job-shop benchmark instances.

Although, some efforts have been made to adopt the GPHH approach to evolve non-dominated dispatching rules for Multi-objective DJSSP. There are some limitations in the existing literature. The articles that used GP to generate DRs for the MO-DJSSP usually assume that all machines are always available by neglecting breakdowns [3]. Machine breakdown is one of the most common and challenging issues in production engineering. Moreover, production interruptions can greatly affect the performance of scheduling algorithms or even make the obtained schedules inapplicable if not considered during the development phase. Although including machine failures in the DJSSP makes the obtained results more practical, the problem becomes more challenging due to the large search space [11]. Another limitation is that the authors usually conclude that the GP rules outperform literature rules without providing further analysis of the rule's structure to determine the important features that lead to this superior results, and which features have a negligible impact on the considered objectives [6].

In order to address the aforementioned literature gaps, this paper proposes a GPHH approach to automatically generate non-dominated DRs for DJSSP with machine breakdowns while minimizing the jobs mean flowtime and makespan. After that, the evolved rules are compared with 12 literature rules over 10 testing scenarios generated by varying the breakdown levels. Moreover, the structure of the best-evolved rule and the distributions of attributes across GP generations are analyzed to gain more insight into the behavior of the evolved rules. To the best of our knowledge, no paper has investigated the adaptability of the GPHH approach in solving MO-DJSSP while considering machine failures. Therefore, the motivation behind this work is to integrate GP with NSGA-II to evolve DRs offline across a set of training instances representing many breakdown situations and use them online for fast application. The remainder of the paper proceeds as follows. The next section provides a brief description of the DJSSP and the developed simulation model. In Section 3, the proposed approach is illustrated in detail. The numerical experiments are given in Section 4. The results obtained from comparing non-dominated DRs with the literature rules are shown in Section 5. Finally, conclusions are presented in Section 6.

2. Problem formulation

The MO-JSSP is an NP-hard combinatorial optimization problem that can be mathematically formulated as follows. There is a set of N jobs $(j = 1, 2, \dots, N)$ has to be processed on a set of M machines $(i = 1, 2, \dots, M)$ using a predefined sequence. The processing of job j on a machine i is denoted as operation O_{ij} with a processing time equals P_{ij}. For static case, the release time of all jobs is zero and due date is deterministic. While for dynamic MO-JSSP, the release time, and due date of job j is R_j, and D_j, respectively, and are randomly distributed. The goal is to schedule each operation on its compatible machine while minimizing a number of objectives $f = (f_1, \dots, F_D)$ where D stands for the number of objectives. Let the variable S_{ij} denotes the starting time of operation O_{ij}. Also, let set O denotes the set of all operations O_{ij} and set A defines the set of all precedence constraints i.e. $O_{ij} \rightarrow O_{hj}$ which means that job j needs to be processed on machine i before machine h. Finally, let C_j denotes the completion time of job j and C is a set of completed jobs in the static JSSP or a set of jobs recorded after the warm-up period of the Discrete Event Simulation (DES) model in the DJSSP.

Minimize f,

Subject to, $\quad S_{hj} - S_{ij} \geq P_{ij}, \quad \forall O_{ij} \rightarrow O_{hj} \in A, \qquad (1)$

$$S_{ij} - S_{ik} \geq P_{ik} \vee S_{ik} - S_{ij} \geq P_{ij} \quad \forall O_{ik}, O_{ij}, \; i = 1, \dots, M \qquad (2)$$

$$S_{ij} \geq 0, \qquad \forall O_{ij} \in O \qquad (3)$$

Constraint (1) ensures that operation O_{hj} cannot start before O_{ij} is completed. The disjunctive constraint (2) ensures the precedence relationship among operations of different jobs on the same machine. Constraint (3) ensures that the starting time of all operations are non-negative integers. The objective functions in this paper consists of mean flowtime (F_{mean}) and makespan (C_{max}) which are formulated as follows:

$$F_{mean} = \frac{\sum_{j \in C} C_j - R_j}{|C|} \qquad (4)$$

$$C_{max} = max_{j \in C}\{C_j\} \qquad (5)$$

The goal is to evolve DRs that can build the complete schedule. Then, the quality of the schedules created with these rules will be evaluated with respect to the predefined objectives. Given two schedules Δ_1 and Δ_2, it is said that Δ_1 dominates Δ_2 if and only if:

$\forall i, 1 \leq i \leq D,\ f_i(\Delta_1) \leq f_i(\Delta_2)$ and $\exists i,\ f_i(\Delta_1) < f_i(\Delta_2)$

DES is the most common technique for estimating the performance of dispatching rules in dynamic scheduling problems. Therefore, a DES model is developed for a frequently used balanced job shop where all machines have the same utilization levels in long simulation runs [3]. Here are the simulation configurations:

- The job shop contains 10 machines.
- Job arrivals follow the Poisson distribution.
- Each job has from 2 to 10 operations.
- The job routing is determined randomly as each machine has the same probability to be selected.
- The job due date is assigned using the total work content method [8] with a tightness factor of 3.

Also, the breakdown events are created using the data provided in [12] where inter-breakdown times and machine repair times follow an exponential distribution. All machines have the same Mean Time To Repair (MTTR) and Mean Time Between Failures (MTBF). The breakdown level (BL= MTTR / (MTTR + MTBF)) represents the percentage of time that the machine is expected to breakdown throughout the simulation run. The machine breakdown configurations investigated in this paper are $MTTR = \{2\bar{p}, 5\bar{p}, and\ 10\bar{p}\}$ where $\bar{p}$ indicates the mean total processing time of a job and $BL = \{5\%, 10\%, and\ 15\%\}$. In each simulation replication, we begin with an empty shop, and the interval from the start until the arrival of the 500th job is considered as the warm-up period. The model terminates after completing 2500 jobs and the statistics are collected from the 500th job to the next finished 2500 jobs. Although a small number of replications is enough during the training phase, a large number of replications is needed to assess the steady-state performance of rules in the testing scenarios [9]. As depicted in Table 1, four scenarios are employed to train the evolved rules and 10 simulation scenarios with 30 replications for each scenario are used to assess their performance with respect to mean flow time and makespan objectives. The tuple (A, B) represents the system settings by defining a breakdown level is A% and a MTTR is $B \times$ mean processing time. The scenario "S" represents the standard model without considering interruptions on the shop floor. The literature rules are shown in Table 2, and for more details please refer to [5].

Table 1. Parameter settings of the training and testing scenarios

Parameter	Training	Testing
Processing time	Uniform [1, 49]	Uniform [1, 49]
Shop utilization	85 %	85 %
Breakdown scenarios	{S, (5, 2), (10, 5), (15, 10)}	{S, (5, 2), (5, 5), (5, 10), (10, 2), (10, 5), (10, 10), (15, 2), (15, 5), (15, 10)}

Table 2. Benchmark dispatching rules

No.	Abbreviation	Description
1	SPT	Shortest processing time
2	FIFO	First in first out
3	WINQ	Work in the next queue
4	MOD	Modified operation due date
5	OPFSLK	Operational flow slack rule
6	CR + PT	Critical ratio plus processing time
7	PT + WT	Processing time plus waiting time
8	SOPN	Slack per remaining operation
9	PT + WINQ	Processing time plus WINQ
10	2PT + WINQ + NPT	Double PT + WINQ + processing time of next operation
11	ATC	Apparent tardiness cost rule
12	WR + PT	Work remaining plus processing time

3. MOGP approach for DJSSP with machine breakdowns

NSGA-II is one of the most popular Pareto-based multi-objective evolutionary algorithms that uses an efficient ranking scheme called non-dominated sorting and adopts a diversity-preserving mechanism (crowding distance) to obtain a set of well-spread Pareto-optimal solutions [6]. In [13], the authors employed NSGA-II and SPEA2 to evolve DRs for multi-objective DJSSP under a single simulation scenario without considering machine breakdowns. They reported that NSGA-II performs better the SPEA2. Therefore, it is expected that the integration between the GP and NSGA-II can lead to superior results compared to the use of the GP approach solely. The proposed framework consists of two main modules: the GPHH reasoning module and the DES evaluation module. In the evolutionary part, GP is merged with NSGA-II as a learning mechanism to evolve DRs. Then, the performance of generated rules is assessed using the DES model. As depicted in Fig. 1., the algorithm starts by generated a diverse population of DRs using the ramped-half-and-half method and predefined function and terminal sets. In the training stage, the fitness values of the evolved heuristics are evaluated by applying each rule to the training instances. These instances represent the job shop under different breakdown scenarios. Then, the objective values are normalized to reduce any bias towards a specific instance i.e., the objective values for the schedule generated by an individual for a specific instance are divided by the objective values obtained by applying a reference rule (the SPT rule in this paper) to the same instance. Then, the rule's fitness is measured by the average value of the specific objective across all training scenarios. After adding offspring to the current population, the NSGA-II is employed to assign ranks and crowding distance to every individual. The individual with

better rank and smaller crowding distance is selected in the new population. If the maximum generation is not reached, an offspring is generated by genetic operations, otherwise, the algorithm returns the set of non-dominated DRs.

Inputs: A simulation model S and a training set I_{train}
Outputs: The pareto front of non-dominated rules P^*
Initialize population $P \leftarrow \{X_1, X_2, \dots, X_{popsize}\}$ and $g \leftarrow 0$
while $g < g_{max}$ **do**
  Apply genetic operations to P to generate offspring Q
  foreach rule $X_i \in Q$ **do**
    foreach I in I_{train} **do**
      Construct a schedule $\Delta(X_i,I)$ using S
      Calculate the objective values $f(\Delta(X_i,I))$
      Calculate the normalized obj. values $f_n(\Delta(X_i,I))$
    end for
    $f(X_i) \leftarrow \frac{1}{|I_{train}|}\sum_{I \in I_{train}} f_n(\Delta(X_i,I))$
  end for
  $T \leftarrow P \cup Q$
  Assign ranks and crowding distance for the rules in T
  $P \leftarrow$ NSGA-II $select(T)$
  $g \leftarrow g + 1$
end while
return The non-dominated individuals (first front) $P^* \subseteq P$

Fig. 1. The pseudocode of the proposed algorithm

4. Experimental design

The GP tree representation is used to represent DRs. Table 3 shows the terminal set and function set used. The terminal set consists of two parts, the upper part represents the local attributes and the other represents the global ones. For the function set, three basic arithmetic operators and a protected division operator are used. Also, the binary “min” and “max” functions as well as “if” are used. Table 4 shows the parameter settings of the MOGP-NSGA-II. Crossover and mutation operations are used. The crossover operator creates children by randomly combining subtrees from two parents. Moreover, a mutation operator selects a node in an individual and replaces the subtree at that node with a randomly generated tree.

Table 3. The GP terminal and function sets

Node name	Description
JR	The release date of job(j)
OR	The operation release date
RO	Number of remaining operation in the job
WR	Work remaining of the job
PT	Operation processing time
CT	Machine ready time (current time)
WT	Operation waiting time
Npt	Processing time of the next operation
WINQ	Work in the next queue
Apt	Average processing time of queued jobs
Function set	+, −, ×, /, if, min, max, abs

Table 4. Parameter settings of the MOGP-NSGA-II

Parameter	Value
Population size	1000
Generations	50
Crossover rate	90 %
Mutation rate	10 %

5. Results and analysis

There are a variety of performance measures proposed for evaluating MOE algorithms. The hypervolume is used to measure the size of the objective space dominated by the obtained rules, and to check whether the algorithm suffers from a premature convergence or not. By evaluating the hypervolume of the set of non-dominated solutions obtained across the 50 generations, the performance of the algorithm was improving significantly until it slightly plateaued from the 36th generation, as depicted in Fig. 2. After the training is completed, 46 non-dominated DRs are evolved at the first Pareto front. The evolved rules are tested across the 10 testing scenarios. Due to space limitations and similar findings are appeared in the other scenarios, only scenarios 1, 2, 9, and 10 are presented in this paper. As shown in Fig. 3., the nondominated DRs evolved by MOGP-NSGA-II outperform the literature rules for both objectives under all scenarios. The gap between the performance of the evolved rules and literature rules, the margin between the nearest blue and red points, widens as the machine breakdowns occur more frequently and last longer as illustrated by the length of the red double arrows in scenarios 1 and 2 compared to scenarios 9 and 10. For the sake of convenience, the rank shown between brackets represent the rule's rank among literature rules. Moreover, the green arrow next to the DRs represents that this rule performs well on the objective the arrow is pointing at. In contrast, the black arrow shows that this rule performs poorly on the objective indicated by the arrow.

For the makespan objective, the SOPN rule (no. 8) obtains a poor performance (10^{th}) in minimizing makespan when breakdowns level is low (scenario 1) while its performance increases (1^{st}) when breakdown events become more significant (scenario 10). In contrast, the OPFSLK rule (no. 5) obtains a superior performance (2^{nd}) in minimizing makespan when machine failures are neglectable while its performance degrades when breakdown events become more significant (11^{th}). Regarding the mean flowtime objective, the literature rules showed a kind of stable performance across scenarios as the performance of each rule does not change greatly when the breakdown level varies. One of the reasons could be that the mean flow time objective is more robust than the makespan objective under different scenarios. These findings support the first claim that the performance of human-made rules varies greatly when machine breakdowns occur more frequently and longer, reflecting the great impact of machine failures on the rule's efficiency. By comparing the average normalized performance of the literature rules for each objective, some observations are revealed. Although, 2*PT+WINQ+Npt rule (no. 10) is the best literature rule (1^{st}) in minimizing the mean flow time, its average performance diminishes in the makespan

objective (5th), as shown in Fig. 4. Also, the CR + PT rule (no. 6) shows a good performance in minimizing the mean flow time (4th), but its performance deteriorates in reducing the makespan (9th). These findings support our second claim that literature rules struggle to obtain consistent performance under conflicting objectives. Also, the performance of human-made rules varies greatly when the system configurations change.

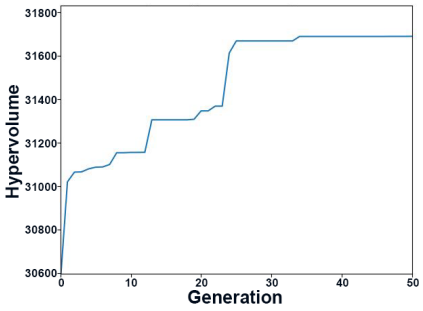

Fig. 2. Evolution of the hypervolume value over the generations

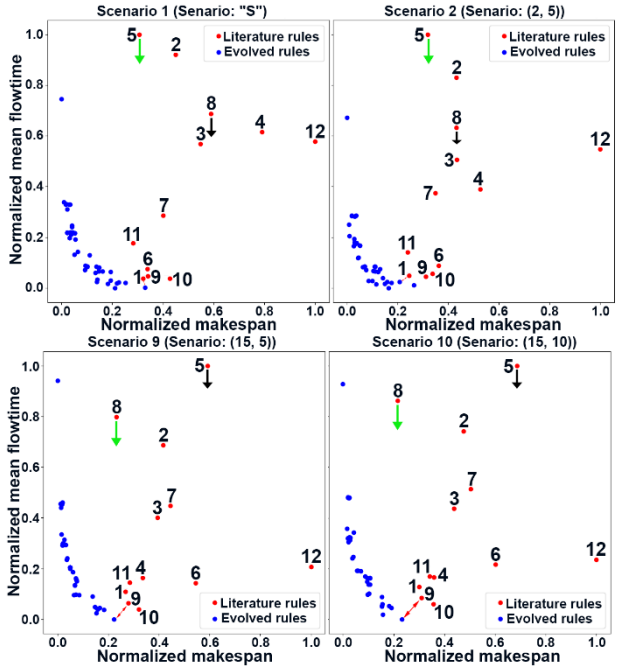

Fig. 3. Performance of the evolved rules compared with the literature rules

Comparing the average performance through the 30 replications is not enough to drive clear conclusions regarding the performance of the evolved rules and the literature rules due to some limitations as follows. Since DJSSP is a stochastic problem, the performance of the DRs varies between different replications. Also, the dominance relationship between the compared rules must be investigated when considering multi-objectives. Therefore, the proposed statistical comparison test in [3] is adopted to compare the evolved rules with the literature rules and check whether one method significantly dominates the other in the 30 replications. Fig.5. shows the number of times the evolved rules dominated the literature rules in each scenario. The maximum value for each scenario equals 12 X 46 = 552 which means that all the 46 evolved rules have dominated all the 12 literature rules in this scenario. The evolved rules dominated literature rules through all scenarios by about 87% on average of all runs. A histogram of the percentage of times the evolved rules have dominated the literature rules in the 552 replications is shown in Fig. 6. As we can see that the distribution of the dominance percentage skewed to the right, indicating a large proportion of the evolved rules significantly suppresses the literature rules. For instance, 25 evolved rules have dominated the literature rules in the range between 83% and 100% of the total replications.

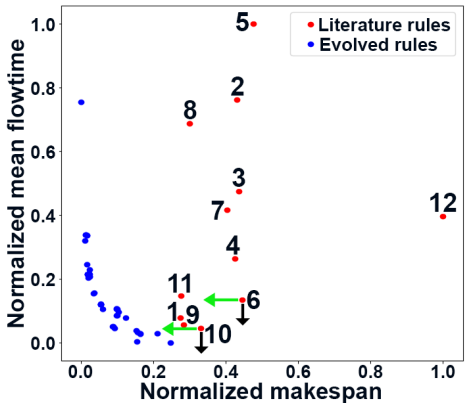

Fig. 4. Average performance through the 10 testing scenarios

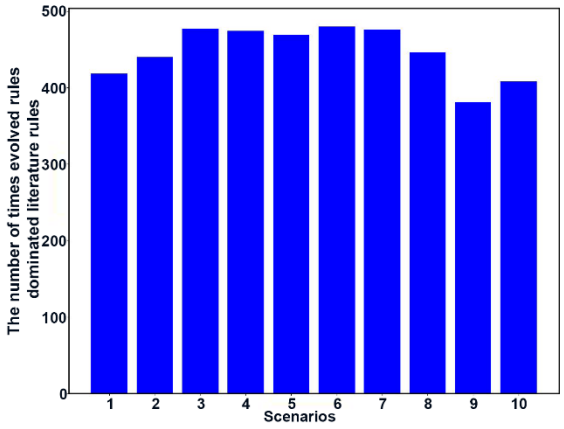

Fig. 5. The number of times the GP rules dominated the literature rules

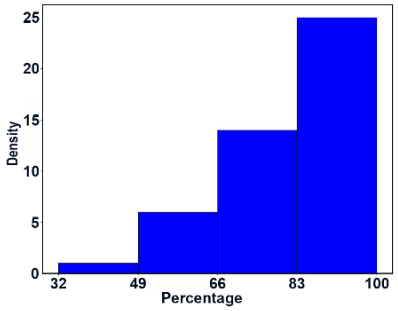

Fig. 6. The percentage of times that the evolved rules dominated the literature rules

An important criterion in evaluating the performance of the DR is its ability to maintain consistent performance across different simulation settings, known as robustness. The author in [3] proposed a method to roughly estimate the robustness of

the evolved rules. Using this method, the role robustness value ranges between 0 and 1, where "1" indicating the most optimistic case when the rule gets the same results in all scenarios, and "0" represents the worst case when the rule is very sensitive to any change in breakdown levels. As shown in Fig. 7., the distribution of robustness values lies in the range of 0.87 and 1 with most of the non-dominated rules (22 rules) having high robustness values between 0.92 and 0.95.

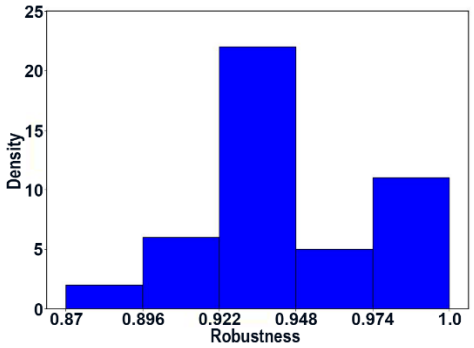

Fig. 7. Robustness of the evolved rules

From the 46 evolved rules obtained, one rule is selected for further analysis. This rule achieved outstanding results in both objectives while maintaining high robustness across the 10 scenarios. The rule's terminals are sorted in descending order based on the number of occurrences to obtain an estimate of their importance. The top 4 features used based on the number of occurrences are, OR, JR, Apr, and PT terminals. Moreover, WR terminal did not appear in the chosen rule and WT terminal rarely appeared. Also, all the global attributes are presented demonstrating the importance of global attributes in alleviating the myopic nature of DRs (the third claim). To assess the importance of each terminal not only in the performance of the best rule but also in the general performance of all GP individuals, terminals analysis across generations is performed. For each terminal, the number of times it has appeared in the 1000 individuals every 20 generations is tracked. As shown in Fig. 8. although all the terminals are distributed evenly in the first generation, the distribution varied greatly in the later generations with JR, RO, PT, and WINQ terminals being the commonly used across generations while WR and WT terminals are rarely used. These findings are consistent with the results obtained from analyzing the structure of the chosen rule.

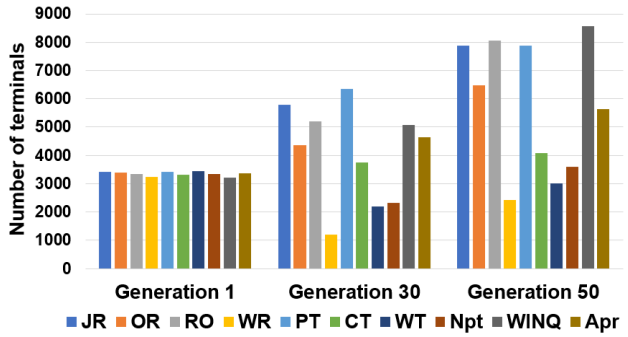

Fig. 8. Terminal distribution across generations

6. Conclusions

This paper proposes an integration of the GP approach with the NSGA-II algorithm to automatically generate dispatching rules for the DJSSP with machine breakdowns. To assess the effectiveness of the proposed approach, the evolved rules are compared with 12 literature rules in minimizing the mean flow time and makespan of jobs on 10 simulation scenarios. The results showed that the generated rules outperform the human-made rules in all scenarios especially when the machine breakdowns happen frequently and last longer. After analyzing the robustness of the evolved rules, we demonstrated that the rules could maintain consistent performance for unseen scenarios. Finally, valuable insights into the importance of GP terminals were gained by analyzing the structure of one chosen rule and the distribution of terminals across generations.

References

[1] Tay, J.C. and Ho, N.B., 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering*, 54(3), pp.453-473.

[2] Ouelhadj, D. and Petrovic, S., 2009. A survey of dynamic scheduling in manufacturing systems. *Journal of scheduling*, 12(4), pp.417-431.

[3] Nguyen, S., Zhang, M., Johnston, M. and Tan, K.C., 2013. Dynamic multi-objective job shop scheduling: A genetic programming approach. In *Automated scheduling and planning*, pp. 251-282. Springer, Berlin, Heidelberg.

[4] M. Pinedo and H. Khosrow, 2012. Scheduling: theory, algorithms and systems development. In *Springer*, Berlin, Heidelberg, Springer US.

[5] Sels, V., Gheysen, N. and Vanhoucke, M., 2012. A comparison of priority rules for the job shop scheduling problem under different flow time-and tardiness-related objective functions. *International Journal of Production Research*, 50(15), pp.4255-4270.

[6] Masood, A., Mei, Y., Chen, G. and Zhang, M., 2016, July. Many-objective genetic programming for job-shop scheduling. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 209-216. IEEE.

[7] Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. and Qu, R., 2013. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12), pp.1695-1724.

[8] Shady, S., Kaihara, T., Fujii, N. and Kokuryo, D., 2020, August. Automatic Design of Dispatching Rules with Genetic Programming for Dynamic Job Shop Scheduling. In *IFIP International Conference on Advances in Production Management Systems*, pp. 399-407. Springer, Cham.

[9] Hildebrandt, T., Heger, J. and Scholz-Reiter, B., 2010, July. Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 257-264.

[10] Nguyen, S., Zhang, M., Johnston, M. and Tan, K.C., 2012, June. A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems. In 2012 *IEEE Congress on Evolutionary Computation*, pp. 1-8. IEEE.

[11] Shady, S., Kaihara, T., Fujii, N. and Kokuryo, D., 2020. A Proposal on Dispatching Rule Generation Mechanism Using GP for Dynamic Job Shop Scheduling with Machine Breakdowns. In *Scheduling Symposium 2020*, pp. 155–160, Japan.

[12] Holthaus, O., 1999. Scheduling in job shops with machine breakdowns: an experimental study. *Computers & industrial engineering*, 36(1), pp.137-162.

[13] Nguyen, S., Zhang, M. and Tan, K.C., 2015, May. Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2781-2788. IEEE.