Available online at www.sciencedirect.com

ScienceDirect

Computers & Industrial Engineering 54 (2008) 453–473

computers & industrial engineering

www.elsevier.com/locate/dsw

Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems

Joc Cing Tay [*], Nhu Binh Ho

Evolutionary and Complex Systems Program, School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore

Received 29 March 2006; received in revised form 27 June 2007; accepted 29 August 2007
Available online 7 September 2007

Abstract

We solve the multi-objective flexible job-shop problems by using dispatching rules discovered through genetic programming. While Simple Priority Rules have been widely applied in practice, their efficacy remains poor due to lack of a global view. Composite dispatching rules have been shown to be more effective as they are constructed through human experience. In this paper, we evaluate and employ suitable parameter and operator spaces for evolving composite dispatching rules using genetic programming, with an aim towards greater scalability and flexibility. Experimental results show that composite dispatching rules generated by our genetic programming framework outperforms the single dispatching rules and composite dispatching rules selected from literature over five large validation sets with respect to minimum makespan, mean tardiness, and mean flow time objectives. Further results on sensitivity to changes (in coefficient values and terminals among the evolved rules) indicate that their designs are robust.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Flexible job shop; Production scheduling; Genetic programming; Dispatching rules

1. Introduction

In today's highly competitive marketplace, a high level of delivery performance has become necessary to satisfy customers. Due to market trends, product orders of low volume, high variety types have been increasing in demand. Hoitomt, Luh, and Pattipati (1993) mentions that these products comprise between 50% and 75% of all manufactured components, thereby making schedule optimization an indispensable step in the overall manufacturing process.

The job-shop scheduling problem (JSP) is one of the most popular manufacturing optimization models used in practice (Jain & Meeran, 1998). It has attracted many researchers due to its wide applicability and inherent difficulty (Carlier & Pinson, 1999; Kolonko, 1999; Nowicki & Smutnicki, 1996; Yamada & Nakano, 1996). It is also well known that the JSP is NP-hard (Garey, Johnson, & Sethi, 1996), hence

[*] Corresponding author. Tel.: +65 6790 6266.
E-mail address: asjctay@ntu.edu.sg (J.C. Tay).

general, deterministic methods of search are inefficient as the problem size grows larger. The *nxm* classical JSP involves *n* jobs and *m* machines. Each job is to be processed on each machine in a predefined sequence and each machine processes only one job at a time. In practice, the shop-floor setup typically consists of multiple copies of the most critical machines so that bottlenecks due to long operations or busy machines can be reduced. As such, an operation may be processed on more than one machine having the same function. This leads to a more complex problem known as the flexible job-shop scheduling problem (FJSP). The extension involves two tasks; assignment of an operation to an appropriate machine and sequencing the operations on each machine. In addition, for complex manufacturing systems, a job can typically visit a machine more than once (known as *recirculation*). These three features of the FJSP significantly increase the complexity of finding even approximately optimal solutions (Pinedo & Chao, 1999, chap. 3). Furthermore, instead of considering only a single objective, most scheduling problems in practice involve simultaneous optimization of several competing objectives. Therefore, in order to tackle the FJSP problems found in practice, efficient optimization strategies are required to deal with both multiple objectives and exponential search space complexity.

The classical JSP and FJSP (in single or multi-objectives) have been solved by many stochastic local search methods, such as Simulated Annealing (Kolonko, 1999), Tabu Search (Brandimarte, 1993; Mastrolilli & Gambardella, 2000; Nowicki & Smutnicki, 1996), Genetic Algorithms (Ho & Tay, 2004; Kacem, Hammadi, & Borne, 2002a, Kacem, Hammadi, & Borne, 2002b; Tay & Wibowo, 2004) or Artificial Immune Systems (Ong, Tay, & Kwoh, 2005). The reported results of applying them show that good approximations of optimality can be found, albeit at the expense of huge computational cost, particularly when the problem size is large. In practice, dispatching rules have been applied to avoid these costs (Blackstone, Phillips, & Hogg, 1982; Oliver & Chandrasekharan, 1997; Panwalkar & Wafik, 1977). Although the qualities of solutions produced by dispatching rules are no better than the local search methods, they are the more frequently applied technique due to their ease of implementation and their low time complexities. Whenever a machine is available, a priority-based dispatching rule inspects the awaiting jobs and selects one with the highest priority to be processed next. Recently, the introduction of composite dispatching rules (CDR) have been increasingly investigated by the some researchers (Jayamohan & Rajendran, 2004; John & Xiaoming, 2004), but typically only for classical JSPs. These rules are the heuristic combination of single dispatching rules that aim to inherit the advantages of the former. Empirically, results show that with careful combination, the composite dispatching rules will perform better than the single ones with regards to the quality of schedules. However, little is yet known about the robustness of such human-made designs to changes in the parameter and operator spaces.

In this paper, we investigate the potential use of genetic programming (GP) for evolving effective and robust composite dispatching rules for solving the multi-objective FJSP. Although there are many multi-objective approaches for searching continuous and/or discrete search spaces (Coello, 2005), a survey of the research literature shows that there are few previous works on dispatching rules that satisfy multiple objectives simultaneously (Barman, 1997; Jayamohan & Rajendran, 2004; Oliver & Chandrasekharan, 1997). The purpose of this research is to find effective and robust CDRs that perform better than the dispatching rules presented in literature for solving the multi-objective FJSP problems. By using a wide training data set, we believe that the evolved CDRs can be applied directly in practice without further modifications. Furthermore, these CDRs can be used for population generation in other local search methods for solving FJSPs, such as Genetic Algorithms (Ho & Tay, 2004; Tay & Wibowo, 2004) or Artificial Immune Systems (Ong et al., 2005).

The remainder of this paper is organized as follows. Section 2 gives the formal definition of the multi-objective FJSP. Section 3 gives an overview of GP, reviews recent works for solving the JSP and FJSP using dispatching rules, as well as the development of multi-objective Evolutionary Algorithms in literature. Section 4 describes our proposed GP framework for evolving CDRs. Section 5 presents the design of experiments for performance evaluation while Section 6 analyzes the performance results of using the evolved CDRs obtained with GP in comparison to the other well-known dispatching rules such as EDD (Earliest Due Date) and SPT (Shortest Processing Time) for solving the multi-objective FJSPs. We present our results by evaluating the components of effective CDRs through single-objective optimizations, and then evaluating the evolved CDRs for multiple objectives simultaneously. Finally, Section 7 gives some concluding remarks and directions for future work.

2. Problem definition

Similar to the classical JSP, solving the FJSP requires the optimal assignment of each operation of each job to a machine with known starting and ending times. However, the task is more challenging than the classical one because it requires a proper selection of a machine from a set of machines to process each operation of each job. Furthermore, if a job is allowed to recirculate, this will significantly increase the complexity of the system (Pinedo, 2002, chap. 2). The multi-objective FJSP is formulated as follows:

- Let $J = \{J_i\}_{1 \leqslant i \leqslant n}$, indexed i, be a set of n jobs to be scheduled.
- Each job J_i consists of a predetermined sequence of operations. Let $O_{i,j}$ be operation j of J_i.
- Let $M = \{M_k\}_{1 \leqslant k \leqslant m}$, indexed k, be a set of m machines.
- Each machine can process only one operation at a time.
- Each operation $O_{i,j}$ can be processed without interruption on one of a set of machines M_k in a given set $\mu_{i,j} \subset M$ with $p_{i,j,k}$ time units.

Let r_i, C_i, and d_i be the release date, completion time and the due date of the job J_i, respectively. The objectives of this problem are to find a schedule that simultaneously satisfies (in a pareto-sense):

- Minimization of makespan: $F_1 = \max\{C_i \mid i = 1, \ldots, n.\}$
- Minimization of mean tardiness: $F_2 = \frac{\sum_{i=1}^{n} \max\{C_i - d_i, 0\}}{n}$
- Minimization of mean flow time: $F_3 = \frac{\sum_{i=1}^{n} (C_i - r_i)}{n}$

The three objectives in the definition are the typical objectives in production scheduling (Jayamohan & Rajendran, 2004) that frequently trade-off against each other. For instance, a schedule that satisfies the minimization of makespan can lead to a larger mean tardiness and mean flow times. Another example is when a job is scheduled as early as possible to minimize tardiness, this may cause an increase in the mean flow time. Therefore, the ideal schedule is a schedule that produces trade-offs among different objectives. In this study, the objective function is constructed by combining the different objectives into a weighted sum where all the objectives have the same priority. It can be defined as follows:

$$F = \frac{1}{3}F_1 + \frac{1}{3}F_2 + \frac{1}{3}F_3$$

The multi-objective FJSP can also be considered to be a Multi-Purpose Machine (MPM) job-shop (Brucker, Jurisch, & Krämer, 1997). Using the $\alpha|\beta|\gamma$ notation of Graham, Lawler, Lenstra, and Kan (1979), the problem we wish to solve can be denoted by

$$J\ MPM \mid prec\ r_i d_i \mid C_{\max}\overline{T}\overline{F}$$

where J denotes a job-shop problem, *MPM* denotes a multi-purpose machine, *prec* represents a set of independent *chains* while r_i and d_i represents *release date* and *due date* given to each job J_i, respectively, $C_{\max}$ represents makespan, $\overline{T}$ represents mean tardiness and finally $\overline{F}$ represents mean flow time.

In this paper, we shall assume that:

- All machines are available at time 0.
- Each job has its own release date and due date.
- The order of operations for each job is predefined and invariant.
- A machine can execute only one operation at a time.

3. Previous works

3.1. Multiple objective optimization for job-shop scheduling

Evolutionary Algorithms (EAs) have been used widely to solve multi-objective optimization problems. Generally, they can be classified into three approaches: population-based, pareto-based, and aggregation function (Coello, 2005). Population-based approaches, such as VEGA (Schaffer, 1985), are based on the division of the current population into s sub-populations where s is the number of objectives. At each generation, s sub-populations are generated by performing proportional selection according to each objective. The crossover and mutation operators can be applied as usual to this new population. The drawback of this approach is to focus only on one objective (per sub-population) at a time. Therefore, the results that are good for more than one objective may be discarded before combining together to form new population. Pareto-based approaches use the concept of domination to find the optimal results. It is described as follows: a solution X dominates a solution Y if the solution X has at least one objective that is better than the corresponding objective in the solution Y. The image of the Pareto set under the objective function is called the Pareto front. There are many Pareto-based approaches that are presented in literature. The more popular ones are NSGA-II (Deb, Pratap, Agarwal, & Meyarivan, 2002) and SPEA (Zitzler & Thiele, 1999) that depend on elitist selection, fitness sharing, and Pareto ranking. Elitist selection preserves the elite individuals from the last generation, fitness sharing degrades the fitness values of all competing members of a niche while Pareto ranking uses dominance concepts for selection to eliminate inferior individuals. The challenge of Pareto-based approaches is to keep the diversity of the population towards the Pareto front of the problems. The aggregation function approach, such as the application of Memetic Algorithms for solving flow shop problems in Ishibuchi, Yoshida, and Murata (2003), is to combine all objectives into a single one using appropriate combination operators. Due to its simplicity and applicability to GP-based evolution, we employ this approach to solve the multi-objective FJSP with the same priority for each objective (see Section 2). The difference between this work and previous approaches on multi-objective evolutionary optimization is that the latter samples the search space directly without prior training (or with at most incremental training). This is computationally expensive. Our approach is to use evolutionary-based learning to discover good dispatching rules and then apply them to schedule jobs. The cost incurred by this training phase is offset by the constant time complexity of scheduling jobs using the evolved CDRs. We will show that the evolved CDRs can obtain acceptably good results.

3.2. Types of dispatching rules

Dispatching rules have received much attention from researchers over the past decades (Blackstone et al., 1982; Oliver & Chandrasekharan, 1997; Panwalkar & Wafik, 1977). In general, whenever a machine is freed, a job with the highest priority in the queue is selected to be processed on a machine or work center. A comprehensive survey on dispatching rules is by Panwalkar and Wafik (1977) and Blackstone et al. (1982). Depending on the specification of each rule, it can be classified into (Panwalkar & Wafik, 1977):

- Simple Priority Rules,
- CDRs,
- weighted priority indexes, and
- heuristic scheduling rules.

Simple Priority Rules (SPR) are usually based on a single-objective function. They usually involve only one model parameter, such as processing time, due date, number of operations or arrival time. The Shortest Processing Time (SPT) rule is an example of a SPR. When a machine is freed, the next job with the shortest time in the queue will be removed for processing. SPT has been found to be the best rule for minimizing the mean flow time and number of tardy jobs simultaneously (Oliver & Chandrasekharan, 1997). The Earliest Due Date (EDD) rule is another example of a SPR where the next job to be processed is the one with the earliest due date. Unfortunately, no SPR performs well across every performance measure such as tardiness or flow time (Barman, 1997). To overcome this limitation, CDRs have been studied to combine good features from such SPRs.

There are two kinds of CDRs presented in literature; the first type involves deploying a select number of SPRs at different machines or work centers. Each machine or work center employs a single rule. When a job enters a specific machine or work center, it is processed by the SPR that is preselected for that machine or work center. For instance, Barman (1997) applied three different SPRs to solve the flow shop problem corresponding to three work centers. Experimental results show that it obtains better results than a single SPR that is common to all three machines. However, this approach may not be suitable for a shop floor with large number of machines or work centers, and the best independent distribution of single SPRs is difficult to predetermine. Furthermore, it still has the limitation of a localized view. The second type involves applying the composition of several SPRs (otherwise known as a CDR) to evaluate the priorities of jobs on the queue (Oliver & Chandrasekharan, 1997). The latter type is executed similarly to SPRs; when a machine is free, this CDR evaluates the queue and then selects the job with the highest priority. For example, Oliver and Chandrasekharan (1997) present five CDRs for solving the JSP. Their results indicate that CDRs are more effective compared to individual SPRs in that the former inherits the simplicity of SPRs while achieving some scalability as the number of machines increase. Furthermore, if well designed, CDRs can solve realistic problems with multiple objectives (Pinedo & Chao, 1999, chap. 3). However, the challenge is to find a good combination and way of combining of SPRs to form effective CDRs.

Weighted priority index rules are the linear combination of SPRs described above with computed weights (Jayamohan & Rajendran, 2004; John & Xiaoming, 2004). Depending on specific business domains, the importance of a job determines its weight. For instance, consider n jobs with different weights w, each job J_i is assigned weight w_i. The minimization of weighted tardiness of all jobs is considered as an objective function. The sum of the weighted tardiness is given as follows:

$$T = \sum_{i=1}^{n} w_i T_i = \sum_{i=1}^{n} w_i \times \max(0, C_i - d_i)$$

In this paper, weighted priority rules are not considered as they are a generalization of our current formulation of total tardiness where we have assumed instead that all jobs have unit weights (or all jobs are equally important) (see Section 2).

Heuristic rules are rules that depend on the configuration of the system. These rules are usually used together with previous rules, such as SPRs, CDRs or weighted priority index rules. For instance, the heuristic rules may use the expertise of human experience, such as inserting an operation of a job into an idle time slot by visual inspection of a schedule (Panwalkar & Wafik, 1977).

The results from recent researchers (Barman, 1997; Oliver & Chandrasekharan, 1997) show that CDRs outperform individual SPRs in improving the overall efficiency of a shop floor. Furthermore, Jayamohan and Rajendran (2000) mention that there are no rules so far that have been found to perform well on multiple objectives related to flow time and due date. In this work, we focus on finding a computational method to build effective and robust CDRs that are based on the composition of fundamental measures rather than on the algebraic combination of SPRs. These CDRs are evolved to obtain good schedules on multiple objectives, instead of only a single objective. However, this may be difficult to enumerate manually due to the large parameter and operator space, hence we employ a GP framework.

3.3. Evolving dispatching rules with genetic programming

Genetic programming (GP) (Koza, 1992) belongs to a family of evolutionary computation methods. It is based on the Darwinian principle of reproduction and survival of the fittest. Given a set of functions and terminals and an initial population of randomly generated syntax trees (representing programs), these programs are then evolved through genetic recombination and natural selection. GP has been applied to many different problems; from classical tasks, such as function fitting or pattern recognition, to non-trivial tasks that are competitive with significant human endeavours such as designing electrical circuits (Koza, Bennett, Andre, Keane, & Dunlap, 1997) or antennas (Lohn, Hornby, & Linden, 2004).

The most important feature that distinguishes GP from the canonical GA is its ability to dynamically vary the logical structure and size of evolved computer programs. It can therefore solve more challenging problems that have eluded the canonical GA due to the latter's requirement of a fixed-length chromosome. However, GP has rarely been directly applied to solve schedule optimization (Dimopoulos & Zalzala, 2001) problems; this is due to the direct permutation property of scheduling where jobs and/or machines can be simply reordered (in the case of JSP) to improve optimality. For instance, the chromosomes presented in Kacem et al. (2002a, 2002b), Ho and Tay (2004), Tay and Wibowo (2004) have fixed lengths, which can be evolved easily by direct permutation. On the other hand, GP uses a tree-based encoding with dynamic length; making it difficult to encode the JSP (for that matter, a FJSP) into a tree-based chromosome. Unlike previous approaches (Barman, 1997; Jayamohan & Rajendran, 2004; John & Xiaoming, 2004; Oliver & Chandrasekharan, 1997), where a predefined set of SPRs were combined in advance by human experience, we apply GP to find superior constructions of CDRs which are *composed of* fundamental terminals (see Table 1).

4. Design of the GP framework

In GP, an individual (i.e., computer program) is composed of terminals and functions. Therefore, when applying GP to solve a specific problem, they should be carefully selected and designed to satisfy the requirements of the current problem. After evaluating many parameters related to the FJSP, the terminal set and the function set that are used in our algorithm are described as follows.

4.1. Terminal set

In job-shop scheduling, there are many operational parameters that can effect the quality of results; potentially, all of them can be considered to comprise a dispatching rule. However, in order to create a short and meaningful dispatching rule, only a small number of parameters is used. Doing so also reduces the search space and improves the efficiency of the scheduling algorithm. Based upon the dispatching rules involving due dates in Panwalkar and Wafik (1977), Blackstone et al. (1982), Oliver and Chandrasekharan (1997) and our experimental work, the terminal set proposed in this study is given in Table 1.

In Table 1, `CurrentTime` represents the time when a particular machine is free and starts to select a job to process on its queue. `RemainingTime` corresponds to the elapsed time for the current job to finish. Some previous dispatching rules use total processing time of each job as one of their parameters. However, in FJSP, as an operation of each job can be processed on a set of machines (see Section 2), we therefore normalize the average processing time of each operation with the following formula:

$$\bar{p}_{i,j} = \frac{\sum_{n(F(O_{i,j}))} p_{i,j,k}}{n(F(O_{i,j}))}$$

where $p_{i,j,k}$ stands for processing time of operation $O_{i,j}$ on machine M_k and $n(F(O_{i,j})$ represents the number of machines that can process $O_{i,j}$.

Table 1
Terminal set

Terminal	Meaning
`ReleaseDate`	Release date of a job (RD)
`DueDate`	Due date of a job (DD)
`ProcessingTime`	Processing time of each operation (PT)
`CurrentTime`	Current time (CT)
`RemainingTime`	Remaining processing time of each job (RT)
`numOfOperations`	Number of operations of each job (nOps)
`avgTotalProcTime`	Average total processing time of each job (aTPT)

Table 2
Function set

Function	Meaning
$+$	Addition
$-$	Subtraction
$*$	Multiplication
$/$	Division
ADF(x_1, x_2)	Automatically defined function

4.2. Function set

Similar to other applications of GP (Koza, 1992; Koza et al., 1997; Lohn et al., 2004) for solving optimization problems, we use four basic operators: addition, subtraction, multiplication, and division for creating our CDR. Furthermore, we employ the well-known automatically defined function (ADF) (proposed by Koza, 1994, chap. 4). The ADF is sub-tree which can be used as a function in the main tree. The size of the ADF is varied in the same manner as the main tree. It enables GP to define useful and reusable subroutines dynamically during its run. The results from Koza (1994, chap. 4) indicate that GP using ADF outperforms GP without ADF in solving the same optimization problem. As more parameters are used in ADF, more changes will be needed for GP to evolve good subroutines. This can lead to a higher number of generations, hence we limit the ADF used in our approach to two parameters. The operators used in the ADF are also the four basic operators mentioned above (listed in Table 2).

4.3. Fitness function

The obtained results from each generation of GP are a set of computer programs represented as trees. As mentioned in Section 2, the objective in our study is to find effective CDRs for solving the multi-objective FJSPs. Therefore, we propose a method to form a CDR from the tree-based result of GP. This CDR is then combined with the *least waiting time assignment* (Ho & Tay, 2004) to evaluate the fitness value of the FJSPs. These two processes are described in detail as follows.

To find a suitable machine to process an operation $O_{i,j}$, we apply the *least waiting time assignment* on the set of setting machines that can process $O_{i,j}$. This assignment is intended to reduce the workloads of the machines by balancing operations to be assigned. It is calculated by summing all the subsequent operations in the waiting list *plus* the remaining processing time on each machine and the processing time of $O_{i,j}$. Therefore, it depends on the total time this operation has to wait to be processed in the worst case, not relying only on its own processing time.

In determining the proper order of operations on the queue of a particular machine, we use the CDR generated by GP. When a machine is freed, the generated rule is applied directly to the set of operations that are waiting in the queue of the machine. The operation with the highest priority is then selected to be processed on the machine. Fig. 1 below gives an example of a dispatching rule tree generated by GP:

Fig. 1 shows the overall structure of the generated tree that gives a possible CDR. The left child of *progn* shows the function-defining branch (containing the *defun*). In this case, the ADF function is defined by: $\mathrm{ADF}(x_1, x_2) = x_1 * x_2$. The right child gives the result-producing branch. This CDR therefore represents the following formula:

$$\frac{(\mathrm{DD} - \mathrm{CR})}{(\mathrm{DD} - \mathrm{RD}) + \mathrm{ADF}(\mathrm{PT}, \mathrm{nOps})}$$

Since $\mathrm{ADF}(x_1, x_2) = x_1 * x_2$, we obtain:

$$\frac{(\mathrm{DD} - \mathrm{CR})}{(\mathrm{DD} - \mathrm{RD}) + (\mathrm{PT} * \mathrm{nOps})}$$

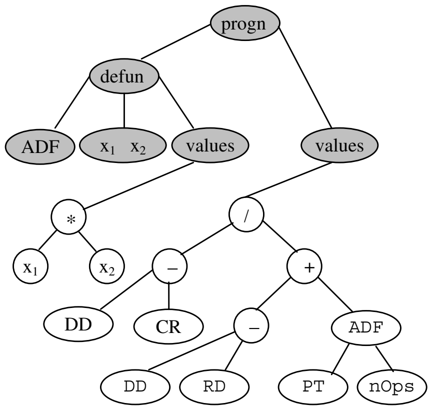


Fig. 1. Example of a GP tree with defined functions and terminals.

Any tree in the genomic population of GP that contains our defined functions and terminals can be interpreted as a CDR in the same way.

5. Experimental design

In this section, we describe the parameter settings used for our GP framework and the generation of test cases for evolving and evaluating CDRs.

5.1. GP parameter setting

Through experimentation, the set of parameters used in our GP framework is listed in Table 3.

We implemented *Ramped half and half* to generate the initial population of GP. This method was proposed by Koza (1992) and it has been widely used by previous researchers. It divides the initial population into two parts; half of which contains the randomly generated trees with maximum depth (in this experiment is 7) and the remaining half contains the randomly generated trees with depth values ranging from one to the maximum depth. In order to keep the best trees that may be destroyed by GP's operators, we sort the current population and copy five of them to the next generation. Although the size of the population is 100, we maintain its diversity by using high values of crossover, mutation rates (maximum depth for creation 7, maximum depth for crossover 17, crossover probability 100%, swap mutation probability 3%, shrink mutation probability 3%) and the creation type *Ramped half and half*.

5.2. Test case generation

Various experiments were conducted to evaluate the efficiency of our proposed algorithms. In practice, an operation can be processed on any of a group of machines that constitute a work center. We model this with

Table 3
Choice of parameter values

Parameter	Values
Population size	100
Number of generations	200
Creation type	Ramped half and half
Maximum depth for creation	7
Maximum depth for crossover	17
Crossover probability	100%
Swap mutation probability	3%
Shrink mutation probability	3%
Number of best rules copy to new generation	5

Table 4
Simulation parameters

Parameter	Value
Flexibility	100% (FJSP-100), 50% (FJSP-50), 20% (FJSP-20)
$nJobs \times nMachines$	10×5, 20×5, 50×5, 20×10, 50×10, 100×10, 50×15, 100×15, and 200×15
$p_{i,j,k}$	$U[(nMachines)/2, (nMachines) \times 2]$
$\mathrm{dev}(p_{i,j,k}, p_{i,j,l})$	5
r_i	if $nJobs \geq 50$: $U[0,40]$; otherwise, $U[0,20]$
c	1.2 (tight), 1.5 (moderate), 2 (loose)
d_i	$r_i + c \times \sum_{j=1}^{n_i} p_{ij}$

three experimental designs: FJSP with 100% flexibility (FJSP-100), FJSP with 50% flexibility (FJSP-50), and FJSP with 20% flexibility (FJSP-20). FJSP with c% flexibility implies that at most c% of the total number of machines are available to process an operation. The number of jobs (*nJobs*) and the number of machines (*nMachines*) range from 10 to 200 and 5 to 15, respectively. The processing time of each operation was drawn out of $U[(nMachines)/2, (nMachines) \times 2]$, where U refers to the uniform distribution. The variance of these processing times is in practice, ideally zero or very small. Therefore, in our test cases, we set the maximum processing time difference between two operations to be 5 unit times. The release date of each job depends on the number of jobs in a particular test case. If the number of jobs is larger than 50, the release date is drawn out of $U[0,40]$, else it is taken from $U[0,20]$. Baker (1984) proposed a formula to estimate the due date of a job using the TWK-method:

$$d_i = r_i + c \times \sum_{j=1}^{n_i} p_{ij}$$

where r_i and d_i denote the release and due dates of job i, respectively. p_{ij} presents the processing time of operation O_{ij}, and c denotes the tightness factor of the due date. The larger the value of c, the looser is the job's due date. We adapt this formula to generate due dates of jobs by replacing the parameter p_{iq} with $\bar{p}_{iq}$. Depending on the tightness of the due date, we separate the samples of each class FJSP-100, FJSP-50 or FJSP-20 into tight, moderate, or loose due date corresponding to values of $c = 1.2$, 1.5, and 2. We also generate mixed samples where each sample contains 34% of jobs with tight due dates, 33% of jobs with moderate due dates and the remaining ones with loose due dates. Specifically, the class FJSP-100 holds 9 samples of tight due dates, 9 samples of moderate due dates, 9 samples of loose due dates, and 9 samples of mix due dates. Similarly for FJSP-50 and FJSP-20, with 36 samples each. The simulation parameters are summarized in Table 4 below:

With the simulation parameters described above, our training set contains three classes of 108 FJSP problems with different quantities of jobs and machines, associated with different flexibilities. The total number of jobs that are used to calculate mean tardiness and mean flow time for each class is 2400. Another five validation sets of a similar composition using the same simulation parameters were generated. The training set is employed as input to the GP framework to evolve CDRs. These rules are applied to then solve the multi-objective FJSP problems in the five validation sets. The average results of the five validation sets are reported in the following section.

6. Analysis of experimental results

This section reports and analyses the results of our experiments for evolving CDRs using our GP framework. The system was implemented using C++, running on a 2 GHz PC with 512 MB RAM. Firstly, the best five evolved CDRs and five selected SDRs and CDRs in literature are reported. As mentioned earlier in Section 4.3, the *least waiting time assignment* (Ho & Tay, 2004) is used to find a suitable machine to process an operation $O_{i,j}$. However for the first assignment, all the machines are idle, therefore there are no operations on the machine's queues waiting for processing. Hence, we generate a random array of job orders and assign their first operations for the first machine assignment. The remaining ones are automatically assigned by the *least waiting time assignment* and by a CDR. Depending on the positions of the first operations in the job order, the end result could be different. In order to analyze the efficacy of the evolved rules and the appropriate combination of parameters to generate CDRs by GP, they are evaluated with respect to mean values of makespan, total tardiness and mean flow time after 500 runs. Besides the three objectives described in Section 2, our results also show that the percentage of tardy jobs is reduced significantly. This measurement is common in literature in comparing the performance of dispatching rules (Barman, 1997; Jayamohan & Rajendran, 2004; Oliver & Chandrasekharan, 1997). The performances of the evolved CDRs on each objective and on all objectives are analyzed and discussed. For comparative studies, we employed the technique called one way analysis of variances (ANOVA) (Johnson, 2001; Quek & Tay, 2000). The function of the ANOVA is based on the ratio of variations. It tests the difference between the means of two or more groups. In this paper, it is used to compare the sample mean of a particular objective for an evolved rule with other sample means (for other rules) that overlap with the former's confidence interval (CI). If an overlap exists, this implies some uncertainty concerning the existence of a performance differential. The values of 95% CI for each sample mean were calculated and presented.

6.1. The best five evolved CDRs

The best five dispatching rules selected from 5 runs of GP on the training set are given in Table 5; where possible, they have been simplified algebraically.

In order to compare the effectiveness of the evolved rules over that of human-made rules presented in literature, five frequently used single and composite dispatching rules were selected as benchmarks:

- FIFO (First In First Out): select the next job in front of the queue. This rule is often used in practice since it is simple and easy to implement (Blackstone et al., 1982).
- SPT (Shortest Processing Time): select the job with the shortest processing time. This rule is commonly used as a benchmark for minimizing mean flow time and percentage of tardy jobs (Jayamohan & Rajendran, 2000).
- EDD (Earliest Due Date): select the job with the earliest due date. This rule is the most popular due date based rule. It is known to be used as a benchmark for reducing maximum tardiness and variance of tardiness (Jayamohan & Rajendran, 2000).
- MDD (Modified Due Date) ($=\max\{CT + PT_i, DD_i\}$): process the jobs in non-decreasing order of MDD. This rule is aimed to minimize total tardiness of jobs (John & Xiaoming, 2004).
- SL (Slack Time) ($=DD_i - CT - RT_i$): select the job with the minimum slack time. This rule is also used to reduce total tardiness of jobs (Oliver & Chandrasekharan, 1997).

Blackstone et al. (1982) mentions that the study of job shops by analytic techniques, such as queuing theory, becomes extremely complex even for small problems. Therefore, the use of simulation for analyzing dispatching rules is unavoidable. Due to the same difficulty in examining the dispatching rules for solving FJSPs, we also rely on simulation to study the rules' effectiveness.

Table 5
GP generated dispatching rules

Rule	Expression
Rule_1	RD + 2PT + 2aTPT + nOps
Rule_2	RD + PT + 2aTPT
Rule_3	7aTPT + 11PT + 12(nOps + RD)
Rule_4	(RD + DD) + 2(RD + aTPT) + PT − nOps
Rule_5	(RD + DD) + (aTPT + PT) − 2(RD/nOps)

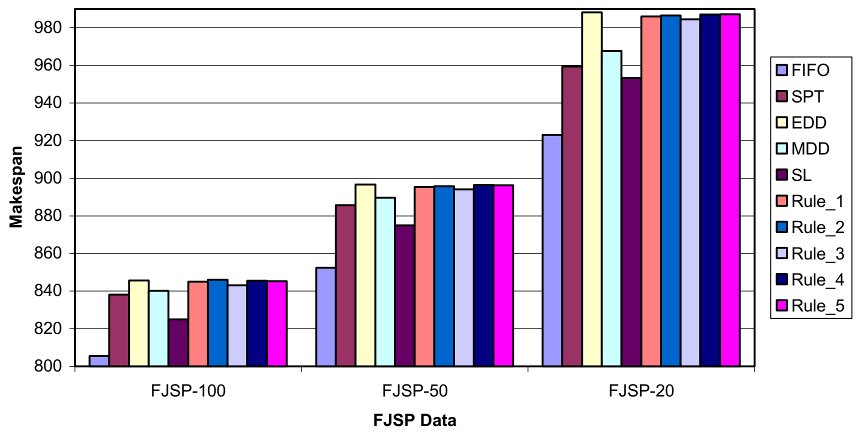


Fig. 2. Comparing makespan of dispatching rules on different FJSP data.

6.2. Makespan

Fig. 2 shows the makespan results of dispatching rules presented in Section 6.1 on different FJSP data. The results indicate that the FIFO rule emerges as the best in minimizing the makespan objective in all cases regardless of changing the flexibility of the problem instances. This is because FIFO uses the release dates (RD) to schedule the jobs on the queue. The jobs with early release dates are more likely to be selected to be processed than the later ones. Therefore, the overall completion time of the schedule (makespan) is reduced. The composite dispatching rule SL ranks second in minimizing the makespan objective. The reason for good performance of this rule is that it reduces the deviation of job completion from the due date. Consider two jobs that have the same due date, the job with the larger remaining time (RT) will have higher priority. Among the five selected rules from literature (FIFO, SPT, EDD, MDD, SL), the EDD rule is the worst in minimizing the makespan. This is because EDD concentrates only on due dates. The obtained schedule minimizes the objectives related to due date, such as mean tardiness, but the completion time of whole system is ignored.

The makespan values of the five evolved rules fare better than the EDD. Fig. 3 represents the mean makespans of EDD and five evolved rules with 95% CI after 500 runs. The X axis represents the rules and the Y axis represents the mean values of each rule with 95% CI. Among these rules, *Rule_3* achieves better results than the others. Furthermore, its CI does not overlap with the others. Therefore, it is considered the best rule among EDD and the evolved rules for minimizing the makespan. These results could be explained by the presence of the parameter $12 \times \text{RD}$ in its formula. Similar to the FIFO rule where release date (RD) contributes mainly in reducing makespan objective, this parameter also helps *Rule_3* obtain better results than remaining ones. Another plausible inference of the importance of the parameter RD is the fact that *Rule_3* is better than the others due in part as their formulas contain only $1 \times \text{RD}$, $2 \times \text{RD}$, or $3 \times \text{RD}$.

6.3. Mean tardiness

Fig. 4 compares the mean tardiness of dispatching rules on different FJSP data. The FIFO rule obtains good results in minimizing the makespan (see Section 6.2). However, it performs poorly in minimizing mean tardiness in comparison with the others. This is because the due dates of jobs are ignored by FIFO. The composite dispatching rule SL can obtain better results than FIFO in minimizing mean tardiness but its results are still poor in comparison to the remaining rules. The results of minimizing mean tardiness in Fig. 4 indicate that MDD and EDD outperform SL. From the definitions of MDD and SL described above, we observe that although these two composite rules contain similar parameters (DD and CT), the gap between the results of the two rules are quite large due to different algebraic combinations of the parameters. This implies that the functions that combine the rules can significantly affect the results. Blackstone et al. (1982) mentions that the SPT seems to be the best rule when the problem does specify due dates or have very tight due dates for a

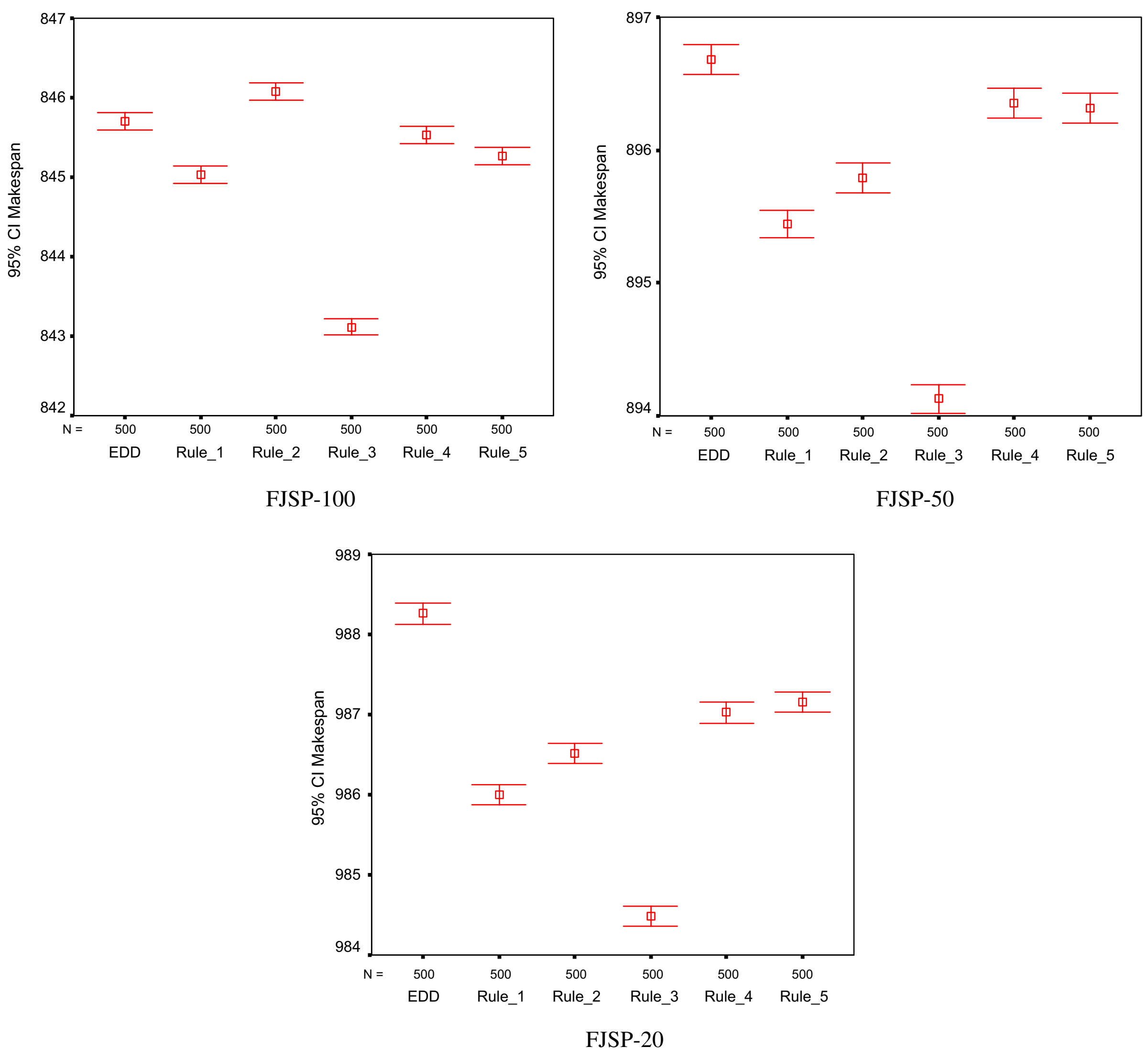


Fig. 3. Mean makespan data distribution with 95% confidence interval after 500 runs.

classical job shop. When the problem specifies loose or moderate due dates then EDD seems to be the best. However, the results presented in Fig. 4 indicate the contrary, that EDD outperforms SPT for all classes of due date constraints. The EDD is the best rule among five dispatching rules selected from literature for solving the FJSPs. This could be explained by the flexibility feature of FJSPs where an operation can be processed on one of several machines. When a FJSP specifies tight due dates, each job in this problem still has alternative routes to take through the system, not just one route as in the classical JSP. Therefore, if the job on the queue is selected by EDD, it is more likely to finish on time. Although the other rules such as SL or MDD also contains the parameter – due date (DD), EDD obtains almost 30% better results than these rules. This again demonstrates that if an ineffective composite dispatching rule is applied to specific problems, it may achieve worse results than the single ones.

Fig. 5 represents the mean tardiness values of EDD and five evolved rules with 95% CI. The best performing rule in minimizing mean tardiness is the generated rule – *Rule_1* (RD + 2PT + 2aTPT + nOps). Since its CI does not overlap with the others, it is considered to be the best rule among them to minimize mean tardiness. Statistical analysis shows that all evolved rules perform statistically better than the EDD at the 5% level

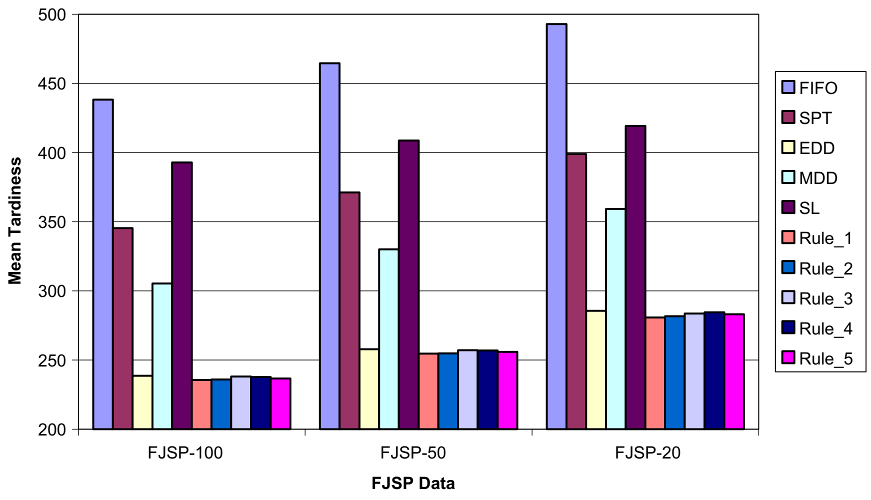


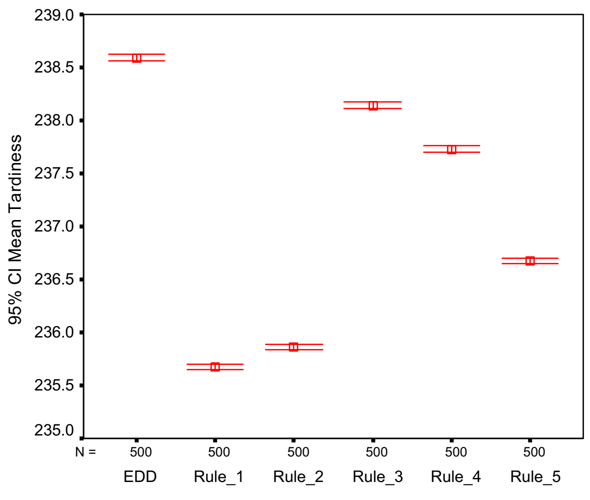
Fig. 4. Comparing mean tardiness of dispatching rules on different FJSP data.

of significance. Jayamohan & Rajendran (2000) mentions that the use of both due date information and processing time can lead to good results in minimizing mean tardiness. Our five evolved rules present evidence for this conclusion as their formulation contains these parameters. Furthermore, we find that some parameters, such as total number of operations (nOps) and total processing time of job (aTPT), are ignored or considered insignificant by previous researchers but do contribute to reducing mean tardiness. For example, the formula of *Rule_3* indicates that jobs with higher number of operations have higher priority. Furthermore, the results of *Rule_4* in Fig. 5 also demonstrate the sensitivity of parameter combinations. *Rule_4* also has parameter nOps. However, its results are worse than the other due to the factor −1 is used (−nOps) to form the rule.

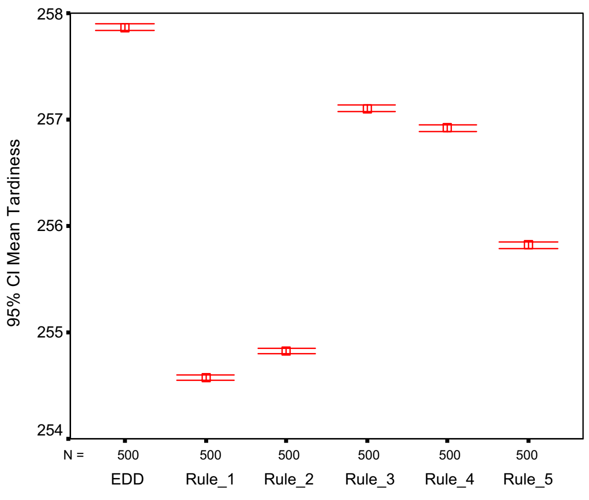
6.4. Mean flow time

Fig. 6 shows the mean flow times of dispatching rules on different FJSP data. The results indicate that FIFO obtains the worst result since it has the same disadvantages described in the previous section. The EDD rule gets the best results in all cases among the five selected rules from literature. The second rank goes to the MDD rule. These results are contrary to the results of minimizing the flow time for a flow shop (Barman, 1997) where the SPT was reported to outperform the EDD. This anomaly could be explained by the flexibility of the FJSP where an operation has more than one machine to select to be processed. The *least waiting time assignment* (see Section 4.3) already helps to find suitable machines to assign jobs. Consequently, the waiting time of the whole system is reduced. Therefore, if a job with early due date is considered, it is more likely to finish sooner.

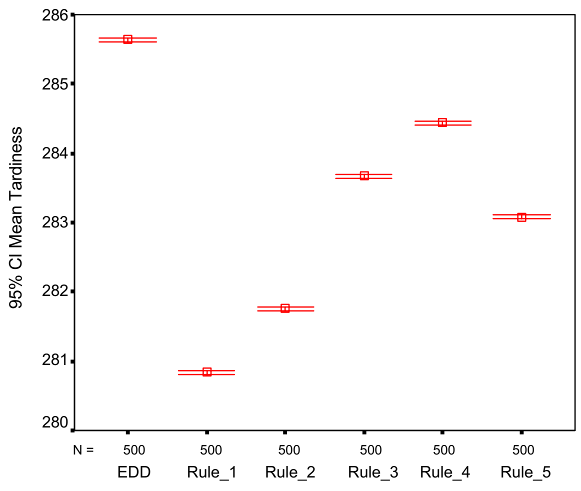
Fig. 7 represents the mean flow times of EDD and five evolved rules with 95% CI. *Rule_3* is the best for solving both FJSP-100 data and FJSP-50 data. In the FJSP-20 data, although the mean value of *Rule_1* is smaller than the mean value of *Rule_3*, we cannot conclude that *Rule_1* is better than *Rule_3* as their CIs overlap. In order to verify that *Rule_1* is significantly different from the remaining ones, we applied ANOVA to analyze the data. Since $F = 1248.86$ is greater than $F_{\text{critical}} = 2.21$, there is sufficient evidence to reject the null hypothesis that the samples are not significantly different. Therefore, *Rule_1* is the best for solving FJSP-20 data with mean flow time objective. Fig. 7 also shows that the evolved rules outperform EDD in all cases. A closer look at the formulas of the five evolved rules validates the conclusion from Barman (1997) where the combination of the SPT and the EDD provide excellent mean flow time performance. All of them contain the parameters – release date (RD) and processing time (PT) which help to reduce the flow time of the jobs. Moreover, the two best rules *Rule_1*, *Rule_3* also contain the parameter – average total processing time (aTPT). This provides an important factor to reduce the travel time of jobs in the system where the mean tardiness objective is concerned. With the contribution of this factor, a job with shorter total processing time is more likely to finish early, and consequently the total flow time of all jobs will be minimized.



FJSP-100



FJSP-50



FJSP-20

Fig. 5. Mean Tardiness Data Distribution with 95% confidence interval after 500 runs.

6.5. Percentage of tardy jobs

Fig. 8 shows the percentage of tardy jobs of dispatching rules on different FJSP data. Among the five selected rules from literature, the FIFO rule again obtains the worst results in minimizing the percentage of tardy jobs. The MDD rule emerges to be the best for this category. The EDD and the SPT rank second and third, respectively. The deviation between the EDD and the MDD is statistically insignificant. The better performance of the EDD and MDD in comparison to the SPT could be explained by the use of the due date parameter. The job with early due date is scheduled to be processed sooner. Therefore, the number of tardy jobs will increase.

Fig. 9 represents the mean percentage of tardy jobs of the EDD, MDD and the five evolved rules with 95% CI. Among all the rules tested in Fig. 9, we can conclude that *Rule_2* is the best for solving FJSP-50 data and FJSP-20 data since it obtained the smallest mean values and its CIs do not overlap with the other rules. However, on FJSP-100 data, its mean value is the smallest one but its CI overlaps with the mean value of *Rule_1*. By applying ANOVA, we obtain $F = 5699.78$ which is greater than $F_{\text{critical}} = 2.10$. Therefore, there is sufficient evidence to reject the null hypothesis that the samples are not significantly different. It implies that *Rule_2* is

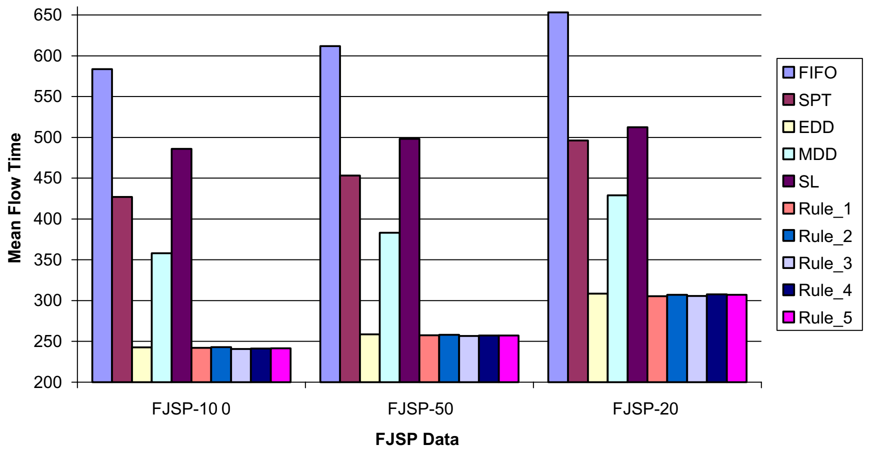


Fig. 6. Comparing mean flow time of dispatching rules on different FJSP data.

the best for solving FJSP-100 data. Furthermore, the other evolved rules also get better results than the EDD and MDD rules. The results of these rules are in agreement with the results in literature (Blackstone et al., 1982; Jayamohan & Rajendran, 2000) where process time (PT) plays an important role with respect to minimizing percentage of tardiness. The parameter PT appears in the formulas of our five evolved rule. Besides this parameter, we found that the release date (RD) parameter also contributes much to the success of minimizing this objective. It is used in all five evolved rules. The job with early release date will be selected to be processed, and consequently the percentage of tardy jobs is reduced.

6.6. *Sensitivity of parameters*

Although the five evolved rules are only better than the EDD in minimizing the makespan objective, they are all better than the other five selected rules with respect to the remaining objectives. In this sense, it can be concluded that the evolved rules produced by our GP framework are very competitive with the human-made rules selected from literature. In order to understand why these evolved rules are effective in solving the FJSP with respect to multiple objectives, we now analyze the sensitivity of the parameters.

While single rules consider only one parameter of the shop, the evolved rules employ almost all the important parameters. However, as mentioned early, the combination of these parameters plays an essential role to the success of the rule. The parameters PT and RD could be important for solving the problem. They are present in all the rules and contribute mainly to change the priority of one operation to be selected in a queue. We also found that the two parameters aTPT and nOps appear in all the formulas of the evolved rules but are usually ignored in literature. Consider the parameter aTPT in the formula of *Rule_1*. When the average total processing time of a job is small, it indirectly effects the tight due date of the job, and the result of evaluating *Rule_1* will be small. This job then has high priority to be selected on the queue. Similarly, consider *Rule_3* where nOps is one of its parameters. When the number of operations of the job is large, its formula produces a small value. It therefore helps to assign a high priority to the job. Note that using the important parameters is only one aspect of the success of the rule. The other aspect that is also essential is the algebraic combination of these parameters. In order to understand how the parameters as well as the algebraic combination of these parameters influence the obtained results. We generated five other modified rules from *Rule_1* by eliminating each parameter or slightly changing the parameter's coefficient. The modified rules are presented in Table 6.

In Table 6, *Rule_1_1* and *Rule_1_2* are constructed from *Rule_1* by eliminating the parameters RD and PT, respectively. By changing the coefficient of aTPT in *Rule_1*'s formula from 2 to 12, we obtain *Rule_1_3*. Similarly, *Rule_1_4* is built by changing the coefficient of nOps in *Rule_1*'s formula from 1 to 10. These modified rules are then applied to solve FJSP-50. Fig. 10 below compares their mean makespan, mean total tardiness, mean flow time and mean percentage of tardy jobs with 95% CIs to *Rule_1*'s results after 500 runs.

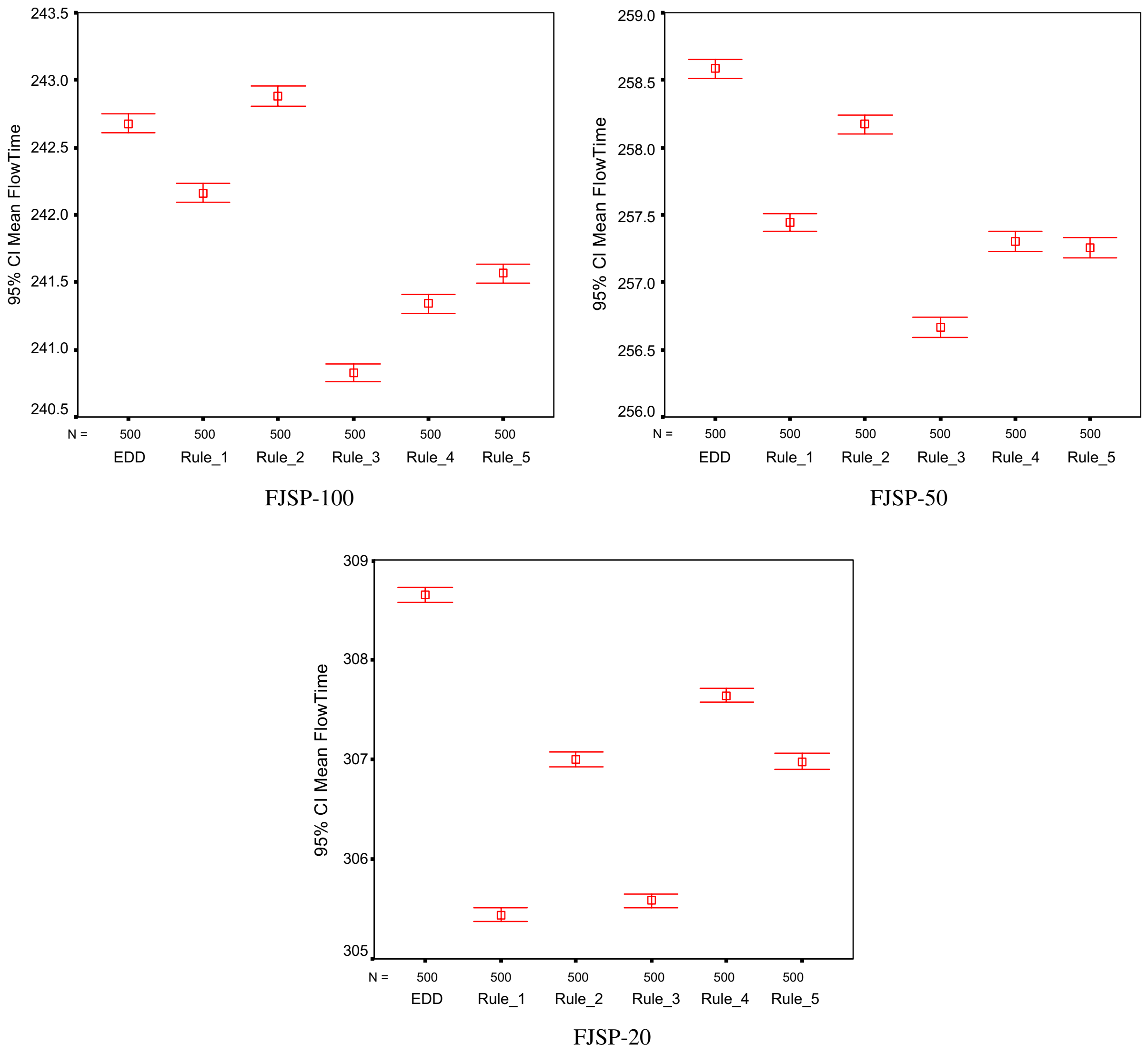


Fig. 7. Mean Flow Time Data Distribution with 95% confidence interval after 500 runs.

Fig. 10 indicates that although small modifications were made to a small number of parameters of an evolved rule, some results from the modified rules were worse than the original one. This implies that the evolved dispatching rules from the GP framework are well designed and robust. It also validates the importance of selecting proper parameters and of the proper algebraic combination of these parameters to construct efficient CDRs. Any changes on the evolved rules could lead to poorer results of some objectives.

6.7. Overall observation

Fig. 11 below shows the overall performance of dispatching rules using the aggregation function *F* (described in Section 2) on different FJSP data. Among the five selected rules from literature, the EDD rule obtains the best results. The MDD rule ranks second but still far from the results of EDD. The worst results come from the FIFO rule. Table 7 presents the detailed mean values drawn in Fig. 11. It shows that the five evolved CDRs achieve better results than the five rules selected from literature and *Rule_1* is the best.

The experimental results from Sections 6.2 to 6.6 and the overall results in Table 7 indicate that when the flexibility of the FJSPs decrease (FJSP-100 to FJSP-20), the objective values increase. The reason is that in the

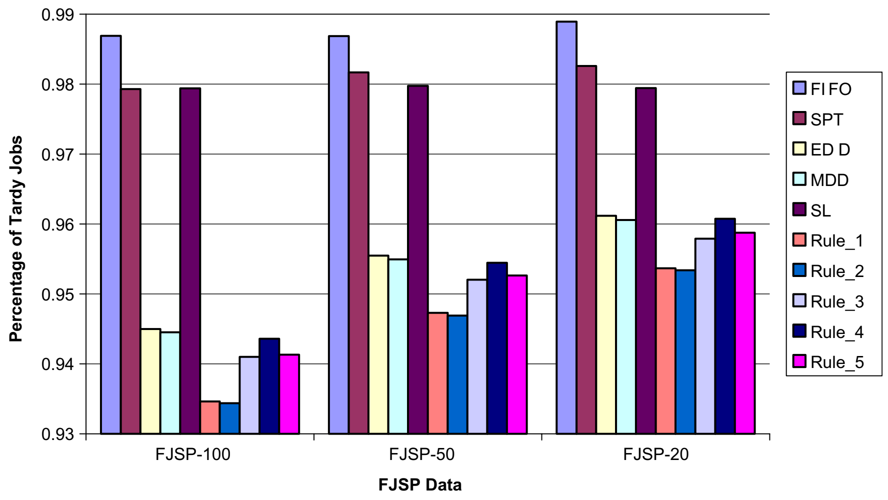


Fig. 8. Comparing percentage of tardy jobs of dispatching rules on different FJSP data.

less flexible FJSP, the operations of the jobs do not have many machines to select to be processed on. Among the five selected rules from literature, FIFO is the best with respect to minimizing makespan, EDD is the best for minimizing both mean tardiness and mean flow time while the MDD and the EDD are found to be the best in minimizing percentage of tardy jobs. This evidence emphasizes the results from literature (Blackstone et al., 1982) that no rule is singularly effective for all criteria. As mentioned in Section 2, a trade-off always exists in minimizing multiple objectives. The results of the FIFO rule explain this observation. The FIFO rule obtains very good results in minimizing the makespan. However, it is poorer than the remaining ones in other objectives. The converse is true for EDD. This is because when a rule prioritizes on shortening the completion time of the whole system, the jobs with loose due dates will finish quite early. However, this leads to the tardiness of jobs with short or moderate due dates. Consequently, this influences the objectives of mean tardiness and percentage of tardy jobs objectives. The results also suggest that the combination of the rules is not always effective. They are sometimes worse than the results of SPRs. For instance, the MDD and SL have the same parameters DD and CT but the MDD is more efficient than the SL in almost all obtained results. Furthermore, they fail to get better results than the EDD which contains only one parameter DD. This indicates that the efficacy of the rules not only depends on suitable parameters but also on the way that these parameters are combined.

Generally, the overall experimental results justify that no SPR rules fare well on all objectives. CDR rules are considered to obtain better results on multi-objective FJSPs. This is evidenced by the evolved CDR Rule_*1* and its variants. Furthermore, two parameters aTPT and nOps that have received limited study from previous research were found to contribute significantly towards the effectiveness of CDRs. However, the importance of selecting proper parameters is only one aspect of building effective CDRs. The way to combine these parameters is also important. By investigating the potential use of GP for evolving effective CDRs, both parameters and their combination are explored. The experimental results indicate that the obtained rules are promising to apply to solve real world FJSPs.

7. Conclusion and future works

In this paper, a GP-based approach for discovering effective composite dispatching rules for solving the multi-objective FJSP has been presented and analyzed. CDRs have been studied widely by previous researchers (Blackstone et al., 1982; Oliver & Chandrasekharan, 1997; Panwalkar & Wafik, 1977). However, all of them were constructed based on the experience of a human scheduler. We employ a GP framework to generate a CDR based on algebraic fundamental terminals that can effectively solve the multi-objective FJSP (together with a machine assignment rule). Five composite dispatching rules were generated by our GP framework over

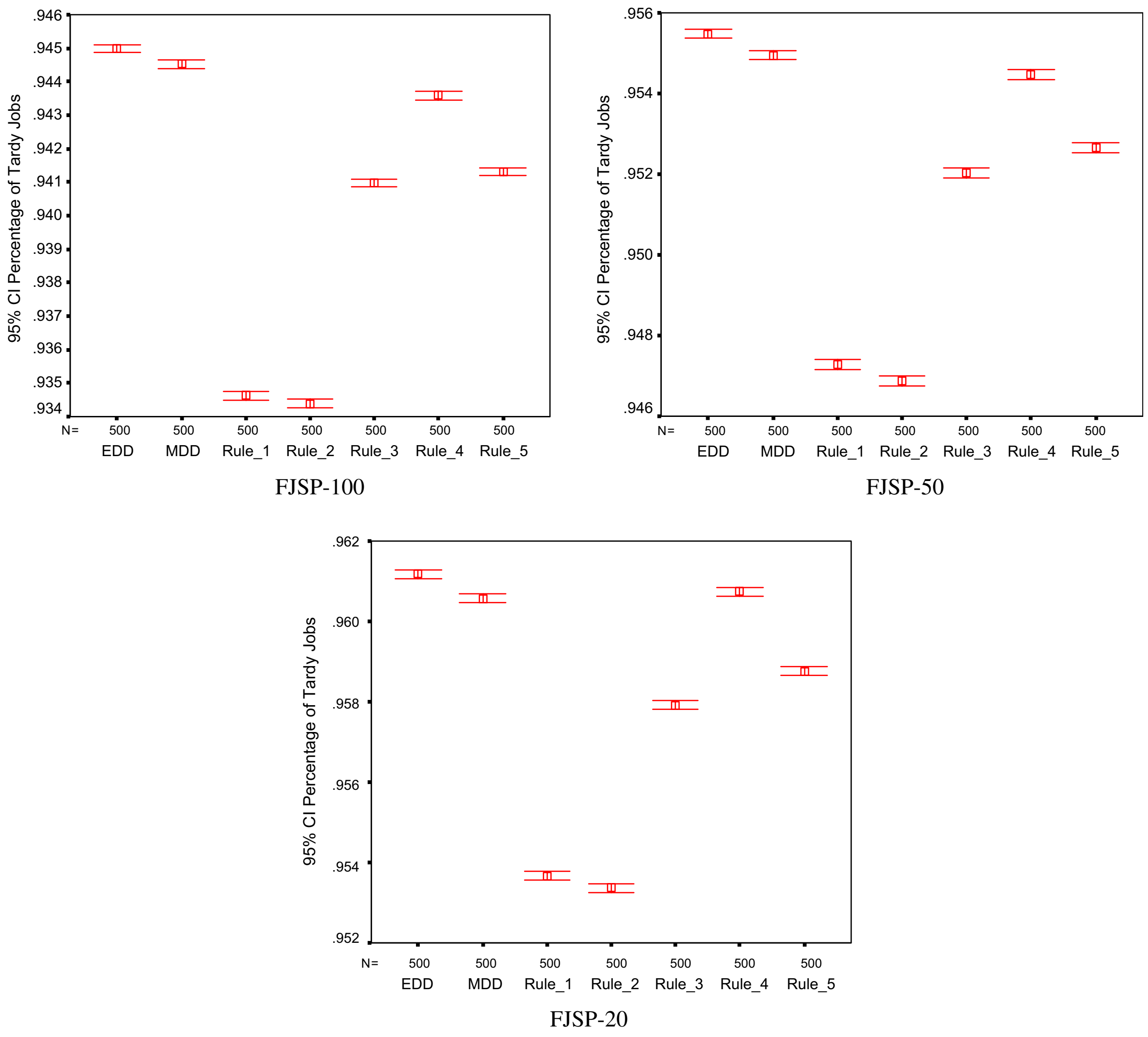


Fig. 9. Mean Percentage of Tardy Jobs Data Distribution with 95% confidence interval after 500 runs.

Table 6
Four modified dispatching rules from *Rule_1*

Rule	Expression	Modification(s) from *Rule_1*
Rule_1	RD + 2PT + 2aTPT + nOps	Original version
Rule_1_1	2PT + 2aTPT + nOps	Removed RD
Rule_1_2	RD + 2aTPT + nOps	Removed PT
Rule_1_3	RD + 2PT + 12aTPT + nOps	Changed aTPT's coefficient from 1 to 12
Rule_1_4	RD + 2PT + 2aTPT + 10nOps	Changed nOps's coefficient from 1 to 10

a large training set. These rules are based on the combination of parameters such as processing time, release date, due date, current time, number of operations and average total processing time of each job. Extensive simulations have been carried out to evaluate the performance of the five evolved rules over varying degrees of problem flexibility and due date tightness. Five other popular rules selected from literature were also evaluated and used as performance benchmarks.

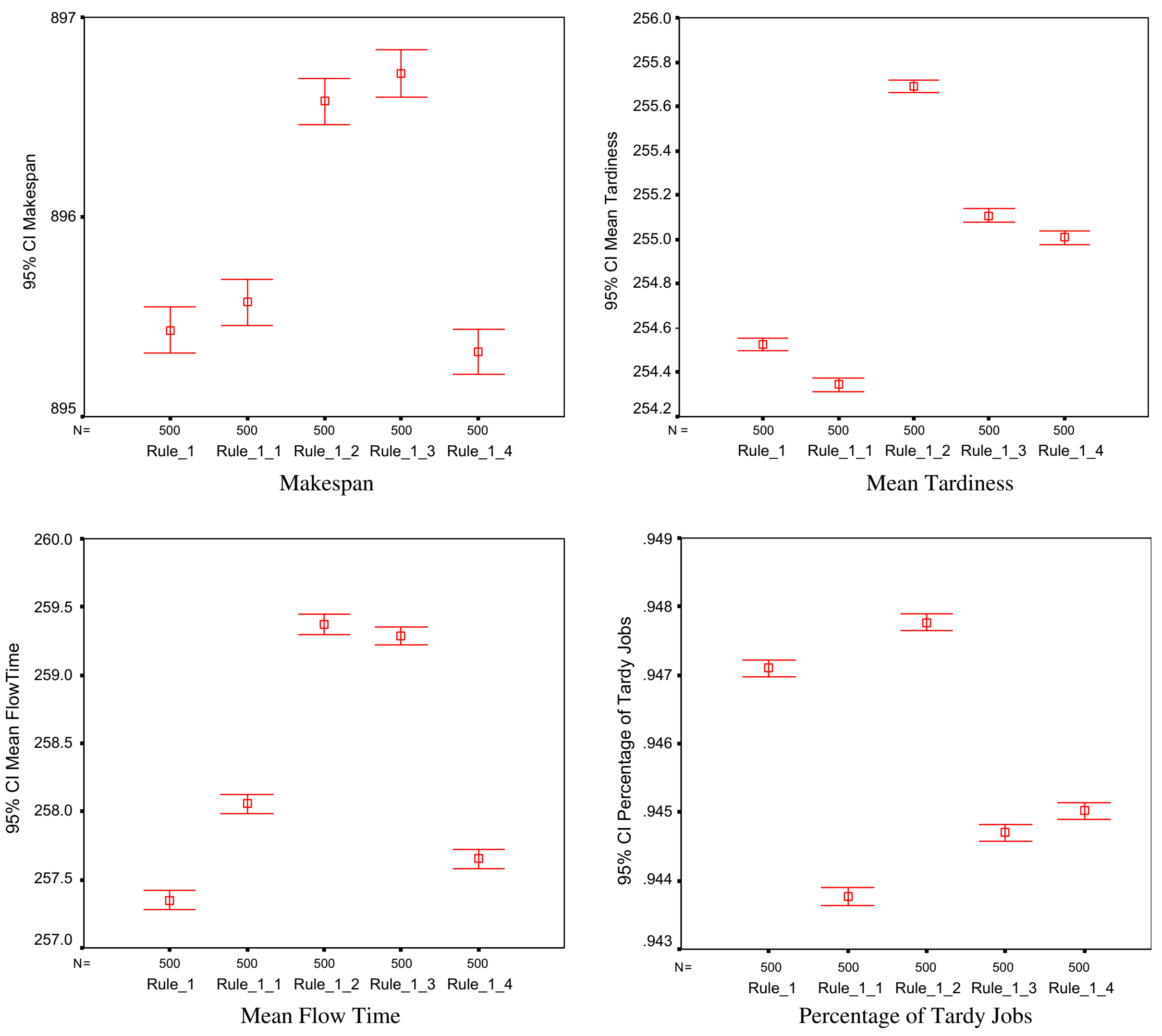


Fig. 10. Comparing data distribution of each objective of modified rules with 95% confidence interval to *Rule_1* after 500 runs.

The experimental results of these popular rules validate the literature finding that no rule performs well on all criteria, and the algebraic combination of the parameters contributes to the efficacy of the rules. It was found that EDD is significantly better than the other rules from literature in minimizing mean tardiness, mean flow time and percentage of tardy jobs while it is poor in minimizing the makespan. However, all the generated rules found by GP outperformed the EDD with respect to all objectives. It shows that the application of our GP framework for constructing effective composite dispatching rules for solving the multi-objective FJSPs has been successful. In particular, two parameters aTPT and nOps that have received limited study from previous research was found to contribute significantly to the effectiveness of evolved CDRs. We have also shown statistically that our evolved CDRs are sufficiently well-designed through the use of ANOVA (which analyzed the sensitivity to changes in the coefficient values and terminal parameters). Finally, by using a large training data set, we believe that our evolved CDRs can be applied directed in practice without further modifications.

Several possible extensions of this study can be developed. Similar to other applications of GP where the parameters are sensitive, denser terminal sets and more varied ADFs should be investigated to improve the generated rules. The approach of this study can be applied to find the efficient composite dispatching rules for other similar problems, such as a flow shop or the classical job shop. Further research about minimizing

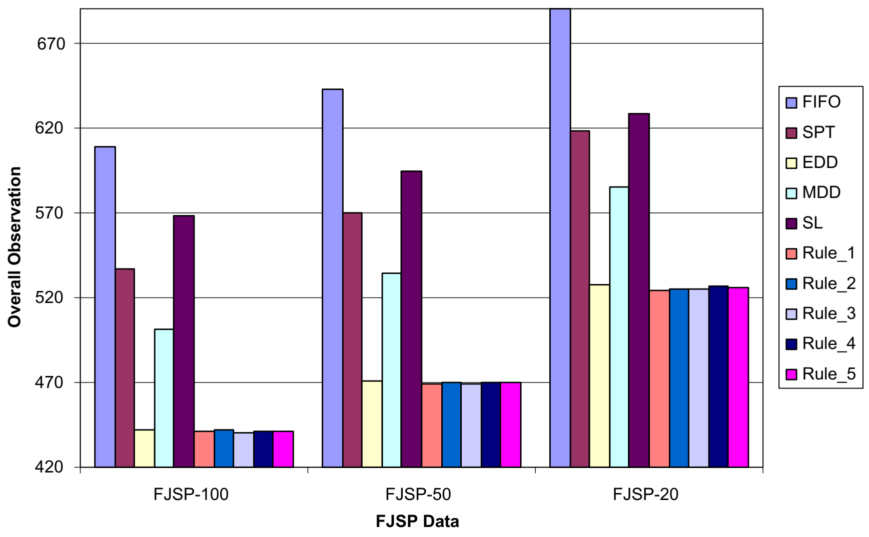


Fig. 11. Comparing overall observation of dispatching rules on different FJSP data.

Table 7
Overall observation data

	FIFO	SPT	EDD	MDD	SL	*Rule_1*	*Rule_2*	*Rule_3*	*Rule_4*	*Rule_5*
FJSP-100	609.13	536.82	442.32	501.20	567.99	440.96	441.61	440.70	441.53	441.17
FJSP-50	642.97	570.02	471.05	534.30	593.99	469.15	469.60	469.30	470.19	469.80
FJSP-20	689.65	618.18	527.52	585.30	628.29	524.09	525.09	524.58	526.37	525.74

both earliness and tardiness could be considered. The application of GP framework to construct CDRs for dynamic job shop with breakdown machines will be investigated. The rules evolved from this GP framework are still quite complex in structure. Therefore, an algebraic simplification tool could be used to make the formula more meaningful. Consideration could even be given to including the number of parameters used as a measure for minimization. In this paper, the same weight 1/3 is assigned to the three same priority objectives. Different specifications of the weight values of different priority objectives can also be considered.

Acknowledgments

This research was funded in part by Nanyang Technological University and CEI Contract Manufacturing Limited Company, Singapore.

References

Baker, K. R. (1984). Sequencing rules and due-date assignments in job shop. *Management Science, 30*(9), 1093–1104.

Barman, S. (1997). Simple priority rule combinations: An approach to improve both flow time and tardiness. *International Journal of Production Research, 35*(10), 2857–2870.

Blackstone, J. H., Phillips, D. T., & Hogg, G. L. (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research, 20*(1), 27–45.

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research, 22*(3), 158–183.

Brucker, P., Jurisch, B., & Krämer, A. (1997). Complexity of scheduling problems with multi-purpose machines. *Annals of Operations Research, 70*, 57–73.

Carlier, J., & Pinson, E. (1999). An algorithm for solving the job-shop problem. *Management Science, 35*(2), 164–176.

Coello, C. A. C. (2005). Recent trends in evolutionary multiobjective optimization. In Ajith Abraham, Lakhmi Jain, & Robert Goldberg (Eds.), *Evolutionary multiobjective optimization: Theoretical advances and applications* (pp. 7–32). London: Springer-Verlag.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation, 6*(2), 181–197.

Dimopoulos, C., & Zalzala, A. M. S. (2001). Investigating the use of genetic programming for a classic one-machine scheduling problem. *Advances in Engineering Software, 32*(6), 489–498.

Garey, M. R., Johnson, D. S., & Sethi, R. (1996). The complexity of flow shop and job-shop scheduling. *Mathematics of Operations Research, 1*(2), 117–129.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*, 236–287.

Ho, N. B., & Tay, J. C. (2004). GENACE: An efficient cultural algorithm for solving the flexible job-shop problem. In *Proceedings of the congress on evolutionary computation CEC2004* (pp. 1759–1766).

Hoitomt, D. J., Luh, P. B., & Pattipati, K. R. (1993). Practical approach to job-shop scheduling problems. *IEEE Transactions on Robotics and Automation, 9*(1), 1–13.

Ishibuchi, H., Yoshida, T., & Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation, 7*(2), 204–223.

Jain, A. S., & Meeran, S. (1998). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research, 113*(2), 390–434.

Jayamohan, M. S., & Rajendran, C. (2000). New dispatching rules for shop scheduling: A step forward. *International Journal of Production Research, 38*, 563–586.

Jayamohan, M. S., & Rajendran, C. (2004). Development and analysis of cost-based dispatching rules for job shop scheduling. *European Journal of Operational Research, 157*(2), 307–321.

Johnson, R. A. (2001). *Statistics: Principles and methods*. John Wiley.

John, J. K., & Xiaoming, L. (2004). A weighted modified due date rule for sequencing to minimize weighted tardiness. *Journal of Scheduling, 7*(4), 261–276.

Kacem, I., Hammadi, S., & Borne, P. (2002a). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man and Cybernetics, 32*(1), 1–13.

Kacem, I., Hammadi, S., & Borne, P. (2002b). Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation, 60*(3), 245–276.

Kolonko, M. (1999). Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research, 113*(1), 123–136.

Koza, J. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambrige, MA: MIT Press.

Koza, J. (1994). *Genetic programming II, automatic discovery of resuable programs*. MIT Press.

Koza, J. R., Bennett, F. H., III, Andre, D., Keane, M. A., & Dunlap, F. (1997). Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation, 1*(2), 109–128.

Lohn, J. D., Hornby, G. S., & Linden, D. S. (2004). An evolved antenna for deployment on NASA's Space Technology 5 Mission. In *Proceedings of the genetic programming theory practice 2004 workshop (GPTP-2004)*.

Mastrolilli, M., & Gambardella, L. M. (2000). Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling, 3*(1), 3–20.

Nowicki, E., & Smutnicki, C. (1996). A fast Taboo Search algorithm for the job shop problem. *Management Science, 42*(6), 797–813.

Oliver, H., & Chandrasekharan, R. (1997). Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics, 48*(1), 87–105.

Ong, Z. X., Tay, J. C., & Kwoh, C. K. (2005). Applying the clonal selection principle to find flexible job-shop schedules. In *Proceedings of the international conference for artificial immune systems, LNCS 3627* (pp. 442–455).

Panwalkar, S., & Wafik, I. (1977). A survey of scheduling rules. *Operations Research, 25*(1), 45–61.

Pinedo, M. (2002). *Scheduling theory, algorithms, and systems*. Prentice Hall.

Pinedo, M., & Chao, X. (1999). *Operations scheduling with applications in manufacturing and services*. McGraw-Hill.

Quek, H. C., & Tay, J. C. (2000). Issues in the performance measurement of constraint satisfaction techniques. *Artificial Intelligence in Engineering, 14*, 281–294.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the first international conference on genetic algorithms* (pp. 93–100). Lawrence Erlbaum.

Tay, J. C., & Wibowo, D. (2004). An effective chromosome representation for evolving flexible job-shop schedules. In *Proceedings of the genetic and evolutionary computation GECCO2004* (pp. 210–221).

Yamada, T., & Nakano, R. (1996). A fusion of crossover and local search. In *Proceedings of the IEEE International Conference on Industrial Technology* (pp. 426–430).

Zitzler, E., & Thiele, L. (1999). Multi-objective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation, 4*(3), 257–271.