J Ind Eng Int (2017) 13:297–306
DOI 10.1007/s40092-017-0185-y

ORIGINAL RESEARCH

New scheduling rules for a dynamic flexible flow line problem with sequence-dependent setup times

Hamidreza Kia[1] · **Seyed Hassan Ghodsypour**[1] · **Hamid Davoudpour**[1]

Received: 27 October 2015 / Accepted: 4 January 2017 / Published online: 19 January 2017

Abstract In the literature, the application of multi-objective dynamic scheduling problem and simple priority rules are widely studied. Although these rules are not efficient enough due to simplicity and lack of general insight, composite dispatching rules have a very suitable performance because they result from experiments. In this paper, a dynamic flexible flow line problem with sequence-dependent setup times is studied. The objective of the problem is minimization of mean flow time and mean tardiness. A 0–1 mixed integer model of the problem is formulated. Since the problem is NP-hard, four new composite dispatching rules are proposed to solve it by applying genetic programming framework and choosing proper operators. Furthermore, a discrete-event simulation model is made to examine the performances of scheduling rules considering four new heuristic rules and the six adapted heuristic rules from the literature. It is clear from the experimental results that composite dispatching rules that are formed from genetic programming have a better performance in minimization of mean flow time and mean tardiness than others.

Keywords Scheduling · Dynamic flexible flow line · Simulation · Heuristics · Genetic programming

✉ Seyed Hassan Ghodsypour
Ghodsypo@aut.ac.ir

Hamidreza Kia
hamid_R_kia@aut.ac.ir

Hamid Davoudpour
hamidp@aut.ac.ir

[1] Department of Industrial Engineering and Management Systems, Amirkabir University of Technology (Tehran Polytechnics), 424 Hafez Avenue, Tehran 15916-34311, Iran

Introduction

Scheduling involves the allocation of resources over a period of time to perform a collection of tasks (Baker 1974). It is a decision-making process that plays an important role in most manufacturing and service industries (Pinedo 1995). The hybrid flow line (HFL) scheduling problem is defined in the literature, e.g., Kianfar et al. (2012) and Gómez-Gasquet et al. (2012). In HFL, there are g stages and there is at least one stage with more than one machine where the jobs arrive continuously during time and pass the stages sequentially from stage one through g with the same order. If the jobs skip some stages, it is called flexible flow line (FFL) scheduling problem, e.g., Kurz and Askin (2004), Quadt and Kuhn (2005) and Kia et al. (2010).

Salvador (1973) proposed for the first time a definition of the HFL problem for minimizing makespan. He presented a branch and bound method to solve the problem. A double-stage hybrid flow shop problem, with one machine in stage two, is examined for minimizing makespan by Gupta (1988). He proved that the problem was NP-hard and developed a heuristic rule for it. Therefore, FFL is NP-hard. Sawik (1993) proposed a heuristic rule to minimize makespan for a limited buffer FFL problem and later proposed a new rule for the same problem with no in-process buffer (Sawik 1995). Kia et al. (2010) proposed two new scheduling rules with sequence-dependent setup times (SDST) considering non-zero job arrival times for a dynamic FFL problem. Although several papers have been written on the extent of hybrid flow shop and hybrid flow line, most of them are limited to a special case of a double stage (e.g., Gupta 1988; Guinet et al. 1996) or to a particular framework of machines in each stage (e.g., Kochhar and Morris 1987; Sawik 1993, Sawik 2002; Mirabi et al.

2014; Maleki-Darounkolaei et al. 2012). The papers that were in FFL with non-zero job arrival times focused on the heuristic rules that were rarely seen, except Kia et al. (2010). Jolai et al. (2012) proposed a novel hybrid meta-heuristic rule in FFL with SDST. A new dispatching rule for two-stage FFL is proposed by Li et al. (2013). Finally, a comprehensive survey in scheduling problems is presented by Allahverdi (2015) and also Neufeld et al. (2016).

In this paper, scheduling rules are studied to solve a dynamic flexible flow line scheduling problem with SDST using simulation. We used genetic programming to create new composite dispatching rules and a discrete-event model that examined the performance of scheduling rules in terms of mean flow time and mean tardiness.

The rest of the paper is organized as follows. The definition of the problem is given in "Problem definition". "Problem mathematical modeling" introduces a mathematical model for considering the problem. The next section "Scheduling rules" presents those rules adapted from the literature. The genetic programming framework that is illustrated for generating composite dispatching rules is represented in "Genetic programming". The next section is the "Simulation model", and in "Experiment design" the details of experiments are designed and presented for scheduling rules. "Experiment results" provide the results and analyses, and the final section is the "Conclusion".

Problem definition

It is possible to find dynamic flexible flow line (DFFL) configuration in different studies such as Kurz and Askin (2003) and Kia et al. (2010). In this paper, a DFFL system is developed. The DFFL system is presented on the basis of the following assumptions:

- The number of stages is g and also there is at least one stage with more than one machine.
- All jobs visit all stages through stage g, while skipping some stages is possible.
- Machines that are placed in each stage are identical.
- All machines are always available.
- Preemption is not allowed.
- There is no buffer constraint.
- A machine can process at most one job at a time and a job can be processed by at most one machine at a time.
- No job can be processed in one stage, except when the processing had been completed on the previous stage.
- All jobs are not in the system from the beginning, but they can continuously enter over time.
- The sequence-dependent setup time is assumed for every job on each machine.
- The setup time of every job on each machine is sequence dependent.
- When a job comes into the system, its characteristics, i.e., processing times in each stage, setup times and due dates, are identified.
- There is no priority between jobs.
- There is no machine breakdown.

Problem mathematical modeling

In this section, the problem mathematical model is presented. The problem model is 0–1 mixed integer linear programming.

Notation

n: Number of jobs.
i, j: Index of the number of job, $i, j = 1, 2, \ldots, n$.
g: Number of stages in the shop.
l: Index of the stage, $l = 1, 2, \ldots, g$.
m^l: Number of parallel machines at stage l.
k: Index of the machine, $k = 1, 2, \ldots, m^l$.
r_i: Arrival time of job i.
d_i: Due date of job i.
p_i^l: Processing time of job i at stage l.
s_{ij}^l: Setup time from job i to job j at stage l.
C_i^l: Completion time of job i at stage l.
T_i: Tardiness of job i.
F_i: Flow time of job i.
X_{ijkl}: 1 if job j is processed immediately after job i on machine k at stage l, otherwise 0.

Mathematical model

$$\text{Minimize} \quad \alpha\bar{F} + (1-\alpha)\bar{T}. \tag{1}$$

Subject to:

$$\bar{F} = \frac{1}{n}\sum_{i=1}^{n} F_i \quad \forall i, \tag{2}$$

$$\bar{T} = \frac{1}{n}\sum_{i=1}^{n} T_i \quad \forall i, \tag{3}$$

$$\sum_{k=1}^{m^l}\sum_{i=0}^{n} X_{ijk}^l = 1 \quad \forall l, j; j \neq 0, \tag{4}$$

$$\sum_{k=1}^{m^l}\sum_{j=1}^{n+1} X_{ijk}^l = 1 \quad \forall l, i; i \neq n+1, \tag{5}$$

$$\sum_{j=1}^{n+1} X_{0jk}^l = 1 \quad \forall l, k, \tag{6}$$

$$\sum_{i=0}^{n} X_{i(n+1)k}^{l} = 1 \quad \forall l,k, \tag{7}$$

$$X_{iik}^{l} = 0 \quad \forall l,k,i, \tag{8}$$

$$\sum_{i=0}^{n} X_{ijk}^{l} = \sum_{i=1}^{n+1} X_{jik}^{l} \quad \forall l,k,j; j \neq 0; j \neq n+1, \tag{9}$$

$$C_i^0 = r_i \quad \forall i, \tag{10}$$

$$C_j^1 \geq r_j + p_j^1 + \sum_{k=1}^{m^1} \sum_{i=0}^{n} s_{ij}^1 X_{ijk}^1 \quad \forall j, \tag{11}$$

$$C_j^l - C_i^l \geq p_j^l + s_{ij}^l + M\left(\left(\sum_{k=1}^{m^l} X_{ijk}^l\right) - 1\right) \quad \forall l,k,i,j; i \neq j, \tag{12}$$

$$C_j^l - C_j^{l-1} \geq p_j^l + \sum_{k=1}^{m^l} \sum_{i=0}^{n} s_{ij}^l X_{ijk}^l \quad \forall l,j; i \neq j, \tag{13}$$

$$C_i^l \geq 0 \quad \forall l,i, \tag{14}$$

$$T_i \geq C_i^g - d_i \quad \forall i, \tag{15}$$

$$T_i \geq 0 \quad \forall i, \tag{16}$$

$$F_i \geq C_i^g - r_i \quad \forall i, \tag{17}$$

$$X_{ijk}^l \text{ is binary} \quad \forall l,k,i,j. \tag{18}$$

Eq. (1) shows the linear convex combination of the dual criteria problem. The objective function is minimization of mean flow time and mean tardiness. The constraints (2) and (3) calculate the mean flow time and mean tardiness value, respectively. The constraints (4) and (5) guarantee assigning only one job to each sequence position at each stage. Furthermore, the constraints (6) and (7) guarantee assigning one job to the first and last sequence position on each machine at each stage, respectively. It is clear that job 0 and job $(n + 1)$ are not real jobs and just stated for formularization. The constraint (8) ensures that each job at each stage is processed once. The constraint (9) forces consistent sequence at each stage. The constraint (10) ensures that job processing cannot be started before release time of the job at the first stage. The constraint (11) forces that just at the first stage, the completion time for each job cannot be less than the sum of the release, processing time and setup time of that job. The constraint (12) states that at each stage on the particular machine, starting the processing of the next job before completing the previous job is not possible. The constraint (13) states that for each job, its processing at the next stage cannot be started before completing it at the previous stage. The constraint (14) states that completion time for each job at each stage is non-negative. The constraint (15) determines the tardiness value for each job and constraint (16) states that the tardiness is non-negative. The constraint (17) determines the flow time value for each job and, finally, the constraint (18) shows that the problem variables are binary.

Scheduling rules

Generally, whenever a machine is free, a job with the most priority level is chosen for processing among all existing jobs in the queue. In a research on Scheduling rules by Panwalkar and Iskander (1977), the following classification is proposed:

- *Simple priority rules* (*SPR*): These rules usually consist of just one parameter and are suitable for single-objective problems such as process time and due date.
- *Composite dispatching rules* (*CDR*): These rules consist of the application of a combination of several SPRs, and when the machine becomes free then this CDR evaluates the queue and then chooses a job with the most priority level for processing on the machine.

If the CDR is made well, then it is proper for solving real multi-objective problems. In the literature, e.g., Barman (1997), it is clear that CDRs have a better performance than SPRs. Furthermore, Jayamohan and Rajendran (2000) stated that there were no rules with a good performance considering flow time and due date. So, we intend to indicate the efficiency of the proposed new CDRs and also compare them with the six scheduling rules in Kia et al. (2010) that are presented by considering two objectives of mean flow time and mean tardiness for the DFFL environment. The six scheduling rules in Kia et al. (2010) that are adapted for this study are as follows:

Earliest modified due date (EMDD): In this rule at each stage whenever a machine becomes free, a job is to be chosen that has the highest priority considering the earliest modified due date among all jobs waiting in the queue for processing. The modified due date of job i on the stage of q at the time of t is calculated as follows:

$$\max\left\{0, d_i - t - \sum_{l=q}^{g} p_i^l\right\}. \tag{19}$$

Wilkerson and Irvin's rule (W&I): According to this scheduling rule at each stage whenever a machine is free, a job with the highest priority is chosen between two jobs i and j waiting in the queue for processing. This priority is stated by Eq. (20). If Eq. (20) is true, then the job with a shorter processing time is selected; otherwise, the job with an earlier due date will be selected. In fact,

W&I rule uses both SPT and EDD according to the system status:

$$t + \max\{p_i^l, p_j^l\} > \max\{d_i, d_j\}. \quad (20)$$

Hybrid shortest processing time and cyclic heuristics (HSPTCH): In this rule at each stage whenever a machine is free at the first step, jobs waiting in the queue for processing on the machine are arranged according to SPT. In the second step, a job that has minimum completion time on the machine relative to the other jobs waiting in the queue, according to the sequence-dependent setup time of that job, is allocated to the machine.

Hybrid least work remaining and cyclic heuristics (HLWKRCH): In this rule at each stage whenever a machine is free at the first step, jobs waiting in the queue for processing on the machine are arranged according to the least total remaining process time. In the second step, jobs possessing minimum completion time on the machine, in relation to the other jobs waiting in the queue, is allocated to the machine according to the sequence-dependent setup time of that job.

Hybrid earliest modified due date and cyclic heuristics (HEMDDCH): In this rule at each stage whenever a machine becomes free at the first step, jobs waiting in the queue for processing in the machine are arranged according to EMDD. In the second step, a job having the least completion time on the machine, in relation to the other jobs waiting in the queue, is allocated to the machine, according to the sequence-dependent setup time of that job.

Hybrid Wilkerson & Irvin and cyclic heuristic (HW&ICH): In this rule at each stage whenever a machine is free at the first step, jobs waiting in the queue for processing in the machine are arranged according to *W&I*. In the second step, a job having the least completion time on the machine in relation to the other jobs waiting in the queue is allocated to the machine according to the sequence-dependent setup time of that job.

In this paper, we focus on a computational method to make an effective CDR to solve a DFFL problem by a suitable algebraic combination of SPRs, but due to the width of the operator and parameter space, CDR's efficiency evaluation is very difficult in comparison with applying SPRs manually. So, we used genetic programming to evaluate it.

Genetic programming

Genetic programming (GP) is one of the evaluation computing methods based on the survival and reproduction principle (Koza 1992). Each individual, i.e., computer program, is a syntax tree in the random initial population produced in GP including a set of function and terminals; thus, it is essential to select the function and terminal set accurately to create proper CDRs for solving DFFL problems. The function and terminal set and GP parameter setting are stated in the two following subsections.

Function and terminal set

There are various function and terminal sets that can affect the results' quality and efficiency. Each of these terminals includes a dispatching rule that only a few of them are used due to the reduction of the search space.

The proposed terminal set in this study is summarized in Table 1. The proposed function set is also stated in Table 2. In this table, ADF is a sub-tree of the main tree with a variable size like the main tree. In fact, ADF is a function through which GP can generate proper subroutines dynamically. The results of Koza (1994) indicate that GP has better performance in comparison with GP without ADF for solving the same optimization problem.

GP parameter setting

Table 3 shows the GP parameter values. These values are tuned through a large number of experiments. Ramped half and half method, applied by quite a few researchers, e.g., Koza (1992) and Tay and Ho (2008), is used to generate the initial population. This method bisects the initial population.

It generates the first half randomly with a maximum depth of 5 and the second half with a variable depth between 1 to maximum depth. The population (rules) size is 100 and we generate it 200 times. We maintain variation via crossover, mutation, and the creation type ramped half and half. At each time, we arrange the generated population based on the performance measurement and then copy the four best rules in the following population to be preserved and not to be deleted in the next generation. The information of parameter values is summarized in Table 3.

Simulation model

Conditions like random arrival times, machine breakdowns and due date changes state a dynamic scheduling environment. The simulation is one of the ways to analyze the dynamic environment. In this paper, the arrival time of jobs is random, so a DFFL environment is present. A developed discrete-event simulation model is presented to evaluate the four best CDRs and the six heuristic rules. We use C++ programming language on the PC with 2.2 MHz CPU and 512 MB RAM.

Table 1 Terminal set

Terminal	Terminal meaning
ReleaseDate	Release date of a job (RD)
DueDate	Due date of a job (DD)
ProcessingTime	Processing time of a job for each operation (PT)
CurrentTime	Current time (CT)
RemainingTime	Remaining processing time of each job (RT)
avgTotalProcTime	Average total processing time of each job (aTPT)

Table 2 Function set

Function	Function meaning
+	Addition
−	Subtraction
×	Multiplication
/	Division
ADF(x1,x2)	Automatically defined function
avgTotalProcTime	Average total processing time

Table 3 Choice of parameter values for GP

Parameter	Parameter values
Population size	100
Number of generations	200
Creation type	Ramped half and half
Maximum depth for creation	5
Maximum depth for crossover	15
Crossover probability	90%
Swap mutation probability	3%
Shrink mutation probability	3%
Number of best rules copy to new generation	4

In the literature, it is common to consider setup times 20–40% of the mean process time. In this study, the process time is a uniform distribution of [20–60], so the setup time on the basis of 20 and 40% of process time follows the uniform distribution [4–12] and [8–24]. The number of stages (g) is fixed and equals 8. The probability of job skip in each stage is defined in three separate levels of (0.00, 0.05 and 0.40) and the occurrence probability of these three levels is equal. The number of machines in the lth stage (m^l) also follows uniform distribution [3–5]. The due date is also calculated according to Naderi et al. (2009) as follows:

$$d_i = r_i + [(p_i + s_i) \times (1 + \text{random} \times 3)], \quad \forall i \in n, \qquad (21)$$

where p_i is the total process time of the job i, s_i the total mean setup times of job i at all stages, r_i the arrival time

Table 4 Simulation parameters

Parameter	Value
Number of jobs (n)	1450
Number of stages in the shop (g)	8
Number of parallel machines at stage l (m^l)	Uniform [3–5]
Processing time of job i at stage l (p_i^l)	Uniform [20–60]
Setup time of job i before job j at stage l (s_{ij}^l)	Uniform between [4–12] and [8–24]
Skipping probability of each job at each stage	0.00, 0.05 and 0.40
Mean of inter-arrival time (a)	9–10

of job i and random a random number with uniform distribution of (0,1). The mean interval time parameter follows the Poisson distribution and is calculated as follows:

$$a = \frac{\mu_p \times \mu_g}{U \times M}, \qquad (22)$$

where μ_p is the mean process time of every job at each stage, μ_g the mean number of non-skipped stages of each job, U the percentage of workshop utilization and M the mean of the total number of machines in the shop. Since μ_g equals $g \times (1 - \mu_{skip})$ and M equals $g \times \mu_m$, we have

$$a = \frac{\mu_p \times (1 - \mu_{skip})}{U \times \mu_m}, \qquad (23)$$

where μ_{skip} is the mean skip probability and μ_m the mean machine number parameter at each stage. In this study, the parameter values are $\mu_p = 40$, $\mu_{skip} = 0.15$, $\mu_m = 4$, $\alpha = 9$ and 10.

There are two events in the system. The type one is when a job comes into the system and type two is when a machine is free as a consequence of a job processing completion. Whenever an event occurs in the system, the developed discrete-event model reschedules the system on the basis of the predefined scheduling rules. Table 4 presents the simulation model parameters of the system. The simulation model includes the following modules:

(I) Initialization.
(II) Job data generation.
(III) Timing.
(IV) Events.
(V) Scheduling.
(VI) Status.
(VII) Report.

Each of these modules has its own particular role in the simulation model and is run as follows:

Table 5 Experimental settings for the scenarios

Scenario	Experimental setting		Purpose of investigation
	Shop utilization percentage (U)	Setup time ratio (%) (s)	
DFFL-I	95	20	Base case—analyze the performance of scheduling rules
DFFL-II	95	20, 40	Analyze the effect of changing setup time ratio
DFFL-III	95, 85	20	Analyze the effect of changing the mean inter-arrival time

Table 6 GP-generated dispatching rules

Rule	Rule expression
GP-1	RD + 5PT + 2aTPT
GP-2	7aTPT + 11PT + 12RD
GP-3	3RD + 2DD + 3aTPT + PT-2RD
GP-4	2DD + 8RD + 2aTPT−5PT

Step 1: Run module (I) and set type of scheduling rule.
Step 2: Run module (II).
Step 3: Run module (III) and set $t = 0$.
Step 4: Run module (IV) to determine the next event and advanced simulation time to the next event time.
Step 5: Run module (V) [according to the selected scheduling rule in step (I)] to schedule/reschedule the system.
Step 6: Run module (VI) to analyze the status of the model.

If the termination condition is not met, go to step 3. If not, go to the next step.

Step 7: Run module (VII) to report the computational results.

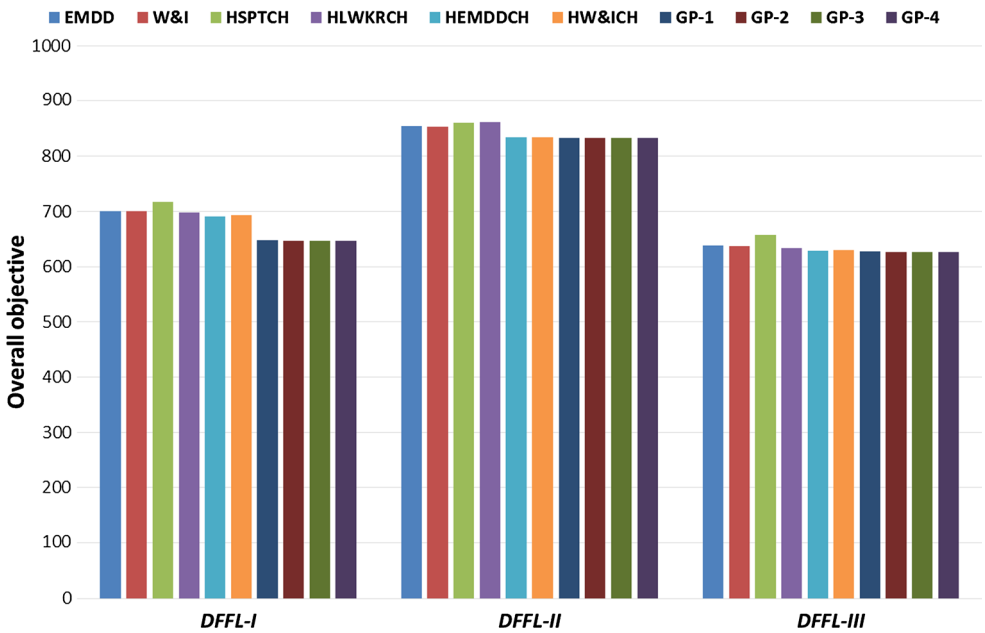

Fig. 1 Comparison of the overall objective function of heuristic algorithms for different scenarios

Table 7 Overall objective function values for different scenarios

	EMDD	W&I	HSPTCH	HLWKRCH	HEMDDCH	HW&ICH	GP-1	GP-2	GP-3	GP-4
DFFL-I	700.9	701.0	717.4	698.2	691.3	693.2	647.5	646.7	647.2	647.1
DFFL-II	854.3	852.8	860.1	861.3	833.9	834.1	832.9	832.7	833.3	833.2
DFFL-III	638.0	637.4	657.0	633.3	628.3	629.4	627.0	626.2	626.9	626.5

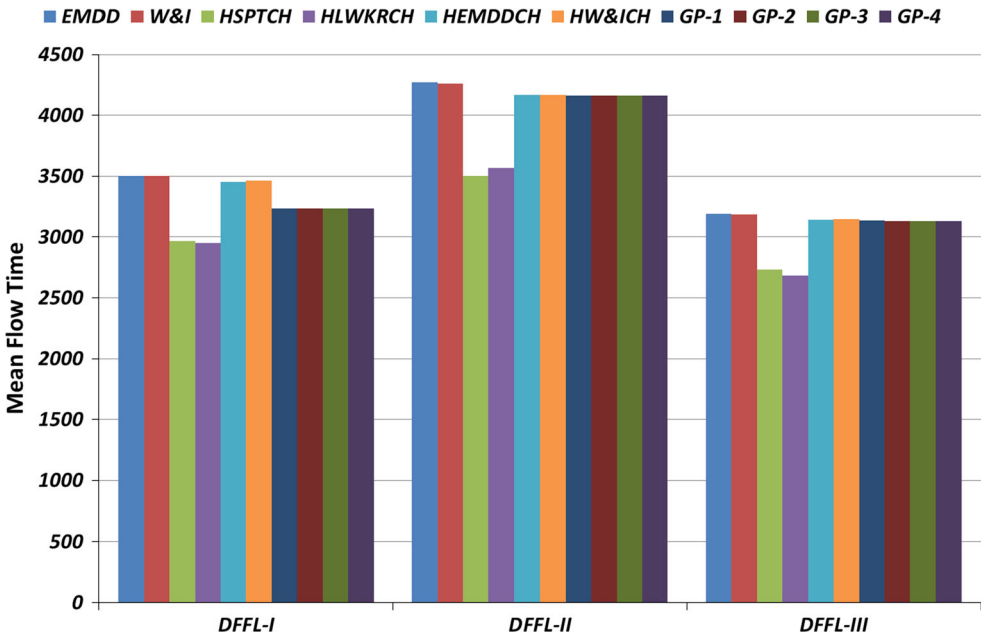

Fig. 2 Comparison of the mean flow time of heuristic rules for different scenarios

Table 8 Mean flow time values for different scenarios

	EMDD	W&I	HSPTCH	HLWKRCH	HEMDDCH	HW&ICH	GP-1	GP-2	GP-3	GP-4
DFFL-I	3502.7	3502.9	2966.8	2947.8	3454.7	3464.2	3235.5	3231.3	3234.2	3233.3
DFFL-II	4269.3	4261.8	3500.7	3567.1	4167.2	4168.1	4162.2	4161.5	4164.3	4164.0
DFFL-III	3187.9	3185.1	2732.4	2683.6	3139.5	3145.2	3132.9	3129.1	3132.7	3130.3

Experiment design

To examine the performance of the four proposed CDRs and six adapted scheduling rules from the literature, three exactly the same scenarios are defined Kia et al. (2010). In each of these scenarios, to measure the performance, it is necessary to use the system data in the steady state. So we warm up the system with 1000 jobs and then for calculating the performance, the results of the next 450 jobs are applied. The information on the scenarios is outlined in Table 5.

Experiment results

The simulation experiment results are examined to determine the effect of factors on the performance measurement by analysis of variance (ANOVA). In scenario 1, there is only one factor with ten levels that are scheduling rules while, in the two other scenarios, there are two factors. We use the one-way ANOVA in scenario I and two-way ANOVA in the other two scenarios. All of the experiments are done at a significant level of 5%. In the null hypothesis (H_0), all of the means are the same, while, in the alternative hypothesis (H_1), at least two of the means are significantly different. The four elite scheduling rules are obtained from GP at each iteration. Table 6 shows that the best four evolved scheduling rules after ten iterations of GP are simplified by algebraic operations. For example, the release date of a job (RD) plus five times the processing time of a job for each operation (PT) plus two times the average total processing time of each job (aTPT) are defined as the first proposed rule (GP-1) that was achieved from genetic programing.

As mentioned above, all experiments are conducted in ten iterations and the results of the scenarios are stated in the following three subsections considering the performance measurements.

Scenario results for the overall objective function

In Fig. 1, the performances of the scheduling rules in different scenarios are indicated according to the defined

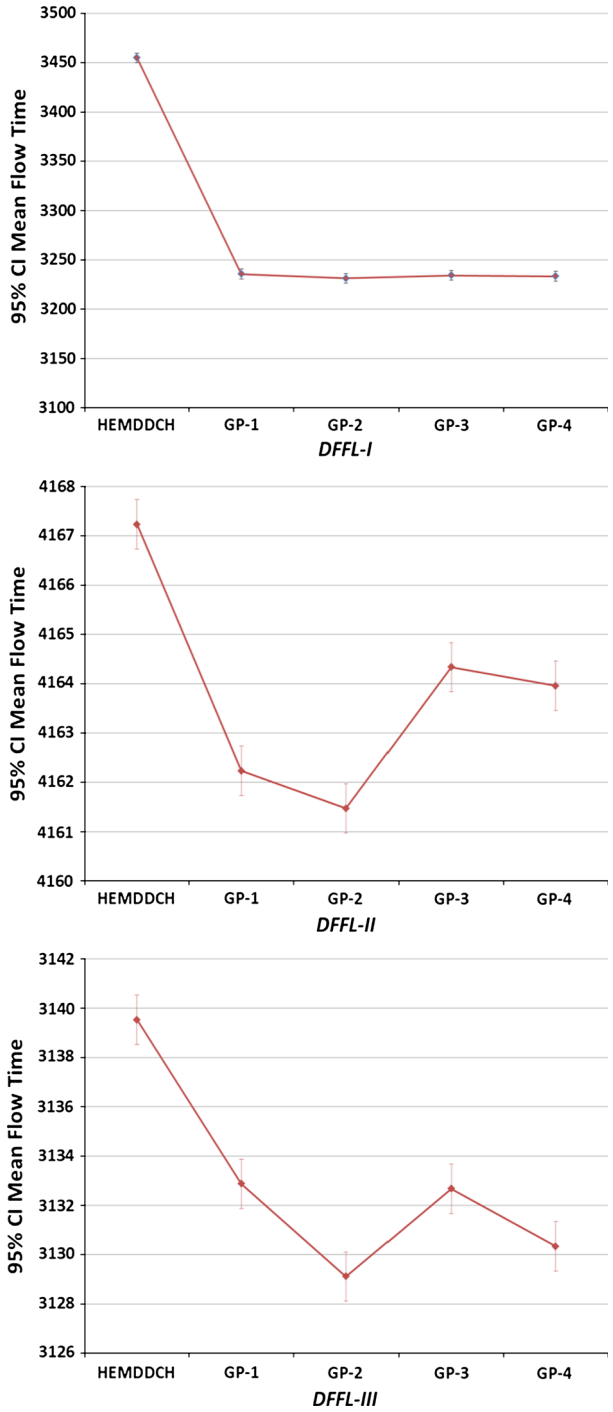

Fig. 3 Interaction plot with 95% confidence interval for different scenarios–mean flow time

overall objective function in Eq. (1). It is clear from Fig. 1 that in all the scenarios, the four evolved CDRs have a better performance than the six adapted rules. The proposed rules achieve better performance than the adapted rules. The GP-2 reaches the best result among all rules. EMDDCH and HW&ICH achieve the best and HSPTCH the worst result among the six adapted rules. The details are stated in Table 7.

Scenarios' results for the mean flow time

The experimental results for the mean flow time are indicated in Fig. 2. HLWKRCH and HSPTCH achieve the best results among the six adapted rules, while the other four scheduling rules achieve the worst results for the reason that four scheduling rules only minimize objectives related to the due date like mean tardiness, so they perform weakly in minimizing mean flow time.

Table 8 indicates the details of Fig. 2. Although HLWKRCH provides the best solution on the basis of the mean flow time, the four proposed rules achieve better performance in terms of the mean flow time than EMDDCH that achieves the best solution among the six adapted rules in the overall objective function.

Figure 3 indicates the performance of the four proposed rules and EMDDCH with 95% confidence interval (CI) for the mean flow time. It is clear from Fig. 3 that in scenario I (DFFL-I), the GP-2 achieves the best result among the proposed rules and there is no overlap between its CI and others. Although GP-2 achieves the best solution in scenarios II and III (DFFL-I and III), there is an overlap between CI of GP-2 and GP-1 in scenario II and CI of GP-2 and GP-4 in scenario III; so, no significant difference is found between them.

Scenario's results for mean tardiness

Table 9 indicates the results of mean tardiness for different scenarios. Two HSPTCH and HLWKRCH rules reach the worst solution for the reason that these two rules only minimize the mean flow time, so they perform weakly in minimizing the mean tardiness. As mentioned above, EMDCH obtained the best solution for the overall objective function among the six adapted rules from the literature.

Table 9 Mean tardiness values for different scenarios

	EMDD	W&I	HSPTCH	HLWKRCH	HEMDDCH	HW&ICH	GP-1	GP-2	GP-3	GP-4
DFFL-I	0.518	0.525	155.0	135.8	0.503	0.505	0.502	0.501	0.503	0.503
DFFL-II	0.585	0.611	200.0	184.9	0.574	0.590	0.570	0.561	0.568	0.569
DFFL-III	0.495	0.505	138.1	120.7	0.499	0.506	0.500	0.495	0.498	0.496

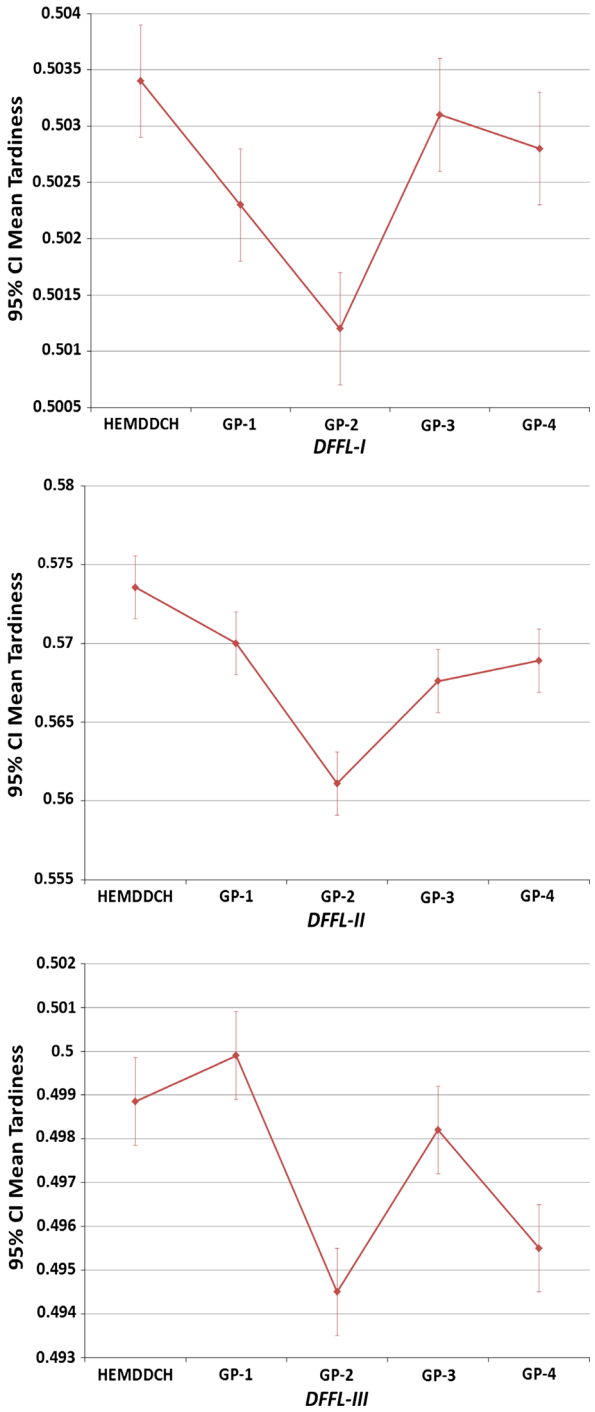

Fig. 4 Interaction plot with 95% confidence interval for different scenarios–mean tardiness

Figure 4 indicates the mean tardiness values of the four proposed rules and EMDDCH with 95% CI for different scenarios. In scenarios I and II (DFFL-I & II), the GP-2 has no overlap with others, so the GP-2 is the best rule between them, but in scenario III (DFFL-III), there is no significant difference between GP-2 and GP-4.

Conclusion

In this paper, we used the GP framework to obtain proper and effective CDRs for solving a DFFL problem. Finally, we created four evolved CDRs and proposed them for solving the DFFL problem. Experimental results indicated that the four proposed CDRs with 95% CI obtained a better solution in comparison with selected scheduling rules from the literature.

For future research, it is possible to consider the variation of the terminal set and ADF to develop the GP framework. It is also possible to use GP to find proper and effective CDRs for different objectives. Furthermore, it is useful to apply GP for other scheduling environments such as job shop or open shop and dynamic assumptions like machine breakdowns.

References

Allahverdi A (2015) The third comprehensive survey on scheduling problems with setup times/costs. Eur J Oper Res 246(2):345–378

Baker KR (1974) Introduction to sequencing and scheduling. Wiley, New York

Barman S (1997) Simple priority rule combinations: an approach to improve both flow time and tardiness. Int J Prod Res 35(10):2857–2870

Gómez-Gasquet P, Andrés C, Lario FC (2012) An agent-based genetic algorithm for hybrid flow shops with sequence dependent setup times to minimise makespan. Expert Syst Appl 39(9):8095–8107

Guinet A, Solomon MM, Kedia PK, Dussauchoy A (1996) A computational study of heuristics for two-stage flexible flow shops. Int J Prod Res 34(5):1399–1415

Gupta JN (1988) Two-stage, hybrid flowshop scheduling problem. J Oper Res Soc 39(4):359–364

Jayamohan MS, Rajendran C (2000) New dispatching rules for shop scheduling: a step forward. Int J Prod Res 38(3):563–586

Jolai F, Rabiee M, Asefi H (2012) A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times. Int J Prod Res 50(24):7447–7466

Kia HR, Davoudpour H, Zandieh M (2010) Scheduling a dynamic flexible flow line with sequence-dependent setup times: a simulation analysis. Int J Prod Res 48(14):4019–4042

Kianfar K, Ghomi SF, Jadid AO (2012) Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA. Eng Appl Artif Intell 25(3):494–506

Kochhar S, Morris RJ (1987) Heuristic methods for flexible flow line scheduling. J Manuf Syst (4):299–314

Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection, vol 1. MIT Press

Koza JR (1994) Genetic programming II: automatic discovery of reusable subprograms. Cambridge, MA, USA

Kurz ME, Askin RG (2003) Comparing scheduling rules for flexible flow lines. Int J Prod Econ 85(3):371–388

Kurz ME, Askin RG (2004) scheduling flexible flow lines with sequence-dependent setup times. Eur J Oper Res 159(1):66–82

Li ZT, Chen QX, Mao N, Wang X, Liu J (2013) Scheduling rules for two-stage flexible flow shop scheduling problem subject to tail group constraint. Int J Prod Econ 146(2):667–678

Maleki-Darounkolaei A, Modiri M, Tavakkoli-Moghaddam R, Seyyedi I (2012) A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. J Ind Eng Int 8(1):1–7

Mirabi M, Ghomi SF, Jolai F (2014) A novel hybrid genetic algorithm to solve the make-to-order sequence-dependent flow-shop scheduling problem. J Ind Eng Int 10(2):1–9

Naderi B, Zandieh M, Roshanaei V (2009) Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness. Int J Adv Manuf Technol 41(11–12):1186–1198

Neufeld JS, Gupta JN, Buscher U (2016) A comprehensive review of flowshop group scheduling literature. Comput Oper Res 70:56–74

Panwalkar SS, Iskander W (1977) A survey of scheduling rules. Oper Res 25(1):45–61

Pinedo M (1995) Scheduling theory, algorithms, and systems. Prentice-Hall, New York

Quadt D, Kuhn H (2005) A conceptual framework for lot sizing and scheduling of flexible flow lines. Int J Prod Res 43(11):2291–2308

Salvador MS (1973) A solution to a special class of flow shop scheduling problems. Symposium on the theory of scheduling and its applications. Springer, Berlin Heidelberg, pp 83–91

Sawik TJ (1993) A scheduling algorithm for flexible flow lines with limited intermediate buffers. Appl Stoch Models Data Anal 9(2):127–138

Sawik T (1995) Scheduling flexible flow lines with no in-process buffers. Int J Prod Res 33(5):1357–1367

Sawik T (2002) Balancing and scheduling of surface mount technology lines. Int J Prod Res 40(9):1973–1991

Tay JC, Ho NB (2008) Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. Comput Ind Eng 54(3):453–473