A Fitness-based Selection Method for Pareto Local Search for Many-Objective Job Shop Scheduling

Atiya Masood, Gang Chen, Yi Mei, Harith Al-Sahaf, Mengjie Zhang
School of Engineering and Computer Science, Victoria University of Wellington
Wellington, 6012, New Zealand
{masoodatiy, aaron.chen, yi.mei, harith.al-sahaf, mengjie.zhang}@ecs.vuw.ac.nz

***Abstract*—Genetic programming (GP) is considered the most popular method for automatically discovering and constructing dispatching rules for scheduling problems. Pareto Local Search (PLS) is a simple and effective local search method for tackling multi-objective combinatorial optimization problems. Researchers have studied the application of PLS to multi-objective evolutionary algorithms (MOEAs) with some success. In fact, by hybridizing global search with local search, the performance of many MOEAs can be noticeably improved. Despite its preliminary success, the practical use of PLS in GP is relatively limited. In this study, our aim is to enhance the quality of evolved dispatching rules for many-objective Job Shop Scheduling (JSS) through hybridizing GP with PLS techniques and designing an effective selection mechanism of initial solutions for PLS. In this paper, we propose a new GP-PLS algorithm that investigates whether the fitness-based selection mechanism for selecting initial solutions for PLS can increase the chance of discovering highly effective dispatching rules for many-objective JSS. To evaluate the effectiveness of our new algorithm, GP-PLS is compared with the current state-of-the-art algorithms for many-objective JSS. The experimental results confirm that the proposed method can outperform the four recently proposed algorithms because of the proper use of local search techniques.**

I. Introduction

Job Shop Scheduling (JSS) [1] is a significant scheduling problem which has a wide range of applications in many industries such as manufacturing and cloud computing. A JSS problem deals with a group of tasks or jobs by using different resources or machines. The goal of a JSS problem is to design a schedule where jobs can be processed in an optimal way through a set of machines so that the predefined objectives are optimized.

There is also a growing interest in industries to tackle problems with many objectives [2], [3]. Some studies consider that JSS by nature several potentially conflicting objectives [4], such as makespan, mean tardiness, and mean flowtime. The many-objective optimization problems (MaOPs) require to find a set of non-dominating solutions, known as the Pareto-front and the many-objective optimization algorithms provide a good representative approximation of the Pareto-front.

JSS has been proven to be *NP-hard* [1]. Thus, heuristics become a promising method to obtain near-optimal solutions within a limited time budget. There have been a variety of heuristics proposed for JSS [5], [6]. The *dispatching rule* is a branch of heuristics which has been successfully used in JSS due to its flexibility, scalability and quick response to the dynamic environment. So far, there has been a variety of dispatching rules designed for different JSS variants [5], [7].

A dispatching rule is a rule to decide which job in the queue is to be processed by the idle machine at each decision point. A dispatching rule can be seen as a *priority function*, which assigns priority to each waiting job. The job then will be selected based on the priority value. In designing a dispatching rule for JSS, there are two main issues. First, dispatching rules are time-consuming to design manually, especially for optimizing multiple potentially conflicting objectives which are frequently demanded in a manufacturing environment. Second, the behavior of a dispatching rule can vary from one scenario to another. For example, the rule which is good at minimizing the mean flowtime may perform poorly in minimizing the maximal tardiness [5].

In order to deal with these issues, hyper-heuristics have been adopted to design the dispatching rules automatically, i.e., optimize the priority function. A comprehensive survey of methods for automatically designing dispatching rules is given in [8]. Genetic Programming (GP) has been a promising approach for designing dispatching rules automatically thanks to its ability of evolving priority functions with flexible representation [9], [10]. The hyper-heuristic approach that uses GP to solve JSS problems is known as GP-based Hyper-Heuristic (GP-HH) [8], [9]. However, GP tends not to be very good at locally tuning a solution [11]. Applying local search to GP [11] is major technique to improve the quality of dispatching rules.

Many researchers [12], [13] combine the local search with MOEAs to avoid the MOEAs converging too fast to the Pareto-optimal front. PLS is an effective local search method for tackling multi-objective combinatorial optimization problems such as JSS [12]. Researchers have studied the application of PLS to MOEAs with some success [12]. In fact, by hybridizing global search (MOEAs) with local search, the performance of many MOEAs can be noticeably improved [12], [13]. Such a hybrid algorithm is often referred to as memetic algorithms. Despite the preliminary success of PLS, the practical use of PLS on many-objective JSS is relatively limited. Based on our survey in Section II, there is only one study that uses PLS in GP-HH for many-objective JSS [14]. The study in [14] used a random mechanism to select PLS solutions. Although random sampling is simple, it may fail to select the solutions with the greatest potential to be improved by PLS.

In this paper, we aim to propose an effective selection mechanism for selecting the initial solutions for PLS. To

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

achieve this goal, a new fitness-based selection mechanism is proposed in this study. Our proposed fitness-based selection mechanism adopts convergence and diversity selection strategies with penalty parameter which can properly balance convergence and diversity criteria.

The research objectives of this paper are to: 1) develop a new fitness-based selection mechanism in GP-PLS-F (GP-PLS-Fitness), 2) investigate whether the inclusion of fitness-based selection mechanism in GP-PLS-F can increase the chance of discovering highly effective dispatching rules for many-objective JSS, and 3) verify the effectiveness of GP-PLS-F by comparing it with four existing many-objective algorithms on a group of benchmark JSS problems.

The rest of the paper is organized as follows: Section II gives the background, including the problem description and related work. Section III describes the proposed algorithm (GP-PLS-F). Section IV provides the experimental design. Section V covers the results and discussions. Finally, Section VI presents the conclusions and suggestion on future work.

II. Background

In this section, the problem descriptions for JSS and many-objective optimization are introduced and followed by the related works.

A. Problem Description

1) Job Shop Scheduling: In a JSS problem, we are given a set of N jobs and a set of M machines. Each job j_i has a sequence of m operations to be performed, i.e., $\{o_i^1, \ldots, o_i^m\}$ where $1 \leq i \leq N$. Each operation o_i^k has a fixed processing time $p_i^k > 0$ and has to be processed on a specific machine $m_i, 1 \leq i \leq M$. Any solution to a JSS problem has to comply with important rules, as described below.

- Each machine can only process at most one operation of any job at any given time.
- All operations are *non-preemptive*. This means that once an operation starts its processing on a machine, it cannot be interrupted by other operations during this period of time.
- All the M machines in the job shop are *immediately available* to process new operations whenever they become idle.

2) Many-Objective Optimization: The quality of the schedules will then be evaluated with respect to a number of objectives $\boldsymbol{f} = (f_1, f_2 \ldots, f_D)$, where D is the number of objectives.

Without losing generality, we assume that all the dimensions of $\boldsymbol{f}$ are minimized. Here, we considered $D \geq 4$, i.e., there are *four or more* objectives. In many-objective JSS with two schedules Δ_1 and Δ_2, it is said that Δ_1 *dominates* Δ_2 if and only if

$$\forall i, 1 \leq i \leq D, f_i(\Delta_1) \leq f_i(\Delta_2), \tag{1}$$

and

$$\exists i, f_i(\Delta_1) < f_i(\Delta_2). \tag{2}$$

If a solution Δ^* is not dominated by any other solution then it is called a *Pareto-optimal* solution. The set of all Pareto-optimal solutions jointly forms the Pareto-front in the objective space and the Pareto set in the solution space.

B. Related Works

GP is considered the most popular method for discovering and constructing dispatching rules for scheduling problems [15], [11]. Previous studies have shown that GP has been successfully used to evolve effective dispatching rules for JSS problems automatically [16], [9].

GP shows its effectiveness not only in single-objective JSS problems, but can also evolve useful rules for multi-objective and many-objective JSS problems [17], [4]. Previous studies have shown that the exploitation ability of GP is limited to the population size but applying local search can improve its exploitation ability [11], [14]. However, GP-based methods for multi-objective and many-objective JSS overlooked the opportunity of enhancing the quality of evolved rules through local search.

PLS is very effective for tackling *NP-hard* multi-objective JSS problems [18]. In particular, [13] showed that suitable candidates for local search should be carefully selected based on certain scalarization mechanisms. [19] applied PLS and improved the overall quality of the evolved Pareto-front. However, only one research work [14] has been studied PLS in GP-HH for many-objective JSS. The study in [14] used a random selection mechanism for PLS solutions. Although random sampling is simple, it may not select the solution of all the sub-regions of the objective space and mainly affects the solution's diversity. Further, this study may not show clearly how the local search contributes to the final Pareto solutions. To address this limitation, we investigate the effectiveness of the fitness-based selection mechanism for the solutions of PLS in GP-HH in our current study. The investigation in this paper is expected to inspire many future studies on PLS in GP-HH for many-objective JSS.

III. Proposed Algorithm

This section describes the general framework of the proposed algorithm which combine GP with PLS.

A. Representation of Rules

In GP, dispatching rules can be represented as a GP tree. In line with the tree-based representation of dispatching rules, the function set in GP includes $\{+, -, \times, /\}$ (the protected division operator returns 1 if the denominator is zero), the 2-argument "$\min$" and "$\max$" operators and the 3-argument "If" operator that returns the second argument if the first argument is positive, and the third argument otherwise. The terminals are summarized in Table I.

Consider the popular manually-designed 2PT+WINQ+NPT rule [5] in Fig. 1 where the terminals in the tree are {2, PT, WINQ, NPT} (please refer to Table I for a summary of all terminal types used in this paper) and the functions are $\{+, *\}$.

TABLE I: Terminal set for GP for JSS.

Attribute	Symbol
Processing time of the operation	PT
Processing time of the next operation	NOPT
Ready time of the next machine	NMRT
Work Remaining	WKR
Number of operation remaining	NOW
Work in the next queue	WINQ
Number of operations in the next queue	NOINQ
Flow due date	FDD
Due Date	DD
Weight	W
Number of operations in the queue	NOIQ
Work in the queue	WIQ
Ready time of the machine	MRT

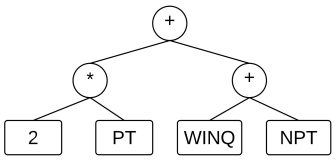


Fig. 1: The GP tree representation of the 2PT+WINQ+NPT rule.

B. General framework of GP-PLS-F

The framework of GP-PLS-F is shown in Algorithm 1. GP-PLS-F starts with the initialization by using the ramped-half-and-half method. To evaluate the quality of dispatching rules in terms of each objective (lines 1 and 6 of Algorithm 1), it is applied to a set of JSS training instances $\boldsymbol{I}_{train}$ to generate schedules for them. Then, for each objective, the quality of a rule p is defined as the average objective value of the schedules generated across all training instances.

C. Pareto Local Search

The PLS is described in lines 11 to 23 of Algorithm 1. First, PLS selects K individuals based on their fitness values from the archive ($\boldsymbol{P_k}$). Then, for each p in the archive, the restricted mutation is used to generate a neighboring rule around p. A maximum of $step_{max}$ such neighbors can be generated, and the new neighbors (p_{new}) are compared with p. If the new neighbor (p_{new}) is better than p, then it is added into $\boldsymbol{P_{best}}$. This local search mechanism will help to enhance the exploitation ability and explore promising rules in the proximity of each selected candidate rule. The PLS steps are discussed here:

1) Fitness-based selection: In GP-PLS-F, we used the following steps for the selection of K initial solutions.

i. **Determines search direction of each solution:** In this study, the decomposition-based approach is used to split the objective space into a number of independent sub-regions according to a set of reference points. This decomposition of the objective space determines the appropriate search direction of each solution. For example, the two solutions, s_1 and s_2, have identical search directions if they are associated with the same reference point. The reference points are positive and appear inside the first quadrant, therefore, population $\boldsymbol{R}_g$ is normalized (as seen in the line 6 of Algorithm 2) before it is partitioned into *2N* sub-populations (R_{g1}, R_{g2},. . . ,R_{g2N}) by associating each individual with its closest reference point. The association of r is described in Fig. 2.

The acute angle measures the association between an individual and the reference points [20], [21]. In our proposed algorithm, the vector angle reflects the similarity of search directions between two individuals. The angle information between two individuals in the objective space is then later used to maintain the diversity. The acute angle can be calculated as:

$$cos\theta_{i,j} = \frac{r_{i,j} \cdot w_{i,j}}{\parallel r_{i,j} \parallel}, \tag{3}$$

Algorithm 1: The framework of GP-PLS-F.

Input : training set $\boldsymbol{I}_{train}$
Output: A set of non-dominated solutions (rules) $\boldsymbol{P^*}$

1 Initialize of rules and Evaluate the population $\boldsymbol{P_0}$;
2 $g \leftarrow 0$;
3 **while** $g < g_{\max}$ **do**
4 $\boldsymbol{P_{best}} \leftarrow \emptyset$;
5 Apply genetic operators to $\boldsymbol{P_g}$ to generate offspring $\boldsymbol{Q_g}$;
6 **foreach** $Q \in \boldsymbol{Q}_g$ **do** Evaluate rule Q;
7 Combine $\boldsymbol{P_g}$ and $\boldsymbol{Q_g}$ ($\boldsymbol{R_g}$=$\boldsymbol{P_g} \cup \boldsymbol{Q_g}$);
8 $archive$= Solution-selection($\boldsymbol{R_g}$);
9 Select K individuals based on their fitness values from $archive$ and save in $\boldsymbol{P_K}$;
10 /* *Apply the Pareto local search on* $\boldsymbol{P_K}$ */;
11 **foreach** $p \in \boldsymbol{P_K}$ **do**
12 $p_{new} \leftarrow p$;
13 **for** $step = 1 \rightarrow step_{max}$ **do**
14 $p' \leftarrow$ mutate(p); // neighbors
15 evaluate(p');
16 **if** p' *is better than* p_{new} **then**
17 $p_{new} \leftarrow p'$;
18 **end**
19 **if** p_{new} *is better than* p **then**
20 $\boldsymbol{P_{best}} \setminus p$;
21 $(\boldsymbol{P_{best}})_{\boldsymbol{g}} \leftarrow (\boldsymbol{P_{best}})_{\boldsymbol{g}} \cup p_{new}$
22 **end**
23 **end**
24 $g \leftarrow g + 1$;
25 **end**
26 **return** *The non-dominated individuals* $\boldsymbol{P^*} \subseteq \boldsymbol{P}_{\boldsymbol{g}_{\max}}$;

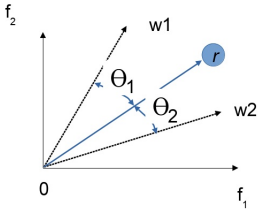


Fig. 2: Example showing how to associate an individual r with a reference points.

Algorithm 2: Solution-selection($\boldsymbol{R}_g$).

Input : A set of non-dominated solutions (rules) $\boldsymbol{R}_g$
Output: archive of selected solutions for neighbourhood

Generate reference points $\boldsymbol{W}$ /* Population Partition
for *j =1 to* $\| \boldsymbol{R} \|$ **do**
  Calculate the ideal point Z_j^{min}=$min_{r \in R}\ f_j(r)$;
  Calculate the worst point Z_j^{max}=$max_{r \in R}\ f_j(r)$
end
for *i = 1 to PopSize* **do**
  $\hat{f(i)}$=$\frac{f_i - Z_j^{min}}{Z_j^{max} - Z_j^{min}}$;
end
foreach $r \in R_g$ **do**
  foreach $w \in W$ **do**
    compute the acute angle $\langle \hat{f(r)}, w \rangle$;
  end
  Assign $\hat{w} = w : argmin_{w \in W}\ \langle \hat{f(r)}, w \rangle$;
  Assign $\theta_r = \langle \hat{f(r)}, w \rangle$;
  save r in $E(\hat{w})$
end
/* fitness of individual from each sub-region */;
foreach $w \in W$ **do**
  foreach $r \in E(\hat{w})$ **do**
    Compute the convergence criteria $C(r)$;
    Compute the diversity criteria $D(r)$;
    Compute the fitness of each individual $FV(r)$ by using equation (6)
  end
end
foreach $w \in W$ **do**
  Select solutions according to the $FV(r)$ add selected solutions($P_1, P_2, \ldots, P_N$) from each subspace into the $archive$
end
return $archive$;

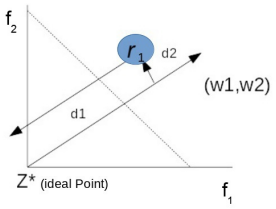


Fig. 3: Distance measure in the context of minimization with respect to a reference direction.

If the individuals $r_{i,j}$ and $w_{i,j}$ have a minimal acute angle among all the reference points ($w_{i,j}$), then $r_{i,j}$ becomes the member of the sub-population $\boldsymbol{R}_{g,k}$.

ii. **Obtain fitness value of each solution:** Once the population $\boldsymbol{R}_g$ is partitioned into $2N$ sub-populations, associated solutions are selected from each subregion using Equation (4).

$$(N_t) = \left(\frac{number\ of\ solution\ from\ each\ subspace}{Total\ population} \right) \times N, \tag{4}$$

where $t = 1, 2 \ldots, 2N$
The selection criteria based on the fitness value (FV) are designed based on two sub-criteria: (1) the convergence criterion (d_1 in Fig. 3) and (2) the diversity criterion (d_2 in Fig. 3). The d_1 is represented by the distance from the solution $(r_{i,j})$ to the ideal point (Z^*) ,i.e., $\| r_{i,j} - Z^* \|$. Similarly, d_2 is represented by the inverse of the acute angle between $(r_{i,j})$ and $w_{i,j}$, i.e., $\theta_{i,j}$. In order to balance between the convergence criterion ($C(r)$ = d_1) and the diversity criterion ($D(r)$ = d_2), the total FV of each individual can be formulated as a scalarization function:

$$FV = d1 + \frac{d2}{\theta_m}, \tag{5}$$

where θ_m is used in equation (5) to normalize $\theta_{i,j}$. This angle normalization process is adopted from RVEA [22] which is meaningful when some of the reference points are densely distributed. As a result, angles between the candidate solutions and the reference points are either extremely small or extremely large.

iii. **Introduces penalty parameter:** It is a good idea to apply high selection pressure on convergence during the exploration phase and push the population toward the Pareto-front of the search process. However, during the exploitation phase, the constant pressure applies to diversity. Therefore, we introduce the penalty parameter $\frac{g}{g_{max}}$ in equation (6) which can regulate the proportion of convergence and diversity information in a good way. Therefore, the new FV is defined as:

$$FV = d1 + \frac{g}{g_{max}} \times \frac{d2}{\theta_m}. \tag{6}$$

where g is a current generation and g_{max} is a maximum number of generations.

iv. **Determine K representatives for Neighborhood exploration:** In the early stage, FV determines the convergence value (d_1) because $g << g_{max}$, therefore the diversity criterion (d_2) ≈ 0. However, when g approaches g_{max}, the penalty parameter gradually increases to emphasize the importance of the diversity criterion $\theta_{i,j}$. After getting the fitness values of each individual, then N solutions are selected and saved in the *archive*. From the *archive*, K solutions are selected based on their fitness values and saved in $\boldsymbol{P_k}$. These solutions are selected for the neighborhood exploration. The value of K is determined by sensitivity analysis in Section IV-D.

2) Neighborhood Solution: In GP-PLS-F, a neighboring rule of any given rule p is obtained by using the restricted mutation operator [14]. When the restricted mutation is applied to rule p, we randomly select a node in p whose corresponding sub-tree has a depth of 2. A randomly generated depth-2 sub-

tree then replaces the selected node and its sub-tree. With the help of this restricted mutation, GP-PLS-F can effectively prevent a new neighboring rule discovered during the local search process from being significantly different from the original rule.

3) Exploration: In the local search algorithm, the exploration strategy determines the size of the neighborhood for exploration. One can either explore the neighborhood entirely (best-improvement) [19] or only partially until the termination criterion is met [19]. In this study, the partial strategy is used since the entire neighborhood is infinitely large. Specifically, we sample a neighbor based on the dominance relation from the neighborhood repetitively until the maximum number of steps ($step_{max}$) is reached, and return the best neighbor sampled so far. The reason for using partial strategy is because it is less computationally expensive than the best improvement strategy, especially for a problem with a large number of features.

4) Comparison: For comparing two rules p' and p_{new} during PLS (e.g. line 16 of Algorithm 1), we consider the replacement strategy [12] which is based on the dominance relation. Whenever we compare two rules p_{new} and p', there are three possible outcomes:

i. if p_{new} dominates p', choose p_{new};
ii. if p' dominates p_{new}, choose p;
iii. if p_{new} and p' are incomparable, choose one randomly.

It is obvious that replacing the current rule p with any neighborhood rule p_{new} that dominates it in the PLS archive ($\boldsymbol{P_k}$) imposes selection pressure on the $\boldsymbol{P_k}$ and push it towards the Pareto-front. The new population $\boldsymbol{P_{best}}$ combined with $\boldsymbol{P_{N/best}}$ creates a new population $\boldsymbol{P_{g+1}}$.

IV. Experiment Design

In this section, experimental studies will be conducted to compare the proposed algorithm GP-PLS-F with GP-NSGA-III, GP-A-NSGA-III, GP-PLS-s [14], and GP-PLS-r [14]. Here GP-PLS-s refers to the variation of GP-PLS where the scalarization approach is used for selection in [14]. On the other hand, GP-PLS-r represents the variation where the replacement strategy is used for selection in [14]. GP-A-NSGA-III [23] is an adaptive extension of GP-NSGA-III. The following subsections, we will describe the dataset used in the experiments, parameter settings of the algorithms, initial results, and discussions.

A. Dataset

In this paper, the widely used Taillard static (TA) job shop benchmark set [24] is selected as the dataset for the experiments. There are 80 indexed problem instances in the TA set which can be further divided into eight groups (denoted as the TA-1, TA-2, ..., TA-8 groups). The problem instances in the same group have the same number of jobs and machines, but the processing time matrices were generated by using different random seeds. Across different groups, the number of jobs varies from 15 to 100, and the number of machines varies from 15 to 20. In the experiments, the total 80 instances were divided into the *training set* and the *test set*, each consisting of 40 instances. Specifically, the training set consists of all the instances with the odd ID, and the test set contains all the instances with the even ID.

B. Parameter Settings

For all competing algorithms, the crossover, mutation and reproduction rates are set to 85%, 10%, and 5%, respectively, based on previous works [4]. The maximal tree depth is set to 8. As a common practice, the population is initialized by the ramp-half-and-half method. In each generation, the parents are selected by the tournament selection method with a tournament size of 7. For GP-NSGA-III and GP-A-NSGA-III, the population size is set to 1000 and the maximal number of generations is set to 100. For GP-PLS-F, the population size is set to 1000 but a maximal number of generations will be set after the sensitivity analysis (as discussed in Section IV-D). GP-PLS-F has two additional parameters, size of the archive (K) and the maximum number of local search steps ($step_{max}$). The parameter settings for GP-PLS-s and GP-PLS-r are taken from [14]. In our experiment, we aim to minimize four objectives, i.e., the mean flowtime (Obj1), maximal flowtime (Obj2), mean weighted tardiness (Obj3), and maximal weighted tardiness (Obj4). Existing work [4] showed that the four objectives are mutually conflicting.

C. Performance Measures

There are a variety of performance measures proposed for evaluating multi-objective optimization algorithms from different perspectives. In this paper, following a common practice, we choose the *Hyper-Volume* (HV) [25] and *Inverted Generational Distance* (IGD) [26] as the two main performance measures. Theoretically, a set of tradeoff dispatching rules should have a *larger* HV value and a *smaller* IGD value.

D. Sensitivity analysis

In a hybridized algorithm, it is important to understand how to divide the available computation time between the local search and the global search. In order to prevent the local search from spending almost all available computation time, we decide to use the partial strategy, which restricts the number of iterations in the local search. If we use a very small value of $step_{max}$ (e.g., $step_{max}$= 1), the local search procedure may be terminated sooner than desired. On the contrary, if we use a large value of $step_{max}$ (e.g., $step_{max}$= 10), the local search procedure tends to evaluate more solutions than necessary.

Given the above, we need to carefully adjust the computation time spent by the local search procedure in our hybrid algorithm. Therefore, in this experiment, we examined different combinations of the parameters where population size is equal to 1000. The combination values of (K, $steps_{max}$, generations) are (1000,3,25), (500,2,50), and (250,4,50). The sensitivity analysis applies to GP-PLS-F and select parameters are later used in the algorithm. The three-parameter settings have the same total number of fitness evaluations (100000). For a fair comparison, the number of fitness evaluations is

TABLE II: The mean and standard deviation over the average HV and IGD values on different combinations in the four-objective experiment. The significantly better results are shown in bold.

	HV ($\bar{x} \pm s$)	IGD ($\bar{x} \pm s$)
Comb1-(1000,3,25)	0.684±0.015	0.00127±0.00012
Comb2-(500,2,50)	0.676±0.015	0.00130±0.00024
Comb3-(250,4,50)	**0.7133±0.0102**	**0.00120±0.00006**

kept identical to GP-NSGA-III and GP-A-NSGA-III. In the sensitivity analysis, for each combination of the parameters, 30 independent runs were performed to produce 30 final sets of dispatching rules. The Wilcoxon rank-sum test [27], with the significance level of 0.05 is applied to the HV and IGD of the Pareto-front evolved by the three compared combinations for PLS.

For the case of (1000,3,25), GP-PLS-F uses the whole population during the local search with three $step_{max}$ for exploring the neighborhood solutions. For the case of (500,2,50), GP-PLS-F can explore the solution space very well through 50 generations of evolution but has a small number of local searches. In contrast with the first two parameter combinations, (250,4,50) has a proper balance between global search (50 generations) and local search (4 steps during the local search) capabilities.

From the results summarized in Table II, we found that the total number of generations and the maximum number of local search steps is highly influential on the performance of GP-PLS-F. They together provide varying trade-offs between global and local searches in GP-PLS-F. The result showed that GP-PLS-F could not search the solution space extensively with a small number of generations in (1000,3,25). On the other hand, if GP-PLS-F cannot perform a sufficient number of local search steps in (500,2,50), the power of local search cannot be effectively utilized. Further the result showed that a combination (250,4,50) performed significantly better in terms of HV and IGD as compared to the other two combinations for PLS.

V. Results and Discussions

For each algorithm in the experiment, 30 GP runs were conducted to obtain 30 sets of dispatching rules. Then, the rules were tested on the 40 test instances.

A. Performance of Obtained Dispatching Rules

Tables III shows the mean and standard deviation of the training performance in terms of HV and IGD of the rules obtained by GP-NSGA-III, GP-PLS-s, GP-PLS-r, GP-PLS-F, and GP-A-NSGA-III. The Wilcoxon rank-sum test [27], with the significance level of 0.05 is applied to the HV and IGD of the Pareto-front evolved by the five compared algorithms.

Table III reveals that GP-PLS-F performs significantly better than other competing algorithms in terms of both HV and IGD. This is because GP-PLS-F select rules from each subspaces based on their convergence and diversity. Then, GP-PLS-F will locally tune a rule that is closer to the Pareto-front.

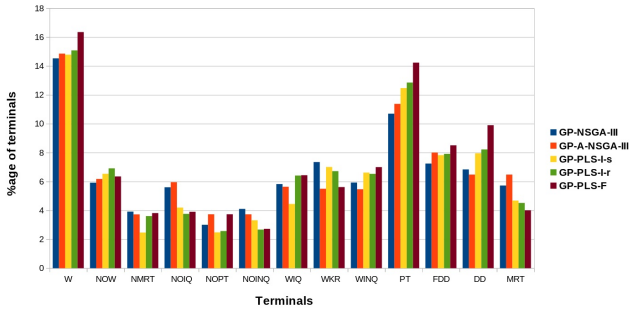


Fig. 4: Frequency of terminals in GP-NSGA-III, GP-PLS-s, GP-PLS-r,GP-A-NSGA-III GP-PLS-F.

Table III also reveals that GP-PLS-r is highly competitive with GP-PLS-F. Both algorithms (GP-PLS-r and GP-PLS-F) use the replacement strategy. Therefore, we can confirm that the replacement strategy is more effective than the scalarization strategy.

From Table III, we can see GP-PLS-F performed significantly better than GP-A-NSGA-III in terms of HV and IGD because GP-A-NSGA-III may not evolve a well-distributed set of Pareto-optimal solutions. This limitation is also introduced in [23].

GP-PLS-F also outperformed GP-NSGA-III in terms of HV and IGD because the local search enhances the exploitation ability of the GP-PLS. Further, the fitness-based selection of a solution from each search direction explores promising rules from evolved Pareto-front.

Table III also summarizes the testing performance of all algorithms in terms of IGD and HV. The obtained test results exhibit the same patterns as the training performance results. In the case of HV, GP-PLS-F performs significantly better than the other compared algorithms. However, in the case of IGD, GP-PLS-F is significantly better than GP-NGA-III and also better than GP-A-NSGA-III, GP-PLS-s, and GP-PLS-r.

B. Further Analysis

To further investigate how PLS with fitness assignment affects the GP search process, we analysis the occurrence of relevant terminals in each algorithm to optimize the flowtime and tardiness objectives in Fig. 4. Further, we plotted (a) the average HV and IGD of non-dominated solutions evolved by GP-PLS across multiple generations in Figs. 5(a) and 5(b) and (b) parallel coordinate plots of non-dominated solutions evolved by all algorithms on training instances in Figs. 6(a) to 6(e).

1) Analysis of dispatching rules: The bar chart in Fig. 4 shows the percentage of terminals in evolved rules from each algorithm. From the frequency of terminals, we can further analyze the relevant and irrelevant terminals.

According to the existing studies [5], [28], MRT, PT, WKR, WINQ, NOINQ, FDD, and NOPT are useful terminals for optimizing flowtime objectives. Specifically, PT, WINQ, and WKR are the most important three terminals for optimizing the flowtime objective. On the other hand, WINQ, NOINQ, NOPT,

TABLE III: The mean and standard deviation over the average HV and IGD values on training and test instances of the compared algorithms in the four-objective experiment. The significantly better results are shown in bold.

	Training		Test	
	HV ($\bar{x} \pm s$)	IGD ($\bar{x} \pm s$)	HV ($\bar{x} \pm s$)	IGD ($\bar{x} \pm s$)
GP-NSGA-III	0.68850±0.0221	0.00128±0.00015	0.52422±0.02423	0.00177±0.00020
GP-A-NSGA-III	0.70467±0.0115	0.00126±0.00016	0.55996±0.02336	0.00162±0.00025
GP-PLS-s	0.69048±0.0160	0.00127±0.00013	0.53625±0.02445	0.00164±0.00027
GP-PLS-r	0.70513±0.0130	0.00124±0.00012	0.56844±0.02147	0.00161±0.00016
GP-PLS-F	**0.7133±0.0102**	**0.00120±0.00006**	**0.59744±0.02317**	0.00160±0.00016

W, PT, MRT, and DD are the useful terminals for optimizing tardiness objectives. Of these, PT, DD, and W are the most useful terminals for optimizing tardiness objectives.

It can be seen in Fig. 4 that more than 10 per cent of evolved rules from each algorithm have W and PT terminals. Also, it can be seen from Fig. 4 that the number of occurrences of W, WINQ, PT, DD, MRT, NOPT and FDD terminals are higher in the local search algorithms (GP-PLS-r and GP-PLS-F) than in the GP-NSGA-III and GP-A-NSGA-III algorithms. This is because all the GP-PLS algorithms enhanced the exploitation ability and evolved significantly better rules as compared to the other algorithms in terms of HV and IGD. Therefore, these algorithms select the rules which are well-optimized. As a result, there are more chances of occurrences of useful terminals in GP-PLS-F algorithms. Further, it can be seen from Fig. 4 that GP-PLS-F has more useful terminals (W, PT, DD, FDD, and WINQ) than GP-PLS-s, GP-PLS-r. This analysis shows the effectiveness of a selection of solutions based on the fitness value.

2) Convergence Curves: Fig. 5(a) and Fig. 5(b) reveal that GP-PLS-F has better convergence behaviors in terms of both HV and IGD than other competing algorithms. These also show that the GP-PLS-F achieved better performance than GP-PLS-r and GP-PLS-s in terms of HV and IGD. These results reveal that the selection of the solutions based on FV (convergence and diversity) can improve the overall performance of the GP-PLS-F algorithm. Moreover, in the last few generations, when the solutions were very close to the Pareto-front, GP-PLS-F achieved significantly better HV and IGD.

3) Parallel Coordinate Plots: The parallel coordinate plots in Fig. 6(a) to Fig. 6(e) depict the non-dominated set of dispatching rules obtained by all the competing algorithms. As it can be observed from Fig. 6, the rules evolved by GP-PLS-F and GP-PLS-r show a more diversified solutions and cover a much wider range of the objective space than other compared algorithms. Meanwhile, GP-A-NSGA-III manages to cover a much wider range of values for objective 2 and objective 4 as compared to GP-NSGA-III and GP-PLS-s. Fig. 6 also reveals that GP-PLS-F obtained better coverage for the third and fourth objectives (i.e. mean weighted tardiness and maximum weighted tardiness) than GP-PLS-r.

VI. Conclusions and Future Work

In this current study we successfully investigate the effectiveness of fitness-base selection mechanism in GP-PLS.

In this paper, we combine GP with a PLS for solving many-objective JSS problems. The key idea of this approach is to perform multiple local search steps and effectively find the neighborhood of non-dominated dispatching rules. This local search mechanism helps create excellent exploitation abilities in GP-PLS. Similarly, GP-PLS-F features the use of a newly designed fitness-based selection approach. A fitness-based selection criterion was proposed for selecting initial solutions for neighborhood exploration — the selection criteria based on convergence and diversity. Extensive experiments have been performed to understand the effectiveness of the proposed GP-PLS-F by using the Taillard static job shop benchmark set. Experiment results showed that GP-PLS performed significantly better than other compared algorithms in terms of both HV and IGD. The proposed algorithm has been further analyzed to reveal the different preferences over the use of terminals. It is a first step investigation of PLS with the fitness-based selection mechanism in GP.

In future studies, we will enhance the performance of our proposed PLS by developing an intelligent local search operator to guide exploitation based on recently evaluated rules and adaptive selection methods. To unleash the great potential of the local search techniques on many-objective GP-HH, more investigations are required.

References

[1] M. L. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2012.

[2] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[3] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: handling constraints and extending to an adaptive approach." *IEEE Transactions. Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, 2014.

[4] A. Masood, Y. Mei, G. Chen, and M. Zhang, "Many-objective genetic programming for job-shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 209–216.

[5] O. Holthaus and C. Rajendran, "Efficient jobshop dispatching rules: Further developments," *Production Planning & Control*, vol. 11, no. 2, pp. 171–178, 2000.

[6] B. Peng, Z. Lü, and T. Cheng, "A tabu search/path relinking algorithm to solve the job shop scheduling problem," *Computers & Operations Research*, vol. 53, pp. 154–164, 2015.

[7] V. Sels, N. Gheysen, and M. Vanhoucke, "A comparison of priority rules for the job shop scheduling problem under different flow time-and tardiness-related objective functions," *International Journal of Production Research*, vol. 50, no. 15, pp. 4255–4270, 2012.

[8] J. Branke, S. Nguyen, C. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A Review." *IEEE Transactions Evolutionary Computation*, vol. 20, no. 1, pp. 110–124, 2016.

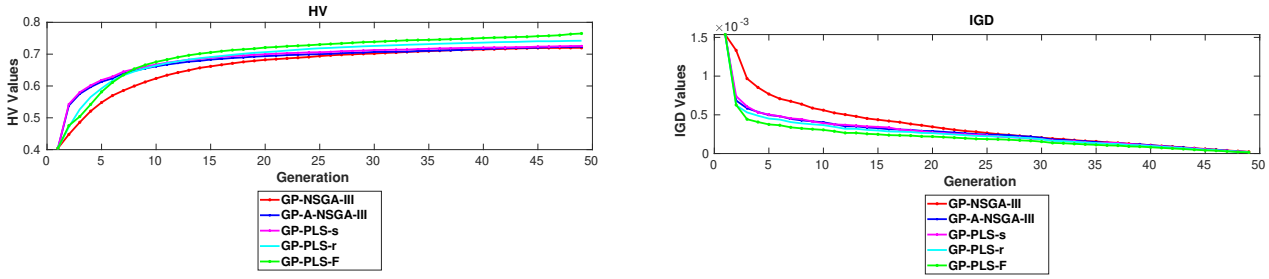
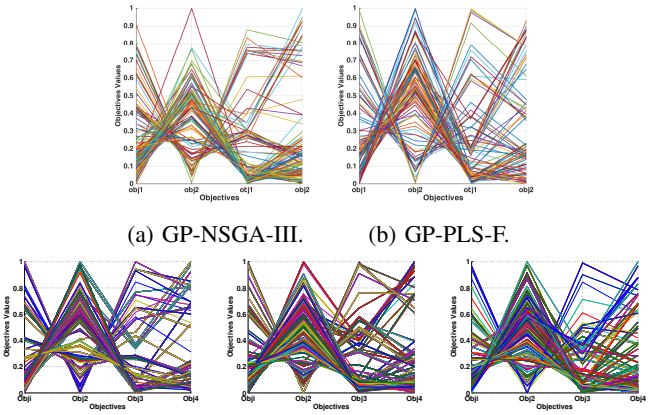


Fig. 5: The curves of the average number of (a) HV value, and (b) IGD value of the non-dominated solutions on the training set during the 30 independent GP runs.



(a) GP-NSGA-III. (b) GP-PLS-F.

(c) GP-A-NSGA-III. (d) GP-PLS-s. (e) GP-PLS-r.

Fig. 6: Parallel coordinates of non-dominated front obtained by each algorithm.

[9] S. Nguyen, M. Zhang, and M. Johnston, "A genetic programming based hyper-heuristic approach for combinatorial optimisation," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 1299–1306.

[10] F. Kolahan and V. Kayvanfar, "A heuristic algorithm approach for scheduling of multi-criteria unrelated parallel machines," *World, Academy of Science, Engineering and Technology*, vol. 59, p. 102, 2009.

[11] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic programming via iterated local search for dynamic job shop scheduling," *IEEE Transactions on Cybernetics*, vol. 45, no. 1, pp. 1–14, 2015.

[12] B. Chen, W. Zeng, Y. Lin, and D. Zhang, "A new local search-based multiobjective optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 50–73, 2015.

[13] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,*, vol. 28, no. 3, pp. 392–403, 1998.

[14] A. Masood, G. Chen, Y. Mei, H. Al-Sahaf, and M. Zhang, "Genetic programming with Pareto local search for many-objective job shop scheduling." in *Australasian Conference on Artificial Intelligence*, ser. Lecture Notes in Computer Science, J. Liu and J. Bailey, Eds., vol. 11919. Springer, 2019, pp. 536–548.

[15] S. Nguyen, "Automatic design of dispatching rules for job shop scheduling with genetic programming," Ph.D. dissertation, 2013.

[16] J. Park, S. Nguyen, M. Zhang, and M. Johnston, "Evolving ensembles of dispatching rules using genetic programming for job shop scheduling," in *Genetic Programming*. Springer, 2015, pp. 92–104.

[17] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Dynamic multi-objective job shop scheduling: A genetic programming approach," in *Automated Scheduling and Planning*. Springer, 2013, pp. 251–282.

[18] A. Blot, L. Jourdan, and M.-É. Kessaci, "Automatic design of multi-objective local search algorithms: case study on a bi-objective permutation flowshop scheduling problem," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 227–234.

[19] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle, "Anytime pareto local search," *European Journal of Operational Research*, vol. 243, no. 2, pp. 369–385, 2015.

[20] S. Jiang and S. Yang, "A strength Pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 329–346, 2017.

[21] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 838–856, 2015.

[22] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

[23] H. Jain and K. Deb, "An improved adaptive approach for elitist non-dominated sorting genetic algorithm for many-objective optimization," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2013, pp. 307–321.

[24] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 7, pp. 1359–1371, 2014.

[25] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[26] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition," *University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report*, pp. 1–30, 2008.

[27] F. Wilcoxon, *Individual comparisons by ranking methods*. Springer, 1992.

[28] Y. H. Lee, K. Bhaskaran, and M. Pinedo, "A heuristic to minimize the total weighted tardiness with sequence-dependent setups," *IIE Transactions*, vol. 29, no. 1, pp. 45–52, 1997.