A PSO-based Reference Point Adaption Method for Genetic Programming Hyper-heuristic in Many-Objective Job Shop Scheduling

Atiya Masood, Yi Mei, Gang Chen, and Mengjie Zhang

Victoria University of Wellington, Wellington, New Zealand
{masood.atiy, yi.mei, aaron.chen, mengjie.zhang}@ecs.vuw.ac.nz

Abstract. Job Shop Scheduling is an important combinatorial optimisation problem in practice. It usually contains many (four or more) potentially conflicting objectives such as makespan and mean weighted tardiness. On the other hand, evolving dispatching rules using genetic programming has demonstrated to be a promising approach to solving job shop scheduling due to its flexibility and scalability. In this paper, we aim to solve many-objective job shop scheduling with genetic programming and NSGA-III. However, NSGA-III is originally designed to work with uniformly distributed reference points which do not match well with the discrete and non-uniform Pareto front in job shop scheduling problems, resulting in many useless points during evolution. These useless points can significantly affect the performance of NSGA-III and genetic programming. To address this issue and inspired by particle swarm optimisation, a new reference point adaptation mechanism has been proposed in this paper. Experiment results on many-objective benchmark job shop scheduling instances clearly show that prominent improvement in performance can be achieved upon using our reference point adaptation mechanism in NSGA-III and genetic programming.

Keywords: Job shop scheduling. Many-objective optimisation. Genetic programming. Reference points.

1 Introduction

Job Shop Scheduling (JSS) [15] is one of the most important combinatorial optimisation problems in practice and has a wide range of applications in many industries such as manufacturing and cloud computing. In a JSS problem, a group of jobs with predetermined routes are assigned to a fixed set of machines. The problem requires us to design a schedule that dictates job processing in an optimal way so that some pre-defined objectives (e.g. makespan, tardiness and total revenue) can be achieved successfully without violating any domain-specific constraints.

JSS problems are known to be *NP-hard* [2]. As a matter of fact, no exact methods exist in practice to obtain optimal schedules within a reasonable amount of time when the number of jobs and machines becomes large. Hence

heuristic and meta-heuristic methods play a more dominating role for solving many practical JSS problems. Among these methods, *dispatching rules* are frequently exploited due to their flexibility, scalability and efficiency, especially when a job shop exhibits high levels of dynamics.

Conceptually, dispatching rules can be treated as priority functions that assign priorities to every job waiting to be processed by a machine. Whenever the machine becomes idle, the job with the highest priority value according to a dispatching rule will be selected for processing. Such a dispatching rule will therefore incrementally determine the complete schedule for any JSS problems. In this paper we will focus mainly on *non-delay* dispatching rules since they avoid any unnecessary delay on idle machines whenever there are incomplete jobs pending on them.

Despite of prominent success, designing useful dispatching rules remains to be a challenging research problem. Particularly dispatching rules designed manually based on one JSS problem instance or one scheduling objective (e.g. minimizing the mean flow time) often perform poorly when the scheduling conditions change (e.g. minimizing the maximal tardiness). To tackle this challenge, many researchers have successfully developed *Genetic Programming Hyper-Heuristic* (GP-HH) techniques for automatic design of dispatching rules under various scheduling objectives. A comprehensive survey on related GP-HH research works can be found in [3,4].

It is widely evidenced in the literature that JSS by nature presents several potentially conflicting objectives, including for example the makespan, mean flow-time, maximum tardiness, maximum lateness, total workload and proportion of tardy jobs. A very recent work also suggests that it is important to consider *many objectives* (i.e. more than three objectives) concurrently while solving a wide range of JSS problems [13]. Driven by this understanding, a new algorithm called GP-NSGA-III that seamlessly combines GP-HH with NSGA-III [6] was proposed, which is one of the state-of-the-art evolutionary many-objective optimisation algorithm [13]. In comparison to other approaches that combine GP-HH with multi-objective optimisation algorithms including NSGA-II [**?**] and SPEA2 [**?**], GP-NSGA-III can achieve significantly better performance on 4-objective and 5-objective JSS problems [13].

Despite the promising results with GP-NSGA-III, it was found that many references points are *useless*, i.e. they are never associated with any dispatching rules on the evolved Pareto front. Similar observations have also been witnessed in [8] while applying NSGA-III to other optimisation problems. There is a close relationship between the number of useless points and the performance of the algorithm. Therefore, to evolve high-quality dispatching rules, it is essential to match the distribution of the reference points with the distribution of the Pareto front, which is often irregular (e.g. discrete and non-uniform).

To address this key issue in GP-NSGA-III, the main goal of this study is to develop an effective reference point adaptation mechanism to enhance the match between reference points and the Pareto-front evolved by GP-HH. Guided by this goal, we have developed a new adaptation mechanism inspired by *Particle*

Swarm Optimisation (PSO) which has been proven to be highly effective for approximating arbitrary distributions such as the fitness landscape [**?**,**?**]. In this paper, to prevent all reference points from converging to a small area in the objective space, we have modified the standard particle dynamics in PSO so that every reference point (i.e. a separate particle in PSO) can be optionally attracted towards one of multiple *global best* locations. We have also removed the influence of *personal best* locations since they may not promise good matches with the evolved Pareto-front in future generations.

Driven by the aim of reducing the number of useless reference points and enhancing the matches between the reference points and the evolved Pareto-front, we organize the rest of paper as follows. Section 2 covers the research background, including the JSS problem description, the reference point adaptation problem and the related works. Section 3 introduces GP-A-NSGA-III in detail. Section 4 reports the experimental design. Section 5 covers results and discussions. Finally, Section 6 concludes this paper and highlights possible future research.

2 Research Background

In this section, the JSS problem will be described first. Then we will discuss some related works.

2.1 Description of JSS Problems

In a JSS problem, N jobs are initially assigned to M machines. Each job j_i, $1 \leq i \leq N$ has a sequence of m operations to be performed, i.e. $\{o_i^1, \ldots, o_i^m\}$. Each operation o_i^k should be processed on one machine $m_i^k, 1 \leq i \leq M$ with the processing time p_i^k. Let r_i^k be the time for operation o_i^k to be ready for processing. R_i is the *ready time* of the first operation of job j_i, which is also known as the *release time*. Any solution to such a JSS problems has to comply with three common constraints, as described below.

1. An operation is performed on a machine without interruption. This means that all operations are non-preemptive.
2. The operation cannot start until its previous operation has been completed. In the other words, the operations follow precedence constraints.
3. Each machine is persistently available and should process only one operation of any job at any given time.

Similar to [13], we consider JSS problems with four objectives, i.e. the mean flowtime, maximal flowtime, mean weighted tardiness and maximal weighted tardiness. Clearly, each objective is expected to be minimised.

2.2 Problem for using Uniformly Distributed Reference Points

As mentioned in the introduction, NSGA-III is originally designed to work with a fixed set of uniformly distributed reference points. However, for many

combinatorial optimisation problems such as the JSS problems, the true Pareto-front is usually irregular and discontinuous. Therefore direct application of NSGA-III will lead to many *useless points*.

In essence, reference points are adopted in NSGA-III to replace the crowding distance mechanism introduced in NSGA-II for high-dimensional objective space and hence to promote *solution diversity* [6]. Clearly, if only a few reference points are truly associated with the Pareto-optimal dispatching rules evolved by GP-HH at the current generation, it is not easy to distinguish and select these rules to improve diversity for future generations. In other words, a majority of useful reference points will be associated with many candidate dispatching rules. Some rules can be far from the corresponding reference point. Such a dispatching rule should enjoy higher selection opportunity but may not be selected during evolution simply because it is associated with a popular reference point. Due to the above reason, we found that better matches between reference points and the evolved Pareto-front can help enhance solution diversity and therefore the performance of GP-NSGA-III. As a consequence, developing a reference point adaptation mechanism becomes a key research issue to be tackled in this paper.

2.3 Related Work

A study of the literature shows that many past research works focused primarily on solving single-objective JSS problems [**?**]. Recently, huge efforts have been made in the *evolutionary computation* (EC) community to develop multi-objective algorithms for JSS. Among these algorithms, Pareto-optimal methods have clearly attracted substantial research attention [14]. Inspired by their success, a few efforts have also been made towards addressing many-objective JSS problems [13]. Particularly, the GP-NSGA-III algorithm proposed in [13] is amongst the first in the literature to try to cope with exponentially increasing objective space by using NSGA-III. However, the use of uniformly distributed reference points in NSGA-III presents a new research challenge.

Deb. *et al.* have already proposed an interesting mechanism for adjusting reference points in NSGA-III [8]. However, their mechanism requires the number of reference points to be changed dynamically and is difficult to implement in high-dimensional objective space. When the number of reference points becomes large, the efficiency of their mechanism may also be affected. Nevertheless, their mechanism is very effective for addressing constrained general-purpose optimisation problems. Without considering constraints, we prefer to adopt a much simpler approach for reference point adaptation. Specifically our approach does not change the total number of reference points during evolution and is easy to implement regardless of the number of objectives under consideration. It is particularly suitable for the JSS problems described in Subsection 2.1.

3 Adaptive Reference Points for Many-Objective JSS

The framework of the proposed GP-A-NSGA-III is described in Algorithm 1. The framework is similar to that of the GP-NSGA-III [13], and the differences

(for adaptive reference points) are highlighted. Particularly, when initialising the reference points (line 2), a velocity vector $\boldsymbol{V}$ is initialised along with the position vector $\boldsymbol{X}$ for each reference point. Then, the PSO parameters are specified (line 3). In each generation, the reference points are updated (line 10) after each population update.

Algorithm 1: The framework of GP-A-NSGA-III.

Input : A training set $\boldsymbol{I_{train}}$ and rules P
Output: A set of non-dominated rules $\boldsymbol{P}^*$
1 Initialise and evaluate the population $\boldsymbol{P}_0$ of rules by the ramped-half-and-half method;
2 Initialise the position $\boldsymbol{X}$ and velocity $\boldsymbol{V}$ of reference points $\boldsymbol{Z}$;
3 Set parameter w_{min}, w_{max}, c_2;
4 Calculate the reference points $\boldsymbol{Z}$ Set $g \leftarrow 0$;
5 **while** $g < g_{\max}$ **do**
6 Generate the offspring population $\boldsymbol{Q}_g$ using the crossover, mutation and reproduction of GP;
7 **foreach** $Q \in Q_g$ **do** Evaluate rule Q;
8 $\boldsymbol{R}_g \leftarrow \boldsymbol{P}_g \cup \boldsymbol{Q}_g$;
9 Form the new population $\boldsymbol{P}_{g+1}$ from $\boldsymbol{R}_g$ by the NSGA-III selection;
10 Update($\boldsymbol{Z}$);
11 $g \leftarrow g + 1$;
12 **end**
13 **return** *The non-dominated individuals* $P^* \subseteq P_{g_{\max}}$;

3.1 Reference Point Update

The PSO-based reference point update scheme Update($\boldsymbol{Z}$) is described in Algorithm 2. In the algorithm, each reference point is seen as a particle. The fitness of a particle is defined as the number of individuals in the population associated to it, which is to be maximised. That is, the global best particle is the one with the most individuals associated to it. Note that when updating the velocity (line 6), the term for the local best position is ignored. It can be seen that the fitness of particles (reference points) depends on the distribution of the whole swarm, and the positions of other particles. Thus, it may not be meaningful to move towards the local best, which can become worse upon movements of other particles. Furthermore, there is no elitism for the global best (line 3). This way, the reference points can have sufficient diversity.

3.2 Fitness Evaluation

For evaluating each GP individual (lines 1 and 7 of Algorithm 1), it is applied to a set of JSS training instances $\boldsymbol{I}_{train}$ as a dispatching rule, and the normalised

Algorithm 2: Update of the reference points.

Input : Reference points $\boldsymbol{Z} = (\boldsymbol{X}, \boldsymbol{V})$
Output: Updated reference points $\boldsymbol{Z}^g$
Calculate fitness $fit(\boldsymbol{Z}_i)$= number of individuals associated for each $\boldsymbol{Z}_i \in \boldsymbol{Z}$;
Calculate $w = w_{max} - g \cdot (w_{max} - w_{min})/g_{max}$;
Calculate the global best $\boldsymbol{Z}^* = \arg\max\{fit(\boldsymbol{Z})\}$;
for $i = 1 \rightarrow NumParticles$ **do**
    for $j = 1 \rightarrow NumObjs$ **do**
        $\boldsymbol{V}^{\boldsymbol{g}}_{\boldsymbol{i,j}} = w * \boldsymbol{V}^{g-1}_{i,j} + c_2 * rand() * (\boldsymbol{Z}^* - \boldsymbol{X}^{g-1}_{i,j})$;
        $\boldsymbol{X}^{g}_{i,j} = \boldsymbol{X}^{g-1}_{i,j} + \boldsymbol{V}^{g}_{i,j}$;
    end
end
return $\boldsymbol{Z}^g = (\boldsymbol{X}^g, \boldsymbol{V}^g)$;

Algorithm 3: The fitness evaluation.

Input : A training set $\boldsymbol{I}_{train}$ and an individual (rule) P
Output: The fitness $\boldsymbol{f}(P)$ of the rule P
foreach I *in* $\boldsymbol{I}_{train}$ **do**
    Construct a schedule $\Delta(P, I)$ by applying the rule P to the JSS instance I;
    Calculate the objective values $\boldsymbol{f}(\Delta(P, I))$;
end
$\boldsymbol{f}(P) \leftarrow \frac{1}{|\boldsymbol{I}_{train}|} \sum_{I \in \boldsymbol{I}_{train}} \boldsymbol{f}(\Delta(P, I))$; ;
return $\boldsymbol{f}(P)$;

objective values of the resultant schedules are set to its fitness values. The pseudo code of the fitness evaluation is given in Algorithm 3.

4 Experimental Design

To verify the effectiveness of the proposed adaptive reference point scheme, we compared the performance of GP-A-NSGA-III with the baseline GP-NSGA-III in the experimental studies. We selected the Taillard (TA) static JSS benchmark instances [16] as the testbed. The TA set has been widely used as the test JSS instances in literature. It consists of 80 instances (ID from 1 to 80) divided into 8 groups, each having 10 instances. The number of jobs varies from 15 to 100 and the number of machines ranges from 15 to 20. The instances belonging to the same group have the same numbers of jobs and machines. In the experiments, the 80 TA instances were split in half, each with 40 instances. Then, one subset was used as the training set and the other was the test set. Since the instances are static, all the jobs are available from time zero. The due date $dd(j_i)$ of each job j_i is calculated as $dd(j_i) = \lambda \times \sum_{k=1}^{m} p_i^k$,

where the due date factor λ is set to 1.3. In the experiments, we considered four potentially conflicting objectives: *mean flowtime (MF), maximal weighted*

tardiness (MaxWT), maximal flowtime (MaxF) and mean weighted tardiness (MWT). For both GP-A-NSGA-III and GP-NSGA-III, 40 independent runs were conducted.

4.1 Parameter Settings

Both GP-A-NSGA-III and GP-NSGA-III adopt the GP representation (tree-based) and evolutionary operators (e.g. initialisation, crossover and mutation) along with the fitness evaluation scheme of NSGA-III. For both compared algorithms, the population size is set to 1024. The crossover, mutation and reproduction rates are set to 85%, 10% and 5% respectively. The maximal depth is set to 8. The population is initialised by the ramp-half-and-half method. In each generation, the parents are selected by the tournament selection with size of 7. The maximal number of generations is set to 51. The terminal set is described in Table 1. The function set includes the basic arithmetic operators (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.

For the PSO parameters, we set $c_2 = 2$, $w_{min} = 0.4$ and $w_{max} = 0.9$, which are standard settings used by many existing works.

Table 1: Terminal set of GP for JSS.

Attribute	Notation
Processing time of the operation	PT
Inverse processing time of the operation	IPT
Processing time of the next operation	NOPT
Ready time of the operation	ORT
Ready time of the next machine	NMRT
Work Remaining	WKR
Number of operation remaining	NOR
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

In the experiments, the two commonly used measures in multi-objective optimisation, i.e. *Inverted Generational Distance* (IGD) [17] and *Hyper-Volume* (HV) [18] are used to compare the algorithms. To calculate IGD and HV, we first normalised the objectives into the range [0,1] by linear normalisation. The maximal and minimal values for each objective were obtained from the results of all the runs of both compared algorithms. After the normalisation, the nadir point was set to $(1, 1)$ for calculating HV. Note that IGD needs a set of uniformly distributed points in the true Pareto front, which cannot be known in

multi-objective JSS problems. Therefore, we approximate the true Pareto front by selecting the non-dominated solutions among the final solutions from all the runs of the two compared algorithms.

5 Results and Discussions

During the GP search process, a rule is evaluated on the 40 training instances, and the fitness function for each objective is defined as the average normalised objective value of the schedule obtained by applying that rule to each of the 40 training instances. For each algorithm, 40 GP runs obtained 40 final dispatching rules. Then, the rules were tested on the 40 test instances.

5.1 Overall Results

Table 2 shows the mean and standard deviation of the test performance (HV and IGD) of the rules obtained by GP-NSGA-III and GP-A-NSGA-III. In addition, for each test instance, the *Wilcoxon rank sum* test with the significance level of 0.05 was conducted separately on both the HV and IGD of the rules obtained by the two compared algorithms.That is, if p-value is smaller than 0.05, then the best algorithm is considered significantly better than the other algorithm. The significantly better results was marked in bold. The table reveals that GP-A-NSGA-III performs significantly better than GP-NSGA-III in most of the test instances. In the case of HV, GP-A-NSGA-III performed significantly better on 24 out of 40 test instances. On the other hand, GP-NSGA-III performed significantly better only on 6 instances. For the remaining 10 instances, the two compared algorithms performed statistically the same. In regard to IGD, Table 2 exhibits the same pattern. GP-A-NSGA-III achieved significantly better performance on 21 out of the 40 test instances. In contrast, GP-NSGA-III performed significantly better on 13 instances.

When taking a closer look at the Tables 2, it can be found that GP-A-NSGA-III not only performed better on smaller instances but also is more effective on more challenging and larger instances. For some test instances (e.g. instances 1, 10, 24), GP-A-NSGA-III achieved a huge improvement. This demonstrates the usefulness of the proposed adaptive reference point scheme, which can find better association with population members and thus obtained well distributed reference points.

5.2 Further Analysis

To further investigate how the adaptive reference point scheme affect the GP search process, we plot (a) the average number of useless references points (those associated with no individual in the population); (b) the average HV of the non-dominated solutions obtained so far and (c) the average IGD of the non-dominated solutions obtained so far for each generation during the 40 independent runs of the two compared algorithms, as given in Fig. 1.

From Fig. 1, it is obvious that the adaptive reference point scheme can significantly reduce the number of useless points during the GP search process. Without adaptive points, the number of useless references points in GP-NSGA-III kept increasing from 910 to about 980. On the contrary, in GP-A-NSGA-III, the number of useless reference points first increased and then decreased to almost the same level as the beginning. This indicates that especially at the later stage of the search, the adaptive reference point scheme led to less useless reference points, and thus a better refinement of the regions around the population.

Fig. 1 show that GP-A-NSGA-III obtained better convergence curves in terms of both HV and IGD. This shows that the reduction of useless reference points can lead to better non-dominated sets.

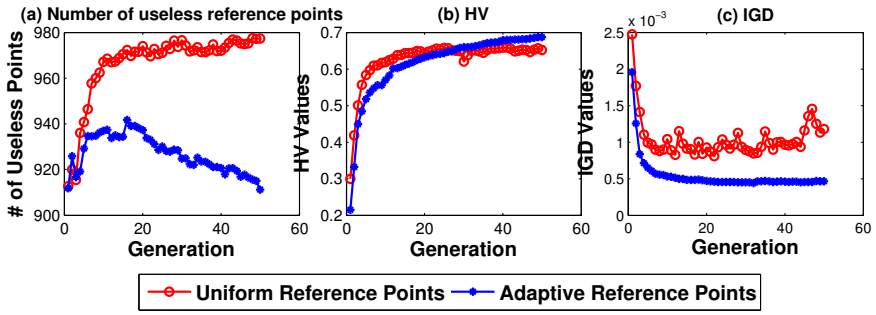
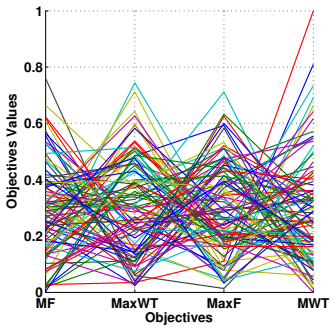


Fig. 1: The curves of the average number of useless reference points, HV and IGD values of the non-dominated solutions on the training set during the 40 independent GP runs.

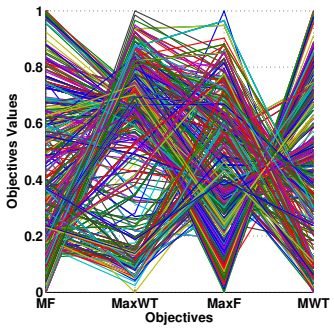
Fig. 2 shows the distribution of the reference points and the fitness values of the population in generations 1 and 50 of GP-A-NSGA-III. It can be seen that in generation 1, the reference points are close to the initial uniform distribution, and the fitness distribution of the population is relatively uniform as well. In generation 50, on the other hand, the distributions of the reference points and the fitness values of the population become very similar to each other. This is consistent with our expectation, which is to use a similar distribution of reference points as that of the population to fine tune the promising area around the population.

6 Conclusion

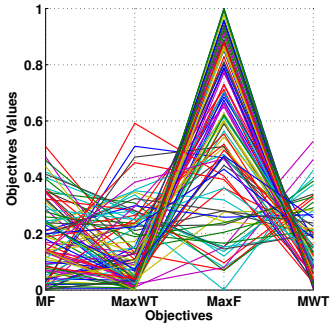
The goal of this study was to identify a key research issue involved in using NSGA-III effectively, i.e. the simple adoption of uniformly distributed reference points failed to promote solution diversity during evolution and affected the performance of GP-HH. This goal has been successfully achieved by proposing a new reference point adaptation mechanism inspired by PSO. Important changes to particle dynamics in PSO have also been introduced in our mechanism to prevent majority of reference points from converging to small areas in the objective



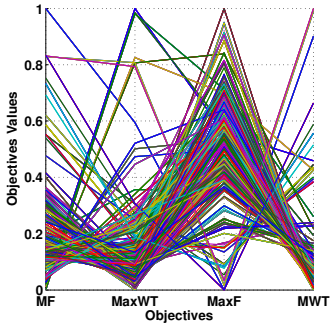
(a) Distribution of uniform reference points in generation 1



(b) Fitness values of the population in generation 1



(c) Distribution of adaptive reference points in generation 50



(d) Fitness values of the population in generation 50

Fig. 2: Parallel coordinate plot for the distribution of the reference points and the fitness values of the population in generations 1 and 50 of GP-A-NSGA-III.

space. In the subsequent experimental evaluations based on the Taillard benchmark set, we successfully showed that the proposed reference point adaptation mechanism can significantly improve the performance of GP-HH and NSGA-III in terms of both HV and IGD.

In the future, we plan to study the potential usefulness of Gaussian Processing modelling techniques for more accurate approximation of the Pareto front. The use of local search and niching techniques could also significantly improve solution diversity and boost performance of GP-NSGA-III. Their use with our reference point adaptation mechanism deserves further investigation.

References

1. Arroyo, J.E.C., Armentano, V.A.: Genetic local search for multi-objective flowshop scheduling problems. European Journal of Operational Research 167(3), 717–738

(2005)
2. Błażewicz, J., Domschke, W., Pesch, E.: The job shop scheduling problem: Conventional and new solution techniques. European journal of operational research 93(1), 1–33 (1996)
3. Branke, J., Nguyen, S., Pickardt, C., Zhang, M.: Automated Design of Production Scheduling Heuristics: A Review. IEEE Trans. Evolutionary Computation 20(1), 110–124 (2016)
4. Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: Exploring hyper-heuristic methodologies with genetic programming. In: Computational intelligence, pp. 177–201. Springer (2009)
5. Burke, E.K., Hyde, M.R., Kendall, G., Woodward, J.: Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. pp. 1559–1565. ACM (2007)
6. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. Evolutionary Computation, IEEE Transactions on 18(4), 577–601 (2014)
7. Herrero, J.G., Berlanga, A., Lopez, J.M.M.: Effective evolutionary algorithms for many-specifications attainment: Application to air traffic control tracking filters. Evolutionary Computation, IEEE Transactions on 13(1), 151–168 (2009)
8. Jain, H., Deb, K.: An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach. IEEE Trans. Evolutionary Computation 18(4), 602–622 (2014)
9. Kacem, I., Hammadi, S., Borne, P.: 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 (2002)
10. Koza, J.R.: Genetic Programming – On the Programming of Computers by Means of Natural Selection. MIT Press (1992)
11. Li, B., Li, J., Tang, K., Yao, X.: Many-objective evolutionary algorithms: A survey. ACM Computing Surveys (CSUR) 48(1), 13 (2015)
12. Luke, S., et al.: A Java-based Evolutionary Computation Research System . https://cs.gmu.edu/~eclab/projects/ecj/
13. Masood, A., Mei, Y., Chen, G., Zhang, M.: Many-Objective Genetic Programming for Job-Shop Scheduling. IEEE WCCI 2016 conference proceedings, IEEE (2016)
14. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Dynamic multi-objective job shop scheduling: A genetic programming approach. In: Automated Scheduling and Planning, pp. 251–282. Springer (2013)
15. Pinedo, M.L.: Scheduling: theory, algorithms, and systems. Springer Science & Business Media (2012)
16. Taillard, E.: Benchmarks for basic scheduling problems. european journal of operational research 64(2), 278–285 (1993)
17. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P.N., Liu, W., Tiwari, S.: 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)
18. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. Evolutionary Computation, IEEE Transactions on 7(2), 117–132 (2003)

Table 2: The mean and standard deviation over the HV and IGD values of the 40 independent runs of the compared algorithms in the 4-obj experiment. The significantly better results are shown in bold.

Problem Instances		HV		IGD	
ID	#J_#M	GP-NSGA-III	GP-A-NSGA-III	GP-NSGA-III	GP-A-NSGA-III
1	15_15	.0096(.0145)	**.2414(.0368)**	.0265(.0008)	**.0088(.0113)**
2	15_15	.1600(.0182)	**.1666(.0110)**	.0254(.0005)	**.0218(.0004)**
3	15_15	.0806(.0125)	**.0906(.0127)**	**.0205(.0008)**	.0292(.0006)
4	15_15	**.1263(.0183)**	.0692(.0091)	**.01765(.0007)**	.0218(.0006)
5	15_15	.1661(.0202)	.1752(.0192)	**.0190(.0005)**	.0237(.0007)
6	20_15	.1271(.0176)	**.1421(.0139)**	**.0154(.0004)**	.0211(.0004)
7	20_15	.2488(.0683)	**.2538(.0695)**	.0079(.0013)	.0077(.0014)
8	20_15	.1115(.0201)	**.2015(.0253)**	.0186(.0006)	.01819(.0026)
9	20_15	.1700(.0148)	**.1839(.0219)**	.0170(.0005)	**.0149(.0005)**
10	20_15	.1086(.0151)	**.3580(.0366)**	**.0156(.0002)**	.0160(.0003)
11	20_20	.0114(.0038)	**.1087(.0118)**	.0299(.0006)	**.0240(.0008)**
12	20_20	.0852(.0133)	**.1206(.0134)**	**.0173(.0004)**	.0222(.0006)
13	20_20	.1540(.0151)	.1569(.0326)	.0188(.0003)	**.0137(.0002)**
14	20_20	.0504(.01103)	**.0704(.0118)**	.0328(.0008)	**.0272(.0007)**
15	20_20	**.3985(.0225)**	.2454(.0199)	**.0109(.0004)**	.0150(.0002)
16	30_15	.2465(.030)	.2375(.0234)	**.0144(.0006)**	.0140(.0002)
17	30_15	.1813(.0096)	**.2278(.0229)**	.0138(.0003)	**.0097(.0004)**
18	30_15	**.3198(.0233)**	.1957(.0132)	**.0131(.0004)**	.0152(.0003)
19	30_15	.2789(.0126)	**.3004(.0197)**	.0150(.0004)	**.0114(.0004)**
20	30_15	**.2575(.0312)**	.2124(.0312)	**.0144(.0005)**	.0195(.0003)
21	30_20	.1347(.0657)	**.2325(.0792)**	.0135(.0022)	**.0098(.0014)**
22	30_20	.2365(.0472)	**.3027(.0470)**	.0065(.0010)	**.0052(.0006)**
23	30_20	.2944(.0398)	.2984(.0410)	.0046(.00004)	.0045(.0005)
24	30_20	.3812(.0503)	**.6161(.0174)**	.0070(.0009)	**.0018(.0004)**
25	30_20	.5199(.0477)	.5290(.0396)	.0059(.0012)	**.0046(.0005)**
26	50_15	.4563(.0417)	**.4872(.0270)**	.0051(.0009)	**.0039(.0004)**
27	50_15	.5710(.0361)	.5685(.0304)	.0040(.0009)	**.0032(.0003)**
28	50_15	.4598(.0398)	**.4966(.0250)**	.0049(.0010)	**.0036(.0003)**
29	50_15	.4862(.0372)	**.5125(.0251)**	.0045(.0007)	**.0037(.0003)**
30	50_15	.4510(.0406)	**.4732(.0240)**	.0033(.0005)	**.0026(.0002)**
31	50_20	.5085(.0424)	**.5147(.0295)**	.0053(.0008)	**.0042(.0004)**
32	50_20	.4378(.0476)	.4266(.0375)	.0046(.0006)	.0041(.0004)
33	50_20	.3422(.0266)	**.4383(.0838)**	.0125(.0006)	**.0069(.0051)**
34	50_20	.3828(.0384)	.4089(.0262)	.0036(.00005)	**.0030(.00029)**
35	50_20	.5558(.0349)	**.5763(.0165)**	.0025(.0005)	**.0020(.0001)**
36	100_20	**.3648(.0179)**	.2972(.0093)	**.010(.0003)**	.0130(.0006)
37	100_20	**.3442(.0142)**	.2844(.0101)	**.0086(.0003)**	.0107(.0003)
38	100_20	.3006(.0196)	.3025(.0136)	.0067(.0009)	.0066(.0002)
39	100_20	.6495(.0185)	.6515(.0191)	.0010(.0001)	.0010(.0001)
40	100_20	.3658(.0158)	**.3828(.0100)**	**.0085(.0003)**	.0088(.0002)