Constrained Dimensionally Aware Genetic Programming for Evolving Interpretable Dispatching Rules in Dynamic Job Shop Scheduling

Yi Mei[1], Su Nguyen[1,2], and Mengjie Zhang[1]

[1] Victoria University of Wellington, Wellington, New Zealand
[2] Advanced Analytics Lab, La Trobe University, AU
{yi.mei, su.nguyen, mengjie.zhang}@ecs.vuw.ac.nz

Abstract. This paper investigates the interpretability of the Genetic Programming (GP)-evolved dispatching rules for dynamic job shop scheduling problems. We incorporate the physical dimension of the features used in the terminal set of GP, and assume that the rules that aggregate the features with the same physical dimension are more interpretable. Based on this assumption, we define a new interpretability measure called dimension gap, and develop a Constrained Dimensionally Aware GP (C-DAGP) that optimises the effectiveness and interpretability simultaneously. In C-DAGP, the fitness is defined as a penalty function with a newly proposed penalty coefficient adaptation scheme. The experimental results show that the proposed C-DAGP can achieve better tradeoff between effectiveness and interpretability compared against the baseline GP and an existing DAGP.

1 Introduction

Job Shop Scheduling (JSS) [18] has applications in a variety of real-world domains such as manufacturing [3], project scheduling [23] and cloud computing. It aims to schedule the jobs arriving at a job shop (e.g. factory) subject to some constraints (e.g. each job much follow a pre-specified routing, and each machine can process no more than one job at one time) and optimise some criteria such as flowtime and tardiness.

In the real world, the job arrival process is an ongoing process. Therefore, it is more realistic to consider the *Dynamic* JSS (DJSS), in which there are unpredicted job arrivals occurring in real time. More specifically, in DJSS, at any given time point, only the information of the jobs that have arrived before the current time is available, while the future jobs are still unknown. In this paper, we focus on solving DJSS, which is closer to reality than the *static* JSS counterpart.

For solving DJSS, traditional optimisation approaches such as mathematical programming and genetic algorithms are not directly applicable since they are trying to obtain a solution (schedule). When the environment changes, e.g. a new job arrives, it is non-trivial to effectively adjust the current schedule to adapt to

the new environment. *Dispatching Rules* (DRs), on the other hand, are promising heuristics for solving DJSS due to their low complexity, scalability and flexibility. Instead of optimising the schedule as a whole, a DR gradually builds the schedule step by step by taking the latest information into account. Specifically, a DR uses a *priority function* to decide for each idle machine which job in its waiting queue should be processed next. Common DRs include First-Come-First-Serve (FCFS), Earliest Due Date (EDD), Shortest Processing Time (SPT), etc. A lot of DRs have been designed manually (e.g. [22, 10, 20]) by considering job shop attributes such as operation processing time, due date, work remaining and slack. However, the existing manually designed DRs are normally not effective enough, and restricted to the particular job shop scenario they are designed for.

The effectiveness of DRs depends on various factors such as objective, due date tightness and job shop utilisation [22]. Therefore, it is hard to design effective DRs *manually* under a given job shop scenario. To address this issue, Genetic Programming (GP) is a promising approach to automatically design DRs as a hyper-heuristic. Evolving DRs with GP has achieved some success in scheduling [2, 15], and the GP-evolved DRs have shown to be much more effective than the manually designed rules.

Most existing related works focused on the effectiveness of DRs. However, they ignored another important property of the GP-evolved DRs, which is *interpretability*. As a result, the GP-evolved DRs are too complicated to be interpreted and understood. The practitioners may feel less confident of using the DRs due to the lack of understanding of the inner mechanism, despite of their effectivenesses shown on the training instances.

In this paper, we aim to consider both *effectiveness* and *interpretability* of the DRs during the GP process. We employ the Dimensionally Aware GP (DAGP), which considers the *physical dimensions* (time, count and weight) of the job shop attributes, and favours the combinations between the attributes with the same physical dimension. Specifically, we have the following research objectives:

- Develop a Constrained DAGP (C-DAGP) algorithm for DJSS based on the physical dimensions of the job shop attributes.
- Propose a new penalty coefficient adaptation scheme for C-DAGP.
- Compare between the penalty adaptation schemes for C-DAGP, and compare C-DAGP with an existing DAGP and the baseline GP.

The rest of the paper is organised as follows: Section 2 gives the background introduction. Then, the proposed C-DAGP is described in Section 3. Experimental studies are carried out in Section 4. Finally, Section 5 gives the conclusions and future work.

2 Background

2.1 Job Shop Scheduling

JSS is to process a set of jobs with a set of given machines subject to some constraints. Each job has an arrival time, a due date, and a sequence of operations. Each operation has an eligible machine which is the only machine that

can process it, as well as a processing time. An operation cannot be processed before the completion before its precedent operations. Each machine can process at most one operation at a time. The commonly considered JSS objectives include minimising the makespan ($C_{\max}$), total flowtime ($\sum C_j$), total weighted tardiness ($\sum w_j T_j$), number of tardy jobs, etc [18].

2.2 Related Works

So far, there have been extensive studies [5, 9, 14, 17, 7]) in evolving DRs for DJSS using GP, and successfully achieved much better DRs than the previously man-made rules. Comprehensive reviews can be found in [2, 15].

Most existing works focused on the effectiveness (i.e. the test performance) of the GP-evolved DRs. Only a few recent works tried to improve the *interpretability* of the DRs. Nguyen et al. [14] investigated different representations and proposed a grammar-based representation to evolve more meaningful rules. Some studies tried to use *feature selection* to implicitly improve the interpretability of the evolved rules, assuming that using fewer terminals tends to generate more meaningful rules. Along this direction, Mei et al. [13] proposed to a feature selection algorithm that obtained more compact terminal set for GP. Riley et al. [21] proposed a similar feature selection approach.

To focus on more meaningful combinations of terminals, Hunt et al. [8] considered the physical dimensions of the job shop attributes (e.g. time, count and weight), and developed a strongly-typed GP that evolves DRs that only allows the "meaningful" combinations between the attributes with the same physical dimension. Following similar ideas, Durasević et al. [4] developed a DAGP that considers the compatibility between the physical dimensions of the terminals. They designed initialisation and evolutionary operators so that no semantically incorrect rule (e.g. adding time to weight) is generated.

However, it has been shown [8, 4] that when restricting the combination between terminals of GP, the rules obtained by the strongly-typed GP and DAGP had worse test performance than the baseline GP. The main reason is that the restrictions on the combinations of terminals make a huge part of the search space infeasible, and the resultant search space consists of many isolated feasible regions. It may be hard to jump from one feasible region to another. Thus, the final rule largely depends on the initial rules, and the search gets stuck into poor local optima easily.

Keizer and Babovic [11, 1] proposed a dimension-based brood selection scheme for DAGP, which addressed the overly restricted search space to some extent. The proposed algorithm allows dimensional inconsistent combinations, and uses a *culling function* to measure the total *dimensional inconsistency* (i.e. dimensional violations) of each individual. Then, each crossover/mutation operator generates $m(> 1)$ offsprings, and the best one in terms of dimensional inconsistency is selected as the offspring produced by the operator.

However, the culling function is not flexible enough, and our preliminary studies showed that in DJSS, even $m = 2$ can lead to a dramatic deterioration in test performance. In this paper, we propose to improve the flexibility of DAGP by considering the dimensional consistency as a constraint.

3 Constrained Dimensionally Aware Genetic Programming

The framework of the proposed C-DAGP is given in Algorithm 1. There are two important features in the framework, highlighted in lines 3 and 4. The first feature is the *dimension gap* (line 3), which reflects the degree of dimension inconsistency based on the *physical dimensions* (or *units*) of each terminal introduced in DAGP [11]. The calculation of the dimension gap will be described in detail in Section 3.1. The second feature is the *constrained fitness function* defined by both the objective value and the dimension gap. The two terms are aggregated by a *penalty coefficient* α on the dimension gap. Since the objective value and dimension gap have significantly different scales, an open issue is to set a proper α value to achieve good balance between the objective value and dimension gap. We propose a new penalty coefficient adaptation scheme, which will be described in Section 3.2.

Algorithm 1: The framework of C-DAGP.

```
1 Initialise a population using Grow method;
2 while Stopping criteria not met do
3 |   Calculate the objective value obj(x) and the dimension gap dimGap(x) for
  |     each individual x in the population ;
4 |   Calculate the fitness of each individual using a penalty function
  |     fit(x) = obj(x) + α · dimGap(x) ;
5 |   Generate a new population by selection and evolutionary operators;
6 end
7 return The best individual in the population;
```

3.1 Calculation of Dimension Gap

First, we introduce the physical dimensions of the terminals used in GP for evolving DRs in DJSS. We define three physical dimensions as follows:

1. `TIME`: including terminals such as processing time, due date, slack, etc.
2. `COUNT`: including terminals such as number of operations remaining, number of jobs in the queue, etc.
3. `WEIGHT`: the weight of a job.

Each node in the GP-tree is associated with a 3D vector $\boldsymbol{\theta} = (T, C, W)$, representing its exponentials of the three dimensions. For example, a terminal `PT` (processing time) is associated with a vector $(1, 0, 0)$, since it belongs to the `TIME` dimension, that is, its dimension exponential is 1 in `TIME`, and 0 in all the other dimensions. The dimension exponential values of a non-terminal node depends on that of its children and the function that the node represents. Table 1 shows how the calculation is conducted for the functions used in the proposed C-DAGP. For multiplication (division), the exponentials of the two children are added (subtracted). For addition, subtraction, max and min operators, since we allow children with inconsistent exponentials, we set the exponentials of the

result to be the average of that of the two children. If the two children have the same dimension exponentials, then the result will have the same dimension exponentials with the children as well.

Table 1. The calculation of the dimension vector values of a non-terminal node.

Function(s)	Children Vector Values	Result
$+$, $-$, max and min	(T_1, C_1, W_1), (T_2, C_2, W_2)	$\left(\frac{T_1+T_2}{2}, \frac{C_1+C_2}{2}, \frac{W_1+W_2}{2}\right)$
$\times$	(T_1, C_1, W_1), (T_2, C_2, W_2)	$(T_1+T_2, C_1+C_2, W_1+W_2)$
$/$	(T_1, C_1, W_1), (T_2, C_2, W_2)	$(T_1-T_2, C_1-C_2, W_1-W_2)$

Then, we calculate the dimension gaps for each node and the entire GP-tree. For multiplication and division, the dimension gap is always zero since these two operators have no restriction on the dimensions of the children. For the other operators, the dimension gap is the sum of the differences between the children in all the dimensions. The dimension gaps are calculated as follows:

$$\texttt{dimGap}(\texttt{node}) = \begin{cases} 0, & \text{if } \texttt{node} = \times \text{ or } / \\ \delta(\boldsymbol{\theta}(\texttt{c}_1), \boldsymbol{\theta}(\texttt{c}_2)), & \text{otherwise.} \end{cases} \tag{1}$$

$$\texttt{dimGap}(\texttt{tree}) = \sum_{\texttt{node} \in \texttt{tree}} \texttt{dimGap}(\texttt{node}), \tag{2}$$

where $\delta(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = |T_1 - T_2| + |C_1 - C_2| + |W_1 - W_2|$.

Fig. 1 gives an example of the dimension gap calculation. In this example, all the terminals have zero dimension gaps. Then, according to Eq. (1), the dimension gaps of the "$*$" and "$/$" non-terminal nodes are 0, and that of the root "$-$" is 2.

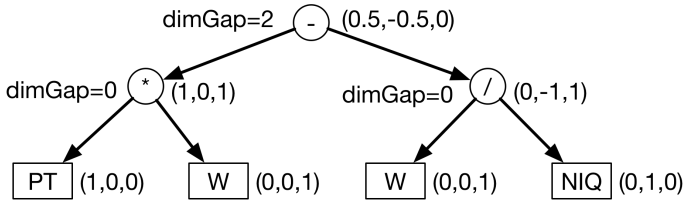

Fig. 1. An example of the dimension gap calculation.

3.2 Penalty Coefficient Adaptation

It is a non-trivial task to set a proper penalty coefficient to achieve a good balance between the test performance and interpretability (represented by dimension gap). Specifically, the penalty coefficient should be set according to the distribution of the current population. If the current population is located in

regions with high dimension gap, then the coefficient should be high to push the search towards the more interpretable areas. On the other hand, if most individuals in the current population have zero dimension gap, then the coefficient should be low to encourage the search to jump out of the current region of zero dimension gap ("feasible" region in terms of dimension consistency) through the intermediate area with positive dimension gap ("infeasible" regions). Here, we extend a mechanism proposed for bloating control in GP [19]. In that work, to keep the average program size in the population staying at the same level during the GP process, a *parsimony pressure* method is used and the adjusted fitness of a solution x is defined as $\texttt{fit}(x) = \texttt{obj}(x) + \alpha \cdot \texttt{size}(\texttt{x})$, where the parsimony coefficient α at generation t is determined by

$$\alpha(t) = -\frac{\text{cov}(\texttt{size}, \texttt{obj})}{\text{var}(\texttt{size})} \tag{3}$$

where the covariance $\text{cov}(\texttt{size}, \texttt{obj})$ and variance $\text{var}(\texttt{size})$ are calculated empirically using the program sizes and objective values of the individuals in the current population [19].

In this paper, we borrow this idea and design the penalty coefficient adaptation so that the average dimension gap of the individuals in the population stays at the same level during the GP process. To this end, we simply replace `size` with `dimGap` in Eq. (3). Furthermore, to reduce the empirical estimation bias of the covariance and variance, we propose the following two strategies:

1. Instead of using all the individuals in the population, we use the top 10% individuals in terms of the objective value in the population for estimating the covariance and variance measures. This way, we expect to reduce the effect of the individuals with very poor objective values.
2. The penalty coefficient is updated by a *moving average* technique as follows:

$$\alpha(0) = -\frac{\text{cov}(\texttt{dimGap}(pop_0), \texttt{obj}(pop_0))}{\text{var}(\texttt{dimGap}(pop_0))}, \tag{4}$$

$$\alpha(t+1) = \alpha(t) - \eta\left(\frac{\text{cov}(\texttt{dimGap}(pop_t), \texttt{obj}(pop_t))}{\text{var}(\texttt{dimGap}(pop_t))} + \alpha(t)\right). \tag{5}$$

where t is the generation index, and $0 \leq \eta \leq 1$ is a user-defined step size parameter. When $\eta = 0$, $\alpha(t+1) = \alpha(t)$ for all $t \geq 0$, i.e. the coefficient is fixed throughout the GP process. When $\eta = 1$, α is completely memoryless, and $\alpha(t+1)$ is independent of $\alpha(t)$.

4 Experimental Studies

To evaluate the effectiveness of the proposed C-DAGP, we first conduct sensitivity analysis on the parameter η. Then, we compare C-DAGP with the baseline GP (denoted as BaselineGP) and the GP with culling function [11] (denoted as CullingGP). For CullingGP, each crossover/mutation operator generates 2 offsprings, and the one with least dimension gap is selected.

In the experiments, we consider 3 objectives: maximal tardiness (Tmax), mean tardiness (Tmean) and total weighted tardiness (TWT). For each objective, we consider utilisation levels of 0.85 and 0.95. This results in $3 \times 2 = 6$ different job shop scenarios. The configuration parameters of the simulation model are given in Table 2, which has been used in previous studies [16, 6, 12]. The parameter setting of the compared GP algorithms is given in Table 3.

Table 2. The DJSS simulation configuration.

Parameter	**Value**	**Parameter**	**Value**
#machines	10	#jobs	5000
#warmup jobs	1000	#operations/job	Random from 2 to 10
Job arrival	Poisson process	Utilisation level	{0.85, 0.95}
Due date	4×total processing time	Processing time	$U[1, 99]$

Table 3. The parameter setting of the compared GP algorithms.

Parameter	**Value**	**Parameter**	**Value**
Terminal set	See Table 4	Function set	$\{+, -, *, /, \min, \max\}$
Population size	1024	Maximal depth	8
Crossover rate	80%	Mutation rate	15%
Reproduction rate	5%	#generations	51

Table 4. The terminals used in the GP algorithms.

Notation	Description	Dimension
WIQ	Work In Queue	`TIME`
MWT	Machine Waiting Time	`TIME`
PT	Processing Time	`TIME`
NPT	Next Processing Time	`TIME`
OWT	Operation Waiting Time	`TIME`
NWT	Next Machine Waiting Time	`TIME`
WKR	Work Remaining	`TIME`
WINQ	Work In Next Queue.	`TIME`
rFDD	Relative FDD	`TIME`
rDD	Relative DD	`TIME`
TIS	Time In System	`TIME`
SL	Slack	`TIME`
NIQ	Number of operations In Queue	`COUNT`
NOR	Number of Operations Remaining	`COUNT`
NINQ	Number of operations In Next Queue	`COUNT`
W	Weight	`WEIGHT`

During the training process, an individual is evaluated using a randomly generated simulation. To improve generalisation, the random seed for generating the training simulation changes per generation. In addition, the fitness is normalised by the objective value of the reference rule. The reference rule is set to EDD, ATC and WATC for Tmax, Tmean and TWT, respectively. Finally, the best individual in the last generation is selected as the best individual of the GP run.

For testing, a test set of 50 simulation replications is randomly generated for each scenario. The test fitness of a rule x is defined as the normalised total objective value over the test replications, i.e. $\Gamma(x, \Pi, F) = \frac{\sum_{\pi \in \Pi} F(x,\pi)}{\sum_{\pi \in \Pi} F(\text{RefRule(Obj)},\pi)}$, where $F \in \{\text{Tmax}, \text{Tmean}, \text{TWT}\}$.

All the compared GP approaches were implemented in Java using the ECJ library. The experiments were run on desktops with Intel(R) Core(TM) i7 CPU @3.60GHz. Both algorithms were run 30 times independently for each scenario.

4.1 Parameter Sensitivity Analysis

First, we conducted the sensitivity analysis to study the effect of the step size η on the performance of the algorithm. To this end, we compared the test performance and the dimension gap of the rules obtained by the C-DAGP with $\eta = 1$, 0.1 and 0.01, as shown in Tables 5 and 6. We conducted Wilcoxon's rank sum test with significance level of 0.05, and found no statistical significance between the compared η values in terms of both test performance and dimension gap. Table 6 shows that $\eta = 0.1$ tends to achieve smaller (although not significant due to the high standard deviation) dimension gap than $\eta = 1$ and $\eta = 0.01$. Therefore, we choose $\eta = 0.01$ in the subsequent experiments.

Table 5. The mean and standard deviation (in brackets) of the **test performance** obtained by the C-DAGP with $\eta = 1$, 0.1 and 0.01.

Scenario	Dimension Gap		
	$\eta = 1$	$\eta = 0.1$	$\eta = 0.01$
$\langle \texttt{T}_{\texttt{max}}, 0.85, 4 \rangle$	0.49(0.02)	0.49(0.01)	0.49(0.01)
$\langle \texttt{T}_{\texttt{mean}}, 0.85, 4 \rangle$	0.51(0.06)	0.51(0.06)	0.51(0.05)
$\langle \texttt{TWT}, 0.85, 4 \rangle$	0.59(0.10)	0.57(0.08)	0.59(0.09)
$\langle \texttt{T}_{\texttt{max}}, 0.95, 4 \rangle$	0.72(0.02)	0.71(0.02)	0.70(0.02)
$\langle \texttt{T}_{\texttt{mean}}, 0.95, 4 \rangle$	0.69(0.02)	0.68(0.02)	0.69(0.03)
$\langle \texttt{TWT}, 0.95, 4 \rangle$	0.80(0.04)	0.81(0.04)	0.80(0.04)

4.2 Results and Discussions

The proposed C-DAGP is compared with BaselineGP and CullingGP [11]. BaselineGP does not consider dimension gap at all. CullingGP is a DAGP that reduces the dimension gap by repeatedly generating several (2 in the experiment) offsprings in each crossover and mutation, and selecting the one with the minimal

Table 6. The mean and standard deviation (in brackets) of the **dimension gap** obtained by the C-DAGP with $\eta = 1$, 0.1 and 0.01.

Scenario	Dimension Gap		
	$\eta = 1$	$\eta = 0.1$	$\eta = 0.01$
$\langle \texttt{T}_{\texttt{max}}, 0.85, 4\rangle$	9.11(6.89)	5.35(6.07)	11.54(9.54)
$\langle \texttt{T}_{\texttt{mean}}, 0.85, 4\rangle$	15.42(8.11)	12.69(7.36)	15.61(7.59)
$\langle \texttt{TWT}, 0.85, 4\rangle$	13.12(6.31)	11.92(6.45)	14.31(7.88)
$\langle \texttt{T}_{\texttt{max}}, 0.95, 4\rangle$	10.31(9.97)	9.10(7.91)	13.69(8.79)
$\langle \texttt{T}_{\texttt{mean}}, 0.95, 4\rangle$	16.10(8.95)	15.29(7.83)	15.21(6.27)
$\langle \texttt{TWT}, 0.95, 4\rangle$	16.77(8.03)	15.27(6.09)	16.95(6.63)

dimension gap. It is not flexible in adjusting the balance between test performance and dimension gap.

Figs. 2 and 3 show the convergences curves of the compared BaselineGP, CullingGP [11] and C-DAGP in terms of test performance and dimension gap. For each curve, the center is the mean value, and the ribbon is the standard error In Fig. 2, the curves for the last 5 generations are zoomed in and shown in the blocks inside.

From Fig. 2, one can see that in terms of performance of C-DAGP (blue) was generally better than CullingGP (green). It outperformed CullingGP in $\langle \texttt{T}_{\texttt{mean}}, 0.85, 4\rangle$, $\langle \texttt{TWT}, 0.85, 4\rangle$, $\langle \texttt{T}_{\texttt{mean}}, 0.95, 4\rangle$ and $\langle \texttt{TWT}, 0.95, 4\rangle$, and almost the same as CullingGP in other cases. C-DAGP was slightly worse than BaselineGP (red). It was outperformed by BaselinGP in $\langle \texttt{T}_{\texttt{mean}}, 0.95, 4\rangle$ and $\langle \texttt{TWT}, 0.95, 4\rangle$, performed better in $\langle \texttt{T}_{\texttt{mean}}, 0.85, 4\rangle$, and achieved comparable performance as BaselineGP in other cases.

Fig. 3 clearly shows that the dimension gap obtained by C-DAGP is between the dimension gaps of BaselineGP and CullingGP. BaselineGP ignores the dimension gap during the evolutionary process, and thus obtained very high dimension gaps. CullingGP, on the other hand, focused too much on the dimension gap. As a result, it achieved very low dimension gap (close to zero) at the cost of significantly worse test performance than BaselineGP (the green curves v.s. red curves in Fig. 2).

Overall, C-DAGP sat in the middle of BaselineGP and CullingGP. It achieved better test performance than CullingGP and smaller dimension gap than BaselineGP. Although the current results do not clearly show the advantage of C-DAGP, the new constrained optimisation framework enables a finer control on the balance between the test performance and dimension gap than CullingGP (it is reduced to BaselineGP when $m = 1$).

4.3 Further Analysis

As a further analysis, we investigated the structure of a rule obtained by C-DAGP for $\langle \texttt{TWT}, 0.85, 4\rangle$, which obtained both promising test performance (0.47 versus the mean of 0.57 over 30 runs) and dimension gap ($\texttt{dimGap} = 2$). The

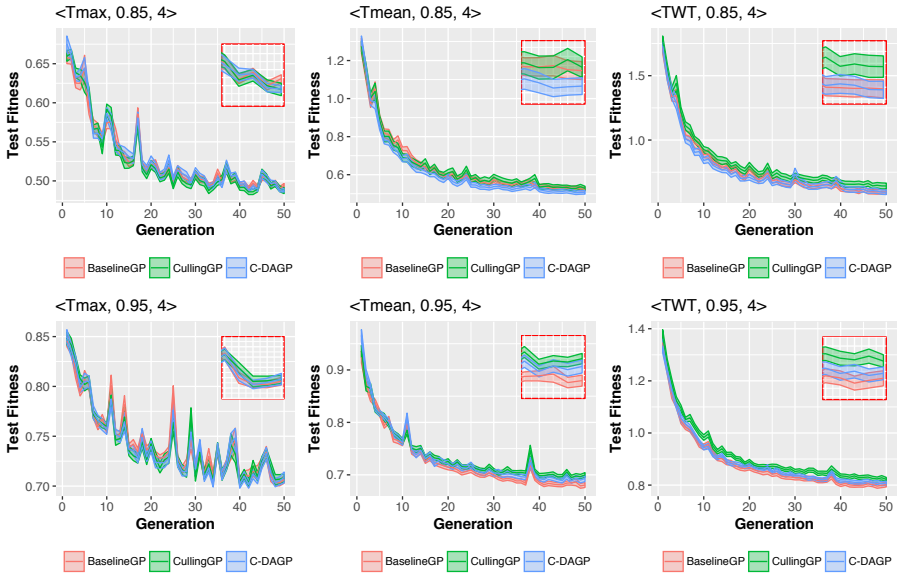

Fig. 2. The convergence curves (mean and standard error) of the **test performance** of BaselineGP, CullingGP and C-DAGP, with the last 5 generations zoomed in.

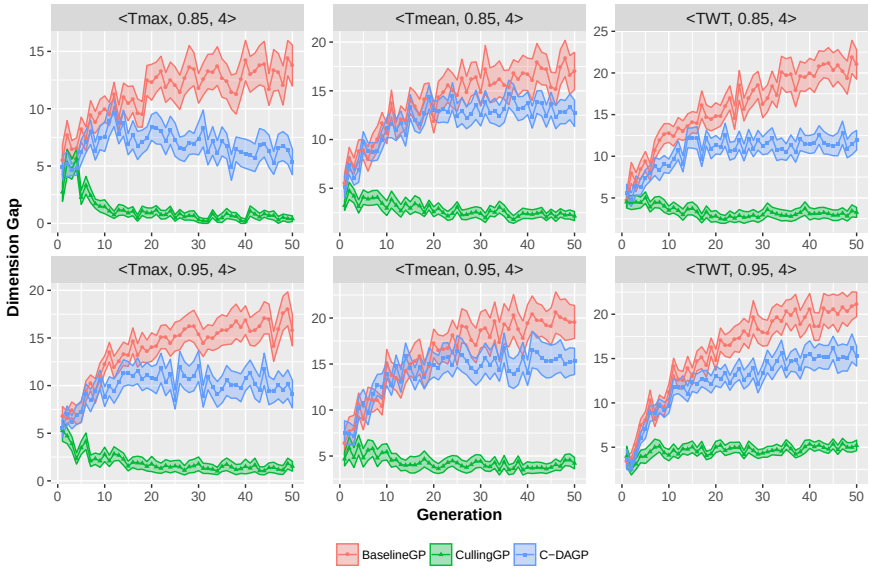

Fig. 3. The convergence curves (mean and standard error) of the **dimension gap** of BaselineGP, CullingGP and C-DAGP.

structure of the rule is as follows.

$$\begin{aligned}
&\texttt{rule} = \texttt{B}_1/\texttt{B}_2,\\
&\texttt{B}_1 = \texttt{max}((\texttt{SL}+\texttt{PT}) * \texttt{max}(\texttt{min}(\texttt{SL}, \texttt{WINQ}), \texttt{PT})/\texttt{WKR}, \texttt{PT}),\\
&\texttt{B}_2 = \texttt{W} * \texttt{WKR}/(\texttt{max}((\texttt{SL}+\texttt{PT}), \texttt{WKR}) * \mathbf{max}(\mathbf{W}, \mathbf{PT}))
\end{aligned}$$

One can see that the rule contains important features for the weighted tardiness (PT, W, SL, WINQ, WKR), and similar patterns to the WSPT rule (PT/W), which is a promising rule for minimising the weighted tardiness. The only dimension inconsistency occurred in $\texttt{max}(\texttt{W}, \texttt{PT})$ in $\texttt{B}_2$.

5 Conclusions and Future Work

In this paper, we propose a Constrained Dimensionally Aware GP (C-DAGP) to optimise both test performance and interpretability of job shop scheduling rules. Based on the physical dimension of the job shop attributes, we define a *dimension gap* measure to reflect the degree of interpretability of the evolved rules. Then, we develop a new penalty coefficient adaptation scheme to achieve a good balance between the performance and dimension gap during the GP search process. The experimental results show that the proposed C-DAGP sits between the baseline GP and an existing DAGP (CullingGP [11]) in terms of test performance and dimension gap. Although C-DAGP did not show consistent outperformance in both test performance and dimension gap, the new constrained optimisation framework enables a finer control on the balance between test performance and dimension gap.

In the future, we will investigate more penalty adaptation schemes to further improve the performance of C-DAGP. In addition, we will consider multi-objective frameworks and treat dimension gap as an objective rather than a constraint.

References

1. Babovic, V., Keijzer, M.: Genetic programming as a model induction engine. Journal of Hydroinformatics 2(1), 35–60 (2000)
2. Branke, J., Nguyen, S., Pickardt, C., Zhang, M.: Automated design of production scheduling heuristics: A review. IEEE Transactions on Evolutionary Computation 20(1), 110–124 (2016)
3. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. IEEE Transactions on Evolutionary Computation 18(2), 286–300 (2014)
4. Durasević, M., Jakobović, D., Knežević, K.: Adaptive scheduling on unrelated machines with genetic programming. Applied Soft Computing 48, 419–430 (2016)
5. Hildebrandt, T., Heger, J., Scholz-Reiter, B.: Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. In: Proceedings of Genetic and Evolutionary Computation Conference. pp. 257–264. ACM (2010)
6. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. Evolutionary computation 23(3), 343–367 (2015)

7. Hunt, R., Johnston, M., Zhang, M.: Evolving less-myopic scheduling rules for dynamic job shop scheduling with genetic programming. In: Proceedings of the 2014 conference on Genetic and evolutionary computation. pp. 927–934. ACM (2014)
8. Hunt, R., Johnston, M., Zhang, M.: Evolving dispatching rules with greater understandability for dynamic job shop scheduling. Tech. rep., Victoria University of Wellington, Wellington, NZ, Technical Report, ECSTR-15-6 (2015)
9. Jakobović, D., Budin, L.: Dynamic scheduling with genetic programming. In: Genetic Programming, pp. 73–84. Springer (2006)
10. Jayamohan, M., Rajendran, C.: New dispatching rules for shop scheduling: a step forward. International Journal of Production Research 38(3), 563–586 (2000)
11. Keijzer, M., Babovic, V.: Dimensionally aware genetic programming. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2. pp. 1069–1076. Morgan Kaufmann Publishers Inc. (1999)
12. Mei, Y., Nguyen, S., Zhang, M.: Evolving time-invariant dispatching rules in job shop scheduling with genetic programming. In: European Conference on Genetic Programming. pp. 147–163. Springer (2017)
13. Mei, Y., Zhang, M., Nyugen, S.: Feature selection in evolving job shop dispatching rules with genetic programming. In: Proceedings of Genetic and Evolutionary Computation Conference. pp. 365–372. ACM (2016)
14. Nguyen, S., Zhang, M., Johnston, M., Tan, K.: A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. IEEE Transactions on Evolutionary Computation 17(5), 621–639 (2013)
15. Nguyen, S., Mei, Y., Zhang, M.: Genetic programming for production scheduling: a survey with a unified framework. Complex & Intelligent Systems pp. 1–26 (2017)
16. 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)
17. Pickardt, C., Hildebrandt, T., Branke, J., Heger, J., Scholz-Reiter, B.: Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. International Journal of Production Economics 145(1), 67–77 (2013)
18. Pinedo, M.L.: Scheduling: theory, algorithms, and systems. Springer Science & Business Media (2012)
19. Poli, R., McPhee, N.F.: Parsimony pressure made easy. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation. pp. 1267–1274. ACM (2008)
20. Rajendran, C., Holthaus, O.: A comparative study of dispatching rules in dynamic flowshops and jobshops. European Journal of Operational Research 116(1), 156–170 (1999)
21. Riley, M., Mei, Y., Zhang, M.: Feature selection in evolving job shop dispatching rules with genetic programming. In: IEEE Congress on Evolutionary Computation. pp. 3362–3369. IEEE (2016)
22. Sels, V., Gheysen, N., Vanhoucke, M.: A comparison of priority rules for the job shop scheduling problem under different flow time-and tardiness-related objective functions. International Journal of Production Research 50(15), 4255–4270 (2012)
23. Xiong, J., Liu, J., Chen, Y., Abbass, H.A.: A knowledge-based evolutionary multiobjective approach for stochastic extended resource investment project scheduling problems. IEEE Transactions on Evolutionary Computation 18(5), 742–763 (2014)