Evolving Dispatching Rules for Dynamic Job Shop Scheduling with Uncertain Processing Times

Deepak Karunakaran, Yi Mei, Gang Chen and Mengjie Zhang
School of Engineering and Computer Science,
Victoria University of Wellington, PO Box 600, Wellington, New Zealand.
Email: {deepak.karunakaran, yi.mei, aaron.chen, mengjie.zhang}@ecs.vuw.ac.nz

***Abstract*—Dynamic Job shop scheduling (DJSS) is a complex and hard problem in real-world manufacturing systems. In practice, the parameters of a job shop like processing times, due dates, etc. are uncertain. But most of the current research on scheduling consider only deterministic scenarios. In a typical dynamic job shop, once the information about a job becomes available it is considered unchanged. In this work, we consider genetic programming based dispatching rules to generate schedules in an uncertain environment where the process time of an operation is not known exactly until it is finished. Our primary goal is to investigate methods to incorporate the uncertainty information into the dispatching rules. We develop two training approaches, namely *ex-post* and *ex-ante* to evolve the dispatching rules to generate good schedules under uncertainty. Both these methods consider different ways of incorporating the uncertainty parameters into the genetic programs during evolution. We test our methods under different scenarios and the results compare well against the existing approaches. We also test the generalization capability of our methods across different levels of uncertainty and observe that the proposed methods perform well. In particular, we observe that the proposed *ex-ante* training approach outperformed other methods.**

I. Introduction

Production scheduling faces a wide range of uncertainties e.g. machine breakdowns, fluctuating arrival rates of orders, uncertainty in processing times etc. But generally, the job shop parameters viz., processing times are assumed as constants under deterministic models [1].

In this work, we consider *dynamic* job shop scheduling (DJSS) with uncertain processing times. In a dynamic environment, the information about jobs is known only upon their arrival, but this information is constant. When we consider uncertain processing times (a parameter which has impact on most scheduling objectives), the value of the actual processing time of a job could be different from its *expected* value which becomes available when the job arrives.

The methods to describe uncertainty influence the techniques used toward optimization under uncertainty. Literature review [1] shows that: (1) $bounded\ \ form$, (2) $probability\ description$ and (3) $fuzzy\ description$ are the widely used methods for uncertainty description. The Literature review [1] also shows that the probability description is generally preferred.

A broad classification of approaches to scheduling under uncertainty would include $preventive\ scheduling$ and $reactive\ scheduling$. Preventive scheduling employs the historical information to generate schedules which are capable of absorbing the disruption caused by an uncertain event. Further classification of preventive scheduling includes stochastic scheduling [2], robust optimization [3], fuzzy programming [4] and sensitivity analysis [5]. Reactive scheduling modifies the generated schedule in the case of an uncertain event e.g. mixed integer linear programming in [6]. Most of the works under reactive scheduling are based primarily on using dispatching rules because they are efficient in generating schedules. Depending upon the scheduling objectives, different DRs are used in the job shops. Some of the popular DRs for dynamic job shop scheduling problem are $FIFO$ (first in first out), SPT (shortest processing time), $COVERT$ (cost over time) [7], ATC (apparent tardiness cost), etc.

Dispatching rules for dynamic job shop scheduling have shown promising results [8]. Literature surveys [9], [10], [11] show numerous approaches for job shop scheduling under uncertainty using dispatching rules. Lawrence et al. [11] show that with the increase in uncertainty in processing times, dispatching rules outperform optimization approaches like branch-and-bound. Pinedo and Weiss [12] show that for a certain class of scheduling problems on parallel machines "largest variance first" policy minimizes flow time and makespan. This has been considered as a dispatching rule in [11]. Later, Pinedo [13] shows that "smallest variance first" minimizes the expected makespan and total expected completion time for a certain class of stochastic batch scheduling problems. Matsuura et al. [10] show that their method using dispatching rules perform better in dynamic environments with rush jobs and specification changes (number of machines a job needs to visit).

Though dispatching rules are advantageous because they generate schedules instantaneously, their design involves rigorous experimental studies which is difficult. Genetic programming (GP) has been employed as a technique to automatically evolving dispatching rules [14], [15] which produce schedules of good quality. Many empirical studies [8], [16], [17] have been conducted to show the advantages of dispatching rules evolved using genetic programming. Even though genetic programming based hyper-heuristic approaches to evolving dispatching rules has shown promise recently, its application to evolve dispatching rules which are capable of generating quality schedules in a stochastic environment is still nascent. Recently, Karunakaran et al. [18] evolved dispatching rules using GP by considering terminals based on the moving

average of the disturbances in processing times.

Considering that dynamic job shop scheduling under uncertainty is a difficult problem, genetic programming is a key technology to evolve dispatching rules. In particular, the flexible representation of genetic programs could be exploited toward incorporating the uncertainty parameters of the shop into the dispatching rules. We consider this as a key research gap. Furthermore, the generalization of the evolved rules under different levels in processing time uncertainty is important for the dispatching rules to have practical utility.

A. Goals

In this work we consider the problem of dynamic job shop scheduling under uncertainty in processing times. We focus on developing methods which can incorporate the uncertainty information into the rules. We study the generalization of these rules for different configurations of processing time uncertainty. The following are the specific objectives.

1) Propose two new methods, *ex post* and *ex ante* training methods, to integrate the estimated uncertainty information into the dispatching rules. These are compared with the standard GP approach and the method presented in [18].
2) Study the generalization of these methods under different levels of uncertainty.

II. Background

A. Job Shop Scheduling

We briefly introduce the dynamic job shop scheduling problem (DJSS). Jobs arriving to the shop are assumed to follow Poisson process [8]. Every job j has n_j operations which must be processed in a predefined route. For a set of operations O_j in job j, a route could be defined as $(o_{j,1} \rightarrow o_{j,2} \rightarrow, \ldots, o_{j,n_j})$. Let $p_{j,i}$ be the processing time of the operation. In practice, the processing time realized in the shop, which is denoted as $p'_{j,i}$ is different from this value. The assumption that $p'_{j,i} > p_{j,i}$ is usually practical when considering events like repair and rework. Makespan, mean weighted tardiness and total flowtime are some of the scheduling objectives considered in literature. Total flowtime is defined as

$$F = \sum_j (C_j - r_j)$$

r_j and C_j are the release times and completion times of the jobs. We consider this objective for this work.

The general assumptions made about DJSS are followed in this work viz. no preemption, recirculation or machine failure and zero transit times between machines with no alternate routing.

Jobs arrive continuously to the shop and their operations are queued on the machines. Under the reactive scheduling methodology, dispatching rules are used to assign priorities to the operations queued on a machine. These priority values decide the sequence of operations to be processed on the machine.

B. Genetic Programming Based Hyper Heuristics (GPHHs)

Hyper-heuristics are techniques which search in the heuristic space rather than in the solution space. For hard problems like combinatorial optimization [19] ones, these methods are used to automate the design of the heuristics.

The genetic programming based hyper heuristic approaches [20] have shown good success. The flexible representation of the genetic program is very useful to represent a dispatching rule. This has compared well with linear and neural network representations [15]. Moreover, the variable tree depth and the ease of incorporating useful functions and terminals makes it possible to incorporate diverse characteristics [8] (e.g. multiple scheduling objectives) in dispatching rules. Considering the success of GPHH in evolving dispatching rules in dynamic scheduling environment [21], [8], [18] we develop new GPHH methods for DJSS under uncertain processing times.

C. Ex-post and Ex-ante Optimization

Our aim is to develop methods to incorporate the uncertainty information into the dispatching rules. In order to do that we consider the concepts of *ex-ante* and *ex-post* optimization. Ex-ante means 'before the event' and its antonym ex-post means 'after the event'. In ex-post optimization, decision is made after the uncertainty is revealed unlike the ex-ante optimization. These ideas originate in economic theory, first proposed by Myrdal [22] and subsequently considered in many areas including decision theory, game theory [23], bayesian statistics [24] and stochastic optimization problems [3], [25].

Kouvelis et al. [3] consider these ideas to define *robustness* of solutions to discrete optimization problems under uncertainty. For a decision obtained ex-ante the robustness is calculated by comparing the ex-ante decision against the optimal decision obtained ex-post (perfect information is known). Belien et al. [26] consider ex-post optimization to determine the optimal strategy in fantasy sport games toward team selection and management. Tapiero et al. [25] propose an ex-post inventory control approach for the just-in-time control problem, using the information after the demand uncertainty is realized. They compare it with ex-ante optimization approach which determines the optimal inventory control policy using the demand forecasts.

Similar to these approaches, we develop ex-post and ex-ante methods to evolve dispatching rules for scheduling under uncertainty, with minimizing total flowtime as our scheduling objective. We will describe our proposed methods in detail, in the next section.

III. Proposed Methods

We propose two new techniques to evolve dispatching rules toward generating good schedules for DJSS problem with uncertain processing times. Before we describe our methods, we briefly explain the simulation model which we used for our experiments.

A. Simulation Model

We assume that the uncertainty in processing times of the operations of a job follow the same distribution but the operations of different jobs could follow dissimilar distributions. This model is observed in practical scenarios e.g. [27]. Similar model is followed in [18]. For every operation $o_{j,i}$, the processing time without uncertainty $p_{j,i}$ and processing time with uncertainty $p'_{j,i}$ are related as

$$p'_{j,i} = (1 + \delta_{j,i})p_{j,i},\ \delta_{j,i} \geq 0.$$

Here, the delay ratio ($\delta_{j,i}$) is a measure of the level of severity in the disturbances in processing times. We provide further details of the simulation model when we explain our experimental design (Section IV).

B. Ex-post Training (EXP) Method

One way of evolving dispatching rules for DJSS under uncertainty is by incorporating the uncertainty information into terminals of genetic programs. For example, Karunakaran et al. [18] have proposed an exponential moving average (EMA) terminal to incorporate the uncertainty information into the genetic programs. Though this method compared well against the standard GP approach in evolving rules with the desired characteristics, the heuristic search is rendered more complex with the addition of new terminals. This increases the difficulty for the evolutionary algorithms to find good solutions.

Therefore, instead of using a terminal we propose an ex-post training approach. Like in ex-post optimization, we evolve the dispatching rules using training instances which consider the processing time values **after** the realization of uncertainty. This is different from the standard GP method, where the dispatching rules are trained using the processing times which are uncertain and the actual value is known after the schedule has been generated. Consequently, EXP training approach results in dispatching rules which perform well on the instances for which the realized processing times are known. In order to use these dispatching rules on test instances, we replace the processing time terminal with its estimated realized value. This is explained below.

Suppose, $\mathcal{D}^{EXP}$ is a dispatching rule evolved using ex-post training approach, by using the realized processing times. When $\mathcal{D}^{EXP}$ is used to generate schedules for test instances, each processing time terminal PT^{exp} is replaced by the modified terminal PT^* such that:

$$PT^* = p_{j,i}(1 + \delta_{j,i}) \tag{1}$$

In order to determine the value of $\delta_{j,i}$ we consider the exponential moving average(EMA) [18] value of the delay ratio which is explained as follows. $\delta_{j,i}$ is calculated for each job, with the assumption that the disturbances in the processing times of the operations of a particular job follow a particular distribution. This assumption leads us to use the exponential moving average as the mean value of $\delta_{j,i}$s which is denoted as $\bar{\delta}_{j,i}$:

$$\bar{\delta}_{j,1} = 0, \tag{2}$$

$$\bar{\delta}_{j,i+1} = \kappa \times \delta_{j,i} + (1 - \kappa) \times \bar{\delta}_{j,i}, \forall i > 1 \tag{3}$$

$\kappa = 0.2$ is considered in the experiments [18]. The experimental results of [18] show that the uncertainty information could be reliably estimated using the moving average method for each job. We illustrate the replacement of the terminal using an evolved rule in Section V.

Table I: **PT** terminals for ex-ante and ex-post approaches.

Dispatching rule	train	test
D^{EXP}	PT^{exp}	PT^*
D^{ENT}	PT^*	PT^*

C. Ex-ante Training (ENT) Method

In this method we consider ex-ante optimization where the objective is to determine the optimal solution without knowing the outcome but using an estimate of the future outcome. The dispatching rules are trained using problem instances by using an estimate of the processing times. The processing time terminal used while training using ex ante approach is PT^*, as shown in Equation 1. The estimation of $\delta_{j,i}$ is done as explained in the previous method using the Equations 2-3.

The different terminals used with the training and testing instances for the two approaches are shown in Table I. For the rules evolved using ex-ante training approach, D^{ENT}, both the training and test runs consider the same terminals. But for the ex-post training approach, D^{EXP}, there is a clear distinction in the terminals for training and testing runs. In contrast to the method used in [18], which we denote as EMA, in D^{ENT} the processing time terminal is modified instead of using an additional terminal.

IV. Experiment Design

The aim of our experiments is two-fold. Firstly we compare our proposed methods with standard GP which does not consider incorporation of uncertainty information. We also compare with the approach described in [18] which propose the use of EMA terminal. Secondly, we aim to analyze the generalization of our methods on DJSS under uncertain processing times.

We describe our experiment starting with the simulation configuration. We use a discrete event simulation system (see Jasima [28]) to generate problem instances of DJSS problems. Each problem instance is made of ten machines and eight operations per job. The processing times of the operations are sampled uniformly from $[1, 49]$. This configuration has been considered in other studies [8]. The arrival of the jobs follows a Poisson process with a rate $\lambda = 0.85$ [18].

We consider total flowtime as the objective for minimization and the jobs from $500 - 2000$ are considered for analysis, out of a 2500 total jobs arriving at the shop [8]. The delay factor

$\delta_{j,i}$ follows a Gamma distribution. The different gamma distributions are obtained by varying its parameters, namely scale (β) and shape (α). The specific training and test configurations are described below.

A. Training and Test Configurations

We considered different settings of parameters for generating the problem instances for testing and training. By associating a particular level of uncertainty to a job, we characterize a job type. Then by varying the ratio of the job types we create problem instances. These problem instances enable us to simulate the variability in the shop. They are described below.

For **training**, we consider two configurations by considering different number of job types. We use these two training configurations because they represent problem instances with contrasting variability in job shop.

- In the first training configuration, two job types are considered equi-proportion such that delay ratio follows the gamma distribution with $(\alpha = 1, \beta = 0.6)$ and $(\alpha = 1, \beta = 0.1)$ respectively.
- In the second training configuration, five job types are considered equi-proportion and the gamma distribution parameters are $\alpha = 1, \beta \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$

For **testing**, the configurations are give in Table II. The last column #job-type-ratio tells the ratio between the number of jobs in each type.

B. Genetic Programming System

A list of the functions and terminals is given in Table III. The function if has 3 argument; it returns second argument if the first is greater than 0, and the third one otherwise. The population size is 1024, generation count is 50, maximal tree depth is 8, crossover rate is 0.85, mutation and elitism are 0.1 and 0.05 respectively [8].

V. Results and Discussion

In this Section, we present the results from our experiments. The notation used for different methods is given in Table IV. Since we have used two training configurations, we suffix the method name with '2' or '5' depending on the #job-types used for training. E.g. ENT2 means ex-ante training approach using the training instances with 2 job-types (the first training configuration as explained in Section IV-A).

As seen in Table IV, we have 4 methods to compare. For each method, we consider 2 training configurations. This leads to 8 sets of results collected in Tables V- IX for a total of 7 test configurations. Each element of the table is represented as a triplet in the form of $[Win-Draw-Lose]$ which is a result of the Wilcoxon-rank-sum-test under a significance level of 0.05, used to compare the 30 test instances for each test configuration. For example, consider the column corresponding to the test configuration III and the row corresponding to $ENT5$ in Table V. $[2-25-3]$ means that $ENT2$, when compared with $ENT5$, performs significantly better in 2 test instances, poorly in 3 test instances and shows similar performance in the remaining 25; among a total of 30 test problem instances. Thus we have done an exhaustive comparison across all methods with varied test and training configurations.

Table II: Test Configurations

Test-set	scale (β)	#job-type-ratio
I	$\{0.1, 0.6\}$	$1:1$
II	$\{0.1, 0.6\}$	$2:1$
III	$\{0.1, 0.6\}$	$3:1$
IV	$\{0.1, 0.3, 0.6\}$	$1:1:1$
V	$\{0.1, 0.3, 0.6, 0.8\}$	$1:1:1:1$
VI	$\{0.1, 0.2, 0.3, 0.4, 0.5\}$	$1:1:1:1:1$
VII	$\{0.1, 0.3, 0.6, 0.8, 1.2\}$	$1:1:1:1:1$

Table III: Function and Terminal Sets for GP.

Function Set	Meaning
$+$	Addition
$-$	Subtraction
$*$	Multiplication
$/$	Protected Division
Max	Maximum
Min	Minimum
If	Conditional
Terminal Set	**Meaning**
DD	Due date of job
PT	Processing time of operation
RO	Remaining operations for job
RJ	Ready time of job
RT	Remaining processing time of job
RM	Ready time of machine
ERC	Ephemeral Random constant
EMA	Exponential moving average of delay ratio*
	*(terminal not used with standard GP)

Table IV: Notation

	Description
ENT	Ex-ante training approach.
EXP	Ex-post training approach.
EMA	Using exponential moving average terminal. [18]
NAT	Standard GP approach (No additional terminals).

Firstly, we compare the proposed methods with standard GP approach (NAT) and the approach using EMA terminal described in III-B. Secondly, we present our observations while comparing the two proposed methods. Thirdly, we consider the generalization capabilities by comparing the evolved rules associated with different training configurations. Finally, we analyze the evolved dispatching rules to get more insights.

A. Comparison with EMA/NAT

We present our observations for the comparisons with the existing methods. We use the first two plots in Figure 1 for this discussion. In each plot, a group of boxplots pertain to a single problem test instance. For example, in the first plot, with the caption $[ENT2-EXP2-EMA2-NAT2](TestSet-I)$, the boxplots are grouped 4 at a time. The caption also says that the test problem instances belong to test configuration I. In every group, the order of methods is the same as mentioned in the caption. The method(s) which performs the best are colored in yellow. So in the first group of boxplots, $ENT2$ outperforms all other methods. To get a complete result on all 30 problems refer the Tables V - VIII.

In particular, if we focus on the two-job-configuration, i.e. $ENT2$, the results are much better than the standard GP ($NAT2$) and the exponential moving average terminal based approach ($EMA2$) for test configurations I-V. This can be inferred from the first five columns in the second and third rows of Table V. The corresponding cells are shown in bold. Similarly ex-post training approach ($EXP2$) outperforms $EMA2$ and $NAT2$ for the same test configurations. The corresponding cells are shown in bold in Table VII. Therefore, both our proposed methods show significant improvement when compared against the existing methods.

When the training configurations VI-VII are considered while comparing $ENT5/EXP5$, none of the methods is a clear winner. In fact, in some cases the proposed method under-performs e.g. see the cells marked in blue in Table VI. This is understandable, as five-job-type configuration is associated with higher variability in the job shop and evolving good rules using the estimation of uncertainty parameter yields poor results, due to difficulty in accurate estimation.

B. ex-post vs ex-ante

When we compare our two methods with each other, we observe that generally both of them are similar across all the test configurations. Though $ENT2$ is slightly better than $EXP2$ and $ENT5$ is slightly better than $EXP5$. This can be observed in the first plot of Figure 1. And more clearly in #row-1 and #row-5 in Tables V & VI respectively. The corresponding rows are shown in grey.

C. Generalization

Till now we presented the observations of comparison among methods that use identical training configurations. In order to determine the generalization of our methods, we compare dispatching rules evolved using different training configurations.

1) two-job-type-configuration: Firstly, we consider the two-job-type-configuration which is associated with less variability in the job shop. The third plot in Figure 1 gives an intuition to the comparison, where we compare the five methods against each other on the test configuration I. If we compare $EXP5/ENT5$ with $EXP2/ENT2$ on test configurations I-V and consider the comparison of $EMA5$ with $EMA2$, we can conclude that the generalization characteristics of our methods are better than EMA.

We explain this observation using some of the examples. Consider the cells marked in green in Tables VI, VIII & X. The cells corresponding to test configuration I are [0-18-12], [0-27-3] and [0-10-20] for ENT5, EXP5 and EMA5 respectively. The generalization of EMA5 is the worst among the three methods in this example. The same trend could be observed in other columns too.

2) five-job-type-configuration: Refer the fourth plot in Figure 1. Now we try to compare the performance of $ENT2/EXP2/EMA2/NAT2$ against $ENT5/EXP5/EMA5/NAT5$ on test configurations VI-VII. Referring cells marked in Tables V, VII & IX, the generalization characteristics of our methods show high competitiveness.

D. Analysis

To summarize, we observed that for the scheduling objective of total flowtime, the performance of the different methods is approximately in the following order.

$$ENT > EXP >> EMA > NAT$$

We analyze our key results and try to gain more insights to our methods. Firstly, we show one of the best dispatching rules evolved using the EXP method and illustrate how the processing time terminals are changed at the time of using the rule to generate schedules on test cases. The Dispatching Rule shown below is the rule which was evolved during training. The terminals PT^{exp}, which are shown in bold are replaced by PT^* (refer Equation 1)

Dispatching Rule 1: EXP method - Training

(If (Max (If (+ PT^{exp} (If RO RM RM))(- (Max RJ RT) (Max 0.002 RM)) (If DD 0.59 0.47))(+ (- (* DD RO) (Max 0.002 RM))(Max RJ RT))) (* (/ (If (* DD RO) (* RO PT^{exp}) (If DD RJ 0.72)) (* (/ 0.78 PT^{exp}) (- 0.24 0.28))) (If (/ (/ DD RJ) (-RM PT^{exp})) (/ (If RM RT RM) (If RJ RJ RO)) (If RM RT RM))) (If (/ RJ RT) (+ PT^{exp} RO) (* DD RO)))

Secondly, we determine the frequency of terminals in 30 best rules evolved using each method. The histogram plot is shown in Figure 2. The histogram plot shows that there is a consistent pattern for the frequency of terminals among all methods. Since the maximum depth of the genetic programs is fixed, the introduction of terminal EMA reduces the number of other terminals in the genetic programs. Also it makes the search space more complex due to an extra terminal. These observations point to the reason for the proposed algorithms to outperform EMA.

Thirdly, we look at the performance of the evolved rules on the training instance. The box-plots for this comparison is shown in Figure. 3. The plot shows the performance of best rules evolved from the 30 independent runs. As expected, the *ex-post* training approach corresponding to EXP method shows the best performance on training instance as it uses the realized processing time. We believe that EXP method suffers from high-variance problem where as the method ENT performs well because it simultaneously considers the estimation component during training.

VI. Conclusions

In this work, we present genetic programming based approaches for dynamic job shop scheduling under uncertain processing times. We develop an ex-post and an ex-ante training approach to incorporating uncertainty information into dispatching rules. In the ex-post training approach, the dispatching rules were evolved using the realized uncertainty information. The processing time terminals are then replaced with the estimated values during testing and practical use. In the ex-ante training approach, the processing time terminals

Table V: ENT-2 (Ex-ante method, 2-job types training)

	I	II	III	IV	V	VI	VII
EXP-2	[13-17-0]	[5-25-0]	[2-28-0]	[8-22-0]	[4-26-0]	[5-25-0]	[0-30-0]
EMA-2	**[22-8-0]**	**[29-1-0]**	**[29-1-0]**	**[29-1-0]**	**[24-6-0]**	[0-29-1]	[6-17-7]
NAT-2	**[28-2-0]**	**[30-0-0]**	**[30-0-0]**	**[29-1-0]**	**[29-1-0]**	[3-24-3]	[11-13-6]
ENT-5	[12-18-0]	[2-28-0]	[2-25-3]	[3-27-0]	[8-22-0]	[6-24-0]	[6-24-0]
EXP-5	[19-11-0]	[9-21-0]	[1-29-0]	[10-20-0]	[25-5-0]	[11-19-0]	[6-24-0]
EMA-5	[27-3-0]	[29-1-0]	[30-0-0]	[29-1-0]	[29-1-0]	[2-19-9]	[15-12-3]
NAT-5	[27-3-0]	[30-0-0]	[29-1-0]	[29-1-0]	[30-0-0]	[4-22-4]	[13-12-5]

Table VI: ENT-5 (Ex-ante method, 5-job types training)

	I	II	III	IV	V	VI	VII
ENT-2	[0-18-12]	[0-28-2]	[3-25-2]	[0-27-3]	[0-22-8]	[0-24-6]	[0-24-6]
EXP-2	[1-29-0]	[2-28-0]	[7-23-0]	[3-27-0]	[0-30-0]	[4-25-1]	[0-30-0]
EMA-2	[12-17-1]	[28-2-0]	[27-3-0]	[25-5-0]	[13-17-0]	[1-24-5]	[1-12-17]
NAT-2	[25-5-0]	[30-0-0]	[29-1-0]	[27-3-0]	[23-7-0]	[2-21-7]	[7-11-12]
EXP-5	[2-28-0]	[4-26-0]	[6-24-0]	[0-29-1]	[3-27-0]	[1-29-0]	[0-30-0]
EMA-5	[25-5-0]	[29-1-0]	[29-1-0]	[28-2-0]	[26-4-0]	**[2-11-17]**	[11-14-5]
NAT-5	[25-5-0]	[29-1-0]	[29-1-0]	[28-2-0]	[25-5-0]	**[3-18-9]**	[9-12-9]

Table VII: EXP-2 (Ex-post method, 2-job types training)

	I	II	III	IV	V	VI	VII
ENT-2	[0-17-13]	[0-25-5]	[0-28-2]	[0-22-8]	[0-26-4]	[0-25-5]	[0-30-0]
EMA-2	**[12-17-1]**	**[28-2-0]**	**[20-10-0]**	**[22-8-0]**	**[17-13-0]**	[0-24-6]	[2-21-7]
NAT-2	**[27-3-0]**	**[30-0-0]**	**[28-2-0]**	**[27-3-0]**	**[26-4-0]**	[2-19-9]	[8-16-6]
ENT-5	[0-29-1]	[0-28-2]	[0-23-7]	[0-27-3]	[0-30-0]	[1-25-4]	[0-30-0]
EXP-5	[3-27-0]	[1-28-1]	[0-27-3]	[0-30-0]	[6-24-0]	[2-27-1]	[0-30-0]
EMA-5	[26-4-0]	[28-2-0]	[26-4-0]	[28-2-0]	[27-3-0]	[1-11-18]	[12-15-3]
NAT-5	[25-5-0]	[30-0-0]	[28-2-0]	[28-2-0]	[29-1-0]	[2-16-12]	[8-16-6]

Table VIII: EXP-5 (Ex-post method, 5-job types training)

	I	II	III	IV	V	VI	VII
ENT-2	[0-11-19]	[0-21-9]	[0-29-1]	[0-20-10]	[0-5-25]	[0-19-11]	[0-24-6]
EXP-2	[0-27-3]	[1-28-1]	[3-27-0]	[0-30-0]	[0-24-6]	[1-27-2]	[0-30-0]
EMA-2	[7-21-2]	[28-2-0]	[24-6-0]	[22-8-0]	[9-21-0]	[0-20-10]	[3-14-13]
NAT-2	[26-4-0]	[30-0-0]	[30-0-0]	[27-3-0]	[19-11-0]	[2-18-10]	[9-13-8]
ENT-5	[0-28-2]	[0-26-4]	[0-24-6]	[1-29-0]	[0-27-3]	[0-29-1]	[0-30-0]
EMA-5	[25-5-0]	[29-1-0]	[28-2-0]	[27-3-0]	[21-9-0]	[1-10-19]	[12-13-5]
NAT-5	[24-6-0]	[29-1-0]	[28-2-0]	[27-3-0]	[22-8-0]	[2-15-13]	[10-14-6]

Table IX: EMA-2 (Exponential moving avg. method, 2-job types training)

	I	II	III	IV	V	VI	VII
ENT-2	[0-8-22]	[0-1-29]	[0-1-29]	[0-1-29]	[0-6-24]	[1-29-0]	[7-17-6]
EXP-2	[1-17-12]	[0-2-28]	[0-10-20]	[0-8-22]	[0-13-17]	[6-24-0]	[7-21-2]
NAT-2	[24-6-0]	[17-13-0]	[18-12-0]	[16-14-0]	[12-18-0]	[2-27-1]	[11-19-0]
ENT-5	[1-17-12]	[0-2-28]	[0-3-27]	[0-5-25]	[0-17-13]	[5-24-1]	[17-12-1]
EXP-5	[2-21-7]	[0-2-28]	[0-6-24]	[0-8-22]	[0-21-9]	[10-20-0]	[13-14-3]
EMA-5	[20-10-0]	[13-17-0]	[6-24-0]	[15-15-0]	[23-7-0]	[0-21-9]	[26-4-0]
NAT-5	[23-7-0]	[19-11-0]	[13-17-0]	[19-11-0]	[22-8-0]	[0-28-2]	[12-18-0]

Table X: EMA-5 (Exponential moving avg. method, 5-job types training)

	I	II	III	IV	V	VI	VII
ENT-2	[0-3-27]	[0-1-29]	[0-0-30]	[0-1-29]	[0-1-29]	[9-19-2]	[3-12-15]
EXP-2	[0-4-26]	[0-2-28]	[0-4-26]	[0-2-28]	[0-3-27]	[18-11-1]	[3-15-12]
EMA-2	[0-10-20]	[0-17-13]	[0-24-6]	[0-15-15]	[0-7-23]	[9-21-0]	[0-4-26]
NAT-2	[0-30-0]	[2-28-0]	[8-22-0]	[0-30-0]	[0-28-2]	[3-27-0]	[0-18-12]
ENT-5	[0-5-25]	[0-1-29]	[0-1-29]	[0-2-28]	[0-4-26]	[17-11-2]	[5-14-11]
EXP-5	[0-5-25]	[0-1-29]	[0-2-28]	[0-3-27]	[0-9-21]	[19-10-1]	[5-13-12]
NAT-5	[0-30-0]	[2-28-0]	[1-29-0]	[1-29-0]	[0-30-0]	[3-27-0]	[0-23-7]

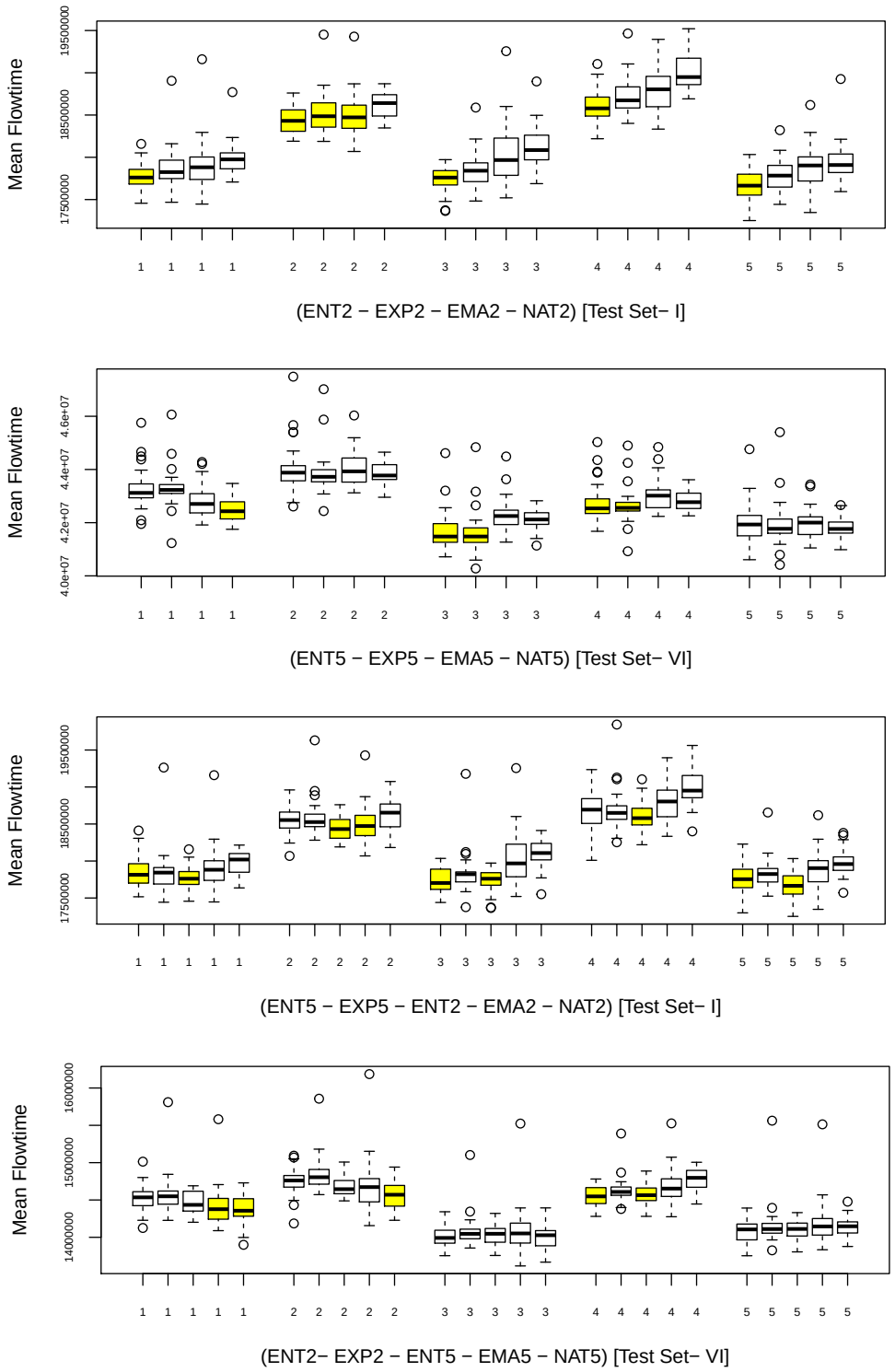

Figure 1: Boxplots: The order of boxplots is same as mentioned in the caption. The significantly better result is shown with a colored boxplot.

used during training encapsulate the estimated uncertainty information. We test our methods on a number of configurations with varying uncertainty levels.

The results show that our methods outperformed the standard GP approach and the EMA method. By using different configurations of uncertainty, we showed that our methods have better generalization characteristics. In particular, the ex-ante training approach was observed to outperform all other methods. With the ex-ante approach, the evolved rules also showed better generalization characteristics.

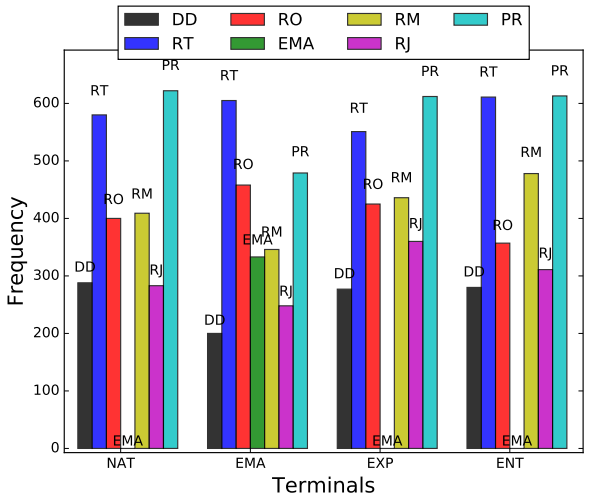

Figure 2: Histogram of Frequency of Terminals

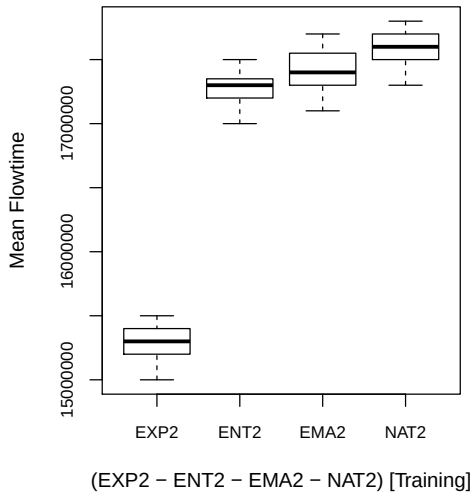

Figure 3: Boxplots for training instance

Since both our approaches rely on estimation of the uncertainty information, we will consider improving the estimation methods in our future work. Furthermore, the source of uncertainty considered in processing times is job related. Uncertainty related to machines e.g. deterioration in machine tool quality etc. will be considered in our future work.

References

[1] Z. Li and M. Ierapetritou, "Process scheduling under uncertainty: Review and challenges," *Computers & Chemical Engineering*, vol. 32, no. 4, pp. 715–727, 2008.

[2] J. Balasubramanian and I. Grossmann, "Approximation to multistage stochastic optimization in multiperiod batch plant scheduling under demand uncertainty," *Industrial & engineering chemistry research*, vol. 43, no. 14, pp. 3695–3713, 2004.

[3] P. Kouvelis and G. Yu, *Robust discrete optimization and its applications*. Springer Science & Business Media, 2013, vol. 14.

[4] P. Fortemps, "Jobshop scheduling with imprecise durations: a fuzzy approach," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 4, pp. 557–569, 1997.

[5] B. Penz, C. Rapine, and D. Trystram, "Sensitivity analysis of scheduling algorithms," *European Journal of Operational Research*, vol. 134, no. 3, pp. 606–615, 2001.

[6] M. Rodrigues, L. Gimeno, C. Passos, and M. Campos, "Reactive scheduling approach for multipurpose chemical batch plants," *Computers & chemical engineering*, vol. 20, pp. S1215–S1220, 1996.

[7] G. Salvendy, *Handbook of industrial engineering: technology and operations management*. John Wiley & Sons, 2001.

[8] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," *Evolutionary Computation, IEEE Transactions on*, vol. 17, no. 5, pp. 621–639, 2013.

[9] K. Z. Gao, P. N. Suganthan, M. F. Tasgetiren, Q. K. Pan, and Q. Q. Sun, "Effective ensembles of heuristics for scheduling flexible job shop problem with new job insertion," *Computers & Industrial Engineering*, vol. 90, pp. 107–117, 2015.

[10] H. Matsuura, H. Tsubone, and M. Kanezashi, "Sequencing, dispatching and switching in a dynamic manufacturing environment," *The International Journal of Production Research*, vol. 31, no. 7, pp. 1671–1688, 1993.

[11] S. R. Lawrence and E. C. Sewell, "Heuristic, optimal, static, and dynamic schedules when processing times are uncertain," *Journal of Operations Management*, vol. 15, no. 1, pp. 71–82, 1997.

[12] M. Pinedo and G. Weiss, "The largest variance first policy in some stochastic scheduling problems," *Operations Research*, vol. 35, no. 6, pp. 884–891, 1987.

[13] M. Pinedo, "Stochastic batch scheduling and the "smallest variance first" rule," *Probability in the Engineering and Informational Sciences*, vol. 21, no. 04, pp. 579–595, 2007.

[14] D. Jakobović, L. Jelenković, and L. Budin, "Genetic programming heuristics for multiple machine scheduling," in *Genetic Programming*. Springer, 2007, pp. 321–330.

[15] J. Branke, T. Hildebrandt, and B. Scholz-Reiter, "Hyper-heuristic evolution of dispatching rules: A comparison of rule representations," *Evolutionary computation*, vol. 23, no. 2, pp. 249–277, 2015.

[16] J. A. Vazquez-Rodriguez and G. Ochoa, "On the automatic discovery of variants of the neh procedure for flow shop scheduling using genetic programming," *Journal of the Operational Research Society*, vol. 62, no. 2, pp. 381–396, 2011.

[17] W.-J. Yin, M. Liu, and C. Wu, "Learning single-machine scheduling heuristics subject to machine breakdowns with genetic programming," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 2. IEEE, 2003, pp. 1050–1055.

[18] D. Karunakaran, Y. Mei, G. Chen, and M. Zhang, "Dynamic job shop scheduling under uncertainty using genetic programming," *Intelligent and Evolutionary Systems*, p. 195, 2016.

[19] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[20] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational intelligence*. Springer, 2009, pp. 177–201.

[21] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010, pp. 257–264.

[22] G. Myrdal, *Monetary equilibrium*. London: W. Hodge, 1939.

[23] M. Aghassi and D. Bertsimas, "Robust game theory," *Mathematical Programming*, vol. 107, no. 1-2, pp. 231–273, 2006.

[24] G. Koop, D. J. Poirier, and J. L. Tobias, *Bayesian econometric methods*. Cambridge University Press, 2007.

[25] C. S. Tapiero, "Ex-post inventory control," *International journal of production research*, vol. 38, no. 6, pp. 1397–1406, 2000.

[26] J. Beliën, D. Goossens, and D. Van Reeth, "A mixed integer programming model for ex post optimization in fantasy sport games," in *European Conference on Operational Research (EURO XXVI-2013)*, 2013, pp. 423–423.

[27] S. Rai, C. B. Duke, V. Lowe, C. Quan-Trotter, and T. Scheermesser, "Ldp lean document production-or-enhanced productivity improvements for the printing industry," *Interfaces*, vol. 39, no. 1, pp. 69–90, 2009.

[28] T. Hildebrandt, "Jasima – an efficient java simulator for manufacturing and logistics," *http://code.google.com/p/jasima*, 2012.