2022 IEEE 18th International Conference on Automation Science and Engineering (CASE 2022)

A Simulation Optimization-Aided Learning Method for Design Automation of Scheduling Rules

Hang Ma
Joint Work with Cheng Zhang and Zhongshun Shi

Intelligence, Dynamics, Emulation and Automation for Systems (IDEAS) Lab
Department of Industrial and Systems Engineering
University of Tennessee Knoxville

hma19@vols.utk.edu

August 23, 2022

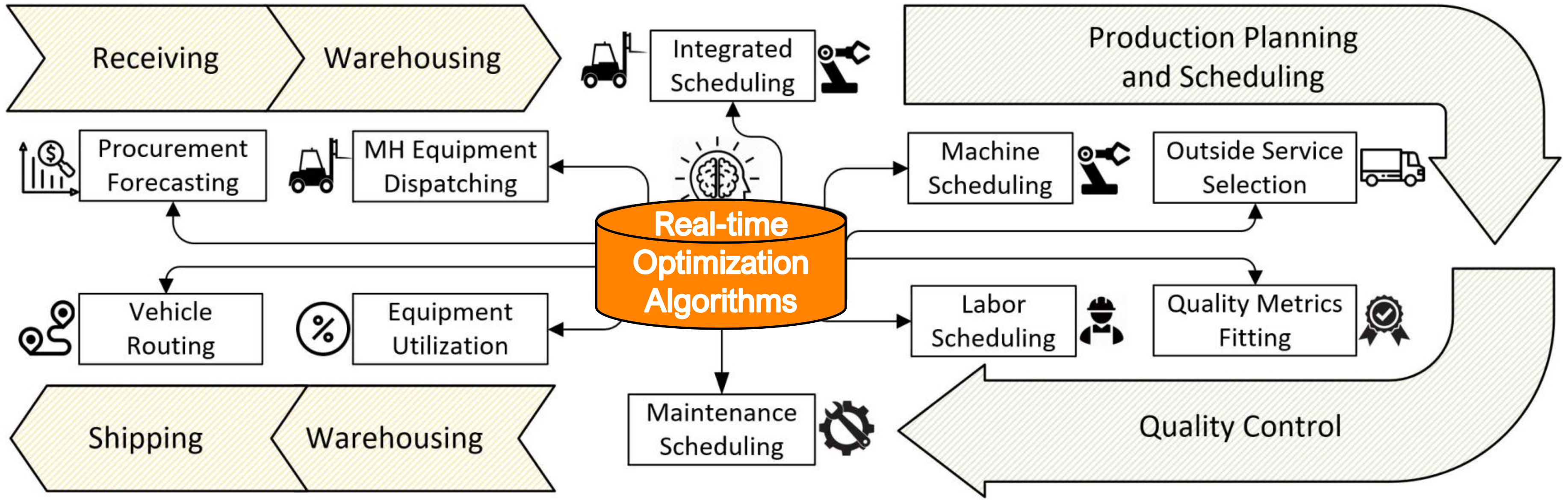
Outline

1. Introduction
2. Testbed Problem: Dynamic Job Shop Scheduling
3. Proposed Method: Simulation Optimization-aided Learning (SOaL)
4. Computational Experiments
5. Conclusions and Future Work

THE UNIVERSITY OF TENNESSEE KNOXVILLE

1. Introduction

- **Real-time optimization** in daily operations management of manufacturing facilities



- Challenges:
  - Almost all the dispatching rules (DR) are manually designed
  - Time consuming
  - Large loss of accuracy for complex problems

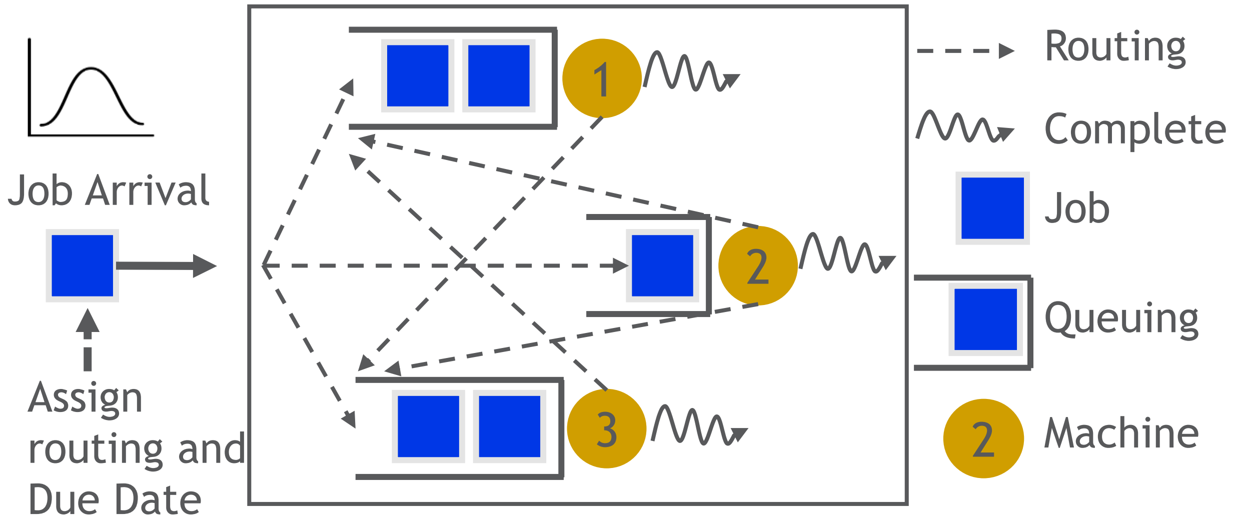
1. Introduction

- Related work
  - **Machine learning for automated design of scheduling rules**: genetic programming (Branke et al., 2015), neural network (Gupta et al., 2020), and reinforcement learning (Zhang et al., 2020).
  - **Genetic programming (GP):** advantages in **flexibility** and **interpretability**, widely used in academia and industry, e.g., Nguyen et al., (2013), Pickardt et al., (2013) and Shi et al., 2021.
  - **Rule evaluation of GP**: most studies of GP focus on rule generation.
    - In Hildebrandt et al., (2010), improvements were obtained by using replications of random seed in each generation.
    - In Shi et al., (2019), a novel ranking and selection algorithm was proposed to improve the rule selection efficiency in each generation.

- Research gap
  - Few research studies the possibility to enhance the rule evaluation in the final stage for evaluating all the generated dispatching rules

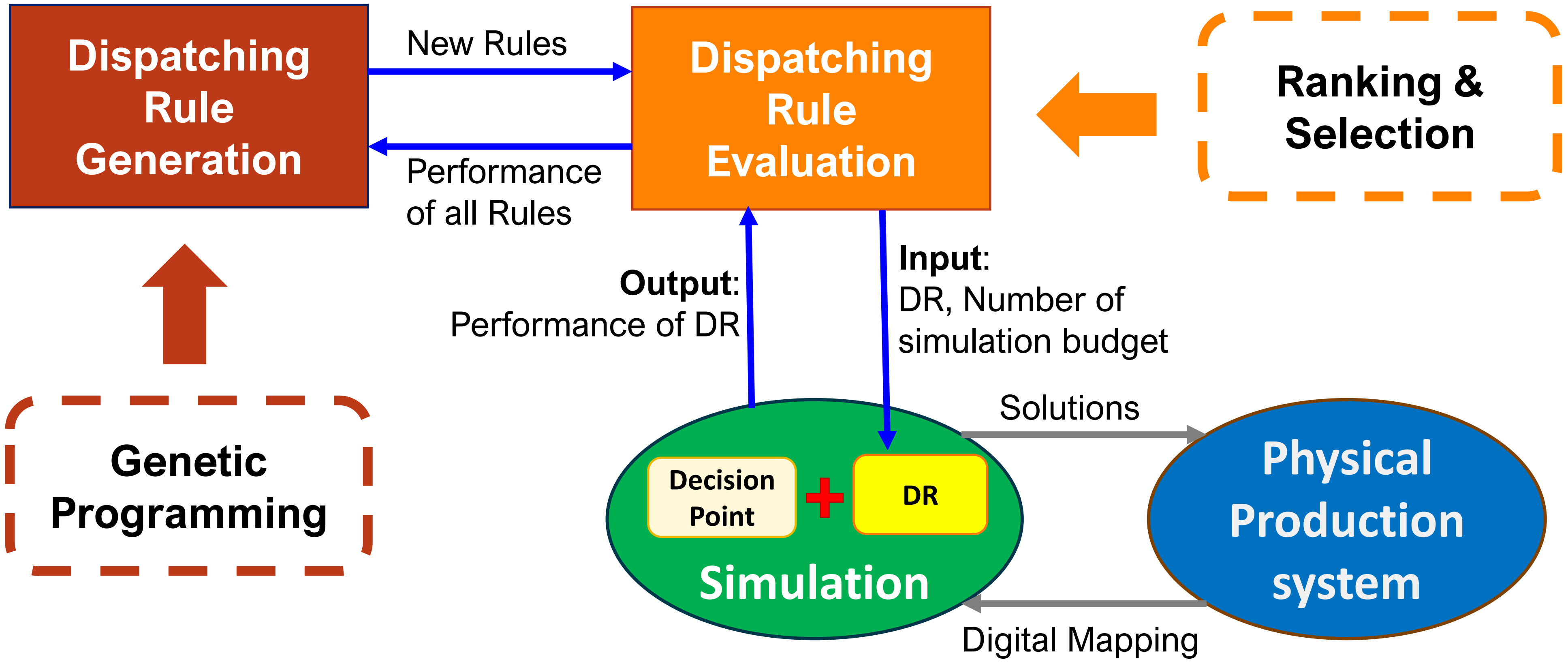
2. Testbed Problem: Dynamic Job Shop Scheduling

- Job and machine environment
  - J: a set of n jobs
  - M: a set of m machines
  - O_j: a sequence of operations (routing) of job j that can only be determined when the job arrives at a dynamic release time according to some stochastic process
  - p_{ji}: processing time of operation i of job j on a specific machine
  - No preemption and interruption



- Objectives: we consider two objectives in this paper
  1. Minimize flowtime $\bar{F}$
  2. Minimize number of tardy jobs $\bar{T}$

3. Proposed Method: Simulation Optimization-aided Learning

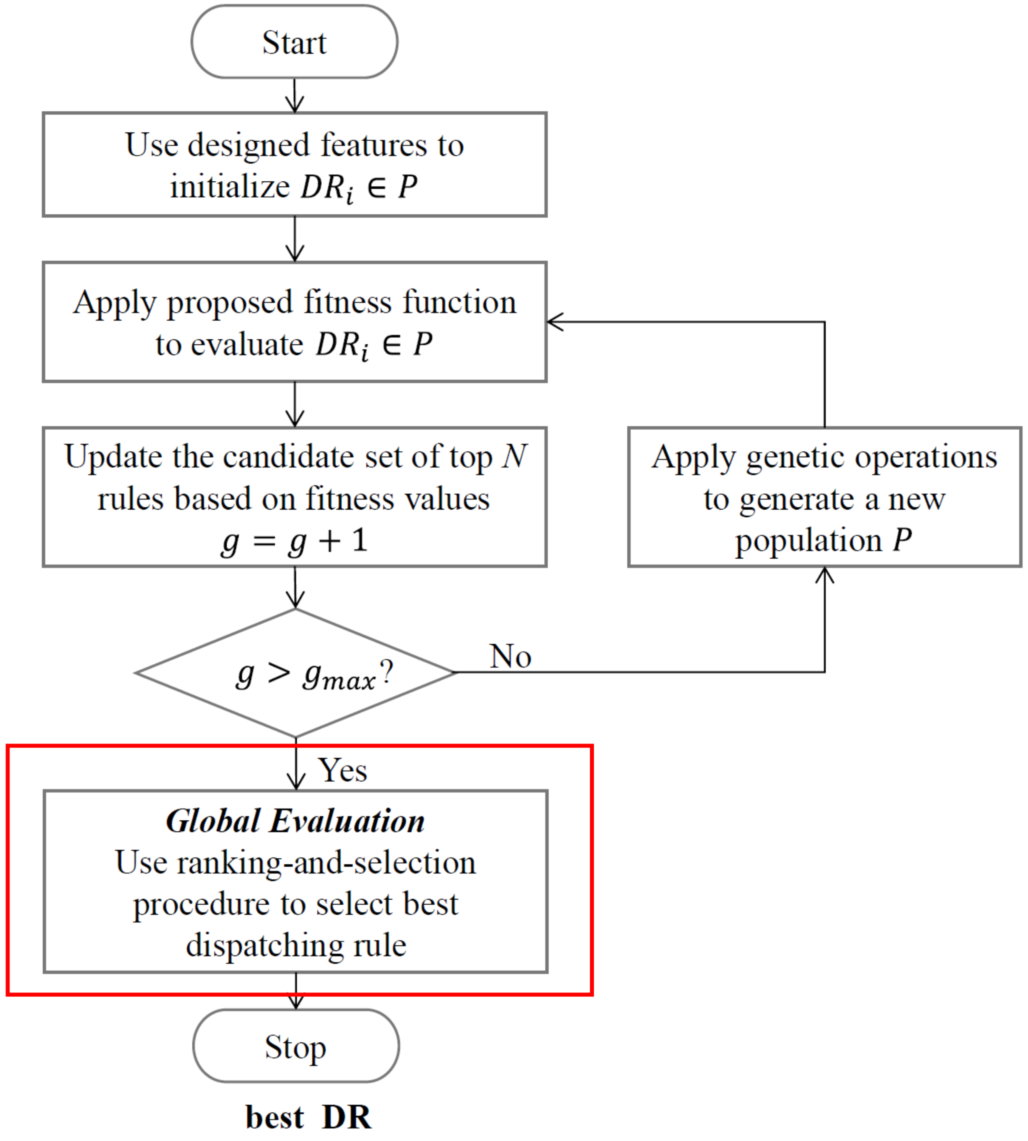


Dispatching Rule Generation
New Rules
Performance of all Rules
Dispatching Rule Evaluation
Ranking & Selection
Genetic Programming
Output: Performance of DR
Input: DR, Number of simulation budget
Decision Point
DR
Simulation
Solutions
Digital Mapping
Physical Production system

Main idea of SOaL: we treat the automated design of scheduling rules as a **simulation optimization problem**, use **genetic programming** algorithm to guide the rule generation, and introduce **ranking and selection** algorithm to improve the rule evaluation accuracy.

3.1 SOaL: Main Procedure

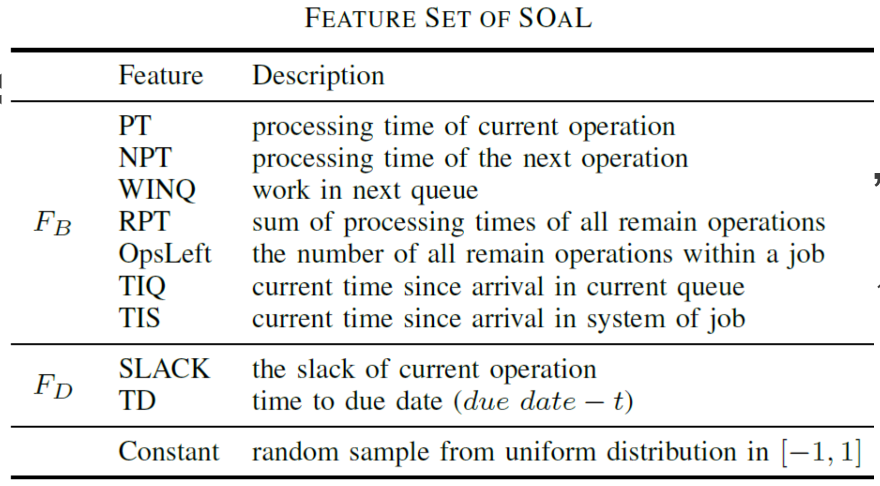
- SOaL adopts an iterative evolution procedure based on GP
  - Initialization of population of dispatching rules
  - Evaluating the performance
  - Generate new dispatching rules with genetic operators, like mutation and crossover
  - Global evaluation when GP stops
- Notations:
  - $\mathcal{P}$: the population of dispatching rules for current iteration
  - $\mathcal{S}$: a set of all evolved dispatching rules during the GP process
  - DR_i: the i-th dispatching rule
  - g_{max} and g: maximum iteration and its index



Start
Use designed features to initialize $DR_i \in P$
Apply proposed fitness function to evaluate $DR_i \in P$
Update the candidate set of top N rules based on fitness values $g = g + 1$
Apply genetic operations to generate a new population P
$g > g_{max}$?
No
Yes
Global Evaluation
Use ranking-and-selection procedure to select best dispatching rule
Stop
best DR

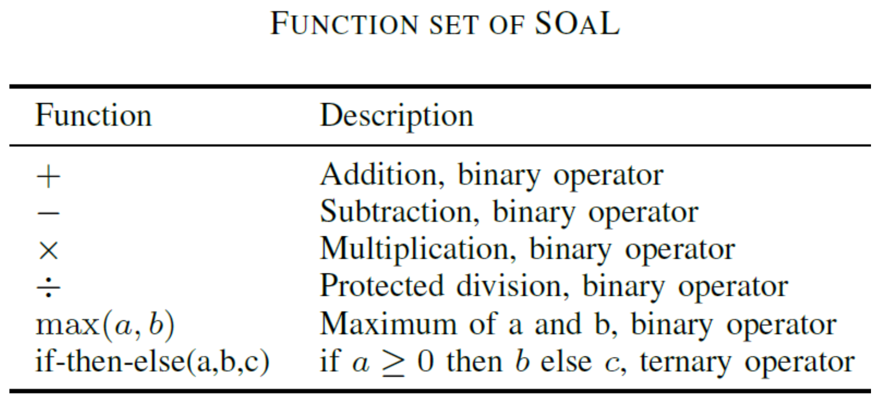
3.2 Rule Generation: Genetic Programming

- Tree-based representation of di
  - **Feature set**: processing time PT, p , work in next queue WINQ, etc.
  - **Function set**: arithmetic $(+,-,\times,\div)$ max) and mathematical functions (such as sin, cos)



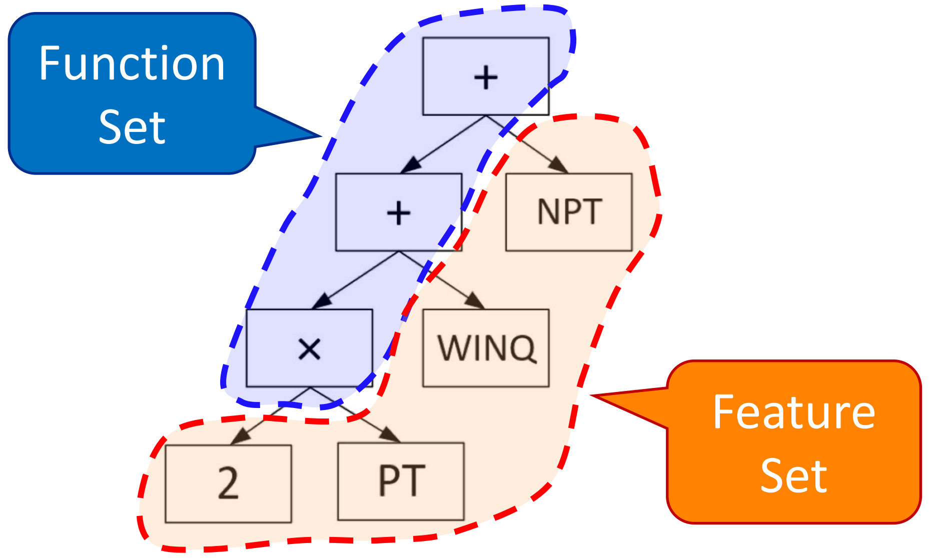
FEATURE SET OF SOAL

	Feature	Description
F_B	PT	processing time of current operation
	NPT	processing time of the next operation
	WINQ	work in next queue
	RPT	sum of processing times of all remain operations
	OpsLeft	the number of all remain operations within a job
	TIQ	current time since arrival in current queue
	TIS	current time since arrival in system of job
F_D	SLACK	the slack of current operation
	TD	time to due date ($due\ date - t$)
	Constant	random sample from uniform distribution in $[-1, 1]$

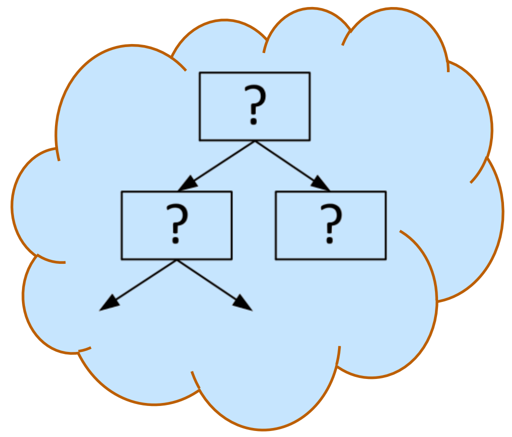


FUNCTION SET OF SOAL

Function	Description
$+$	Addition, binary operator
$-$	Subtraction, binary operator
$\times$	Multiplication, binary operator
$\div$	Protected division, binary operator
$\max(a, b)$	Maximum of a and b, binary operator
if-then-else(a,b,c)	if $a \geq 0$ then b else c, ternary operator



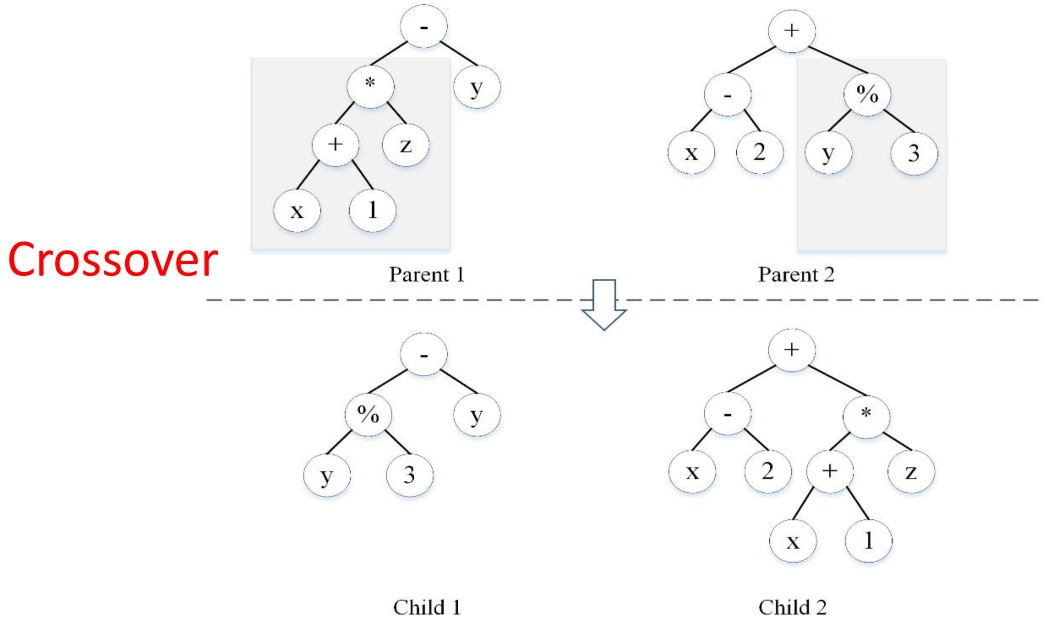
Rajendran and Holthaus Rule: $2PT + WINQ + NPT$

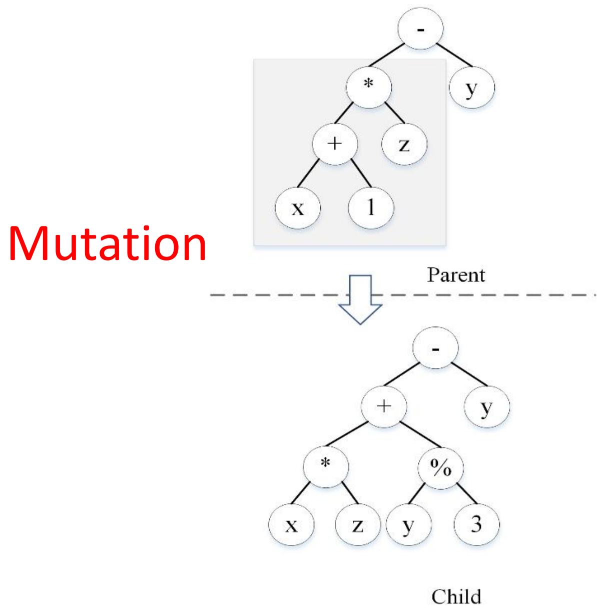


Search on the *space of dispatching rules*

3.2 Rule Generation: Genetic Programming

- GP generate new rules using **crossover** and **mutation** from Genetic Algorithm





- Fitness function

$$f(DR_i) := \frac{1}{K} \sum_{I_n \in \{I_1, \cdots, I_K\}} dev(DR_i, I_n) \quad \text{where} \quad dev(DR_i, I_n) = \frac{Obj(DR_i, I_n) - Ref(I_n)}{Ref(I_n)},$$

  – $obj(DR_i, I_n)$ is the objective value by applying DR_i to simulation I_n
  – $I_n \in \{I_1, \cdots, I_K\}$ is a set of K simulation replications assigned to evaluate DR_i

3.3 Rule Evaluation: Global Evaluation

- **Global Evaluation: Budget allocation problem**
  - $\mathcal{T}$: A total of T candidate dispatching rules that are evolved after the GP stops
  - Given N simulation budgets
  - N_i: number of budget allocated to rule i
- Goal: How to allocate N simulation budgets on T dispatching rules to accurately select the best one?

Theorem 1 (OCBA, Chen et al., (2000))

Under independent normal distribution condition, the computational budget allocation problem is asymptotically optimized as $T \to \infty$ when

1) $\frac{N_i}{N_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2$, $i, j \in \{1,2,\cdots,T\}$, and $i \neq j \neq b$,

2) $N_b = \sigma_b \sqrt{\sum_{i=1,i\neq b}^{T} \frac{N_i^2}{\sigma_i^2}}$,

where $\delta_{b,i} = \mu_b - \mu_i$ and $\mu_b \leq \min_i \mu_i$

THE UNIVERSITY OF TENNESSEE KNOXVILLE

3.3 Rule Evaluation: Ranking and Selection

Use the K simulation during GP process as initialization of OCBA

Algorithm 2 Global Evaluation

```
Input: $\mathcal{S}$: the set of all evolved dispatching rules
    $T$: the number of candidate rules for screening out
    $N$: total simulation budget for final evaluation
Output: the best dispatching rule $DR^*$
1:  Set $\mathcal{C} \leftarrow \emptyset$
2:  for $DR_i \in \mathcal{S}$ do
3:      if $f(DR_i) \notin \{f(DR_j) | DR_j \in \mathcal{C}\}$ then
4:          $\mathcal{C} \leftarrow \mathcal{C} \cup \{DR_i\}$
5:      end if
6:  end for
7:  $\mathcal{T} \leftarrow$ select the best $T$ rules $\{DR_1, DR_2, \cdots, DR_T\}$ in
    $\mathcal{C}$ by increasing order of $f(DR_i)$
8:  Set $l \leftarrow 1$; let $N^l$ be the total accumulated number of
    simulations used for all $DR_i \in \mathcal{T}$ by the $l$-th iteration
    of final evaluation, and $N_i^l$ that used for $DR_i$;
9:  Set $N^0 = K \cdot T$; set $N_i^0 = K \ \forall \ i = 1, \cdots, T$
10: Let $\mu_i$ and $\sigma_i^2$ be the mean and variance of $f(DR_i)$ by
    $N_i^0$ simulations for $DR_i \in \mathcal{T}$
11: while $\sum_{i=1}^{T} N_i^{l-1} < N_0 + N$ do
12:     $N^l \leftarrow N^{l-1} + \Delta$
13:     Get $N_1^l, \cdots, N_T^l$ by Algorithm 3 with $N^l$, $\mu_i$ and $\sigma_i^2$
14:     Evaluate $DR_i \in \mathcal{T}$ by performing additional
        $\max(0, N_i^l - N_i^{l-1})$ simulations
15:     Update $\mu_i$ and $\sigma_i^2$ for $f(DR_i)$
16:     $l \leftarrow l + 1$
17: end while
18: $DR^* \leftarrow \arg\min_{DR_i \in \mathcal{T}} f(DR_i)$
19: return $DR^*$
```

Increase a small number Δ every iteration

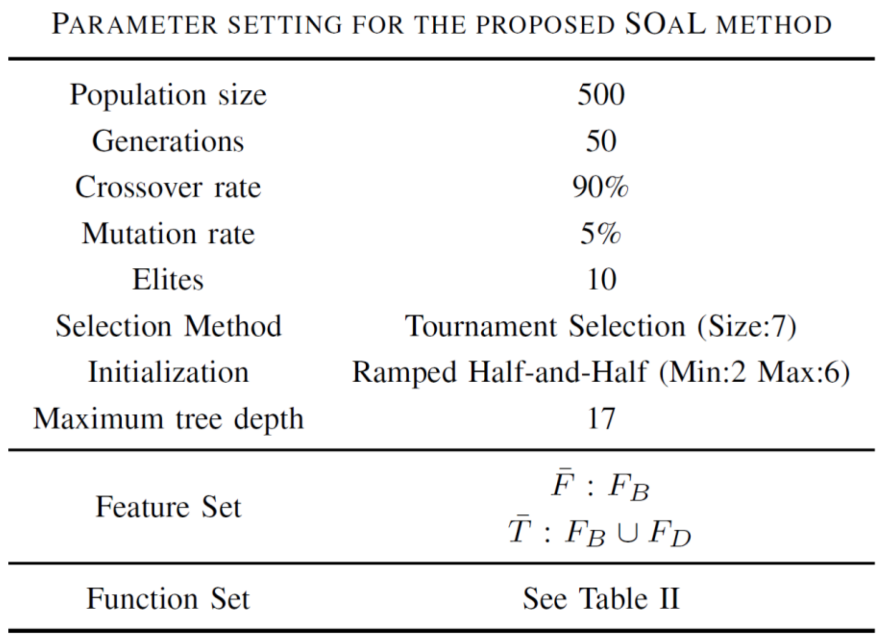
Ratios for assigning the budgets to each DR

Algorithm 3 OCBA

```
Input: $N^l$: total available simulation budget
    $\mu_i$, $\sigma_i^2$: mean and variance of $f(DR_i)$, $i = 1, \cdots, T$
Output: $N_i^l$: simulation budget allocated to $DR_i \in \mathcal{T}$
1:  Set $r_1, r_2, \cdots, r_T \leftarrow 0$
2:  Get the best mean $b \leftarrow \arg\min_i \mu_i$
3:  Get the second best mean $s \leftarrow \arg\min_{i \neq b} \mu_i$
4:  $r_s \leftarrow 1$
5:  for $i = 1$ to $T$ do
6:      if $i \neq b$ and $i \neq s$ then
7:          $r_i \leftarrow \left(\frac{\mu_b - \mu_s}{\mu_b - \mu_i}\right)^2 \frac{\sigma_i^2}{\sigma_s^2}$
8:      end if
9:  end for
10: $r_b \leftarrow \sigma_b \sqrt{\sum_{i \neq b} r_i^2 / \sigma_i^2}$
11: set $M \leftarrow \sum r_i$
12: for $i = 1$ to $T$ do
13:     Obtain the ratio of samples for each $DR_i \in \mathcal{T}$:
        $ratio_i \leftarrow r_i / M$
14: end for
15: Get $N_1^l, N_2^l, \cdots, N_T^l$ by rounding of $ratio_i \times N^l$
16: return $N_1^l, N_2^l, \cdots, N_T^l$
```

4. Computational Experiments

- Experiment environment
  - SOaL is based GP implementation of ECJ[1]. All methods are coded in java and run on an Intel Core i7-10700H 2.90 GHz CPU and 16 GB RAM
- Parameter settings for SOaL
  - GP parameters as show right; Halthaus rule is selected as reference rule
  - OCBA parameters: $K = 15$, $T = 1000$, $N = 15000$, $\Delta = 15$
- Simulation model of DJSS
  - Based on jasima[2]
  - Job routing is assigned randomly; processing time follows a discrete uniform distribution U[1,49]; job arrives follow a Poisson process
  - $m \in \{10{,}50\}$ machines;
  - $\alpha \in \{4{,}6\}$: allowance factor to set the due date $D_i = RT_i + \alpha PT_i$
  - $\mu \in \{85\%, 95\%\}$: utilization of the shop to control the mean of Poisson process
  - The first 500 jobs for warm-up; the following 2000 jobs are used for performance evaluation
- Data instances
  - Generated in simulation with common random numbers


PARAMETER SETTING FOR THE PROPOSED SOAL METHOD

Population size	500
Generations	50
Crossover rate	90%
Mutation rate	5%
Elites	10
Selection Method	Tournament Selection (Size:7)
Initialization	Ramped Half-and-Half (Min:2 Max:6)
Maximum tree depth	17
Feature Set	$\bar{F} : F_B$ $\bar{T} : F_B \cup F_D$
Function Set	See Table II

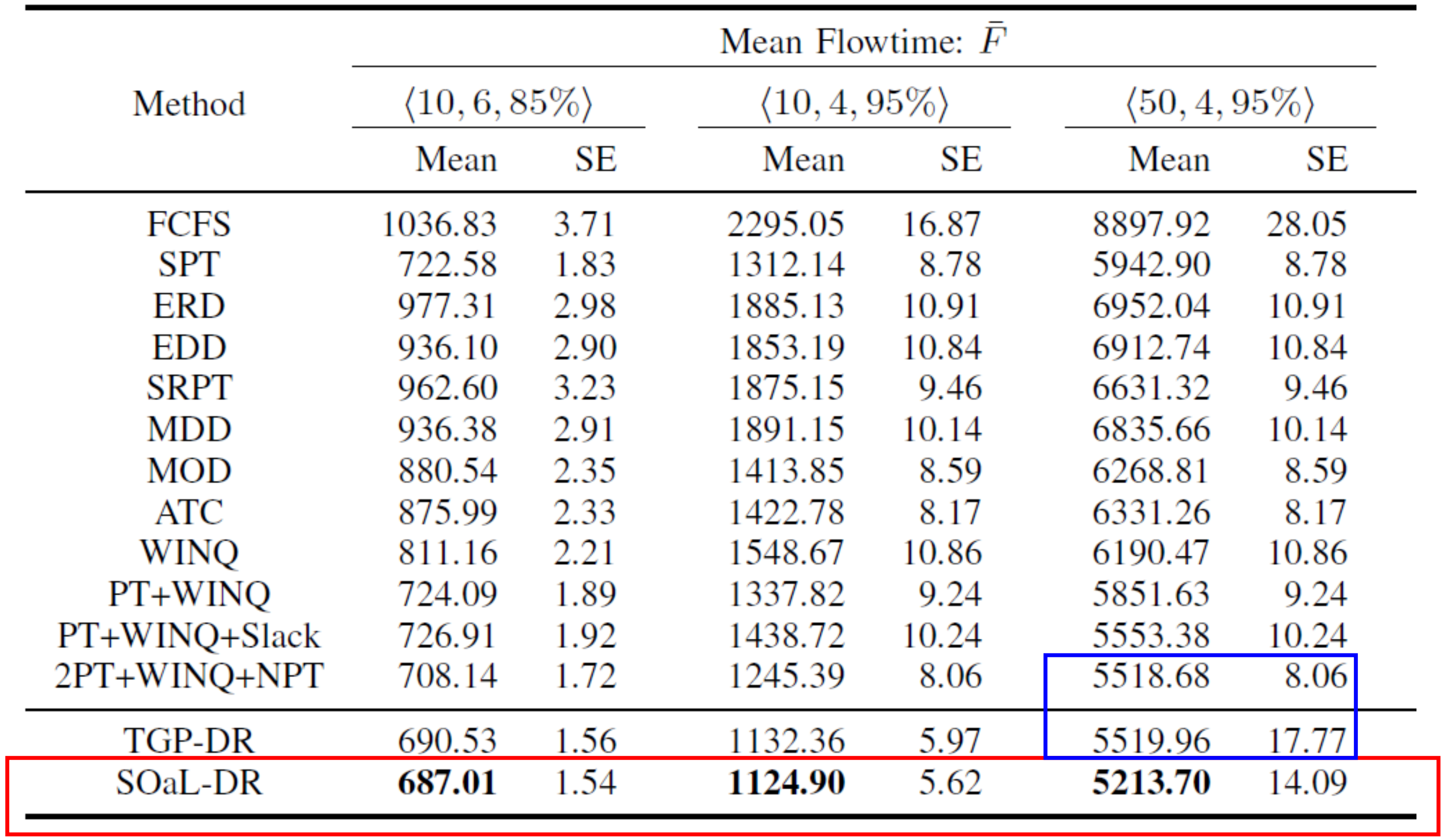
[1]. ECJ 27. A Java-based Evolutionary Computation Research System. https://cs.gmu.edu/~eclab/projects/ecj/
[2]. jasima. Java Simulator for Manufacturing and Logistics. https://jasima.net/

THE UNIVERSITY OF TENNESSEE KNOXVILLE

4. Computational Experiments

- Comparison methods
  - 12 classical benchmark dispatching rules
  - TGP-DR: learned dispatching rules using state-of-the-art GP -- TGP (Branke et al., 2015)
- Result analysis
  - SOaL-DR significantly outperforms all benchmarks, with 9.22% deduction on average
  - SOaL is generally better than the TGP-DR with statistical significance within $\pm 2SE$
- Global evaluation is effective in improving the effectiveness

$\langle m, \alpha, \mu \rangle$ =(number of machines, allowance factor, utilization of the shop)



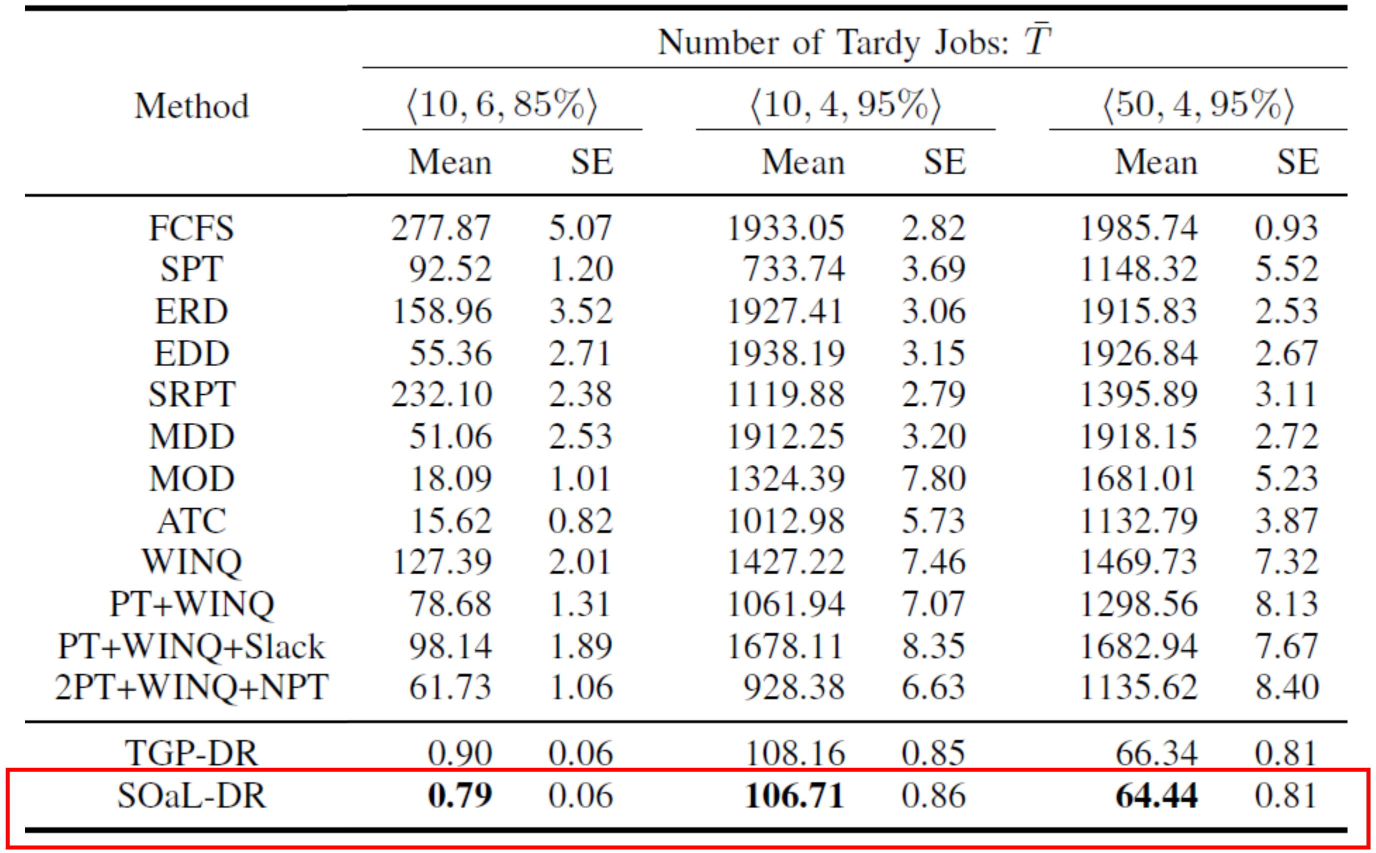
Method	Mean Flowtime: $\bar{F}$					
	$\langle 10, 6, 85\% \rangle$		$\langle 10, 4, 95\% \rangle$		$\langle 50, 4, 95\% \rangle$	
	Mean	SE	Mean	SE	Mean	SE
FCFS	1036.83	3.71	2295.05	16.87	8897.92	28.05
SPT	722.58	1.83	1312.14	8.78	5942.90	8.78
ERD	977.31	2.98	1885.13	10.91	6952.04	10.91
EDD	936.10	2.90	1853.19	10.84	6912.74	10.84
SRPT	962.60	3.23	1875.15	9.46	6631.32	9.46
MDD	936.38	2.91	1891.15	10.14	6835.66	10.14
MOD	880.54	2.35	1413.85	8.59	6268.81	8.59
ATC	875.99	2.33	1422.78	8.17	6331.26	8.17
WINQ	811.16	2.21	1548.67	10.86	6190.47	10.86
PT+WINQ	724.09	1.89	1337.82	9.24	5851.63	9.24
PT+WINQ+Slack	726.91	1.92	1438.72	10.24	5553.38	10.24
2PT+WINQ+NPT	708.14	1.72	1245.39	8.06	5518.68	8.06
TGP-DR	690.53	1.56	1132.36	5.97	5519.96	17.77
SOaL-DR	**687.01**	1.54	**1124.90**	5.62	**5213.70**	14.09

4. Computational Experiments

- Result analysis
  - Similar conclusions can hold
  - Significantly outperforms than benchmarks with 89.48% deduction on average
  - Better than TGP within $\pm 2SE$

- Computation time
  - Benchmarks: < 0.1 sec
  - SOaL
    - Training: 3 hrs
    - Testing (SOaL-DR): 0.1~0.5 sec

- Effective and real-time optimization dispatching rules!



Method	Number of Tardy Jobs: $\bar{T}$					
	$\langle 10, 6, 85\% \rangle$		$\langle 10, 4, 95\% \rangle$		$\langle 50, 4, 95\% \rangle$	
	Mean	SE	Mean	SE	Mean	SE
FCFS	277.87	5.07	1933.05	2.82	1985.74	0.93
SPT	92.52	1.20	733.74	3.69	1148.32	5.52
ERD	158.96	3.52	1927.41	3.06	1915.83	2.53
EDD	55.36	2.71	1938.19	3.15	1926.84	2.67
SRPT	232.10	2.38	1119.88	2.79	1395.89	3.11
MDD	51.06	2.53	1912.25	3.20	1918.15	2.72
MOD	18.09	1.01	1324.39	7.80	1681.01	5.23
ATC	15.62	0.82	1012.98	5.73	1132.79	3.87
WINQ	127.39	2.01	1427.22	7.46	1469.73	7.32
PT+WINQ	78.68	1.31	1061.94	7.07	1298.56	8.13
PT+WINQ+Slack	98.14	1.89	1678.11	8.35	1682.94	7.67
2PT+WINQ+NPT	61.73	1.06	928.38	6.63	1135.62	8.40
TGP-DR	0.90	0.06	108.16	0.85	66.34	0.81
SOaL-DR	**0.79**	0.06	**106.71**	0.86	**64.44**	0.81

5. Conclusions and Future Work

- Conclusions
  - A novel and effective simulation optimization-aided machine learning (SOaL) method for design automation of scheduling rules
  - Significantly outperforms existing benchmark dispatching rules with 9.22% deduction of mean flowtime, and 89.48% deduction of mean number of tardy jobs on average. Outperforms state-of-the-art TGP with statistical significance.

- Future work
  - Theoretical and algorithmic foundations for design automation of scheduling rules
  - Applying for complex production scheduling problems in practice