
경영과학연구실 2023 여름방 학 워크샵

발표자 : 김변민

- 1. A Reinforcement Learning Approach to Robust Scheduling of Semiconductor Manufacturing Facilities**
- 2. Dynamic Job shop Scheduling Algorithm Based on Deep Q Network**
- 3. Reinforcement learning for an intelligent and autonomous production control of complex job-shops under time constraints**

Park, In-Beom, et al. "A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities." *IEEE Transactions on Automation Science and Engineering* 17.3 (2019): 1420-1431.

Introduction

Objective of this paper

- To minimize the makespan for an Job shop scheduling problem in semiconductor FAB

Factors that complicate the operation of a semiconductor FAB

- Reentrant production flows
- Sequence-dependent setups
- Alternative machines

Traditional methods to solve job shop scheduling

- Rule based method
- Metaheuristic method
- Machine learning

Problem statement

***Deciding which job to assign when a machine becomes idle in the dieattach*
and wire bonding stages of a semiconductor packaging line.**

Algorithm

- Multi agent DQN

Objective

- Minimize makespan

Network structure

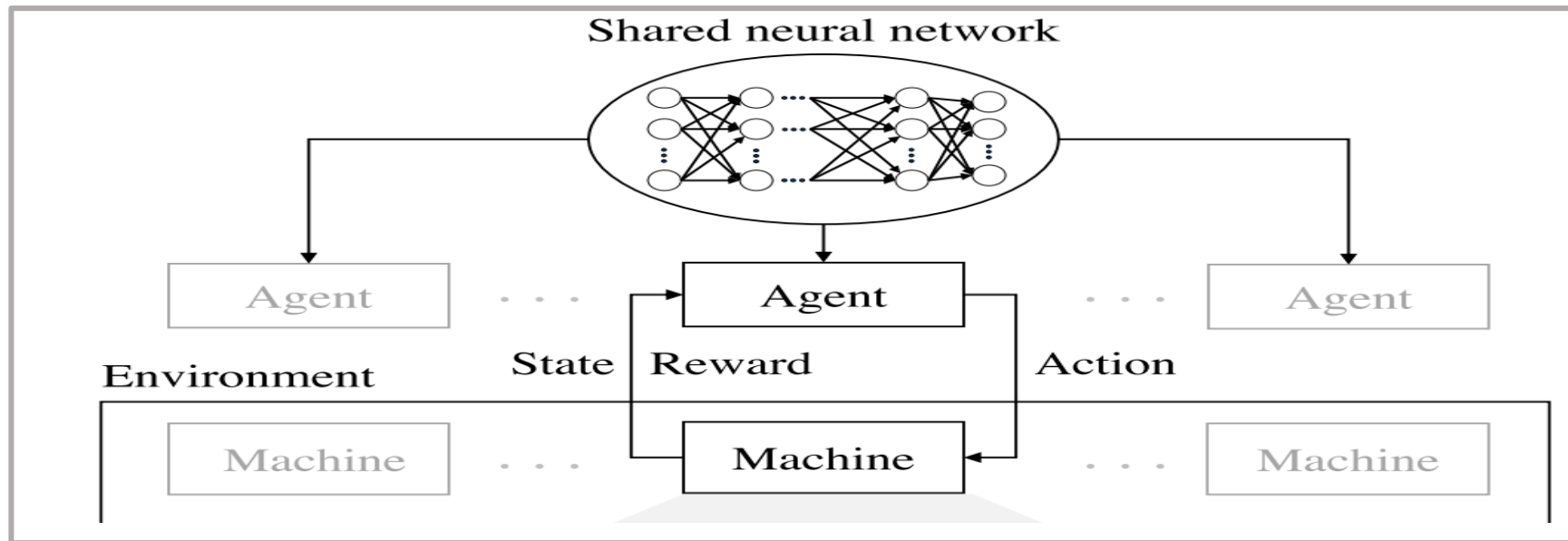
- ANN

Issue that this paper aims to address

- Variabilities in production requirements, the number of available machines, and initial setup status that make it challenging to create high quality schedule

Key idea

1. Each agent determines actions in a decentralized manner and learns a centralized policy by sharing a neural network among the agents to deal with the changes in the number of machines



2. State, action, and reward are proposed to address the variabilities in production requirements and initial setup status

State, Action

State

Features	Descriptions	Dimension
Waiting operations	The number of waiting operations of $O_{j,k}$ which can be processed by the machine	N_O
Setup status	Setup type of the machine represented as one-hot encoding	N_O
Action history	The number of performed actions on the machine	$N_O + 1$
Utilization history	The amounts of processing, setup, and idle time of the machine	3

N_o : Number of operation types

O_{jk} : Kth operation of job type j

Action

when machine becomes idle



If number of $O_{jk} > 0$

Choose O_{jk}

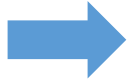

else

δ_0 (dummy action)

Each agent corresponds to each machine

Reward

$$r_i = \begin{cases} -(\tau(s_{i+1}) - \tau(s_i) - p_{j,k}), & a_i = O_{j,k} \\ -(\tau(s_{i+1}) - \tau(s_i)), & a_i = \delta_0. \end{cases}$$

 Reward = setup time of machine
 Reward = idle time of machine

State transition occurs when operation finished or at least one operation occurs

Parameters & variables

N_o : Number of operation types

O_{jk} : Kth operation of job type j

δ_0 : dummy action (when no O_{jk} to choose)

P_{jk} : processing time of Kth operation of job type j

$\tau(s_{i+1})$: time of next state

If number of $O_{jk} > 0$

$\tau(s_{i+1}) =$ when operation finished

else

$\tau(s_{i+1}) =$ Earliest time when there is at least one operation

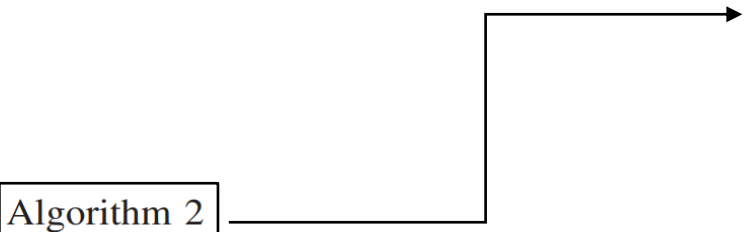
Multi agent DQN training

Algorithm 1 Scheduling With Q -Network

Input: Scheduling problem

Output: Q -network

```
1: Initialization: Set network  $Q$  with random weight  $\theta$ , target
   network  $\hat{Q}$  with  $\hat{\theta} = \theta$ , and replay buffer  $B$  to size  $N_B$ .
2: for  $e = 1, 2, \dots, N_E$  do
3:   Initialize jobs and  $N_M$  machines
4:    $t \leftarrow 0$ 
5:   for  $l = 1, 2, \dots, N_M$  do
6:     Observe  $s_1$  as the first state of  $M_l$ 
7:     Execute action  $a_1$  according to  $\varepsilon$ -greedy policy
8:      $H(M_l) \leftarrow (s_1, a_1)$ 
9:   end for
10:  while  $t < C_{\max}$  do
11:    Build  $E_t$ 
12:    repeat
13:      Get  $s, a,$  and  $M_l$  using Algorithm 2
14:       $(s_i, a_i) \leftarrow H(M_l)$ 
15:      Observe  $r_i$  and set  $(s_{i+1}, a_{i+1}) = (s, a)$ 
16:       $H(M_l) \leftarrow (s_{i+1}, a_{i+1})$ 
17:      Store transition  $(s_i, a_i, r_i, s_{i+1})$  in  $B$ 
18:      Sample  $N_{TR}$  transitions  $(s_u, a_u, r_u, s_{u+1}) \in B$ 
19:      Set  $q_u = Q(s_u, a_u; \theta)$ 
20:      Set  $y_u = r_u + \gamma \mathbb{1}_F(s_{u+1}) \max_{a'} \hat{Q}(s_{u+1}, a'; \hat{\theta})$ 
```



Algorithm 2 Machine Selection With ε -Greedy Policy

Input: E_t

Output: State s , action a , and M_l on which a is executed

```
1: Sample a random number  $X \in [0, 1]$ 
2: if  $X < \varepsilon$  then
3:   Select a machine  $M_l$  randomly from  $E_t$ 
4:   Observe  $s$  of  $M_l$ 
5:   Select  $a$  randomly from  $A(s)$ 
6: else
7:   for  $M_l \in E_t$  do
8:     Observe  $s^l$  of  $M_l$ 
9:      $(q^l, a^l) = \left( \max_{a \in A(s^l)} Q(s^l, a; \theta), \operatorname{argmax}_{a \in A(s^l)} Q(s^l, a; \theta) \right)$ 
10:  end for
11:  Select  $l = \operatorname{argmax}_l q^l$ ,  $s = s^l$ , and  $a = a^l$ 
12: end if
13: return  $s, a,$  and  $M_l$ 
```

Experiment Result

N_m : Number of Machines

N_j : Number of Job types

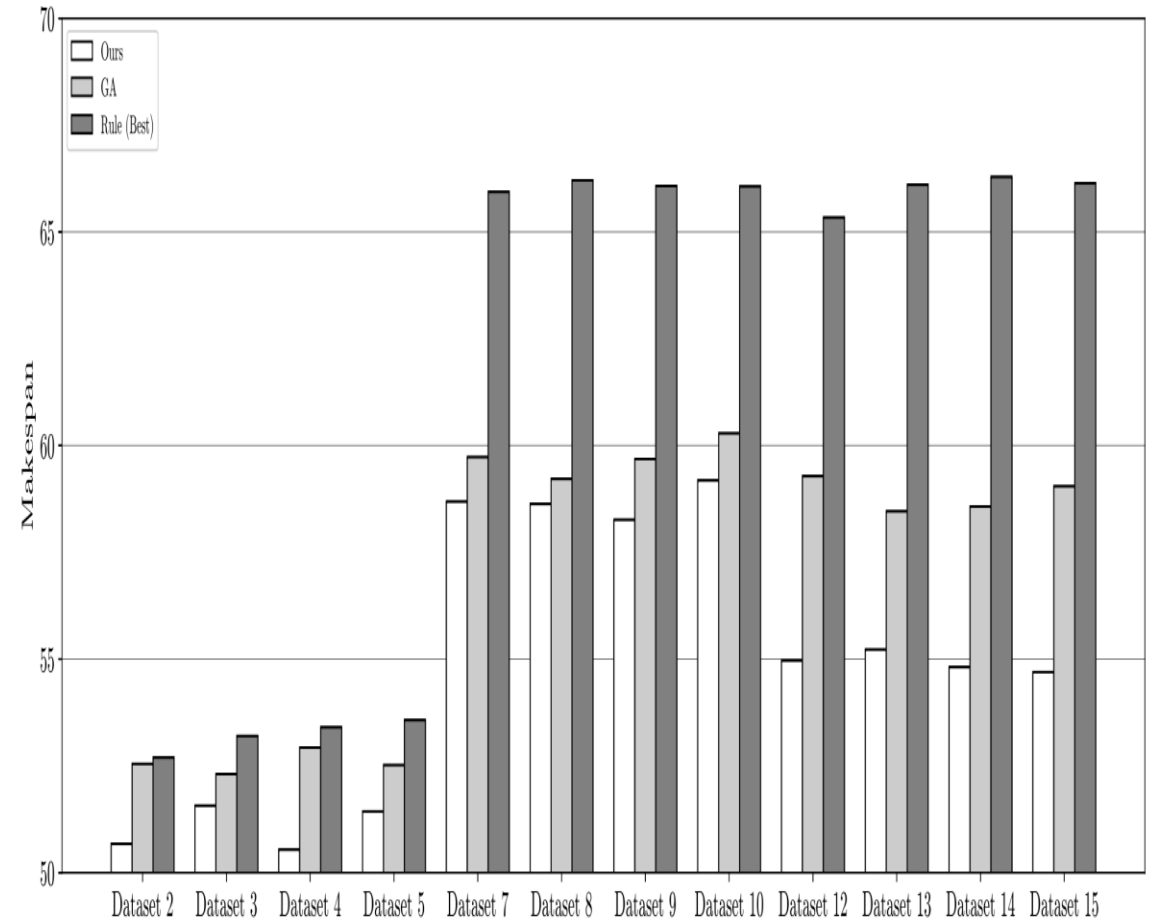
N_o : Number of operation types

Episode : average 4000

Dataset

Dataset No.	N_M	N_J	N_O	The average number of operations
1	35	7	28	610
2	70	7	28	1220
3	105	7	28	1830
4	140	7	28	2440
5	175	7	28	3050
6	35	10	47	620
7	70	10	47	1240
8	105	10	47	1860
9	140	10	47	2480
10	175	10	47	3100
11	35	12	64	590
12	70	12	64	1180
13	105	12	64	1770
14	140	12	64	2360
15	175	12	64	2950

Result



Zhao, Yejian, et al. "Dynamic jobshop scheduling algorithm based on deep q network." *Ieee Access* 9 (2021): 122995-123011.

Problem statement

***Deciding which job to assign when a machine becomes idle in FAB using*
reinforcement learning**

Algorithm

- Single agent DQN

Objective

- Minimize tardiness of Jobs

Network structure

- DNN

Issue that this paper aims to address

- Inefficient MDP modeling in previous studies using reinforcement learning to solve the job shop scheduling problem

Key idea

- 1. State variable representation method is proposed which describes the various production states of the system continuously and comprehensively.**
- 2. Reward function is defined that reflects the current scheduling state in detail**
- 3. An action selection strategy based on the “softmax” function is proposed, which has higher randomness of action selection than the traditional “greedy strategy**

State

Variables	State Variable Description
$U_{ave}(t)$	Average utilization of all machines at time t
$D_{std}(t)$	Standard deviation of utilization of all machines at time t
$P(t)$	Work load rate of all machines at time t
$CR_{ave}(t)$	Average completion rate of all jobs at time t
$RPWR_{ave}(t)$	Average remaining process waiting rate of all jobs at time t



$$U_{ave}(t) = \sum_{k=1}^m U_k(t)$$

$$D_{std}(t) = \sqrt{\frac{\sum_{k=1}^m (U_k(t) - U_{ave}(t))^2}{m}}$$

$$P(t) = \frac{EART(t)}{EAST(t) + \mu}$$

$$CR_{ave}(t) = \frac{\sum_{i=1}^n NPM_i(t)}{\sum_{i=1}^n n_i}$$

$$RPWR_{ave}(t) = \frac{\sum_{i=1}^n (n_i - NPM_i(t))}{\sum_{i=1}^n n_i}$$

Parameters & variables

$U_k(t)$ = Utilization of machine k

$EAST(t)$ = Estimated average slack time of all jobs in time t

$EART(t)$ = Estimated average remaining processing time ""

$NPM_i(t)$ = Number of passing machines of job i in time t

$PCR_i(t)$ = Processing completion rate of job i in time t

$CTFP_i(t)$ = Completion time of the final process ""

PT_{ik} = Processing time for job i in machine k

Action

Action

Apply on of ten dispatching rules to the equipment

Action selection Mechanism

$$P(\varphi_t, a_i) = \frac{\exp(\mu Q(\varphi_t, a_i, \theta))}{\sum_{a \in A} \exp(\mu Q(\varphi_t, a, \theta))}$$



Applying the Softmax function to select action instead of the Epsilon-greedy method

State transition occurs when operation finished or at least one operation occurs

10 dispatching rules

- Longest remaining processing time rule
- Least remaining processing time rule
- Shortest processing time rule
- Longest processing time rule
- Least number of remaining processes rule
- Most remaining processes rule
- The smallest slack rule
- Smallest ratio of slack per work remaining rule
- Smallest critical ratio rule
- Random

Reward

Reward function

$$R = \frac{EAST(t)}{EART(t) + 0.1}$$

$EAST(t)$ = Estimated average slack time of all jobs in time t

$EART(t)$ = Estimated average remaining processing time of all jobs in time t

Indicator of minimal delay time

As the execution time of the scheduling system increases, the reward function is designed in a way that $EAST(t)$ continually increases and $EART(t)$ decreases



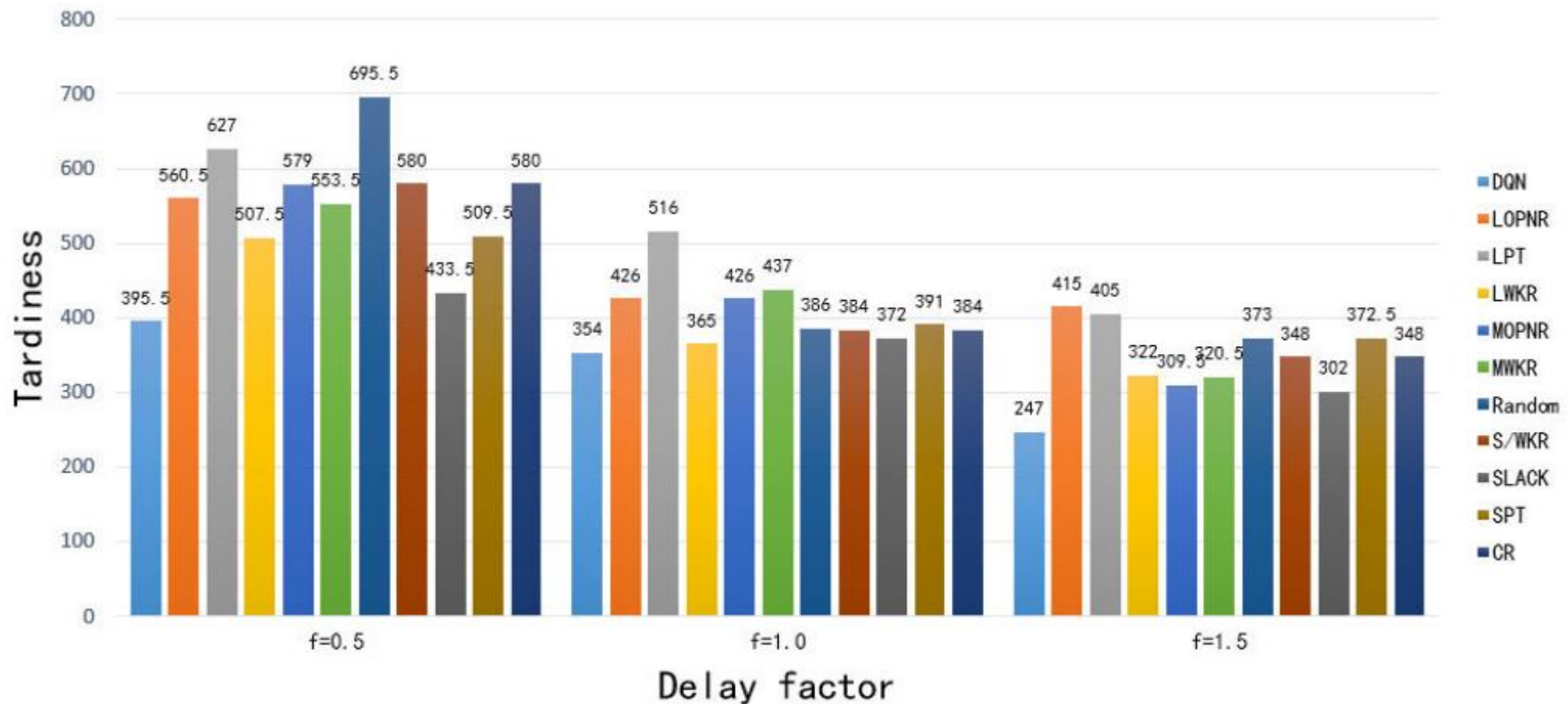
This way of defining the reward function can more allows the scheduling system to better reflect its dynamic characteristics.

Experiments result

Delay factor : smaller f value indicates higher priority of the job to be processed

Number of job : 10X

Number of Machine : 5



**Altenmüller, Thomas, et al.
"Reinforcement learning for an
intelligent and autonomous
production control of complex job-
shops under time
constraints." *Production
Engineering* 14 (2020): 319-328.**

Problem statement

***Deciding which job to assign when a machine becomes idle in FAB using*
reinforcement learning with time constraint situation**

Algorithm

- Single agent DQN

Objective

- Minimize number of jobs violating time constraints

Network structure

- ANN

Issue that this paper aims to address

- Challenges of scheduling considering time constraints

Key idea

- 1. Present the first MDP modeling that considers time constraints with such characteristics, where, besides customer due dates, time constraints can also exist between two or more consecutive process steps**

State

State

- Current machine where the action is asked for
- Loading status of all machines (binary, idle or busy)
- Product setup per machine
- Product variant in the current machine's buffer slots
- Order status per buffer slot (real-value)
- Full or empty status per buffer slot

$$UR := \max \left\{ \frac{T_C - (t - t_{\text{finished last step}})}{T_C}, -1 \right\}$$

Action

- Selecting an order from a buffer slot (20 actions)
- Idle and choose no order (1 action)

***Randomly shuffling the location information of the orders prevents the agent*
from learning a biased policy that is oriented towards picking specific slots.**

Reward

Local reward

$$r_{\text{local}} = 2 \cdot 5^{(UR+1)/2} \quad \longrightarrow \quad \text{reward is higher for less urgent tasks for each machine}$$

Global reward

$$r_{\text{global}} = 10 \cdot \left\langle \exp \left(-\frac{V_p}{3} \right) \right\rangle_p \quad \longrightarrow \quad \text{Higher reward for fewer time constraint violations in recent 50 order of a specific product}$$

WIP reward

$$r_{\text{WIP}} = \prod_{i \in \text{buffers}} 1 + \max \{ \beta \cdot (\text{WIP}_i - \text{WIP}_i^{\text{target}}), 0 \} \quad \longrightarrow \quad \text{Higher reward for closer adherence to target WIP level}$$

Total reward

$$r_{\text{local}} * r_{\text{wip}} \quad \longrightarrow \quad r_{\text{global}} * r_{\text{wip}} \quad \longrightarrow \quad \text{Total reward transits over the course of the exploration phase from } r_{\text{local}} \text{ to } r_{\text{global}} \text{ to smoothly aim at global learning}$$

Experiments Result

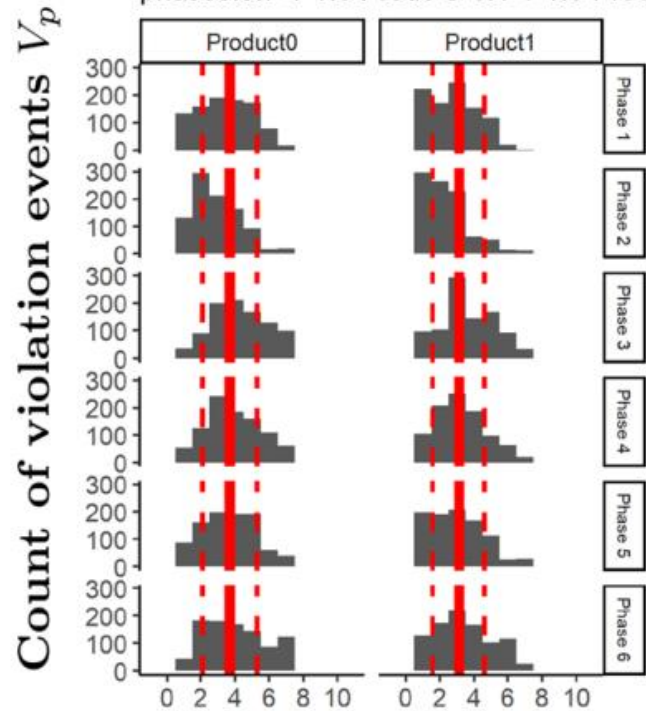
Product type : product type1, product type 2

Number of machines : 10

Number of Jobs : 10

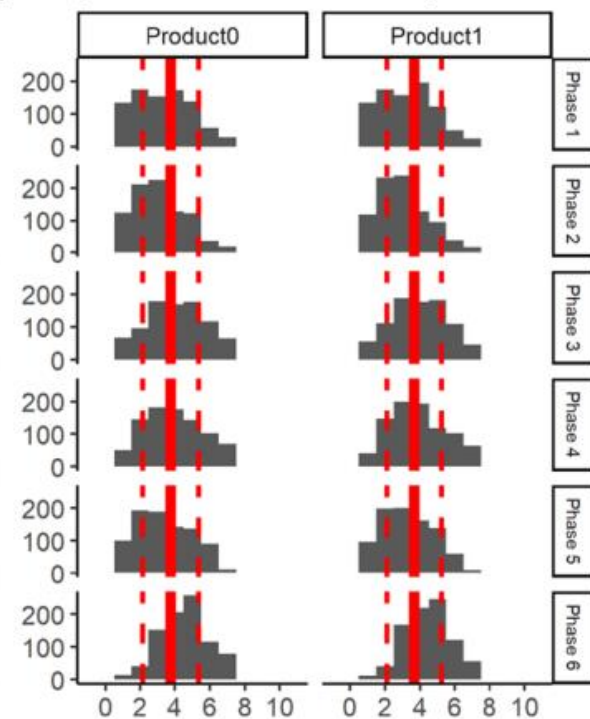
FIFO

Distribution mean +/- std dev over all phases: 3.7 +/- 1.6 Prod0 & 3.1 +/- 1.5 Prod1



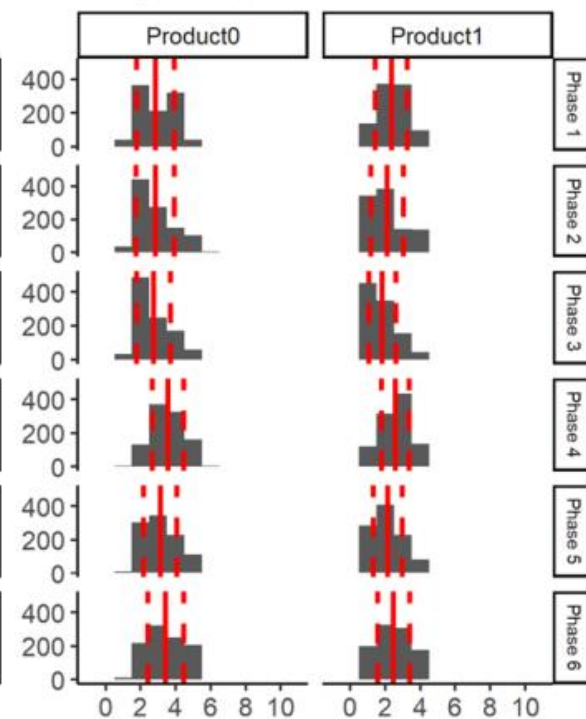
TC

Distribution mean +/- std dev over all phases: 3.7 +/- 1.6 for both products



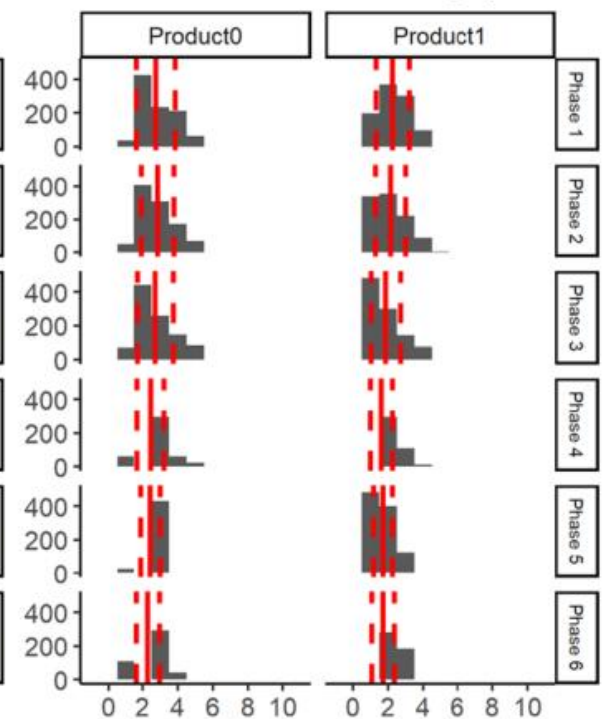
RL local

Distribution mean & std dev do not change over phases, width smaller than heuristics



RL global

Distribution means and std dev further reduced while learning by ~20%



Number of time constraint violations V_p

Thank you
