FACULTY OF
ECONOMICS AND MANAGEMENT

Machine learning approaches for Scheduling problem

Master Thesis Presentation

Name: Nga Nguyen – 221188

Email: nga1.nguyen@st.ovgu.de

Program: Operations Research and Business Analytics

Supervisor: Dr. Janis Sebastian Neufeld

AGENDA

01 Introduction to Scheduling problems and Machine Learning

02 Permutation Flow-shop Scheduling problem

03 Solution approach: Q-Learning

04 Experimental results

05 Conclusion and further research

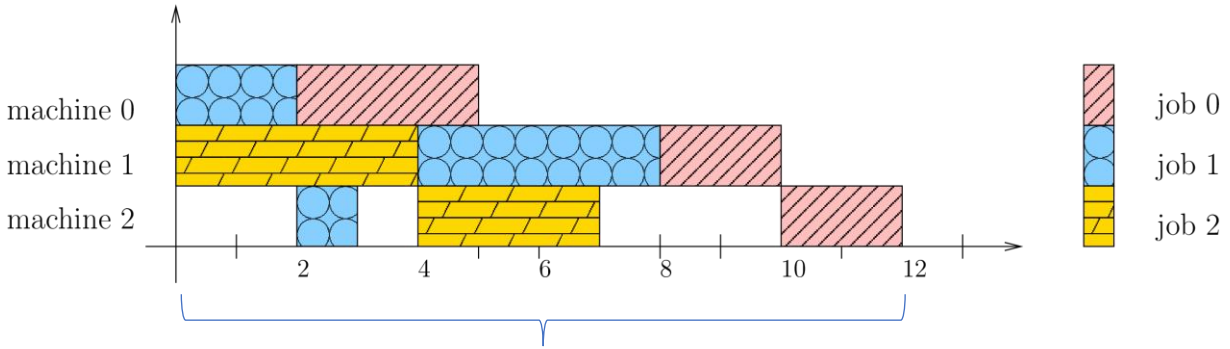
1. Introduction

Scheduling Problems

- **Scheduling** is a decision-making process that is used on a regular basis in many manufacturing and service industries.
- It deals with the allocation of jobs(i.e. the tasks in which each job is divided) to resources over given time periods. The goal is to optimize one or more objectives.
- In manufacturing process:
  - Job: the item you wish to manufacture
  - Resources: Machines
- Minimize production time and costs, by deciding when to make, with which staff, on which equipment.

Time required for each job in each machine

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3



Makespan = 12

Source: Google OR tools

1. Introduction

Different types of scheduling

Flow shop Scheduling Problem (FSSP)

- All jobs have the same processing order through the machines.
- The order of the jobs in each machine can be the same(permutation FSSP) or not necessarily the same(non-permutation FSSP)

Job shop Scheduling(JSSP)

- Each job has its own routing

Open shop Scheduling (OSSP)

- A given set of jobs is processed in an arbitrary order

1. Introduction

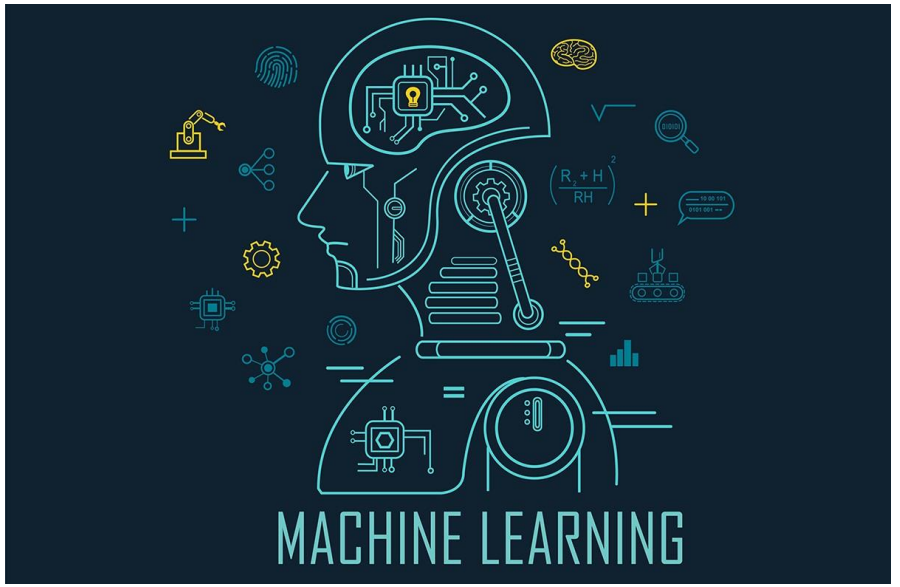
Scheduling Objectives

Flowtime:	Makespan:	Lateness:	Earliness-tardiness:
• The sum of all jobs' finishing times.	• The maximum finishing time, which is the time when the last job is completed.	• The difference between the due date and the finishing time. When this value is positive, it refers to tardiness, and when negative, the measure is earliness.	• Sum of earliness and tardiness.

1. Introduction

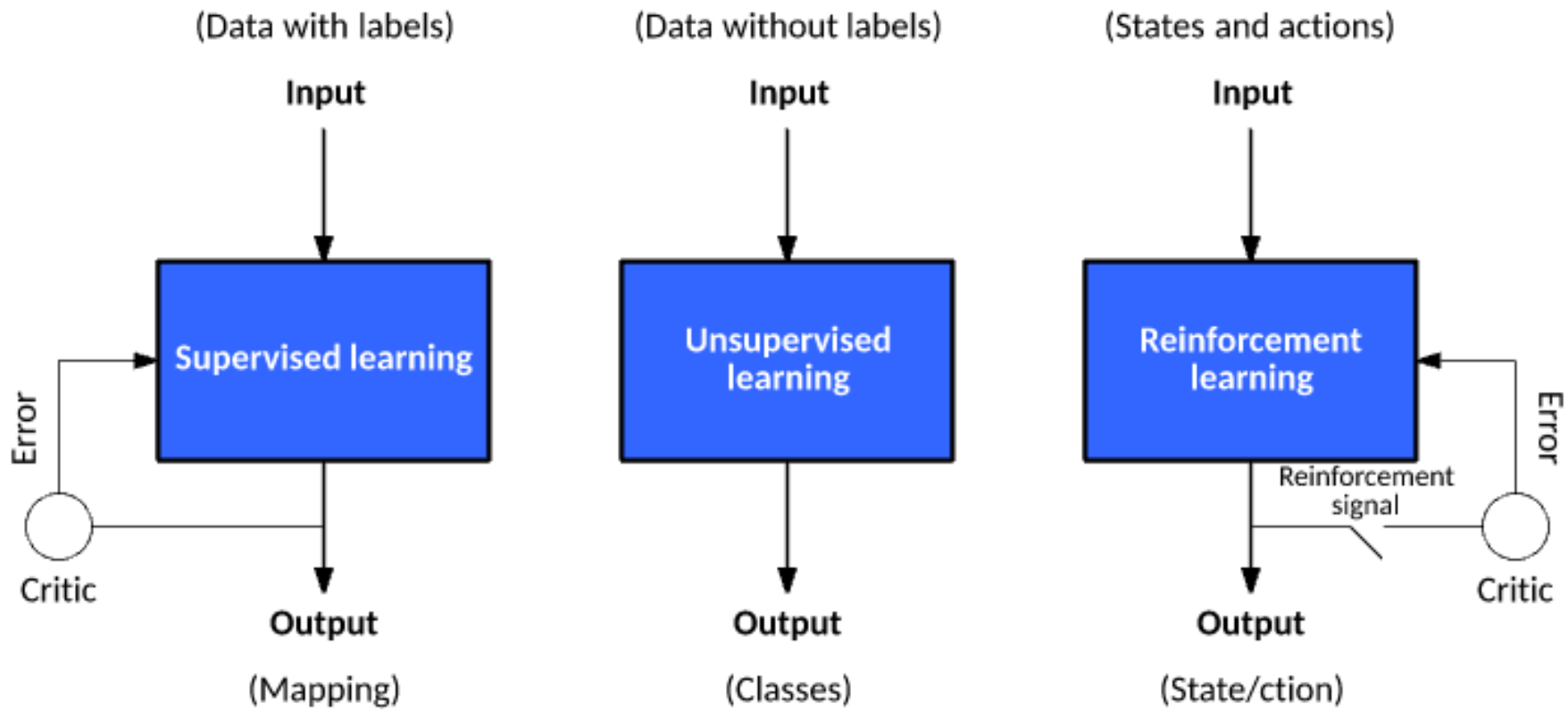
Machine learning

- "**Machine Learning** is the field of scientific study that concentrates on induction algorithms and on other algorithms that can be said to "learn."" - Ron Kohavi (1998)
- **Machine learning** algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without explicitly programmed to do so - Arthur Samuel(1959)
- **Machine learning** is considered as an optimization.
- Application: automatic recommendation, personalized advertisements, object detection in videos and pictures, etc.



1. Introduction

Types of Machine learning



1. Introduction

Literature review

- The earliest work to be known used **expert systems** to solve scheduling problems (Kanet and Adelsberger, 1987). They proposed a reformulative approach, which focused on proactively restating the problem until a satisfactory solution is easy to be found
- Haldun Aytug and Siddhartha (1994) presented a review of Machine Learning in Scheduling

Supervised learning	Reinforcement learning
Training examples must be generated from scheduling data. Then these examples are used to build a machine learning model.	Construct the algorithm directly from scheduling data
• Shahzad and Mebarki (2012) generated training examples with an optimization module which solves instances base on tabu search • Frank Benda [2019] generated training samples including different feasible solutions using constraint programming and mixed-integer programming	• Gabel and Riedmiller (2007) use Reinforcement Learning for scheduling problems, more specifically job shop scheduling. • Reyna, Jiménez, Cabrera and Hernández (2015,2017) followed the idea to solve JSSP and FSSP • Reyna, Jiménez, Cabrera and Hernández (2019) proposed a new combination of RL, more specifically Q-Learning and a local search in order to give better solution.

2. Problem description

Permutation flow shop Scheduling (P-FSSP)

- A set of ***m*** machines and a set of ***n*** jobs
- All jobs have the same processing order through machines
- Each machine processes the jobs in the same order
- Operations cannot be interrupted and each machine can process only one operation at a time.

=> find the job sequences on the machines that minimize the Makespan, which is the maximum completion time of all the operations.

- n! feasible sequences

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

2. Problem description

Permutation flow shop Scheduling (P-FSSP)

Input:

- A set of ***m*** machines and a set of ***n*** jobs
- p(i,j) is the processing time of job i on machine j

Need to find:

- A job permutation $\{J_1, J_2,\ldots,J_n\}$

Objective:

- Minimize $C(J_i, j)$, the completion time of J_i on machine j is defined as follow:

$$C(J_1, 1) = p(J_1,1)$$
$$C(J_i, 1) = C(J_{i-1}, 1) + p(J_i,1) \text{ for } i = 2,\ldots,n$$
$$C(J_i, 1) = C(J_1, j-1) + p(J_1,j) \text{ for } j = 2,\ldots,m$$
$$C(J_i, j) = \max \{C(J_{i-1}, j), C(J_i, j-1) + p(J_i, j)\}$$
$$\text{for } i = 2,\ldots,n \text{ ; for } j = 2,\ldots,m$$
$$C_{max} = C(J_n,m) \Rightarrow \text{Minimize!}$$

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

Feasible Sequence	Makespan
0-1-2	16
0-2-1	15
Etc.	…

3. Q –Learning

Definition

- Q-Learning is a type of reinforcement learning
- Q-Learning is a model-free algorithm
- Q-Learning learns in an interactive environment by trial and error feedback

An agent

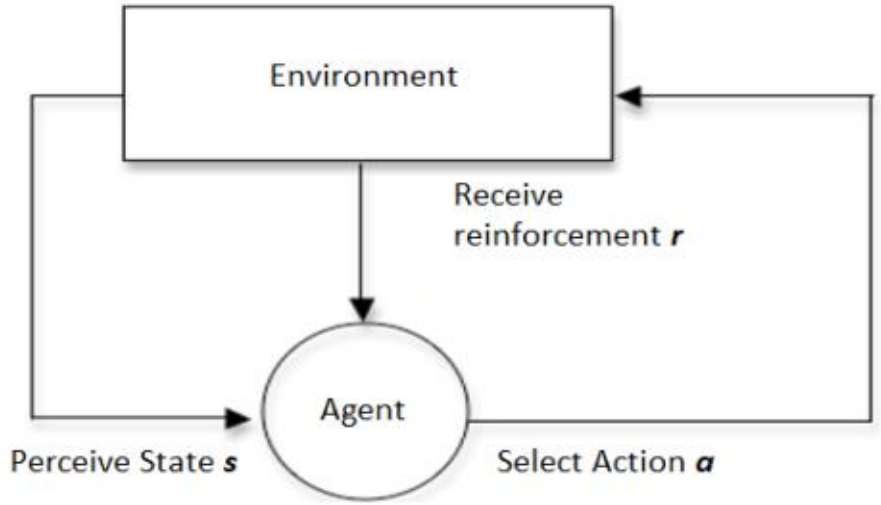
- has a goal
- is able to observe/monitor states
- is able to perform actions that change its current state

Q-value

- indicates the quality of a particular action *a* in a given state *s* : Q(*s*,*a*)
- is current estimate of sum of future rewards

Q-table

- Store Q-values
- Has one row for each possible state, and one column for each possible action



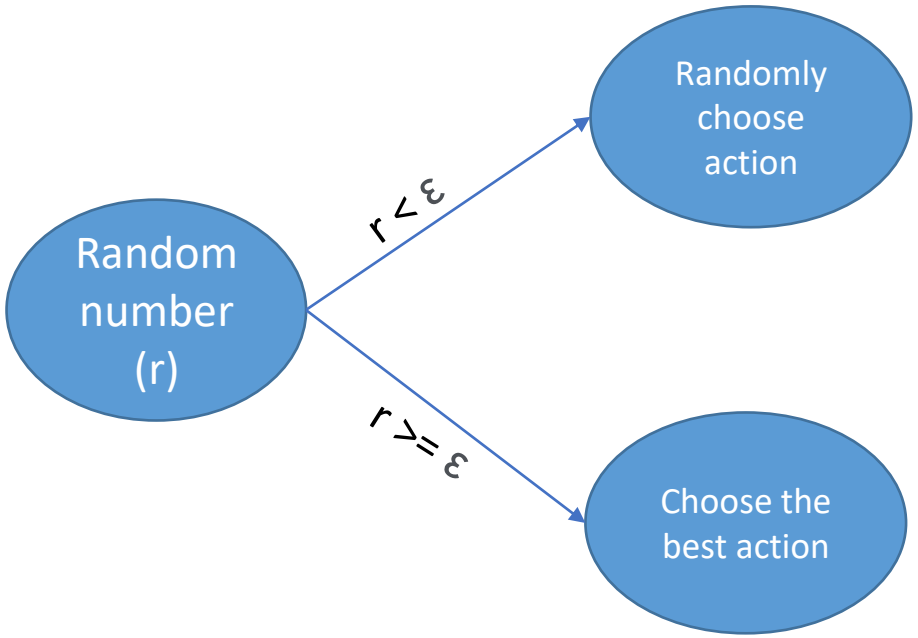
3. Q –Learning

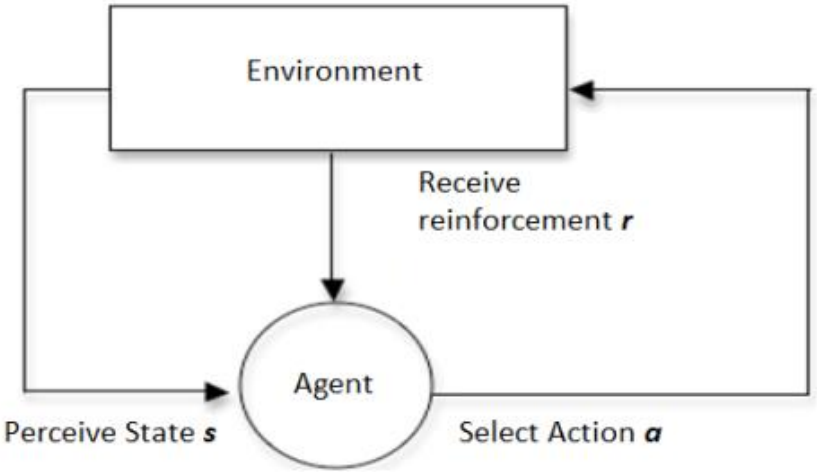
Definition

- Actions will be chosen using an **ε-greedy algorithm**. This algorithm balances between exploration and exploitation. ε is set at the beginning.
- When action **a** is selected by the agent located in state ***s***, the Q-value for that state-action pair is updated based on the reward received when selecting that action and the best Q-value for the subsequent state

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a))]$$

- $\alpha \in [0, 1]$ is the learning rate
- γ discount factor
- ***r*** the reward resulting from taking action **α**
- $Max_{a'}Q(s',a')$ is the largest Q value for any action in the current state





3. Q –Learning for P-FSSP

Define Environment

- States:
  - Jobs left to be processed => 2^n states, n is #jobs
- Actions:
  - Take one job to process => n actions
- Rewards to be maximized:
  - r = $\frac{1}{Makespan}$
- ε = 0.2 # greedy policy
- α = 0.4 # learning rate
- γ = 0.8 # discount factor

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

States \ Actions	Take Job 0	Take Job 1	Take Job 2
0,1,2	0	0	0
0,1	0	0	
0,2	0		0
1,2		0	0
0	0		
1		0	
2			0
Ø			

3. Q –Learning for P-FSSP

Pseudo-code

Initialize :

Q(s,a) ={}

Best = {}

for each episode step **do**

Initialize s = {0,…,n}

while not_finished(all jobs)

Choose a from s using policy derived from Q(e.g., ε-greedy)

Take action a, observe state s' and r as 1/**makespan**(s’)

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a))]$$

$s \leftarrow s'$

end while

if makespan(s) < makespan(Best):

$Best \leftarrow s$

end for

3. Q –Learning for P-FSSP

Learning

$\varepsilon = 0.2$
$\alpha = 0.4$
$\gamma = 0.8$

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

1st episode:

- Current state: [0,1,2]
- Choose a random action: [1]
- Sequence: [1]
- Reward r = $\frac{1}{Makespan(1)} = \frac{1}{2+4+4} = 0.1$
- New Q([0,1,2],[1])

= Q([0,1,2],[1]) + α (r + γ*max(Q([0,2],0),Q([0,2],2)) - Q([0,1,2],[1]))

= 0 +0.4(0.1 + 0.8* 0 - 0)

= 0.04

States \ Actions	Take Job 0	Take Job 1	Take Job 2
0,1,2	0	0	0
0,1	0	0	
0,2	0		0
1,2		0	0
0	0		
1		0	
2			0
∅			

3. Q –Learning for P-FSSP

Learning

$\varepsilon = 0.2$
$\alpha = 0.4$
$\gamma = 0.8$

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

1st episode:

- Current state: [0,1,2]
- Choose a random action: [1]
- Sequence: [1]
- Reward r = $\frac{1}{Makespan(1)} = \frac{1}{2+4+4} = 0.1$
- New Q([0,1,2],[1])

= Q([0,1,2],[1]) + α (r + γ*max(Q([0,2],0),Q([0,2],2)) - Q([0,1,2],[1]))

= 0 +0.4(0.1 + 0.8* 0 - 0)

= 0.04

States \ Actions	Take Job 0	Take Job 1	Take Job 2
0,1,2	0	0.04	0
0,1	0	0	
0,2	0		0
1,2		0	0
0	0		
1		0	
2			0
∅			

3. Q –Learning for P-FSSP

Learning

ε = 0.2
α = 0.4
γ = 0.8

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

1st episode:

- Current state: [0,2]
- Choose a random action: [2]
- Sequence: [1,2]
- Reward r = $\frac{1}{Makespan(1,2)} = \frac{1}{13} = 0.0769$
- New Q([0,2],[2])

= Q([0,2],[2]) + α (r + γ*max(Q([0],0)- Q([0,2],[2]))

= 0 +0.4(**0.0769**+ 0.8* 0 - 0)

= 0.0307

States \ Actions	Take Job 0	Take Job 1	Take Job 2
0,1,2	0	0.04	0
0,1	0	0	
0,2	0		0
1,2		0	0
0	0		
1		0	
2			0
Ø			

3. Q –Learning for P-FSSP

Learning

ε = 0.2
α = 0.4
γ = 0.8

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

1st episode:

- Current state: [0,2]
- Choose a random action: [2]
- Sequence: [1,2]
- Reward r = $\frac{1}{Makespan(1,2)} = \frac{1}{13} = 0.0769$
- New Q([0,2],[2])

= Q([0,2],[2]) + α (r + γ*max(Q([0],0)- Q([0,2],[2]))

= 0 +0.4(**0.0769**+ 0.8* 0 - 0)

= 0.0307

States \ Actions	Take Job 0	Take Job 1	Take Job 2
0,1,2	0	0.04	0
0,1	0	0	
0,2	0		0.0307
1,2		0	0
0	0		
1		0	
2			0
Ø			

3. Q –Learning for P-FSSP

Learning

$\varepsilon = 0.2$
$\alpha = 0.4$
$\gamma = 0.8$

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

1st episode:

- Current state: [0]
- Choose a random action: [0]
- Sequence: [1,2,0]
- Reward r = $\frac{1}{Makespan(1,2,0)} = \frac{1}{13} = 0.0769$
- New Q([0],[0])

= Q([0],[0]) + α (r + γ*max(Q([0],Ø)- Q([0],[0])

= 0 +0.4(**0.0769**+ 0.8* 0 - 0)

= **0.0307**

States \ Actions	Take Job 0	Take Job 1	Take Job 2
0,1,2	0	0.04	0
0,1	0	0	
0,2	0		0.0307
1,2		0	0
0	0		
1		0	
2			0
Ø			

3. Q –Learning for P-FSSP

Learning

$\varepsilon = 0.2$
$\alpha = 0.4$
$\gamma = 0.8$

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

1st episode:

- Current state: [0]
- Choose a random action: [0]
- Sequence: [1,2,0]
- Reward r = $\frac{1}{Makespan(1,2,0)} = \frac{1}{12} = 0.0769$
- New Q([0],[0])

= Q([0],[0]) + α (r + γ*max(Q([0],Ø)- Q([0],[0])

= 0 +0.4(**0.0769**+ 0.8* 0 - 0)

= 0.0307

- Makespan= 12

States \ Actions	Take Job 0	Take Job 1	Take Job 2
0,1,2	0	0.04	0
0,1	0	0	
0,2	0		0.0307
1,2		0	0
0	0.0307		
1		0	
2			0
Ø			

3. Q –Learning for P-FSSP

Learning

$\varepsilon = 0.2$
$\alpha = 0.4$
$\gamma = 0.8$

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

2^{st} episode:

- Current state: [0,1,2]
- Base on **ε-greedy algorithm:**
  - Generate a random number r=0.3 > ε
  - Choose the best action => choose job 1
- Sequence: [1]
- Reward r = $\frac{1}{Makespan(1)} = \frac{1}{10} = 0.1$
- New_Q([0,1,2],[1])

= Q([0,1,2],[1]) + α (r + γ*max(Q([0,2],0),Q([0,2],2)) - Q([0,1,2],[1])

= 0.04 +0.4(0.1 + 0.8*max(0;0.0307) - 0.04)

= 0.0738

States \ Actions	Take Job 0	Take Job 1	Take Job 2
0,1,2	0	0.04	0
0,1	0	0	
0,2	0		0.0307
1,2		0	0
0	0.0307		
1		0	
2			0
Ø			

3. Q –Learning for P-FSSP

Learning

$\varepsilon = 0.2$
$\alpha = 0.4$
$\gamma = 0.8$

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

2st episode:

- Current state: [0,1,2]
- Base on **ε-greedy algorithm:**
  - Generate a random number r=0.3 > ε
  - Choose the best action => choose job 1
- Sequence: [1]
- Reward r = $\frac{1}{Makespan(1)} = \frac{1}{10} = 0.1$
- New_Q([0,1,2],[1])

= Q([0,1,2],[1]) + α (r + γ*max(Q([0,2],0),Q([0,2],2)) - Q([0,1,2],[1])

= 0.04 +0.4(0.1 + 0.8*max(0;0.0307) - 0.04)

= 0.0738

States \ Actions	Take Job 0	Take Job 1	Take Job 2
0,1,2	0	0.0738	0
0,1	0	0	
0,2	0		0.0307
1,2		0	0
0	0.0307		
1		0	
2			0
Ø			

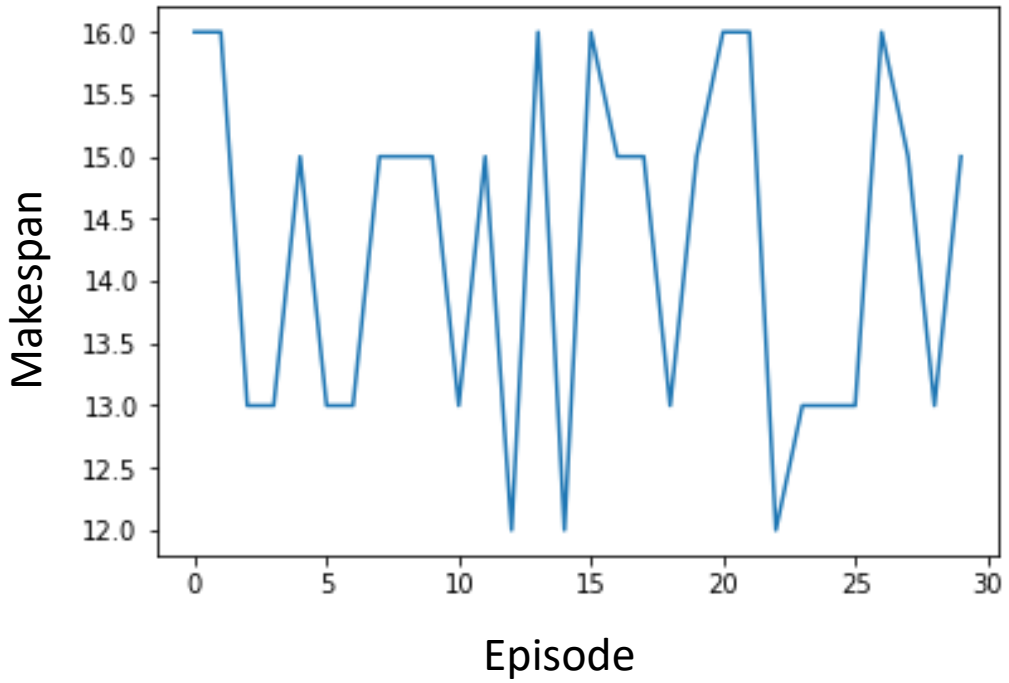
3. Q –Learning for P-FSSP

Learning

$\varepsilon = 0.2$
$\alpha = 0.4$
$\gamma = 0.8$

	Machine 0	Machine 1	Machine 2
Job 0	3	2	0
Job 1	2	4	4
Job 2	2	1	3

Makespan of each episode while learning

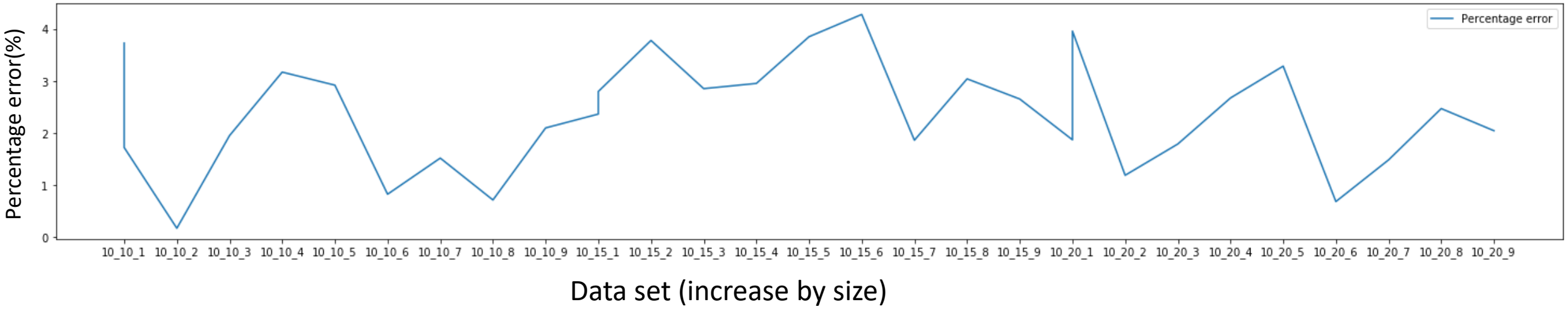


Best sequence: 2-1-0
Best Makespan: 12

4. Experimental results

- The algorithm is tested on 30 small datasets with 10 jobs and up to 20 machines
  - EPSILON = 0.2 # greedy police
  - ALPHA = 0.4 # learning rate
  - GAMMA = 0.8 # discount factor
  - MAX_EPISODES = 3000
- Percentage error = $\frac{\text{Makespan}_{\text{Q learning}} - \text{Makespan}_{\text{optimal}}}{\text{Makespan}_{\text{optimal}}} \times 100\%$

Instance	Optimal Solution	Q learning
10x10	1097	1138
10x15	1307	1338
10x20	1652	1683



5. Conclusion and Further research

- Conclusion
  - The proposed method constitutes an interesting alternative to solve complex scheduling problems
  - This new approach gives a satisfactory solution

- Further research
  - Application of the approach in other types of scheduling problems
  - The effect of parameters to the result
  - Apply the algorithm with bigger dataset
  - Compare with metaheuristics in terms of computational time and result

5. References

- Yunior César Fonseca-Reyna (2019)- A reinforcement learning approach for scheduling problems (2015)
- Yunior César Fonseca-Reyna(2017) An Improvement of Reinforcement Learning Approach for Permutation of Flow-Shop Scheduling Problems
- Yunior César Fonseca-Reyna(2015) Q-learning algorithm performance for m-machine, n-jobs flow shop scheduling problems to minimize makespan
- Frank Benda (2019) - A machine learning approach for flow shop scheduling problems with alternative resources, sequence-dependent setup times, and blocking
- Haldun Aytug, Bhattacharyya (1994)- A Review of Machine Learning in Scheduling
- Yailen Martínez Jiménez (2012)- A Generic Multi-Agent Reinforcement Learning Approach for Scheduling Problems
- Tom Mitchell (1997) - Machine Learning

THANK YOU FOR YOUR ATTENTION