Vijayan S ✉
Parameshwaran Pillai T

https://doi.org/10.21278/TOF.461033121
ISSN 1333-1124
eISSN 1849-1391

APPLICATION OF A MACHINE LEARNING ALGORITHM IN A MULTI STAGE PRODUCTION SYSTEM

Summary

This paper examines a permutation flow-shop scheduling problem, which is a complex combinatorial problem encountered in many practical applications. The objective of the research is to reduce the maximum completion time, i.e., the makespan of all jobs. In order to increase productivity and to meet the demand, manufacturers are continuously under pressure to attain the shortest possible completion time. Estimation of accurate cycle time can tremendously help production planning and scheduling in manufacturing industries. Since production planning is characterised by NP-hardness and a wide range, traditional optimization methods and heuristic rules are unable to find satisfactory solutions. Q-learning, a type of reinforcement learning algorithm, is used in this paper to find a solution that is close to being optimal. Q-learning is a branch of machine learning referring to the way an intelligent agent should act in order to maximize the concept of cumulative reward in a given environment. To validate the performance of the algorithm, Taillard's benchmark problems were solved and compared with the upper bound value. The results showed that the performance of the algorithm is better and has low computational time. Based on the performance of the proposed algorithm, two case studies were done and the solutions are compared with the performance of a metaheuristic algorithm. The result shows that the proposed algorithm can effectively and efficiently solve the problem stated above and that it is an interesting solution to resolving complex scheduling problems.

Key words: *permutation flow shop, Q-learning, reinforcement learning, metaheuristics*

1. Introduction

The permutation flow-shop scheduling is characterised by a unidirectional flow of work in which a range of jobs are processed sequentially in a single pass. All jobs processing times are known in advance and each job is completed in the same order on different machines. It is difficult to recommend a sequence that will decrease the makespan time. As a result, the task entails using an appropriate model to get the optimal sequence to achieve minimum makespan time. The aim of this research is to reduce the completion time of the last job exit from the system, i.e., the makespan time of all jobs. An extensive literature review has been carried out for flow-shop scheduling. Taillard [1] presented 260 randomly created scheduling problems and matched with real-time industry implications. Joanna Jedrzejowicz et al. [2] implemented

the population learning algorithm for solving the permutation flow-shop scheduling problem. Young Hae Lee et al. [3] made use of a mixed integer programming model for attaining the optimum schedule for a no-wait flow-shop scheduling problem.

Several researchers, Deva Prasad [4], Rameshkumar et al. [5], Yannis et al. [6], Radha Ramanan et al. [7], Mohammed Kazen Sayadi et al. [8], and Xuelian Pang et al.[9], worked on various biologically inspired metaheuristic algorithms such as the genetic algorithm, particle swarm optimization, fuzzy particle optimization, a new hybrid particle swarm algorithm, the PSO-NEH-VNS algorithm, a discrete firefly algorithm, a whale optimization algorithm and an improved firework algorithm with the aim of reducing the makespan value and the weighted mean flow time.

Emna Dhouib et al. [10] solved the permutation flow-shop scheduling problem with time lag constraints and sequence dependent setup time. The main objective is to reduce the number of tardy jobs. Two mathematical programming formulations and a simulated annealing algorithm are developed. Deepak Gupta et al. [11] proposed an alternative heuristic algorithm for more than two machines in flow-shop scheduling and compared it with the benchmark algorithms, i.e., Palmer's, the NEH and CDS algorithms. Amar Jukuntla [12] examined the flow-shop scheduling challenges with the goal of reducing the total completion time of all jobs and maximizing the machine utilization. The proposed hybrid mechanism produces a better result when compared with the traditional algorithm. Goksu Erseven et al. [13] solved the real world problem of quality control process which has a permutation flow-shop scheduling problem (PFSP) layout. A combination of two heuristic methods is employed to solve this problem. Arshad Ali et al. [14] considered the distributed PFSP which is the extension of PFSP. Tabu search (TS), a metaheuristic approach is proposed, and numerical experiments are carried out on benchmark problems. Narayana Prasad et al. [15] solved the m-machine no-wait flow-shop scheduling problem using a branch and bound algorithm with the objectives of weighted tardiness, weighted earliness, and weighted flowtime. Abduljaleeljersin et al. [16] were focused on developing the optimal solution for the n-jobs, m-machine permutation flow-shop scheduling problem. The Lagrangian relaxation (LR) techniques were used to solve that problem. A comprehensive review of branch and bound algorithms used for solving the permutation flow-shop scheduling problem is presented by Caiopaziani Tomazella et al. [17]. Recently, researchers, such as Frank Benda et al. [18], Jatinder N.D. Gupta et al. [19], Jianfeng Ren et al. [20], have attached more significance to approaches related to machine learning, artificial neural networks, and reinforcement learning algorithms.

From the literature review, it seems that a small number of researchers have contributed to the implementation of machine learning algorithm in solving the permutation flow-shop scheduling problem. In order to solve the PFSP, reinforcement learning (RL) is proposed in this paper. Reinforcement learning algorithms belong to a subclass of machine learning algorithm; they are focused on how software agents would take actions in a given environment. The remainder of the paper is structured as follows: formulation of a mathematical model is discussed in Section 2; The Reinforcement Learning Algorithm has been discussed in detail in Section 3; the experimental results on the benchmark problem are presented in Section 4 and case studies are validated in Section 5; conclusion and future scope are elaborated in Section 6.

2. Mathematical Model

In the flow-shop scheduling problem, there are n number of machines and m number of jobs. There are exactly 'n' operations in each job, carried out in the same sequence of operations. The first operation is carried out on the first machine, followed by the second operation on the second machine, once the first operation is completed, and so on, until the

completion of the n-th operation. Figure 1 shows that no machine can perform more than one operation at an instant.

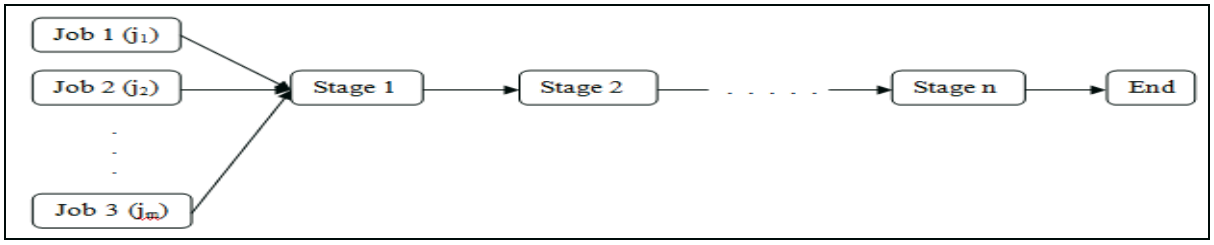

Fig. 1 Layout of flow-shop scheduling

The processing times of all jobs are determined. The jobs, on the other hand, can be finished in any order. According to the problem specification, the job order is the same for each machine. The challenge is determining the best job sequence for a minimum makespan. Due to the NP-hard nature of the flow-shop scheduling problem, it is usually solved using approximation or heuristic methods [21]. The notation n|m|p|Cmax [22] is used to describe the problem studied in this study; the assumptions of the permutation flow-shop problem are shown below.

1. At the same time, a machine cannot process more than one job;
2. Each job is processed on a single machine at a time;
3. Each job processing time on each machine is specified;
4. The setup times of operations are sequence independent and are included in the processing time;
5. Each machine type has a single unit;
6. At any given time, no more than one operation of the same job can be performed;
7. Pre-emption is not allowed.

The processing time of job i on machine j, the permutation of job (J1, J2,...,Jn), and the makespan calculation are shown below.

$C(J_1, 1) = p(J_1, 1)$

$C(J_i, 1) = C(J_{i-1}, 1) + p(J_i, 1)\ for\ i = 2, \dots, n$

$C(J_1, j) = C(J_1, j-1) + p(J_1, j)\ for\ j = 2, \dots, m$

$C(J_i, j) = max\{C(J_{i-1}, j), C(J_i, j-1) + p(J_i, j)\}\ for\ i = 2, \dots, n;\ for\ j = 2, \dots, m$

$Cmax = C(J_n, m)$

Cmax represents the completion time of the last job exit from the last machine, ie. the makespan time.

3. Reinforcement Learning Algorithm

Machine learning is a part of artificial intelligence that is facilitated by learning from data, thereby identifying patterns and making decisions with minimal interference from humans. The Reinforcement Learning (RL) algorithm is a type of machine learning technique in which a software agent learns by trial and error and thereby gets rewards and punishment based on the action.

Sutton and Barto [23] pioneered the concept that underpins reinforcement learning and applied it to the topics of Artificial Intelligence (AI). The RL algorithm performs based on a set of parameters which includes the environment, state, reward, policy, and value. In the reinforcement learning model, an agent interacts with its environment as illustrated in Fig. 2.

The agent understands its current state ‘s’ and performs an action ‘a’ in each interaction step, thereby transmitting to the new state. While transmitting, the RL agent receives a signal ‘r’ which is a reward or a punishment. The RL algorithm learns to maximize the total reward signal ‘r’ received by the agent ‘a’.

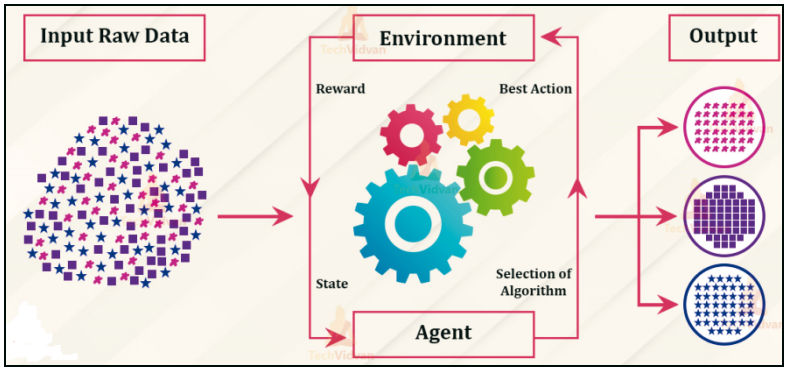

Fig. 2 Reinforcement learning

The Markov property has a typical type of environment. In such an environment, the future state depends only on the current state. Most of reinforcement learning researchers have concentrated on learning in this type of environment, which has resulted in the development of a number of notable reinforcements learning methods such as the Q-learning algorithm.

3.1 Q-Learning Algorithm for Solving the Flow-Shop Scheduling Problem (FSSP)

Q-Learning (QL) is a well-known RL technique that learns an action-value function that exposes the cumulative reward of executing a given action in a given state [24]. Figure 3 shows the QL algorithm process.

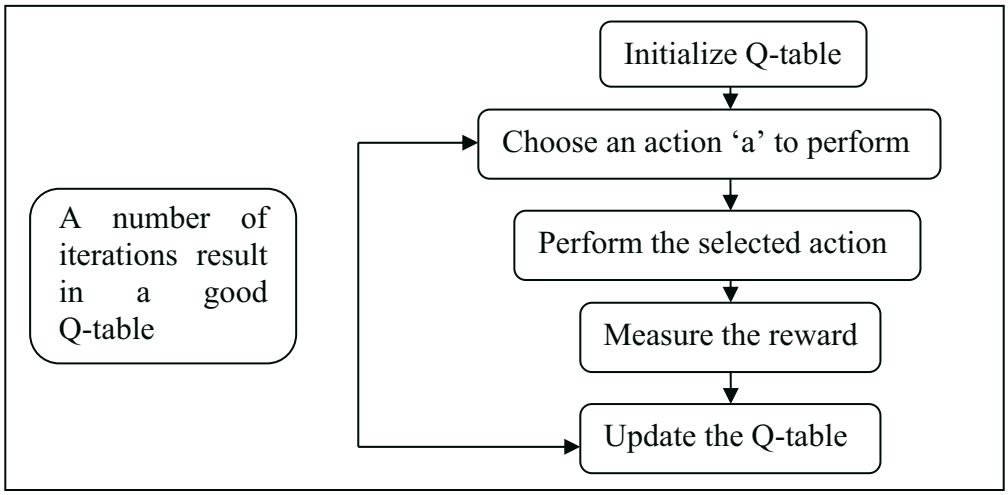

Fig. 3 Q- learning algorithm process

The method is based on the basic value iteration update, with a Q-value assigned to each state-action pair (s, a). When an agent in the state s selects an action a, the Q-value for that state-action pair is updated using the reward earned for selecting that action as well as the optimal Q-value for the next state s′. The state action pair (s, a) update rule is as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\,[r + \gamma\, max a'\; Q(s', a') - Q(s, a))]$$

The learning rate is α which belongs to [0, 1] in this expression, and, as a result of taking the action ‘a' in the state ‘s'; the reward or penalty is ‘r'. The degree to which the old

value is updated is affected by the learning rate. In the Markov decision processes, QL has been proven to converge to the best policy under specific conditions.

In the FSSP, all the jobs passing through the machine have the same operation order. This model takes the operational times as input parameters in order to find a sequence of jobs which reduces the idle time on a long-term basis.

In this study, n|m|p|Cmax are considered in the case when there is only one agent connected with the first resource (machine). This agent will make future decisions regarding how to proceed. The task of this agent is to take action to choose which job to do next from the list of currently available jobs. When a job is chosen, it is sent to all the machines for processing. The agent can either choose the best job based on the related q-value (exploration) or choose a task at random (exploration). The greedy strategy outlined in [24] is used to carry out the action selection process.

The proposed algorithm is given as:

Application of Q-learning algorithm to solve FSSP

```
Initialize
Q (s,a) = {}
Best = {}
For each episode step do
    Initialize s = {}
    While not _finished (all jobs)
        Select the jobs with the longest processing time Jm
        Commence actions as the possible insertion set points of Jm in state
        Select action from state using the policy obtained from Q (e.g., ε-greedy)
        Take an action a, find state s and reward r as 1/makespan (s)
        Q(s,a) ← Q (s,a) – α [ r – γ max a Q(s',a') – Q( s,a)
        s ← s'
  end while
  if makespan (s) < makespan (Best)
  Best ⟵ state
End for
```

4. Experimental Result

In order to assess the result of the algorithm, several benchmark instances of the OR-Library [25] were used. The OR-Library is a collection of problem instances that cover a wide range of operation research problems. E. Taillard [1] proposes 120 instances for FSSP, which are divided into ten groups based on the number of machines and jobs. The benchmark problem consists of the number of machines, jobs, as well as the processing time of each job on each machine. Different benchmark problems were selected randomly, and the results were compared with the upper bound makespan value.

Taillard's data set was used to select 47 instances, with problem sizes of 20x5, 20x10, 20x20, 50x5, 50x10, 50x20, and 100x5. Q-learning was implemented in the Python programming and ran with a configuration of a Core i5 processor, 8 GB of RAM, and a 2.3 GHz CPU.

Initially, an experiment was carried out to examine the learning process in relation to various parameter settings. The combination of learning process parameters is shown below in Table1.

Table 1 Learning process parameters

Episodes	α	γ	E
C1: episodes = $n * m$	0.1	0.8	0.2
C2: episodes = $n * m$	0.1	0.9	0.1
C3: episodes = $n * m$	0.1	0.8	0.1
C4: episodes = $n * m$	0.1	0.9	0.2

After carrying out different experiments with these four combinations, the third combination (α = 0.1, γ = 0.8, ε = 0.1) yielded better results. With this parameter, Q- learning was performed for different instances which were selected from Taillard's benchmark dataset. For each problem, the procedure was repeated with 20 epochs and the best makespan values were selected. The relative error (RE) was calculated as follows:

$$RE = [\ (MK - UB/\ UB) * 100]$$

The upper bound value and the optimum makespan value of the Q-learning algorithm, the relative error, the mean relative error (MRE), and computational time are shown in Table 2.

Table 2 Makespan value and computational time of the QL algorithm

Benchmark Instance		QL Makespan	Computational Time (min)	Optimal Makespan	Relative Error (%)
Ta001		**1278**		**1278**	**0.00**
Ta002		1366		1359	0.51
Ta003	20 x 5	1098	0.2312	1081	1.57
Ta004		1307		1293	1.08
Ta005		**1236**		**1236**	**0.00**
Ta006		**1206**		**1206**	**0.00**
				MRE	**0.53 %**
Ta011		1584		1582	0.13
Ta012		**1659**		**1659**	**0.00**
Ta013		1515		1496	1.27
Ta014	20 x 10	1398	0.2579	1378	1.45
Ta015		1450		1419	2.18
Ta017		**1484**		**1484**	**0.00**
Ta020		1598		1591	0.43
				MRE	**0.78 %**
Ta021		2330		2297	1.43
Ta022		2110		2100	0.48
Ta023	20 x 20	2342	0.4180	2326	0.68
Ta024		2229		2223	0.27

Benchmark Instance		QL Makespan	Computational Time (min)	Optimal Makespan	Relative Error (%)
Ta025		2301		2291	0.44
Ta027		2295		2273	0.97
				MRE	**0.71 %**
Ta031		**2724**		**2724**	**0.00**
Ta032		2842		2834	0.28
Ta033		**2621**		**2621**	**0.00**
Ta034		2762		2751	0.39
Ta035	**50 x 5**	**2863**	1.4160	**2863**	**0.00**
Ta036		2831		2829	0.07
Ta037		2728		2725	0.11
Ta040		**2782**		**2782**	**0.00**
				MRE	**0.11 %**
Ta041		3059		3025	1.12
Ta042		2934		2892	1.45
Ta043	**50 x 10**	2932	1.5833	2864	2.37
Ta044		3115		3064	1.66
Ta045		3115		2986	4.32
				MRE	**2.18 %**
Ta051		4010		3875	0.03
Ta052		3728		3715	0.34
Ta053		3808		3668	3.81
Ta054	**50 x 20**	3844	1.6000	3752	2.45
Ta055		3815		3635	4.95
Ta057		3760		3716	1.18
Ta059		3810		3765	1.19
				MRE	**1.99 %**
Ta061		**5493**		**5493**	**0.00**
Ta062		5290		5268	0.42
Ta063		5177		5175	0.039
Ta064	**100 x 5**	5023	1.5000	5014	0.18
Ta065		5266		5250	0.30
Ta067		**5246**		**5246**	**0.00**
Ta070		**5328**		**5328**	**0.00**
				MRE	**0.13 %**

From Table 2, one can see that the makespan of QL algorithm matches the upper bound value with the precision of the result shown in Fig. 4. Based on the result, the Q-learning algorithm shows better results. The average relative error of all instances was less than one percent. Based on the efficiency of this algorithm, two case studies were done.

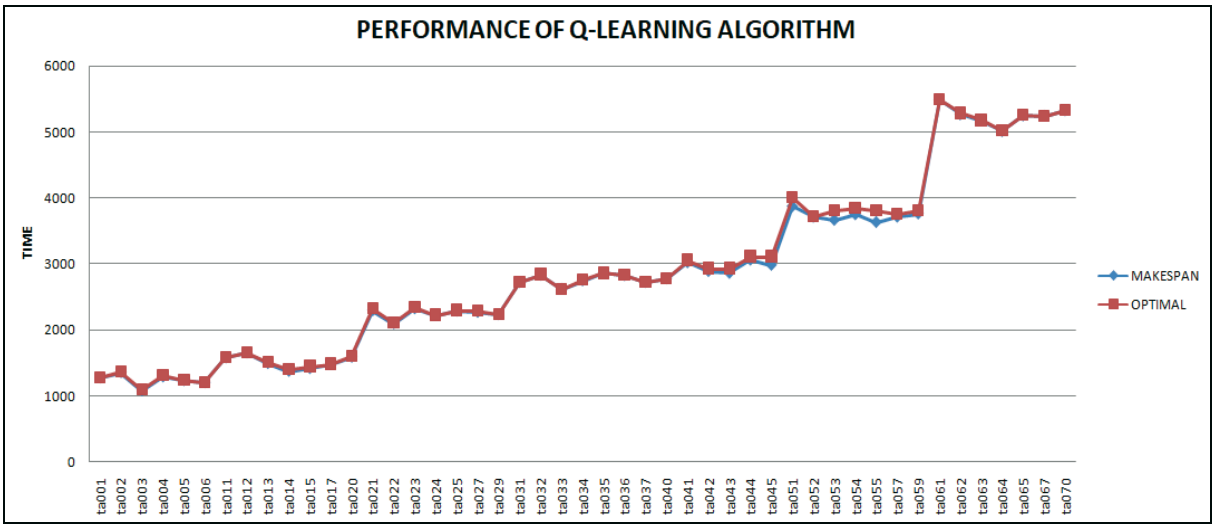

Fig. 4 Performance of the QL algorithm

5. Case Study Validation

5.1 Case Study 1: Plastic Toy Manufacturing

Based on the performance of the algorithm on benchmark instances, its efficiency is tested on a toy manufacturing problem [9]. A plastic toy set is composed of seven small parts. Every part has to undergo a seven-stage process; the stages in the process are: melting, shaping, grinding, surface decoration, dyeing, testing, and packing. Table 3 lists the processing time for each job in each machine.

Table 3 Processing times for plastic toy manufacturing

Stages	Jobs						
	0	1	2	3	4	5	6
Melting	692	581	475	23	158	796	542
Shaping	310	582	475	196	325	874	205
Grinding	832	14	785	696	530	214	578
Surface decoration	630	214	578	214	785	236	963
Dyeing	258	147	852	586	325	896	325
Testing	147	753	002	356	565	898	800
Packing	255	806	699	877	412	302	120

The above problem is a permutation flow-shop problem consisting of 7 jobs to be done in 7 machines. The main objective is to reduce the completion time of the last job in the last machine; this is also known as the makespan value. In order to achieve this objective, the algorithm was run 20 times independently and the number of iterations was set to 200. The optimum parameters of the QL algorithm are given in Table 4.

Table 4 Optimum learning process parameters

Algorithm	Parameters	Value
Q-learning	Alpha	0.8
	Beta	0.1
	Gamma	0.1

The makespan time of the proposed Q-learning algorithm is 6590s, as shown in Table 5. The result of the algorithm is compared with the performance of the metaheuristic algorithm presented in [9]. The proposed algorithm shows the minimum completion time and a low average value. The computational time is reduced by 18%. The optimum sequence and the scheduling time, i.e., in and out time of each machine, and the optimum makespan are shown in Table 6 and are also presented by means of the Gantt chart shown in Fig. 5.

Table 5 Comparison result of the QL algorithm

Result	Algorithm				
	QL	IFWA	FWA	PSO	WOA
Optimal Value	6590	6590	6590	6590	6590
Average	6595.3	6595.3	6603.2	6665	6614
Worst	6643	6643	6643	7016	6643
Time	0.046	2.14	3.40	0.34	0.26

Table 6 Scheduling time of optimum sequence

Job	M/C 1		M/C 2		M/C 3		M/C 4		M/C 5		M/C 6		M/C 7	
	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
4	0	158	158	483	483	1031	1031	1798	1798	2123	2123	2688	2688	3100
3	158	181	483	679	1013	1709	1798	2012	2123	2709	2709	3065	3100	3977
1	181	762	762	1344	1709	1723	2012	2226	2709	2856	3065	3818	3977	4783
5	762	1558	1558	2432	2432	2646	2646	2882	2856	3778	3818	4716	4783	5085
6	1558	2100	2432	2637	2646	3224	3224	4187	4187	4512	4716	5516	5516	5636
2	2100	2575	2637	3112	3224	4009	4187	4765	4765	5617	5617	5619	5636	6335
0	2575	3267	3267	3577	4009	4841	4841	5471	5617	5875	5875	6022	6335	**6590**

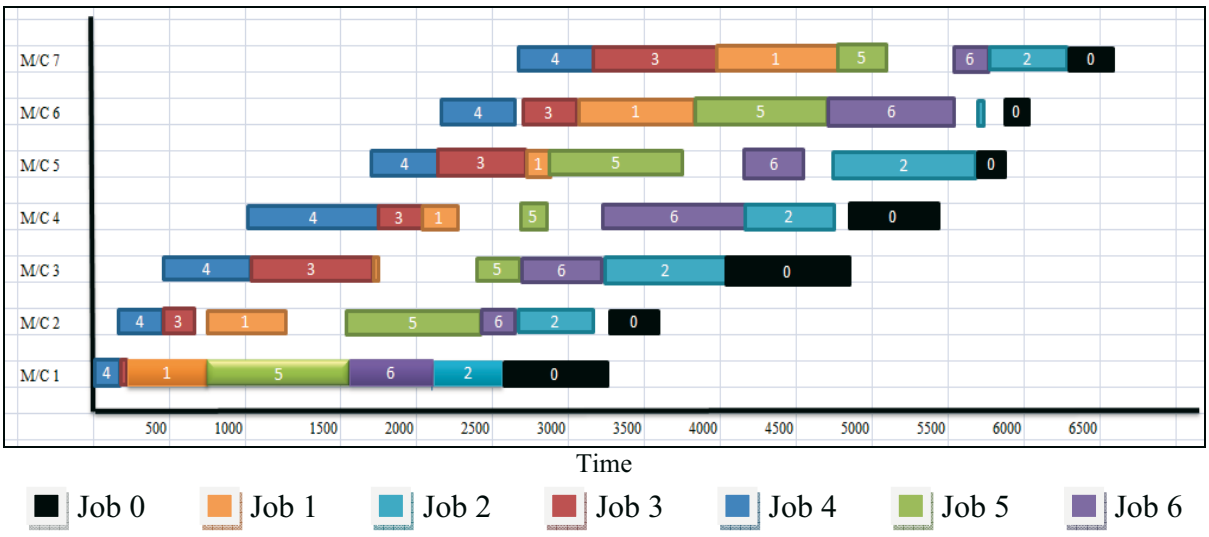

Fig. 5 A schedule for optimum sequence

5.2 Case study 2: Stator Core Manufacturing

The second case study is the manufacturing of stator core which supports and protects the three-phase winding of the stator. The stator core for a vehicle alternator is manufactured using thin metal laminated plates to form an annular cylinder which has a predetermined thickness in the stacking direction. The thin metal plates are bonded together to form the stator core.

The stator core manufacturing units were investigated in a leading automobile component manufacturing company, Chennai, India. The layout followed by the company is the permutation flow-shop layout. The stator core manufacturing has seven workstations (stages) consisting of a blanking and curling machine, a riveting machine, a coining machine, a T'Bolt milling machine, a groove milling machine, another coining machine and an OD turning machine in series. There are nine types of stator core jobs. Each job has a fixed processing time on the stages as shown in Table 7. All the jobs have to follow the same sequence 1, 2……. M. The company production sequence is done manually; consequently, there is a gap between demand and production. In order to increase productivity, scheduling of jobs is needed. This problem is termed a permutation flow-shop scheduling problem with 'n' jobs and 'm' machines. The main objective here is to reduce the makespan time.

Table 7 Processing time for stator core manufacturing

Part No	Blanking & Curling	Riveting	Coining	T'Bolt Milling	Groove Milling	Second Coining	OD-Turning
0	27	18	28	0	22	28	44
1	27	18	21	0	22	21	31
2	27	24	28	35	0	28	74
3	27	18	30	35	0	30	31
4	27	18	32	35	0	32	74
5	27	24	30	0	22	30	44
6	27	24	28	35	0	28	44
7	27	24	21	35	0	21	31
8	27	24	32	35	0	32	74

In order to achieve a minimum makespan time, the Q-learning algorithm is used. The minimum makespan time achieved is 566 s. The optimum sequence of jobs for the stator manufacturing, the scheduled time of each machine, and the computational time are shown in Table 8 and 9.

Table 8 Optimum values of the Q-learning algorithm

Optimum Sequence	Makespan Value (Sec)	Computational Time (Sec)
1-0-7-5-3-4-2-8-6	566	2.49

Table 9 Scheduling time for the optimum sequence (stator core manufacturing)

Job	M/C 1		M/C 2		M/C 3		M/C 4		M/C 5		M/C 6		M/C 7	
	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
1	0	27	27	45	45	66	-	-	66	88	88	109	109	140
0	27	54	54	72	72	100	-	-	100	122	122	150	150	194
7	54	81	81	105	105	126	126	161	-	-	161	182	194	225
5	81	108	108	132	132	162	-	-	162	184	184	214	225	269
3	108	135	135	153	162	183	183	218	-	-	218	248	269	300
4	135	162	162	180	183	215	218	253	-	-	253	285	300	374
2	162	189	189	213	215	243	253	288	-	-	288	316	374	448
8	189	216	216	240	243	275	288	323	-	-	323	355	448	522
6	216	243	243	267	275	303	323	358	-	-	358	386	522	**566**

The scheduling process in the stator core manufacturing company is done manually; consequently, the demand is not met and there is a decrease in productivity. Hence, the proposed computational method is cost effective, it is not time- consuming, and it can help in achieving the optimal schedule. The optimal schedule with a minimum makespan time greatly helps in reducing the gap between demand and production.

6. Conclusion

In this paper, the Q-learning algorithm was implemented to resolve permutation flow-shop scheduling problems. The algorithm performance was assessed using the benchmark problem which is in the OR-Library. The proposed algorithm was applied to two case studies. Based on the results obtained in the case studies, it was concluded that the proposed Q-learning algorithm is an effective solution for resolving complex scheduling problems; it is simple and easy to implement. The makespan value of the Q-learning algorithm matches the upper bound value of the benchmark problem. The algorithm gave the minimum makespan time for the two case studies with minimum computational time and it helped increase productivity. In future research, the proposed algorithm will be used to solve various types of layout problems, including multi-objective problems in a dynamic environment with simulation.

REFERENCES

[1] Taillard, E. Benchmarks for basic scheduling problems. European Journal of Operational Research **1993**, 64, 278-285. https://doi.org/10.1016/0377-2217(93)90182-M

[2] Joanna Jedrzejowicz.; Piotr Jedrzejowicz. New Upper Bounds for the Permutation Flowshop Scheduling Problem, *Innovations in Applied Artificial Intelligence* **2005**, 232-235. https://doi.org/10.1007/11504894_33

[3] Young Hae Le.; Jung Woo Jung. New Heuristics for No-Wait Flowshop Scheduling with Precedence Constraints and Sequence Dependent Setup Time, *Computational Time and its application* **2005**, 467–476. https://doi.org/10.1007/11424925_50

[4] Deva Prasad,S. A Genetic Algorithmfor Flowshop Scheduling with Multiple Objectives, *Operational Research Society of India* **2007**, VOL. 44, 1-16. https://doi.org/10.1007/BF03398787

[5] Rameshkumar,K.; Suresh, R.K.; and Mohanasundaram,K.M. Discrete Particle Swarm Optimization (DPSO) Algorithm for Permutation Flowshop Scheduling to Minimize Makespan, *Advances in natural Computation* **2005**, 572 – 581. https://doi.org/10.1007/1153990270

[6] Yannis Marinakis.; Magdalene Marinaki.; A Hybrid Particle Swarm Optimization Algorithm for the Permutation Flowshop Scheduling Problem, *Optimization Theory, Decision Making, and Operations Research Applications* **2013**, 91-101. https://doi.org/10.1007/978-1-4614-5134-1_6

[7] Radha Ramanan.; Muhammed Iqbal.; Umarali,K. A particle swarm optimization approach for permutation flow shop scheduling problem, *International Journal for Simulation and Multidisciplinary Design Optimization* **2014**, 5(2):A20. https://doi.org/10.1051/smdo/2013006

[8] Mohammad Kazem Sayadia.; Reza Ramezaniana.; Nader Ghaffari-Nasaba. A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. *International Journal of Industrial Engineering Computations* **2010**, 1-10. https://doi.org/10.5267/j.ijiec.2010.01.001

[9] Xuelian Pang.; Haoran Xue.; Ming Lang Tseng.; Ming K, Lim.; Kaihua Liu, Hybrid flowshop scheduling problems using improved Fireworks Algorithm for Permutation, *Applied Sciences* **2020,** 10,1174. https://doi.org/10.3390/app10031174

[10] Emna Dhouib.; Jacques Teghem.; Taicir Loukil. Minimizing the Number of Tardy Jobs in a Permutation Flowshop Scheduling Problem with Setup Times and Time Lags Constraints, *Journal of Mathematical Modelling and Algorithms* **2013**,85–99. https://doi.org/10.1007/s10852-012-9180-x

[11] Deepak Gupta.; Kewal Krishan Nailwal.; Sameer Sharma. A Heuristic for Permutation Flowshop Scheduling to Minimize Makespan, *Advances in Intelligent Systems and Computing* **2014**, 259. https://doi.org/10.1007/978-81-322-1768-8_38

[12] Amar Jukuntla. Optimization Technique for Flowshop Scheduling Problem, Advances *in Intelligent Systems and Computing* **2018**, 668. https://doi.org/10.1007/978-981-10-7868-2_18

[13] Goksu Erseven.; Gizem Akgun.; Aslıhan Karakaş.; Gozde Yarıkcan.; ozgun Yucel.; Adalet oner. An Application of Permutation Flowshop Scheduling Problem in Quality Control Processes, *Proceedings of the International Symposium for Production Research* **2019**, 849–860. https://doi.org/10.1007/978-3-319-92267-6_68

[14] Arshad Ali.; Yuvraj Gajpal.; Tarek,Y.; Elmekkawy. Distributed permutation fowshop scheduling problem with total completion time objective, *Operational Research Society of India* **2020**. https://doi.org/10.1007/s12597-020-00484-3

[15] Narayanaprasad Madhushini.; Chandrasekharan Rajendran. Branch-and-bound algorithms for scheduling in an m-machine no-wait flowshop, *Indian Academy of Sciences* **2020**. https://doi.org/10.1007/s12046-020-01432-z

[16] Abduljaleel Jessin.; Sakthivel Madankumar.; Chandrasekharan Rajendran. Permutation flowshop scheduling to obtain the optimal solution/a lower bound with the makespan objective, *Indian Academy of Sciences* **2020**, 45:228. https://doi.org/10.1007/s12046-020-01444-9

[17] Caio Paziani Tomazella.; Marcelo Seido Nagano. A comprehensive review of Branch-and-Bound algorithms: Guidelines and directions for further research on the flowshop scheduling problem, *Expert Systems with Applications* **2020**, 158. https://doi.org/10.1016/j.eswa.2020.113556

[18] Frank Benda.; Roland Braune.; Karl F. Doerner.; Richard F. Hartl. A machine learning approach for flow shop scheduling problems with alternative resources, sequence –dependent setup times, & blocking,*OR spectrum* **2019**,41:871-893. https://doi.org/10.1007/s00291-019-00567-8

[19] Jatinder N. D. Gupta.; Arindam Majumder.; Dipak Laha. Flowshop scheduling with artificial neural networks, *Journal of the Operational Research Society* **2020**, https://doi.org/10.1080/01605682.2019.1621220

[20] Jianfeng Ren.; Feng Yang. Solving flow-shop scheduling problem with a reinforcement learning algorithm that generalizes the value function with neural network, *Alexandria engineering journal* **2021**, 2787 – 2800. https://doi.org/10.1016/j.aej.2021.01.030

[21] Alvarez, M.; Toro,E.; Gallego,R. Simulated Annealing Heuristic For Flow Shop Scheduling Problems. *Scientia et Technica* **2008**, XIV, 159-164.

[22] Reeves, C. R. A genetic algorithm for flowshop sequencing. *Computers & Operations Research* (1995), 22, 5-13.https://doi.org/10.1016/0305-0548(93)E0014-K

[23] Sutton, R.; Barto, A.Reinforcement Learning (An Introduction). *The MIT Press, Cambridge, Massachusetts* 1998.

[24] Martinez, Y.A Generic Multi-Agent Reinforcement Learning Approach for Scheduling Problems. *PhD Thesis, Vrije Universiteit Brussel* **2012**, 169 p.

[25] Beasley.; J. E. OR-Library: distributing test problems by electronic mail, Journal of the Operational Research Society 41(11) (1990), 1069-1072. http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

Submitted: 08.8.2021

Accepted: 08.02.2022

Vijayan.S*
Assistant Professor,
Department of Mechanical Engineering,
J.J. College of Engineering and Technology, Trichy – 620009, India
Vijayselva.87@gmail.com

Dr.T.Parameshwaran Pillai
Assistant Professor (Sr.Gr),
Department of Mechanical Engineering,
University College of Engineering,
BIT Campus, Anna University,
Trichy-620024, India
Paramesh551@yahoo.com
*Corresponding author.