Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

MONTE CARLO TREE SEARCH-BASED ALGORITHM FOR DYNAMIC JOB SHOP SCHEDULING WITH AUTOMATED GUIDED VEHICLES

Duyeon Kim
Hyun-Jung Kim*

Department of Industrial and Systems Engineering
Korea Advanced Institute of Science and Technology (KAIST)
291 Daehak Street
Daejeon, 34141, REPUBLIC OF KOREA

ABSTRACT

A dynamic job shop scheduling problem where jobs are transported by automated guided vehicles (AGVs) is considered to minimize the mean flow time. This problem is first modeled with a timed Petri net (TPN) which is widely used for modeling and analyzing discrete event systems. A firing rule of transitions in a TPN is modified to derive more efficient schedules by considering jobs that have not arrived yet and restricting the unnecessary movement of the AGVs. We propose a Monte Carlo Tree Search (MCTS)-based algorithm for the problem, which searches for schedules in advance within a time given limit. The proposed method shows better performance than combinations of other dispatching rules.

1 INTRODUCTION

Many manufacturing systems have been using automatic transport robots, such as overhead hoist transports (OHTs) or automated guided vehicles (AGVs), to automate the systems and improve their productivity. It is important to obtain a schedule by considering both machines and robots at the same time because these two different resource types are closely related in that jobs must use two resource types in turn.

There have been several studies on job shops with transport robots. Bilge and Ulusoy (1995) addressed integrated scheduling of machines and robots with a time window heuristic approach and provided the benchmark instances, which are used for performance evaluation in this study. Other studies have also developed efficient meta heuristic based algorithms, such as the tabu search, simulated annealing, and genetic algorithm (Deroussi et al. 2008; Zheng et al. 2014; Reddy and Rao 2006). Recently, Ham (2021) applied constraint programming(CP) to solve large-sized instances of job shop scheduling with robots.

In practice, it is common to observe dynamic automated manufacturing environments, such as dynamic job arrivals, machine breakdowns, and battery charging of the robots. However, most previous studies on job shops with AGVs have focused on deriving a schedule without considering dynamic events, due to their complexity. We, therefore, consider a dynamic job shop scheduling problem with AGVs where the arrival time of jobs is not known. Unpredictable arrival of jobs frequently occurs in flexible and customized manufacturing systems. There have been many studies on dynamic job shops by developing dispatching rules (Dabbas et al. 2001; Dominic et al. 2004), simulation methods (Xiong et al. 2017), heuristics (Kundakc and Kulak 2016), and reinforcement learning (Liu et al. 2022). However, they have not considered AGVs in the system.

In this study, a timed Petri net (TPN) is used for modeling the job shop scheduling problem with AGVs (Wang 2012). A TPN, which consists of places, transitions, arcs, and tokens, is widely used for modeling and analyzing discrete event dynamic systems. An extended disjunctive graph can also be used

for modeling the problem, but it is hard to represent robots moving between machines with the graph (Lacomme et al. 2013).

In dynamic job shops with AGVs, it is important to use future state information when assigning jobs to machines or AGVs to obtain good schedules. Specifically, it can be better to wait for a job that is being processed in the previous machine if its processing time in the next machine is large. However, firing only enabled transitions in a TPN cannot derive such a schedule. We, therefore, propose a modified enabling rule with relaxed conditions compared to the original TPN in order to consider such jobs that are not available for machines or robots at the current time. We further propose a strategy that restricts the unnecessary movement of transport robots to reduce the search space. We then propose a Monte Carlo tree search (MCTS)-based scheduling algorithm based on the TPN model and adjust a search step for a dynamic job shop problem. The proposed method shows better performance compared to several combinations of machine and AGV dispatching rules. We first explain the problem and TPN model in Section 2 and the MCTS algorithm in Section 3. We then show the experimental results and conclusion in Sections 4 and 5, respectively.

2 PROBLEM AND MODELING

2.1 Problem Description

We consider a job shop problem with N AGVs and M machines (Bilge and Ulusoy 1995). There are J job types, each of which has multiple sequential operations, o_{jk}, $1 \leq j \leq J, 1 \leq k \leq K$, which indicates the kth operation of job j. Each job type j has n_j units. All jobs enter or exit the system through the load/unload (L/U) stocker, where all the robots are located initially. A robot can transport one job between two machines, and the transportation time depends on the distance between them. When the AGV moves to a machine for unloading a job, it is called an empty trip whereas a load trip is used when the AGV is transporting a job for its next process. All of the processing times and the transportation times are deterministic. Jobs arrive randomly to the L/U stocker, and it is assumed that the arrival time interval follows the Exponential distribution. Other distribution can also be assumed.

Figure 1 shows four different layouts of the job shop with AGVs (Bilge and Ulusoy 1995). Each machine has an unlimited input/output buffer for pickup/delivery (P/D) points for the robots, respectively, and no preemption is allowed for all the operations. It is assumed that each robot moves along the shortest path, no collision occurs, and there is no delay in travel time from the congestion. The objective is to minimize the mean flow time of all jobs.

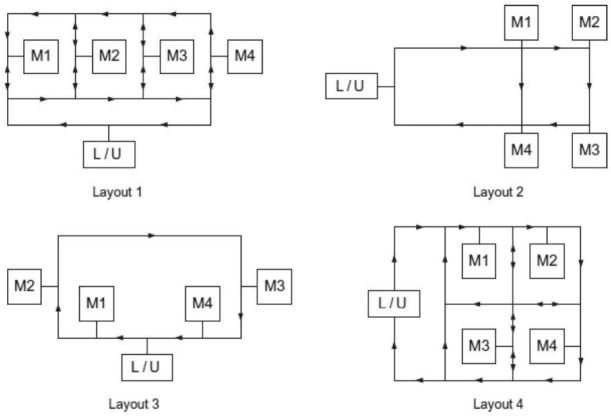

Figure 1: Problem layout configurations.

2.2 Problem Modeling: Timed Petri Net (TPN)

We first model the job shop scheduling problem with AGVs by using a TPN. A TPN is a bipartite directed graph, which can well represent system dynamics with tokens and interrelations between resources (Wang 2012). Figure 2 shows a simple example with two machines, two job types, and two AGVs. The token marked at p10 means that job 2 has arrived at the L/U stocker, and the token at p22 represents the availability of AGV located in front of machine 1. A detailed description of each place and the transition is given in Table 1. We define four types of places, p_{ma}, p_{bf}, p_{lt}, and p_{ra} which indicate places for machine availability, input/output buffers, load trips, and robot availability, respectively, and six types of transitions, t_{start_lt}, t_{end_lt}, t_{start_et}, t_{end_et}, t_{start_ps}, and t_{end_ps}, which represent the start and end of the load trip, empty trip, and process, respectively.

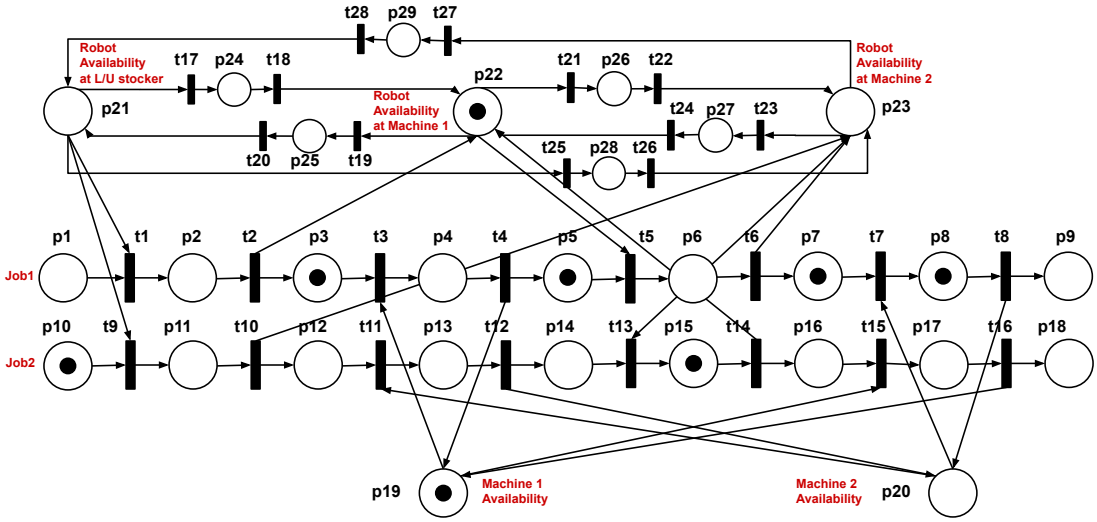

Figure 2: A TPN with two machines.

2.3 Modified Enabling Rule

A transition is enabled if all of its input places contain a token, and the token stays in the input place longer than the holding time. Scheduling of a TPN is to determine the firing sequence of enabled transitions connected to a conflict place.

Figure 3 is a part of Figure 2, and left and right figures show enabled transitions in red according to the original enabling rule and our modified enabling rule, respectively. In Figure 3, t3 is enabled and can be fired, which leads to the process of job type 1 in machine 1. Or, waiting for job type 2 (t15), which is being transported to machine 1, can also be an alternative and provide a better schedule. However, with the original enabling rule, t14 cannot be enabled, and we, therefore, modify the enabling rule as follows:

Definition 1 A transition of type t_{start_ps} is enabled if its p_{ma} type input place is marked and p_{bf} type input place or its preceding place p_{lt} is marked.

This enabling rule makes it possible to consider jobs that are being transported to the corresponding machine or processing in the previous machine.

In Figure 3, the token marked p22 indicates that there is an available AGV in front of machine 1. In this case, transitions for empty trips (t19, t21) are always enabled, which can cause unlimited and unnecessary firing of empty trip transitions. An empty trip transition should be fired by following the corresponding load trip transition if the AGV is not located at the requested machine. The following enabling rules explain

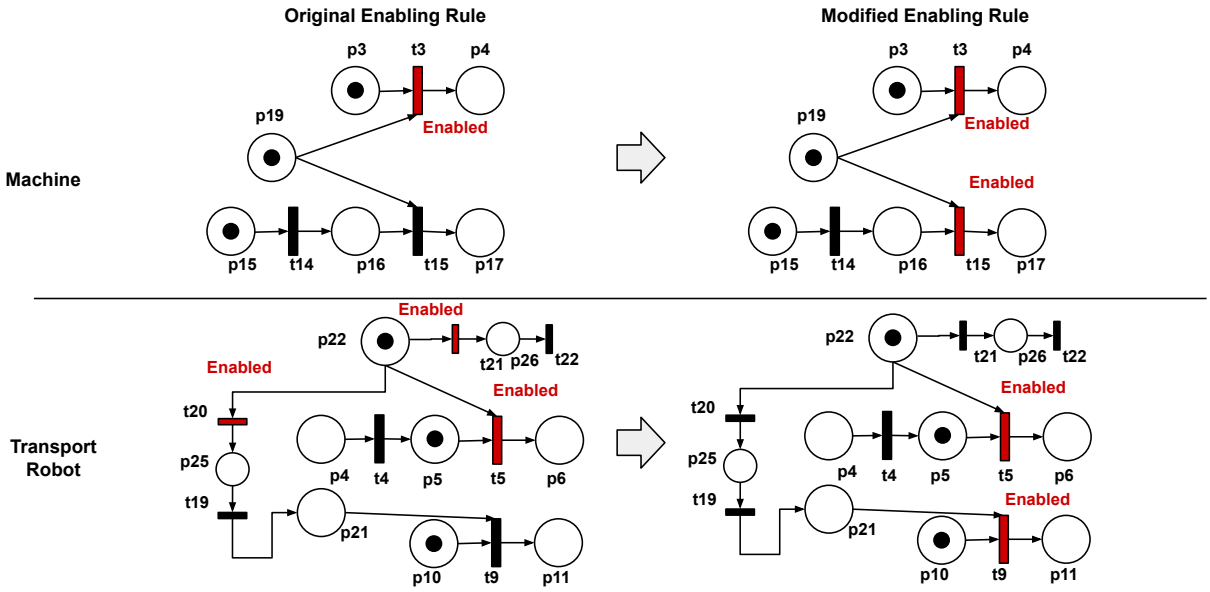

Figure 3: Modified enabling rules.

that a load trip transition is enabled regardless of the AGV location, and the empty trip needs to be fired if needed.

Definition 2 A transition of type t_{start_lt} is enabled if its p_{bf} type input place is marked, and at least one of places of type p_{ra} is marked. Also when p_{bf} is not marked, but its preceding p_{ps} place is marked, then t_{start_lt} is enabled.

Definition 3 A transition of type t_{start_et} is fired when enabled t_{start_lt} is fired, p_{ra} type input place is not marked, and the trip destination of t_{start_et} is p_{ra}. After t_{start_et} and its succeeding transition t_{end_et} are fired, then t_{start_lt} is fired automatically.

In Figure 3, there is one idle AGV located at machine 1 (p22 is marked). Jobs 1 and 2 can be transported from machines 1 to 2 and L/U stocker to machine 2, respectively (p5 and p10 are marked). Then t5 and t9 are considered as enabled transitions, and t5 can be fired immediately because the idle AGV is located at machine 1. For job 2, located at L/U stocker, the AGV needs to make the empty trip from machine 1 to L/U stocker. Thus, we should fire the empty trip transitions, t19 and t20, to fire t9. After firing t19 and t20, t9 is fired automatically.

In addition, when p5 is not marked, but p4 is marked (machine 2 processes job 1), we can still consider t5 as an enabled transition because the AGV can go ahead and wait for the process completion. In this case, t5 is fired automatically after firing t4.

3 METHODOLOGY

3.1 Monte carlo tree search

MCTS is an iterative algorithm that searches the state space and builds statistical evidence about the decisions available in particular states (Browne et al. 2012). MCTS repeats the iterations of four 4 steps: *Selection*, *Expansion*, *Simulation*, and *Backpropagation*. We explain each step in detail.

Selection: It starts from the root node and selects successive child nodes until a leaf node is reached. We use a general selection policy, the upper confidence bound (UCT), which attempts to balance between exploration and exploitation (Kocsis and Szepesvári 2006).

Table 1: Description of places and transitions of a TPN in Figure 2

Place	**Type**	**Description**
p1, p10	p_{bf}	L/U stocker for job 1,2
p2, p11	p_{lt}	Load trip from the L/U stocker to the M1(M2) input buffer for job 1,2
p3, p12	p_{bf}	M1(M2) input buffer for job 1,2
p4, p13	p_{ps}	M1(M2) processes a job 1,2
p5, p14	p_{bf}	M1(M2) output buffer for job 1,2
p6, p15	p_{lt}	Load trip from the M1(M2) output buffer to the M2(M1) input buffer for job 1,2
p7, p16	p_{bf}	M2(M1) input buffer for job 1,2
p8, p17	p_{ps}	M2(M1) processes a job 1,2
p9, p18	p_{bf}	M2(M1) output buffer for job 1,2
p19, p20	p_{ma}	M1,M2 available
p21, p22, p23	p_{ra}	L/U stocker, M1, M2 available for robot
p24, p25	p_{et}	Empty trip from the L/U(M1) to the M1(L/U stocker)
p26, p27	p_{et}	Empty trip from the M1(M2) to the M2(M1)
p28, p29	p_{et}	Empty trip from the L/U(M2) to the M2(L/U stocker)

Transition	**Type**	**Description**
t1, t9	t_{start_lt}	Start load trip from the L/U stocker to the M1(M2) input buffer for job 1,2
t2, t10	t_{end_lt}	End load trip from the L/U stocker to the M1(M2) input buffer for job 1,2
t3, t15	t_{start_ps}	M1 starts process for job 1,2
t4, t16	t_{end_ps}	M1 finishes process for job 1,2
t5, t13	t_{start_lt}	Start load trip from the M1(M2) to the M2(M1) input buffer for job 1,2
t6, t14	t_{end_ps}	End load trip from the M1(M2) to the M2(M1) input buffer for job 1,2
t7, t11	t_{start_ps}	M2(M1) starts process for job 1,2
t8, t12	t_{end_ps}	M2(M1) finishes process for job 1,2
t17, t18	t_{start_et}, t_{end_et}	Start, End empty trip from the L/U stocker to the M1
t19, t20	t_{start_et}, t_{end_et}	Start, End empty trip from the M1 to the L/U stocker
t21, t22	t_{start_et}, t_{end_et}	Start, End empty trip from the M1 to the M2
t23, t24	t_{start_et}, t_{end_et}	Start, End empty trip from the M2 to the L/U stocker
t25,t26	t_{start_et}, t_{end_et}	Start, End empty trip from the L/U stocker to the M2
t27, t28	t_{start_et}, t_{end_et}	Start, End empty trip from the M2 to the L/U stocker

$$selection\ policy = argmax_{action}(Q(s,a) + c\sqrt{\frac{ln[N(s)]}{N(s,a)}}).$$

where $Q(s,a)$, used for exploitation, denotes the average simulation result of action a in state s. $N(s)$ is the number of visits of state s, and $N(s,a)$ is the number of samples that action a is selected in state s. c is a coefficient. Each action means firing a load trip or process transition in our problem. If there are multiple idle robots, each pair of a load trip and robot are considered as an action.

Expansion: When a leaf node is reached, child nodes are expanded and added to the tree according to the available transitions. Child nodes are generated by firing enabled transitions that indicate the start load trip and start process in the leaf nodes TPN. If there are enabled start process transitions in the current TPN, they are first fired. Otherwise, enabled load trip transitions are first considered as child nodes.

Simulation: Random rollout of the problem is performed until the terminal state from the leaf node state. In our problem, it is impossible to reach the terminal state because the jobs randomly arrive. Also, rollout until the terminal state is not useful with the dynamic job arrivals. Hence, the time window parameter to limit the rollout times is used. A large time window parameter can allow for many future states in advance but the generated schedule may not be useful due to newly arriving jobs. We only consider jobs that have arrived and select enabled transitions randomly within the time window.

Backpropagation: To minimize the mean flow time, the return of the simulation, z, is set to the average job waiting time within the time window. The job waiting time is by adding the time during which the job

stays in the input and output buffers. The waiting times are normalized with the current time to have values between 0 and 1. The return, z, and the number of visits of nodes are propagated and updated through the path selected in the tree up to the root. After the given number of iterations, the most visited child node is selected to be the next transition to fire, and the TPN simulator steps over to the next state.

4 EXPERIMENTAL RESULTS

We compare our MCTS method with the combination of the machine and AGV dispatching rules. Tables 2 and 3 show the mean flow time of the proposed method and the dispatching rules. For the comparison, shortest processing time (SPT), longest processing time (LPT), most work remaining (MWKR), and first in first out (FIFO), are used for scheduling the machines. For the AGVs, minimum empty trip time job (MET), minimum load trip time job (MLT), minimum robot empty trip + load trip time job (MELT), and FIFO are used. The benchmark instances with four machines and 10 job types in Bilge and Ulusoy (1995) are used for the comparison. The arrival time interval is assumed to follow the Exponential distribution with $\lambda = 0.05$. The total number of newly arrival jobs is 20, and a job type is selected randomly. The MCTS exploration coefficient, c, is set to $\sqrt{2}$. The number of iterations in MCTS is 50, and the time window size is set to 100. It takes less than 10 s to determine the next action on average.

Table 2: Performance result: Mean flow time

	t/p	MCTS(W,P)	SPT/MET	SPT/MLT	SPT/MELT	SPT/FIFO	LPT/MET	LPT/MLT	LPT/MELT	LPT/FIFO
ex11	0.59	93.88	149.95	146.47	146.84	112.14	151.45	153.64	150.7	115.66
ex12	0.47	80.64	106.28	109.18	93.14	87.17	114.13	115.93	122.96	87.17
ex13	0.52	75.48	116.94	103.62	108.83	86.6	119.52	104.97	108.15	86.65
ex14	0.74	150.52	193.55	203.15	185.43	147.49	193.53	222.69	185.43	149.73
ex21	0.61	76.27	85.3	89.07	84.47	77.41	85.27	89.07	84.47	78.57
ex22	0.49	58.46	58.91	61.04	59.92	57.91	59.0	61.04	60.2	57.82
ex23	0.53	57.04	64.79	66.02	63.45	61.82	65.41	66.73	64.83	62.65
ex24	0.76	89.65	92.54	95.64	90.03	90.33	92.54	95.61	90.03	90.53
ex31	0.59	79.88	119.56	134.96	114.44	93.59	120.46	137.29	115.08	97.22
ex32	0.47	65.46	79.08	83.27	82.52	70.4	79.82	82.59	82.52	69.84
ex33	0.52	64.81	82.26	81.67	74.23	71.03	82.91	89.12	74.23	71.18
ex34	0.74	99.35	177.18	209.27	176.78	169.72	186.69	208.97	180.46	170.59
ex41	0.91	175.2	211.12	226.65	218.85	216.82	213.23	227.1	219.06	216.98
ex42	0.73	90.48	146.92	170.37	152.16	130.74	149.35	169.88	152.91	142.29
ex43	0.80	98.48	165.39	187.62	160.09	167.83	164.88	187.72	160.08	170.66
ex44	1.14	205.32	261.89	326.58	273.96	373.12	261.89	326.58	273.96	375.37
ex51	0.85	81.4	108.34	110.03	105.02	100.22	109.35	109.85	104.95	98.14
ex52	0.68	66.6	69.89	79.04	67.58	70.66	70.92	82.05	75.7	71.32
ex53	0.74	71.96	76.24	78.09	76.59	76.62	78.49	80.44	76.91	76.09
ex54	1.06	106.04	146.76	166.08	143.71	146.36	146.76	167.67	143.71	140.18
ex61	0.62	97.27	178.93	238.29	156.53	116.23	171.32	237.36	154.85	116.1
ex62	0.50	70.54	97.87	119.4	100.61	81.46	101.82	120.65	104.08	82.54
ex63	0.54	70.81	98.15	158.98	106.55	85.62	98.98	148.06	105.19	85.9
ex64	0.78	106.04	225.48	230.8	199.81	153.23	225.48	230.36	199.81	153.52
ex71	0.78	72.21	71.19	76.11	71.37	75.36	71.19	76.38	71.37	74.89
ex72	0.62	50.14	53.07	53.57	51.28	56.54	52.75	52.61	51.28	56.54
ex73	0.68	55.5	55.3	60.85	54.89	57.75	53.88	60.86	54.13	58.49
ex74	0.97	88.29	85.99	100.95	87.19	95.59	86.25	100.98	87.46	95.7
ex81	0.58	115.69	231.23	238.64	237.36	155.88	231.23	236.58	237.36	157.89
ex82	0.46	94.96	142.21	181.11	145.67	120.72	150.8	182.14	146.04	124.98
ex83	0.50	101.23	157.67	201.2	155.52	129.53	161.32	199.64	156.07	128.2
ex84	0.72	145.46	291.78	343.58	303.66	248.58	291.78	343.19	303.66	250.41
ex91	0.61	139.08	240.41	249.47	248.2	209.4	237.0	243.98	256.84	207.89
ex92	0.49	102.88	156.76	184.15	158.06	105.34	153.55	173.1	158.06	105.01
ex93	0.53	105.68	157.46	147.12	155.56	106.89	153.41	140.04	155.56	109.6
ex94	0.76	131.08	265.23	271.06	268.46	284.72	265.23	270.91	268.21	253.27
ex101	0.55	133.58	249.62	287.81	265.54	179.53	256.93	288.23	268.48	183.96
ex102	0.44	115.31	219.3	231.15	210.27	138.34	220.81	232.33	214.05	144.86
ex103	0.49	115.12	219.76	235.82	217.13	141.95	233.96	237.46	223.25	170.09
ex104	0.69	197.88	305.15	324.32	295.56	253.71	305.21	327.79	295.63	265.22
	Average	**99.89**	150.39	166.56	149.18	130.11	151.71	167.04	150.94	131.34

The proposed algorithm is tested in two versions, MCTS(W,P), which considers the proposed enabling rules from Definitions 1 to 3, and MCTS(P), which only considers rules in Definitions 2 and 3. In the table, t/p indicates the ratio of the average transportation time to the average processing time. We implement the MCTS algorithm and dispatching rules with Python and run all experiments on a PC with an Intel i9-9900 CPU @ 3.1 GHz and 32GB of RAM.

It is observed that both proposed methods show better performance than the dispatching rules in most cases. MCTS(W,P) performs better than MCTS(P) on average, confirming that it might be beneficial to wait for not yet enabled transitions in some cases.

Table 3: Performance result(continued): Mean flow time

	t/p	MCTS(P)	MWKR/MET	MWKR/MLT	MWKR/MELT	MWKR/FIFO	FIFO/MET	FIFO/MLT	FIFO/MELT	FIFO/FIFO
ex11	0.59	102.0	147.5	146.47	146.84	112.14	153.32	146.55	150.7	111.31
ex12	0.47	78.44	111.05	111.88	94.56	83.75	105.61	107.12	107.62	84.64
ex13	0.52	78.56	118.16	105.0	107.98	86.5	116.66	104.14	109.88	86.59
ex14	0.74	137.52	193.53	211.59	185.43	148.19	193.55	202.45	185.43	147.49
ex21	0.61	78.65	85.27	89.07	84.47	78.57	85.12	89.07	84.47	77.41
ex22	0.49	57.35	59.0	61.04	60.2	57.91	58.91	61.04	59.92	57.82
ex23	0.53	57.04	65.41	66.73	64.83	62.65	64.73	65.99	63.54	61.78
ex24	0.76	89.65	92.54	95.61	90.03	90.53	92.54	95.54	90.03	90.33
ex31	0.59	80.58	119.56	134.99	114.44	92.73	119.08	135.53	114.47	93.02
ex32	0.47	64.62	78.02	83.15	81.61	72.1	80.04	82.79	82.44	69.59
ex33	0.52	64.81	83.27	83.6	74.06	70.96	82.88	87.16	74.23	71.25
ex34	0.74	100.77	177.12	209.22	176.78	169.72	186.69	209.0	180.46	169.72
ex41	0.91	169.76	211.49	222.82	219.06	216.3	211.23	225.32	218.85	211.23
ex42	0.73	91.92	149.25	170.94	155.62	134.92	146.92	169.36	152.91	129.04
ex43	0.80	101.04	164.88	185.67	160.08	176.93	165.24	189.89	160.33	145.7
ex44	1.14	200.88	261.89	326.58	273.96	373.94	261.89	326.58	273.96	373.52
ex51	0.85	81.36	109.41	105.66	105.58	102.95	108.34	108.37	104.4	97.24
ex52	0.68	63.92	70.92	80.82	67.94	71.01	70.85	80.82	75.04	69.1
ex53	0.74	70.44	76.37	80.65	77.62	75.46	78.87	78.47	76.59	75.46
ex54	1.06	109.84	146.76	167.67	143.71	140.18	146.76	166.08	143.71	146.21
ex61	0.62	97.27	173.06	237.13	154.85	116.1	171.32	237.06	156.53	115.89
ex62	0.50	72.19	101.82	120.65	104.08	82.73	95.59	120.59	100.58	82.37
ex63	0.54	75.5	98.98	148.06	105.19	85.9	97.72	158.43	106.27	85.67
ex64	0.78	106.04	225.48	230.36	199.81	153.52	225.48	230.3	199.81	153.02
ex71	0.78	73.43	71.19	76.29	71.37	75.18	71.19	75.99	71.37	74.89
ex72	0.62	50.39	52.75	52.61	51.28	56.54	52.75	52.61	51.28	56.5
ex73	0.68	53.5	53.88	60.86	54.13	58.49	53.92	60.12	54.13	57.11
ex74	0.97	87.32	86.25	100.98	87.31	95.7	86.25	100.98	87.46	95.59
ex81	0.58	112.65	231.23	238.19	237.36	166.41	231.23	239.22	237.36	147.03
ex82	0.46	94.65	150.34	181.02	148.09	127.3	141.41	182.26	145.65	124.62
ex83	0.50	102.23	161.32	203.11	156.07	130.86	159.16	204.81	155.94	132.03
ex84	0.72	143.5	291.78	343.19	303.66	243.25	291.78	343.28	303.66	243.98
ex91	0.61	163.44	237.01	243.98	256.84	210.1	236.97	246.75	248.2	205.94
ex92	0.49	90.2	153.55	183.21	158.06	105.01	153.56	172.15	158.06	102.85
ex93	0.53	121.56	153.43	140.43	155.56	109.6	152.49	146.41	155.56	107.22
ex94	0.76	164.92	265.23	271.06	268.46	252.85	265.23	271.35	268.21	251.57
ex101	0.55	131.96	255.68	287.8	273.18	196.05	250.65	287.8	253.02	177.81
ex102	0.44	119.08	216.38	236.04	208.49	145.87	219.32	236.45	211.36	140.79
ex103	0.49	118.35	222.73	236.06	216.87	156.08	219.64	235.44	216.06	143.91
ex104	0.69	204.92	305.21	327.49	295.63	255.48	305.15	324.68	295.56	256.37
	Average	101.56	150.72	166.44	149.78	131.01	150.25	166.45	149.63	128.09

More experiments are performed with larger instances, named FT10, which have 10 machines and 10 jobs by modifying them for dynamic job shops with AGVs (Muth and Thompson 1963). We generate two layouts for dynamic job shops where the ratio of the average transportation time to the processing time is 0.2 and 0.4, respectively. There are five transport robots, and the job arrival interval is set to $\lambda = 0.01$. The total number of jobs is 30. The time window size of MCTS is 100, and the number of iterations is set to 200. It is observed that the proposed MCTS method provides better solutions compared to the dispatching rules as in Table 4.

Table 4: Performance result with larger instance

Instance	t/p	MCTS(W,P)	SPT/MET	SPT/MLT	SPT/MELT	SPT/FIFO	LPT/MET	LPT/MLT	LPT/MELT	LPT/FIFO
FT10	0.2	1239.60	1390.43	1766.64	1796.69	1089.36	1504.72	2131.34	1936.29	1244.19
	0.4	**1216.13**	1725.18	2498.42	2322.61	1430.94	1782.05	2699.03	2225.68	1521.36
		MCTS(P)	MWKR/MET	MWKR/MLT	MWKR/MELT	MWKR/FIFO	FIFO/MET	FIFO/MLT	FIFO/MELT	FIFO/FIFO
		1057.73	1474.37	1949.54	1960.87	1181.55	1428.88	1946.17	1853.86	1141.17
		1344.23	1760.64	2704.18	2315.56	1510.05	1727.47	2627.52	2119.58	1436.15

5 CONCLUSION

We have addressed the dynamic job shop scheduling problem with AGVs by applying the MCTS-based algorithm. We have modeled the problem with a TPN and modified the transition enabling rules to consider the machine or robot waiting instead of selecting only the available jobs. We have further restricted empty trip transitions from firing infinitely. We have shown that the proposed method can provide better schedules in terms of the average flow time compared to other dispatching rules in most cases.
However, the computation time of our algorithm is still long when the instance size becomes large. Therefore, we are planning to replace the return value of simulation with a parameterized value function using neural networks and reinforcement learning to shorten the time-consuming simulation step of MCTS.

ACKNOWLEDGMENTS

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education [2022R1A2C400197811].

REFERENCES

Bilge, Ü., and G. Ulusoy. 1995. "A Time Window Approach to Simultaneous Scheduling of Machines and Material Handling System in an FMS". *Operations Research* 43(6):1058–1070.

Browne, C. B., E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. 2012. "A Survey of Monte Carlo Tree Search Methods". *IEEE Transactions on Computational Intelligence and AI in Games* 4(1):1–43.

Dabbas, R. M., H.-N. Chen, J. W. Fowler, and D. Shunk. 2001. "A Combined Dispatching Criteria Approach to Scheduling Semiconductor Manufacturing Systems". *Computers & Industrial Engineering* 39(3):307–324.

Deroussi, L., M. Gourgand, and N. Tchernev. 2008. "A Simple Metaheuristic Approach to the Simultaneous Scheduling of Machines and Automated Guided Vehicles". *International Journal of Production Research* 46(8):2143–2164.

Dominic, P. D. D., S. Kaliyamoorthy, and M. S. Kumar. 2004. "Efficient Dispatching Rules for Dynamic Job Shop Scheduling". *The International Journal of Advanced Manufacturing Technology* 24(1):70–75.

Ham, A. 2021. "Transfer-robot Task Scheduling in Job Shop". *International Journal of Production Research* 59(3):813–823.

Kocsis, L., and C. Szepesvári. 2006. "Bandit Based Monte-carlo Planning". In *Machine Learning: ECML 2006*, edited by J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, 282–293. Berlin, Heidelberg: Springer Berlin Heidelberg.

Kundakc, N., and O. Kulak. 2016. "Hybrid Genetic Algorithms for Minimizing Makespan in Dynamic Job Shop Scheduling Problem". *Computers & Industrial Engineering* 96:31–51.

Lacomme, P., M. Larabi, and N. Tchernev. 2013. "Job-shop Based Framework for Simultaneous Scheduling of Machines and Automated Guided Vehicles". *International Journal of Production Economics* 143(1):24–34.

Liu, R., R. Piplani, and C. Toro. 2022. "Deep Reinforcement Learning for Dynamic Scheduling of a Flexible Job Shop". *International Journal of Production Research* 0(0):1–21.

Muth, J. F., and G. L. Thompson. 1963. *Industrial Scheduling*. Prentice-Hall.

Reddy, B. S. P., and C. S. P. Rao. 2006. "A Hybrid Multi-objective GA for Simultaneous Scheduling of Machines and AGVs in FMS". *The International Journal of Advanced Manufacturing Technology* 31(5):602–613.

Wang, J. 2012. *Timed Petri Nets: Theory and Application*, Volume 9. Springer Science & Business Media.

Xiong, H., H. Fan, G. Jiang, and G. Li. 2017. "A Simulation-based Study of Dispatching Rules in a Dynamic Job Shop Scheduling Problem with Batch Release and Extended Technical Precedence Constraints". *European Journal of Operational Research* 257(1):13–24.

Zheng, Y., Y. Xiao, and Y. Seo. 2014. "A Tabu Search Algorithm for Simultaneous Machine/AGV Scheduling Problem". *International Journal of Production Research* 52(19):5748–5763.

AUTHOR BIOGRAPHIES

DUYEON KIM is a Ph.D. candidate in the Department of Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST). He is interested in scheduling methodologies and applications and operations management. His email address is duyeon@kast.ac.kr.

HYUN-JUNG KIM is an Associate Professor with the Department of Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST). She received B.S., M.S., and Ph.D. in industrial and systems engineering from KAIST. Her research interests include discrete event systems modeling, scheduling, and control. Her email address is hyunjungkim@kaist.ac.kr.