Appendix

In this appendix, we include the following items, described in the subsequent subsections respectively.

- Code, datsets and model weights.
- Formulation of Markov decision process for solution construction.
- Illustration of graph representation for partial solutions in Figure 2.
- REINFORCE algorithm in our approach.
- Experimental settings.
- Additional Experiments.

A Code, Datasets and Model Weights

Our code, datasets and pre-trained weights are in the supplementary folder `ResidualScheduling`. The code includes the following: First, setup the virtual environment, Then, install required packages in the environment. Finally, execute the code with the commands in the `README` file. All hyperparameters are also listed in `README` file.

We collect the following dataset for evaluation. For JSP dataset, we collect 13 collections generated for the experiments shown in Figure 4, and other seven JSP benchmarks, TA, ABZ, FT, ORB, YN, SWV and LA. For FJSP dataset, we collect two FJSP benchmarks, MK and LA, and the instances generated from the work (Song et al., 2023).

There are four weights according to the final models for RS and RS+op with JSP and FJSP cases, denoted as `RS_JSP`, `RS+op_JSP`, `RS_FJSP`, and `RS+op_FJSP`.

B Formulation of Markov Decision Process for Solution Construction

In Subsection 2.2, an approach based on solution construction can be also viewed as the so-called *Markov decision process* (*MDP*). An MDP is a stochastic decision-making process widely used in reinforcement learning, generally defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$, where $\mathcal{S}$ is the finite set of the states, $\mathcal{A}$ is the set of the available actions, $\mathcal{R}$ is the reward function and $\mathcal{P}$ is the transition probability function. The objective is to find a policy that maximizes the agent's cumulative rewards[1].

Following the above MDP definition, we formulate the process of solution construction for JSP and FJSP problems as follows.

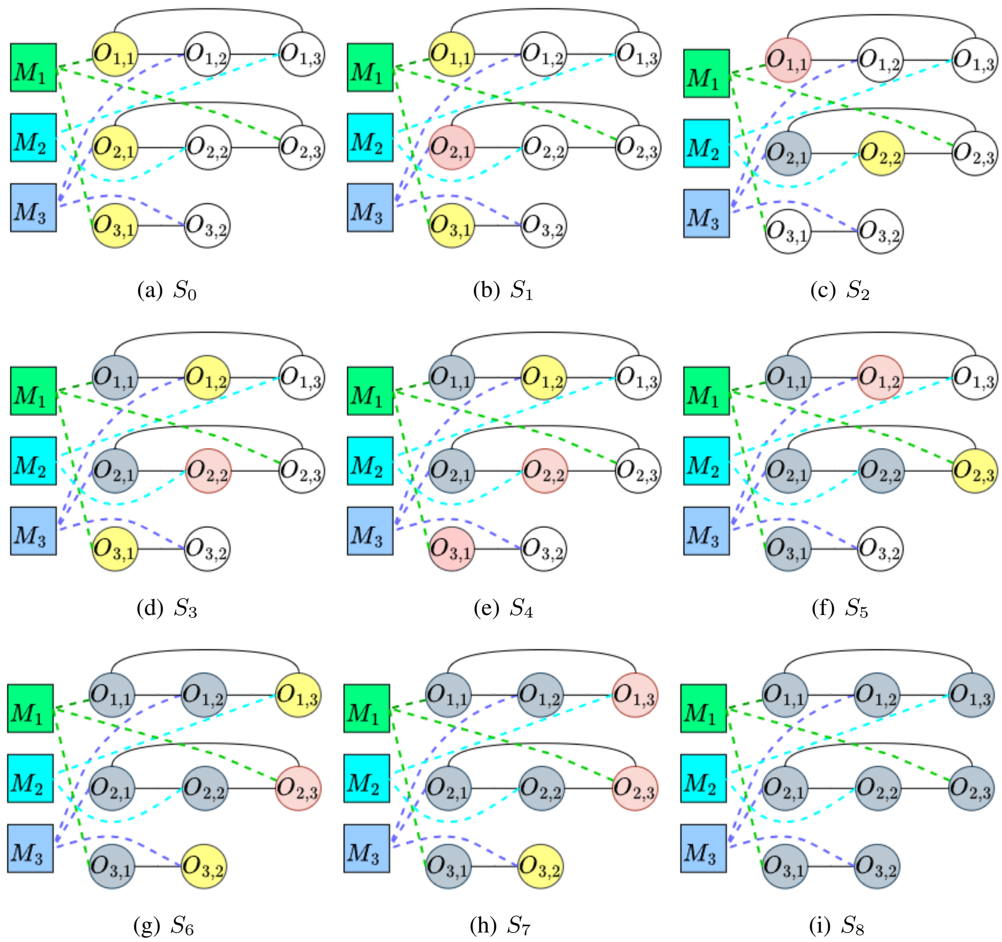
- $\mathcal{S}$ specifies states each representing an instance associated with the information of partial solutions S like those in Figure 2.
- $\mathcal{A}$ specifies actions each of which selects machine-operation pairs (M, O) to dispatch the operation O on the machine M on given states.
- $\mathcal{R}$ specifies a reward to indicate a negative of the additional processing time for the dispatched action.
- $\mathcal{P}$ specifies how states are transited from one partial solution S to the next S' after an action, as described in Subsection 2.2 accordingly.

An episode starts from the initial partial solution and repeats transitions till the end when the partial solution is a solution. The cumulative reward is the negative of makespan $T^{(mksp)}$, i.e., the total complete time. The objective is to maximize the cumulative reward, i.e., minimize the makespan. In the past, many methods for JSP and FJSP based on reinforcement learning such as (Park et al., 2020; Luo, 2020; Zhang et al., 2020; Park et al., 2021b,a) followed this MDP formulation.

[1]Discount factor γ is not included in the tuple, since it is always one and thus not used in this paper.

C Illustration of Graph Representation for Partial Solutions

In Figure 3 (a), we illustrate graph representation of S_3. In this section, Figure 5 illustrates the step-by-step graph representations respectively corresponding to all the partial solutions in Figure 2 in Subsection 3.2. Again, the gray nodes are *completed*, the red *ongoing*, the yellow *ready* and the white *unready*.



(a) S_0 (b) S_1 (c) S_2

(d) S_3 (e) S_4 (f) S_5

(g) S_6 (h) S_7 (i) S_8

Figure 5: (a)-(i) are corresponding to partial solutions in Figure 2 (a)-(i), respectively.

D REINFORCE Algorithm in Our Approach

Our residual scheduling approach basically follows REINFORCE (Sutton and Barto, 2018) to update our model by policy gradient with a normalized advantage makespan based on a baseline policy and an entropy bonus to ensure sufficient exploration like PPO. Algorithm 1 (next page) shows our REINFORCE training process with the normalized advantage makespan described in Section 3.4. The model policy π_θ is parameterized by θ initialized at random, and a baseline policy π_b is given, which is a lightweight PDR, MWKR, in this paper. The learning rate $\alpha \in (0, 1)$ is initialized to 10^{-4} and decayed by a factor of 0.99 every $P = 1000$ episodes in this paper.

E Experimental Settings

In this section, we first describe settings of our model in Appendix E.1 and then list the details of JSP and FJSP open benchmarks used in our experiments in Appendix E.2.

Algorithm 1 REINFORCE with a normalized advantage for JSP and FJSP

1: **Input:** Initial policy π_θ, learning rate $\alpha \in (0, 1)$, entropy bonus coefficient c, a baseline policy π_b;
2: **for** Episode $e = 0, 1, \cdots$ **do**
3: Use policy π_θ to rollout an episode: $\{S_1, a_1, S_2, a_2, \cdots, S_T\}$
4: **for** $t = 0, 1, 2, \cdots T$ **do**
5: Use the baseline policy π_b to rollout and obtain $T_{\pi_b}^{(mksp)}(S_t, a_t)$
6: Calculate $T_{\pi_\theta}^{(mksp)}(S_t, a_t)$
7: Calculate normalized advantage $A_{\pi_\theta}(S_t, a_t)$ following Equation (5)
8: Update policy $\theta = \theta + \alpha \nabla_\theta (log\pi_\theta A_{\pi_\theta}(S_t, a_t) + cH_{\pi_\theta}(S_t))$
9: **end for**
10: Update learning rate every $P = 1000$ episodes
11: **end for**

E.1 Model Settings

In this subsection, we first list the settings of our model in Table 5. Particularly, the entropy bonus coefficient c is used for the weight of entropy in line 8 of Algorithm 1, and the initial learning rate α is used in lines 1 and 8 of Algorithm 1.

Table 5: Model hyperparameters.

Item	Value	Description
HGIN layers	3	Layer number of GNN.
FC layers of MLP	2	Layer number of MLP.
Hidden dimension	256	Hidden vector dimension for GNN and policy network.
Episodes	1,000,000	Training episodes
Entropy bonus coefficient (c)	10^{-2}	The weight of entropy bonus in the loss function.
Initial learning rate (α)	10^{-4}	Initialized with 10^{-4} and then decayed with 0.99 rate for every 1000 episodes
Optimizer	Adam	Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$

In practice, one model is trained for each of (N, M), (6,6), (10,10) and (15,15), and each of baseline policies, MWKR, MOR, SPT and FIFO. Each of models is trained with 1000 iterations, for each of which model parameters are updated with 1000 episodes (one million of episodes in total). For each episode, one scheduling instance is generated for training by the procedure subject to (N, M) as in Subsection 4.1.

For JSP, we also generate a validation dataset of the three collections, 20 $\times$ 20, 50 $\times$ 20 and 100 $\times$ 20, each with 10 instances. Among the 1000 sets of model parameters, the one performing best for the validation dataset is chosen for testing in the rest of experiments. Table 6 shows the average makespans of the four models trained with different baselines respectively and with (10,10) on the validation dataset. The one with MWKR clearly outperforms other baselines on the validation dataset. Thus, MWKR is chosen as the baseline in this paper. Similarly, Table 7 shows the average makespans of three models trained with different training sizes (6,6), (10,10) and (15,15) respectively and with MWKR on the validation dataset. The one with (10,10) performs best, except for that the one with (15,15) performs slightly better for the collection of 100 $\times$ 20. Thus, this paper still chooses (10,10) since it performs well in general and has less training costs when compared to (15,15).

Table 6: Average makespans of models with different baseline policies π_b on JSP validation dataset.

π_b	20×20	50×20	100×20
MWKR	**1803.1**	**3147.3**	**5676.0**
MOR	1831.7	3229.8	5728.3
SPT	1813.8	3201.7	5718.7
FIFO	1826.4	3177.6	5692.9

Table 7: Average makespans of models with different (N, M) on JSP validation dataset.

Method	20×20	50×20	100×20
RS (6,6)	1822.8	3242.7	5784.4
RS (10,10)	**1803.1**	**3147.3**	5676.0
RS (15,15)	1820.7	3170.6	**5670.4**

For FJSP, we simply use the validation set used by (Song et al., 2023). Table 8 shows the average makespans of three models trained with different training sizes (6,6), (10,10) and (15,15) respectively and with MWKR on the validation dataset. The one with (10,10) outperforms others for all the collections in Song et al. (2023). Hence, (10,10) is also chosen in FJSP experiments in this paper.

Table 8: Average makespans of models with different (N, M) on FJSP validation dataset.

Method	10 × 5	15 × 10	20 × 5	20 × 10
RS (6,6)	106.25	164.98	209.77	210.57
RS (10,10)	**106.00**	**163.88**	**209.07**	**210.18**
RS (15,15)	106.15	163.95	209.57	210.47

E.2 Open Benchmarks

Table 9 lists the information about JSP and FJSP open benchmarks used in this paper. For JSP, TA benchmark has most instances, including 8 categories, 15 × 15, 20 × 15, 20 × 20, 30 × 15, 30 × 20, 50 × 15, 50 × 20 and 100 × 20. For each category, there are 10 instances generated by their procedure. For other JSP benchmarks, they cover different problem sizes and data distributions as in the table.

For FJSP, two popular open benchmarks are MK, also known as Brandimarte instances, and LA, also known as Hurink instances (to distinguish LA benchmark for JSP). Hurink instances are categories into three types, edata, rdata and vdata, according to different distribution for assignable machines, denoted by LA(edata), LA(rdata) and LA(vdata) respectively, as described in (Hurink et al., 1994).

Table 9: Open benchmarks.

Type	Benchmark	Instances	Min size	Max size
JSP	TA (Taillard, 1993)	80	15×15	100×20
	ABZ (Adams et al., 1988)	5	10×10	20×15
	FT (Muth and Thompson, 1963)	3	6×6	20×5
	ORB (Applegate and Cook, 1991)	10	10×10	10×10
	YN (Yamada and Nakano, 1992)	4	20×20	20×20
	SWV (Storer et al., 1992)	20	20×10	50×10
	LA (Lawrence, 1984)	40	10×5	30×10
FJSP	MK (Brandimarte, 1993)	10	10×6	20×15
	LA (Hurink et al., 1994)	120	10×5	30×10

F Additional Experiments

This section shows additional experiments conducted to support residual scheduling as follows.

- Appendix F.1 shows the comparisons between RS and other construction heuristics for other six JSP benchmarks.
- Appendix F.2 shows the comparisons between RS and other methods for FJSP instances generated by Song et al. (2023).
- Appendix F.3 shows the computation times of construction heuristics for comparison.
- Appendix F.4 shows the makespans of all instances in more detail.

F.1 Other JSP Benchmarks

Table 10 shows the average makespan gaps for other six JSP open benchmarks, where the best performances (the smallest gaps) are marked in bold. Similar to the results in TA benchmark, in general, RS still performs the best, and generally outperforms the other methods except for that RS has slightly higher gaps than RS+op for the ORB benchmark. In fact, RS+op also generally outperforms the rest of methods, except for that RS+op is slightly higeher than but very close to SchN for ABZ and YN benchmarks. It is also concluded that RS generally performs better than other construction heuristics.

Table 10: Average makespan gaps for other benchmarks.

Dataset	ABZ	FT	ORB	YN	SWV	LA
RS	**0.116**	**0.089**	0.165	**0.166**	**0.122**	**0.075**
RS+op	0.156	0.162	**0.159**	0.186	0.202	0.097
MWKR	0.166	0.196	0.251	0.197	0.284	0.126
MOR	0.180	0.232	0.290	0.228	0.358	0.138
SPT	0.251	0.280	0.262	0.306	0.230	0.199
FIFO	0.247	0.288	0.297	0.258	0.373	0.188
Park	0.214	0.222	0.218	0.247	0.228	0.141
SchN	0.147	0.184	0.199	0.184	0.288	0.099

F.2 FJSP Dataset from Song et al.

This subsection also compares our model with others on the FJSP instances generated by Song et al. (2023), as listed in Tables 11 and 12, where there are 100 FJSP instances for each problem size. The two tables also show the makespan results for RS, RS+100, DRL-G, DRL+100, MOR, MWKR, SPT, FIFO and OR-Tools, since they only provided the average makespans, not makespan for each instance. Note that the average makespan gap is not able to be calculated with Equation (6) without makespans from individual instances. Besides, for OR-Tools, their results are taken 1800 seconds to find the best-effort solutions. From Table 11, RS outperforms all the other construction heuristics, including DRL-G, for all problem sizes. From Table 12, RS+100 clearly outperforms DRL+100, DRL-G and RS by significant margins, like Table 4 in Subsection 5.

Table 11: Average makespans for the FJSP dataset by Song et al.

Method	10×5	15×10	20×05	20×10	30×10	40×10
OR-Tools	96.59	145.42	188.45	197.24	294.1	397.36
RS	**109.06**	**165.45**	**207.27**	**213.12**	**311.16**	**413.46**
DRL-G	111.67	166.92	211.22	215.78	313.04	416.18
MOR	116.69	173.40	217.17	221.86	320.18	425.19
MWKR	115.29	169.18	216.98	220.85	319.89	425.70
SPT	129.06	198.20	229.89	254.59	347.40	443.30
FIFO	119.62	185.08	216.13	234.21	328.50	427.22

Table 12: Average makespans for the FJSP dataset by Song et al.

Method	10×5	15×10	20×05	20×10	30×10	40×10
OR-Tools	96.59	145.42	188.45	197.24	294.1	397.36
RS	109.06	165.45	207.27	213.12	311.16	413.46
RS+100	**103.31**	**157.30**	**201.73**	**207.86**	**305.27**	**407.28**
DRL-G	111.67	166.92	211.22	215.78	313.04	416.18
DRL+100	105.61	160.36	207.50	214.87	312.20	415.14

F.3 Computation Time

Table 13 and Table 14 show the computation time taken by methods for TA JSP benchmark and FJSP benchmarks. The four heuristics, MWKR, MOR, SPT and FIFO, take almost the same time to obtain the scheduling solutions, so they are merged into the PDRs category. For TA benchmark, the computation times for L2D and SchN were reported by their works (Zhang et al., 2020) and (Park et al., 2021a), respectively, which also reported to use a machine with AMD Ryzen 3600 CPU and a single Nvidia GeForce 2070S GPU. To our knowledge, there were no time records available for the work (Park et al., 2021b), marked as "-". We observe that the time for RS is much less than SchN, and comparable to other DRL-based construction heuristics.

Table 13: Average computation times for TA JSP instances.

Size	15×15	20×15	20×20	30×15	30×20	50×15	50×20	100×20	*Avg.*
RS	0.47s	0.83s	0.91s	1.93s	2.21s	5.30s	6.96s	27.32s	5.74s
PDRs	0.05s	0.08s	0.12s	0.16s	0.24s	0.41s	0.60s	2.20s	0.48s
L2D	0.40s	0.60s	1.10s	1.30s	1.50s	2.20s	3.6s	28.20s	4.86s
Park	-	-	-	-	-	-	-	-	-
SchN	3.50s	6.60s	11.00s	17.10s	28.30s	52.50s	96.00s	444.0s	82.38s

Table 14: Average computation times for FJSP open benchmarks.

Method	MK	LA(rdata)	LA(edata)	LA(vdata)
RS	0.85s	0.63s	0.80s	0.98s
PDRs	0.41s	0.47s	0.47s	0.46s
DRL-G	0.90s	0.97s	1.04s	0.96s

F.4 Experiment Results in More Detail

This section presents experiment results in more detail by listing makespans for all instances. First, Table 15 shows the average JSP makespan gaps in Figure 4, and Tables 16 and 17 show in more detail the makespans of all individual JSP instances obtained by RS, L2D and OR-tools. For OR-tools, the time limitation is set to half a day, i.e., it is set to the best-effort if OR-tools exceed the time limitation, where the cases are marked with †. For all 60 JSP instances larger than and equal to 100×15, RS reaches all optimum (marked in bold) except for four 100×20 instances and one 150×20 instance in Table 17.

Second, Tables 18 and 19 show the makespans of FJSP MK instances and LA instances, respectively. OPT indicates the best-known results from (Behnke and Geiger, 2012). From the tables, RS+100 obtains makespans the same as (marked in bold) or close to those in OPT for many instances.

Third, Table 20 and Table 21 show the makespans of all TA instances for Table 2. Park and SchN represent the work Park et al. (2021b) and ScheduleNet; and OPT is the best-effort results for OR-tools in 6000 seconds. To indicate the best performance of the construction heuristics, we mark in bold the makespan that is closest to that of OPT. We observe that RS performs best for most instances from Ta30 to Ta80, and outperforms others for nearly all the instances with sizes larger than and equal to 30×15.

Table 15: Average makespan gaps in Figure 4.

Method	OR-Tools	RS	L2D
15×15	0	0.169	0.280
20×15	0†	0.172	0.273
20×20	0†	0.179	0.278
30×15	0†	0.119	0.252
30×20	0†	0.159	0.268
50×15	0	0.035	0.146
50×20	0†	0.082	0.205
100×15	0	0.000	0.045
100×20	0	0.008	0.080
150×15	0	0.000	0.030
150×20	0	0.00005	0.047
200×15	0	0.000	0.013
200×20	0	0.000	0.041

†means the best-effort solution from OR-Tools with a half day computation.

Table 16: Part 1 of makespans of all JSP instances of Table 15.

Instance	$n \times m$	L2D	RS	OR-Tools	$n \times m$	L2D	RS	OR-Tools
01	15×15	1528	1386	1181	-	-	-	-
02	15×15	1422	1360	1172	-	-	-	-
03	15×15	1500	1494	1243	-	-	-	-
04	15×15	1651	1444	1230	-	-	-	-
05	15×15	1663	1502	1302	-	-	-	-
06	15×15	1479	1399	1237	-	-	-	-
07	15×15	1590	1363	1139	-	-	-	-
08	15×15	1432	1343	1148	-	-	-	-
09	15×15	1591	1371	1170	-	-	-	-
10	15×15	1492	1352	1168	-	-	-	-
01	20×15	1754	1438	1290	20×20	1864	1775	1483
02	20×15	1789	1585	1386	20×20	2116	1969	1669
03	20×15	1868	1660	1424	20×20	2225	1963	1697
04	20×15	1896	1714	1437	20×20	1989	1813	1510
05	20×15	1767	1606	1383	20×20	1749	1825	1539
06	20×15	1564	1496	1264	20×20	1950	1817	1524
07	20×15	1603	1467	1315	20×20	2065	1820	1614
08	20×15	1652	1648	1316	20×20	2103	1718	1512
09	20×15	1611	1576	1319	20×20	1914	1778	1482
10	20×15	1660	1605	1335	20×20	2001	1947	1597
01	30×15	2154	1949	1691	30×20	2390	2136	1939
02	30×15	2473	2108	1822	30×20	2501	2398	2056
03	30×15	2430	2129	2078	30×20	2269	2097	1835
04	30×15	2099	1922	1600	30×20	2470	2246	1936
05	30×15	2178	2038	1828	30×20	2420	2186	1958
06	30×15	2158	1955	1778	30×20	2414	2081	1824
07	30×15	2193	1993	1863	30×20	2639	2273	1921
08	30×15	2219	1996	1801	30×20	2233	2225	1804
09	30×15	2354	1950	1750	30×20	2569	2350	1973
10	30×15	1938	1785	1550	30×20	2491	2312	2001
01	50×15	3238	2886	2781	50×20	3732	3164	3047
02	50×15	3571	3119	2954	50×20	3477	3151	2760
03	50×15	3114	2848	2787	50×20	3381	3183	2969
04	50×15	3111	2859	2845	50×20	3446	3022	2847
05	50×15	3074	**2924**	2924	50×20	3267	2991	2775
06	50×15	3113	2801	2588	50×20	3565	3250	2962
07	50×15	3230	**3036**	3036	50×20	3678	3157	2920
08	50×15	3448	3030	2969	50×20	3723	3322	2919
09	50×15	3437	3164	2810	50×20	3538	3197	2873
10	50×15	3397	2879	2873	50×20	3456	**3241**	3241
01	100×15	5945	**5493**	5493	100×20	6215	**5505**	5505
02	100×15	5683	**5517**	5517	100×20	5978	**5663**	5663
03	100×15	5457	**5212**	5212	100×20	6136	5465	5443
04	100×15	6088	**5802**	5802	100×20	6193	5593	5511
05	100×15	5701	**5568**	5568	100×20	6147	**6066**	6066
06	100×15	6094	**5821**	5821	100×20	6176	**5699**	5699
07	100×15	6019	**5756**	5756	100×20	6054	5858	5574
08	100×15	5898	**5876**	5876	100×20	6040	5564	5505
09	100×15	5545	**5367**	5367	100×20	5981	**5886**	5886
10	100×15	5784	**5303**	5303	100×20	6089	**5689**	5689

Table 17: Part 2 of makespans of all JSP instances of Table 15.

Instance	$n \times m$	L2D	RS	OR-Tools	$n \times m$	L2D	RS	OR-Tools
01	150×15	8902	**8470**	8470	150×20	8630	**8146**	8146
02	150×15	8077	**7871**	7871	150×20	8258	**8139**	8139
03	150×15	8281	**7964**	7964	150×20	8693	**8178**	8178
04	150×15	8268	**8215**	8215	150×20	8717	**8219**	8219
05	150×15	8423	**7974**	7974	150×20	8720	**8663**	8663
06	150×15	8544	**8506**	8506	150×20	8577	**8086**	8086
07	150×15	8703	**8227**	8227	150×20	8622	7966	7962
08	150×15	8265	**7940**	7940	150×20	8395	**8101**	8101
09	150×15	8522	**8472**	8472	150×20	8308	**8000**	8000
10	150×15	8099	**7937**	7937	150×20	8576	**8215**	8215
01	200×15	11018	**10696**	10696	200×20	10687	**10545**	10545
02	200×15	10770	**10576**	10576	200×20	11453	**10464**	10464
03	200×15	11030	**10952**	10952	200×20	12069	**11040**	11040
04	200×15	**10881**	**10881**	10881	200×20	11044	**10644**	10644
05	200×15	10706	**10701**	10701	200×20	11070	**10793**	10793
06	200×15	10540	**10510**	10510	200×20	11029	**10427**	10427
07	200×15	10810	**10345**	10345	200×20	10972	**10699**	10699
08	200×15	**10355**	**10355**	10355	200×20	11033	**10710**	10710
09	200×15	10983	**10668**	10668	200×20	10988	**10811**	10811
10	200×15	11036	**10969**	10969	200×20	10787	**10666**	10666

Table 18: Makespans of all MK instances (FJSP).

Instance	$n \times m$	RS	RS+100	OPT
mk01	10×6	42	41	39
mk02	10×6	36	35	26
mk03	15×8	**204**	**204**	204
mk04	15×8	75	67	60
mk05	15×4	185	180	172
mk06	10×15	97	81	58
mk07	20×5	216	188	139
mk08	20×10	**523**	**523**	523
mk09	20×10	316	311	307
mk10	20×15	252	240	197

Table 19: Makespans of all LA instances (FJSP).

		edata			rdata			vdata		
Instance	$n \times m$	RS	RS+100	OPT	RS	RS+100	OPT	RS	RS+100	OPT
la01	10×5	721	621	609	595	586	571	587	575	570
la02	10×5	799	747	655	567	549	530	553	535	529
la03	10×5	628	586	550	520	489	478	486	484	477
la04	10×5	685	623	568	567	513	502	546	513	502
la05	10×5	593	517	503	483	464	457	520	461	457
la06	15×5	847	**833**	833	822	802	799	818	806	799
la07	15×5	931	809	762	820	751	750	776	754	749
la08	15×5	940	860	845	791	769	765	774	771	765
la09	15×5	949	900	878	888	861	853	875	860	853
la10	15×5	879	**866**	866	845	816	804	821	808	804
la11	20×5	1208	1107	1103	1081	1074	1071	1082	1073	1071
la12	20×5	1001	979	960	959	938	936	946	938	936
la13	20×5	1150	1074	1053	1048	1041	1038	1040	1040	1038
la14	20×5	1195	**1123**	1123	1079	1074	1070	1077	1071	1070
la15	20×5	1390	1269	1111	1176	1101	1090	1098	1091	1089
la16	10×10	1002	952	892	870	751	717	763	**717**	717
la17	10×10	794	781	707	767	682	646	690	**646**	646
la18	10×10	911	897	842	751	717	666	**663**	**663**	663
la19	10×10	973	915	796	860	787	700	652	628	617
la20	10×10	1043	935	857	955	809	756	767	**756**	756
la21	15×10	1248	1136	1017	956	916	835	820	817	806
la22	15×10	993	980	882	888	850	760	784	754	739
la23	15×10	1069	1038	950	918	897	842	836	821	815
la24	15×10	1045	1025	909	908	883	808	812	794	777
la25	15×10	1026	1016	941	908	866	791	876	771	756
la26	20×10	1344	1239	1125	1188	1112	1061	1068	1059	1054
la27	20×10	1419	1347	1186	1192	1152	1091	1116	1095	1085
la28	20×10	1402	1294	1149	1167	1124	1080	1100	1077	1070
la29	20×10	1325	1293	1118	1074	1038	998	1014	1001	994
la30	20×10	1433	1373	1204	1185	1135	1078	1080	1078	1069
la31	30×10	1748	1692	1539	1573	1541	1521	1544	1525	1520
la32	30×10	1896	1819	1698	1711	1698	1659	1673	1665	1658
la33	30×10	1716	1697	1547	1580	1512	1499	1514	1503	1497
la34	30×10	1901	1723	1604	1582	1552	1536	1545	1542	1535
la35	30×10	1911	1817	1736	1621	1583	1550	1564	1558	1549
la36	15×15	1298	1254	1162	1146	1138	1030	965	955	948
la37	15×15	1592	1498	1397	1239	1204	1077	1026	993	986
la38	15×15	1407	1302	1144	1195	1083	962	**943**	**943**	943
la39	15×15	1421	1275	1184	1243	1118	1024	984	935	922
la40	15×15	1321	1291	1150	1104	1080	970	961	**955**	955

Table 20: Part 1 of makespans of all TA instances (JSP).

Instance	$n \times m$	SPT	FIFO	MOR	Park	L2D	SchN	RS	OPT
Ta01	15×15	1462	1486	1438	1389	1443	1452	**1356**	1231
Ta02	15×15	1446	1486	1452	1519	1544	1411	**1372**	1244
Ta03	15×15	1495	1461	1418	1457	1440	**1396**	1467	1218
Ta04	15×15	1708	1575	1457	1465	1637	**1348**	1424	1175
Ta05	15×15	1618	1457	1448	**1352**	1619	1382	1378	1224
Ta06	15×15	1522	1528	1486	1481	1601	1413	**1382**	1238
Ta07	15×15	1434	1497	1456	1554	1568	1380	**1356**	1227
Ta08	15×15	1457	1496	1482	1488	1468	**1374**	1438	1217
Ta09	15×15	1622	1642	1594	1556	1627	1523	**1497**	1274
Ta10	15×15	1697	1600	1582	1501	1527	1493	**1442**	1241
Ta11	20×15	1865	1701	1665	1626	1794	1612	**1590**	1357
Ta12	20×15	1667	1670	1739	1668	1805	1600	**1574**	1367
Ta13	20×15	1802	1862	1642	1715	1932	1625	**1555**	1342
Ta14	20×15	1635	1812	1662	1642	1664	1590	**1511**	1345
Ta15	20×15	1835	1788	1682	1672	1730	1676	**1557**	1339
Ta16	20×15	1965	1825	1638	1700	1710	1550	**1549**	1360
Ta17	20×15	2059	1899	1856	**1678**	1897	1753	1739	1462
Ta18	20×15	1808	1833	1710	1684	1794	**1668**	1705	1396
Ta19	20×15	1789	1716	1651	1900	1682	1622	**1571**	1332
Ta20	20×15	1710	1827	1622	1752	1739	1604	**1560**	1348
Ta21	20×20	2175	2089	1964	2199	2252	1921	**1910**	1642
Ta22	20×20	1965	2146	1905	2049	2102	1844	**1758**	1600
Ta23	20×20	1933	2010	1922	2006	2085	1879	**1867**	1557
Ta24	20×20	2230	1989	1943	2020	2200	1922	**1912**	1644
Ta25	20×20	1950	2160	1957	1981	2201	**1897**	1952	1595
Ta26	20×20	2188	2182	1964	2057	2176	**1887**	1918	1643
Ta27	20×20	2096	2091	2160	2187	2132	**2009**	2015	1680
Ta28	20×20	1968	1980	1952	2054	2146	**1813**	1866	1603
Ta29	20×20	2166	2011	1899	2210	1952	**1875**	1916	1625
Ta30	20×20	1999	1941	2017	2140	2035	1913	**1804**	1584
Ta31	30×15	2335	2277	2143	2251	2565	2055	**2004**	1764
Ta32	30×15	2432	2279	2188	2378	2388	2268	**2097**	1784
Ta33	30×15	2453	2481	2308	2316	2324	2281	**2059**	1791
Ta34	30×15	2434	2546	2193	2319	2332	**2061**	2093	1829
Ta35	30×15	2497	2478	2255	2333	2505	2218	**2179**	2007
Ta36	30×15	2445	2433	2307	2210	2497	2154	**2102**	1819
Ta37	30×15	2664	2382	2190	2201	2325	2112	**1968**	1771
Ta38	30×15	2155	2277	2179	2151	2302	**1970**	1999	1673
Ta39	30×15	2477	2255	2167	2138	2410	2146	**1978**	1795
Ta40	30×15	2301	2069	2028	2007	2140	2030	**1986**	1669
Ta41	30×20	2499	2543	2538	2654	2667	2572	**2435**	2005
Ta42	30×20	2710	2669	2440	2579	2664	2397	**2253**	1937
Ta43	30×20	2434	2506	2432	2737	2431	2310	**2190**	1846
Ta44	30×20	2906	2540	2426	2772	2714	2456	**2343**	1979
Ta45	30×20	2640	2565	2487	2435	2637	2445	**2337**	2000
Ta46	30×20	2667	2582	2490	2681	2776	2541	**2330**	2004
Ta47	30×20	2620	2508	2286	2428	2476	2280	**2270**	1889
Ta48	30×20	2620	2541	2371	2440	2490	2358	**2279**	1941
Ta49	30×20	2666	2550	2397	2446	2556	2301	**2249**	1961
Ta50	30×20	2429	2531	2469	2530	2628	2453	**2251**	1923

Table 21: Part 2 of makespans of all TA instances (JSP).

Instance	$n \times m$	SPT	FIFO	MOR	Park	L2D	SchN	RS	OPT
Ta51	50×15	3856	3590	3567	3145	3599	3382	**2989**	2760
Ta52	50×15	3266	3365	3303	3157	3341	3231	**2986**	2756
Ta53	50×15	3507	3169	3115	3103	3186	3083	**2851**	2717
Ta54	50×15	3142	3218	3265	3278	3266	3068	**2913**	2839
Ta55	50×15	3225	3291	3279	3142	3232	3078	**2915**	2679
Ta56	50×15	3530	3329	3100	3258	3378	3065	**2982**	2781
Ta57	50×15	3725	3654	3335	3230	3471	3266	**3126**	2943
Ta58	50×15	3365	3362	3420	3469	3732	3321	**3108**	2885
Ta59	50×15	3294	3357	3117	3108	3381	3044	**2887**	2655
Ta60	50×15	3500	3129	3044	3256	3352	3036	**2861**	2723
Ta61	50×20	3606	3690	3376	3425	3654	3202	**3133**	2868
Ta62	50×20	3639	3657	3417	3626	3722	3339	**3234**	2869
Ta63	50×20	3521	3367	3276	**3110**	3536	3118	3130	2755
Ta64	50×20	3447	3179	3057	3329	3631	2989	**2982**	2702
Ta65	50×20	3332	3273	3249	3339	3359	3168	**3007**	2725
Ta66	50×20	3677	3610	3335	3340	3555	3199	**3153**	2845
Ta67	50×20	3487	3612	3392	3371	3567	3236	**3054**	2825
Ta68	50×20	3336	3471	3251	3265	3680	3072	**2991**	2784
Ta69	50×20	3862	3607	3526	3798	3592	3535	**3305**	3071
Ta70	50×20	3801	3784	3590	3919	3643	3436	**3304**	2995
Ta71	100×20	6232	6270	5938	5962	6452	5879	**5679**	5464
Ta72	100×20	5973	5671	5639	5522	5695	5456	**5297**	5181
Ta73	100×20	6482	6357	6128	6335	6462	6052	**5807**	5568
Ta74	100×20	6062	6003	5642	5827	5885	5513	**5413**	5339
Ta75	100×20	6217	6420	6212	6042	6355	5992	**5623**	5392
Ta76	100×20	6370	6183	5936	5707	6135	5773	**5487**	5342
Ta77	100×20	6045	5952	5829	5737	6056	5637	**5475**	5436
Ta78	100×20	6143	6328	5886	5979	6101	5833	**5426**	5394
Ta79	100×20	6018	6003	5652	5799	5943	5556	**5411**	5358
Ta80	100×20	5848	5763	5707	5718	5892	5545	**5448**	5183