A Branch-and-Bound Algorithm for Single-Machine Scheduling Problems with Candidate Tools and Release Time (Supplementary File)

Shengchao Li, Shixin Liu, *Member, IEEE*, Ziyan Zhao, *Member, IEEE*, and Mengchu Zhou, *Fellow, IEEE*

Appendix A
Examples

We show two examples to understand $1|K_j, r_j|C_{max}$ and its degenerate problem $1|K_j|C_{max}$, as well as the corresponding B&B and BrFS-B&B algorithms presented in this paper. Both examples use the data in Table A.1 where there are 6 jobs and 5 tools. Each job has its release time r_j, processing time p_j, and candidate tool set K_j. Each tool show its setup time s_k.

TABLE A.1
An Instance of $1|K_j, r_j|C_{max}$

		j_1	j_2	j_3	j_4	j_5	j_6
job	r_j	83	963	1359	1500	2432	3532
	p_j	600	598	681	482	695	599
	K_j	k_4,k_5	k_4,k_5	k_3	k_1,k_2,k_5	k_3	k_4,k_5
tool		k_1	k_2	k_3	k_4	k_5	-
	s_k	26	27	32	33	33	-

Note. r_j, p_j, and s_k are measured in minutes.

A. An Example of $1|K_j, r_j|C_{max}$

Example 1. Consider the problem $1|K_j, r_j|C_{max}$ corresponding to Table A.1. An optimal solution is shown in Fig. A.1. The optimal sequence is $\langle(j_1, k_4)$, (j_2, k_5), (j_4, k_5), (j_3, k_3), (j_5, k_3), $(j_6, k_4)\rangle$ with a makespan of 4181. Each column in Fig. A.1 represents the tools that can be used for the current job, and each row indicates the jobs matched with the current tool. The white background area indicates that the current tool is selected by the job; the shaded area indicates the current tool is a candidate for the job but not used; and the gray background area indicates the installation or removal of the current tool. For instance, job j_1 selects tool k_4, thus k_4 is installed on the machine before j_1 begins processing. Since j_2 selects k_5, k_4 must be removed after j_1 is completed. Note that if FCFS is applied, the sequence is $\langle(j_1, k_4), (j_2, k_4), (j_3, k_3), (j_4, k_1), (j_5, k_3), (j_6, k_4)\rangle$ with a makespan of 4297 where the tool with minimum setup time for each job is selected because the candidate tool sets for two adjacent jobs do not intersect. The sequence is not optimal for $1|K_j, r_j|C_{max}$.

The branch-and-bound tree of Example 1 is shown in Fig. A.2. Job j_1 choosing tool k_4 is reduced and is not shown in the figure. There are 5 remaining jobs, so the created nodes

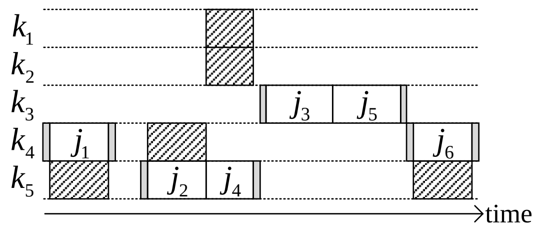

Fig. A.1. An optimal solution of Example 1.

15/20/24/31/32 at level 5 represent feasible solutions to the problem. The lower bound for each of these nodes is the upper bound for the current node and the problem. The initial upper bound UB_0 of the problem obtained by Algorithm 3 is 4297. Notice that node 7 is dominated by node 16 according to Theorem 5 . Nodes 25 and 10 are pruned at the 15th and 19th explorations, respectively, due to the lower bound pruning rule. Meanwhile, nodes 13/14/26/27 and 11/18/22/29 are pruned at the 15th and 19th explorations, respectively, because their lower bounds are not smaller than those of nodes 25 and 10. The Gantt chart of the optimal solution corresponding to node 31 is shown in Fig. A.1.

B. An Example of $1|K_j|C_{max}$

Example 2. Consider the degenerate problem $1|K_j|C_{max}$ corresponding to Table A.1, i.e., excluding the release time of jobs. The branch-and-bound tree is shown in Fig. A.3(a). Jobs j_3 and j_5 do not appear in the figure since they are reduced by the domain reduction of BrFS-B&B. The corresponding optimal sequence is $\langle(j_3, k_3), (j_5, k_3), (j_1, k_5), (j_2, k_5), (j_4, k_5), (j_6, k_5)\rangle$ with a makespan of 3785, and the Gantt chart is shown in Fig. A.3(b).

Appendix B
Proof of Theorem 2

Proof. Consider an arbitrary feasible sequence $\langle(j_2, k^{j_2}), (j_3, k^{j_3}), ..., (j_{|J|}, k^{j_{|J|}})\rangle$ of $J\backslash\{j_1\}$, denoted as $\mathcal{S}$. Note that $T_{j_2} = r_{j_2}$ must hold, because j_2 is the first processed job in $\mathcal{S}$ and satisfies inequality (31) . The proof is divided into two cases below.

Case (a): j_1 does not batch with any other jobs in $\mathcal{S}$. In this case, tool switching must occur before and after j_1

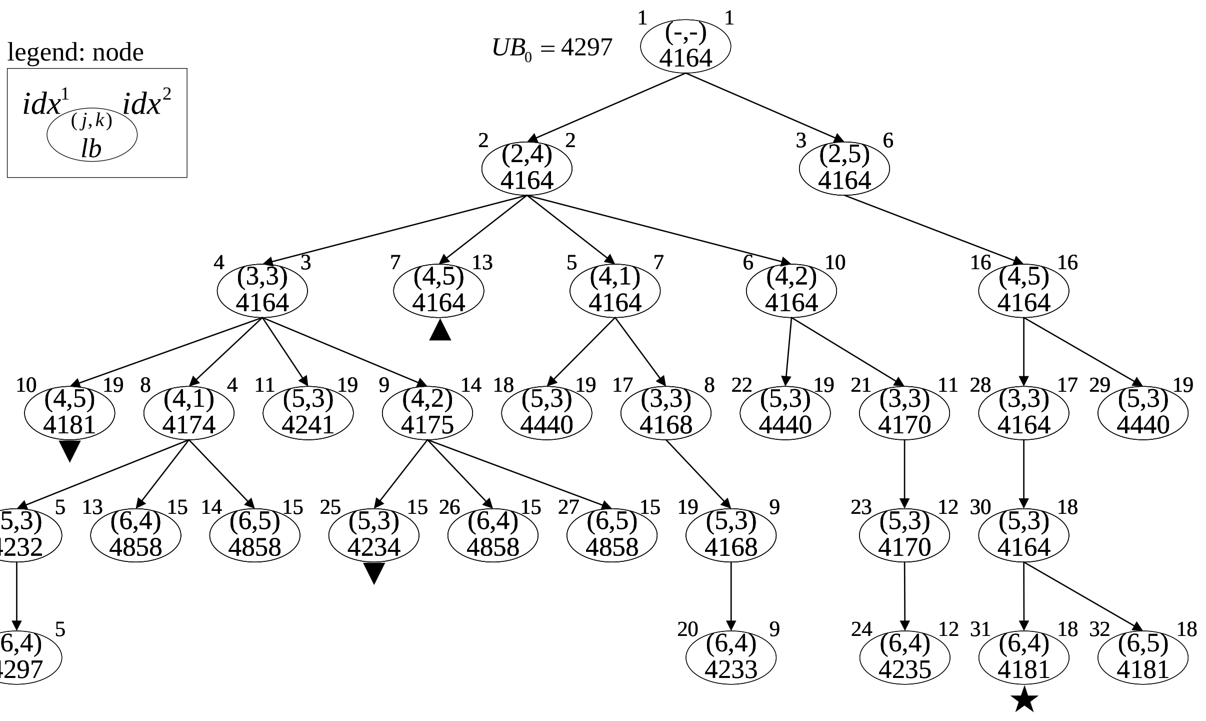

Fig. A.2. The branch-and-bound tree of Example 1. idx^1 and idx^2 indicate the created and explored node indices, respectively. The pair (j, k) indicates the job and tool selected by the current node. In particular, node 1 denotes the root node, which does not store any job and tool information. lb indicates the lower bound of the current node. The nodes marked ▲ and ▼ indicate that the node is pruned due to the dominance rule (i.e., Theorem 5) and the lower bound of the node pruning rule (i.e., Theorem 4), respectively. Node 31 marked ★ is the optimal node obtained by B&B.

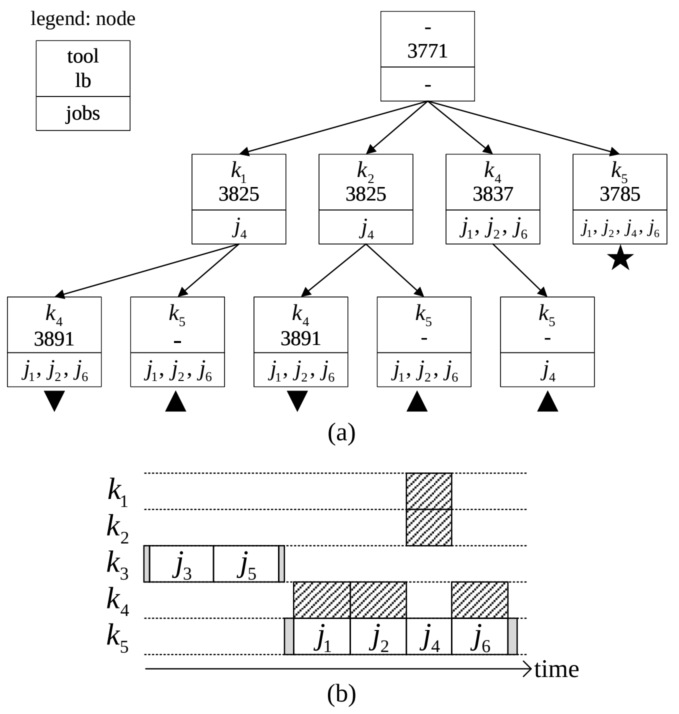

Fig. A.3. The branch-and-bound tree and the optimal solution of Example 2. Fig. (a) is the branch-and-bound tree of Example 2. A node stores the current tool, the lower bound (i.e., lb), and the jobs that can use the current tool. The nodes marked ▲ and ▼ indicate that the node is pruned due to the dominance rule (i.e., Proposition 2) and the lower bound (i.e., Proposition 3), respectively. The node marked ★ is the optimal node obtained by BrFS-B&B, and the corresponding optimal sequence in shown in Fig. (b).

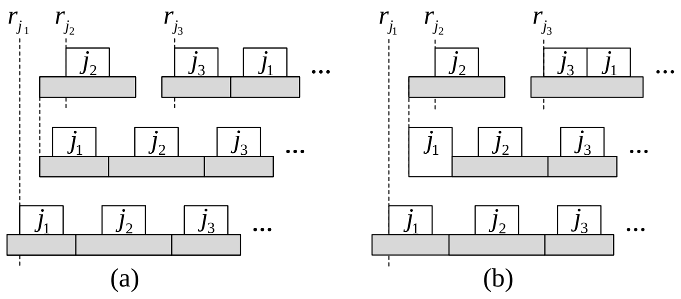

Fig. A.4. Graphical illustration of the proof of Theorem 2 . Figs. (a) and (b) show the cases where j_1 does not batch with any other jobs in $\mathcal{S}$ and j_1 batches with another job in $\mathcal{S}$, respectively.

is processed. If j_1 is not processed first, moving it to the beginning results in a better scheduling sequence (see Fig. A.4(a)). Note that j_1 has the minimum release time in J and relocating it to the beginning can eliminate part of the idle time caused by release time. According to inequality (32) , the start time of j_1 in the adjusted sequence satisfies:

$$\check{T}_{j_1} = r_{j_2} - s_{k^{j_2}} + s_{k^{j_1}} \geq r_{j_2} - \max_{k \in K_{j_1}, k' \in K_{j_2}} H(k, k') \geq r_{j_1}.$$

That is the release time constraint of j_1 is satisfied, so the above adjustment is feasible.

Case (b): j_1 batches with another job in $\mathcal{S}$. In this case, no tool switching is required inside the batch covering j_1. Similar to case (a), moving j_1 to the beginning results in a better schedule (see Fig. A.4(b)). The difference is that the adjustment in this case does not include the installation and removal of the tool selected by j_1. When assigning the tool for j_1, the following equation holds due to inequality (32) .

$$\begin{aligned}\check{T}_{j_1} &= r_{j_2} - s_{k^{j_2}} + p_{j_1} + s_{k^{j_2}} - H(k^{j_1}, k^{j_2}) - p_{j_1} \\ &\geq r_{j_2} - \max_{k \in K_{j_1}, k' \in K_{j_2}} H(k, k') \geq r_{j_1}.\end{aligned}$$

Particularly, the equation still holds when j_1 and j_2 are in the same batch. The release time constraint of j_1 is satisfied, so the above adjustment is feasible.

According to the above two cases, we have the inequality $C_{max}\left(\langle j_1, j_2, j_3, ..., j_{|J|}\rangle\right) \leq C_{max}\left(\langle j_2, j_3, ..., j_1, ..., j_{|J|}\rangle\right)$. Due to the arbitrariness of $\mathcal{S}$, there must exist an optimal sequence where j_1 is processed first. □

Appendix C
Data Generation Process

The complete data generation process is shown as Procedure 1. Firstly, $|K|$ types of tools are generated, including the setup time of each tool type and the job types matched by each tool. Secondly, obtain the matched tool set for each type of job. Finally, $|J|$ jobs are generated. Note that the job release time needs to be done after the processing time is generated. According to the current data generation parameters, it is guaranteed that KT_i $(i \in \{J1, ..., J7\})$ is non-empty (i.e., each instance is feasible) even if each tool type corresponds to only a minimum number of job types (i.e., $|JT_k| = 2, \forall k \in K$).

Procedure 1: Data Generation

```
   Output: $J$, $K$
1  $JT := \{J1, J2, ..., J7\}$;
2  for $k = 1, 2, ..., |K|$ do
3      generate $s_k$ and a random number $num$ using $Ui(2, 3)$;
4      if $|JT| < num$ then
5          $JT_k \leftarrow JT$; $JT \leftarrow \{J1, J2, ..., J7\}$;
6      while True do
7          $JT'_k \leftarrow$ choose $num - |JT_k|$ job types from $JT$ randomly;
8          if $JT'_k \cap JT_k = \emptyset$ then
9              $JT \leftarrow JT \backslash JT'_k$; $JT_k \leftarrow JT_k \cup JT'_k$;
10             break while;
11 for $i = J1, J2, ..., J7$ and $k = 1, 2, ..., |K|$ do
12     if $i \in JT_k$ then $KT_i \leftarrow KT_i \cup \{k\}$;
13 for $j = 1, 2, ..., |J|$ do
14     $j.type \leftarrow$ choose a job type using roulette;
15     $K_j \leftarrow KT_{j.type}$;
16     generate $p_j$;
17 for $j = 1, 2, ..., |J|$ do generate $r_j$;
18 return $J$, $K$.
```

Appendix D
Computational Results and Analysis on Datasets D2-D4

Cases under different data generation parameters are generated to illustrate the generalizability of the algorithm. The setup time of tools in baseline data is much smaller compared to the processing time of jobs, so the range of data generated for setup time increases in this part. In addition, tightening and slackening the release time are considered to reflect the dense and sparse arrival of jobs, respectively.

A. Increase Setup Time (D2)

The results for the small-scale and large-scale cases are shown in Tables A.2 and A.3, respectively. The lower bounds obtained by MILP and CP are difficult to converge. In terms of the solving time for each solved instance, B&B significantly outperforms MILP and CP. Compared to dataset D1, the solvable case scales of both MILP and B&B become smaller, since increasing setup time limits the application of some strategies, making the feasible solution space larger. This can be seen from the values of the columns $avg\ nodes$ in Tables IV and A.3. Due to the unique constraint propagation technique, the solvable case scale for CP remains essentially constant.

The implementation details of B&B for all cases are shown in Fig. A.5(a) and Table A.4. Since r_j is only related to $\sum_{j \in J} p_j$, increasing the setup time makes job arrival dense. That makes the condition in line 15 of Algorithm 3 difficult to satisfy, and thus the percentage of job reduction decreases significantly. The percentage of the dominance rule pruning does not change significantly. However, it take longer to determine whether the current node is dominated or not because of the increased number of explored nodes. In addition, the branching strategy for the root node is limited because the inequalities (31) and (32) are not easily satisfied. These all lead to limitations on solution performance.

B. Tighten Release Time (D3)

The results for the small-scale and large-scale cases are shown in Tables A.2 and A.3, respectively. Tightening the release time makes the feasible solution space larger, and thus the case scale that can be solved optimally by MILP and CP is smaller. In fact, the number of ready jobs in the scheduling process increases when the jobs arrive more centrally, which benefits the job batching (i.e., building the subtrees of lines 10 and 13 in Algorithm 4). In particular, when all jobs have the same release time, the problem degenerates to the easily solvable problem $1|K_j|C_{max}$. However, MILP and CP cannot recognize this property and performs poorly. On the contrary, B&B can utilize it to solve fairly large-scale cases.

The implementation details of B&B for all cases are shown in Fig. A.5(b) and Table A.4. As can be seen in the figure, the significantly large percentage of middle nodes suggests that Theorem 3 plays an important role. In contrast, the enumeration branching is hardly ever used. Although the percentage of job reduction decreases significantly and the branching strategy for the root node is limited, they have little effect on solving large-scale problems.

C. Slacken Release Time (D4)

The results for the small-scale and large-scale cases are shown in Tables A.2 and A.3, respectively. For MILP and

TABLE A.2
COMPARISON OF SMALL-SCALE RESULTS BETWEEN MILP, CP, AND B&B UNDER DATASETS D2-D4

$\|J\|$	*method*	*#opt*	*time/s*			gap^L/%			gap^U/%		
			min	*avg*	*max*	*min*	*avg*	*max*	*min*	*avg*	*max*
increase setup time											
6	MILP	20	0.03	0.08	0.16	/	/	/	/	/	/
	CP	20	0.05	0.23	0.48	/	/	/	/	/	/
	B&B	20	<0.01	<0.01	<0.01	/	/	/	/	/	/
9	MILP	20	0.72	6.87	65.13	/	/	/	/	/	/
	CP	20	0.18	0.63	4.12	/	/	/	/	/	/
	B&B	20	<0.01	<0.01	0.02	/	/	/	/	/	/
12	MILP	14	2.66	406.38	2007.47	9.34	14.83	19.19	0.00	0.00	0.00
	CP	20	0.23	3.39	15.34	/	/	/	/	/	/
	B&B	20	<0.01	<0.01	0.03	/	/	/	/	/	/
15	MILP	4	43.25	367.94	637.19	3.26	13.42	21.09	0.00	2.19	2.19
	CP	19	0.47	119.93	1210.07	14.91	14.91	14.91	0.00	0.00	0.00
	B&B	20	<0.01	0.03	0.38	/	/	/	/	/	/
tighten release time											
6	MILP	20	0.03	0.10	0.30	/	/	/	/	/	/
	CP	20	0.05	0.40	0.75	/	/	/	/	/	/
	B&B	20	<0.01	<0.01	0.01	/	/	/	/	/	/
8	MILP	20	0.72	1.27	2.38	/	/	/	/	/	/
	CP	20	0.11	1.52	2.70	/	/	/	/	/	/
	B&B	20	<0.01	<0.01	0.02	/	/	/	/	/	/
10	MILP	20	3.23	283.84	1749.91	/	/	/	/	/	/
	CP	20	1.24	8.08	20.55	/	/	/	/	/	/
	B&B	20	<0.01	<0.01	<0.01	/	/	/	/	/	/
12	MILP	2	287.61	1037.79	1787.97	14.09	23.54	32.15	0.00	0.00	0.00
	CP	16	6.02	147.35	796.47	45.34	48.02	51.08	0.00	0.00	0.00
	B&B	20	<0.01	<0.01	0.01	/	/	/	/	/	/
slacken release time											
10	MILP	20	0.03	1.56	14.09	/	/	/	/	/	/
	CP	20	0.03	0.73	2.84	/	/	/	/	/	/
	B&B	20	<0.01	<0.01	<0.01	/	/	/	/	/	/
15	MILP	17	0.03	31.06	198.39	1.51	3.72	5.86	0.00	0.14	0.14
	CP	19	0.05	28.75	453.21	3.79	3.79	3.79	0.00	0.00	0.00
	B&B	20	<0.01	<0.01	0.02	/	/	/	/	/	/
20	MILP	7	0.06	175.46	1186.22	0.32	5.28	19.91	0.00	0.32	0.32
	CP	16	0.05	127.08	1333.35	3.08	10.52	19.91	0.00	0.00	0.00
	B&B	20	<0.01	<0.01	0.01	/	/	/	/	/	/

Note. Time limit is 3600s.

TABLE A.3
RESULTS OF B&B FOR LARGE-SCALE CASES UNDER DATASETS D2-D4

$\|J\|$	*#opt*	*time/s*			*gap*/‰			*nodes*		
		min	*avg*	*max*	*min*	*avg*	*max*	*min*	*avg*	*max*
increase setup time										
20	20	<0.01	0.29	1.95	/	/	/	313	4474	13461
30	19	<0.01	37.76	615.47	10.26	10.26	10.26	117	42670	327994
40	18	0.02	569.12	3253.04	8.90	12.09	15.28	1843	144193	383911
50	17	4.48	1258.69	3420.12	8.02	19.30	41.58	24963	248765	486979
60	7	0.20	1054.46	2937.36	6.29	19.20	30.87	4513	340281	491620
tighten release time										
100	20	<0.01	0.32	1.44	/	/	/	232	5075	13886
300	19	0.02	62.90	568.94	0.63	0.63	0.63	1122	87956	589234
500	16	1.97	231.06	1292.66	0.16	0.33	0.41	6776	162203	521423
750	12	2.09	766.12	3388.72	0.15	0.27	0.42	12570	215848	653107
1000	9	0.09	675.06	3421.79	0.11	0.23	0.43	2027	160907	523065
slacken release time										
100	20	<0.01	66.47	843.71	/	/	/	1	44259	339892
200	20	<0.01	349.73	3320.65	/	/	/	1	105751	872901
300	17	<0.01	213.36	1726.17	0.99	1.59	1.91	97	138366	551434
400	16	<0.01	0.03	0.30	1.18	2.52	4.10	1	100027	668720
500	17	<0.01	0.09	1.15	0.26	0.77	1.17	1	67643	463524
600	14	<0.01	101.57	1190.87	0.29	1.08	2.01	1	221473	1336733

Note. Time limit is 3600s.

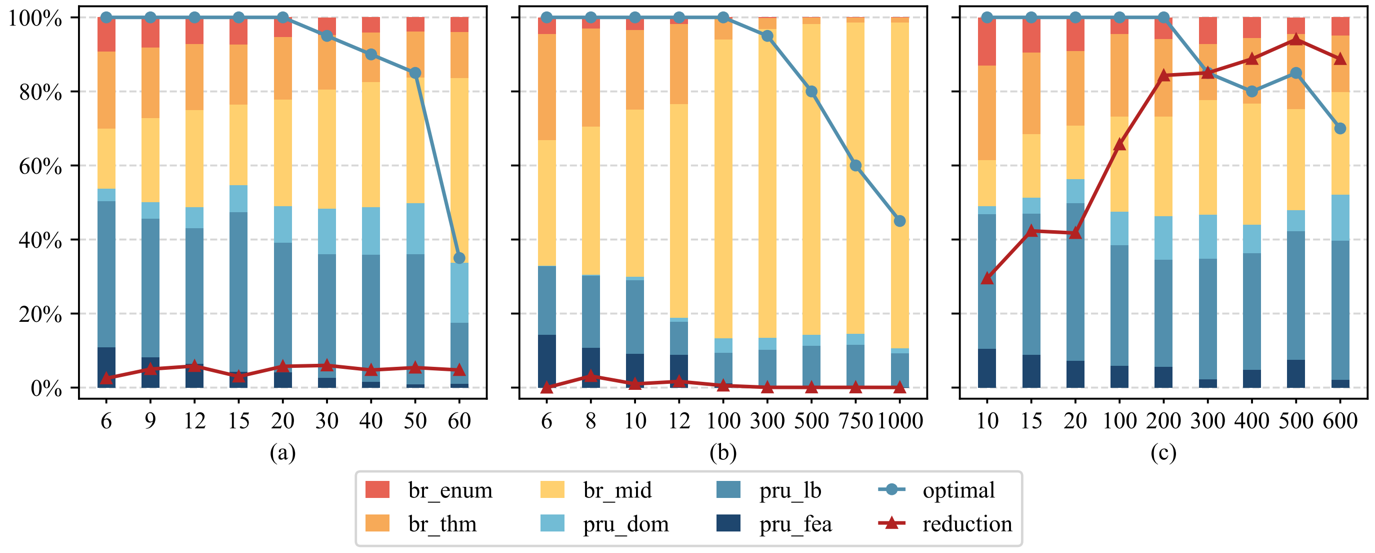

Fig. A.5. Implementation details of B&B for datasets D2-D4. Figs. (a)-(c) correspond to datasets D2-D4, respectively. See Fig. 1 for legend information.

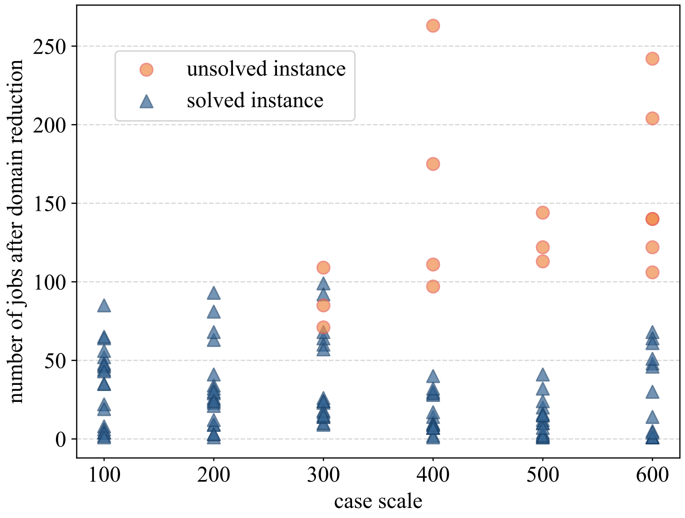

Fig. A.6. Number of jobs for each instance after domain reduction for dataset D4. Only the results for large-scale cases are shown in the figure.

CP, the number of solved instances for each case increases slightly, but the case scale that can be solved optimally remains limited. In contrast, B&B can solve quite large-scale cases in a short time. Note that the solved instances consume quite short time to attain the optimal solutions when $|J| = 400$ and $|J| = 500$. This is because, for the instances generated with the current parameters, the solving time of B&B is not positively related to the case scale, but basically depends on the number of jobs J' after domain reduction. By analyzing the experimental results, we find that: if $|J'| < 50$, B&B can solve the problem in a quite short time; if $|J'| \in [50, 100]$, most problems can be solved optimally in 3600s but need to consume some time; if $|J'| > 100$, the problem can hardly be solved optimally in 3600s. According to Table A.5 and Fig. A.6, the number of jobs after domain reduction is mostly either less than 50 or greater than 100 for the instances with case scales 400 and 500. Therefore, these instances are either solved optimally quite quickly or cannot be solved within 3600s.

The implementation details of B&B for all cases are shown in Fig. A.5(c) and Table A.4. As can be seen in the figure, the job reduction ratio increases with the case scale. But this does not mean that the problem is easier to solve. The difficulty of solving the problem for this dataset mainly depends on the number of jobs after domain reduction. In addition, as can be seen in the table, the number of instances with optimal initial solutions increases. If the relaxation factor α is larger, this number will be even larger, since the optimal sequence can basically be obtained by FCFS.

TABLE A.4
IMPLEMENTATION DETAILS OF B&B FOR ALL DATASETS

$\lvert J\rvert$	opt/%	$\overline{DR}$/%	$\overline{UB_0}$	I^{sol}	$\overline{N}^{cr}$	pruning/%			branching/%		
						fea	lb	dom	mid	thm	$enum$
dataset D1											
10	100	26.00	7651	6	79	11.64	30.97	2.40	19.67	25.16	10.16
15	100	29.67	10865	3	439	7.31	37.86	4.61	18.54	22.99	8.69
20	100	25.25	14344	3	2315	6.43	35.32	6.66	23.31	19.87	8.42
25	100	18.40	17724	1	3548	3.36	37.75	9.52	21.75	19.44	8.19
30	100	32.33	21107	2	7153	5.74	30.55	8.18	30.06	19.76	5.71
40	100	25.75	27793	0	42333	3.40	33.59	12.68	25.82	17.81	6.70
50	100	25.80	34381	0	87507	2.09	31.97	13.32	29.71	17.21	5.70
60	95	27.92	41618	0	254277	2.04	28.87	14.03	34.81	15.30	4.96
70	85	18.79	47270	0	451995	0.93	28.48	17.15	30.71	16.28	6.44
80	80	30.50	53396	1	551550	2.38	24.23	16.57	34.56	16.65	5.61
90	75	30.11	61061	0	697091	1.20	23.83	15.38	40.98	14.37	4.23
100	70	24.55	67167	0	867472	0.62	24.80	16.53	39.96	13.69	4.41
dataset D2											
6	100	2.50	5596	9	109	10.87	39.42	3.38	16.31	20.80	9.23
9	100	5.00	8225	4	394	8.17	37.39	4.48	22.77	19.05	8.14
12	100	5.83	10478	2	1109	6.45	36.63	5.55	26.27	17.92	7.19
15	100	3.00	12562	1	3606	4.20	43.20	7.24	21.80	16.15	7.40
20	100	5.75	16345	1	13392	4.19	34.96	9.78	28.92	16.82	5.33
30	95	6.00	23657	0	117001	2.63	33.33	12.31	32.14	14.76	4.82
40	90	4.75	31152	0	428858	1.57	34.31	12.80	33.81	13.38	4.13
50	85	5.40	38290	0	776121	0.91	35.04	13.85	33.83	12.44	3.92
60	35	4.75	45859	0	1007978	0.99	16.42	16.34	49.81	12.48	3.97
dataset D3											
6	100	0.00	4217	9	42	14.28	18.41	0.26	33.81	28.78	4.46
8	100	3.13	5606	1	59	10.76	19.45	0.21	40.06	26.41	3.10
10	100	1.00	6754	5	129	9.16	19.80	0.95	45.12	21.51	3.47
12	100	1.67	8130	3	196	8.92	8.89	1.03	57.71	21.58	1.87
100	100	0.55	63028	0	49277	1.16	8.18	3.97	80.67	5.70	0.32
300	95	0.07	187193	0	1500595	0.35	9.77	3.31	83.30	3.06	0.20
500	80	0.06	311108	0	3171258	0.22	11.08	2.86	84.04	1.72	0.08
750	60	0.06	465945	0	5406385	0.13	11.45	2.92	84.03	1.40	0.06
1000	45	0.05	621028	0	5735801	0.10	9.08	1.48	87.96	1.34	0.06
dataset D4											
10	100	29.50	7941	12	74	10.51	36.31	2.20	12.40	25.48	13.11
15	100	42.33	11346	6	251	8.92	38.08	4.23	17.28	21.92	9.58
20	100	41.75	14993	6	714	7.18	42.64	6.44	14.42	20.15	9.17
100	100	65.70	70836	5	126800	5.85	32.55	9.05	25.72	22.24	4.59
200	100	84.33	139442	4	230299	5.55	29.01	11.65	26.95	20.95	5.90
300	85	85.02	208587	1	366208	2.30	32.50	11.80	31.05	15.02	7.33
400	80	88.79	275700	4	241737	4.77	31.47	7.70	32.70	17.74	5.62
500	85	94.06	343459	4	212042	7.50	34.65	5.73	27.27	20.28	4.57
600	70	88.76	411401	6	571127	2.05	37.61	12.39	27.78	15.15	5.01

Note. opt: the percentage of solved instances; $\overline{DR}$: the average percentage of reduced jobs; $\overline{UB_0}$: the average initial upper bound for 20 instances; I^{sol}: the number of instances with an optimal initial solution; $\overline{N}^{cr}$: the average number of created nodes; fea, lb, dom: the average percentages of nodes in $\overline{N}^{cr}$ pruned due to the feasibility, the lower bound, and the dominance rule, respectively; mid: the average percentage of nodes in $\overline{N}^{cr}$ that are middle nodes in the branching process; thm, $enum$: the average percentage of nodes in $\overline{N}^{cr}$ that use theorem and branching, respectively.

TABLE A.5
NUMBER OF JOBS FOR EACH INSTANCE AFTER DOMAIN REDUCTION FOR DATASET D4

$\lvert J\rvert$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
100	64	8	65	22	4	52	2	6	35	4	56	46	44	48	35	47	1	85	43	19
200	63	29	93	9	1	23	25	68	34	3	26	21	9	41	32	30	81	12	24	3
300	64	9	10	24	23	99	60	14	26	68	17	57	14	18	$\underline{71}$	15	92	$\underline{109}$	24	$\underline{85}$
400	14	17	9	10	$\underline{263}$	28	7	9	$\underline{175}$	2	40	1	$\underline{111}$	7	$\underline{97}$	30	29	9	7	32
500	10	20	12	1	2	$\underline{144}$	$\underline{122}$	24	3	4	1	10	2	41	7	15	16	$\underline{113}$	32	15
600	$\underline{204}$	5	1	68	$\underline{140}$	64	61	48	46	1	1	$\underline{242}$	$\underline{140}$	30	$\underline{122}$	51	14	1	$\underline{106}$	4

Note. Only the results for large-scale cases are shown in the table. Columns 1-20 indicate the 20 instances of each case. The underlined data indicate that the optimal solutions for these instances cannot be found within 3600s using B&B, i.e., these instances are unsolved.