SAAD Practical Assignment
Group 1A

José Pedro Evans[1], Ricardo Correia Pinto[1], and Vitor Souza Pina[1]

[1]Faculdade de Engenharia da Universidade do Porto, Porto, Portugal

Abstract. Scheduling problems, such as Job Shop (JS) and Open Shop (OS), are classical challenges in operational research, characterized by their combinatorial complexity and significant real-world applications. These problems involve assigning a set of jobs to machines while optimizing specific criteria, such as minimizing makespan. This study presents a comparative analysis of two widely used approaches for solving these problems: Mixed Integer Programming (MIP) and Constraint Programming (CP). The methodologies were evaluated using a comprehensive set of benchmark instances with varying levels of complexity and problem sizes. Performance was assessed based on three key metrics: time to optimality, average gap, and scalability under time constraints of 100 and 200 seconds.

Results reveal a clear advantage of CP over MIP in tackling both JS and OS problems, particularly for larger instances. For JS problems, CP demonstrated higher rates of optimality and consistently produced solutions with smaller gaps across all instance sizes. In contrast, MIP struggled to achieve optimality, especially as problem size increased, often yielding higher gaps and requiring longer computational times. Similarly, for OS problems, CP excelled in identifying optimal solutions across all tested instances, while MIP's performance declined sharply for larger problems. These findings highlight the robustness and scalability of CP in handling the combinatorial complexities inherent in scheduling problems.

This study underscores the potential of CP to outperform traditional MIP approaches. The comparative results also emphasize the importance of selecting suitable modeling techniques based on problem characteristics and computational constraints.

Keywords: Scheduling · Data Science · Operational Research

1 Introduction

One of the most classic challenges in the field of operational research involves combinatorial optimization problems, with machine scheduling problems representing a fundamental application within this domain [1, 2]. In such problems, the objective typically involves processing a set of jobs on a set of machines, with the ultimate goal of optimizing the completion time of all jobs. A significant proportion of scheduling problems are NP-hard, meaning they cannot be

solved in polynomial time, which adds to the intrigue and complexity of their study.

A common subset of scheduling problems is shop scheduling problems, which can be described as a combinatorial problem involving a set M of machines and a set J of n jobs, where each job j comprises a sequence of n_j operations or tasks that must be performed on specific machines [3]. Generally, each operation o_{ij} in each job j_i is characterized by a processing time p_{ij} on a specific machine m_{ij}; additionally, operations are restricted to being processed on only one machine at a time, while each machine can execute only one operation at any given moment. Another key constraint involves the ordering of operation processing, which defines the two main classes of shop scheduling problems: Job Shop (JS) and Open Shop (OS), the focus of this report. Regarding the former, it adheres to precedence constraints, requiring each operation within a job to start only after its preceding operations have been completed, enforcing processing in a predetermined sequence on specific machines [3]. In contrast, the OS problem is a combinatorial optimization challenge that offers greater flexibility: while each operation still requires processing on a set of machines, the order in which operations are carried out is not fixed or predetermined [4]. Finally, in both JS and OS problems, the most typical optimization is the minimization of the makespan, i.e., the period from when the first operation starts until the last operation is finished.

Due to its nature being a combinatorial problem, JS and OS are naturally represented as a constraint satisfaction problem. Constraint-based approaches have been successfully applied to scheduling problems over the years, demonstrating their effectiveness in finding feasible and optimal solutions. [5]. Two of these approaches are Mixed Integer Programming (MIP) and Constraint Programming (CP) models, which have been largely applied to solving this type of problems. MIP relies on linear programming formulations with integer or binary variables, making it particularly suited for problems with linear objectives and constraints. In contrast, CP excels in handling combinatorial complexity by leveraging logical and global constraints, making it more adept at addressing intricate dependencies like precedence relations or temporal constraints. While MIP benefits from robust optimization techniques like branch-and-bound, CP employs domain reduction and constraint propagation to efficiently explore the solution space [6].

Here, we present an implementation of one MIP model and one CP model for JS and OS benchmarking instances, comparing the based on several key performance indicators (KPIs).

2 The MIP model

2.1 Mixed Integer Programing

In linear programming problems, the decision variables are usually defined as non-negative real values within a specified range. When real values are not feasible due to the nature of the problem, the model is referred to as an integer

linear programming problem. If the model includes both integer and continuous variables, it is classified as a mixed-integer programming problem. [7]

2.2 Mathematical modeling for the JS problem

In the context of the JS problem, the mathematical model utilized in this article is based on the disjunctive model proposed by Wen-Yang [8]. This model employs the Big θ method alongside additional variables to ensure compliance with all scheduling rules.

$$\begin{aligned}
&\min C_{\max} \\
&\text{s.t.} \\
&\quad x_{\sigma_h^j,j} \geq x_{\sigma_{h-1}^j,j} + p_{\sigma_{h-1}^j,j}, && \forall j \in J, h = 2,\ldots,m && (1)\\
&\quad x_{ij} \geq x_{ik} + p_{ik} - \theta \cdot z_{ijk}, && \forall j,k \in J, j < k, i \in M && (2)\\
&\quad x_{ik} \geq x_{ij} + p_{ij} - \theta \cdot (1 - z_{ijk}), && \forall j,k \in J, j < k, i \in M && (3)\\
&\quad C_{\max} \geq x_{\sigma_m^j,j} + p_{\sigma_m^j,j}, && \forall j \in J && (4)\\
&\quad z_{ijk} \in [0,1], && \forall j,k \in J, i \in M && (5)\\
&\quad x_{ij} \geq 0, && \forall j \in J, i \in M && (6)
\end{aligned}$$

where:

- J: A finite set of n jobs.
- M: A finite set of m machines.
- σ_{hj}: The h-th operation of job j, representing the sequence of machines through which j must be processed.
- σ_{mj}: The last operation of job j.
- p_{ij}: A non-negative integer representing the processing time of job j on machine i.
- x_{ij}: An integer representing the start time of job j on machine i.
- z_{ijk}: A binary variable equal to 1 if job j precedes job k on machine i, and 0 otherwise.

2.3 Mathematical modeling for the OS scheduling

The OS problem is similar to the JS problem but involves a relaxation of one key constraint. Specifically, the operations of each job do not need to follow a strict sequence. To model this scenario, we adapted the mathematical formulation of the JS problem by removing the precedence constraint and introducing constraints 7 and 8. These additional constraints ensure that tasks belonging to the same job cannot be processed simultaneously on two different machines.

$$
\begin{aligned}
&\min C_{\max} \\
\text{s.t.}\quad & x_{i_2 j} \geq x_{i_1 j} + p_{i_1 j} - \theta \cdot (1 - z_{ijk}), && \forall i_1, i_2 \in M,\ j \in J,\ i_1 \neq i_2, && (7)\\
& x_{i_1 j} \geq x_{i_2 j} + p_{i_2 j} - \theta \cdot z_{ijk}, && \forall i_1, i_2 \in M,\ j \in J,\ i_1 \neq i_2, && (8)\\
& x_{ij} \geq x_{ik} + p_{ik} - \theta \cdot z_{ijk}, && \forall j, k \in J,\ j < k,\ i \in M, && (9)\\
& x_{ik} \geq x_{ij} + p_{ij} - \theta \cdot (1 - z_{ijk}), && \forall j, k \in J,\ j < k,\ i \in M, && (10)\\
& C_{\max} \geq x_{\sigma_m^j, j} + p_{\sigma_m^j, j}, && \forall j \in J, && (11)\\
& z_{ijk} \in [0, 1], && \forall j, k \in J,\ i \in M, && (12)\\
& x_{ij} \geq 0, && \forall j \in J,\ i \in M. && (13)
\end{aligned}
$$

where:

- J: A finite set of n jobs.
- M: A finite set of m machines.
- σ_{hj}: The h-th operation of job j, representing the sequence of machines through which j must be processed.
- σ_{mj}: The last operation of job j.
- p_{ij}: A non-negative integer representing the processing time of job j on machine i.
- x_{ij}: An integer representing the start time of job j on machine i.
- z_{ijk}: A binary variable equal to 1 if job j precedes job k on machine i, and 0 otherwise.

3 The CP model

3.1 Constraint Programming

CP is a problem-solving paradigm in computer science and operational research that involves specifying a problem in terms of constraints that must be satisfied. Instead of explicitly enumerating steps to reach a solution (as in imperative programming), CP focuses on defining the problem's properties and relies on algorithms to find solutions that meet these requirements [9].

3.2 Mathematical modeling for the JS

The CP model for addressing the JS problem employs global constraints, thereby eliminating the need for Big θ method used in the MIP model.

$$
\begin{aligned}
&\min C_{\max} \\
&\text{s.t.} \\
&\quad \text{start}_{t'} = \text{end}_t, \quad \forall t \in \mathcal{T}, t \neq t' && (14)\\
&\quad \text{NoOverlap}(S_m), \quad \forall m \in \mathcal{M}. && (15)\\
&\quad \text{Sequencevar}(\sigma_{ij}) && (16)\\
&\quad \text{makespan} \geq \text{end}_t, \quad \forall t \in \mathcal{T}. && (17)
\end{aligned}
$$

where:

- start-end dependency: ensures the start time of a dependent task aligns with the end time of its predecessor
- machine capacity constraint (*NoOverlap*): prevents tasks from overlapping on the same machine
- task sequencing (*SequenceVar*): maintains the proper sequence of tasks on machine

3.3 Mathematical modeling for the OS

In the OS problem, the primary distinction lies in the relaxation of the strict sequencing constraint required in the JS scheduling problem, as evidenced by the absence of the *SequenceVar* constraint present in the latter. The CP model employed for the OS problem is:

$$
\begin{aligned}
&\min C_{\max} \\
&\text{s.t.} \\
&\quad \text{start}_{t_1} = \text{end}_{t_2}, \quad \forall t \in \mathcal{T}, t_1 \neq t_2 && (18)\\
&\quad \text{NoOverlap}(S_m), \quad \forall m \in \mathcal{M}. && (19)\\
&\quad \text{NoOverlap}(S_j), \quad \forall j \in \mathcal{J}. && (20)\\
&\quad \text{makespan} \geq \text{end}_t, \quad \forall t \in \mathcal{T}. && (21)
\end{aligned}
$$

where:

- start-end dependency: ensures the start time of a dependent task aligns with the end time of its predecessor
- machine capacity constraint (*NoOverlap*): prevents tasks from overlapping on the same machine

4 Results

In this project, we leveraged classical benchmark instances for both the JS and OS problems to evaluate the performance of our models. For the JS problem, we retrieve instances from [10], corresponding to Taillard's benchmark instances available on Moodle, covering instance sets of varying sizes: 15x15, 20x15, 20x20, 30x15, 30x20, 50x15, 50x20, and 100x20. Each set comprised 10 distinct problem instances, all of which are detailed in Appendix 1. For the OS problem, we used Taillard's benchmark instance sets from Moodle, which exclusively consist of rectangular problems (i.e., where the number of jobs j equals the number of machines m). These included sizes 4x4, 5x5, 7x7, 10x10, 15x15, and 20x20, with 10 problem instances per set. We evaluated the performance of the MIP and CP models under two time limits: 100 and 200 seconds. If an optimal solution (i.e., a makespan equal to the lower bound) was found, the solver terminated

early, denoted in our tables by a 0% gap. The solver uses the lower bound as a benchmark to determine optimality; when the objective value equals this bound, the solution is deemed optimal.

Appendix 2 summarizes the outcomes for the JS problem under a 100 second time limit. In this scenario, the CP model significantly outperformed the MIP model. CP achieved optimality in 23.8% of instances (19 out of 80), whereas MIP failed to find an optimal solution for any instance within the time limit. For problems where the time limit was reached, CP consistently achieved smaller gaps, with the highest being 12.05%, compared to MIP's maximum gap of 98.48% (minimum of 11.87%). Interestingly, CP's performance varied across instance sizes. It solved 35% of smaller instances (15x15 and 20x15) optimally, but this percentage declined as the problem size increased, reaching 0% for the largest sizes - except for 50x15 problems where it solved 100% of the instances (Table 1). Notably, the CP model maintained a consistent gap percentage across problem sizes, while MIP's gap increased proportionally with problem size. Overall, CP demonstrated superior performance in minimizing makespan, a trend that became more pronounced with larger instances

Instance Set	**Percentage Optimal** (%)
15x15	50
20x15	20
20x20	0
30x15	10
30x20	0
50x15	100
50x20	10
100x20	0

Table 1. CP model optimality percentage for various instance sets sizes in the JS problem with 100s time limit.

Extending the time limit to 200 seconds (Appendix 3) yielded some interesting insights. Contrary to expectations, the MIP model still failed to achieve optimality for any instance. Its performance metrics, including the gap values, showed little improvement, particularly for larger problems (e.g., a maximum gap of 98.7% for the ta79-100x20 instance and a minimum gap of 7.86% for the ta04-15x15 instance). On the other hand, the CP model showed a modest improvement, solving 33.75% of instances optimally, compared to 23.8% under the 100-second limit. The maximum gap slightly increased to 13.09%, but CP consistently delivered smaller gaps, even for the largest instances. While CP's optimality rates did not change for medium-sized problems (20x15, 20x20, 30x15, and 30x20), improvements were observed for both the smallest (15x15) and largest

instance sizes (Table 2). In accordance with the previous tests, makespan are minimized more efficiently using the CP over the MIP model.

Instance Set	**Percentage Optimal** (%)
15x15	80
20x15	20
20x20	0
30x15	10
30x20	0
50x15	100
50x20	30
100x20	30

Table 2. CP model optimality percentage for various instance sets sizes in the JS problem with 200s time limit.

Appendices 5 and 6 present the results for the OS problems under 100 second and 200 second time limits, respectively. In this context, the CP model demonstrates markedly superior performance compared to the MIP model for instances where m and $j > 7$. Across all OS scenarios, CP achieved a 100% optimality rate, while MIP struggled with instances larger than 5x5.

Instance Set	**Percentage Optimal** (%)
4x4	100
5x5	100
7x7	10
15x15	0
20x20	0

Table 3. MIP model optimality percentage for various instance sets sizes in the OS problem with 100s time limit.

Under the 100 second time limit, CP achieved optimal solutions for all instances, highlighting its clear advantage in solving OS problems. In contrast, MIP achieved 100% optimality for 4x4 and 5x5 instances and only 10% for 7x7 instances, failing to find any solutions for larger problems (Table 3); the maximum gap for the MIP model at this time limit was 56.6%. Extending the time limit to 200 seconds yielded no significant improvement for MIP, with the same optimality rates observed as under the 100 second limit, plus one additional optimal solution in the 7x7 instances. However, the maximum gap increased to 75.1%. The CP model maintained its 100% optimality rate, as expected. How-

ever, it is important to note that for larger instances, specifically 15x15 and 20x20 cases, the solver was unable to identify any feasible solution using MIP.

In terms of solving speed, the CP model was consistently faster. Under the 200 second time limit, for instances where both models achieved optimality, CP had a mean time to optimality of 0.26 seconds, compared to 3.85 seconds for MIP. Similarly, with a 100 second time limit, CP maintained a mean time to optimality of 0.3 seconds, while MIP required an average of 1.7 seconds.

5 Conclusions

In this report, we compared MIP and CP for solving JS and OS scheduling problems.

Our findings highlight a clear advantage of CP over MIP for the JS problem. CP consistently outperformed MIP in both proving optimality and achieving higher-quality solutions, regardless of the instance set size. For the OS problem, the results were more nuanced. Due to the absence of precedence constraints, which introduces greater flexibility, both CP and MIP performed comparably well in finding feasible solutions for smaller problem instances. However, as the problem size increased, CP demonstrated superior performance, breaking the trend and asserting its advantage over MIP.

Altogether, CP's robust handling of combinatorial complexity makes it a preferred method for tackling real-world scheduling challenges.

References

1. John F. Muth and Gerald Luther Thompson, *Industrial Scheduling*, Prentice-Hall, 1963.
2. Matthias Mnich, René van Bevern, *Parameterized complexity of machine scheduling: 15 open problems*, 2018.
3. Giacomo Da Col and Erich Teppan, *Google vs IBM: A Constraint Solving Challenge on the Job-Shop Scheduling Problem*, 2019.
4. Zilong Zhuang, et al., *A Heuristic Rule Based on Complex Network for Open Shop Scheduling Problem With Sequence-Dependent Setup Times and Delivery Times*, IEEE Access
5. Norman M. Sadeh and Mark S. Fox, *Variable and Value Ordering Heuristics for the Job Shop Scheduling Constraint Satisfaction Problem*, Artificial Intelligence, vol. 86, 1996, pp. 1–41, https://doi.org/10.1016/0004-3702(95)00098-4.
6. Michael L. Pinedo, *Scheduling. Theory, Algorithms, and Systems*, 2022.
7. Stephen P. Bradley, et al., *Applied Mathematical Programming*, 19th ed., Addison-Wesley, 1992.
8. Wen-Yang Ku and J. Christopher Beck, *Mixed Integer Programming Models for Job Shop Scheduling: A Computational Analysis*, Computers & Operations Research, vol. 73, Sept. 2016
9. Marriott, Kim, and Peter J. Stuckey. *Programming with Constraints: An Introduction*. MIT Press, 1999.
10. https://github.com/wsgisler/job-shop-scheduling/tree/master/instances

6 Appendix

6.1 Appendix 1

Instances	Instance Set	Instances	Instance Set
ta01	15x15	ta41	30x20
ta02	15x15	ta42	30x20
ta03	15x15	ta43	30x20
ta04	15x15	ta44	30x20
ta05	15x15	ta45	30x20
ta06	15x15	ta46	30x20
ta07	15x15	ta47	30x20
ta08	15x15	ta48	30x20
ta09	15x15	ta49	30x20
ta10	15x15	ta50	30x20
ta11	20x15	ta51	50x15
ta12	20x15	ta52	50x15
ta13	20x15	ta53	50x15
ta14	20x15	ta54	50x15
ta15	20x15	ta55	50x15
ta16	20x15	ta56	50x15
ta17	20x15	ta57	50x15
ta18	20x15	ta58	50x15
ta19	20x15	ta59	50x15
ta20	20x15	ta60	50x15
ta21	20x20	ta61	50x20
ta22	20x20	ta62	50x20
ta23	20x20	ta63	50x20
ta24	20x20	ta64	50x20
ta25	20x20	ta65	50x20
ta26	20x20	ta66	50x20
ta27	20x20	ta67	50x20
ta28	20x20	ta68	50x20
ta29	20x20	ta69	50x20
ta30	20x20	ta70	50x20
ta31	30x15	ta71	100x20
ta32	30x15	ta72	100x20
ta33	30x15	ta73	100x20
ta34	30x15	ta74	100x20
ta35	30x15	ta75	100x20
ta36	30x15	ta76	100x20
ta37	30x15	ta77	100x20
ta38	30x15	ta78	100x20
ta39	30x15	ta79	100x20
ta40	30x15	ta80	100x20

Table 4. Mapping of instances to instance sets for the JS problem.

6.2 Appendix 2

Instances	CP model				MIP model			
	Time	Upper Bound	Lower Bound	Gap (%)	Time	Upper Bound	Lower Bound	Gap (%)
ta01	22.48818588	1231	1231	0	100.3065169	1257	1088.69	13.39
ta02	76.86035895	1244	1244	0	100.2886755	1271	1029	19.04
ta03	45.39319038	1218	1218	0	100.2995234	1271	1081	14.95
ta04	48.87720442	1175	1175	0	100.2847202	1188	1047.01	11.87
ta05	100.051461	1231	1193	3.09	100.4026706	1270	1058	16.69
ta06	100.0516706	1243	1160	6.68	100.2923999	1287	1022.14	20.58
ta07	100.048414	1236	1193	3.48	100.2760041	1264	1052	16.77
ta08	100.0420589	1226	1169	4.65	100.279983	1238	1080.96	12.68
ta09	100.1399124	1274	1218	4.4	100.2743316	1350	1116.39	17.3
ta10	53.40867376	1241	1241	0	100.2754614	1304	1088.43	16.53
ta11	100.0584598	1399	1282	8.36	100.4979215	1526	1050.11	31.19
ta12	100.1414804	1375	1314	4.44	100.477587	1523	1058	30.53
ta13	100.061461	1363	1243	8.8	100.4799767	1502	1028.01	31.56
ta14	13.50846887	1345	1345	0	100.5722136	1426	1011	29.1
ta15	100.0496621	1375	1267	7.85	100.4845519	1458	995	31.76
ta16	100.0620246	1363	1254	8	100.5834846	1485	1057.92	28.76
ta17	84.21710348	1462	1462	0	100.4848652	1665	1110.38	33.31
ta18	100.0604799	1415	1358	4.03	100.5672035	1563	1017.12	34.93
ta19	100.0482616	1368	1252	8.48	100.4783871	1565	1003.62	35.87
ta20	100.0627861	1372	1289	6.05	100.5688434	1455	1079.57	25.8
ta21	100.0749648	1667	1512	9.3	100.6491258	1857	1385	25.42
ta22	100.06828	1638	1459	10.93	100.7381492	1735	1274	26.57
ta23	100.0690203	1618	1450	10.38	100.6247168	1767	1291	26.94
ta24	100.1427975	1668	1572	5.76	100.611124	1798	1272.29	29.24
ta25	100.0910242	1643	1474	10.29	100.6745217	1798	1306.29	27.35
ta26	100.0711303	1684	1494	11.28	100.6184978	1851	1316	28.9
ta27	100.0739603	1742	1592	8.61	100.6961436	1826	1363.12	25.35
ta28	100.0669565	1618	1552	4.08	100.7135551	1850	1327	28.27
ta29	100.065706	1656	1465	11.53	100.6112916	1747	1352.05	22.61
ta30	100.063062	1626	1430	12.05	100.6668649	1751	1305	25.47
ta31	100.0670581	1771	1764	0.4	101.2591257	2174	1037.84	52.26
ta32	100.0828466	1841	1774	3.64	101.0917437	2249	1024.79	54.43
ta33	100.0805976	1834	1774	3.27	101.1219287	2333	1083.25	53.57
ta34	100.0726669	1879	1828	2.71	101.2134001	2307	1024.02	55.61
ta35	64.23992729	2007	2007	0	101.0959933	2285	1046	54.22
ta36	100.0899663	1864	1819	2.41	101.1461475	2254	1041.03	53.81
ta37	100.0828602	1800	1771	1.61	101.0508363	2097	1081.57	48.42
ta38	100.1905305	1708	1673	2.05	31315.80597	2019	996.41	50.65
ta39	100.132962	1808	1795	0.72	101.053122	2332	938	59.78
ta40	100.0863235	1727	1631	5.56	101.207078	2199	1008	54.16
ta41	100.1016123	2086	1861	10.79	101.4699795	2655	1301	51
ta42	100.1059256	2001	1867	6.7	101.5897388	2736	1248	54.39
ta43	100.1062226	1912	1809	5.39	101.7709277	2490	1263.7	49.25
ta44	100.108166	2057	1923	6.51	101.4374425	2615	1281	51.01
ta45	100.1821518	2047	1990	2.78	101.3934963	2502	1285.1	48.64
ta46	100.0958989	2079	1940	6.69	101.6901069	2692	1355	49.67
ta47	100.0980575	1976	1787	9.56	101.8564055	2527	1374.42	45.61
ta48	100.0953946	2039	1905	6.57	101.4746616	2502	1304.54	47.86
ta49	99.24484777	2070	1903	8.07	101.4564574	2584	1237.57	52.11
ta50	100.1115136	2000	1808	9.6	101.4392738	2549	1302.72	48.89
ta51	23.15309095	2760	2760	0	103.0291035	3828	1015	73.48
ta52	37.3810854	2756	2756	0	103.037287	4048	1046	74.16
ta53	28.50673008	2717	2717	0	103.0436053	3723	995	73.27
ta54	14.72426248	2839	2839	0	103.0590243	3907	1144	70.72
ta55	56.15976048	2679	2679	0	103.0413461	3987	1034	74.07
ta56	71.1757319	2781	2781	0	103.3302321	28959	989	96.58
ta57	18.52636433	2943	2943	0	103.0630727	27125	1137	95.81
ta58	22.3669672	2885	2885	0	102.9448984	4193	1051	74.93
ta59	65.08778191	2655	2655	0	102.9808152	24495	1054	95.7
ta60	34.8099699	2723	2723	0	103.0155637	3820	1116	70.79
ta61	100.3435459	2928	2868	2.05	104.0518577	38522	1298	96.63
ta62	100.18835	2931	2869	2.12	104.0362699	4333	1357	68.68
ta63	100.1897562	2810	2755	1.96	103.9973812	40493	1310	96.76
ta64	100.3478668	2748	2702	1.67	103.912281	34951	1283	96.33
ta65	100.1928687	2780	2725	1.98	104.1694477	36610	1289	96.48
ta66	100.1870756	2865	2845	0.7	104.0084698	38195	1317	96.55
ta67	100.3601298	2871	2825	1.6	103.8624783	4664	1311	71.89
ta68	74.03055024	2784	2784	0	104.0373685	4138	1369	66.92
ta69	100.1872768	3141	3071	2.23	104.020704	36701	1416	96.14
ta70	100.344954	3076	2995	2.63	103.944972	37278	1286	96.55
ta71	100.2321658	5630	5464	2.95	115.5850513	81955	1365.15	98.33
ta72	100.2183416	5243	5181	1.18	115.9091187	78390	1292.07	98.35
ta73	100.2209334	5638	5568	1.24	116.4599159	80294	1381	98.28
ta74	100.235821	5348	5339	0.17	115.9438212	79382	1328.15	98.33
ta75	100.1997309	5622	5392	4.09	115.874151	78148	1360.68	98.26
ta76	100.2259352	5494	5342	2.77	115.873745	79489	1282.25	98.39
ta77	100.1992059	5543	5436	1.93	115.8470175	83329	1264.62	98.48
ta78	100.223906	5511	5394	2.12	116.2953899	77775	1327.27	98.29
ta79	100.2166042	5396	5358	0.7	115.8976345	78579	1302.03	98.34
ta80	100.2206953	5240	5183	1.09	116.5359275	76347	1268.17	98.34

Table 5. Results of the CP and MIP models applied to the JS problem using a time limit of 100s.

6.3 Appendix 3

Instances	CP model				MIP model			
	Time	Upper Bound	Lower Bound	Gap (%)	Time	Upper Bound	Lower Bound	Gap (%)
ta01	18.51233029	1231	1231	0	200.3054307	1248	1098.86	11.95
ta02	55.48768306	1244	1244	0	200.2746029	1251	1054	15.75
ta03	28.61849427	1218	1218	0	200.2926743	1254	1094	12.76
ta04	56.51097584	1175	1175	0	200.292742	1183	1090	7.86
ta05	1080.061325	1224	1176	3.92	200.409128	1259	1086.69	13.69
ta06	200.0532234	1244	1155	7.15	200.2793314	1296	1026.81	20.77
ta07	155.2585359	1227	1227	0	200.3259592	1254	1058	15.63
ta08	129.6162121	1217	1217	0	200.2775412	1226	1081.47	11.79
ta09	174.5911696	1274	1274	0	200.2895911	1319	1149.53	12.85
ta10	56.44518781	1241	1241	0	200.2745199	1282	1110	13.42
ta11	200.0659184	1391	1287	7.48	200.471709	1547	1089	29.61
ta12	200.2106967	1392	1333	4.24	200.4727731	1465	1060.39	27.62
ta13	200.0660596	1392	1243	10.7	200.4756427	1462	1054	27.91
ta14	116.5302918	1345	1345	0	200.5922577	1400	1028	26.57
ta15	200.0734746	1387	1266	8.72	200.4731338	1450	997	31.24
ta16	200.0571697	1376	1252	9.01	200.5465996	1496	1099	26.54
ta17	116.0841703	1462	1462	0	200.4776354	1559	1130	27.52
ta18	200.0680749	1415	1358	4.03	200.5726759	1508	1038.43	31.14
ta19	200.7265303	1362	1251	8.15	200.4719577	1495	1004.67	32.8
ta20	200.0633352	1368	1298	5.12	200.5583229	1432	1074.07	25
ta21	200.0732312	1664	1525	8.35	200.6293902	1815	1389.59	23.44
ta22	200.0767901	1646	1465	11	200.7323084	1735	1294.74	25.38
ta23	200.064188	1603	1442	10.04	200.6293812	1653	1297.76	21.49
ta24	200.1655343	1666	1597	4.14	200.630609	1791	1316.91	26.47
ta25	200.074677	1628	1474	9.46	200.6976821	1733	1300	24.99
ta26	200.0728891	1713	1496	12.67	200.6240616	1784	1309.23	26.61
ta27	200.0779669	1719	1590	7.5	200.7170858	1846	1373.55	25.59
ta28	200.0758917	1631	1552	4.84	200.6156678	1735	1344.93	22.48
ta29	200.0702233	1640	1464	10.73	200.6296556	1734	1360.4	21.55
ta30	200.0745039	1642	1427	13.09	200.6889677	1725	1295.19	24.92
ta31	200.0774024	1780	1764	0.9	201.09496	2269	1045	53.94
ta32	200.0805838	1852	1774	4.21	201.1028402	2243	1033	53.95
ta33	200.0957706	1831	1774	3.11	201.1121137	2311	1077	53.4
ta34	200.0836585	1891	1828	3.33	201.1321335	2220	1037	53.29
ta35	33.13963723	2007	2007	0	201.0875549	2349	1019.98	56.58
ta36	200.0811365	1837	1819	0.98	201.3952539	2192	1044.59	52.35
ta37	200.0820079	1811	1771	2.21	201.0416596	2252	1097	51.29
ta38	200.180001	1700	1673	1.59	201.0295651	2084	1029	50.62
ta39	200.1480787	1806	1795	0.61	201.0178585	2248	945	57.96
ta40	200.0899463	1750	1631	6.8	201.104593	2068	1015	50.92
ta41	200.1109719	2137	1861	12.92	201.4333975	2620	1315	49.81
ta42	200.0824988	2005	1867	6.88	201.3870885	2558	1253	51.02
ta43	200.1074371	1916	1809	5.58	201.3956149	2514	1277	49.2
ta44	200.1004205	2051	1923	6.24	201.3993993	2623	1282.9	51.09
ta45	200.1771419	2038	1990	2.36	201.3893437	2531	1285.63	49.2
ta46	200.0997367	2087	1940	7.04	201.6354108	2571	1354.77	47.31
ta47	200.1146219	1973	1787	9.43	201.3830621	2499	1376	44.94
ta48	200.107064	2002	1905	4.85	201.4577115	2608	1317	49.5
ta49	200.106575	2050	1903	7.17	201.4599576	2599	1240	52.29
ta50	200.1071835	2026	1809	10.71	201.4350226	2456	1305.22	46.86
ta51	50.35921097	2760	2760	0	202.9894438	3835	1022	73.35
ta52	50.48763967	2756	2756	0	202.900039	24274	1031	95.75
ta53	37.68022346	2717	2717	0	202.9605069	3885	995	74.39
ta54	14.28488564	2839	2839	0	203.0483618	4153	1144	72.45
ta55	82.49034095	2679	2679	0	202.9696429	4058	1043	74.3
ta56	34.06646705	2781	2781	0	202.9419475	3920	989	74.77
ta57	19.32026982	2943	2943	0	203.0519991	4014	1137	71.67
ta58	22.69480133	2885	2885	0	202.9467678	3974	1051	73.55
ta59	59.494488	2655	2655	0	202.9986548	4063	1054	74.06
ta60	34.73710752	2723	2723	0	202.9863627	3792	1116	70.57
ta61	141.9606428	2868	2868	0	203.8689296	37320	1307	96.5
ta62	200.2020493	2930	2869	2.08	204.0304234	4345	1365	68.58
ta63	199.8209629	2755	2755	0	204.035516	4048	1310	67.64
ta64	117.2968814	2702	2702	0	203.8928494	4187	1283	69.36
ta65	200.1863921	2777	2725	1.87	203.9926181	36285	1270	96.5
ta66	200.1992638	2846	2845	0.04	204.0171545	38694	1317	96.6
ta67	200.3612134	2826	2825	0.04	203.9417503	4331	1311	69.73
ta68	200.1855552	2794	2784	0.36	204.0385051	4112	1343	67.34
ta69	200.1743245	3086	3071	0.49	203.9981296	4444	1416	68.14
ta70	200.3880858	3047	2995	1.71	203.8265829	4404	1286	70.8
ta71	200.2250049	5611	5464	2.62	215.6282616	81289	1405	98.27
ta72	193.2295082	5181	5181	0	215.6964333	78309	1292.07	98.35
ta73	176.4414234	5568	5568	0	216.0723901	80522	1381	98.28
ta74	195.7635953	5339	5339	0	215.7940145	79036	1328.15	98.32
ta75	200.2411535	5420	5392	0.52	215.7769668	78084	1360.68	98.26
ta76	200.2189264	5428	5342	1.58	215.6805248	79244	1282.25	98.38
ta77	200.2281504	5479	5436	0.78	215.6396954	82775	1273.79	98.46
ta78	200.243448	5467	5394	1.34	215.8023124	77235	1327.27	98.28
ta79	200.2105188	5386	5358	0.52	1117.955521	100000	1302	98.7
ta80	200.2293973	5229	5183	0.88	215.6567276	75883	1268.17	98.33

Table 6. Results of the CP and MIP models applied to the JS problem using a time limit of 200s.

6.4 Appendix 4

Instances	Instance Set	Instances	Instance Set
ta01	4x4	ta41	15x15
ta02	4x4	ta42	15x15
ta03	4x4	ta43	15x15
ta04	4x4	ta44	15x15
ta05	4x4	ta45	15x15
ta06	4x4	ta46	15x15
ta07	4x4	ta47	15x15
ta08	4x4	ta48	15x15
ta09	4x4	ta49	15x15
ta10	4x4	ta50	15x15
ta11	5x5	ta51	20x20
ta12	5x5	ta52	20x20
ta13	5x5	ta53	20x20
ta14	5x5	ta54	20x20
ta15	5x5	ta55	20x20
ta16	5x5	ta56	20x20
ta17	5x5	ta57	20x20
ta18	5x5	ta58	20x20
ta19	5x5	ta59	20x20
ta20	5x5	ta60	20x20
ta21	7x7		
ta22	7x7		
ta23	7x7		
ta24	7x7		
ta25	7x7		
ta26	7x7		
ta27	7x7		
ta28	7x7		
ta29	7x7		
ta30	7x7		
ta31	10x10		
ta32	10x10		
ta33	10x10		
ta34	10x10		
ta35	10x10		
ta36	10x10		
ta37	10x10		
ta38	10x10		
ta39	10x10		
ta40	10x10		

Table 7. Mapping of instances to instance sets for the OS problem.

6.5 Appendix 5

Instances	CP model				MIP model			
	Time	Upper Bound	Lower Bound	Gap (%)	Time	Upper Bound	Lower Bound	Gap (%)
ta01	0.1006491	193	193	0	0.0906596	193	193	0
ta02	0.0862677	236	236	0	0.1158950	236	236	0
ta03	0.1368396	271	271	0	0.0747647	271	271	0
ta04	0.0550286	250	250	0	0.0956037	250	250	0
ta05	0.1270985	295	295	0	0.0968563	295	295	0
ta06	0.0376791	189	189	0	0.1157805	189	189	0
ta07	0.0807743	201	201	0	0.0811996	201	201	0
ta08	0.0408592	217	217	0	0.0888731	217	217	0
ta09	0.0823192	261	261	0	0.0811493	261	261	0
ta10	0.1252555	217	217	0	0.0889456	217	217	0
ta11	0.1647424	300	300	0	0.4054968	300	300	0
ta12	0.2004761	262	262	0	0.2630078	262	262	0
ta13	0.2065279	323	323	0	0.5270519	323	323	0
ta14	0.2380971	310	310	0	0.3538005	310	310	0
ta15	0.3037588	326	326	0	0.4747276	326	326	0
ta16	0.1775069	312	312	0	0.5084893	312	312	0
ta17	0.2413866	303	303	0	0.3519642	303	303	0
ta18	0.3157806	300	300	0	0.3957009	300	300	0
ta19	0.1772534	353	353	0	0.4984693	353	353	0
ta20	0.3388664	326	326	0	0.3803575	326	326	0
ta21	0.2882547	435	435	0	100.118418	435	417	4.14
ta22	0.4047939	443	443	0	100.113492	443	428	3.39
ta23	0.9249517	468	468	0	100.143876	468	424	9.4
ta24	0.0575742	463	463	0	100.115307	463	443	4.32
ta25	0.0933043	416	416	0	100.344317	416	355	14.66
ta26	1.2694685	451	451	0	100.117269	451	435.14	3.52
ta27	0.4031074	422	422	0	100.11526	422	359	14.93
ta28	0.1767065	424	424	0	100.132955	424	353	16.75
ta29	0.0654983	458	458	0	100.131543	458	406	11.35
ta30	0.0533919	398	398	0	4.758657	398	398	0
ta31	0.7504172	637	637	0	100.319692	641	356.34	44.41
ta32	0.0671584	588	588	0	100.326508	592	281	52.53
ta33	0.2188613	598	598	0	100.608188	607	290	52.22
ta34	0.0922427	577	577	0	100.320094	577	299	48.18
ta35	0.4087173	640	640	0	100.309162	656	350.96	46.5
ta36	0.0721917	538	538	0	100.313734	538	352	34.57
ta37	0.6094648	616	616	0	100.459107	616	366	40.58
ta38	0.2500381	595	595	0	100.305692	598	363	39.3
ta39	0.0804352	595	595	0	100.300702	606	263	56.6
ta40	0.107727	596	596	0	100.323498	599	322	46.24
ta41	0.1768329	937	937	0				
ta42	0.1298930	918	918	0				
ta43	0.1435914	871	871	0				
ta44	0.1614716	934	934	0				
ta45	0.1202583	946	946	0				
ta46	0.1106529	933	933	0				
ta47	0.2230193	891	891	0				
ta48	0.1516675	893	893	0				
ta49	0.3085224	899	899	0				
ta50	0.4433326	902	902	0				
ta51	0.3192086	1155	1155	0				
ta52	0.8657312	1241	1241	0				
ta53	0.3672554	1257	1257	0				
ta54	0.4166934	1248	1248	0				
ta55	0.2606329	1256	1256	0				
ta56	0.2779159	1204	1204	0				
ta57	0.3096587	1294	1294	0				
ta58	3.3111159	1169	1169	0				
ta59	0.2176589	1289	1289	0				
ta60	0.2635617	1241	1241	0				

Table 8. Results of the CP and MIP models applied to the OS problem using a time limit of 100s.

6.6 Appendix 6

Instances	CP model				MIP model			
	Time	Upper Bound	Lower Bound	Gap (%)	Time	Upper Bound	Lower Bound	Gap (%)
ta01	1.343800	193	193	0	0.0906596	193	193	0
ta02	0.099934	236	236	0	0.1158950	236	236	0
ta03	0.113624	271	271	0	0.0747647	271	271	0
ta04	0.052944	250	250	0	0.0956037	250	250	0
ta05	0.109369	295	295	0	0.0968563	295	295	0
ta06	0.041220	189	189	0	0.1157805	189	189	0
ta07	0.096696	201	201	0	0.0811996	201	201	0
ta08	0.048264	217	217	0	0.0888731	217	217	0
ta09	0.094825	261	261	0	0.0811493	261	261	0
ta10	0.123415	217	217	0	0.0889456	217	217	0
ta11	0.164275	300	300	0	0.4054968	300	300	0
ta12	0.180357	262	262	0	0.2630078	262	262	0
ta13	0.163485	323	323	0	0.5270519	323	323	0
ta14	0.182533	310	310	0	0.3538005	310	310	0
ta15	0.204384	326	326	0	0.4747276	326	326	0
ta16	0.175975	312	312	0	0.5084893	312	312	0
ta17	0.179725	303	303	0	0.3519642	303	303	0
ta18	0.224996	300	300	0	0.3957009	300	300	0
ta19	0.164643	353	353	0	0.4984693	353	353	0
ta20	0.247556	326	326	0	0.3803575	326	326	0
ta21	0.042837	435	435	0	200.10910	435	434	0.23
ta22	0.398388	443	443	0	200.11082	443	427	3.61
ta23	0.397033	468	468	0	52.085301	468	468	0
ta24	0.048973	463	463	0	200.11262	463	443	4.32
ta25	0.156310	416	416	0	200.29730	416	351	15.62
ta26	0.882272	451	451	0	200.11385	451	438	2.88
ta27	0.253048	422	422	0	200.11012	422	358	15.17
ta28	0.210859	424	424	0	200.10572	424	345	18.63
ta29	0.114043	458	458	0	200.13053	458	449	1.97
ta30	0.043550	398	398	0	27.471970	398	398	0
ta31	0.509186	637	637	0	200.31038	639	371	41.94
ta32	0.138138	588	588	0	200.30019	588	357.19	39.25
ta33	0.496158	598	598	0	200.76258	605	347.76	42.52
ta34	0.072205	577	577	0	200.28879	577	352	38.99
ta35	0.429858	640	640	0	200.29484	645	381.65	40.83
ta36	0.064152	538	538	0	200.29584	538	373.65	30.55
ta37	0.088027	616	616	0	200.58349	616	322	47.73
ta38	0.144644	595	595	0	200.28936	595	303.35	49.02
ta39	0.072017	595	595	0	200.30825	595	378.28	36.42
ta40	0.599141	596	596	0	200.30062	596	315	47.15
ta41	0.160313	937	937	0	201.18339	995	248.26	75.05
ta42	0.150203	918	918	0				
ta43	0.128451	871	871	0				
ta44	0.127479	934	934	0				
ta45	0.126269	946	946	0				
ta46	0.128425	933	933	0				
ta47	0.136462	891	891	0				
ta48	0.137649	893	893	0				
ta49	0.180589	899	899	0				
ta50	0.161088	902	902	0				
ta51	0.914045	1155	1155	0				
ta52	0.306743	1241	1241	0				
ta53	0.213053	1257	1257	0				
ta54	0.284412	1248	1248	0				
ta55	0.215802	1256	1256	0				
ta56	0.238846	1204	1204	0				
ta57	0.604022	1294	1294	0				
ta58	0.835934	1169	1169	0				
ta59	0.238191	1289	1289	0				
ta60	0.269581	1241	1241	0				

Table 9. Results of the CP and MIP models applied to the OS problem using a time limit of 200s.