Supplementary material to the paper titled "Assembly Planning by Disjunctive Programming and Geometrical Reasoning"

Markó Horváth[a], Tamás Kis[a], András Kovács[a], and Márk Fekula[a]

[a]Institute for Computer Science and Control, Kende str. 13-17, Budapest, H-1111, Hungary

1 Instances

1.1 Generating procedure

Random instances of given size and characteristics are generated using the tree-structured representation of the assembly planning problem depicted in Figure 1. The underlying idea of this structure is to generate instances where subassemblies must be assembled from the bottom to the top.

The given number of parts are randomly divided into subassemblies with cardinality between $size_{\min}$ and $size_{\max}$, and with each subassembly, a single node is associated (see the gray leaves in Figure 1). Two procedures are used to build the tree. In case of a *fitting procedure*, a fitting node is created and added to the tree as the parent of two selected root nodes (i.e., nodes without ancestors). In case of a *joining procedure*, two nodes, a tool node (see the white leaves in Figure 1) and a joining node are created and added to the tree so that the joining node becomes the parent of the created tool node and the selected non-tool root node. The following steps are applied to build the tree: (1) If there is a single root node, and it is a joining node, then stop. (2) If there is a single root node (note that it is a fitting node), a joining procedure is applied, then stop. (3) A root node, say r, is chosen randomly. (4) If r is not a joining node, with $p_{joining}$ probability a joining procedure is applied on r, then go to step 1. (5) Another root node, say r', is chosen randomly and a fitting procedure is applied on r and r', then go to step 1.

The constituents of the instances are the elements of the sets represented by the leaf nodes (subassemblies and tools). For each subassembly with probability $p_{newfixt}$ a new fixture is created with

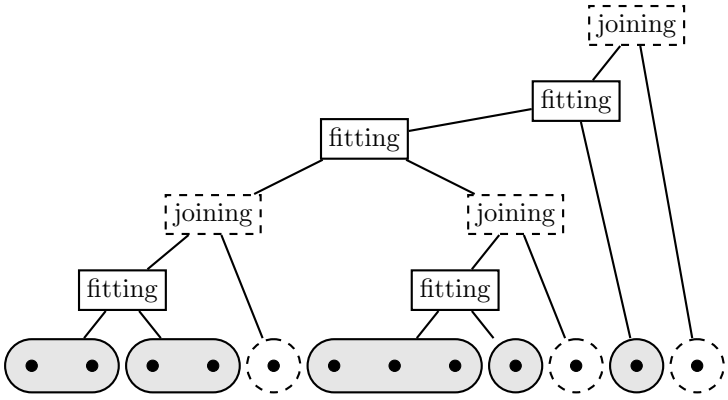


Figure 1: Structure of the randomly generated assembly problem instances. Leaf nodes represent preselected subassemblies (gray sets) and tool-parts (dashed sets). Fitting nodes represent the fitting of the corresponding two subassemblies by human hand. Joining nodes represent the joining of the corresponding subassembly using a tool.

changeover time uniformly chosen from $[t_{\min}^{fixt}, t_{\max}^{fixt}]$ such that its grasped part is randomly chosen from the subassembly, unless the number of existing fixtures exceeds a given limit ($n_{\max}^{fixtures}$). If no fixture exists after the procedure, a single one is created to make the instance feasible. Note that at the end of the full generation procedure, to keep instances feasible even in the presence of conjunctive and disjunctive constraints, the number of fixtures can exceed the given upper limit (see later).

For each pair of parts of a subassembly node a task is created with processing time uniformly chosen from $[t_{\min}^{fitting}, t_{\max}^{fitting}]$ fitting these parts together, however, not all of these tasks are added to the problem. For each subassembly a base part is chosen, and tasks fitting this base part with a non-base part is added to the problem. Then $\min\{\text{number of all extra tasks}, \lfloor |V| \times m^{extra} \rfloor\}$ randomly chosen extra tasks are also added to the problem. Note that without any extra tasks, the underlying liaison graph is a tree.

For each fitting node, a task is created with processing time uniformly chosen from $[t_{\min}^{fitting}, t_{\max}^{fitting}]$, with parts randomly selected from the parts of the subtrees rooted at the corresponding two child nodes, respectively, and with the human hand as the only admissible tool with changeover time 1.0.

For each joining node, a task is also created with processing time uniformly chosen from $[t_{\min}^{joining}, t_{\max}^{joining}]$, and parts randomly selected from the parts of the subtrees rooted at the corresponding two child nodes, respectively. If the number of existing tools does not exceed a predefined limit ($n_{\max}^{tools}$), a new tool is also created with changeover time uniformly chosen from $[t_{\min}^{tool}, t_{\max}^{tool}]$. Otherwise, an existing one is chosen randomly, and it is assigned to the task as the only admissible tool. If no fixture exists in the corresponding subtree after the procedure, a single one is created to make the instance feasible even in the case of conjunctive constraints (see later).

In case of conjunctive constraints, for each joining node a constraint $\langle t \oplus W \rangle$ is created, where t is the task associated with the node and W is the set of parts in the left-hand side subtree. Note that the meaning of such a constraint is that some parts must be fitted together before a joining (for example screwing) procedure. If the parent of a joining node is a fitting node, then a constraint $\langle t \oplus W \rangle$ is also created where t is the task associated with the fitting node and W is the set of parts corresponding to the joining node subtree. Note that the meaning of such a constraint is that some parts must be joined before they are fitted to another subassembly.

For the computational experiments, the disjunctive constraints are also generated a priori by the problem generator. For each joining node u such that its parent node is also a joining node and there are at least two graspable parts of its subtree, a disjunctive constraint $\langle t \ominus W_1, W_2, z, f \rangle$ is created where t and z are the task and the tool associated with u, respectively, and fixture f is randomly chosen from the possible ones. If node u is the left child of its parent, then W_1 is empty, and W_2 is the set of parts of its sibling subtree (i.e., the subtree rooted at the right child), otherwise W_2 is empty, and W_1 is the set of parts of its sibling subtree.

1.2 Instances families

Preliminary experiments were performed in order to identify hard instances by tuning the parameters of the problem generator. For instance, it was found that increasing the number of fixtures makes the problem harder than increasing the number of tools.

As it is impossible to test all combinations of the parameters, in all cases the following settings were used: $size_{\min} = 1$, $size_{\max} = 4$, $m^{extra} = |V|$, $n_{\max}^{tools} = 1$, $n_{\max}^{fixtures} = 5$, $p_{joining} = 0.5$, $p_{newfixt} = 0.5$, $[t_{\min}^{fixt}, t_{\max}^{fixt}] = [6.0, 9.0]$, $[t_{\min}^{fitting}, t_{\max}^{fitting}] = [1.0, 12.0]$, $[t_{\min}^{joining}, t_{\max}^{joining}] = [3.0, 8.0]$, and $[t_{\min}^{tool}, t_{\max}^{tool}] = [1.0, 3.0]$. An upper bound of $\lceil |V|/5 \rceil$ was set on the number of generated conjunctive constraints.

1.2.1 Families 1 to 4

Four families of instances are generated focusing on different constraint scenarios. In case of Family 1, the parameters described before are used, and 5 instances are generated for each value of the number of parts $|V| = 10, \ldots, 30$. A similar procedure is used for Families 2-4, however, with additional conjunctive constraints in Family 2, disjunctive constraints in Family 3, and both conjunctive and disjunctive constraints in Family 4.

The properties of the instances generated by these parameters are shown in Table 1 where each row contains the average values over five instances with the same number of parts (*part*). These values are the number of tasks (*task*), the number of tools (*tool*), the number of fixtures (*fixt*), and the number of conjunctive and disjunctive constraints (*conj* and *disj*). These values show that there are no or only very few cycles in the liaison graph, i.e., $task \approx part - 1$.

1.2.2 Family 5

One additional family of instances is generated with a higher number of cycles in the liaison graph. In order to keep the number of extra tasks under control, this time the modified parameters $size_{\min} = 3$, $size_{\max} = 5$, $m^{extra} = 0.3$ are used. Conjunctive and disjunctive constraints are also generated for the instances. By this, the properties of the instances generated by these parameters are also shown in Table 1 where each row contains the average values over five instances with the same number of parts.

Table 1: Properties of problem instances of Families 1-4 and Family 5

part	Family 1-4			Family 1		Family 2		Family 3		Family 4		Family 5				
	task	tool	fixt	conj	disj	conj	disj	conj	disj	conj	disj	task	tool	fixt	conj	disj
10	9.2	2.0	3.4	0.0	0.0	2.0	0.0	0.0	0.8	2.0	0.8	11.0	2.0	1.6	1.0	0.0
11	10.0	2.0	4.2	0.0	0.0	2.0	0.0	0.0	1.2	2.0	1.2	12.0	2.0	1.6	2.0	0.0
12	11.0	2.0	4.0	0.0	0.0	2.0	0.0	0.0	0.4	2.0	0.4	14.0	2.0	1.8	2.0	0.0
13	12.2	2.0	4.4	0.0	0.0	2.0	0.0	0.0	1.6	2.0	1.6	15.0	2.0	2.2	2.0	0.0
14	13.2	2.0	5.0	0.0	0.0	2.0	0.0	0.0	1.0	2.0	1.0	16.0	2.0	2.0	2.0	0.0
15	14.2	2.0	5.2	0.0	0.0	3.0	0.0	0.0	1.4	3.0	1.4	17.0	2.0	2.2	3.0	0.0
16	15.0	2.0	5.8	0.0	0.0	3.0	0.0	0.0	1.4	3.0	1.4	18.8	2.0	2.4	3.0	0.2
17	16.0	2.0	6.4	0.0	0.0	3.0	0.0	0.0	2.2	3.0	2.2	20.0	2.0	2.8	3.0	0.0
18	17.2	2.0	6.2	0.0	0.0	3.0	0.0	0.0	1.8	3.0	1.8	21.0	2.0	3.4	3.0	0.2
19	18.0	2.0	6.6	0.0	0.0	3.0	0.0	0.0	1.8	3.0	1.8	22.0	2.0	3.0	3.0	0.2
20	19.0	2.0	6.6	0.0	0.0	4.0	0.0	0.0	1.4	4.0	1.4	24.0	2.0	2.8	4.0	0.2
21	20.0	2.0	6.8	0.0	0.0	4.0	0.0	0.0	2.4	4.0	2.4	25.0	2.0	3.0	4.0	0.6
22	21.6	2.0	6.6	0.0	0.0	4.0	0.0	0.0	1.8	4.0	1.8	26.0	2.0	3.8	4.0	0.4
23	22.4	2.0	6.6	0.0	0.0	4.0	0.0	0.0	2.6	4.0	2.6	27.4	2.0	2.8	4.0	0.2
24	23.2	2.0	7.6	0.0	0.0	4.0	0.0	0.0	2.6	4.0	2.6	29.0	2.0	3.4	4.0	1.2
25	24.4	2.0	7.4	0.0	0.0	5.0	0.0	0.0	2.0	5.0	2.0	30.0	2.0	3.4	5.0	0.8
26	25.4	2.0	7.6	0.0	0.0	5.0	0.0	0.0	3.0	5.0	3.0	31.0	2.0	3.8	5.0	0.4
27	26.6	2.0	8.4	0.0	0.0	5.0	0.0	0.0	3.0	5.0	3.0	32.0	2.0	3.6	5.0	0.6
28	27.2	2.0	8.4	0.0	0.0	5.0	0.0	0.0	2.4	5.0	2.4	33.0	2.0	3.6	5.0	0.2
29	28.6	2.0	8.0	0.0	0.0	5.0	0.0	0.0	2.2	5.0	2.2	34.8	2.0	4.6	5.0	0.2
30	29.0	2.0	9.0	0.0	0.0	6.0	0.0	0.0	3.2	6.0	3.2	36.0	2.0	3.8	6.0	1.0

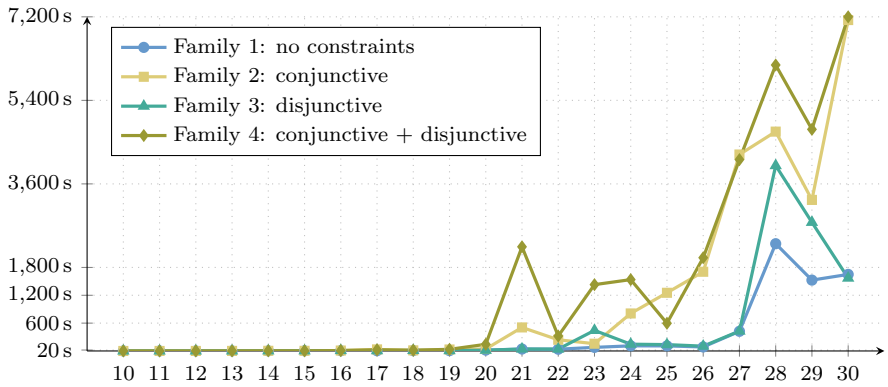


Figure 2: Results on the effect of conjunctive and disjunctive constraints: average computation times on different instance families

2 Computational results

All experiments are performed on a workstation with an i9-7960X 2.80 GHz CPU with 16 cores, under Debian 9 operating system using 4 threads. All experiments were run with a time limit of 7200 seconds.

2.1 Experiments on conjunctive and disjunctive constraints

The aim of these experiments is to investigate the effects of the presence of conjunctive and disjunctive constraints. Thus, these are run on Families 1-4.

First, the MILP-based approach of the authors is evaluated (see Section 2.1.1). Then, the MILP-based approach and the CP-based approach of (Kardos et al., 2020) are compared. In case of the CP-based method two versions of the OR-Tools solver are used (version 7.1.6720 and 9.0.9048), because the newer version yielded surprisingly bad results (see Section 2.1.2). Finally, a problem-specific branching strategy is investigated (see Section 2.1.3).

2.1.1 Evaluation of the MILP-based approach

In Figure 2, the average solution times of the MILP-based approach are depicted for Families 1-4. One can observe that the presence of conjunctive constraints has a huge impact on the solution time. Instances up to 19 parts are solved to optimality within 10 seconds, and instances up to 22 parts are solved within a minute. Larger instances without conjunctive constraints are solved within 10 minutes. However, in case of conjunctive constraints, solution time grows rapidly depending on the part numbers, that is, for example, 12 out of the largest 25 instances are not solved to optimality within 7200 seconds.

In Table 2, the average and maximum relative gaps are displayed for the largest instances of the families after 600, 1800, 3600 and 7200 seconds, grouped by the number of parts. The gap is calculated as $100 \times (UB - LB)/UB$, where UB and LB are the best upper bound (i.e., the value of the best solution) and the best lower bound found so far, respectively. In most of the cases, at least one instance is not solved to optimally within 7200 seconds, that is, the maximum final gap after 7200 seconds is not zero. For Families 2 and 4, the final gaps are quite large, that is, for instances with at least 29 parts the average gaps are between 4.00% and 9.50%, and the maximum gaps are between 11.20% and 19.00%.

2.1.2 Comparison of the MILP- and the CP-based approaches

In Figure 3, the average solution times on Families 1-4 are depicted, comparing the MILP-based approach and the CP-based approach with solver versions 7.1 and 9.0 as well.

First of all, the CP-based approach with solver version 9.0 struggles on these instances, that is, could not end within 7200 seconds even for the smaller instances, thus this approach is tested only on instances

with at most 18 parts.

The CP-based approach with solver version 7.1 and the MILP-based approach yielded disparate results. On the smaller instances (up to 20 parts) there are no big differences between the two methods except for a few salient instances, where the CP-based approach required much more time. In most of the bigger instances the MILP-based approach outperformed the other one, however, on some instances the CP-based approach yielded better results.

2.1.3 Evaluation of problem-specific branching strategy

In Figure 4 the average execution times on Families 1-4 are depicted, comparing the default branch-and-cut procedure and the one with the proposed problem-specific branching strategy. Note that the maximum values of the y-axes are differ for the distinct instance families. In case of instances with at most 20 parts, the proposed branching strategy has no remarkable impact. However, on larger instances, this branching rule reduced solution times significantly (in some cases this improvement is 96%).

In Table 2 the relative gaps corresponding to these experiments are also displayed for the largest instances of Families 1–4 grouped by the number of parts. In contrast to the default branching, for the most of these groups all of the instances are solved to optimality within 7200 seconds, that is, final gaps are zero. Using problem-specific branching, for the instances of Families 2 and 4 with at least 29 parts, average gaps are between 3.60% and 4.10%, and maximum gaps are between 8.10% and 11.50%.

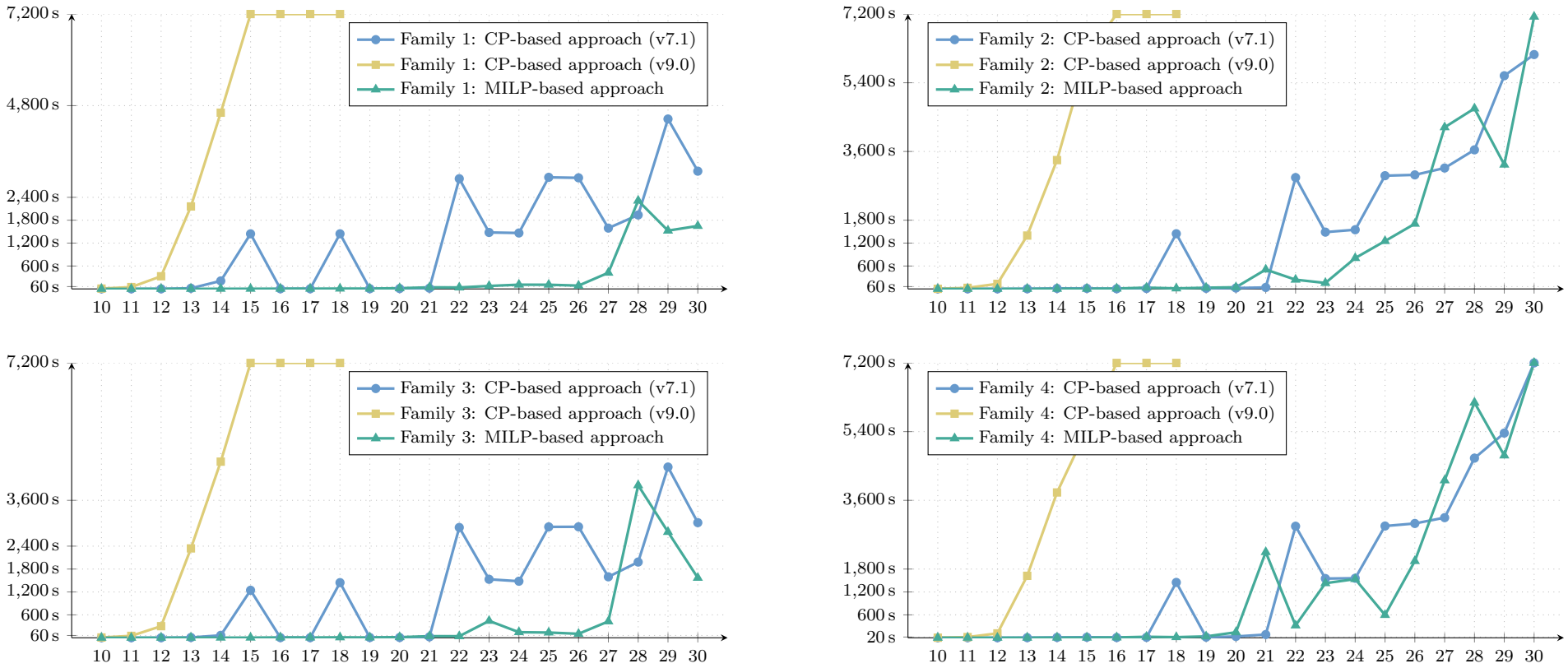


Figure 3: Comparison of different solution approaches: average computation times on Familes 1-4.

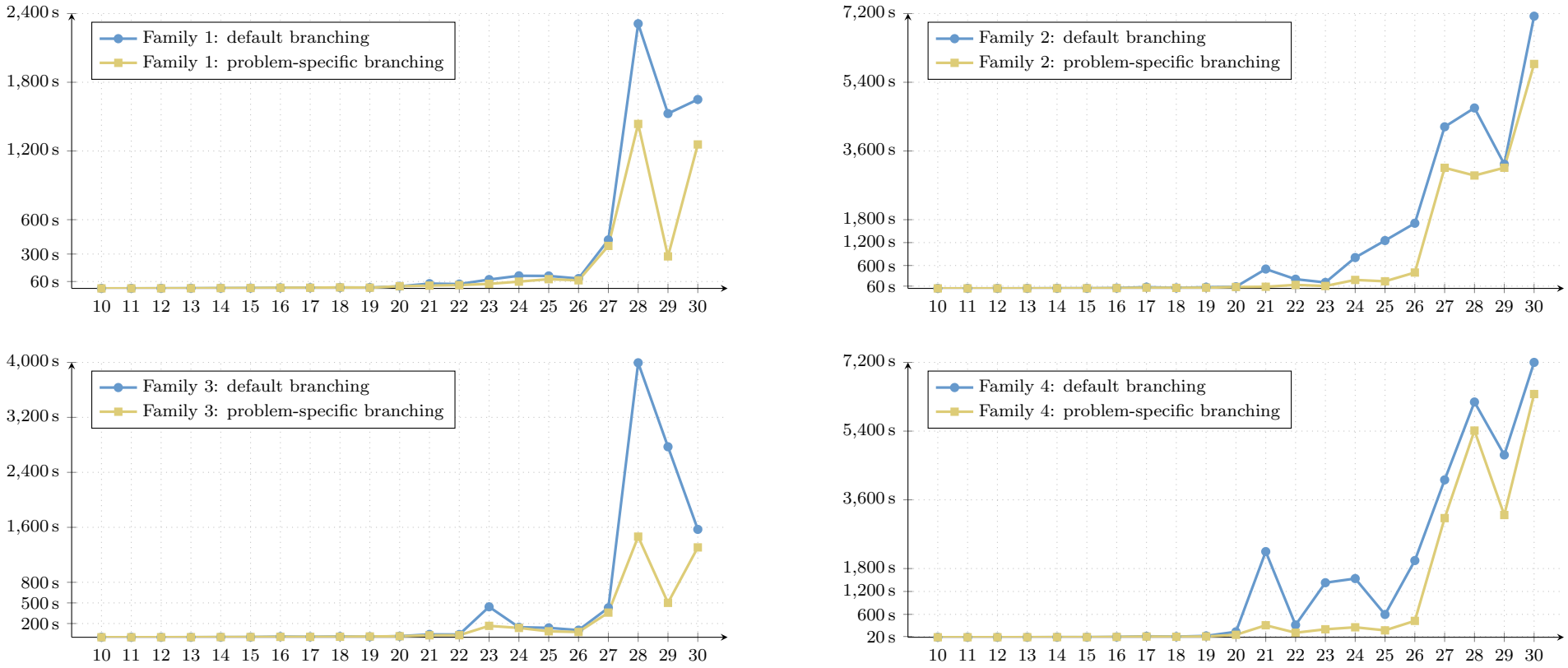


Figure 4: Results on problem-specific branching strategy: average computation times on Familes 1-4. Note that the maximum values of the y-axes are differ for the distinct instance families.

Table 2: Average and maximum gaps (%) after different amounts of computation time with different branching strategies

parts	default branching								problem-specific branching							
	600 s		1800 s		3600 s		7200 s		600 s		1800 s		3600 s		7200 s	
	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max
Family 1																
26	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
27	0.70	3.40	0.00	0.00	0.00	0.00	0.00	0.00	1.30	6.60	0.00	0.00	0.00	0.00	0.00	0.00
28	2.70	6.20	1.00	4.20	0.60	3.20	0.50	2.40	1.80	9.00	0.70	3.70	0.60	2.80	0.00	0.00
29	0.90	4.30	0.60	2.90	0.40	2.00	0.10	0.50	1.00	5.10	0.00	0.00	0.00	0.00	0.00	0.00
30	2.20	9.50	1.30	6.40	1.10	5.40	0.70	3.50	2.30	11.30	1.60	7.80	1.40	7.10	0.00	0.00
Family 2																
26	2.80	11.40	0.90	4.40	0.70	3.40	0.40	2.10	2.30	11.50	0.00	0.00	0.00	0.00	0.00	0.00
27	9.30	17.00	3.50	7.90	2.50	6.10	1.60	5.00	7.80	13.80	3.60	7.10	1.60	6.40	0.90	4.50
28	7.00	14.00	5.50	12.40	4.70	11.10	3.30	8.20	8.80	11.90	3.50	9.60	1.80	9.20	1.80	8.80
29	7.50	17.70	5.30	15.70	4.50	12.30	4.00	11.20	6.60	14.70	4.70	11.90	3.70	9.60	3.60	9.30
30	13.00	22.20	10.60	18.20	8.20	13.80	6.80	13.40	10.10	17.00	7.00	13.20	5.40	10.40	3.90	8.10
Family 3																
26	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
27	1.00	5.10	0.00	0.00	0.00	0.00	0.00	0.00	1.40	7.10	0.00	0.00	0.00	0.00	0.00	0.00
28	3.40	6.10	1.70	3.60	0.80	2.90	0.30	1.30	3.20	8.30	0.80	3.80	0.50	2.60	0.00	0.00
29	1.80	4.40	0.80	2.90	0.60	2.00	0.10	0.40	1.30	5.10	0.00	0.00	0.00	0.00	0.00	0.00
30	2.00	10.10	1.40	7.00	1.30	6.60	1.20	6.20	2.00	10.10	1.70	8.40	1.30	6.60	0.00	0.00
Family 4																
26	3.60	11.00	1.20	6.20	1.10	5.40	1.00	4.80	2.00	10.20	0.00	0.00	0.00	0.00	0.00	0.00
27	7.20	13.80	4.50	9.60	2.80	7.80	1.40	5.00	7.40	15.20	3.70	7.10	2.20	6.40	0.40	2.10
28	12.60	15.50	8.50	12.70	6.40	11.00	4.40	9.30	10.80	13.60	7.30	10.80	5.30	9.80	3.80	8.60
29	10.10	21.00	6.50	15.20	5.70	14.80	4.70	14.00	7.90	18.70	5.80	15.60	4.40	13.00	4.10	11.50
30	14.40	21.60	11.70	20.30	10.60	19.40	9.50	19.00	11.20	15.50	8.10	14.00	5.60	10.50	4.00	8.10

2.2 Experiments on cycles of the liaison graph

The aim of these experiments is to investigate the effects of cycles in the liaison graph. Thus, these are run on Family 5.

First, the MILP-based approach of the authors is evaluated (see Section 2.2.1). Then, the MILP-based approach and the CP-based approach of (Kardos et al., 2020) are compared. Again, in case of the CP-based method two versions of the OR-Tools solver are tested (see Section 2.2.2). Finally, a problem-specific branching strategy is investigated (see Section 2.2.3).

2.2.1 Evaluation of the MILP-based approach

In Figure 5a, the average execution times of the MILP-based approach are depicted for Family 5. Note that the bar plot in the background refers to the average number of extra tasks (that, is the number of tasks minus the number of parts plus one), where the lowest bar refers to 2 extra tasks, and the highest one refers to 7 extra tasks.

The approach scales well with the increase of problem size. Instances with at most 16 parts and 3.8 extra edges are solved to optimality within a second. Instance with at most 23 part and 5.4 extra edges are solved to optimality within a minute. The larger instances are solved to optimality within 2000 seconds, on average.

2.2.2 Comparison of the MILP- and the CP-based approaches

In Figure 5b, the average solution times on Family 5 are depicted, comparing the MILP-based approach and the CP-based approach with solver versions 7.1 and 9.0 as well.

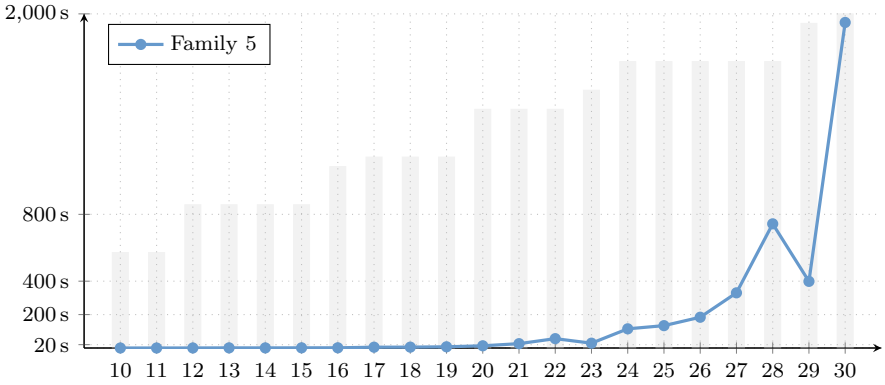
The CP-based approach with solver version 9.0 is struggling on these instances, thus, it is tested only on instances with at most 18 parts. The CP-based approach with solver version 7.1 performed better, however, the MILP-based approach significantly outperforms them. For example, the CP-based approach could not solve to optimality any of the instances with at least 21 parts. The minimum, average and maximum optimality gaps on these instances are 1.8%, 20.0% and 42.5%, respectively.

2.2.3 Evaluation of problem-specific branching strategy

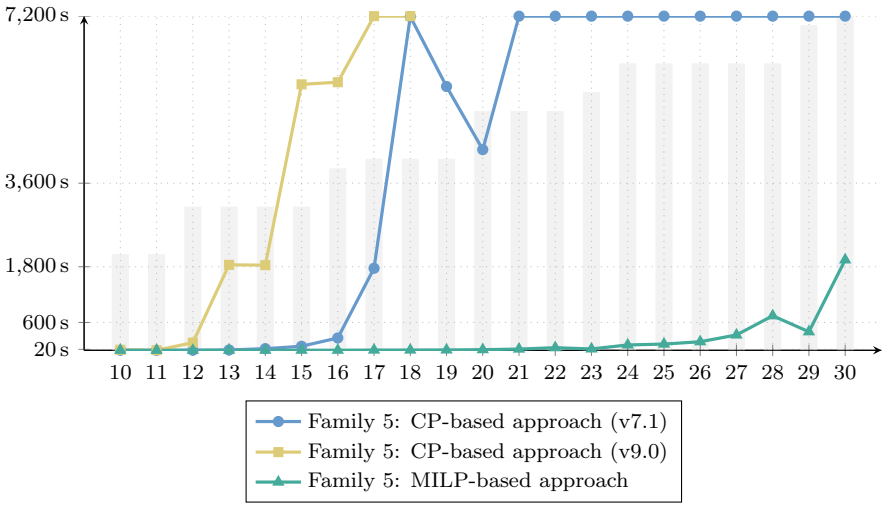
In Figure 5c the average solution times on Family 5 are depicted, comparing the default branch-and-cut procedure and the one with the proposed problem-specific branching strategy. In case of instances with at most 25 parts, the proposed branching strategy has no remarkable impact. However, on larger instances, this branching rule reduced solution times significantly (for the largest instances this improvement is 74%, on average).

References

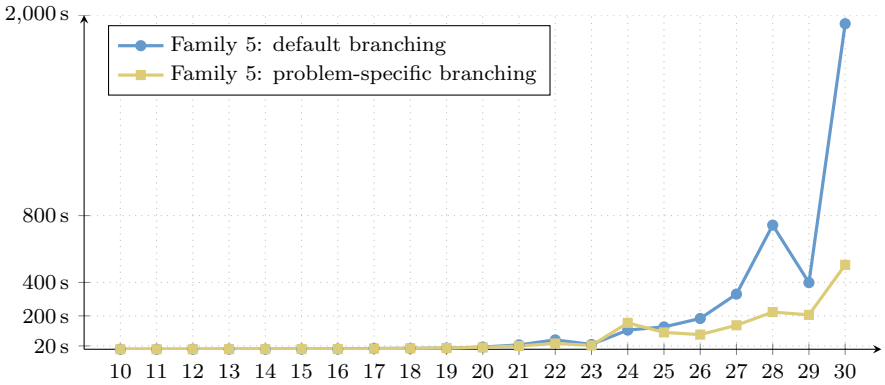
Kardos, C., Kovács, A., Váncza, J., 2020. A constraint model for assembly planning. Journal of Manufacturing Systems 54, 196–203.



(a) Evaluation of the MILP-based approach



(b) Comparison of the CP- and the MILP-based approaches



(c) Comparison of the default and the problem-specific branching strategies

Figure 5: Results on Family 5.