CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	A collection of constraint programming models for the three-dimensional stable matching problem with cyclic preferences
Authors	Cseh, Ágnes;Escamocher, Guillaume;Genç, Begüm;Quesada, Luis
Publication date	2021-10-15
Original Citation	Cseh, Á, Escamocher, G., Genç, B. and Quesada, L. (2021) 'A collection of Constraint Programming models for the three-dimensional stable matching problem with cyclic preferences', 27th International Conference on Principles and Practice of Constraint Programming (CP 2021), Montpellier, France, 25 - 29 October, 22 (19pp). doi: 10.4230/LIPIcs.CP.2021.22
Type of publication	Conference item
Link to publisher's version	https://doi.org/10.5281/zenodo.5156119 - 10.4230/LIPIcs.CP.2021.22
Rights	© 2021, Ágnes Cseh, Guillaume Escamocher, Begüm Genç, and Luis Quesada. Licensed under Creative Commons License CC-BY 4.0 - https://creativecommons.org/licenses/by/4.0/
Download date	2026-10-31 17:47:53
Item downloaded from	https://hdl.handle.net/10468/12115

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

A collection of Constraint Programming models for the three-dimensional stable matching problem with cyclic preferences

Supplementary Material

A Example

We present in Figure 1 an example of a 3DSM-CYC instance I for $n = 4$. This instance was actually used in our experiments, it comes from the Random dataset. For each agent $a_i \in A$, the four agents in a_i's preference list are listed in decreasing order of preference. The same is true for agents from B and C. For example, a_1's most preferred, top-choice agent is b_2, her second choice is b_4, her third choice is b_3, and her least preferred, fourth-choice agent is b_1.

We give in Figure 2 an example of a matching M for I. Each line in the figure represents a triple of agents who are assigned to each other in M. For example, a_1, b_3, and c_4 are assigned to each other. We first explain why M is weakly stable. Suppose that there is a strongly blocking triple (a_i, b_j, c_k) for M. According to the definition of weak stability in Section 2.1 in the main part of the paper, c_k prefers a_i to the agent from A she is assigned to in M. But c_1, c_2, and c_4 are assigned in M to their most preferred agent from A (a_2, a_3, and a_1 respectively), while c_3 is assigned to her second-choice agent. So c_k must be c_3, and a_i must be the the agent ranked first by c_3. However, M assigns c_3's first ranked agent a_3 to her top choice b_2. Therefore there can be no strongly blocking triple for M, meaning that M is weakly stable.

We next show that M is not strongly stable. Indeed, consider the triple $t = (a_3, b_2, c_3)$. While a_3 is assigned to the same agent from B (b_2) in M and in t, both b_2 and c_3 prefer their assignments in t (c_3 and a_3 are both first-choice agents for b_2 and c_3 respectively) over their assignments in M (c_2 is last in the preference list of b_2 and a_4 is second in the preference list of c_3). Therefore, from the definition of strong stability in Section 2.1 of the main part of the paper, t is a weakly blocking triple for M, meaning that M is not strongly stable.

Let us now look at the cost of M according to the fairness notions studied in the paper. Agent a_1 is assigned to her third-choice agent from B, a_2 is also assigned to her third-choice agent, a_2 is assigned to her top-choice agent, and a_4 is assigned to her last-choice agent, so the sum of the ranks of the agents from B in the preference lists of the agents from A whom they are assigned to is $3 + 3 + 1 + 4 = 11$. Similarly, the sum of the ranks of the agents from C in the preference lists of the agents from B whom they are assigned to is $1 + 4 + 2 + 1 = 8$, and the sum of the ranks of the agents from A in the preference lists of the agents from C whom they are assigned to is $1 + 1$

a_1:	b_2	b_4	b_3	b_1		b_1:	c_3	c_4	c_2	c_1		c_1:	a_2	a_3	a_1	a_4
a_2:	b_3	b_1	b_4	b_2		b_2:	c_3	c_1	c_4	c_2		c_2:	a_3	a_2	a_1	a_4
a_3:	b_2	b_3	b_1	b_4		b_3:	c_2	c_4	c_1	c_3		c_3:	a_3	a_4	a_1	a_2
a_4:	b_2	b_4	b_3	b_1		b_4:	c_1	c_2	c_4	c_3		c_4:	a_1	a_3	a_4	a_2

Figure 1: A 3DSM-CYC instance I with 4 agents in each agent set.

(a_1, b_3, c_4)
(a_2, b_4, c_1)
(a_3, b_2, c_2)
(a_4, b_1, c_3)

Figure 2: A matching M for I.

$+ 2 + 1 = 5$. The egalitarian cost of M is the sum of these three sums, so $11 + 8 + 5 = 24$. The minimum regret cost of M is 4, because at least one agent are assigned the fourth and last ranked agent in her preference list (a_4 are assigned her least preferred agent b_1). Finally, the sex-equal cost of M is the pairwise absolute difference of the sums, so $|11 - 8| + |8 - 5| + |5 - 11| = 12$.

B Dataset

The compressed directory "Dataset" contains the instances studied in the experiments, as well as the code used to generate them.

C Heuristic selection

We compared the UNI and DIV models using various built-in search strategies on constraint solvers: Gecode and Chuffed. The built-in variable choice annotations that we considered from Minizinc are: input_order, first_fail, smallest; and the variable constraint strategies are: indomain_min, indomain_median, indomain_max, and indomain_split. These annotations were chosen as their combinations already exist for both Gecode and Chuffed. In the rest of this section, each strategy combination is labeled with the identifiers of relevant annotations. For instance, smallestmax strategy corresponds to a combination of the variable choice annotation smallest and the variable constraint indomain_max. Similarly, failsplit corresponds to first_fail and indomain_split, etc..

In Figure 3 and Figure 4 we observe the total time and failure performances of each model, i.e. DIV-agents, DIV-ranks, HS, UNI-agents, UNI-ranks, when solving the satisfiability problem under strong stability using Gecode. On these plots and also by our observation on larger instances, we confirm that failmin and nonemin strategies perform well on the DIV models. Similarly, failsplit strategy performs well on the UNI models. Furthermore, Figure 5 and Figure 6 present the same information when the tests are performed using the Chuffed solver. In Figure 5 and Figure 6 we can also infer that failmin and nonemin strategies perform well for DIV, and failsplit performs well for UNI. Note that the HS model is not included in Chuffed plots as this model has been directly implemented in Gecode. The performances of the heuristics show a similar trend for both weak and strong stability, and also the optimisation variants i.e. minimum regret, sex-equal, egalitarian. We do not add the remaining plots here considering that adding the alternative plots is not interesting as the observation on the search strategies is the same. Hence, we selected a subset of well performing strategies and some contrasting ones to perform tests on larger instances. The strategies that we selected are: failmin, nonemin, nonemax for DIV models; failsplit, nonemin, nonemax for UNI models; and nonemin and nonemax for the HS model.

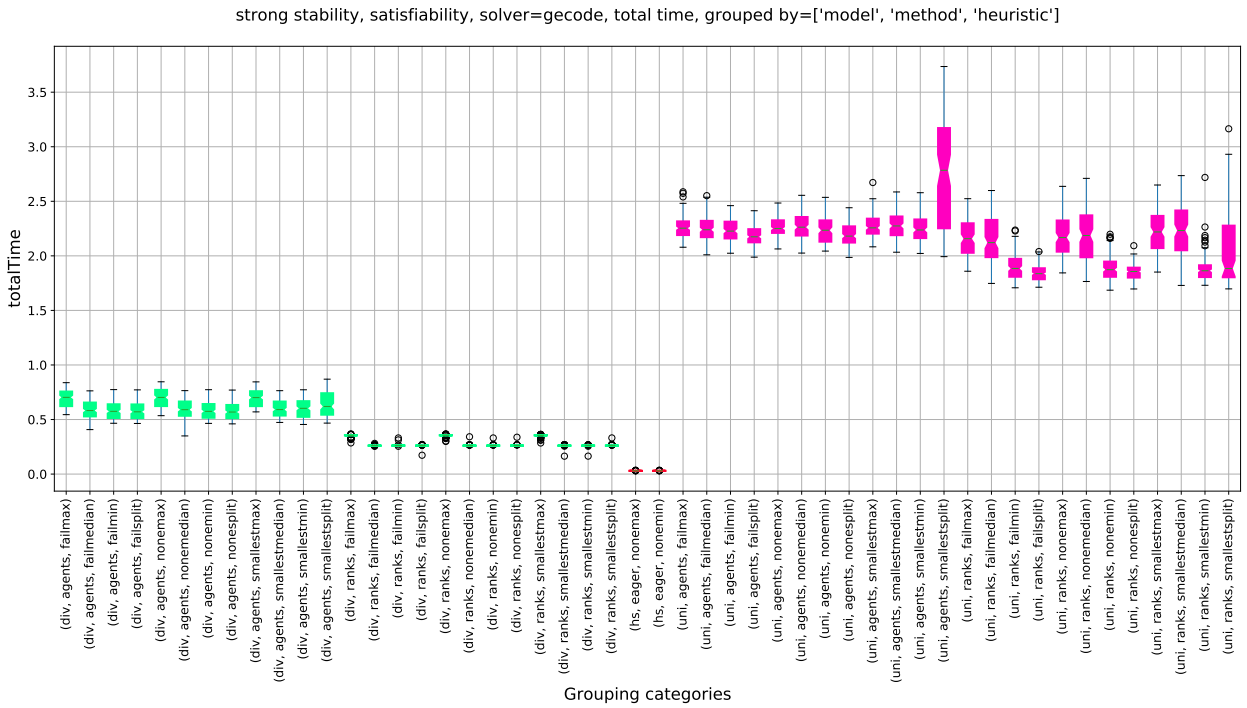


Figure 3: A comparison of total time of each model solving the satisfiability 3DSM-CYC problem using Gecode with different search strategies for $n = 5$ under strong stability for the entire dataset.

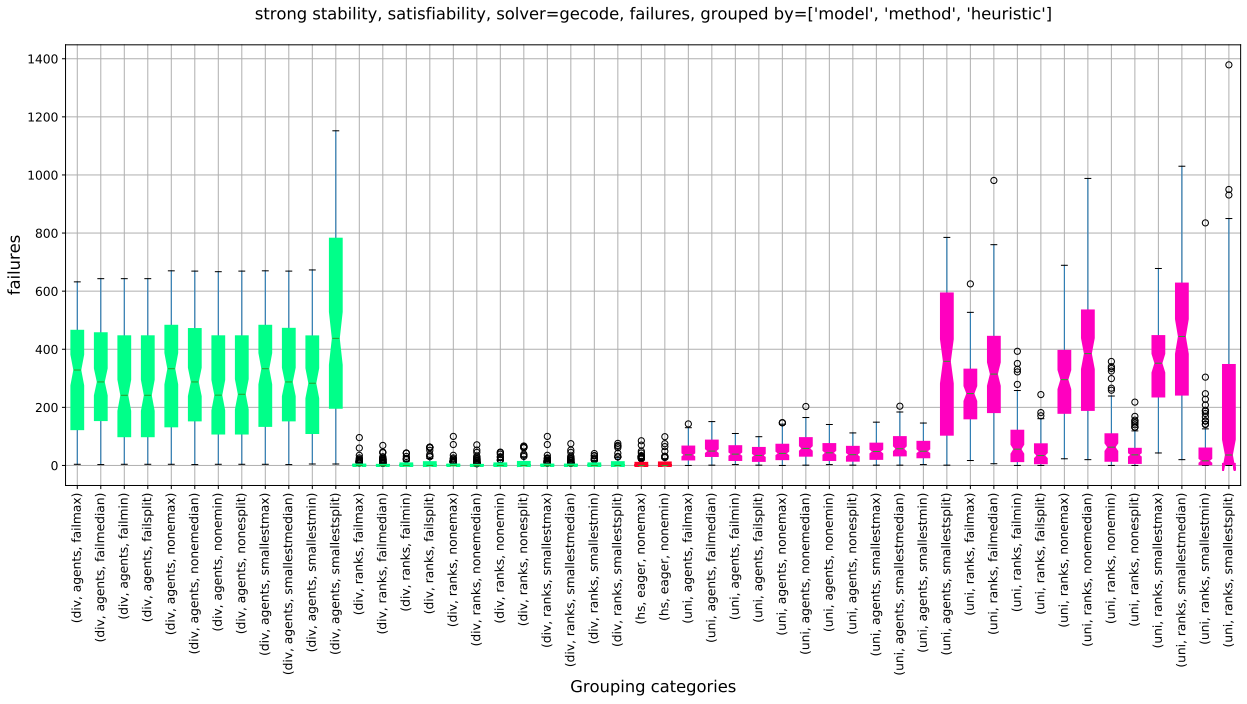


Figure 4: A comparison of failures of each model solving the satisfiability 3DSM-CYC problem using Gecode with different search strategies for $n = 5$ under strong stability for the entire dataset.

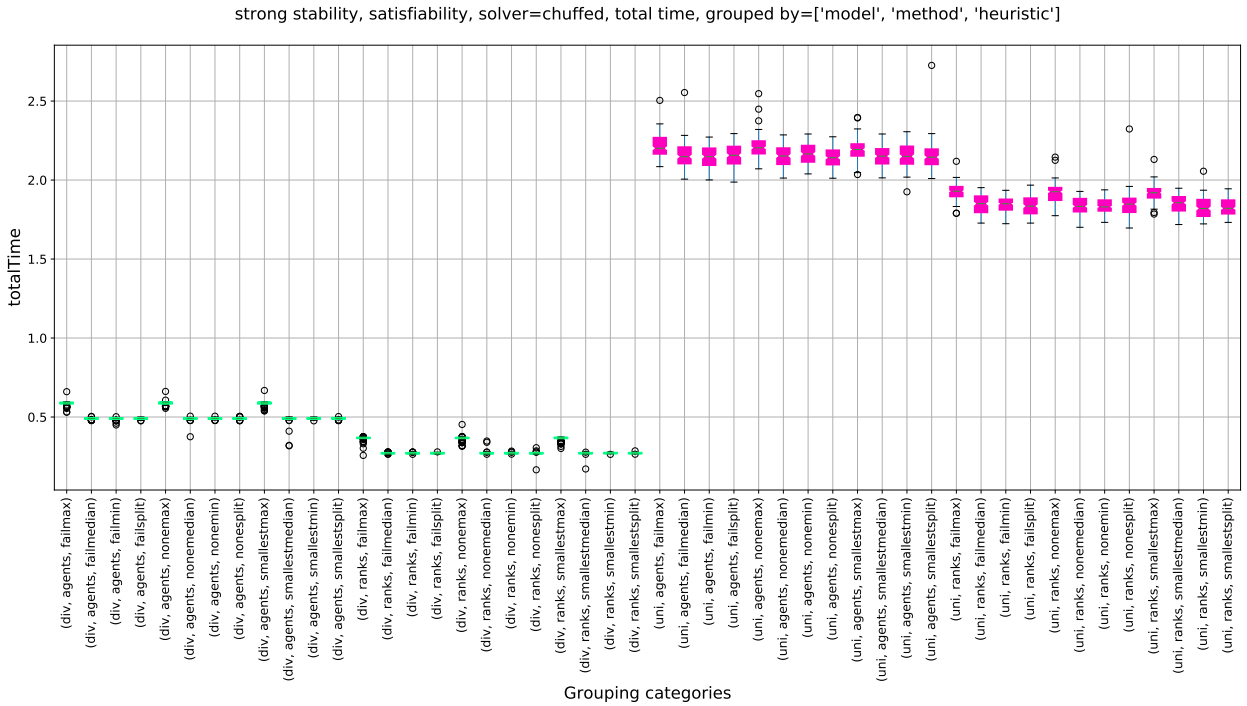


Figure 5: A comparison of total time of each model solving the satisfiability 3DSM-CYC problem using Chuffed with different search strategies for $n = 5$ under strong stability for the entire dataset.

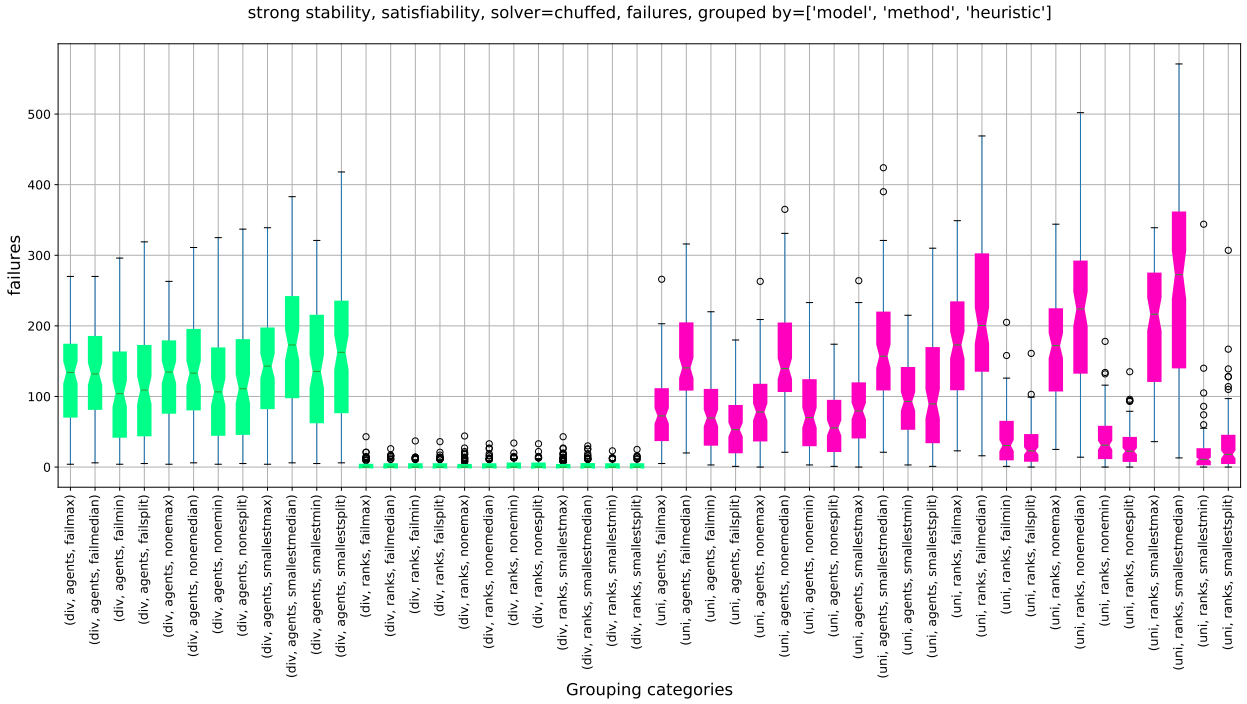


Figure 6: A comparison of failures of each model solving the satisfiability 3DSM-CYC problem using Chuffed with different search strategies for $n = 5$ under strong stability for the entire dataset.

We performed further tests comparing the selected heuristics on $n \in \{6, 7, 8\}$. From these plots, we present Figure 7 and Figure 8 as they are interesting for UNI models since we can clearly observe the performances of different search strategies. From these plots, we infer that failsplit is the best performing heuristic among the selected ones for larger instances, where $n = 8$. Additionally, Figure 9 and Figure 10 demonstrate the same metrics on the sex-equal variant of the 3DSM-CYC problem. One can easily observe that the comparison of strategy performances across different variants have similar results.

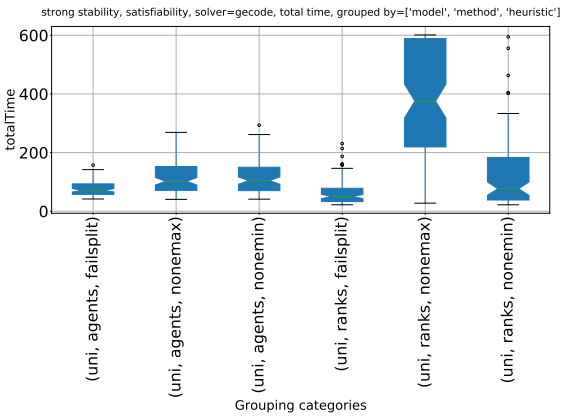


Figure 7: A comparison of total time of UNI solving the satisfiability 3DSM-CYC using Gecode with selected strategies for $n = 8$ under strong stability for the entire dataset.

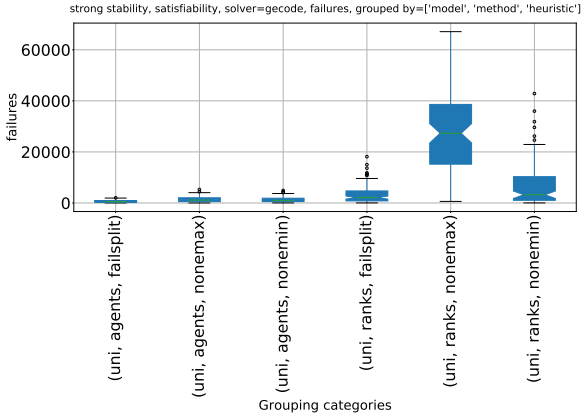


Figure 8: A comparison of failures of UNI solving the satisfiability 3DSM-CYC using Gecode with selected strategies for $n = 8$ under strong stability for the entire dataset.

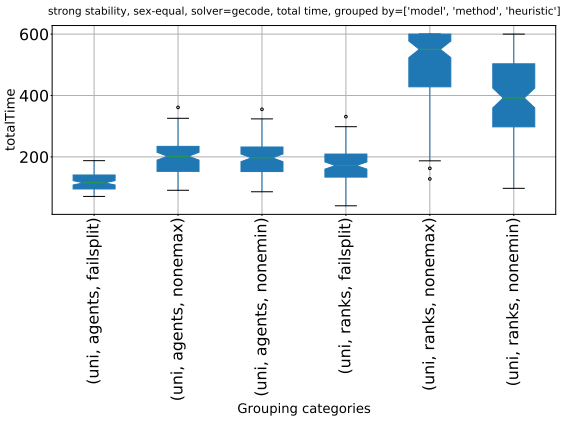


Figure 9: A comparison of total time of UNI solving the sex-equal 3DSM-CYC using Chuffed with selected strategies for $n = 8$ under strong stability for the entire dataset.

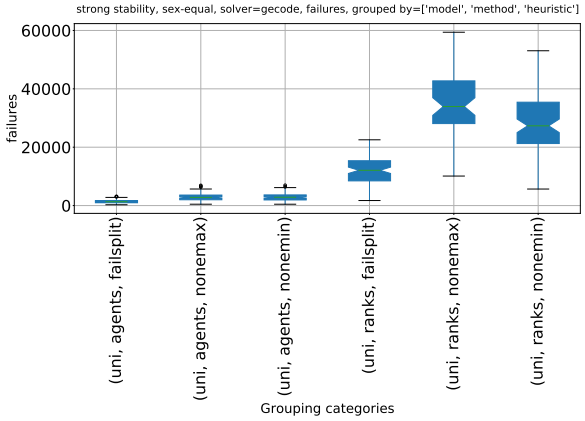


Figure 10: A comparison of failures of UNI solving the sex-equal 3DSM-CYC using Chuffed with selected strategies for $n = 8$ under strong stability for the entire dataset.

In addition to the plots of UNI models, one can observe in figures from Figures 11, 12, 13, and 14 the time performance of DIV models under different strategies when $n = 8$. Note that, as mentioned previously, the set of strategies have similar performances for different problem variants. In order to demonstrate the performance of different strategies on DIV models, in Figure 11 and Figure 12 we present the time performance of DIV models with selected strategies for the minimum regret

problem under weak stability using Gecode and Chuffed, respectively. Similarly, in Figure 13 and Figure 14 we present the results of the same setting under strong stability. Note that, the plots for DIV-ranks models under strong stability in Gecode are not interesting when $n = 8$ as a solution is found very quickly for DIV-ranks, and time-limit is met for DIV-agents. From these tests, one can observe that the failmin strategy is the best one for DIV models.

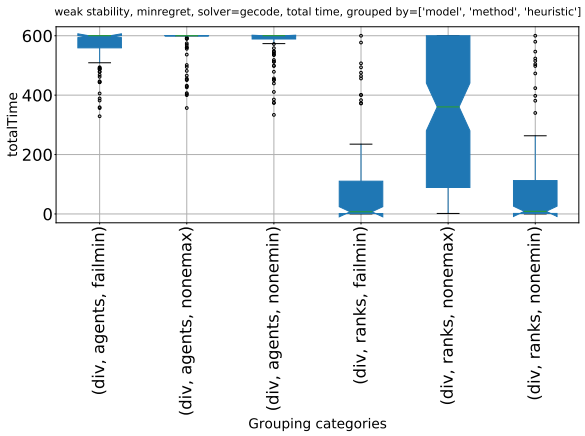


Figure 11: A comparison of total time of DIV solving the minimum regret 3DSM-CYC using Gecode with selected strategies for $n = 8$ under weak stability for the entire dataset.

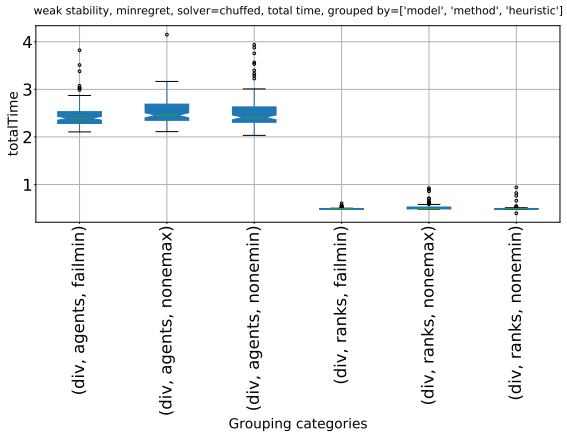


Figure 12: A comparison of total time of DIV solving the minimum regret 3DSM-CYC using Chuffed with selected strategies for $n = 8$ under weak stability for the entire dataset.

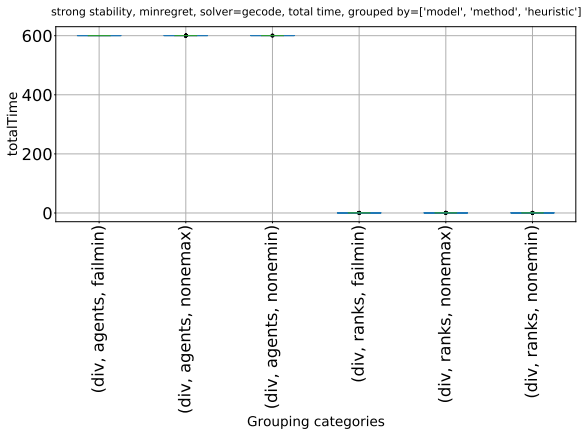


Figure 13: A comparison of total time of DIV solving the minimum regret 3DSM-CYC using Gecode with selected strategies for $n = 8$ under strong stability for the entire dataset.

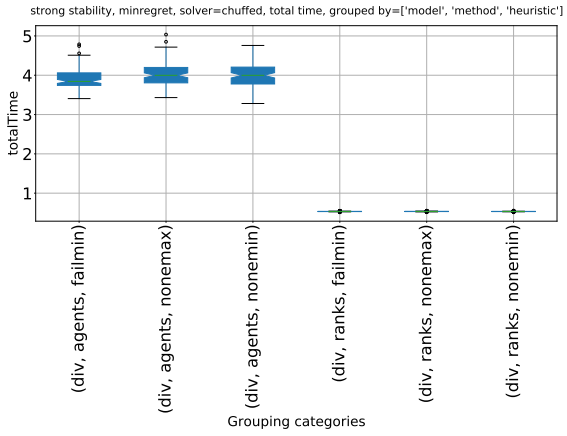


Figure 14: A comparison of total time of DIV solving the minimum regret 3DSM-CYC using Chuffed with selected strategies for $n = 8$ under strong stability for the entire dataset.

We compare the nonemax and nonemin strategies for the HS model when solving 3DSM-CYC variants. Figure 15 presents a comparison of these strategies on HS when using strong stability for each problem variant. Similarly, Figure 16 presents the same information under weak stability. As one may observe, the strong stability plots are not as informative as the weak stability ones. From these two figures, we infer that nonemax and nonemin strategies are very competitive on HS. Our experiments on $n \in \{5, \ldots, 11\}$ confirm that nonemax and nonemin strategies do not cause a significant difference in the performance. Therefore, we pick one of them for HS, i.e. the nonemax strategy, which seems to have slightly better median performance on the average.

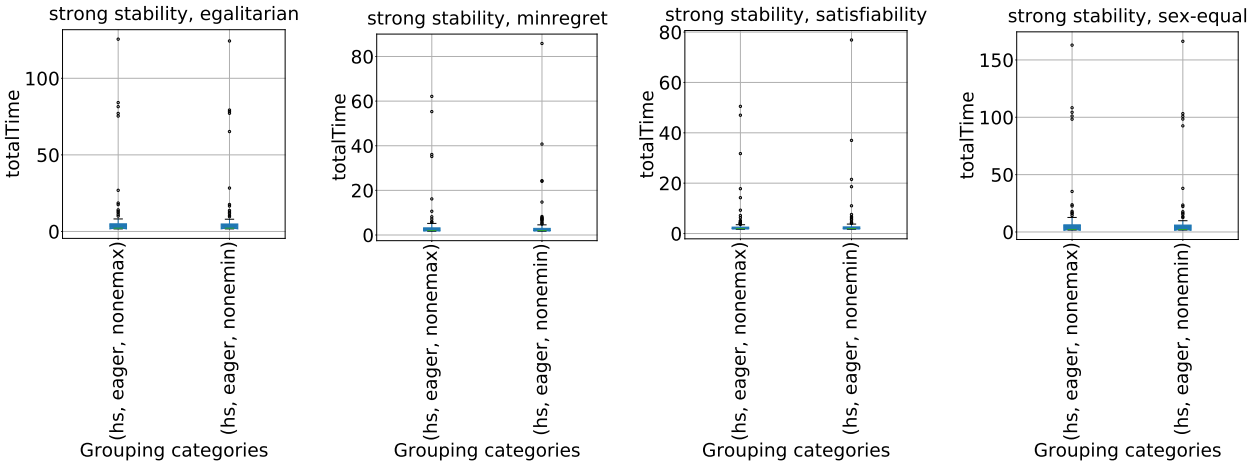


Figure 15: A comparison of total time required for HS solving 3DSM-CYC for egalitarian, minimum regret, satisfiability, and sex-equal versions in order using Gecode with selected strategies for $n = 11$ under strong stability for the entire dataset.

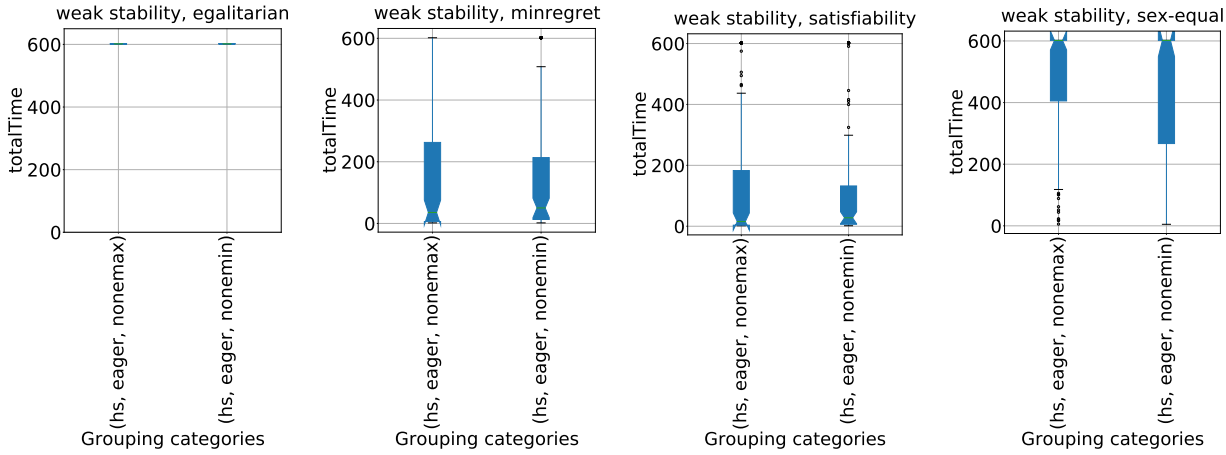


Figure 16: A comparison of total time required for HS solving 3DSM-CYC for egalitarian, minimum regret, satisfiability, and sex-equal versions in order using Gecode with selected strategies for $n = 11$ under weak stability for the entire dataset.

D Model Comparison in terms of the number of failures

In this section we look at the number of failures that the different models have when using the two solvers under the two notions of stability considered (Figures 17, 18, 19 and 20). In general, there is a correlation between the number of failures and the total run time. However it is important to remark that the ratio of the number of failures per unit of time (i.e., speed of exploration) is not the same for all models. One remarkable example has to do with the UNI model. The number of failures obtained with this model is very low in most of the cases (in particular when we use Gecode (see Figures 17 and 18). However, we end up running out of time with this model since propagation is very expensive, which leads to a very low speed of exploration. This situation is not only observed in UNI but also in DIV-agents.



Figure 17: A comparison of the number of failures under weak stability on small instances using Gecode.

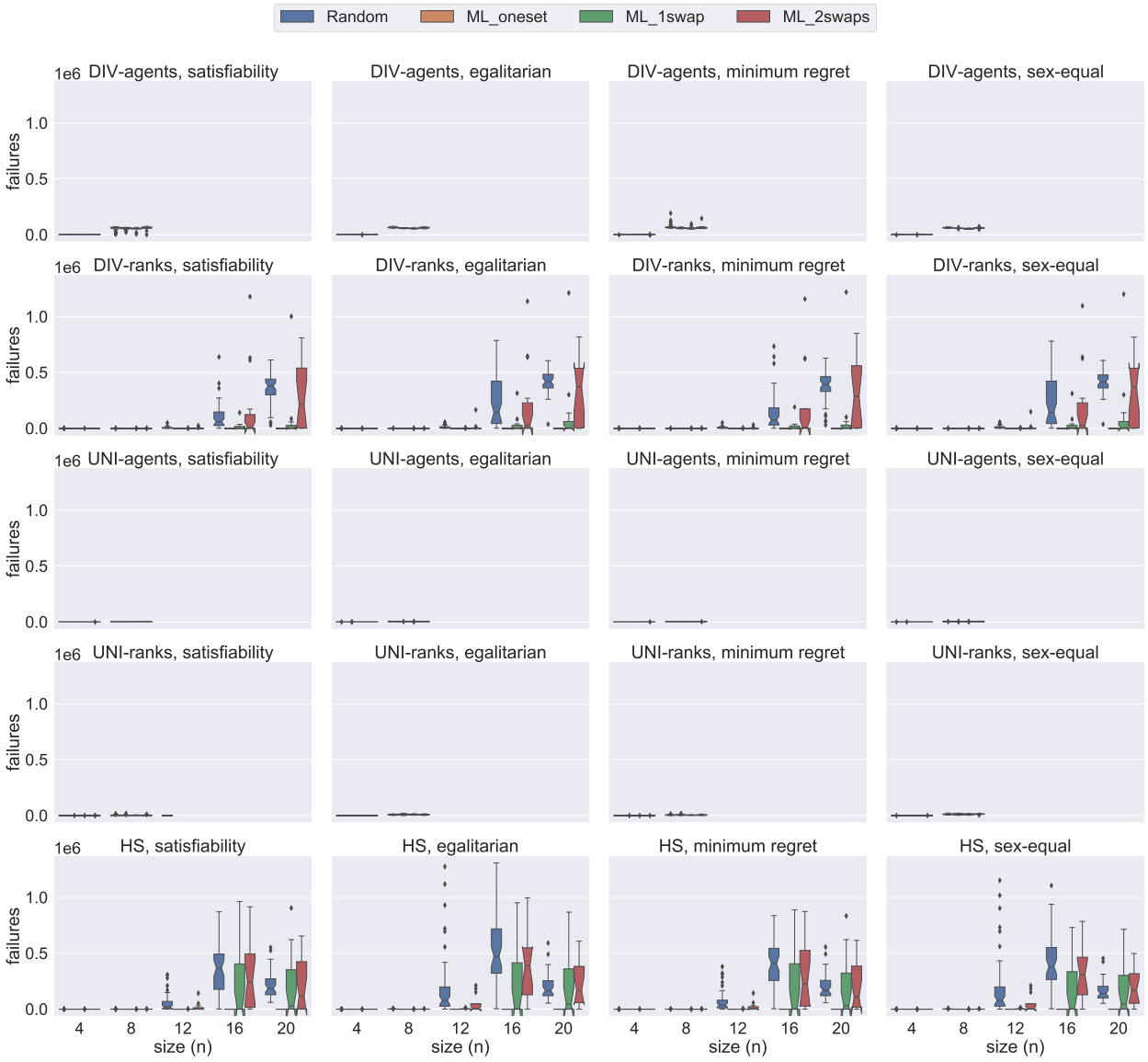


Figure 18: A comparison of the number of failures under strong stability on small instances using Gecode.

As expected, for those cases where the time limit has been reached, there is an inverse correlation between the number of failures and the size of the instances. The bigger the instance the more expensive propagation is. This is why, for instance, we observe that for weak stability, HS reports more failures in the egalitarian cases for $n = 9$ than for $n = 11$ (see Figure 17).

We found a dependency between the number of failures and the solver used. For instance if we look at the number of failures of DIV-ranks and UNI-ranks when using Gecode on the weak stability instances, we observe that DIV-ranks tends to fail more than UNI-ranks. However, in Chuffed, we observe the opposite situation. Our guess is that, some how Chuffed is able to take more advantage of the way how constraints are modelled in UNI-ranks for the generation of nogoods. However, this certainly needs further investigation.

In the plot reporting the scalability (with respect to the number of failures) of DIV-ranks in the

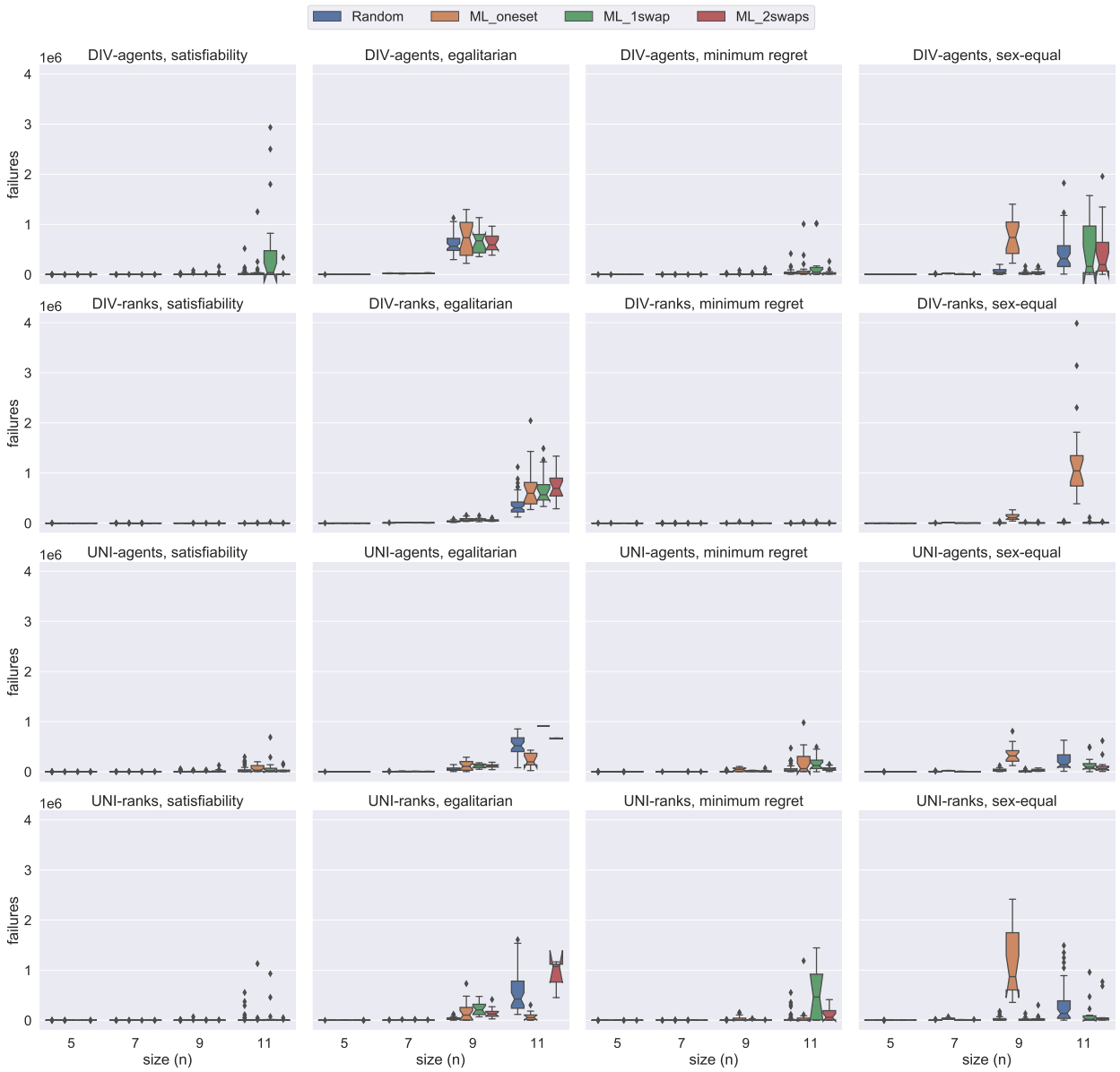


Figure 19: A comparison of the number of failures except HS under weak stability on small instances using Chuffed.

different scenarios (Figure 21), we observe more clearly how easy ML instances are under strong stability. From Lemma 1, we know that the satisfiability problem can be solved in polynomial time for ML_oneset. This also means that the problem can be solved in polynomial time when the objective is min-regret. We can observe in Figure 21 that we are able to solve these instances almost without failing. The complexity of the other two optimisation cases still remains open. However, we can see that the behaviour is pretty much the same. In fact. the same observation holds for ML_1swap and ML_2swaps.

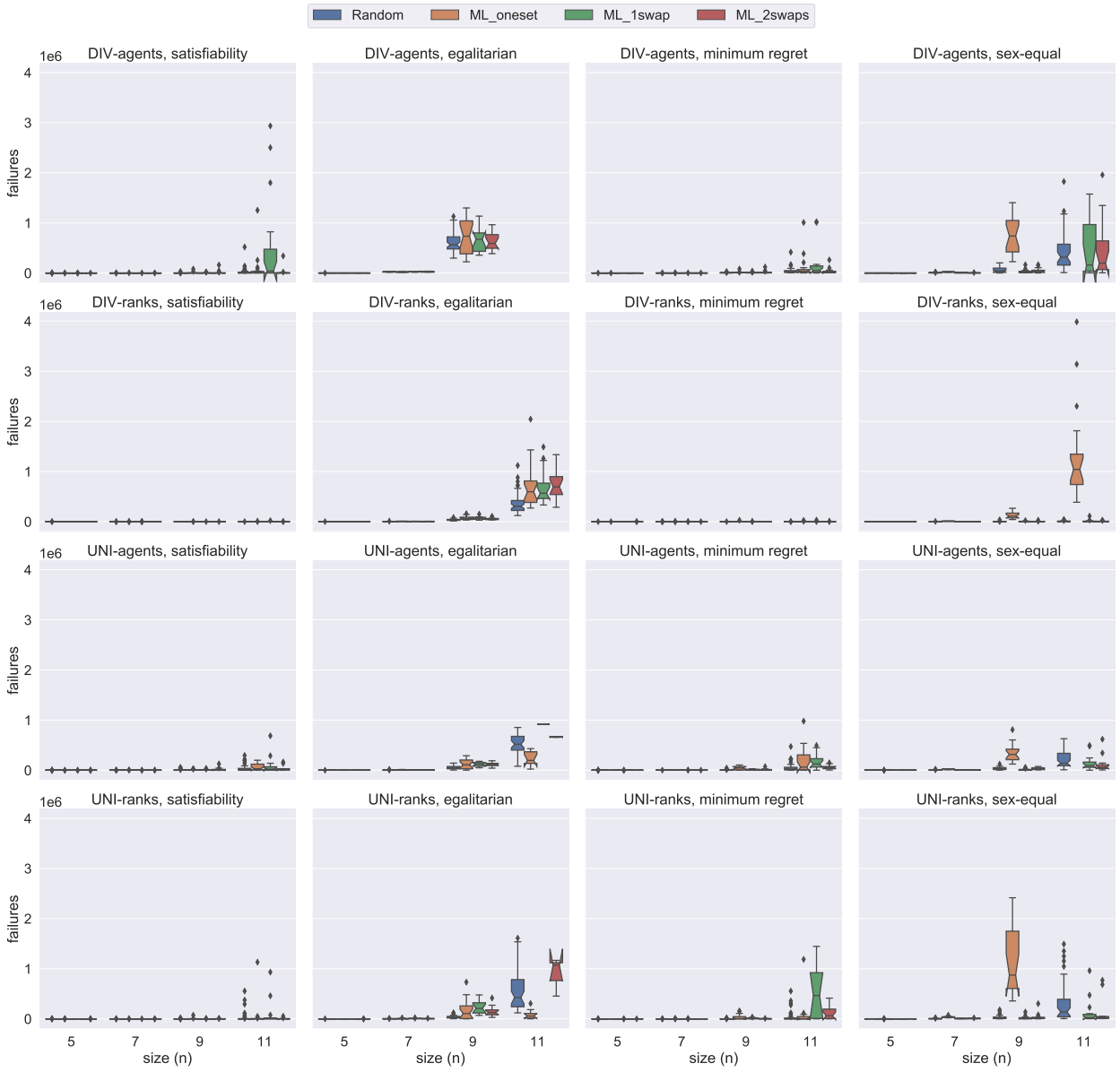


Figure 20: A comparison of the number of failures except HS under strong stability on small instances using Chuffed.

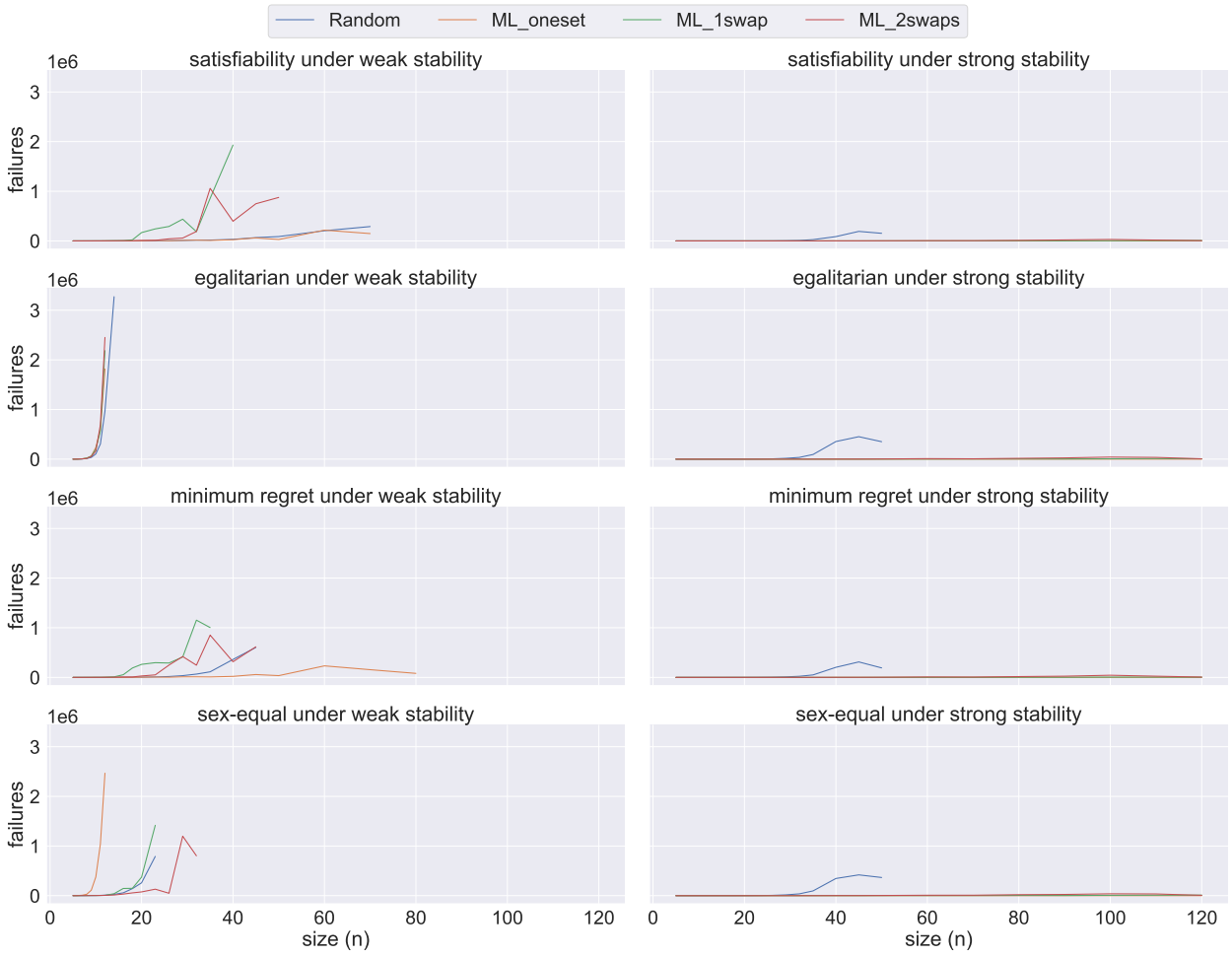


Figure 21: An overview of the number of failures of DIV-ranks using Chuffed under both weak and strong stability when solving problem instances of different sizes for each dataset.