Planning as Satisfiability

Son Ngo, Venecia Xu, Adela Yang

1 Introduction

1.1 Problem Statement

The purpose of the project is to understand the implementation of DPLL-based stochastic satisfiability (SSAT) and to examine the effectiveness of unit clause propagation (UCP), pure variable elimination (PVE) and the three different heuristics that we devise. The main purpose of having heuristics is to guide the solver to choose smartly which variable to consider and appropriate value it should have instead of blindly choosing the next active variable, so that the running time will be improved significantly. In the end, we will look at the results of each algorithm and see what proves to be the most effective and what turns out to be not as effective as we thought it should be.

2 Methodology

2.1 Major Data Structures

In this assignment, we rely heavily on maps, which are very efficient in accessing time as we need to constantly update our variables and clauses as we solve the SSAT problem. With that being said, the two most important maps are *variables* and *clauses*. One big advancement we implement in this assignment is the struct *varInfo*, which is used to store the variables quantifier (or probability if it is a chance variable). This struct also contains a *clauseMembers* whose key is the clause that the variable is in, and value is its sign in the clause. This implementation truly enhances the functionality of our program because it allows us to be able to virtually access information about variables and clauses in constant time. For example, if we want to know the sign of a variable in order to decide whether it is a pure variable or not, instead of going through every entry in the *clauses* map, we only need to look at the clauses that the variable is in, which is conveniently stored in the *clauseMembers* map.

2.2 Splitting Heuristics

A splitting heuristic is a rule for picking the next variable to assign when there are no more unit clauses or pure variables. The following are the three splitting heuristics that we have decided to implement:

2.2.1 Random Variable (RANDOMVAR)

For the random variable heuristic, given the current first block, it chooses a random variable from the block for further exploration. The random variables heuristic was chosen because there is the possibility of better performance on average if the next random variable being split on satisfied many clauses. This serves more as a base comparison for the UCP-PVE and MAXVAR and MINCLAUSE. Finding a random variable also takes constant time, so it is faster than MINCLAUSE or MAXVAR in terms of coming up with a variable.

2.2.2 Maximum Variable (MAXVAR)

For the maximum variable heuristic, given the current first block, it chooses the variable that appears in the most clauses for further exploration. The maximum variable heuristic was chosen because we wanted to choose the variable which appeared in the most clauses because there is a higher chance of the most clauses being satisfied or variable being eliminated from a clause so the process of exploration would speed up. In other words, the problem size will be significantly reduced as we continue to explore on this MAXVAR variable.

2.2.3 Minimum Clause (MINCLAUSE)

Given the current first block, the minimum clause heuristic chooses a variable from the smallest clause for further exploration. There is a higher chance of satisfying clauses with smaller variables, since the probability that any variable will satisfy the clause is increased with less variables. Compare a two variable clause, where either variable has a fifty-percent chance of satisfaction, to a five variable clause, where the chances drop to twenty-percent. The heuristic was chosen with the efficient elimination of clauses in mind to speed up exploration.

In the interest of experimentation, we also created a *maximum* clause (MAXCLAUSE) heuristic. Our first conjecture is that MAXCLAUSE would work better than MINCLAUSE since we reduce the size of the problem by satisfying bigger clauses than smaller clauses. However, our data show the opposite result. MAXCLAUSE is not as effective as MINCLAUSE, but still a little bit better than RANDOMVAR. As a result, we decided to not record the results for MAXCLAUSE, but the algorithm is still implemented in the program.

3 Results & Analysis

3.1 Test Cases

The tests are run on a Macintosh OS X Yosemite computer which has 16GB of memory and a 3.5 GHz Intel Core i7 processor. We run a total of 126 tests on 18 problems, each problem is run with 7 different versions of the DPLL-algorithm (naive, UCP only, PVE only, both UCP and PVE, and our three heuristics). The 18 problems are also divided into 6 subgroups and each subgroup has 3 different clause arrangements:

- The *e* problems have the ordering 36 choice
- The *r* problems have the ordering 36 chance
- The *er* problems have the ordering 18 choice, 18 chance.
- The *re* problems have the ordering 18 chance, 18 choice.
- The *erer* problems have the ordering 9 choice, 9 chance, 9 choice, 9 chance.
- The *rere* problems have the ordering 9 chance, 9 choice, 9 chance, 9 choice.

The tests are generated by the instructor's code with students' arguments and designed in such a way that different ordering of the variables will affect the behavior of the program differently. The order also matters because our heuristics strictly follow this ordering, meaning that we only pick variables that have the same quantifier as the currently active block.

3.2 Results and Discussion

The following is the averages the seven algorithms and their respective properties (results) across all test cases.

Algorithm	Solution Time	# of UCP	# of PVE	# of VS	% of VS
NAIVE	520.8781889	0	0	23875029.83	0.034742762
UCP	17.02099733	795435.9444	0	538000.4444	0.000782894
PVE	208.02355	0	515304.8333	7849527.778	0.011422595
UCPPVE	4.501015389	187092.7778	24081.77778	126400.1667	0.079125687
RANDOMVAR	5.124146556	217753	23244.05556	140584.0556	0.000204577
MAXVAR	1.050697111	33871.72222	8323	21162.22222	3.07951E-05
MINCLAUSE	2.053862833	57283.27778	12057.83333	51763.33333	7.53256E-05

Table 1: Averages all the different algorithms and their properties across all test cases

Due to the large amount of reported data, the full table of results will be appended at the end of this write-up. There are 5 types of information that are recorded: solution time, number of unit clause propagations (UCP), number of pure variable eliminations (PVE), number of variable splits (VS) in both absolute value and the percentage out of the total possible $2^n - 1$ splits. We then carefully study and analyze these information to acquire a better understanding of the behavior of each algorithm and the effectiveness of enhancements and splitting heuristics we add to the native algorithm. Generally, we observe that most of our modifications (except for PVE-only algorithm) improve not only the running time but also the percentage of variable split out of all possible splits.

3.2.1 Analysis of Naive, UCP, PVE and UCP-PVE

Within this section, we only consider the analysis of the naive, UCP, PVE and UCP-PVE algorithms.

According to Table 1, we see that in the realm of the naive, UCP, PVE and UCP-PVE algorithms, the naive algorithm performs the worst with the average time of all the tests amounting to about 521 seconds which is about 115 times the fastest average running time of UCP-PVE of 4.5 seconds. We can see clearly that the naive algorithm is inefficient and has the worst statistics overall, which is also evident in the massive amounts of variable splits which is about 190 times more than UCP-PVE's variable splits. This is because the naive algorithm does not take advantage of UCP or PVE, thus it must explore every possible scenario.

UCP algorithm's average solution time comes in at a close second with its solution time at 17 seconds, which is about 4 times the fastest running time of UCP-PVE. PVE algorithm's average solution time comes in at third at 208 second, which is about 46 times greater than the fastest running time. The same is true for the number of variable splits with UCP coming is second at 4 times the least number of variable splits of UCP-PVE and PVE coming in third at 62 times of the least number of variable splits. It is very clear that between UCP and PVE, UCP's performance is much better and more effective when it comes to the efficiency of finding the solution. PVE is not as reliable because it needs to explore find a pure variable to eliminate and going through all the clauses to find one is extremely time consuming and more likely to be nonexistent, which can clearly be seen because the number of UCP is much greater than PVE between those two algorithms. This also contributes to the higher number of variable splits for PVE.

ID	Algorithm	Test	Solution time	# of UCP	# of PVE	# of VS	% of VS
1	Naive	e3	406.72	0	0	21589091	0.0314163
2	UCP only	e3	56.7467	2419490	0	1853379	0.00269702
3	PVE only	e3	34.3682	0	652946	261360	0.000380329
4	UCP & PVE	e3	3.28609	23867	62414	8531	1.24E-05
5	RANDOMVAR	e3	1.35444	10255	29092	4564	6.64E-06
6	MAXVAR	e3	0.114887	374	1972	146	2.12E-07
7	MINCLAUSE	e3	0.440613	1787	7346	760	1.11E-06

Figure 1: Results for e3 Test Case

However, for file *e3* (see figure 1), the PVE algorithm outperforms UCP, with PVE's time about half the solution time of UCP. In this specific case, PVE's variable splits is $\frac{1}{7}$ that of UCP, greatly contributing to its faster solution time with about a 22 second difference.

UCP-PVE combines the benefits and detriments of UCP and PVE and clearly the benefits outweigh the detriments. The UCP and PVE algorithms combined covers all the benefits of finding all the UCPs and PVEs and together they significantly decreases the number of variable splits needed for explorations and less number of UCP and PVE in general. This in turn contributes to the fastest solution time.

However, for file *r1, r2, r3* (see figure 11, 13), the UCP algorithm outperforms UCP-PVE. However, the greatest difference of the solution times is that UCP-PVE is 1.14 times the UCP solution time and this is not of much significance because it is on a small scale of at most a 4 second difference.

Overall, among these four algorithms, UCP-PVE is the winner in all aspects for the most efficient algorithm.

3.2.2 Analysis of the Splitting Heuristics

In this section, we will specifically analyze the results of heuristics algorithm using the data from Table 1. We will not compare this with the naive algorithm because these heuristics algorithm also apply UCP and PVE modifications, so our base algorithm will be UCP-PVE. The heuristics overally ran quicker than the base algorithm. While MAXVAR and MINCLAUSE showed stellar improvements, RANDOMVAR did not. UCP-PVE clocked in at an average solution time of 4.50 seconds, slightly quicker than RANDOMVAR at 5.12 seconds. This difference is further compounded by RANDOMVARs high number of VS, which was 11.22% higher than UCP-PVE and several times higher than the other heuristics. Picking a variable at random may have even less direction than picking the next in-order variable.

Clearly, RANDOMVAR did not do well as a heuristic. While the remaining two succeeded, MAXVAR was the most effective. At a solution time of 1.05 seconds, it is immensely faster than the naive algorithm, at 520.87 seconds. Even compared to algorithms that ran at vaguely comparable times, like UCP-PVE or MINCLAUSE, the MAXVAR heuristic has far smaller numbers of cases of UCP, and PVE. MAXVAR has roughly 60% less numbers of VS than MINCLAUSE, its closest competitor.

In conclusion, we can conclude that RANDOMVAR performed poorly as a heuristic, as its solution time and the numbers of UCP and VS surpassed UCP-PVE, the base algorithm. MINCLAUSE and MAXVAR extremely fast, at 1.05 and 2.05 seconds of average solution time, respectively. In comparison to the naive and PVE algorithms, the difference between the two heuristics are neglible. Upon close examination, we find again that MAXVAR performs roughly twice as effectively as MINCLAUSE in the areas of solution time, numbers of UCP, and numbers of PVE. Overall, though, we can confirm that the heuristics overwhelmingly improved the UCP-PVE only algorithms.

3.2.3 Overall Analysis

The longest solution time is 1615.14 seconds which was performed by PVE-only algorithm on test *r3* (see figure 12). The fastest solution time is 0.057994 seconds which was performed by MAXVAR algorithm on test *r2* (see figure 15). Among algorithms that run with student-devised heuristics, then MAXVAR-algorithm also produces lowest number of UCP and PVE in test *e3* and *er3* respectively (see figure 15). Interestingly, RANDOMVAR-algorithm produces highest number of UCP in test *r1* (see figure 14) and UCP-PVE combined algorithm produces highest number of PVE in test *re3* (see figure 12). Not as surprisingly, MAXVAR produces lowest number of VS (in both cases) and Naive produces the highest in virtually all of the tests.

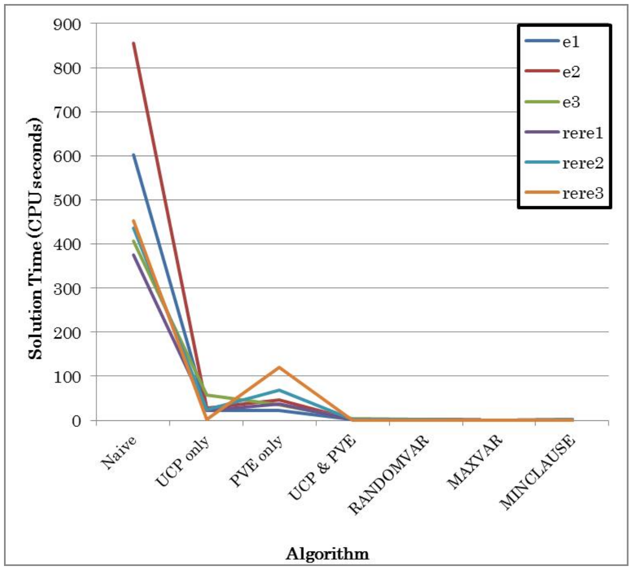

Figure 2: Solution of each method with respect to E and RE tests

The tests are chosen as representative for a range of complexity in the distribution of choice and chance variables. Figure 2 indicates that even though the tests are noticeably different in structures, the behavior of our 7 algorithms are quite consistent despite the distribution of the tests. In general, we can see that all of our modifications to the naive algorithms improve the speed of the program. The student-devised heuristics seem to be the most effective, and then UCP and UCP-PVE and the least effective is PVE, but still quite significant. There are certain cases that PVE is actually worse than naive is because there are so few, or none at all, pure variables that the algorithm is wasting time looking for pure chance variable. This shows most obviously in r tests when there are no choice variable at all.

To look into this more deeply, we can examine the average solution time across all tests of each algorithm from Table 1. The first observation we make is that UCP is the evidently the best improvement in terms of speeding up the solve. We can show this in not just UCP-only algorithm but other algorithms that also employ UCP. UCP-only algorithm's average solution time is only 3.26% of the average of naive solution, which can be considered a extremely significant impact on efficiency. This behavior is also consistent throughout all the test cases. Other than UCP, PVE does help improve the algorithm but only by 60%. As mentioned before, there are cases in which PVE performs worse than naive algorithm simply because there is no pure variable. However, based on the results, given these randomized samples, we would still choose to apply PVE due to its improvement of performance. Even more interestingly, the heuristics are proven to be quite effective, even though no where near as significantly as UCP (or even PVE in some cases). They help to reduce a 4-second problem to a less-than-2-second problem, which can be considered a minor improvement in terms of CPU time. Even though on the data, the solution time of these heuristics seems really fast, we need to compare it with UCP-PVE algorithm since these heuristics are also run with both PVE and UCP enabled. One interesting thing to note is that RANDOMVAR is just a little bit slower (on average) than UCP-PVE, which is not surprising since we are not actually improving the direction of the solver directly by picking a variable in the block randomly. Of course, it is still more "exciting" than simply picking the next in-order variable. Having splitting heuristics in general definitely help with the performance of the solver.

In order to analyze the overall behavior of these algorithms, we can also look at the number of UCP, number of PVE and number of VS across all algorithms. These three records are highly correlated (if not directly related) to the solution time of each algorithm. One interesting thing to note is that even though the naive algorithm simply picks the next in-order variable in the active block, it does not cover all possible splits since as we update the clauses and variables set, the problem size can be reduced significantly in the process. This poses the idea that if we have a good-enough structure that can update itself very quickly, the naive algorithm might not be "too bad." By looking at the average of each properties, we can see that the

faster the algorithm performs, the lower the number of UCP, PVE and VS it has. This comes as a surprise to us at first since we would expect the number of UCP to be higher since the UCP helps significantly reduce the problem size. However, if all three numbers are low then it makes perfect sense to see a better performance from the algorithm with such results. Conclusively, the final ordering of performance of all 7 algorithms can be given as follow (from worst to best):
naive $\longrightarrow$ PVE-only $\longrightarrow$ UCP-only $\longrightarrow$ UCP-PVE $\longrightarrow$ RANDOMVAR $\longrightarrow$ MINCLAUSE $\longrightarrow$ MAXVAR.

4 Conclusion

We can conclude in this assignment that there are many ways to improve the efficiency of the SSAT-solver algorithm. Given the 7 algorithms, it is safe to conclude that MAXVAR was the best algorithm for its efficiency and supreme performance over the other algorithms. The naive algorithm was unsurprisingly the worst and each additional modification has improved its performance. The 6^{th} best algorithm is PVE because there are less PVE compared to UCP in general so not as many variables are eliminated, resulting in mpre variable splits, which is why UCP does better than PVE because there are usually more UCPs so there is a higher chance of other clauses being satisfied, resulting in less variable splits and a faster run time. UCP-PVE tops the naive, UCP-only and PVE-only algorithms because it has the combined strength of the UCP and PVE algorithms, allowing it to be more efficient. RANDOMVAR does slightly better than UCP-PVE. MINCLAUSE does better than RANDOMVAR as it provides a better method of choosing the next variable to split on, but it still does not perform as well as MAXVAR. We can conclude that having UCP and heuristics is very important in the implementation to the DPLL-algorithm.

Appendix

	File Type	Solution Time	# of UCP	# of PVE	# of VS	% of VS
Naïve	e	621.2023333	0	0	28231807	0.041083
	er	412.096	0	0	15462122	0.0225
	erer	175.8973	0	0	7079071	0.010301
	r	432.2061667	0	0	30774553	0.044783
	re	1062.354667	0	0	42907563	0.062439
	rere	421.5126667	0	0	18795063	0.02735
AVERAGE		520.8781889	0	0	23875030	0.034743

Figure 3: Averages of naive properties across each file type

	File Type	Solution Time	# of UCP	# of PVE	# of VS	% of VS
UCP only	e	35.49163333	1554392	0	1092736	0.00159
	er	8.173103333	363530	0	245344.3	0.000357
	erer	8.341983333	335314.7	0	222068.3	0.000323
	r	9.646920667	696005.7	0	528202.3	0.000769
	re	24.6916	1114663	0	695481	0.001012
	rere	15.78074333	708710.7	0	444170.7	0.000646
AVERAGE		17.02099733	795435.9	0	538000.4	0.000783

Figure 4: Averages of UCP properties across each file type

		Solution Time	# of UCP	# of PVE	# of VS	% of VS
PVE Only	e	33.9967	0	596464	381169.7	0.000555
	er	386.761	0	4996.667	11216258	0.016322
	erer	80.40693333	0	234081	1836953	0.002673
	r	594.2808333	0	0	30774553	0.044783
	re	77.59663333	0	1454459	981931	0.001429
	rere	75.0992	0	801828.7	1906302	0.002774
AVERAGE		208.02355	0	515304.8	7849528	0.011423

Figure 5: Averages of PVE properties across each file type

	File Type	Solution Time	# of UCP	# of PVE	# of VS	% of VS
UCP & PVE	e	2.25876	21895.67	42432	7691.667	0.473662
	er	4.852686667	185186.7	1809.333	124327.7	0.000181
	erer	3.444399333	101743	15712	51256.33	7.46E-05
	r	10.997606	696005.7	0	528202.3	0.000769
	re	3.930076667	76069	64590.67	28365.67	4.13E-05
	rere	1.522563667	41656.67	19946.67	18557.33	2.7E-05
AVERAGE		4.501015389	187092.8	24081.78	126400.2	0.079126

Figure 6: Averages of UCP & PVE properties across each file type

	File Type	Solution Time	# of UCP	# of PVE	# of VS	% of VS
RANDOMVAR	e	1.826496667	19533.67	34776.67	6929.333	1.01E-05
	er	4.746483333	176949	1696.333	111654	0.000162
	erer	3.345344333	102671.3	14092.33	50902.33	7.41E-05
	r	14.589017	878342	0	623987	0.000908
	re	4.819803333	93811	70920.33	33247	4.84E-05
	rere	1.417734667	35211	17978.67	16784.67	2.44E-05
AVERAGE		5.124146556	217753	23244.06	140584.1	0.000205

Figure 7: Averages of RANDOMVAR properties across each file type

	File Type	Solution Time	# of UCP	# of PVE	# of VS	% of VS
MAXVAR	e	0.14108	680.3333333	2760.333333	252	3.66708E-07
	er	0.730642667	26012.33333	762.3333333	17339.66667	2.52325E-05
	erer	2.099377	62413	10749	31418.66667	4.572E-05
	r	1.087559667	65033.33333	0	58288.33333	8.48208E-05
	re	1.575241333	34623.33333	27282.33333	12991.33333	1.89049E-05
	rere	0.670282	14468	8384	6683.333333	9.72552E-06
AVERAGE		1.050697111	33871.72222	8323	21162.22222	3.07951E-05

Figure 8: Averages of MAXVAR properties across each file type

	File Type	Solution Time	# of UCP	# of PVE	# of VS	% of VS
MINCLAUSE	e	0.439619667	2713	8078.667	1165.667	1.7E-06
	er	2.27356	73730.67	1353	54591	7.94E-05
	erer	2.523434333	70484.33	12123.33	38210.67	5.56E-05
	r	3.869276	132538.7	0	188327.3	0.000274
	re	2.260586667	41600.67	39404.67	17772.33	2.59E-05
	rere	0.956700333	22632.33	11387.33	10513	1.53E-05
AVERAGE		2.053862833	57283.28	12057.83	51763.33	7.53E-05

Figure 9: Averages of MINCLAUSE properties across each file type

Algorithm	Test	Solution time	# of UCP	# of PVE	# of VS	% of VS
Naive	e1	602.192	0	0	25653485	0.0373307
	e2	854.695	0	0	37452845	0.0545011
	e3	406.72	0	0	21589091	0.0314163
	er1	439.514	0	0	14059535	0.0204593
	er2	232.314	0	0	10970712	0.0159645
	er3	564.46	0	0	21356119	0.0310772
	erer1	307.688	0	0	10946465	0.0159292
	erer2	165.535	0	0	7902204	0.0114992
	erer3	54.4689	0	0	2388545	0.00347579
	r1	1177.11	0	0	89119942	0.129687
	r2	80.5991	0	0	1886246	0.00274485
	r3	38.9094	0	0	1317470	0.00191717
	re1	1349.36	0	0	55331247	0.0805176
	re2	1159.09	0	0	46667661	0.0679104
	re3	678.614	0	0	26723782	0.0388882
	rere1	374.646	0	0	16708281	0.0243137
	rere2	436.781	0	0	21560199	0.0313742
	rere3	453.111	0	0	18116708	0.0263633

Figure 10: Naive Results

Algorithm	Test	Solution time	# of UCP	# of PVE	# of VS	% of VS
UCP only	e1	22.1623	936142	0	573603	0.000834702
	e2	27.5659	1307544	0	851226	0.0012387
	e3	56.7467	2419490	0	1853379	0.00269702
	er1	4.72064	218372	0	97730	0.000142216
	er2	4.93637	235490	0	182683	0.000265839
	er3	14.8623	636728	0	455620	0.000663014
	erer1	14.7961	565340	0	336404	0.000489532
	erer2	8.91265	381745	0	293045	0.000426437
	erer3	1.3172	58859	0	36756	5.35E-05
	r1	28.1959	2056925	0	1567291	0.00228071
	r2	0.213466	9279	0	5425	7.89E-06
	r3	0.531396	21813	0	11891	1.73E-05
	re1	22.4617	1019219	0	659639	0.000959901
	re2	23.4907	1017432	0	699865	0.00101844
	re3	28.1224	1307337	0	726939	0.00105784
	rere1	22.2517	1056088	0	568409	0.000827144
	rere2	23.7836	1014650	0	728864	0.00106064
	rere3	1.30693	55394	0	35239	5.13E-05

Figure 11: UCP Results

Algorithm	Test	Solution time	# of UCP	# of PVE	# of VS	% of VS
PVE only	e1	22.0961	0	451154	241457	0.000351366
	e2	45.5258	0	685292	640692	0.00093233
	e3	34.3682	0	652946	261360	0.000380329
	er1	308.161	0	4130	7194508	0.0104694
	er2	237.026	0	3404	8359073	0.0121641
	er3	615.096	0	7456	18095192	0.026332
	erer1	193.552	0	575869	4209281	0.00612531
	erer2	30.3747	0	73958	860836	0.00125268
	erer3	17.2941	0	52416	440743	0.000641365
	r1	1615.14	0	0	89119942	0.129687
	r2	113.975	0	0	1886246	0.00274485
	r3	53.7275	0	0	1317470	0.00191717
	re1	57.2245	0	1056678	863206	0.00125613
	re2	54.1864	0	1023527	609554	0.000887018
	re3	121.379	0	2283171	1473033	0.00214355
	rere1	36.6792	0	597640	812648	0.00118256
	rere2	68.8354	0	1086135	1589870	0.00231357
	rere3	119.783	0	721711	3316389	0.00482598

Figure 12: PVE Results

Algorithm	Test	Solution time	# of UCP	# of PVE	# of VS	% of VS
UCP & PVE	e1	2.06923	27338	34369	8715	1.27E-05
	e2	1.42096	14482	30513	5829	1.42096
	e3	3.28609	23867	62414	8531	1.24E-05
	er1	2.403	98117	1751	43498	6.33E-05
	er2	1.32726	53669	1168	41901	6.10E-05
	er3	10.8278	403774	2509	287584	0.00041849
	erer1	7.71019	226691	31895	111176	0.000161782
	erer2	2.1967	62669	14252	34836	5.07E-05
	erer3	0.426308	15869	989	7757	1.13E-05
	r1	32.1473	2056925	0	1567291	0.00228071
	r2	0.241078	9279	0	5425	7.89E-06
	r3	0.60444	21813	0	11891	1.73E-05
	re1	2.81443	50670	53899	19011	2.77E-05
	re2	3.06198	54283	44863	17422	2.54E-05
	re3	5.91382	123254	95010	48664	7.08E-05
	rere1	2.46861	72509	33394	28611	4.16E-05
	rere2	1.69514	43888	22676	23494	3.42E-05
	rere3	0.403941	8573	3770	3567	5.19E-06

Figure 13: UCP-PVE Results

Algorithm	Test	Solution time	# of UCP	# of PVE	# of VS	% of VS
RANDOMVAR	e1	2.47973	31710	43822	10634	1.55E-05
	e2	1.64532	16636	31416	5590	8.13E-06
	e3	1.35444	10255	29092	4564	6.64E-06
	er1	2.39821	95891	1197	43210	6.29E-05
	er2	2.40183	89985	1759	63457	9.23E-05
	er3	9.43941	344971	2133	228295	0.000332213
	erer1	7.56704	236254	29648	111391	0.000162095
	erer2	1.93359	53603	11457	32344	4.71E-05
	erer3	0.535403	18157	1172	8972	1.31E-05
	r1	42.3196	2582955	0	1850204	0.0026924
	r2	0.332131	12206	0	5894	8.58E-06
	r3	1.11532	39865	0	15863	2.31E-05
	re1	3.65091	60318	61980	21823	3.18E-05
	re2	5.30701	95854	71268	31263	4.55E-05
	re3	5.50149	125261	79513	46655	6.79E-05
	rere1	1.96566	48976	25275	20587	3.00E-05
	rere2	1.93722	49610	25005	26628	3.87E-05
	rere3	0.350324	7047	3656	3139	4.57E-06

Figure 14: RANDOMVAR Results

Algorithm	Test	Solution time	# of UCP	# of PVE	# of VS	% of VS
MAXVAR	e1	0.139854	728	2906	315	4.58E-07
	e2	0.168499	939	3403	295	4.29E-07
	e3	0.114887	374	1972	146	2.12E-07
	er1	0.501482	18487	764	8528	1.24E-05
	er2	0.484356	17396	839	13417	1.95E-05
	er3	1.20609	42154	684	30074	4.38E-05
	erer1	4.80762	144849	23889	70916	0.000103196
	erer2	1.23826	33542	7620	18603	2.71E-05
	erer3	0.252251	8848	738	4737	6.89E-06
	r1	3.02315	187033	0	170453	0.000248042
	r2	0.057994	1903	0	1245	1.81E-06
	r3	0.181535	6164	0	3167	4.61E-06
	re1	1.42024	25690	27237	9408	1.37E-05
	re2	0.924124	17477	15702	6572	9.56E-06
	re3	2.38136	60703	38908	22994	3.35E-05
	rere1	0.973429	22034	12099	9216	1.34E-05
	rere2	0.739562	15502	10273	8519	1.24E-05
	rere3	0.297855	5868	2780	2315	3.37E-06

Figure 15: MAXVAR Results

Algorithm	Test	Solution time	# of UCP	# of PVE	# of VS	% of VS
MINCLAUSE	e1	0.39897	2940	7833	1364	1.98E-06
	e2	0.479276	3412	9057	1373	2.00E-06
	e3	0.440613	1787	7346	760	1.11E-06
	er1	1.53767	58020	999	28617	4.16E-05
	er2	0.65514	21343	1118	18794	2.73E-05
	er3	4.62787	141829	1942	116362	0.000169329
	erer1	5.6563	157816	26008	84054	0.000122315
	erer2	1.56346	41546	9497	24388	3.55E-05
	erer3	0.350543	12091	865	6190	9.01E-06
	r1	11.0883	382033	0	554646	0.000807116
	r2	0.144313	4271	0	3086	4.49E-06
	r3	0.375215	11312	0	7250	1.06E-05
	re1	1.97955	34820	37566	13630	1.98E-05
	re2	1.48515	22094	25591	9374	1.36E-05
	re3	3.31706	67888	55057	30313	4.41E-05
	rere1	1.36878	34320	15927	13945	2.03E-05
	rere2	1.16187	26906	14911	14659	2.13E-05
	rere3	0.339451	6671	3324	2935	4.27E-06

Figure 16: MINCLAUSE Results