Optimization and Uncertainty: Planning as Satisfiability
Group: Jake Adicoff, Ethan Zhou, Chad Carrera

Different Methods of Literal Propagation and the Observed Results on Satisfiability and Time Efficiency

Description of Work

Given an SSAT problem generated by Professor Majercik's ssat-generator program, our solver will find the satisfying variable assignment with its associated satisfiability probability. Our code implements six splitting heuristics, the last 3 of which we designed ourselves.

1. Variable Ordering Splitting (naïve approach)
2. Unit Clause Propagation
3. Pure Variable Elimination
4. Split on variable in shortest active clause
5. Split on variable that occurs most often on active clauses
6. Split on "purest" variable: i.e. split on the most common (positive/negative) literal

We hypothesized that Heuristic 4 would work well because selecting variables in the shortest clauses would create unit clauses more often than simple the VOS approach. Shorter clauses are also 'harder' to satisfy in the sense that there are fewer opportunities for the clause to be satisfied simply because they contain fewer literals that may be true. As a result, splitting on shorter clauses may reveal unsatisfiable assignments faster.

Heuristic 5 could improve on VOS because it selects the variable that affects the most clauses at once, so with each step, the algorithm can satisfy many clauses *and* reduce the clause length of many others, thus increasing the chance of finding a unit clause. This heuristic aims to change as many clauses as possible with each step. Because such widespread changes are being made each step, improper variable assignments would be found faster.

Heuristic 6 operates on a very similar principle, but prioritizes clause satisfaction over literal elimination. This heuristic will satisfy the most clauses per given step, and thus would perform best in problems with high variance in positive/negative literal distribution. This heuristic is a compromise between PVE and Heuristic 5, and would improve on VOS for the same reasons.

Six tests were run on each problem in addition to the naïve (VOS) approach:

1. UCP when possible, VOS otherwise
2. PVE when possible, VOS otherwise
3. UCP and PVE when possible, VOS otherwise
4. UCP and PVE when possible, Heuristic 1 otherwise
5. UCP and PVE when possible, Heuristic 2 otherwise
6. UCP and PVE when possible, Heuristic 3 otherwise

Experimental Methodology

Initially, the program was tested with small problem sizes amount of time spent debugging, since these tests would run very quickly. There were five test problems three to ten literals, with eight to forty clauses, respectively. These results are not included in the data report, as their solve times are too short to demonstrate differences among the different heuristics.

For the final tests, we generated 18 problems using code provided by Professor Majercik. Each took about 5 minutes (300 seconds) to generate on our own computer. The generated problems each had 34 variables and 120-130 clauses. The problems were created in groups of three, with each group having a unique ordering of variable "blocks" (i.e. choice variable blocks, E, and chance variable blocks, R), where a block is a sequence of variables all with the same variable type. The sequences generated were:

- All Choice Variables: E
- All Chance Variables: R
- Choice, then Chance: ER
- Chance, then Choice: RE
- … ERER
- … RERE

(Three of each type generated for a total of 18 problems)

We ran the baseline VOS/naïve test plus the six tests listed in the above section for all 18 problems, for a total of 126 tests. For each test, we recorded the computation time required to return the proper variable assignment, so that we could compare the relative effectiveness of each method.

Results

The left chart of Figure 2 shows that UCP outperforms PVE, especially for problems that include chance (R) variables. This makes sense, because PVE can only be used within choice (E) variable blocks, while UCP can be used for all variables across all blocks. In fact, PVE ran slower than the naïve method in the problems that only had R variables, since it ran the search for pure variables, but could never find one due to all variables being E. (Our algorithm could have been improved by aborting the search for R variable blocks.) PVE was only comparable to UCP in the test with only E variables.

However, using *both* UCP and PVE was faster than using either method alone, and reduced the problem run time by an average of 98.893%, relative to the naïve method.

The right chart of Figure 2 shows that our three heuristics (splitting variables in the shortest clause, splitting the most popular variable, and splitting the most popular literal) **each** improves on the naïve method, when ran in conjunction with UCP and PVE. They worked best in problems with fewer blocks, but still ran faster in problems with many blocks. This means that all three heuristics we selected improve on the naïve method, as expected by the hypotheses explained in the *Description of Work* section.

Figure 3 and Figure 4 present time improvement averages per problem and overall, which both agree with the above analysis. Figure 4 also shows that splitting on the most popular variable is the most effective of the three heuristics, followed closely by splitting on the most popular literal, then by splitting in the shortest clause.

The fact that splitting on most popular *variable* (heuristic 2) is very comparable to the more granular splitting on most popular *literal* (heuristic 3) approach makes sense, as the two would almost invariably select the same variable every step. This is because the problems were randomly generated, and the positive and negative literals of each variable would very likely be evenly split. Hypothetically, heuristic 3 ought to work better in problems with many variables that have highly uneven literal ratios and a varied variable distribution. This scenario could give rise to instances where selecting the most popular variable may not satisfy nearly as many clauses as selecting the most popular literal.

Figure 5 shows the standard deviations of the runtimes of each solver method across all problem sets, which represents how effective the test is for all problems in general. Large deviations suggest that the method works well for some problems but not others, while small deviations suggest that the method works well across all problem types. UCP + PVE is the most consistent method compared to either of the two methods alone, while all three heuristics are fairly evenly distributed.

These data suggests that the best approach among our test methods of solving SSAT problems is to use UCP, PVE, and Heuristic 2 simultaneously, instead of VOS.

Figure 1:

Test Times, in Seconds

	VOS	UCP	PVE	UCP,PVE	UCP,PVE,H1	UCP,PVE,H2	UCP,PVE,H3
E1	533.69	15.53	10.4	0.14	0.05	0.01	0.02
E2	286.19	0.159	13.16	0.008	0.002	0.001	0.001
E3	307.56	16.68	6.59	0.061	0.05	0.008	0.011
R1	311.73	17.535	347.4	18.323	2.712	0.931	1.403
R2	446.22	10.75	494.03	10.695	1.273	0.62	0.841
R3	308.73	15.58	335.87	15.71	1.04	0.503	0.697
ER1	318.49	20.27	202.23	6.768	0.845	0.517	0.667
ER2	445.73	10.648	449.1	8.659	0.703	0.458	0.713
ER3	571.55	8.539	462.07	0.467	0.342	0.217	0.409
RE1	249.29	14.35	12.801	0.242	0.144	0.152	0.137
RE2	348.51	3.808	8.309	0.073	0.024	0.019	0.022
RE3	367.56	11.59	9.03	0.18	0.09	0.07	0.09
ERER1	247.83	13.983	50.27	1.462	1.244	0.955	0.893
ERER2	466.67	8.558	121.34	2.123	1.349	1.243	1.264
ERER3	574.49	8.867	103.08	1.006	0.961	0.829	0.773
RERE1	248.47	14.489	34.434	0.713	0.388	0.299	0.346
RERE2	395.67	23.238	96.312	1.912	1.134	1.026	1.095
RERE3	293.92	15.683	77.294	0.609	0.304	0.249	0.28

Each problem (left column) was ran with seven different tests (top row). The VOS method is used as a control.

Optimization and Uncertainty: Planning as Satisfiability
Group: Jake Adicoff, Ethan Zhou, Chad Carrera

Figure 2:

% Time Improvement, Relative to VOS method Runtime

	UCP	PVE	UCP,PVE
E1	97.090%	98.051%	99.974%
E2	99.944%	95.402%	99.997%
E3	94.577%	97.857%	99.980%
R1	94.375%	-11.443%	94.122%
R2	97.591%	-10.714%	97.603%
R3	94.954%	-8.791%	94.911%
ER1	93.636%	36.504%	97.875%
ER2	97.611%	-0.756%	98.057%
ER3	98.506%	19.155%	99.918%
RE1	94.244%	94.865%	99.903%
RE2	98.907%	97.616%	99.979%
RE3	96.847%	97.543%	99.951%
ERER1	94.358%	79.716%	99.410%
ERER2	98.166%	73.999%	99.545%
ERER3	98.457%	82.057%	99.825%
RERE1	94.169%	86.142%	99.713%
RERE2	94.127%	75.659%	99.517%
RERE3	94.664%	73.702%	99.793%

% Time Improvement, Relative to UCP + PVE Runtime

	Heuristic 1	Heuristic 2	Heuristic 3
E1	64.286%	92.857%	85.714%
E2	75.000%	87.500%	87.500%
E3	18.033%	86.885%	81.967%
R1	85.199%	94.919%	92.343%
R2	88.097%	94.203%	92.137%
R3	93.380%	96.798%	95.563%
ER1	87.515%	92.361%	90.145%
ER2	91.881%	94.711%	91.766%
ER3	26.767%	53.533%	12.420%
RE1	40.496%	37.190%	43.388%
RE2	67.123%	73.973%	69.863%
RE3	50.000%	61.111%	50.000%
ERER1	14.911%	34.679%	38.919%
ERER2	36.458%	41.451%	40.462%
ERER3	4.473%	17.594%	23.161%
RERE1	45.582%	58.065%	51.473%
RERE2	40.690%	46.339%	42.730%
RERE3	50.082%	59.113%	54.023%

The left chart shows the relative time improvement of running UCP alone, PVE alone, and UCP with PVE **against the VOS method runtime**.

The right chat shows the relative time improvement of Heuristic 1 (split using shortest clause), Heuristic 2 (split on most popular variable, both positive and negative), and Heuristic 3 (split on most popular literal) **against the UCP + PVE runtime.**

Cells for each table are *independently* colored on a gradient based on percentile: the lowest values are red, near-median values are yellow, highest values are green.

*The percentages are relative to different values since the three heuristics were all *very* fast relative to VOS. Comparing to the UCP + PVE test is more meaningful since our heuristics ran in addition to those methods, so using UCP + PVE as a new baseline provides more meaningful data on the individual heuristics.

Figure 3:

Average % Time Improvement per problem type, Relative to VOS method Runtime

	UCP	PVE	UCP,PVE
E	97.204%	97.103%	99.984%
R	95.640%	-10.316%	95.546%
ER	96.584%	18.301%	98.617%
RE	96.666%	96.675%	99.944%
ERER	96.994%	78.591%	99.593%
RERE	94.320%	78.501%	99.674%

Average % Time Improvement per problem type, Relative to UCP + PVE Runtime

	Heuristic 1	Heuristic 2	Heuristic 3
E	52.440%	89.081%	85.060%
R	88.892%	95.307%	93.348%
ER	68.721%	80.202%	64.777%
RE	52.540%	57.425%	54.417%
ERER	18.614%	31.241%	34.181%
RERE	45.452%	54.506%	49.409%

The left chart shows the *average of three* relative time improvement of running UCP alone, PVE alone, and UCP with PVE **against the VOS method runtime**.

The right chat shows the *average of* three relative time improvement of Heuristic 1 (split using shortest clause), Heuristic 2 (split on most popular variable, both positive and negative), and Heuristic 3 (split on most popular literal) **against the UCP + PVE runtime.**

Cells for each table are *independently* colored on a gradient based on percentile: the lowest values are red, near-median values are yellow, highest values are green.

Figure 4:

Overall Average % Time Improvement, Relative to VOS method Runtime

	UCP	PVE	UCP,PVE
Average	96.235%	59.809%	98.893%

Overall Average % Time Improvement, Relative to UCP + PVE Runtime

	Heuristic 1	Heuristic 2	Heuristic 3
Average	54.443%	67.960%	63.532%

Figure 5:

Standard Deviation in % Time Improvements Across All Problem Tests

	UCP	PVE	UCP,PVE	Heuristic 1	Heuristic 2	Heuristic 3
Std Dev	0.020700	0.427783	0.017721	0.284358	0.256064	0.268590

Optimization and Uncertainty: Planning as Satisfiability

Group: Jake Adicoff, Ethan Zhou, Chad Carrera

Conclusions

Of the six different SSAT solver methods we tested, almost all improved on the naïve Variable Ordering Splitting method, with the exception of Pure Variable Elimination, as its scope is the narrowest of the tested heuristics. However, combining Unit Clause Propagation and Pure Variable Elimination dramatically reduced the computation time, from solving problems on the order 5 to 10 minutes to the order of fractions of a second. Applying non-naïve splitting methods in addition to UCP + PVE further reduces the computation time, especially for problems with few choice/chance blocks.

Overall, the best approach (of the methods tested in this experiment) to solve any given SSAT problem is to apply UCP, PVE, and splitting on the most occurring variable, which improves on the naïve VOS method computation time by over 99%.