Mike Leske

R00183658

COMP9058 - Metaheuristic Optimization

Assignment 2 – Local Search

Table of Contents

Preface

All experiments documented as part of this report were executed on the same computer:

- Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
- 16 GB 2400 MHz DDR4

For each experiment the following statistics will be listed:

- Executions Overview: Total & Mean execution runtime in µsec, success rate, total iterations across all executions
- Restarts: Statistics on search restarts
- Total Iterations: Statistics on total search iterations
- Iterations to solve: Statistics for the search runs that solved the instance

Note: Statistics shown exclude unsuccessful executions according to lecture slides ("Protocol for obtaining the empirical RTD…")

Note: Time granularity based on µsec has been chosen to evaluate time duration of individual search iterations.

Note: Time measurement is based on Python time.perf_counter_ns(), which offers the largest available time resolution (10e-9 vs 10e-6 for time.process_time_ns()). It must be noted the perf_counter_ns() also includes IDLE times, but especially when measuring the uf20-* instances with WalkSAT the gain in time resolution outweighs the potential missing restriction to process time only. Larger SAT instances should be measured using time.process_time_ns(). [1]

Note: Only Section 3 Run-Time Distribution Evaluation expands on problem setup and restart time.

1 GWSAT

This section analyses the impact of the GWSAT parameters (restarts, iterations, wp) on the algorithm's operations against a selected set of CNF instances (uf20-01, uf20-02, uf50-01). The number of executions is fixed to 30. All statistics will be collected with a random seed defined for experiment repeatability.

The default behaviour of GSAT is to consider every variable in every iteration, calculate the effect of each of the potential flip and select the variable that provides the largest net_gain – which can also be negative. This greedy operation can lead to long plateaus and the algorithm getting stuck in a local minimum altogether. The default behaviour for the algorithm to escape from a local minimum is "restarting" after a fixed number of iterations. Such a restart creates a new variable assignment and the GSAT algorithm starts again. Considering that a classical GSAT step takes every variable into consideration in every iteration one might not speak of a solution intensification, despite GSAT's main metric of selecting the variable with the minimal net_gain.

By introducing the random walk parameter "wp" GWSAT provides a second option for the algorithm to escape a local minimum. "wp" represents the probability with which GWSAT will randomly select a variable currently participating in at least one unsatisfied clause and flip it, no matter how good and bad this move is. In iterations with a random walk, GWSAT will not evaluate the net_gain of any other variable. The random walk provides a means of diversifying the search space.

Therefore, a careful selection of the walk parameter "wp" can assist the GWSAT algorithm with:

1. Escape local minima
2. Reduce algorithm runtime

The other 2 parameters "restart" and "max_iterations" have a strong relationship. The parameter "max_iterations" defines how many search steps are executed before the algorithm starts over. While reducing "max_iterations" can upper bound the CPU cycles or time spent in a single search round, such change will result in an increase of restarts the GWSAT algorithm will have to go through. Therefore, depending on the size of the problem instance the parameters "restart" and "max_iterations" need to be carefully chosen, i.e. reducing "max_iterations" to less than the average iterations required to solve a given instance has to be avoided.

A large number of experiments will be executed and documented on the next pages to demonstrate the parameters' impact on the algorithm's operation. The parameters' impact will first be measured against algorithm search performance of the uf50-01 instance. Afterwards, found hypothesis will be validated against the small uf20 instances as well.

1.1 Evaluation of "wp"

Experiment	Instance	Executions	Restarts	Iterations	wp
1.1.0	uf50-01	30	10	1000	0.4
1.1.1	uf50-01	1	10	1000	0.001
1.1.2	uf50-01	1	10	1000	0.05
1.1.3	uf50-01	1	10	1000	0.2
1.1.4	uf50-01	1	10	1000	0.4
1.1.5	uf50-01	1	10	1000	0.6
1.1.6	uf50-01	1	10	1000	0.8
1.1.7	uf50-01	30	10	1000	0.001
1.1.8	uf50-01	30	10	1000	0.05
1.1.9	uf50-01	30	10	1000	0.2
1.1.10	uf50-01	30	10	1000	0.6
1.1.11	uf50-01	30	10	1000	0.8

Experiment 1.1.0 represents a baseline experiment against instance uf50-01 with parameters provided in the assignment brief. From the statistics we can see that on average 370 iterations are needed for search runs (restarts) where a solution is found. We also see that the longest search took 3379 iterations, that is 3 restarts after 1000 iterations without finding a solution plus another 379 iterations to solve the instance.

Experiment	Execution Overview	Restarts	Total Iterations	Iterations to solve
1.1.0	Total : 17718895 µsec Mean : 590629 µsec Success rate: 100.00% Total iter: 27072 Avg iter time: 655 µsec	Sum : 16 Min : 0 Max : 3 Mean : 0.5 Median: 0.0 STDEV : 0.8	Sum : 27072 Min : 23 Max : 3379 Mean : 902.4 Median: 548.5 STDEV : 821.9 VarCo : 0.911	 Min : 23 Max : 876 Mean : 369.1 Median: 363.5 STDEV : 204.2 VarCo : 0.553

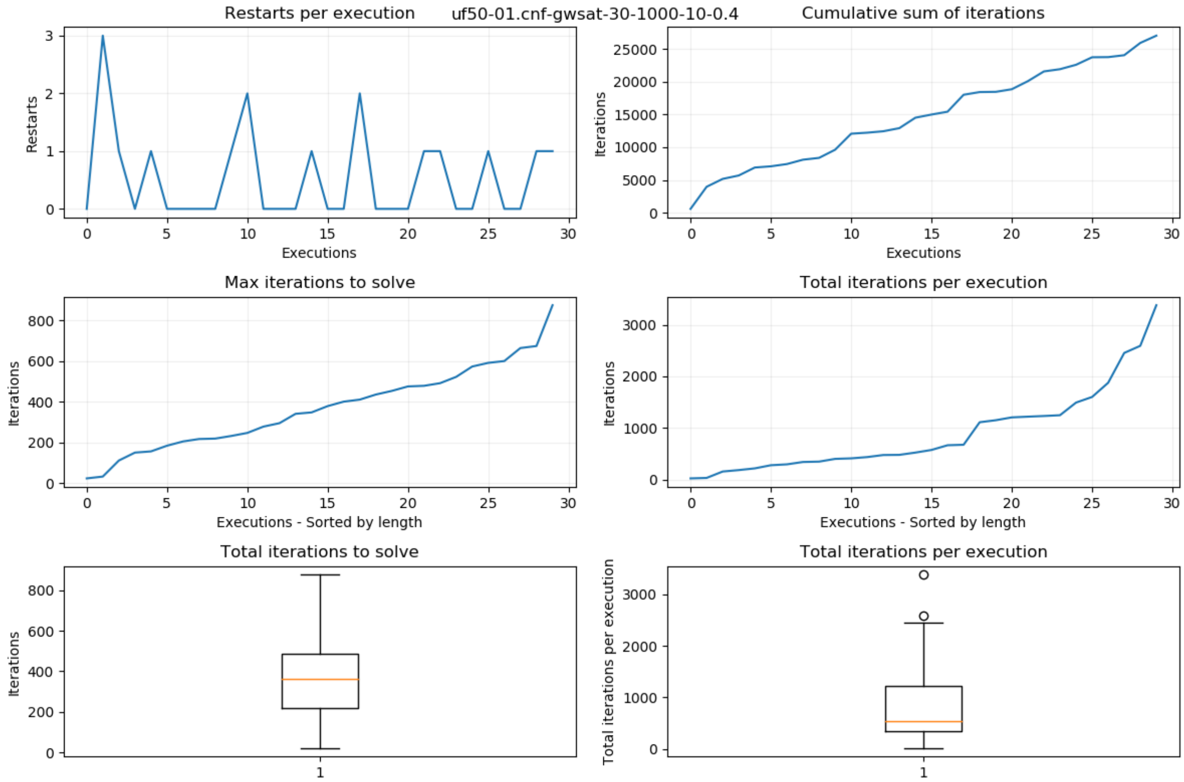
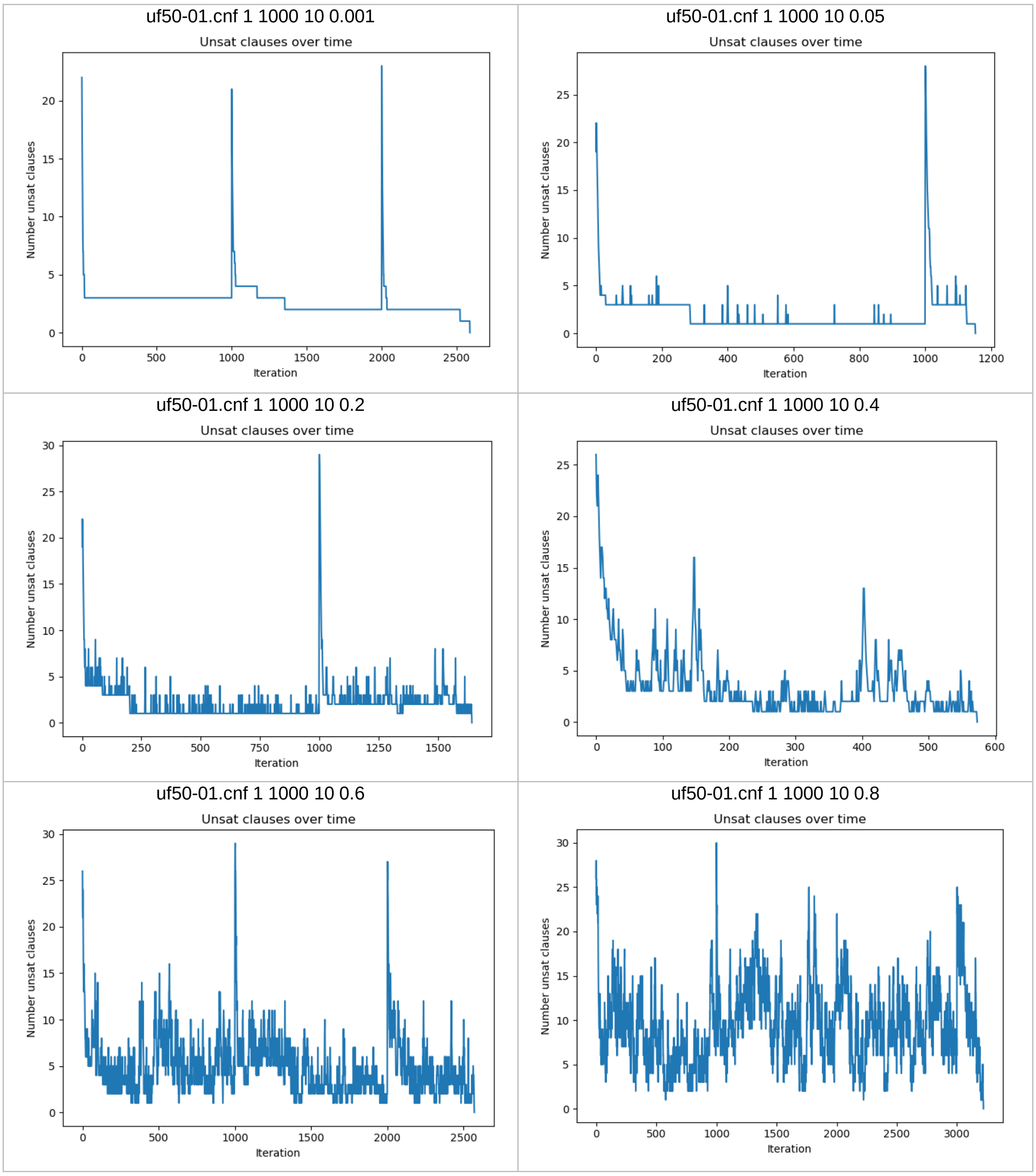


Figure 1 Restart and Iterations statistics for experiment 1.1.0

Next, experiments 1.1.1 to 1.1.6 will be run with a single execution in order to visualize the impact of parameter “wp” on the number of unsatisfied clauses during the search operation. I will demonstrate the impact of wp = [0.001, 0.05, 0.2, 0.4, 0.6, 0.8]. Given the default “max_iterations” of 1000 these “wp” probabilities will lead 0.01%, 5%, 20%, 40%, 60% and 80% expected random walk steps, or 1 out of 1000 to 800 out of 1000.



From experiment 1.1.1 (wp = 0.001) we can see that GWSAT is almost exclusively relying on a search restart to escape a local minimum. The same seems to be true for experiment 1.1.2. One notices a couple of spikes attributed to the random walk component of GWSAT, but with low “wp” probabilities the next iteration relies on classical GSAT operations which immediately reverts the random step and the algorithm is stuck in the local minimum again.

Experiments 1.1.4 seems to visualize a sweet spot for the random walk component in GWSAT. The random walk probability is high enough to ensure a low number of subsequent random walks to escape a local minimum and avoid the immediate reversal of the random walk step. Experiment 1.1.3 is hard to judge. One notices few isolated subsequent steps resulting in a higher number of unsatisfied clauses, but in general the algorithm quickly drops back into the local minimum it tried to escape.

Experiments 1.1.5 and 1.1.6 more and more develop into an uncontrolled and purely random search and seem to be driven by less suitable “wp” values (0.6 and 0.8).

For comparison lets run experiments 1.1.7 to 1.1.11 with 30 executions. Please note that a 30 executions experiment with wp=0.4 was already executed as part of experiment 1.1.0. The results for experiment 1.1.0 are repeated for comparability.

Experiment	Execution Overview	Restarts	Total Iterations	Iterations to solve
1.1.0	Total : 17718895 µsec Mean : 590629 µsec Success rate: 100.00% Total iter: 27072 Avg iter time: 655 µsec	Sum : 16 Min : 0 Max : 3 Mean : 0.5 Median: 0.0 STDEV : 0.8	Sum : 27072 Min : 23 Max : 3379 Mean : 902.4 Median: 548.5 STDEV : 821.9 VarCo : 0.911	Min : 23 Max : 876 Mean : 369.1 Median: 363.5 STDEV : 204.2 VarCo : 0.553
1.1.7	Total : 108571670 µsec Mean : 3619055 µsec Success rate: 96.67% Total iter: 95519 Avg iter time: 1027 µsec	Sum : 87 Min : 0 Max : 8 Mean : 3 Median: 2 STDEV : 2.5	Sum : 95519 Min : 39 Max : 8860 Mean : 3293.8 Median: 2589 STDEV : 2580.1 VarCo : 0.783	Min : 23 Max : 902 Mean : 293.8 Median: 242 STDEV : 248.9 VarCo : 0.847
1.1.8	Total : 74070084 µsec Mean : 2469002 µsec Success rate: 96.67% Total iter: 65571 Avg iter time: 981 µsec	Sum : 55 Min : 0 Max : 6 Mean : 1.9 Median: 1 STDEV : 1.9	Sum : 65571 Min : 42 Max : 6740 Mean : 2261.1 Median: 1790 STDEV : 1925.9 VarCo : 0.852	Min : 18 Max : 961 Mean : 364.5 Median: 193 STDEV : 339.7 VarCo : 0.932
1.1.9	Total : 39297546 µsec Mean : 1309918 µsec Success rate: 100.00% Total iter: 46080 Avg iter time: 853 µsec	Sum : 35 Min : 0 Max : 6 Mean : 1.2 Median: 1.0 STDEV : 1.8	Sum : 46080 Min : 20 Max : 6961 Mean : 1536 Median: 1062.5 STDEV : 1875.4 VarCo : 1.221	Min : 20 Max : 961 Mean : 369.3 Median: 323.0 STDEV : 255.7 VarCo : 0.692
1.1.10	Total : 16773580 µsec Mean : 559119 µsec Success rate: 100.00% Total iter: 37762 Avg iter time: 444 µsec	Sum : 22 Min : 0 Max : 6 Mean : 0.7 Median: 0.0 STDEV : 1.3	Sum : 37762 Min : 120 Max : 6864 Mean : 1258.7 Median: 677.5 STDEV : 1398.6 VarCo : 1.111	Min : 120 Max : 946 Mean : 525.4 Median: 571.0 STDEV : 215.0 VarCo : 0.409
1.1.11	Total : 33371557 µsec Mean : 1112385 µsec Success rate: 86.67% Total iter: 99583 Avg iter time: 240 µsec	Sum : 85 Min : 0 Max : 9 Mean : 3.3 Median: 3.0 STDEV : 2.7	Sum : 99583 Min : 328 Max : 9838 Mean : 3830.1 Median: 3218.5 STDEV : 2736.2 VarCo : 0.714	Min : 52 Max : 1000 Mean : 560.9 Median: 562.5 STDEV : 271.5 VarCo : 0.484

By comparing absolute search times experiments 1.1.0 and 1.1.10 seem to be on par, but the total number of iterations required is substantially smaller in experiment 1.1.0. However, in experiment 1.1.10 a 60% of the iterations on average were driven by the random walk operation so that the number of classical GSAT iterations is almost identical between experiments 1.1.0 and 1.1.10. One should still note that the mean iterations to solve the problem is much smaller for experiment 1.1.0.
As noted above high probabilities for the random walk parameter “wp” lead to a diversification of the search space resulting in more states to be analysed.

Figure 2 visualizes the key performance indicators discussed above.

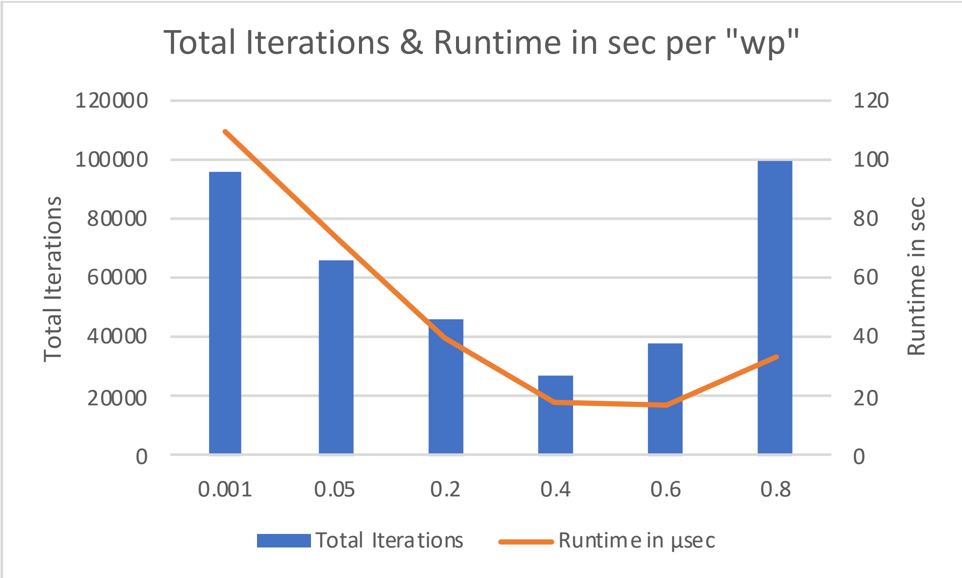


Figure 2 Total Iterations and Runtime in sec per wp parameter

Figure 3 visualizes how the selection of “wp” affects the runtime of a single search iteration. The larger the walk probability “wp” is the more iterations the algorithm will shortcut its classical GSAT protocol. Even for small problem instances with 20 variables not calculating the net_gain for each variable in every iteration shows a huge performance improvement.

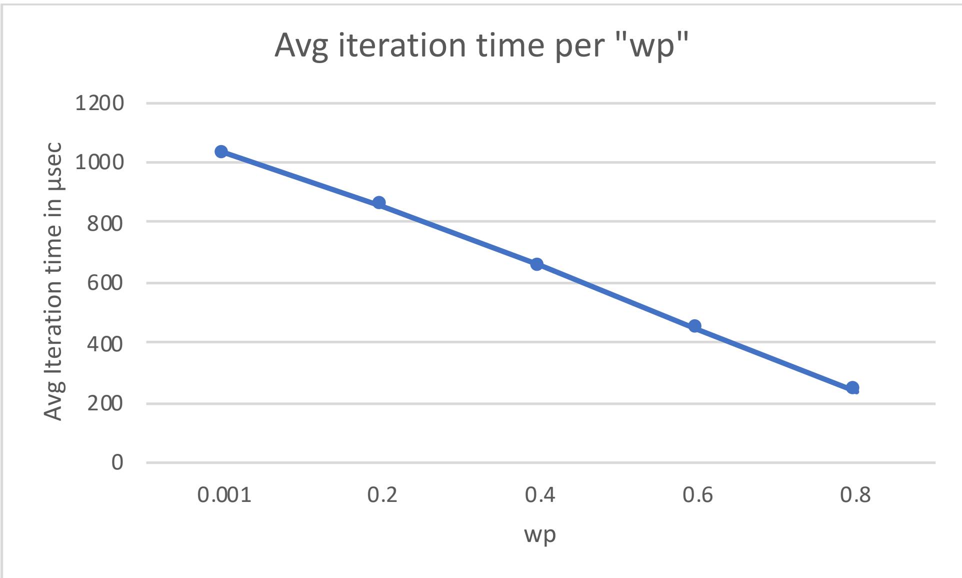


Figure 3 Average Iteration time in µsec per wp parameter

This also highlights GWSATs biggest weakness: With moderate settings of “wp” its time complexity is close to O(m*n), where m is the number of search iterations and n the number of variables in a SAT instance.

1.2 Evaluation of “restart” and “max_iterations”

As already noted above the “restart” parameter is strongly affected by the “max_iterations” parameter. A search that cannot find a solution in a given run, will result in a search restart or terminate without result.

We have seen in section 1.1 that for experiment 1.1.0 the average number of iterations when solving instance uf50-01 is 369 and from Figure 1 one can read that in 27 out of 30 executions the search run finding a solution required 600 or less iterations. This provides already a framing for the next experiments where the number of iterations will be set to 200, 400, 600 and 800 respectively.

Experiment	Instance	Executions	Restarts	Iterations	wp
1.2.1	uf50-01	30	10	200	0.4
1.2.2	uf50-01	30	10	400	0.4
1.2.3	uf50-01	30	10	500	0.4
1.2.4	uf50-01	30	10	600	0.4
1.2.5	uf50-01	30	10	800	0.4
1.2.6	uf50-01	30	10	1200	0.4

Experiment	**Execution Overview**	**Restarts**	**Total Iterations**	**Iterations to solve**
1.2.1	Total : 18373665 µsec Mean : 612455 µsec Success rate: 83.33% Total iter: 18549 Avg iter time: 644 µsec	Sum : 78 Min : 0 Max : 9 Mean : 3.1 Median: 2 STDEV : 2.9	Sum : 18549 Min : 56 Max : 1892 Mean : 742.0 Median: 575 STDEV : 568.6 VarCo : 0.766	 Min : 16 Max : 199 Mean : 118.0 Median: 115 STDEV : 52.6 VarCo : 0.446
1.2.2	Total : 17693343 µsec Mean : 589778 µsec Success rate: 100.00% Total iter: 27659 Avg iter time: 640 µsec	Sum : 55 Min : 0 Max : 9 Mean : 1.8 Median: 1.0 STDEV : 2.5	Sum : 27659 Min : 38 Max : 3929 Mean : 922.0 Median: 516.0 STDEV : 1035.5 VarCo : 1.123	 Min : 38 Max : 391 Mean : 188.6 Median: 156.0 STDEV : 106.8 VarCo : 0.566
1.2.3	Total : 17732704 µsec Mean : 591090 µsec Success rate: 100.00% Total iter: 27107 Avg iter time: 654 µsec	Sum : 40 Min : 0 Max : 6 Mean : 1.3 Median: 1.0 STDEV : 1.6	Sum : 27107 Min : 109 Max : 3244 Mean : 903.6 Median: 717.0 STDEV : 789.2 VarCo : 0.873	 Min : 70 Max : 477 Mean : 236.9 Median: 229.5 STDEV : 119.3 VarCo : 0.504
1.2.4	Total : 18565400 µsec Mean : 618846 µsec Success rate: 100.00% Total iter: 28772 Avg iter time: 645 µsec	Sum : 34 Min : 0 Max : 4 Mean : 1.1 Median: 1.0 STDEV : 1.3	Sum : 30573 Min : 57 Max : 3765 Mean : 1019.1 Median: 810.5 STDEV : 860.5 VarCo : 0.844	 Min : 37 Max : 599 Mean : 279.1 Median: 251.5 STDEV : 163.3 VarCo : 0.585
1.2.5	Total : 19678288 µsec Mean : 655942 µsec Success rate: 100.00% Total iter: 30573 Avg iter time: 644 µsec	Sum : 27 Min : 0 Max : 4 Mean : 0.9 Median: 0.5 STDEV : 1.1	Sum : 27072 Min : 23 Max : 3379 Mean : 902.4 Median: 548.5 STDEV : 821.9 VarCo : 0.911	 Min : 57 Max : 749 Mean : 299.1 Median: 201.0 STDEV : 224.2 VarCo : 0.749
1.2.6	Total : 18795113 µsec Mean : 626503 µsec Success rate: 100.00% Total iter: 29245 Avg iter time: 643 µsec	Sum : 11 Min : 0 Max : 3 Mean : 0.4 Median: 0.0 STDEV : 0.8	Sum : 29245 Min : 108 Max : 4721 Mean : 974.8 Median: 594.5 STDEV : 1078.0 VarCo : 1.106	 Min : 63 Max : 1177 Mean : 534.8 Median: 535.0 STDEV : 343.5 VarCo : 0.642

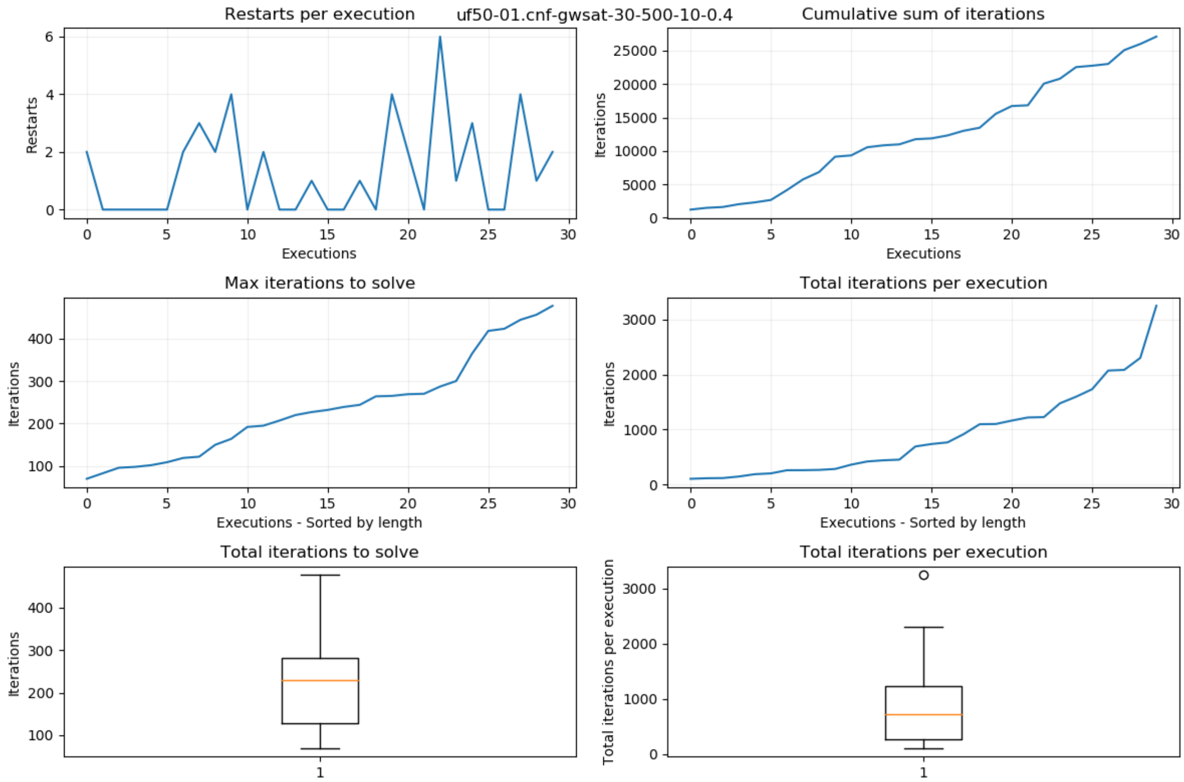


Figure 4 Restart and Iterations statistics for experiment 1.2.3

It seems that for solving instance uf50-01 the limitation of “max_iterations” to 500 is a reasonable choice. The number of overall iterations to solve all 30 executions is only minimally larger than experiment 1.1.0, but the reduced “max_iterations” parameter may help in estimating worst case runtimes. Also with maximum 6 restarts per execution, this setup is still reasonably far away from terminating the search without finding a solution. With “wp” being fixed to 0.4 the computation time per iteration is also fixed.

It should be noted that larger SAT problems may require larger “max_iterations” and “restart” parameters.

1.3 Evaluating instances uf20-01 and uf20-02

After having analysed the impact of the parameters to the larger uf50-01 instance, a few validation checks are run against instances uf20-01 and uf20-02.

Experiment	Instance	Executions	Restarts	Iterations	wp
1.3.1	uf20-01	30	10	1000	0.2
1.3.2	uf20-01	30	10	1000	0.4
1.3.3	uf20-01	30	10	1000	0.6
1.3.4	uf20-02	30	10	1000	0.2
1.3.5	uf20-02	30	10	1000	0.4
1.3.6	uf20-02	30	10	1000	0.6

Experiment	Execution Overview Overview	Restarts	Total Iterations	Iterations to solve
1.3.1	Total : 517798 µsec Mean : 17259 µsec Success rate: 100.00% Total iter: 2187 Avg iter time: 237 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 2187 Min : 6 Max : 656 Mean : 72.9 Median: 31.0 STDEV : 120.2 VarCo : 1.649	Min : 6 Max : 656 Mean : 72.9 Median: 31.0 STDEV : 120.2 VarCo : 1.649
1.3.2	Total : 202389 µsec Mean : 6746 µsec Success rate: 100.00% Total iter: 1123 Avg iter time: 180 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 1123 Min : 6 Max : 109 Mean : 37.4 Median: 27.0 STDEV : 31.1 VarCo : 0.83	Min : 6 Max : 109 Mean : 37.4 Median: 27.0 STDEV : 31.1 VarCo : 0.83
1.3.3	Total : 264895 µsec Mean : 8829 µsec Success rate: 100.00% Total iter: 1943 Avg iter time: 136 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 1943 Min : 8 Max : 246 Mean : 64.8 Median: 47.0 STDEV : 51.7 VarCo : 0.799	Min : 8 Max : 246 Mean : 64.8 Median: 47.0 STDEV : 51.7 VarCo : 0.799
1.3.4	Total : 201304 µsec Mean : 6710 µsec Success rate: 100.00% Total iter: 781 Avg iter time: 258 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 781 Min : 4 Max : 118 Mean : 26.0 Median: 17.0 STDEV : 23.8 VarCo : 0.915	Min : 4 Max : 118 Mean : 26.0 Median: 17.0 STDEV : 23.8 VarCo : 0.915
1.3.5	Total : 198928 µsec Mean : 6630 µsec Success rate: 100.00% Total iter: 1060 Avg iter time: 188 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 1060 Min : 9 Max : 124 Mean : 35.3 Median: 24.5 STDEV : 29.0 VarCo : 0.82	Min : 9 Max : 124 Mean : 35.3 Median: 24.5 STDEV : 29.0 VarCo : 0.82
1.3.6	Total : 153340 µsec Mean : 5111 µsec Success rate: 100.00% Total iter: 1155 Avg iter time: 133 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 1155 Min : 4 Max : 121 Mean : 38.5 Median: 25.0 STDEV : 29.7 VarCo : 0.771	Min : 4 Max : 121 Mean : 38.5 Median: 25.0 STDEV : 29.7 VarCo : 0.771

Using a random walk probability of 0.4 against the smaller SAT problem instances generally proves useful as well. The “max_iterations” parameter could also be safely reduced to 200 without affecting any of the above results. With small problem instances also the total search execution but also the runtime per iteration is proportionally lower.

1.4 Conclusion

The previous experiments have given insights into how the operation of GWSAT is affected by its set of parameters “restart”, “max_iterations” and “wp”. A too low setting for “wp” leads to the random walk component being invisible or reverted immediately by a following classical GSAT step. It seems that with a “wp” value around 0.4 the probability of executing multiple random walk steps in sequence is large enough to be able to escape local minima.

The “wp” probabilities between 0.4 and 0.6 offer an interesting trade-off between a higher number of total iterations, but also more iterations being short-cut by a random walk. “wp” values higher than 0.6 seem to end in a close(r) to random search.

Selecting an appropriate value for “max_iterations” depends on the problem size to work with and how many iterations are usually required to solve a problem. If pre-maturely a search is stopped in favour of a restart, the search needs to work down the number of unsatisfied clauses after re-initializing and consuming extra time.

As noted above, GWSAT’s tendency to evaluate the flip of every variable for classical GSAT steps, makes it scale very badly to large problem instances.

1.5 Debug GWSAT

To debug the operation of the GWSAT implementation set the DEBUG flag to TRUE in file ‘Leske_183658_GWSAT.py’.

This will create an output as follows:

```
…
total_unsat:    2
candidate_vars: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
gains:          [-5, -1, -3, 0, -1, -3, -3, -3, -2, -2, -1, -1, 0, -2, -1, -1, -1, -4, -2, -4]
flip_var:       4

total_unsat:    2
candidate_vars: [10]
gains:          [-2]
flip_var:       10

total_unsat:    4
candidate_vars: [18]
gains:          [-3]
flip_var:       18

total_unsat:    7
candidate_vars: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
gains:          [-4, 0, -3, 0, 0, -1, -3, 0, 1, 1, 0, -2, 0, 0, -1, 0, -1, 3, -1, -4]
flip_var:       18
…
```

This snippet shows 4 iterations. In iteration 1 a classical GSAT step was executed. All variables acted as candidates and for each a flip was simulated and the net_gain calculated. Variables 4 and 13 had the best net_gain of 0. A random tie breaker selected variable 4 to be flipped.

Next we see that the total_unsat counter remained 2 due to 0 net_gain of the flipped variable. This iteration represents a random walk iteration as the candidate list only contains a single variable.

Iteration 3 represents a random walk step as well. The net_gain of the random operation is -3. The classical GSAT step in the 4th iteration shown above finds that the best net_gain is flipping back the previously randomly selected variable 18.

2 WalkSAT/SKC

In contrast to GSAT the original WalkSAT algorithm according to SKC differs in two essential ways:

1. In each iteration an unsatisfied clause is randomly selected. Only variables included in this clause will be flip candidates for the given iteration.
2. The metric to decide which variable to flip is based on the “negative gain” counter, which represents the number of clauses that were satisfied before the intended flip and would be unsatisfied thereafter.

While the focus on a single unsatisfied clause only in any given iteration significantly reduces computing requirements as the impact of flipping less variables has to be analysed, WalkSAT/SKC might miss making the highest net_gain flip possible (as GWSAT does). With its focus on “negative gain” as variable selection metric, WalkSAT/SKC ensures making the least destructive flip in any given iteration. If one or more variables result in a flip with a zero “negative gain”, WalkSAT randomly picks one of these variables to ensure the number of unsatisfied clauses does not increase. If no such variable exists WalkSAT/SKC selects with a probability “p” a random variable from the selected unsatisfied clause, or with probability 1-p the variable that will result in the lowest “negative_gain”. Overall, this extremely careful variable selection metric helps WalkSAT/SKC to compensate for its much smaller search space per iteration. The variable selection process of WalkSAT/SKC can be described to intensifying the search space due to its focus on unsatisfied clauses only. The random operation according to parameter “p” introduce randomicity into the search process.

By leveraging a Tabu list the search process can track the history of variable flips and essentially block single variables to be flipped within in a specified number of iterations “tl”. This also helps the algorithm to escape local minima by avoiding flipping back and forth the same variable once such local minimum has been reached. Blocking recently flipped variables from being selected again within a given iteration interval also increases the diversification of the search process.

A large number of experiments will be executed and documented on the next pages to demonstrate the parameters’ impact on the algorithm’s operation. The parameters’ impact will first be measured against algorithm search performance of the uf50-01 instance. Afterwards, found hypothesis will be validated against the small uf20 instances as well.

2.1 Evaluation of “p”

Experiment	Instance	Executions	Restarts	Iterations	wp	tl
2.1.0	uf50-01	30	10	1000	0.4	5
2.1.1	uf50-01	1	10	1000	0.001	5
2.1.2	uf50-01	1	10	1000	0.05	5
2.1.3	uf50-01	1	10	1000	0.2	5
2.1.4	uf50-01	1	10	1000	0.4	5
2.1.5	uf50-01	1	10	1000	0.6	5
2.1.6	uf50-01	1	10	1000	0.8	5
2.1.7	uf50-01	30	10	1000	0.001	5
2.1.8	uf50-01	30	10	1000	0.05	5
2.1.9	uf50-01	30	10	1000	0.2	5
2.1.10	uf50-01	30	10	1000	0.6	5
2.1.11	uf50-01	30	10	1000	0.8	5

Experiment 2.1.0 represents a baseline experiment against instance uf50-01 with parameters provided in the assignment brief. From the statistics we can see that on average 300 iterations are needed in search runs that find a solution. We also see that the longest search took 1497 iterations, that is 1 restart after 1000 iterations without finding a solution plus another 497 iterations to solve the instance. It is also immediately noticeable that the focus on a single unsatisfied clause lets the algorithm select a variable in much less time than a classical GSAT step.

Experiment	Execution Overview	Restarts	Total Iterations	Iterations to solve
2.1.0	Total : 834222 µsec Mean : 27807 µsec Success rate: 100.00% Total iter: 12990 Avg iter time: 64 µsec	Sum : 4 Min : 0 Max : 1 Mean : 0.1 Median: 0.0 STDEV : 0.3	Sum : 12990 Min : 15 Max : 1497 Mean : 433 Median: 283.5 STDEV : 419.5 VarCo : 0.969	 Min : 15 Max : 853 Mean : 299.7 Median: 266.5 STDEV : 222.7 VarCo : 0.743

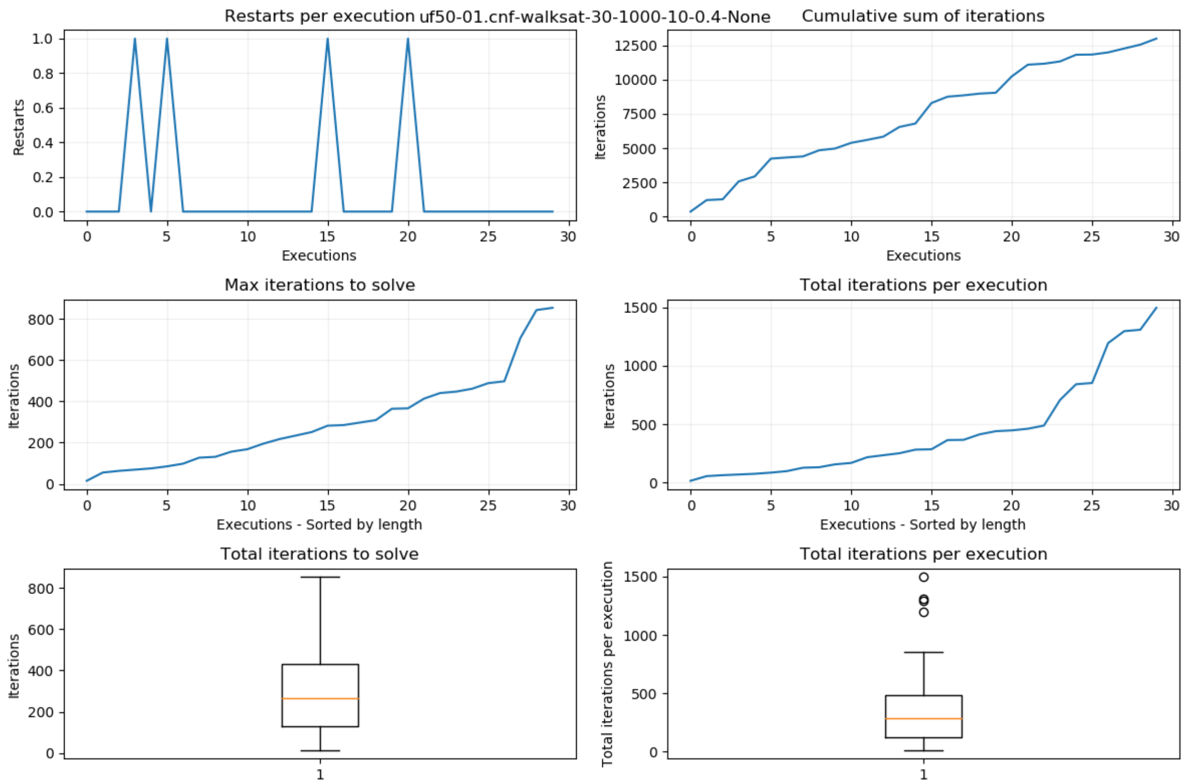
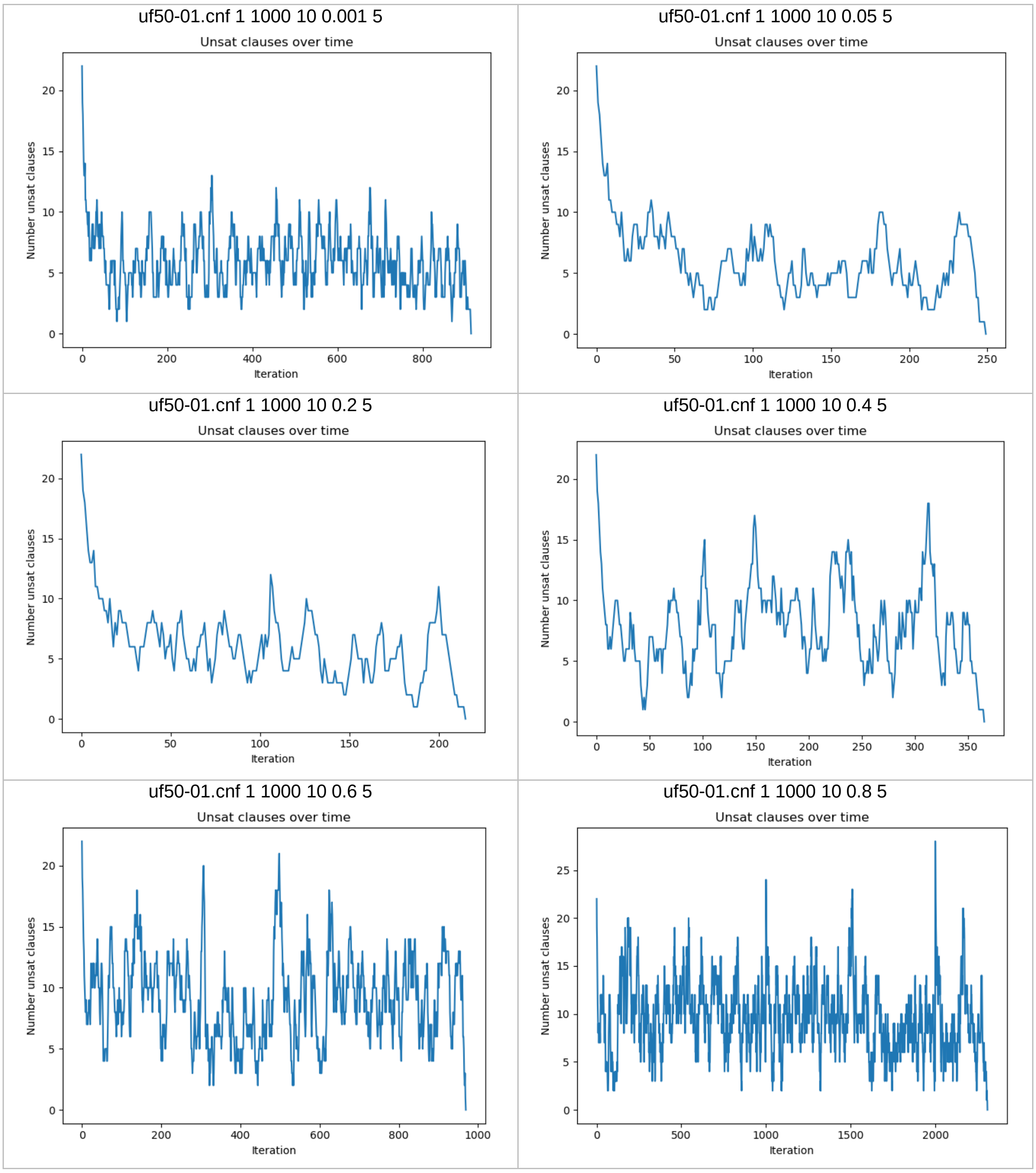


Figure 5 Restart and Iterations statistics for experiment 2.1.0

All statistics as well as the overall search time to complete 30 executions show significant improvement over the GWSAT search

As done for GWSAT, experiments 2.1.1 to 2.1.6 will be run with a single execution in order to visualize the impact of parameter “p” on the number of unsatisfied clauses during the search operation. The same values as for GWSAT “wp” will be used for WalkSAT’s “p” parameter:
[0.001, 0.05, 0.2, 0.4, 0.6, 0.8].



Considering that WalkSAT/SKC solves instance uf50-01 with “p” = 0 in less than 1000 iterations as well, the “p” setting of 0.001 is expected to have extremely limited influence only – if at all.
Despite the higher number of iterations in experiment 2.1.1 and the squeezed graph representation, the actual graph is very close to the one of experiment 2.1.2 (p=0.05). Experiments 2.1.5 and 2.1.6

demonstrate a similar influence of a very high “wp” value for GSAT. The careful selection of variables according to the minimum “negative_gain” gets mostly replaced by random steps.

For comparison lets run experiments 2.1.7 to 2.1.11 with 30 executions and see a general trend when running certain configurations for multiple executions. Please note that a 30 executions experiment with p=0.4 was already executed as part of experiment 2.1.0. The results for experiment 2.1.0 are repeated for comparability.

Experiment	Execution Overview	Restarts	Total Iterations	Iterations to solve
2.1.0	Total : 834222 µsec Mean : 27807 µsec Success rate: 100.00% Total iter: 12990 Avg iter time: 64 µsec	Sum : 4 Min : 0 Max : 1 Mean : 0.1 Median: 0.0 STDEV : 0.3	Sum : 12990 Min : 15 Max : 1497 Mean : 433 Median: 283.5 STDEV : 419.5 VarCo : 0.969	 Min : 15 Max : 853 Mean : 299.7 Median: 266.5 STDEV : 222.7 VarCo : 0.743
2.1.7	Total : 610617 µsec Mean : 20353 µsec Success rate: 100.00% Total iter: 9463 Avg iter time: 65 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 9463 Min : 30 Max : 980 Mean : 315.4 Median: 218.5 STDEV : 263.2 VarCo : 0.835	 Min : 30 Max : 980 Mean : 315.4 Median: 218.5 STDEV : 263.2 VarCo : 0.835
2.1.8	Total : 606179 µsec Mean : 20205 µsec Success rate: 100.00% Total iter: 9231 Avg iter time: 66 µsec	Sum : 1 Min : 0 Max : 1 Mean : 0.0 Median: 0.0 STDEV : 0.2	Sum : 9231 Min : 66 Max : 1128 Mean : 307.7 Median: 271.0 STDEV : 221.1 VarCo : 0.719	 Min : 66 Max : 747 Mean : 274.4 Median: 259.0 STDEV : 160.2 VarCo : 0.584
2.1.9	Total : 826685 µsec Mean : 27556 µsec Success rate: 100.00% Total iter: 12395 Avg iter time: 67 µsec	Sum : 3 Min : 0 Max : 1 Mean : 0.1 Median: 0.0 STDEV : 0.3	Sum : 12395 Min : 25 Max : 1293 Mean : 413.2 Median: 304.0 STDEV : 329.4 VarCo : 0.797	 Min : 25 Max : 806 Mean : 313.2 Median: 276.0 STDEV : 188.8 VarCo : 0.603
2.1.10	Total : 1692692 µsec Mean : 56423 µsec Success rate: 100.00% Total iter: 26330 Avg iter time: 64 µsec	Sum : 11 Min : 0 Max : 3 Mean : 0.4 Median: 0.0 STDEV : 0.7	Sum : 26330 Min : 38 Max : 3495 Mean : 877.7 Median: 738.0 STDEV : 699.1 VarCo : 0.797	 Min : 38 Max : 970 Mean : 511 Median: 543.5 STDEV : 295.3 VarCo : 0.578
2.1.11	Total : 4040370 µsec Mean : 134679 µsec Success rate: 96.67% Total iter: 52646 Avg iter time: 64 µsec	Sum : 40 Min : 0 Max : 4 Mean : 1.4 Median: 1 STDEV : 1.2	Sum : 52646 Min : 68 Max : 4852 Mean : 1815.4 Median: 1618 STDEV : 1159.4 VarCo : 0.639	 Min : 68 Max : 979 Mean : 436.1 Median: 433 STDEV : 279.3 VarCo : 0.641

Experiments 2.1.7 to 2.1.11 confirm that WalkSAT generally works best with very small “p” values for 2 reasons:

1. It is able to honour its main metric of “negative_gain” for variable selection.
2. The random walk based on “p” does not provide any noticeable iteration runtime improvements as in any case only the variables of a single unsatisfied clause are considered as flip candidate in any given iteration. In addition, this list of potential flip candidates may be further reduced if one or more variables are tabued in a given iteration. Search runtime per iteration is constant.

Experiments 2.1.9 to 2.1.11 (and 2.1.0) illustrate that the increase in total number of iterations immediately translates into longer search runtime (for the reason given above).

Figure 5 visualizes the key performance indicators discussed above.

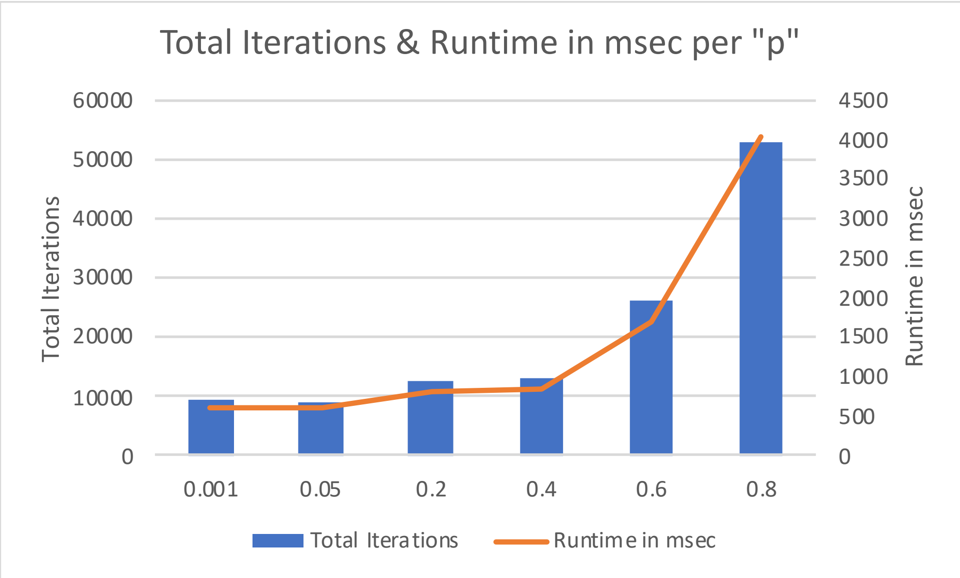


Figure 6 Total Iterations and Runtime in sec per wp parameter

2.2 Evaluation of "restart" and "max_iterations"

The trade-off between the parameters "restart" and "max_iterations" have been discussed in section 1.2 already. The same concepts apply to WalkSAT:

1. A decrease in "max_iterations" will result in more restarts (given the parameter is reduced to less than the longest run seen for a given instance).
2. Each restart has a processing cost in creating a new solution assignment and restart the search.

A few selected experiments will be executed for documenting the impact of changing the "max_iterations" parameter for the current problem instance uf50-01.

Experiment	Instance	Executions	Restarts	Iterations	wp	tl
2.2.1	uf50-01	30	10	400	0.001	5
2.2.2	uf50-01	30	10	600	0.001	5
2.2.3	uf50-01	30	10	800	0.001	5
2.2.4	uf50-01	30	10	400	0.05	5
2.2.5	uf50-01	30	10	600	0.05	5
2.2.6	uf50-01	30	10	800	0.05	5

Experiment	Execution Overview	Restarts	Total Iterations	Iterations to solve
2.2.1	Total : 652134 µsec Mean : 21737 µsec Success rate: 100.00% Total iter: 10058 Avg iter time: 65 µsec	Sum : 12 Min : 0 Max : 3 Mean : 0.4 Median: 0.0 STDEV : 0.7	Sum : 10058 Min : 34 Max : 1327 Mean : 335.3 Median: 194.0 STDEV : 310.6 VarCo : 0.926	 Min : 34 Max : 357 Mean : 175.3 Median: 153.0 STDEV : 95.3 VarCo : 0.544
2.2.2	Total : 440489 µsec Mean : 14682 µsec Success rate: 100.00% Total iter: 6747 Avg iter time: 65 µsec	Sum : 2 Min : 0 Max : 1 Mean : 0.1 Median: 0.0 STDEV : 0.3	Sum : 6747 Min : 28 Max : 882 Mean : 224.9 Median: 176.0 STDEV : 185.3 VarCo : 0.824	 Min : 26 Max : 449 Mean : 184.9 Median: 170.5 STDEV : 117.1 VarCo : 0.633
2.2.3	Total : 560000 µsec Mean : 18666 µsec Success rate: 100.00% Total iter: 8386 Avg iter time: 67 µsec	Sum : 2 Min : 0 Max : 1 Mean : 0.1 Median: 0.0 STDEV : 0.3	Sum : 8386 Min : 22 Max : 944 Mean : 279.5 Median: 192.5 STDEV : 246.6 VarCo : 0.882	 Min : 22 Max : 671 Mean : 226.2 Median: 171.0 STDEV : 184.4 VarCo : 0.815
2.2.4	Total : 664938 µsec Mean : 22164 µsec Success rate: 100.00% Total iter: 10246 Avg iter time: 65 µsec	Sum : 12 Min : 0 Max : 3 Mean : 0.4 Median: 0.0 STDEV : 0.7	Sum : 10246 Min : 44 Max : 1418 Mean : 341.5 Median: 263.5 STDEV : 296.6 VarCo : 0.868	 Min : 23 Max : 375 Mean : 181.5 Median: 172.5 STDEV : 111.6 VarCo : 0.615
2.2.5	Total : 605944 µsec Mean : 20198 µsec Success rate: 100.00% Total iter: 9311 Avg iter time: 65 µsec	Sum : 3 Min : 0 Max : 1 Mean : 0.1 Median: 0.0 STDEV : 0.3	Sum : 9311 Min : 49 Max : 779 Mean : 310.4 Median: 260.0 STDEV : 191.6 VarCo : 0.617	 Min : 49 Max : 574 Mean : 250.4 Median: 246.0 STDEV : 134.1 VarCo : 0.536
2.2.6	Total : 579269 µsec Mean : 19308 µsec Success rate: 100.00% Total iter: 8830 Avg iter time: 66 µsec	Sum : 3 Min : 0 Max : 2 Mean : 0.1 Median: 0.0 STDEV : 0.4	Sum : 8830 Min : 30 Max : 1839 Mean : 294.3 Median: 236.5 STDEV : 342.2 VarCo : 1.163	 Min : 30 Max : 447 Mean : 214.3 Median: 230.5 STDEV : 120.1 VarCo : 0.56

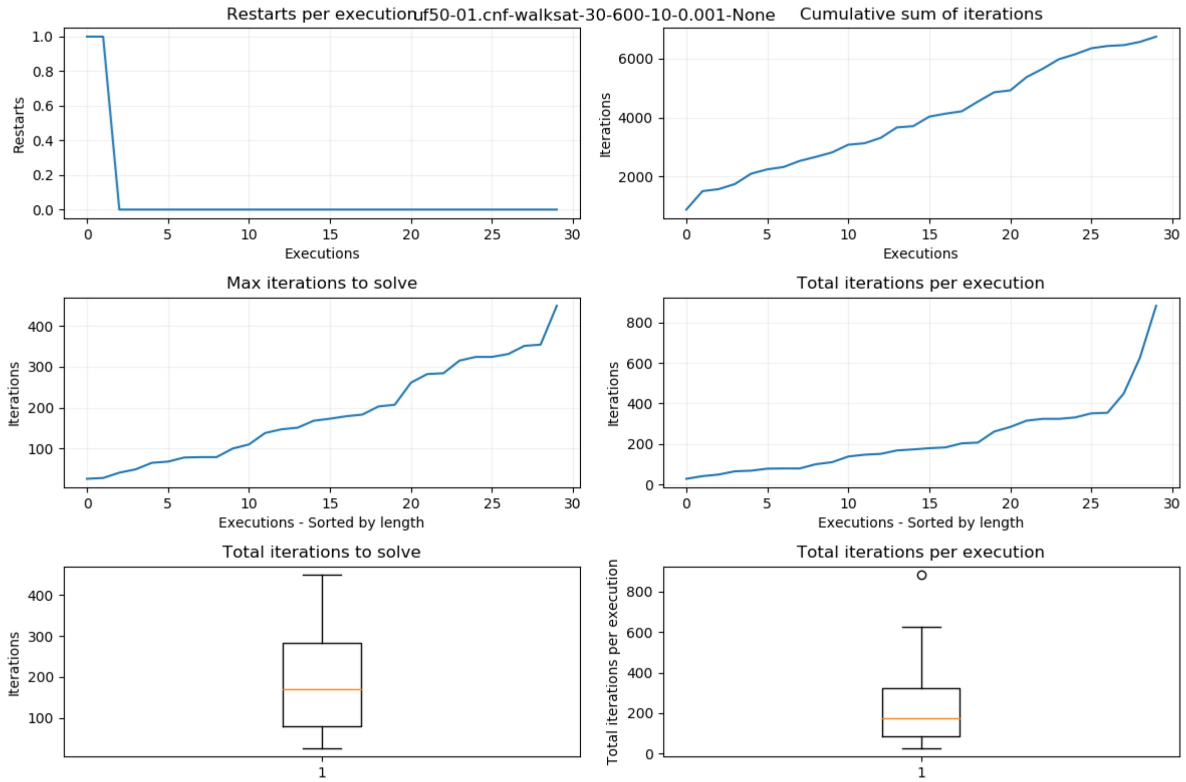


Figure 7 Restart and Iterations statistics for experiment 2.2.2

It seems that choosing a very small “p” value and reducing “max_iterations” to 600 worked extremely good in solving 30 searches for instance uf50-01.

2.3 Evaluation of “tl”

The parameter “tl” specifies the number of iterations a flipped variable is tabued until it can be flipped again. When used in conjunction with WalkSAT, the Tabu list effectively avoids that the algorithm is flipping the same variable back and forth once a local minimum has been reached and therefore ensures the search space’s diversification.

The parameter “tl” should be coupled to the number of variables in a given problem instance. Selecting “tl” too big, will result in too many variables being blocked and therefore limiting search progress. Selecting “tl” too small may lead the search into the same local minimum again.

For problem instance uf50-01 the “tl” setting of 5 achieved the best results. This represents 10% of the variables.

2.4 Evaluating instances uf20-01 and uf20-02

Lastly, experiments 2.4.1 to 2.4.8 measure the performance and conclusions from above against instances uf20-01 and uf20-02. The “restarts” and “max_iterations” parameters will be safely reduced to 3 and 200 respectively. “p” is set set 0.001. “tl” will be evaluated against a short set of values, mainly to see if the intuition of 10% variables blocked by the tabu list holds true. Experiments 2.4.1 and 2.4.5 represent baseline runs according to the assignment brief.

Experiment	Instance	Executions	Restarts	Iterations	wp	tl
2.4.1	uf20-01	30	10	1000	0.4	5
2.4.2	uf20-01	30	3	200	0.001	2
2.4.3	uf20-01	30	3	200	0.05	2
2.4.4	uf20-01	30	3	200	0.001	3
2.4.5	uf20-01	30	3	200	0.05	3
2.4.6	uf20-01	30	3	200	0.001	5
2.4.7	uf20-01	30	3	200	0.05	5

Experiment	Execution Overview	Restarts	Total Iterations	Iterations to solve
2.4.1	Total : 55916 µsec Mean : 1863 µsec Success rate: 100.00% Total iter: 1086 Avg iter time: 51 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 1086 Min : 7 Max : 82 Mean : 36.2 Median: 27.0 STDEV : 23.2 VarCo : 0.64	Min : 7 Max : 82 Mean : 36.2 Median: 27.0 STDEV : 23.2 VarCo : 0.64
2.4.2	Total : 54793 µsec Mean : 1826 µsec Success rate: 100.00% Total iter: 980 Avg iter time: 56 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 980 Min : 4 Max : 121 Mean : 32.7 Median: 19.5 STDEV : 28.7 VarCo : 0.878	Min : 4 Max : 121 Mean : 32.7 Median: 19.5 STDEV : 28.7 VarCo : 0.878
2.4.3	Total : 42507 µsec Mean : 1416 µsec Success rate: 100.00% Total iter: 689 Avg iter time: 62 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 689 Min : 3 Max : 91 Mean : 23.0 Median: 17.0 STDEV : 22.2 VarCo : 0.967	Min : 3 Max : 91 Mean : 23.0 Median: 17.0 STDEV : 22.2 VarCo : 0.967
2.4.4	Total : 41519 µsec Mean : 1383 µsec Success rate: 100.00% Total iter: 761 Avg iter time: 55 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 761 Min : 3 Max : 85 Mean : 25.4 Median: 17.5 STDEV : 22.9 VarCo : 0.902	Min : 3 Max : 85 Mean : 25.4 Median: 17.5 STDEV : 22.9 VarCo : 0.902
2.4.5	Total : 58666 µsec Mean : 1955 µsec Success rate: 100.00% Total iter: 1139 Avg iter time: 52 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 1139 Min : 8 Max : 124 Mean : 38.0 Median: 27.5 STDEV : 30.1 VarCo : 0.792	Min : 8 Max : 124 Mean : 38.0 Median: 27.5 STDEV : 30.1 VarCo : 0.792

2.4.6	Total : 45661 µsec Mean : 1522 µsec Success rate: 100.00% Total iter: 911 Avg iter time: 50 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 911 Min : 8 Max : 195 Mean : 30.4 Median: 16.5 STDEV : 36.8 VarCo : 1.212	 Min : 8 Max : 195 Mean : 30.4 Median: 16.5 STDEV : 36.8 VarCo : 1.212
2.4.7	Total : 55869 µsec Mean : 1862 µsec Success rate: 100.00% Total iter: 1139 Avg iter time: 49 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 1139 Min : 6 Max : 195 Mean : 38.0 Median: 26.0 STDEV : 36.5 VarCo : 0.961	 Min : 6 Max : 195 Mean : 38.0 Median: 26.0 STDEV : 36.5 VarCo : 0.961

The 3 most promising configurations will now be run with 1000 executions.

Experiment	Instance	Executions	Restarts	Iterations	wp	tl
2.4.8	uf20-01	1000	3	200	0.05	2
2.4.9	uf20-01	1000	3	200	0.001	3
2.4.10	uf20-01	1000	3	200	0.001	5

Experiment	**Execution Overview**	**Restarts**	**Total Iterations**	**Iterations to solve**
2.4.8	Total : 1728190 µsec Mean : 1728 µsec Success rate: 100.00% Total iter: 31156 Avg iter time: 55 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 31156 Min : 1 Max : 247 Mean : 31.2 Median: 23.0 STDEV : 26.9 VarCo : 0.862	 Min : 1 Max : 189 Mean : 30.8 Median: 23.0 STDEV : 25.4 VarCo : 0.825
2.4.9	Total : 1683118 µsec Mean : 1683 µsec Success rate: 100.00% Total iter: 30983 Avg iter time: 54 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 30983 Min : 2 Max : 176 Mean : 31.0 Median: 24.0 STDEV : 25.9 VarCo : 0.837	 Min : 2 Max : 176 Mean : 31.0 Median: 24.0 STDEV : 25.9 VarCo : 0.837
2.4.10	Total : 1674545 µsec Mean : 1674 µsec Success rate: 100.00% Total iter: 34838 Avg iter time: 48 µsec	Sum : 2 Min : 0 Max : 1 Mean : 0.0 Median: 0.0 STDEV : 0.0	Sum : 34838 Min : 3 Max : 223 Mean : 34.8 Median: 26.0 STDEV : 30.8 VarCo : 0.884	 Min : 3 Max : 195 Mean : 34.4 Median: 26.0 STDEV : 29.7 VarCo : 0.863

The above experiments confirm that also for very small instances like uf20-* a very low “p” value, e.g. 0.001 or 0.05, provides best results. The experiments 2.4.8 to 2.4.10 confirm that “tl” values of 2 or 3 (10%, or 15%) result in the least total iterations.

Experiment	Instance	Executions	Restarts	Iterations	wp	tl
2.4.11	uf20-02	30	10	1000	0.4	5
2.4.12	uf20-02	30	3	200	0.001	2
2.4.13	uf20-02	30	3	200	0.05	2
2.4.14	uf20-02	30	3	200	0.001	3
2.4.15	uf20-02	30	3	200	0.05	3
2.4.16	uf20-02	30	3	200	0.001	5
2.4.17	uf20-02	30	3	200	0.05	5
2.4.18	uf20-02	1000	3	200	0.001	2
2.4.19	uf20-02	1000	3	200	0.001	5

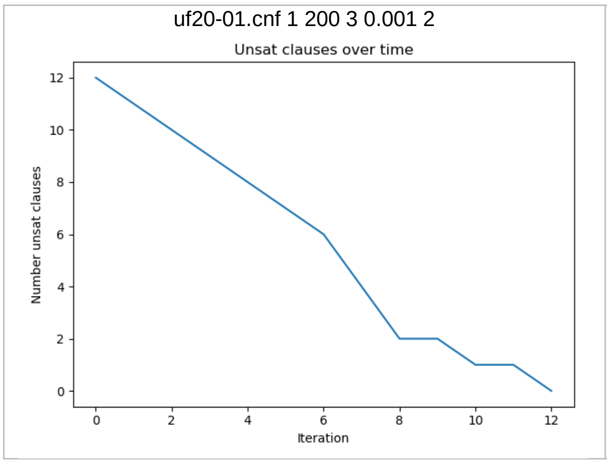
Experiment	**Execution Overview Overview**	**Restarts**	**Total Iterations**	**Iterations to solve**
2.4.11	Total : 50844 µsec Mean : 1694 µsec Success rate: 100.00% Total iter: 989 Avg iter time: 51 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 989 Min : 3 Max : 129 Mean : 33.0 Median: 30.5 STDEV : 24.9 VarCo : 0.755	Min : 3 Max : 129 Mean : 33.0 Median: 30.5 STDEV : 24.9 VarCo : 0.755
2.4.12	Total : 30523 µsec Mean : 1017 µsec Success rate: 100.00% Total iter: 455 Avg iter time: 67 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 455 Min : 5 Max : 34 Mean : 15.2 Median: 15.0 STDEV : 5.8 VarCo : 0.385	Min : 5 Max : 34 Mean : 15.2 Median: 15.0 STDEV : 5.8 VarCo : 0.385
2.4.13	Total : 30998 µsec Mean : 1033 µsec Success rate: 100.00% Total iter: 465 Avg iter time: 67 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 465 Min : 1 Max : 46 Mean : 15.5 Median: 13.5 STDEV : 9.1 VarCo : 0.587	Min : 1 Max : 46 Mean : 15.5 Median: 13.5 STDEV : 9.1 VarCo : 0.587
2.4.14	Total : 32556 µsec Mean : 1085 µsec Success rate: 100.00% Total iter: 487 Avg iter time: 67 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 487 Min : 3 Max : 34 Mean : 16.2 Median: 14.5 STDEV : 9.0 VarCo : 0.556	Min : 3 Max : 34 Mean : 16.2 Median: 14.5 STDEV : 9.0 VarCo : 0.556
2.4.15	Total : 32456 µsec Mean : 1081 µsec Success rate: 100.00% Total iter: 540 Avg iter time: 60 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 540 Min : 5 Max : 44 Mean : 18 Median: 16.5 STDEV : 10.5 VarCo : 0.584	Min : 5 Max : 44 Mean : 18 Median: 16.5 STDEV : 10.5 VarCo : 0.584
2.4.16	Total : 27950 µsec Mean : 931 µsec Success rate: 100.00% Total iter: 451 Avg iter time: 62 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 451 Min : 1 Max : 49 Mean : 15.0 Median: 12.5 STDEV : 10.3 VarCo : 0.685	Min : 1 Max : 49 Mean : 15.0 Median: 12.5 STDEV : 10.3 VarCo : 0.685
2.4.17	Total : 38057 µsec Mean : 1268 µsec Success rate: 100.00% Total iter: 710 Avg iter time: 54 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 710 Min : 2 Max : 71 Mean : 23.7 Median: 18.0 STDEV : 17.2 VarCo : 0.726	Min : 2 Max : 71 Mean : 23.7 Median: 18.0 STDEV : 17.2 VarCo : 0.726

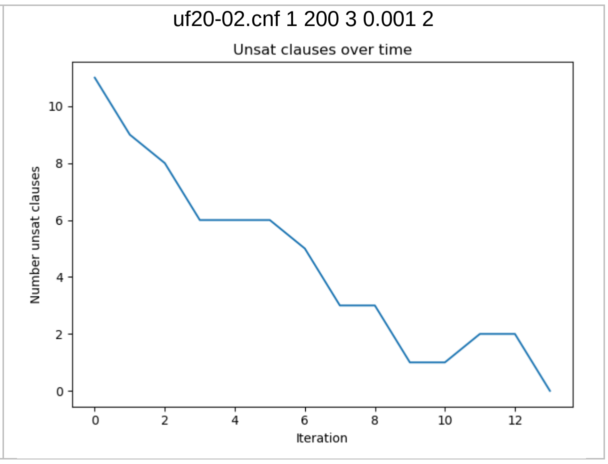
Again, the configurations with very small “p” values complete 30 search executions faster than the baseline experiment. However, most of the search runtimes are so close to each other that 1000 search executions are run against the best 2 experiments (measured by total iterations) 2.4.12 and 2.4.16.

Experiment	**Execution Overview Overview**	**Restarts**	**Total Iterations**	**Iterations to solve**
2.4.18	Total : 970976 µsec Mean : 970 µsec Success rate: 100.00% Total iter: 14802 Avg iter time: 66 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 14802 Min : 1 Max : 61 Mean : 14.8 Median: 13.0 STDEV : 8.6 VarCo : 0.578	Min : 1 Max : 61 Mean : 14.8 Median: 13.0 STDEV : 8.6 VarCo : 0.578
2.4.18	Total : 1045715 µsec Mean : 1045 µsec Success rate: 100.00% Total iter: 18919 Avg iter time: 55 µsec	Sum : 0 Min : 0 Max : 0 Mean : 0 Median: 0.0 STDEV : 0.0	Sum : 18919 Min : 1 Max : 100 Mean : 18.9 Median: 14.0 STDEV : 15.2 VarCo : 0.803	Min : 1 Max : 100 Mean : 18.9 Median: 14.0 STDEV : 15.2 VarCo : 0.803

Like seen with experiments against uf20-01, a very small “p” value (here 0.001) and a tabu of 10% the number of variables in the problem (here 2) produces the most performant search.

Finally, let us visualize the history for the “unsatisfied clauses” counter for both uf20 instances. Both graphs visualize WalkSAT’s careful variable selection that prioritises variables with the lowest “negative gain”.





2.5 Conclusion

The previous experiments have given insights into how the operation of WalkSAT/SKC is affected by its parameters “wp” and “tl”. The parameters “restart” and “max_iterations” have been evaluated in section 1 already and the concept remains untouched.

WalkSAT completed the search operations significantly faster than GWSAT so that “restart” and “max_iterations” can be throttled much stronger than with GWSAT depending on the problem instance.

The experiments have shown that WalkSAT favours very small “p” values of 0.001 or 0.05 and focus on its lowest “negative gain” metric otherwise. Too large “p” values render the WalkSAT search more or less into a random search risking getting into states with more unsatisfied clauses.

Empiric results indicate that a good value for the tabu list length (or better how many iterations a flipped variable should be blocked from being selected again) is around 10%-15% of the number of variables in a given problem. This indication of course would need to be further analysed against additional and more complex problems.

While the general aim of WalkSAT is the intensification of the search by limiting the search space to unsatisfied clauses only and preferring lower negative gains during the variable selection, the introduction of the Tabu list provides diversification of the search process.

WalkSAT’s small neighbourhood for every iteration (1 unsatisfied clause) and the resulting quasi-constant time required per iteration makes it very scalable to larger problem instances.

2.6 Debug WalkSAT

To debug the operation of the GWSAT implementation set the DEBUG flag to TRUE in file 'Leske_183658_WalkSAT.py'.

This out is based on a baseline run against uf50-01.
`python Leske_183658_WalkSAT.py uf50-01.cnf 1 1000 10 0.4 5`

Variable selection based on negative gain 0:

```
…
Iteration                 : 1
Tabu List                 : [None, None, None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
None]
candidate_vars before tabu: [41, 34, 20]
candidate_vars after  tabu: [41, 34, 20]
candidate_vars neg_gain   : [2, 0, 1]
selected variable         : 34
…
```

This output shows that for 3 candidate variables the negative gain has been calculated. Variable 34 has a negative gain of zero and is selected.

Variable selection based on lowest negative gain:

```
…
Iteration                 : 4
Tabu List                 : [8, None, None, None, None, None, None, None, 7, None, None, None, None, None,
None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None,
None, 6, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None]
candidate_vars before tabu: [25, 50, 18]
candidate_vars after  tabu: [25, 50, 18]
random walk
candidate_vars neg_gain   : [2, 2, 1]
selected variable         : 18
…
```

This output shows that none of the candidate variables has a negative gain of zero. Hence, the variable with the lowest negative gain is selected.

Variable selection based on random walk with probability "p":

```
…
Iteration                 : 14
Tabu List                 : [8, None, 14, 11, None, None, 16, None, 7, 15, None, None, 17, None, None,
None, None, 9, None, None, None, None, None, None, None, None, None, 13, None, None, None, 12, None, 6,
None, None, None, 10, None, None, None, None, None, None, None, None, None, None, 18, None]
candidate_vars before tabu: [32, 26, 6]
candidate_vars after  tabu: [32, 26, 6]
random walk
candidate_vars neg_gain   : [3, 2, 2]
selected variable         : 6
…
```

This output shows how parameter "p" fired and a random variable was selected. A side effect of the limited number of candidate variables with WalkSAT is the still the random walk step can be an optimal step for the algorithm. This observation is enforced, when 1 or 2 variables are tabued in a given iteration.

Maintaining tabu list:

```
…
Iteration                 : 14
Tabu List                 : [8, None, 14, 11, None, None, 16, None, 7, 15, None, None, 17, None, None,
None, None, 9, None, None, None, None, None, None, None, None, None, 13, None, None, None, 12, None, 6,
None, None, None, 10, None, None, None, None, None, None, None, None, None, None, 18, None]
candidate_vars before tabu: [32, 26, 6]
candidate_vars after  tabu: [32, 26, 6]
random walk
candidate_vars neg_gain   : [3, 2, 2]
selected variable         : 6

Iteration                 : 15
Tabu List                 : [8, None, 14, 11, None, 19, 16, None, 7, 15, None, None, 17, None, None, None,
None, 9, None, None, None, None, None, None, None, None, None, 13, None, None, None, 12, None, 6, None,
None, None, 10, None, None, None, None, None, None, None, None, None, None, 18, None]
candidate_vars before tabu: [16, 38, 6]
candidate_vars after  tabu: [16, 38]
candidate_vars neg_gain   : [0, 1]
selected variable         : 16
…
```

This output shows that is iteration 14 variable 6 was selected. The tabu list for iteration 17 shows that variables 6 is blocked until and including iteration 19. Starting iteration 20 variable 6 can be selected again.

Filtering candidates based on tabu list:

```
…
Iteration                 : 258
Tabu List                 : [223, 236, 217, 151, None, 230, 250, 247, 238, 262, 175, 243, 84, 227, 258,
159, 213, 176, 237, 259, 251, 210, 260, 222, 209, 189, 249, 171, 164, 242, 221, 196, 261, 123, 193, 245,
248, 231, None, 197, 244, 254, 143, 239, 257, 252, 195, 246, 103, 256]
candidate_vars before tabu: [33, 4, 12]
candidate_vars after  tabu: [4, 12]
candidate_vars neg_gain   : [5, 3]
selected variable         : 12

Iteration                 : 259
Tabu List                 : [223, 236, 217, 151, None, 230, 250, 247, 238, 262, 175, 263, 84, 227, 258,
159, 213, 176, 237, 259, 251, 210, 260, 222, 209, 189, 249, 171, 164, 242, 221, 196, 261, 123, 193, 245,
248, 231, None, 197, 244, 254, 143, 239, 257, 252, 195, 246, 103, 256]
candidate_vars before tabu: [24, 50, 5]
candidate_vars after  tabu: [24, 50, 5]
candidate_vars neg_gain   : [0, 2, 3]
selected variable         : 24
…
Iteration                 : 263
Tabu List                 : [223, 236, 217, 151, None, 230, 250, 247, 238, 262, 175, 263, 84, 227, 258,
159, 213, 176, 237, 266, 251, 210, 260, 264, 209, 189, 249, 171, 164, 242, 221, 196, 261, 123, 193, 245,
248, 231, None, 197, 244, 254, 143, 239, 257, 267, 195, 246, 265, 256]
candidate_vars before tabu: [32, 12, 46]
candidate_vars after  tabu: [32]
candidate_vars neg_gain   : [1]
selected variable         : 32
…
```

From this snippet we can see that variable 12 was selected in iteration 258. Therefore the debug for iteration 259 shows that variable needs to idle for 5 subsequent iterations (259, 260, 261, 262, 263) and can only be used again thereafter.
The debug for iteration 263 indeed confirms that variable 12 is still blocked by the tabu list and removed from the candidate list

3 Run-Time Distribution Evaluation

Performing Local Search for a SAT problem represents a decision problem in which a candidate variable assignment is determined to satisfy the given SAT instance or not. Depending on the randomness involved in creating the initial variable assignment and thereafter the decisions between "normal" and "random" search steps, the runtime for such local search algorithms experiences variance. Hence, the Local Search for SAT problems is defined by the Las Vegas class of algorithms where "runtime" is treated as the random variable that requires quantification.

The statistics collected in sections 1 and 2 already outline a few key findings:

- WalkSAT requires far less iterations and runtime to solve the uf20 or uf50 instances compared to GWSAT.
- Due to the limited search space per iteration WalkSAT offers a quasi-constant runtime per iteration.
- GWSAT runtime per iteration can be reduced by increasing the "wp" parameter at the cost of a more and more random search.
- Instance uf20-02 can be solved much faster both in runtime and iterations than uf20-01.

In order to evaluate the run-time distribution each problem instance uf20-01 and uf20-02 will be executed 1000 times against GWSAT and WalkSAT. Both instances will be run against each algorithm with the same parameters to allow comparison. The following parameters will be used for the run-time distribution evaluation:

Experiment	Algorithm	Instance	Executions	Restarts	Iterations	wp	tl
3.1.1	GWSAT	uf20-01	1000	3	200	0.4	n/a
3.1.2	GWSAT	uf20-02	1000	3	200	0.4	n/a
3.1.3	WalkSAT	uf20-01	1000	3	200	0.001	3
3.1.4	WalkSAT	uf20-02	1000	3	200	0.001	3

For each algorithm these parameters have proven to be good candidates for the complexity of the uf20 SAT instances.

	GWSAT uf20-01	GWSAT uf20-02	WalkSAT uf20-01	WalkSAT uf20-02
Overview	Total : 10055998 µsec Mean : 10055 µsec Success rate: 100.00% Total iter: 54989 Avg iter time: 183 µsec	Total : 5580940 µsec Mean : 5580 µsec Success rate: 100.00% Total iter: 29299 Avg iter time: 190 µsec	Total : 1610836 µsec Mean : 1610 µsec Success rate: 100.00% Total iter: 30983 Avg iter time: 52 µsec	Total : 979001 µsec Mean : 979 µsec Success rate: 100.00% Total iter: 15780 Avg iter time: 62 µsec
Runtime in µsec	Sum : 10273975 Min : 825 Max : 63683 Mean : 10274 Median: 7416.5 STDEV : 9170 VarCo : 0.893 Q0.1 : 2340 Q0.25 : 3972 Q0.5 : 7416 Q0.75 : 13175 Q0.9 : 22857	Sum : 5580940 Min : 684 Max : 34548 Mean : 5581 Median: 4573.5 STDEV : 3992 VarCo : 0.715 Q0.1 : 1793 Q0.25 : 2727 Q0.5 : 4573 Q0.75 : 7157 Q0.9 : 10741	Sum : 1610836 Min : 346 Max : 7510 Mean : 1611 Median: 1303.0 STDEV : 1093 VarCo : 0.679 Q0.1 : 623 Q0.25 : 810 Q0.5 : 1303 Q0.75 : 2043 Q0.9 : 3109	Sum : 979001 Min : 363 Max : 3243 Mean : 979 Median: 863.0 STDEV : 430 VarCo : 0.44 Q0.1 : 560 Q0.25 : 681 Q0.5 : 863 Q0.75 : 1159 Q0.9 : 1531
Total Iterations	Sum : 54989 Min : 2 Max : 379 Mean : 55.0 Median: 39.0 STDEV : 49.9 VarCo : 0.908 Q0.1 : 11 Q0.25 : 21 Q0.5 : 39 Q0.75 : 71 Q0.9 : 120	Sum : 29299 Min : 2 Max : 181 Mean : 29.3 Median: 23.5 STDEV : 22.3 VarCo : 0.762 Q0.1 : 8 Q0.25 : 13 Q0.5 : 23 Q0.75 : 37 Q0.9 : 58	Sum : 30983 Min : 2 Max : 176 Mean : 31.0 Median: 24.0 STDEV : 25.9 VarCo : 0.837 Q0.1 : 8 Q0.25 : 12 Q0.5 : 24 Q0.75 : 40 Q0.9 : 66	Sum : 15780 Min : 2 Max : 67 Mean : 15.8 Median: 13.0 STDEV : 10.0 VarCo : 0.631 Q0.1 : 6 Q0.25 : 9 Q0.5 : 13 Q0.75 : 20 Q0.9 : 29

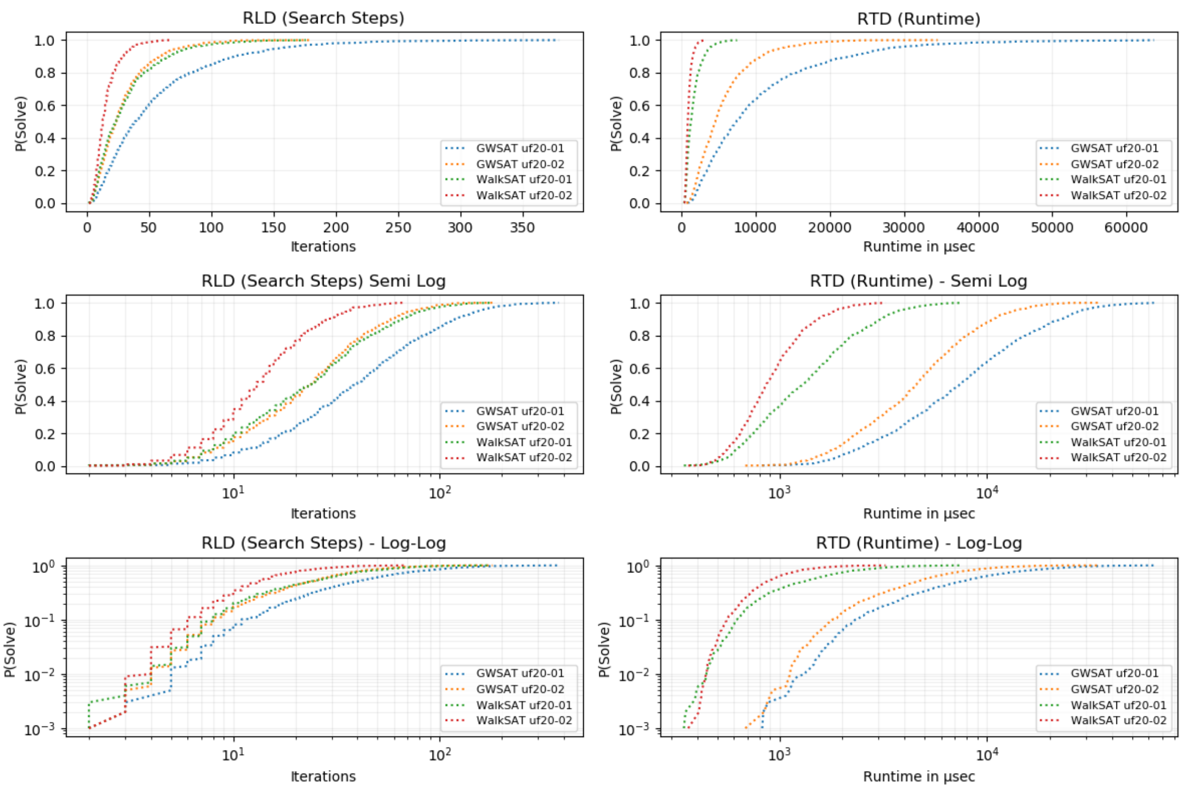


Figure 8 Run-Time Distribution for uf20-01 and uf20-02 against GWSAT and WalkSAT

Figure 8 visualizes many of the statistics given in the table above. Especially the run-time distribution graph clearly demonstrates that:

- Problem instances uf20-01 and uf20-02 are of different complexity levels
- WalkSAT solves both instances 6x to 7x faster than GWSAT

While WalkSAT solved ~75% uf20-01 and ~95% uf20-02 instances in less then 2000 µsec, GWSAT reached those success rate levels only far beyond the 10000 µsec runtime.

The Run-Length Distribution based on the Search Steps (or Iterations) to complete all executions highlights a very interesting detail. The number of search iterations GWSAT needs for solving instance uf20-02 matches the number of search iterations WalkSAT needs for solving instance uf20-01.

The statistics collected also reveal that each algorithm is able to solve 1000 executions of instance uf20-02 in ~50% of the search iterations required for solving 1000 executions of instance uf20-01. Similarly, instance uf20-01 can be solved 1000 times by WalkSAT in 50% the time GWSAT requires for the same task. The same statement can be made about instance uf20-02.

These findings again underline that the complexity for solving SAT instances of a given size can vary a lot – as do the 2 instances to be analysed.

The runtime statistics also show that a single WalkSAT iteration (52 & 62 µsec) is approximately 3-4 times faster than a GWSAT iteration (183 & 190 µsec). This can be explained with the reduced neighbourhood WalkSAT operates on (1 unsatisfied clause). As explained already in section 1, the GWSAT iterations runtime is largely affected by the random walk probability “wp”, which helps GWSAT avoiding analysing the outcome of flipping every variable by just randomly picking a variable to flip.

The cost for initially setting up the problem including reading the problem instance and filling data structures to hold the instance information (variables, literals, clauses) was measured to be ~320 µsec.

```
search = LocalSearch(cnf_file, restarts, max_iterations, alg, p=p, tl=tl, DEBUG=DEBUG)
```

The start of a search execution – be it initial start or a restart – involves generating a random variable assignment and counting the number of unsatisfied clauses. This was measured to take ~170 µsec for GWSAT.

```
self.cnf.initial_solution()
```

For WalkSAT we need to also account for the time to create and initialize the Tabu list. Hence the cost of starting or restarting a search is slightly higher at ~174 µsec.

```
self.cnf.initial_solution()

if self.alg == 'walksat':
    tabu = [ None for x in self.cnf.variables ]
```

Note: All initialization costs (in µsec) given above are only relevant for the uf20 instances to be analysed in the assignment. Larger problem instances will result in appropriately larger setup costs.

One can conclude that we the given experiment setup the cost of a restart is approximately equal to:

- 1 GWSAT iteration
- 3 WalkSAT iterations

Of course, one also needs to add to the restart cost the time penalty required for the iterations to work down the number of unsatisfied clauses. Hence, the restart cost should be accounted much higher than the pure time penalty to create a new random variable assignment and counting the number of unsatisfied clauses initially. This explains why instances uf20-01 and uf20-02 are solved the fastest when avoiding search restarts (or keeping them to a bare minimum as in GWSAT for uf20-01).

The following table is a subset of information given from the above run-time distribution experiments and repeated here to provide context for the following calculations.

	GWSAT uf20-01	**GWSAT uf20-02**	**WalkSAT uf20-01**	**WalkSAT uf20-02**
Overview	Total : 10055998 µsec Mean : 10055 µsec Success rate: 100.00% Total iter: 54989 Avg iter time: 183 µsec	Total : 5580940 µsec Mean : 5580 µsec Success rate: 100.00% Total iter: 29299 Avg iter time: 190 µsec	Total : 1610836 µsec Mean : 1610 µsec Success rate: 100.00% Total iter: 30983 Avg iter time: 52 µsec	Total : 979001 µsec Mean : 979 µsec Success rate: 100.00% Total iter: 15780 Avg iter time: 62 µsec
Runtime in µsec	Q0.75 : 13175 Q0.9 : 22857	Q0.75 : 7157 Q0.9 : 10741	Q0.75 : 2043 Q0.9 : 3109	Q0.75 : 1159 Q0.9 : 1531
Total Iterations	Max : 379 Q0.75 : 71 Q0.9 : 120	Max : 181 Q0.75 : 37 Q0.9 : 58	Max : 176 Q0.75 : 40 Q0.9 : 66	Max : 67 Q0.75 : 20 Q0.9 : 29

With the average time required per iteration and the setup costs mentioned above we can now define upper time limits to solve e.g. 75% or 90% of the uf20 problem instances. Accounting for the restart cost is only required for the max iterations example for GWSAT uf20-01. All other experiments solved all executions without restart.

GWSAT:

uf20-01

75%: 320 µsec + 71*183 µsec = 13313 µsec or 13.3 msec
90%: 320 µsec + 120*183 µsec = 22280 µsec or 22.3 msec
Max: 320 µsec + 379*183 µsec + 170 µsec = 69847 µsec or 69.8 msec

uf20-02

75%: 320 µsec + 37*190 µsec = 7350 µsec or 7.4 msec
90%: 320 µsec + 58*190 µsec = 11340 µsec or 11.3 msec
Max: 320 µsec + 67*190 µsec = 13050 µsec or 13.1 msec

WalkSAT:

uf20-01

75%: 320 µsec + 40*52 µsec = 2400 µsec or 2.4 msec
90%: 320 µsec + 66*52 µsec = 3752 µsec or 3.8 msec
Max: 320 µsec + 176*52 µsec = 9472 µsec or 9.5 msec

uf20-02

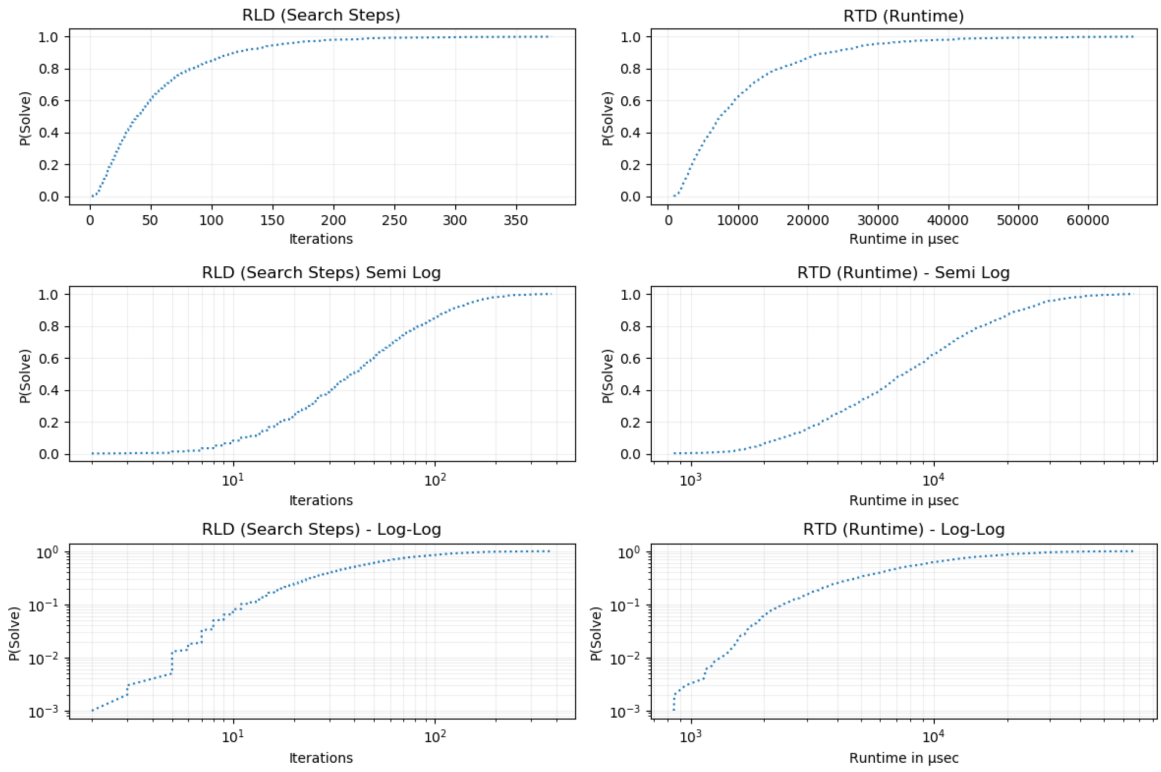
75%: 320 µsec + 20*62 µsec = 1560 µsec or 1.6 msec
90%: 320 µsec + 29*62 µsec = 2118 µsec or 2.1 msec
Max: 320 µsec + 66*62 µsec = 4412 µsec or 4.4 msec

The calculated runtime based on setup time plus runtime per iteration matches roughly the runtime statistics taken for the experiments (which do not contain the setup time). Especially for instance uf20-01 it is obvious that 90% of the executions a solution is found in ~1/3 of the time required to find a solution in 100% of the executions.

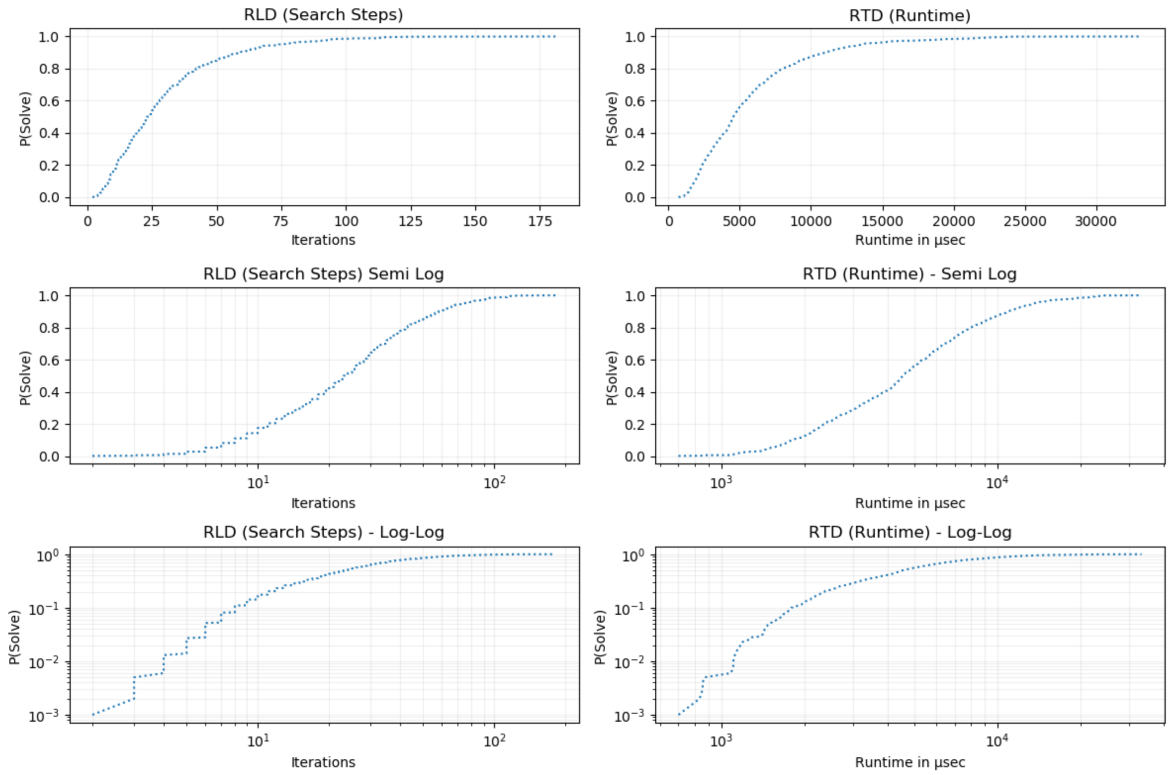
Appendix A

This appendix provides RTD diagrams for the individual RTD experiments for completeness.

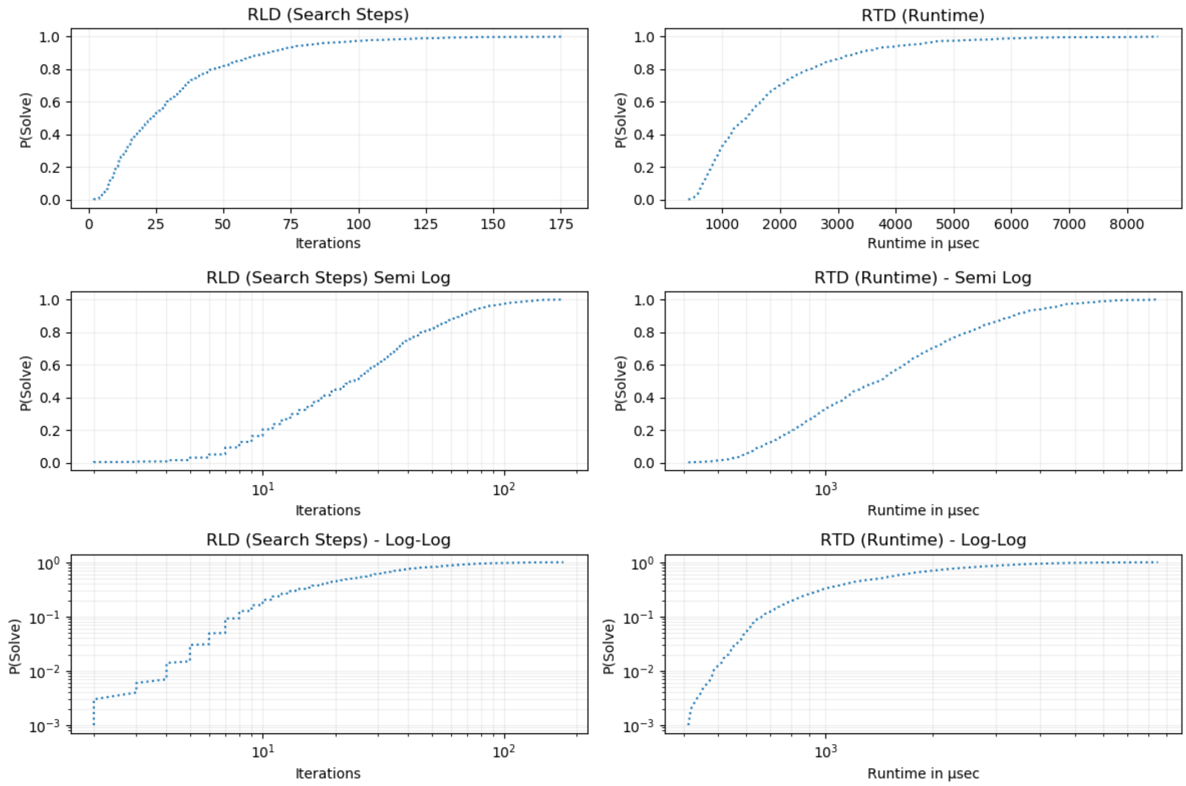
GWSAT uf20-01:



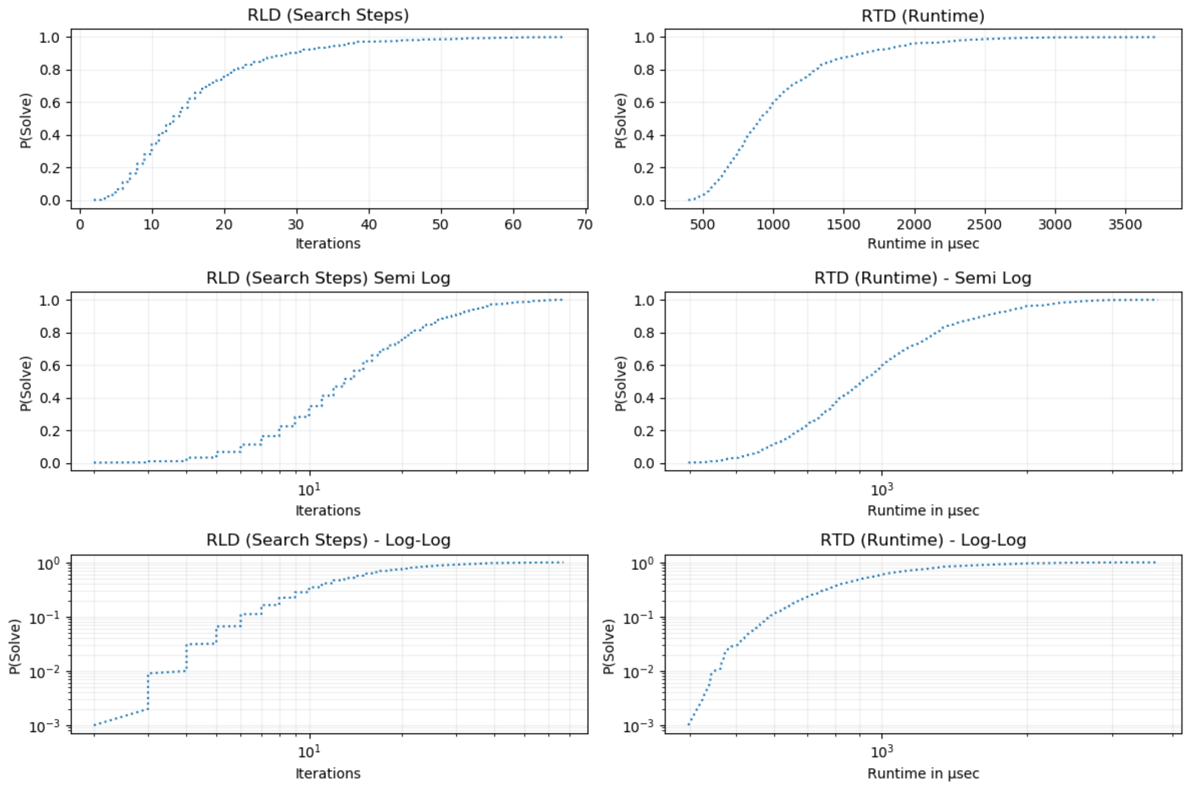
GWSAT uf20-02:



WalkSAT uf20-01:



WalkSAT uf20-02:



References

[1] https://pymotw.com/3/time/