Approximate Solution

Charles Kimbrough

JAMES MADISON UNIVERSITY

JAMES MADISON UNIVERSITY

Approach

1. Start With a Random Assignment
    a. Each variable is randomly set to True/False
    b. All clauses are evaluated once
    c. A list for each of the satisfied and unsatisfied clauses is created
2. Repeatedly Fix an Unsatisfied Clause
    a. Every iteration chooses one random unsatisfied clause and tries to fix it using one of two strategies. Randomly selected which one is chosen.

Strategy A: Random Walk

Has a probability of 0.4 to occur

Pick a literal from the unsatisfied clause and flip its variable

This is to escape any local minima by introducing random noise.

Strategy B: Greedy Improvement

Has a probability of 0.6 to occur

Evaluate the gain from flipping each variable in the clause

gain(x) = Δ(number of satisfied clauses)

Pick the variable with the best positive gain, tie-broken randomly

This is to climb uphill by making the single flip that increases satisfaction the most.

Approach Continued

After a certain number of max flips, the algorithm:

- saves its best score so far
- completely restarts with a new random assignment

This is to avoid getting stuck in a hole, and try a new selection

This runs for a certain time before quitting, to satisfy the anytime.

If the best score equals number of clauses, the algorithm quits early.

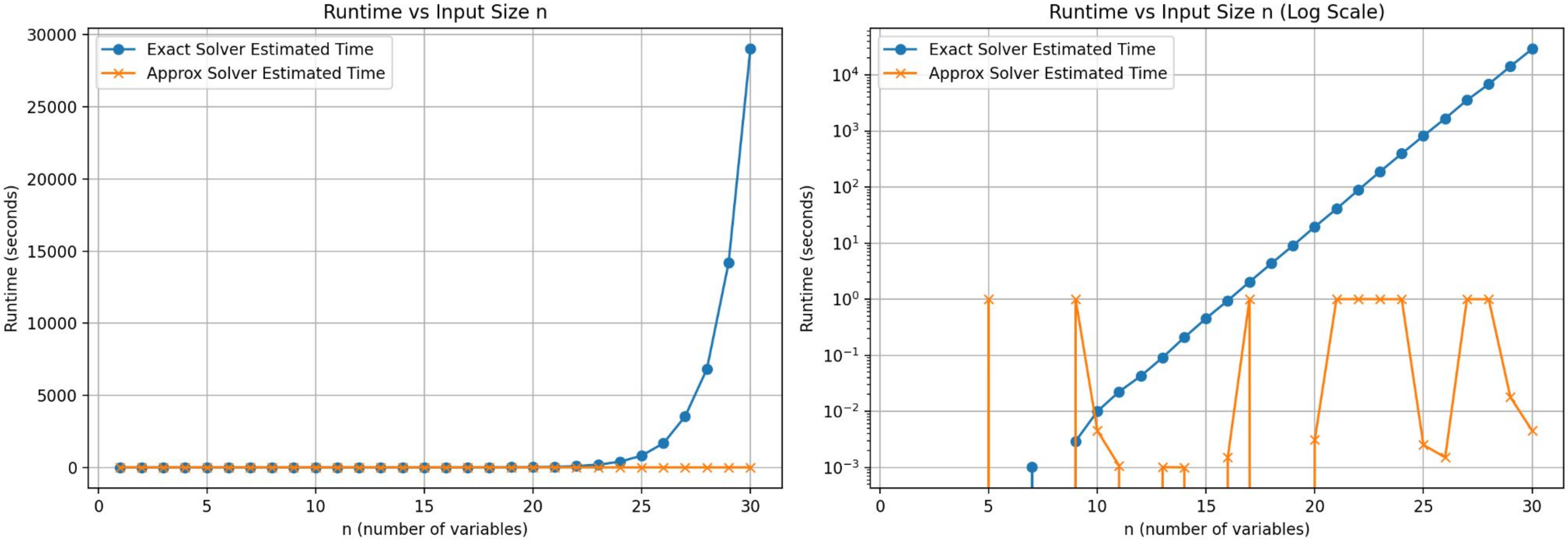
Runtime

n = number of variables

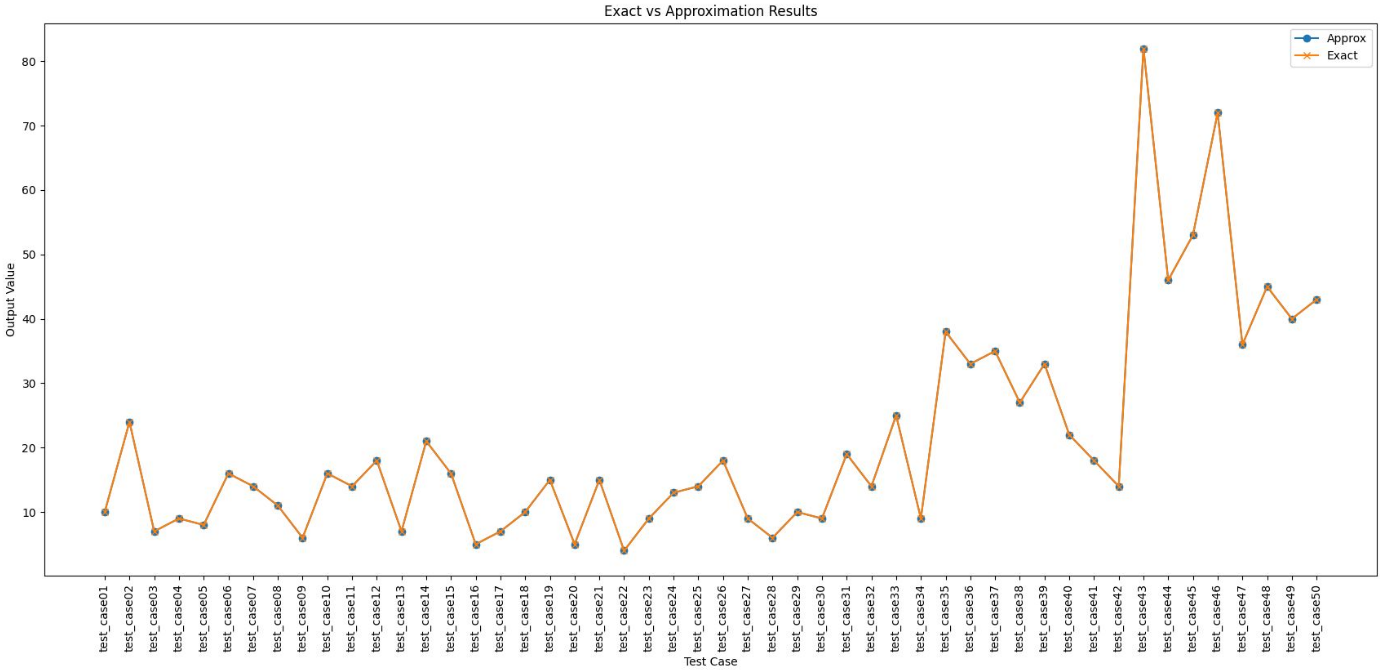
m = number of clauses

The cost per flip is O(m/n). Since this runs for a constant number for flips, the runtime for a single restart is O(m/n). Since most of time m>>n, it is closer to O(m).

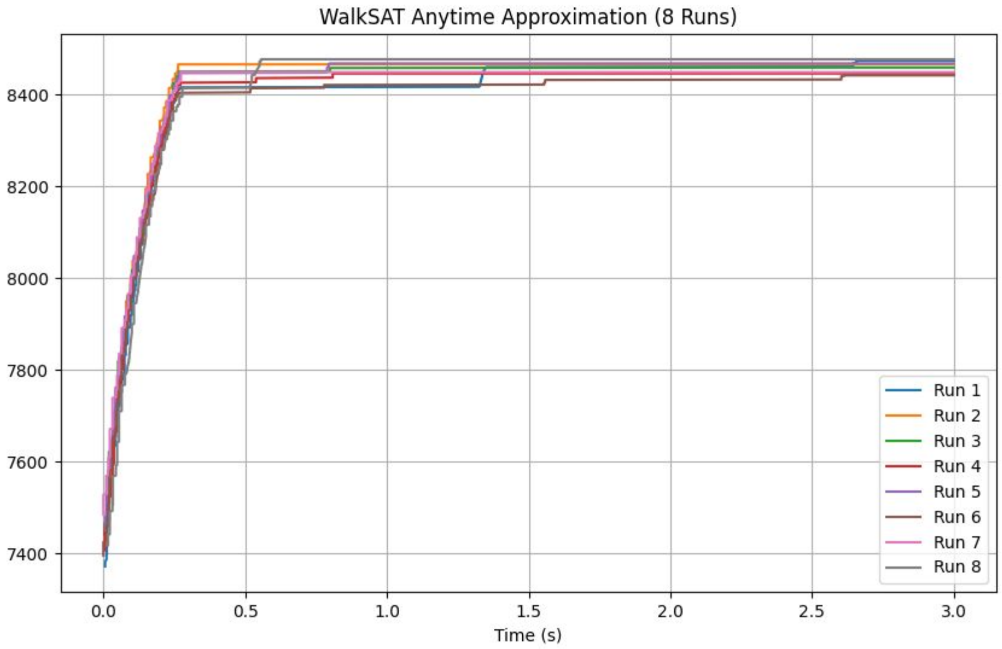
Approximation vs Exact Solution



Approximation vs Exact Solution



Solution Over Runtime



Parallelism

- Adding parallelism, decreased the initial speed of finding a solution.
- There was no consistent improvement in results for different parallelism.