QUBO Ground State Calculations

immediate

ABSTRACT

A program for calculating the ground-state count for QUBOs created from a user-specified 3-SAT formula. Furthermore, the created QUBOs are evaluated on the Dimod Tabu-Solver. The Superimposing process is based on the SATQUBOlib framework.

MOTIVATION

For the creation of QUBOs for 3-SAT formulas, the approach using pattern QUBOs can be used [Zielinski et al. (2023)]. The resulting matrices for a specific formula are here referred to as *Superimposed QUBOs*. Naturally, the question arises how the choice of Pattern QUBOs affects the superimposed QUBO. In this program, the number of ground states for the Superimposed is calculated based on the number of ground states present in each Pattern QUBO.
One aim is to investigate whether the use of Pattern QUBOs with a high number of ground states translates to Superimposed. To this end, when running the program for a 3-SAT formula, a number of Pattern QUBO combinations can be set. The created superimposed QUBOs can then be evaluated on the Dwave Tabu Sampler. The results are then plotted with the ground state count for each combination.

PATTERN QUBO STRUCTURE

Here, the boolean values *TRUE* and *FALSE* are represented by $0/1$, respectively.
The number of ground states mainly depends on the structure of the ancilla variable in the last row of the matrix. For clarification an example with a Type 0 Pattern QUBO:

$$\begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{Anc} \\ -1 & 1 & 1 & -1 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & -2 \end{pmatrix}$$

Figure 1. Type 0 Pattern QUBO Ψ

Note: Type 0 refers to clauses without any negations: $(a \vee b \vee c)$.
(for a detailed explanation refer to the Pattern QUBO paper [Zielinski et al. (2023)])

Since exactly one assignment has to be false, in this case 000 for both ancilla combinations 0000 and 0001, there has to be a minimum of 7 ground states for all possible solutions. The values of the ancilla variable **Anc** determine if a single solution is represented once or twice in the ground state set.

For example, take the valid clause assigment 011. The energy for both Ancilla combinations $0/1$ adds up to exactly -1.0. For this assignment the ancilla variable is called **balanced**. For an assignment like 010 where only one of the ancilla combinations adds up to the minimal energy the ancilla variable is called **unbalanced**.

The full assignments can be seen in the table below. Blue signifies ground states, orange the false assignment combinations.

Assignments	Values
0000	0.0
0001	2.0
0010	-1.0
0011	0.0
0100	-1.0
0101	0.0
0110	-1.0
0111	-1.0
1000	-1.0
1001	0.0
1010	-1.0
1011	-1.0
1100	-1.0
1101	-1.0
1110	0.0
1111	-1.0

Table 1. All Assignments and Their Values for QUBO Ψ.

For completeness the energy level distribution, which is simply counting the occurences:

$$\{-1.0 : 10, 0.0 : 5, 2.0 : 1\}$$

The ancilla state for each assignment for Ψ is saved in a dictionary, which will be used to calculate the superimposed QUBO:

$$\{000 : \oslash, 001 : 1, 010 : 1, 011 : 2, 100 : 1, 101 : 2, 110 : 2, 111 : 1\}$$

Here the false assignment is left in for clarity, this is not necessary in the code. The value 1 corresonds to a unbalanced, 2 to a balanced ancilla state.

ALGORITHM FOR COUNTING GROUND STATES

Since the matrices for 3-SAT formulas using the Pattern QUBO approach can get very large, brute force solvers like the dimod exact solvers are no longer sufficient. This specific solver is efficient for only up to 32 variables in a matrix.

For calucalting the ground states the structure of the ancilla variables can be leveraged. For each Pattern QUBO used in the superimposed construction a dictionary with the ancilla information is created. Then the solutions for the 3-SAT formula used for the matrix are for example generated with an ALLSAT solver [Toda and Soh (2016)]. For each solution the clause assignments are extracted and multiplied with the ancilla dictionary value. Finally all solution counts are added together.

More formally the steps used:

Algorithm 1 Superimposed ground state counter

Require: A set of Pattern QUBOs for each type Q_0, Q_1, Q_2, Q_3, formula ψ

Generate set of all solution bitstrings S for the formula
Generate solution dictionaries d_0, d_1, d_2, d_3 for each Q_i
Set totalgroundstatecount = 0
for solution s in solutions S **do**
    solutioncount = 1
    Obtain $s = \{s_1, s_2, \ldots, s_k\}$
    for clause c in formula **do**
        Extract solution assignments: $(s_i, s_{i+1}, s_{i+2}) \leftarrow$ c
        solutioncount $*=$ $d_i[(s_i, s_{i+1}, s_{i+2})]$
    totalgroundstatecount += solutioncount
    end for
end for
return totalgroundstatecount

EXAMPLE GROUND STATE CALCULATION

For clarification of the algorithm a quick example. The formula is written in dimacs format:

```
p cnf 3 5
1 2 3 0
1 2 -4 0
1 -2 -5 0
```

Here only the number of ground states for a single solution out of the solution set is demonstrated. The calculation for the superimposed QUBO for one solution is as follows:

1. Choose set of Pattern QUBOS Q
2. Generate Solution: (11000)
3. Create Dictionaries with ancilla information for each QUBO:
   - Type 0 $\{001:1, 010:1, 011:2, 100:1, 101:2, 110:2, 111:1\}$
   - Type 1 $\{000:1, 010:1, 011:2, 100:1, 101:2, 110:2, 111:1\}$
   - Type 2 $\{000:1, 001:1, 010:1, 100:1, 101:2, 110:1, 111:1\}$
   - Type 3 $\{000:1, 001:1, 010:1, 011:2, 100:1, 101:2, 110:2\}$
4. Exctract assignments for each clause:
   $Clause_1$: 1 1 0
   $Clause_2$: 1 1 0
   $Clause_3$: 1 1 0
5. solutioncount = First Clause $(110:2) *$ Second Clause $(110:2) *$ Third Clause $(110:1)$
   solutioncount = 2 * 2 * 1 = 4

This results in a ground state count of 4 for the single solution (11000). The same process is then repeated for each solution and all solution counts are added up together.

TABU SOLVER NOTES

The Tabu solver used here is the Dwave implementation.
For the Tabu solver the parameters timeout and number of reads have to be set. The timeout corresponds to the computation time alloted to each sample in milliseconds.
The number of reads corresponds to how many samples are generated for each QUBO. These samples are then aggregated in the resulting plots.
For very precise results, the timeout set to 1000 and the number of reads set to 100 is a good choice. But this is very resource-intensive and slow. If the goal is to simply observe the distribution of the different Pattern QUBO combinations, a timeout of 250 and number of reads of 20 is sufficient.

ALLSAT SOLVER NOTES

For the ALLSAT solver the maximal allocated time to generate all solutions for a formula has to be set in seconds. For a random formula it is not possible to know the solution time. It is recommended to first generate solvable formulas with a maximal solution time limit (for example 120 seconds) and then generate the ground state plots.
Solution time for randomly generated formulas can range from a few seconds to multiple days. For formulas with a large solution space, the available disk space can also become a problem. The resulting solution files can be multi-gigabytes large. When using the ALLSAT-Solver [Toda and Soh (2016)] make sure there is enough disk space available.

DOCKER EXECUTION

For running the ground state counter for your own Pattern QUBO combinations you have to set the following arguments:

```
--formula_path: formula path
--use_random: random QUBO choice Flag
--num_random: Number of random QUBOs Combinations to be analyzed
--num_reads: TABU number of reads
--timeout_TABU: TABU timeout
--timeout_minisat: Minisat timeout in seconds
--pattern_qubo_file: Path to the Pattern QUBO file
```

A detailed explanation of the code can be found here: Repository
The presented algorithm can be run from a Docker image. Running it includes the following steps

1. Make Docker Account: Docker Website
2. download Image: docker pull myusername/python-sat-groundstates:latest
3. Run Docker Image:

```
docker run \
  -v /home/user/Schreibtisch:/satqubolib_groundstates/shared \
  userDocker/python-sat-groundstates \
  --formula_path="/satqubolib_groundstates/shared/10.cnf" \
  --use_random=True \
  --num_random=100 \
  --num_reads=10 \
  --timeout_TABU=100 \
  --timeout_minisat=300 \
  --pattern_qubo_file /pattern_qubos.pkl
```

This results in the a couple of Plots. The chosen Pattern QUBO combinations have been evaluated on the Dwave Tabu Solver on the specified settings. The number of clauses fufilled are plotted on the y-axis, the number of ground states on the x-axis.

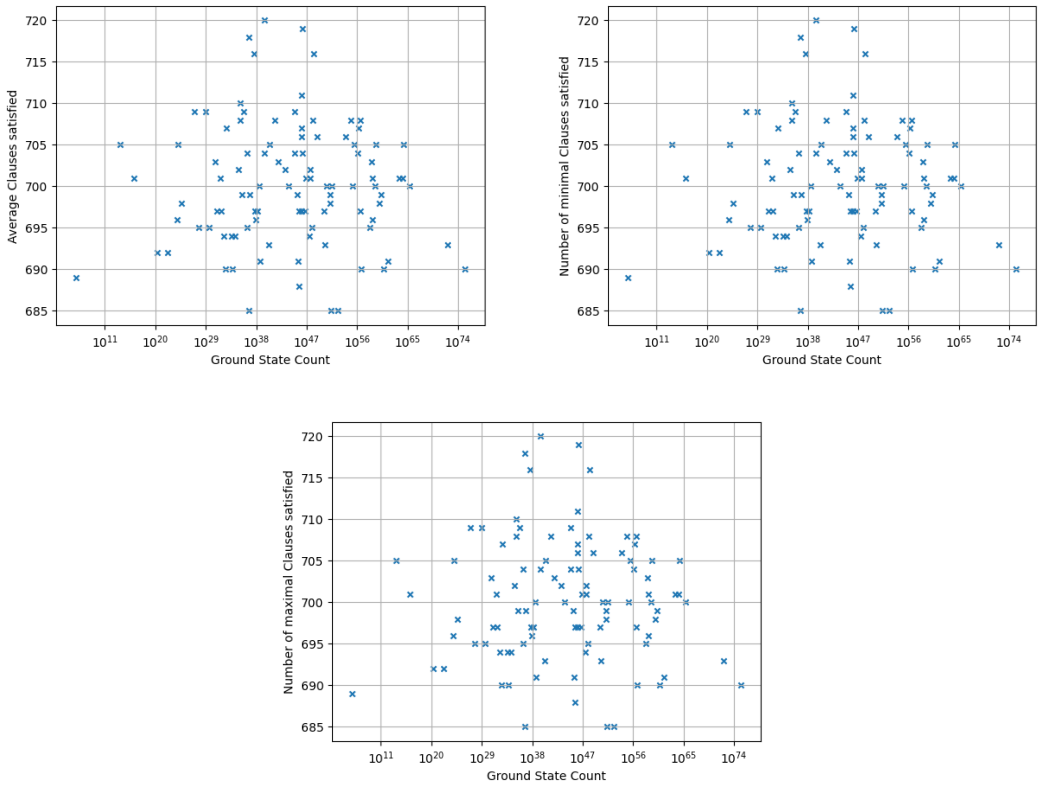

Figure 2. Example Dummy Plots

All the resulting plots and .csv files are saved inside a folder named after the formula used for evaluation.

REFERENCES

Toda, T. and Soh, T. (2016). Implementing efficient all solutions sat solvers. *ACM Journal of Experimental Algorithmics*, 21:1–44.

Zielinski, S., Nüßlein, J., Stein, J., Gabor, T., Linnhoff-Popien, C., and Feld, S. (2023). Pattern qubos: Algorithmic construction of 3sat-to-qubo transformations. *Electronics*, 12(16):3492.