Tutorial on MAXSAT for Ground States

Contact: Wenxuan Huang ([key01027@mit.edu](mailto:key01027@mit.edu)) ([key01026@gmail.com](mailto:key01026@gmail.com))

# Converting to 1/0 formulation

First important key is to convert the +1/-1, or any other formulation into the 1/0 formulation. For example,



We then know the conversion is:



Suppose the final Hamiltonian is: . It could be easily transformed as



This type of conversion should be simple to be implemented in the code. I do it in a way as if I am just expanding the equations term by term. Please tell me if you have any question.

Another example: for more complex ternary/quaternary/5th-nary systems. Example,



Note that this time, we have the additional constraint of . This just means that the spin cannot be both +1 and -1 at the same time. The relationships like  need to be recorded, because we need to feed this type of constraints into MAXSAT solver. I will tell you how to do this later. The transformation would be





here note that since , . And as , we know that, . Essentially, you would need to code things up to do this transformation. And quanternary/5-nary/… should be pretty similar.

More generally, to convert spins into 1/0 formulation, essentially you do:



And substitute and track symbols…

If you find it not intuitive/cannot be implemented, please contact me.

# Convert 0/1 formulation to MAXSAT

Now we have, the Hamiltonian in terms of the 0/1 formulation. And the hard constraints like  or etc. We need to define the MAXSAT formulation.

For MAXSAT, the first element is logic variable si: True or False. We then have the logic operation:



We then have the conjunctive normal form (CNF): a conjunction (logical AND) of disjunctions (logical OR). One example is:



If we use #SAT to denote the number of satisfied ORs, we have the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S1** | **S2** | **D1: S1∨¬S2** | **D2: ¬S1∨S2** | **D1∧D2** | **#SAT(D1,D2)** |
| True | False | True | False | False | 1 |
| False | False | True | True | True | 2 |

Hopefully this table is self-explanatory. Please contact me if not.

In most case, not every disjunction(OR) (frustration) can be satisfied: e.g., .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S1** | **D1: S1** | **D2: ¬S1** | **D1∧D2** | **#SAT(D1,D2)** |
| True | True | False | False | 1 |
| False | False | True | False | 1 |

Then, we would like to Maximize the number of satisfiable clauses. In this case, both s1 = True or False is solution to the MAXSAT (exact MAXSAT solver will only output one of this).

Now every disjunction can be weighted: for example . Then:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S1** | **D1: S1** | **D2: ¬S1** | **2D1∧3D2(weight)** | **#SAT(2D1,3D2)** |
| True | True | False | False(2) | 2 |
| False | False | True | False(3) | **3(MAXSAT result)** |

The complete name for this type of MAXSAT is **weighted MAXSAT**. **Note: the weight cannot be negative.**

Now, some of the clauses need to be exactly satisfied, for example,  (only one spin is allowed at one point) needs to be true. Such constraints is equivalent to  (for every pair of the variable, one of them needs to be false). This type of MAXSAT is called **partial weighted MAXSAT.** For our general ground state problem, we are dealing with partial weighted MAXSAT.

Now that we understand what MAXSAT is, we could convert ground state problems to MAXSAT. One important equation is:



Why? Any False(0) in s, LHS=1, RHS=1. All True(1) in s, LHS=0, RHS=0

Therefore:





Converting negative terms is a little bit difficult.





Then all the terms can be easily combined into final MAXSAT formulation with



Essentially, you just “AND” every term together.

Now, I believe such conversion should not be hard to be implemented with any code…

At the end, you will have a huge CNF formula. For ternary systems or above, you will have the set of must-satisfied constraints like .

# How to run MAXSAT

Now I will discuss about how do you run MAXSAT solver to get the ground state solution. The format we would use is weighted partial MAXSAT format from: <http://maxsat.ia.udl.cat/requirements/>

#### Weighted Partial Max-SAT input format

In Weighted Partial Max-SAT, the parameters line is "p wcnf *nbvar* *nbclauses* *top*". We associate a weight with each clause (disjunction), which is the first integer in the clause. Weights must be greater than or equal to 1, and the sum of all soft clauses smaller than 263. Hard clauses have weight equal or greater than *top* and soft clauses have a weight smaller than *top*. We assure that *top* is a weight always greater than the sum of the weights of violated soft clauses in the solution.

Example of Weigthed Partial Max-SAT formula (test.wcnf):

c  
c comments Weigthed Partial Max-SAT  
c  
p wcnf 4 5 1000000000000000000  
1000000000000000000 1 -2 4 0  
1000000000000000000 -1 -2 3 0  
8 -2 -4 0  
4 -3 2 0  
3 1 3 0

‘c’ is just some comments. ‘p’ stands for partial. ‘wcnf’ stands for weighted CNF. That’s just the necessary syntax. 4 is the number of variable s1 to s4. 5 is the total number of disjunctions. Within it, two are the “hard” (must satisfied disjunctions)



they have 1000000000000000000 (top) weights that must be satisfied.

The soft disjunctions are:



Therefore, to start, go to <https://github.com/dkitch/maxsat-ising> to download the code. Simply do ./build.sh, even though the Ground state code cannot be built, the MAXSAT solver should have been built without problem. You should see the following files in the bin

/maxsat-ising/bin$$ ls

CCLS2014\* CCLS\_to\_akmaxsat\* akmaxsat\_LB\* CCLS2014\_LB\* CCLS\_to\_akmaxsat\_LB\* akmaxsat\*

The ones that we need for the exact solver is CCLS2014, akmaxsat, CCLS\_to\_akmaxsat. To run the exact solver, you should do

./CCLS\_to\_akmaxsat test.wcnf 1e17

The 1e17 should just be there (to ensure the solution is exact) . And then you would see the final results as

c This is the CCLS\_to\_akmaxsat solver, Version: MAXSAT EVALUATION 2014 (2014.03.28)

c Many thanks to the akmaxsat team!

c ./CCLS2014 "test.wcnf" 1 10 > ./ccls\_res\_18183\_1503727980

c start CCLS

c stop CCLS

c ./akmaxsat "test.wcnf" ./ccls\_res\_18183\_1503727980 1e17

c start akmaxsat

c initialized bestCost to 0

o 0

c first\_lower\_bound\_threshold = 1

c 0 branches 0 propagates

c total generalized unit propagation = 0, success = nan%

s OPTIMUM FOUND

c Optimal Solution = 0

v 2 1 -4 3

c stop akmaxsat

Please look at <http://maxsat.ia.udl.cat/requirements/> for more detail about output format.

The only important lines are ‘o 0’, ‘s OPTIMUM FOUND’ and ‘v 2 1 -4 3‘. ‘v 2 1 -4 3‘ says that the result is s1=s2=s3=True and s4=False. ‘o 0’ is the total weight of un-satisfied disjunctions. In this case it is 0 since you can check that all clauses (disjunctions) are satisfied. ‘s OPTIMUM FOUND’ says optimal solution is found