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:



Download my prepared tutorial at: <https://github.com/chivalry123/MAXSAT_tutorial_Wenxuan>

I have prepared two types of solver, the first is **complete solver** CCLS\_to\_akmaxsat\_source\_codes/ (it would not stop until it finds the exact optimum). The other is **incomplete solver** CCEHC\_MAC/ (it will keep searching for good solution for a specific amount of time and provides the best solution it can find up to certain time without providing a proof of optimality). For relative small problems where the search space is 20-40 etc., it is suggested to use complete solver. For more complex problems where exact optimal cannot be found generally within practical time, it is suggested to use incomplete solver. For more state of the art complete/incomplete solvers, please follow the MAXSAT competitions:

<http://maxsat.ia.udl.cat/introduction/>

<http://mse17.cs.helsinki.fi>

Let’s first try complete solver: just do

cd CCLS\_to\_akmaxsat\_source\_codes/

./build.sh

You would get all the required files:

CCLS2015 akmaxsat CCEHC\_to\_akmaxsat

To run MAXSAT solver, simply do:

$ ./CCLS\_to\_akmaxsat test.wcnf

c This is the CCLS\_to\_akmaxsat solver, Version: MAXSAT EVALUATION 2015

c Many thanks to the akmaxsat team!

c start CCLS

c stop CCLS

c start akmaxsat

c initialized bestCost to 0

o 0

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

Suppose the problem is very difficult (example CCEHC\_MAC/IS368\_4.0.3.3.2.1\_softer\_periodic.wcnf), and complete solver might freeze and not able to provide an exact optimal solutions. We could use incomplete solver to handle the problem. Example

$ cd CCEHC\_MAC/

$ make

The executable is CCEHC\_finite

Usage: ./CCEHC\_finite -inst <instance> -seed <seed> -t <cutoff\_time\_in\_seconds>

$ ./CCEHC\_finite -inst IS368\_4.0.3.3.2.1\_softer\_periodic.wcnf -seed 1 -t 10

c This is the CCEHC solver, Version: MAX-SAT Evaluation 2015

c instance = IS368\_4.0.3.3.2.1\_softer\_periodic.wcnf

c problem type = weighted\_partial

c seed = 1

c num\_vars = 72

c num\_clauses = 77472

c num\_hclauses = 36

c num\_sclauses = 78084

c prob = 0.1

c sp = 0.0001

c large\_clause\_count\_threshold = 10

o 658147151284

o 656867386642

o 656712395774

o 651232355211

o 650140245941

o 646330874621

o 645841603064

o 645334855163

o 644747207393

o 644254944567

o 643484160926

o 642305287876

o 641335094669

o 636453833129

s UNKNOWN

c Best found solution with minimum cost: 636453833129

v 1 -2 3 -4 -5 6 7 -8 9 -10 -11 12 13 -14 -15 16 -17 18 19 -20 21 -22 -23 24 25 -26 27 -28 -29 30 31 -32 -33 34 -35 36 37 -38 39 -40 41 -42 43 -44 45 -46 -47 48 49 -50 -51 52 -53 54 55 -56 57 -58 -59 60 61 -62 63 -64 65 -66 67 -68 -69 70 -71 72

note that the ‘o’ values keeps decreasing (‘o’ represents the total weight of unsatisfied clauses). Meaning that more and more clauses(disjunctions) are satisfied (as we are doing MAXSAT).

I hope this MAXSAT tutorial is clear and if you find any additional problems. Please contact me.