MAX-SAT

Dr. Thomas Stützle

FG Intellektik, FB Informatik, TU Darmstadt
stuetzle@informatik.tu-darmstadt.de
www.intellektik.informatik.tu-darmstadt.de/~tom

Outline

- Introduction
- Instances
- Local Search
- Algorithms
- Conclusions

Satisfiability

- important problem in theoretical computer science, artificial intelligence, and applications
- first problem which was shown to be $\mathcal{NP}$-complete
- computationally hard ($\mathcal{NP}$-complete)
- can encode many other problems
- one of the conceptually simplest combinatorial problems

Given

- set $\mathcal{X}$ of n propositional variables $x_1, x_2, \ldots, x_n$
- each variable can be assigned a value 0 or 1 (*true* and *false*, respectively)
- a literal l_i is a variable or a negated variable (x_i or $\neg x_i$)
- set of k clauses $\mathcal{C} = \{C_1, ..C_k\}$, where each clause C_i is a disjunction of literals
- each clause evaluates to 1 (*true*) if at least one of its literals evaluates to 1 (*true*)

- propositional formula Φ, which is a conjunction of clauses

$$\Phi = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m_j} l_{ij}$$

Problem

- find an assignment with $x_i \in \{0,1\}(\{true, false\}$ such that Φ evaluates to 1 (*true*), i.e. all clauses are satisfied

Example

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$$

$\rightsquigarrow$ **satisfiable, two models:**

$x_1 = \text{true}, x_2 = \text{false}$
$x_1 = \text{false}, x_2 = \text{true}$

MAX-SAT

Optimization variant of SAT

- each clause may have an additional weight w_j

Goal

- find assignment maximizing the weight of satisfied clauses

Interest

- decision version (SAT) was first problem to be shown $\mathcal{NP}$-complete
- many problems in encoded as SAT / MAX-SAT
- study of approximate algorithms
- important problem in theory (approximation)

Differences SAT / MAX-SAT

- nature of problem (decision vs. optimization)
- complexity of special cases (E.g. 2-SAT)

MAX-SAT Instances

Main classes

- weighted vs. "unweighted" instances
- random instances vs. encoded instances

Input format

- depends on type of instance (weighted vs. unweighted)
- following DIMACS benchmark format

weighted case

```
c WCNF Format file produced by mwff.c
c with seed  923173271
c Variables per clause = 2
p wcnf 100 200
9 33 -38  0
9 11 64  0
8 -39 -66  0
.
.
%
0
```

unweighted case

```
c CNF Format file produced by mwff.c
c with seed  923173282
c Variables per clause = 3
p cnf 100 200
7 33 -38  0
-13 11 64  0
8 -39 -66  0
.
.
%
0
```


Input format

- comment lines (starting with "c")
- problem line

  (starting with "p"— wncf or cnf — No. variables — No. clauses)
- data (wcnf: weight — list of variables — ending "0")

  (cnf: list of variables — ending "0" (no weight is specified !))
- end file by

  %

  0

Local Search

1-opt local search

- flip in each step the assignment of one variable
- GSAT family
- WSAT family

Larger neighborhoods

- 2 and 3-Flip Neighborhoods
- initial results look promising

Evaluation function

- unweighted case: Number of satisfied clauses (equivalently minimize number of violated clauses)
- weighted case: Weight of the satisfied clauses (equivalently minimize weight of violated clauses)
- non-oblivious function *Battiti, Protasi, 1996*

Plateaus in search space

- for SAT often plateaus in the search space exist
- for SAT it is known that greedy descent local search should not stop at first non-improving solution but continue somewhat to surpass plateaus

Some remarks

- for local search, it does not matter, in principle, whether we attack SAT or MAX-SAT
- local search algorithms for SAT can be directly applied to MAX-SAT and vice versa
- for exact algorithms the difference between SAT and MAX-SAT matters very much


```
procedure local search for SAT / MAX-SAT
  input CNF / WCNF formula Φ, maxTries, maxSteps
  output variable assignment
  for i := 1 to maxTries do
    s := initAssign(Φ);
    for j := 1 to maxSteps do
      if s satisfies Φ then return s;
      else
        x := chooseVariable(Φ, s);
        s := s with flipped truth values for x;
      end if
    end for
  end for
end StochasticLocalSearch for SAT / MAX-SAT
```


The GSAT Algorithm *Selman, Mitchell, Levesque, 1992*

- in each search step, flip the variable which gives the maximal increase (or minimal decrease) in the number of unsatisfied clauses
- ties are broken randomly
- after *maxSteps* steps, restart from randomly chosen assignment

GSAT Extensions

- GWSAT
  - random walk step: randomly choose a currently unsatisfied clause and a literal in that clause, flip the corresponding variable to satisfy this clause
  - GWSAT steps: choose probabilistically between a GSAT step and a Random Walk step with 'walk probability' (noise) *wp*
- Tabu search extensions *Hansen, Jaumard, 1990; Selman, et al. 1993*
- Reactive search

Implementation aspects

- data structures to access easily instance data
  - each variable has list of variables which occur in a common clause
  - each variable has list of clauses
- dynamically update effect of flips
- critically satisfied clauses

The WalkSAT algorithm family

Selman, Kautz, 1994; McAllester et al., 1997

- search initialization: randomly chosen assignment
- search step
  1. randomly select a currently unsatisfied clause
  2. select a literal from this clause according to a heuristic h
- restart after *maxSteps* steps, from randomly chosen assignment

Some WalkSAT algorithms

SKC: Select variable such that minimal number of currently satisfied clauses become unsatisfied by flipping; if 'zero-damage' possible, always go for it; otherwise, with probability *wp* variable is randomly selected

Tabu: Select variable that maxi-mises increase in total number of satisfied clauses when flipped; use constant length tabu-list for flipped variables and random tie-breaking

Novelty: Order variables according to increase in total number of satisfied clauses when flipped; if best variable in this ordering not most recently flipped, always go for it; otherwise, select second-best with probability wp

R-Novelty: Similar to Novelty, but more complex decision between the variable with best and second-best score

Implementation details

- basic data structures analogous to GSAT family
- no need for dynamic update of effects of flips
- selection is done among less variables
  $\rightsquigarrow$ speed advantage

GSAT vs. WSAT family of local search algorithms

- GSAT local search is typically more greedy
- for SAT, WSAT based algorithms are the top performers
- performance on MAX-SAT depends strongly on number of unsatisfied clauses in optimal
- observation for randomly generated, unweighted MAX-SAT regarding No. of unsatisfied clauses in optimal solutions:
  - if very low $\rightsquigarrow$ advantage for WSAT type algorithms
  - if high or very high $\rightsquigarrow$ advantage for GSAT type algorithms

Algorithms

"unweighted" MAX-SAT

- standard local search algorithms for SAT
  - GSAT type algorithms
  - WSAT type algorithms
- "steepest ascent mildest descent" (Tabu Search)
  Hansen, Jaumard, 1990
  - first tabu search for MAX-SAT
  - several later tabu search variants

- simulated annealing *Beringer et al., 1994, Spears, 1996*
  - mainly tested on SAT
  - poor performance on SAT
  - was outperformed by "steepest ascent mildest descent"
- reactive search *Battiti, Protasi, 1997*
  - systematic diversification phases implemented by a tabu search algorithm change with descent phases
  - automatic adjustment of diversification strength
  - important tie-breaking criterion

"weighted" MAX-SAT

- in part same algorithms as for unweighted case
- GRASP *Resende et al., 1996*
  - greedy construction heuristic with local search phase
  - tested algorithm on weighted extension of random 3-SAT instances
- Variable Neighborhood Search *Hansen, Mladenovic, 1998?*
  - straightforward VNS approach to weighted MAX-SAT
  - skewed VNS
  - similar results as with tabu search

- discrete Lagrangian search *Shang, Wah, 1997*
  - extension of Lagrangian methods to SAT / MAX-SAT
  - apparently very good performance
- guided local search *Mills, Tsang, 2000*
  - guided local search embedding a GSAT-type algorithm
  - good results on SAT / MAX-SAT
  - currently best results on Resende's weighted MAX-SAT instances

.. and many others to come

Exact algorithms

- typically limited to small instance sizes
- for "unweighted" MAX-SAT many are straightforward extensions of Davis-Putnam style backtracking algorithms
- for "weighted" case some good results with Branch& Cut algorithms

Conclusions

MAX-SAT is ..

- one of the syntactically simplest optimization problems
- very hard for exact algorithms
- allows the study of the behavior of algorithms

Work in metaheuristics network

- study of algorithm behavior
- comparison to “traditional” approximation algorithms
- applications to large instances
- search space analysis