A HyFlex Module for the MAX-SAT Problem

Matthew Hyde, Gabriela Ochoa, José Antonio Vázquez-Rodríguez, Tim Curtois

Automated Scheduling, Optimisation and Planning (ASAP) Group, School of Computer Science, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham. NG8 1BB. UK

1 Problem Formulation

'SAT' refers to the boolean satisfiability problem. This problem involves determining if there is an assignment of the boolean variables of a formula, which results in the whole formula evaluating to true. If there is such an assignment then the formula is said to be satisfiable, and if not then it is unsatisfiable. We consider here one of its related optimisation problems, the maximum satisfiability problem (MAX-SAT), in which the objective is to find the maximum number of clauses of a given Boolean formula that can be satisfied by some assignment. The problem can also be formulated as a minimisation problem, where the objective is to minimise the number of unsatisfied clauses.

An example formula is given in equation 1, which is satisfied when $x_1 = false$ $x_2 = false$ $x_3 = true$ and $x_4 = false$.

$$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_4) \quad (1)$$

Some of the problem instances included in this problem domain class are examples of the so called 3SAT problem, where each clause contains three variables, as in equation 1. However, not all of the instances have this property.

2 Solution Initialisation

The solutions are initialised by simply randomly assigning a true or false value to each variable.

3 Low Level Heuristics

Sections 3.2, 3.3, 3.4, and 3.5 explain the local search, mutational, ruin-recreate, and crossover low level heuristics respectively. We have implemented nine low level heuristics in total. Some of the descriptions are taken from [1].

3.1 Definitions

Let T be the state of the formula before the variable is flipped, and let T' be the state of the formula after the variable is flipped.

3.1.1 Net Gain

The net gain of a variable is defined as the number of broken clauses in T minus the number of broken clauses in T'.

3.1.2 Positive Gain

The positive gain of a variable is the number of broken clauses in T that are satisfied in T'.

3.1.3 Negative Gain

The positive gain of a variable is the number of satisfied clauses in T that are broken in T'.

3.1.4 Variable Age

The age of a variable is the number of variable flips since it was last flipped.

3.2 Local search heuristics

These heuristics implement 'first-improvement' local search operators. In each iteration, a neighbour is generated, and it is accepted immediately if it has superior or equal fitness. If the neighbour is worse, then the change is not accepted. The behaviour of these heuristics is controlled with the 'depth of search' parameter. At the default value of 0.2, these heuristics iterate 10 times. If it is set higher, the heuristics iterate up to 20 times. Local search heuristics cannot produce a solution of worse fitness.

3.2.1 Flip Random Variable

Flip a variable selected completely at random.

3.2.2 Flip Random Variable from a Broken Clause

Flip a randomly selected variable from a randomly selected broken clause.

3.3 Mutational heuristics

The behaviour of these heuristics is controlled with the 'intensity of mutation' parameter. At the default value of 0.2, these heuristics run once. If it is set higher, the heuristics repeat up to five times, meaning a greater mutation is performed.

3.3.1 Flip Random Variable

Flip a variable selected completely at random.

3.3.2 Flip Random Variable from a Broken Clause

Flip a randomly selected variable from a randomly selected broken clause.

3.3.3 GSAT [2]

Flip the variable with the highest net gain, and break ties randomly.

3.3.4 HSAT [3]

Identical functionality to GSAT, but ties are broken by selecting the variable with the highest age.

3.3.5 WalkSAT [4]

Select a random broken clause BC. If any variables in BC have a negative gain of zero, randomly select one of these to flip. If no such variable exists, flip a random variable in BC with probability 0.5, otherwise flip the variable with minimal negative gain.

3.3.6 Novelty [5]

Select a random broken clause BC. Flip the variable v with the highest net gain, unless v has the minimal age in BC. If this is the case, then flip it with 0.3 probability. Otherwise flip the variable with the second highest net gain.

3.4 Ruin-Recreate heuristics

3.4.1 Reinitialise Variables

A proportion of the variables is randomly reinitialised. Depending on the value of the "intensity of mutation" parameter, either 0.2, 0.4, 0.6, or 0.8 of the solution is reinitialised.

3.5 Crossover heuristics

3.5.1 One point crossover

Standard one point crossover on the boolean strings of variables.

3.5.2 Two point crossover

Standard two point crossover on the boolean strings of variables.

4 Problem Instances

The problem instances are various selections taken from the 'SATLIB' website [6], and the SAT 2007 and 2009 competitions. Instances are also included from the MAXSAT 2010 competition, available at http://www.maxsat.udl.cat/10/benchmarks/. The instances contain between 200 and 800 variables, and between 1000 and 3500 clauses.

References

[1] Alex S. Fukunaga. Automated discovery of local search heuristics for satisfiability testing. *Evolutionary Computation (MIT Press)*, 16(1):31–1, 2008.

[2] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI'92)*, pages 440–446, San Jose, CA, USA, July 1992.

[3] I. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for sat. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93)*, pages 28–33, Washington D.C., USA, July 1993.

[4] B. Selman, H. Kautz, , and B. Cohen. Noise strategies for improving local search. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'94)*, pages 337–343, Seattle, WA, USA, July 1994.

[5] D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI)*, pages 459–465, Providence, Rhode Island, USA, July 1997.

[6] Holger H. Hoos and Thomas Stützle. Satlib: An online resource for research on sat. In I. P. Gent, H. V. Maaren, and T. Walsh, editors, *SAT 2000*, pages 283–292. IOS Press, 2000. SATLIB is available online at www.satlib.org.