CCAnr

Shaowei Cai and Kaile Su
IIIS, Griffith University
shaoweicai.cs@gmail.com; k.su@griffith.edu.au

***Abstract*—This note describes the SAT solver "CCAnr", which is a local search solver designed for non-random SAT instances.**

I. Introduction

Recently, we proposed a diversification strategy for local search, which is called configuration checking (CC) [1]. The CC strategy has been successfully used to improve SAT local search algorithms [2], [3], [4]. Especially, the CCA heuristic in [3] combines an aspiration mechanism with the CC strategy and results in the Swcca algorithm, which shows state-of-the-art performance on a large range of instances. Based on the Swcca algorithm, we have developed an SLS solver called CCASat [5], which performs very well on random instances.

CCAnr (CCA-based algorithm for non-random SAT) is an improved version of Swcca, aiming to solve non-random instances more effectively. CCAnr differs from Swcca in two small but important modifications. First, in the diversification mode, after selecting a random unsatisfied clause, while Swcca picks the oldest variable from the clause to flip, CCAnr picks the variable with the greatest $score$ from the selected clause, breaking ties by favoring the oldest one. Secondly, Swcca utilizes the formula $w(c_i) := \lfloor \rho \cdot w(c_i) \rfloor + \lfloor (1-\rho) \cdot \overline{w} \rfloor$ to smooth clause weights, while in CCAnr, this smoothing formula is generalized as $w(c_i) := \lfloor \rho \cdot w(c_i) \rfloor + \lfloor q \cdot \overline{w} \rfloor$. By setting $q = 0$, the smoothing scheme actually becomes a "forgetting" scheme, which is similar to the one used in a recent local search algorithm for Minimum Vertex Cover [6].

II. Main Techniques

CCAnr is an stochastic local search (SLS) algorithm based on the CCA search framework.

We first give some definitions for the CCA heuristic. A variable x is said configuration changed iff $confChange[x] = 1$. A *configuration changed decreasing* (CCD) variable is a variable with both $confChange[x] = 1$ and $score(x) > 0$. A *significant decreasing* (SD) variable is a variable with $score(x) > g$, where g is a positive integer large enough, and in this work g is set to the averaged clause weight (over all clauses) $\overline{w}$.

Originally proposed in [3], the CCA heuristic (outlined in Algorithm 1) can be generalized as follows: The CCA heuristic switches between the greedy mode and the diversification mode. In the greedy mode, there are two levels with descending priorities. On the first level it picks the CCD variable with the greatest score to flip. If there are no CCD variables, CCA selects the SD variable with the greatest score to flip if there is one, which corresponds to the second level. If there are neither CCD variables nor SD variables, CCA switches to the the diversification mode, where clause weights are updated, and a variable in a random unsatisfied clause is picked to flip.

Algorithm 1: *pickVar*-heuristic CCA

```
//greedy mode
if there exist CCD variables then return a CCD variable with the greatest score;
if there exist SD variables then return an SD variable with the greatest score;
//diversification mode
update clause weights;
pick a random unsatisfied clause c;
return a variable in c;
```

Therefore, to design an SLS algorithm based on the CCA heuristic, we need to specify three technique details: (1), the tie-breaking mechanism in the greedy mode; (2), the clause weighting scheme; and (3), the heuristic to pick a variable from an unsatsified clause in the diversification mode.

III. The CCAnr Algorithm

CCAnr is based on the CCA heuristic, and the three technique details in the CCA heuristic are specified as follows for CCAnr:

1) the tie-breaking mechanism in the greedy mode: CCAnr break ties by favoring the oldest variable in the greedy mode, as Swcca does.
2) the clause weighting scheme: CCAnr adopts a Threshold-based Smoothed Weighting (TSW) scheme. Each time TSW is called, clause weights of all unsatisfied clauses are increased by one; further, if the averaged weight $\overline{w}$ exceeds a threshold γ, all clause weights are smoothed as $w(c_i) := \lfloor \rho \cdot w(c_i) \rfloor + \lfloor q \cdot \overline{w} \rfloor$.
3) the pick-var heuristic in the diversification mode: CCAnr picks the variable with the greatest $score$ from an unsatisfied clause, breaking ties by favoring the oldest one.

IV. Main Parameters

There are three parameters in CCAnr: the average weight threshold parameter γ, and the two factor parameters ρ and q. All of the three parameters are for the TSW weighting scheme.

The parameters are set as follows: $\gamma = 300$; $\rho = 0.3$; q is set to 0 if $r \leq 15$, and 0.7 otherwise (r is the ratio of the instance).

V. Implementation Details

CCAnr is implemented in C++. It is developed based on the codes of Swcca solver [3], which can be downloaded from www.shaoweicai.net/research.html.

VI. SAT Competition 2013 Specifies

CCAnr is submitted to "Core solvers, Sequential, Hard-combinatorial SAT track". It is compiled by g++ with the 'O2' optimization option.

Its running command is:

CCAnr <instance file name> <random seed>.

References

[1] S. Cai, K. Su, and A. Sattar, "Local search with edge weighting and configuration checking heuristics for minimum vertex cover," *Artif. Intell.*, vol. 175, no. 9-10, pp. 1672–1696, 2011.

[2] S. Cai and K. Su, "Local search with configuration checking for SAT," in *Proc. of ICTAI-11*, 2011, pp. 59–66.

[3] ——, "Configuration checking with aspiration in local search for SAT," in *Proc. of AAAI-12*, 2012, pp. 334–340.

[4] C. Luo, K. Su, and S. Cai, "Improving local search for random 3-sat using quantitative configuration checking," in *Proc. of ECAI-12*, 2012, pp. 570–575.

[5] S. Cai, C. Luo, and K. Su, "CCASat: Solver description," in *Proc. of SAT Challenge 2012: Solver and Benchmark Descriptions*, 2012, pp. 13–14.

[6] S. Cai, K. Su, C. Luo, and A. Sattar, "NuMVC: An efficient local search algorithm for minimum vertex cover," *J. Artif. Intell. Res. (JAIR)*, vol. 46, pp. 687–716, 2013.