AI448PBSolver

Yoshiya Imai*

This solver[1] is a Rust implementation of the naive CDCL algorithm. It uses only psudo boolean conflict analysis to generate learnt constraints and does not use LP relaxation, so it does not require external SAT or LP solvers.

The conflict analysis in this solver uses a simple resolving and rounding algorithm. The resolving algorithm(Algorithm1) generates a new conflict constraint from a conflict constraint C and a reason constraint C'. r is a resolving variable and ρ is the current partial assignment. See, for example [2] for definition of ρ, slack(C, ρ) and outline of pseudo boolean conflict analysis. This algorithm uses slack to determine the need for rounding and which constraint to round. The rounding algorithm (Algorithm2) generates a valid inequality whose slack is equal to zero. In select_variables_from, N_u is subset of N_f and can be equal to N_f. However if it is possible to propagate the assignment of r with fewer assignments than ρ, a stronger constraint can be obtained by making N_u a smaller set.

Algorithm 1 resolve($C : \sum_{j \in N} a_j l_j \geq b$, $C' : \sum_{j \in N} a'_j l_j \geq b'$, r, ρ)

$s := \frac{\text{slack}(C,\rho)}{a_r}$
$s' := \frac{\text{slack}(C',\rho)}{a'_r}$
if $s + s' < 1$ **then**
  return $a'_r C + a_r C'$
else if $s > s'$ **then**
  return $a'_r \text{round}(C, r, \rho) + C'$
else
  return $C + a_r \text{round}(C', r, \rho)$
end if

Algorithm 2 round($\sum_{j \in N} a_j l_j \geq b$, r, ρ)

1: $N_f := \{j \in N | j \neq r \wedge \rho(j) = 0\}$
2: $N_u :=$ select_variables_from N_f
3: $N_d := N \backslash (N_u \cup \{r\})$
4: **return** $l_r + \sum_{j \in N_u} \lceil \frac{a_j}{a_r} \rceil l_j + \sum_{j \in N_d} \lfloor \frac{a_j}{a_r} \rfloor l_j \geq \lceil \frac{b}{a_r} - \sum_{j \in N_d} (\frac{a_j}{a_r} - \lfloor \frac{a_j}{a_r} \rfloor) \rceil$

References

[1] https://github.com/AI448/PBSolver25

[2] Gioni Mexi and Timo Berthold and Ambros Gleixner and Jakob Nordström. Improving Conflict Analysis in MIP Solvers by Pseudo-Boolean Reasoning. In *Proceedings of the 29th International Conference on Principles and Practice of Constraint Programming (CP '23)*, Vol. 280 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 27:1–27:19, 2023.

*Freelancer in Japan