

Fast and Flexible Difference Constraint Propagation for DPLL(T)

Paper: Scott Cotton and Oded Maler
Slides: Martin Milata

September 30, 2010

Propositional formula satisfiability problem – SAT

Given propositional formula φ , is there an assignment that makes φ evaluate to *true*?

- ▶ NP-complete.
- ▶ Program that is able to solve SAT is called **SAT solver**.
 - ▶ Lots of attention devoted to development of efficient SAT solvers.
- ▶ We usually require the formula to be in **conjunctive normal form**.

Solving SAT – DPLL algorithm

- ▶ Introduced in **1962** by Davis, Putnam, Logemann and Loveland.
- ▶ Still used nowadays.
- ▶ Basically depth-first search of variable assignments with pruning and unit propagation.
- ▶ In practice used with various heuristics & tricks.

DPLL(φ , I , todo):

if φ is true under I **then**

return True

if φ is false under I **then**

return False

if there is unit clause (l) **then**

 todo := todo \cup $\{l\}$

if there is $l \in$ todo **then**

 todo := todo \setminus $\{l\}$

return Extend(φ , I , todo, l)

else

$l :=$ unassigned-literal(φ)

return Extend(φ , I , todo, l)

 or Extend(φ , I , todo, \bar{l})

Extend(φ , I , todo, l):

$I := I \cup \{l\}$

return DPLL(φ , I , todo)

Satisfiability modulo theory – SMT

Given formula φ of predicate logic, is it satisfiable with respect to theory T ?

- ▶ SAT with propositional variables replaced by atoms of language of T .
- ▶ Formulas usually required to be without quantifiers.

Example theories:

- ▶ Equality and uninterpreted functions
 - ▶ $(x \neq y \wedge f(x) = f(y)) \vee (\neg P(x) \wedge P(f(x)))$
- ▶ Linear arithmetic
 - ▶ $6x + 7y + 8 = z \wedge (\neg(z \geq 2y) \vee x \geq 9y + 5)$
- ▶ Real difference logic
 - ▶ $(\neg(y - x \leq 6) \wedge \neg(x - y \leq 4)) \vee x - z \leq 42$

DPLL(T)

Framework for solving SMT based on the DPLL algorithm.
Algorithm for theory T consists of two parts:

DPLL(X) theory-independent, takes care of the boolean structure, does not know “meaning” of the literals

Solver $_T$ theory-specific, provides information whether conjunction of literals is consistent & eventual hints to DPLL(X)

Each Solver $_T$ instance must provide following methods:

SetTrue add literal & indicate, whether the set is consistent

TheoryProp which unassigned literals are consequences of current assignment?

DPLL-T(φ , I , todo):

if φ is true under I **then**
 return True

if φ is false under I **then**
 return False

if there is unit clause (l) **then**
 todo := todo \cup $\{l\}$

if there is $l \in \text{todo}$ **then**
 todo := todo \setminus $\{l\}$
 return Extend(φ , I , todo, l)

else

$l :=$ unassigned-literal(φ)
 return Extend(φ , I , todo, l)
 or Extend(φ , I , todo, \bar{l})

Extend(φ , I , todo, l):

$I :=$ SetTrue(l)

if $I =$ "inconsistent" **then**
 return False

 tp := TheoryProp(I)

return DPLL(φ , I , todo \cup tp)

Constraint graphs

Goal: efficient implementation of SetTrue and TheoryProp for difference logic.

Constraint graph (=DBM) used to represent conjunction of constraints – edge $x \xrightarrow{c} y$ for each $x - y \leq c$.

Theorem 1

Let Γ be a conjunction of difference constraints, and let G be the constraint graph of Γ . Then Γ is satisfiable if and only if there is no negative cycle in G . Furthermore, if Γ is satisfiable, then $\Gamma \models x - y \leq c$ if and only if y is reachable from x in G and $c \geq d_{xy}$ where d_{xy} is the length of a shortest path from x to y in G .

Consistency check – Set True

Negative cycle detection is expensive – but we may reuse information from previous run.

Solution: maintain **potential function** π – function on vertices such that $\pi(x) + c - \pi(y) \geq 0$ for each edge $x \xrightarrow{c} y$.

If such a function can be found, then the graph does not contain negative cycle:

$$\sum_{i=1}^{n-1} \pi(v_i) + c_i - \pi(v_{i+1}) \geq 0$$

$$\pi(v_1) - \pi(v_n) + \sum_{i=1}^{n-1} c_i \geq 0$$

$$\sum_{i=1}^{n-1} c_i \geq \pi(v_n) - \pi(v_1)$$

i.e. for $v_n = v_1$, $\sum_{i=1}^{n-1} c_i \geq 0$.

Incremental consistency check

Problem: given constraint graph G with valid potential function π , compute potential function π' for the graph with added edge $u \xrightarrow{d} v$ (or detect negative cycle).

$$\gamma(v) := \pi(u) + d - \pi(v)$$

$$\gamma(w) := 0 \text{ for all } w \neq v$$

while $\min(\gamma) < 0 \wedge \gamma(u) = 0$ **do**

$$s := \operatorname{argmin}(\gamma)$$

$$\pi'(s) := \pi(s) + \gamma(s)$$

$$\gamma(s) := 0$$

for $s \xrightarrow{c} t \in G$ **do**

if $\pi'(t) = \pi(t)$ **then**

$$\gamma(t) := \min(\gamma(t), \pi'(s) + c - \pi(t))$$

Time: $\mathcal{O}(m + n \log n)$.

Constraint propagation – TheoryProp

Find which unassigned literals are consequences of the current assignment.

Theorem 1

Let Γ be a conjunction of difference constraints, and let G be the constraint graph of Γ . Then Γ is satisfiable if and only if there is no negative cycle in G . Furthermore, if Γ is satisfiable, then $\Gamma \models x - y \leq c$ if and only if y is reachable from x in G and $c \geq d_{xy}$ where d_{xy} is the length of a shortest path from x to y in G .

- ▶ Compute shortest path in constraint graph and compare with unassigned constraints.
- ▶ We want to do this incrementally, so we are only interested which unassigned constraints became consequences after adding edge $x \xrightarrow{c} y$.

Shortest path through edge $x \xrightarrow{c} y$

1. Compute shortest paths from all vertices to x (δ_x^{\leftarrow}).
2. Compute shortest paths from y to all vertices to (δ_y^{\rightarrow}).
3. The shortest path from u to v through $x \xrightarrow{c} y$ is then $\delta_x^{\leftarrow}(u) + c + \delta_y^{\rightarrow}(v)$.

Steps 1. and 2. can be done using algorithm for computing **single source shortest paths** (SSSP). Takes $\mathcal{O}(mn)$ with arbitrary values on edges.

- ▶ We can use alternative edge weight $\pi(x) + c - \pi(y)$, which is always non-negative.
- ▶ This allows us to use SSSP algorithms (e.g. Dijkstra's), which are $\mathcal{O}(m + n \log n)$.

Further improving constraint propagation

- ▶ Early termination of the SSSP algorithm – do not compute shortest paths for vertices whose minimal distance is clearly unaffected by $x \xrightarrow{c} y$.
- ▶ If the shortest path from vertex u to y does not pass through x , then we don't have to compute it (u is δ_y^{\leftarrow} -irrelevant).
- ▶ If the shortest path from vertex x to v does not pass through y , then we don't have to compute it (v is δ_x^{\rightarrow} -irrelevant).
- ▶ The SSSP algorithm can be modified so it terminates when only irrelevant vertices are left to be processed.

Summary

We can...

- ▶ solve satisfiability of boolean combination of difference constraints
- ▶ use the **DPLL(T)** framework, which means we only have to care about the **SetTrue** and **TheoryProp** methods
- ▶ use constraint graph/DBM for representing conjunction of constraints
- ▶ ensure consistency of the constraints by maintaining the **potential function**
- ▶ find the consequences of added edge by running SSSP twice and checking unassigned constraints
- ▶ speed this up by taking advantage of the potential function and possibility of **early termination**

Summary

We can...

- ▶ solve satisfiability of boolean combination of difference constraints
- ▶ use the **DPLL(T)** framework, which means we only have to care about the **SetTrue** and **TheoryProp** methods
- ▶ use constraint graph/DBM for representing conjunction of constraints
- ▶ ensure consistency of the constraints by maintaining the **potential function**
- ▶ find the consequences of added edge by running SSSP twice and checking unassigned constraints
- ▶ speed this up by taking advantage of the potential function and possibility of **early termination**

Thank you.