Pseudo-Boolean Solving by Incremental Translation to SAT

Pete Manolios Vasilis Papavasileiou

Northeastern University
{pete,vpap}@ccs.neu.edu

October 31, 2011

Motivation: Industrial Design Problems

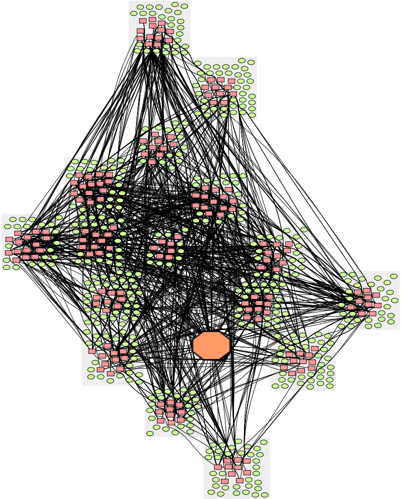
[1] Photo by Luis Argerich, CC-by-2.0

Motivation: Industrial Design Problems

How to connect, integrate, assemble thousands of *components* in an aerospace design, subject to global *requirements*?

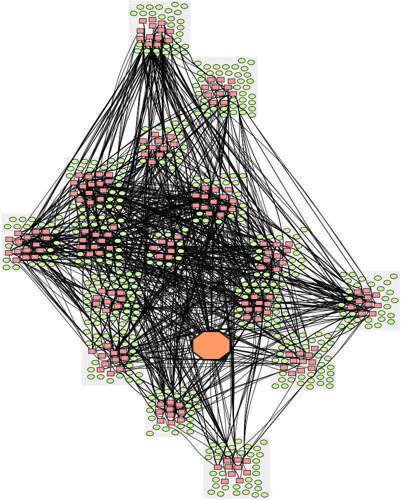
[1] Photo by Luis Argerich, CC-by-2.0

Solution: Synthesizing Architectures[1]



[1] CAV 2011

Solution: Synthesizing Architectures[1]



The core of the problem is *pseudo-Boolean* constraints!

[1] CAV 2011

Pseudo-Boolean (PB) Constraints

Pseudo-Boolean Constraint

Constraint of the form $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \Box r$

- $\Box$ is one of $<$, $\leq$, $=$, $>$, or $\geq$

[1] Een and Sorensson, JSAT, 2006 (MiniSat+)

Pseudo-Boolean (PB) Constraints

Pseudo-Boolean Constraint

Constraint of the form $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \,\Box\, r$

- $\Box$ is one of $<$, $\leq$, $=$, $>$, or $\geq$
- variables $x_i \in \{0, 1\}$

[1] Een and Sorensson, JSAT, 2006 (MiniSat+)

Pseudo-Boolean (PB) Constraints

Pseudo-Boolean Constraint

Constraint of the form $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \Box r$

- $\Box$ is one of $<$, $\leq$, $=$, $>$, or $\geq$
- variables $x_i \in \{0, 1\}$
- integer coefficients c_i

[1] Een and Sorensson, JSAT, 2006 (MiniSat+)

Pseudo-Boolean (PB) Constraints

Pseudo-Boolean Constraint

Constraint of the form $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \Box r$

- $\Box$ is one of $<$, $\leq$, $=$, $>$, or $\geq$
- variables $x_i \in \{0, 1\}$
- integer coefficients c_i
- generalization of clauses

[1] Een and Sorensson, JSAT, 2006 (MiniSat+)

Pseudo-Boolean (PB) Constraints

Pseudo-Boolean Constraint

Constraint of the form $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \,\Box\, r$

- $\Box$ is one of $<$, $\leq$, $=$, $>$, or $\geq$
- variables $x_i \in \{0, 1\}$
- integer coefficients c_i
- generalization of clauses
- can be encoded as CNF [1]

[1] Een and Sorensson, JSAT, 2006 (MiniSat+)

Pseudo-Boolean (PB) Constraints

Pseudo-Boolean Constraint

Constraint of the form $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \Box r$

- $\Box$ is one of $<$, $\leq$, $=$, $>$, or $\geq$
- variables $x_i \in \{0, 1\}$
- integer coefficients c_i
- generalization of clauses
- can be encoded as CNF [1]

Pseudo-Boolean Problem

Conjunction of PB constraints

[1] Een and Sorensson, JSAT, 2006 (MiniSat+)

Two Families of Solvers

SAT and ILP solvers and techniques can be applied!

Two Families of Solvers

SAT and ILP solvers and techniques can be applied!

Goal: improve SAT-based PB solving!

- Impressive performance improvements
- Flexibility, well-engineered interfaces
- Open source, easy to experiment with
- Works well for almost propositional instances

Incremental Translation to SAT

We do not have to encode *all* the constraints

Incremental Translation to SAT

We do not have to encode *all* the constraints

Satisfiable Formulas

Just enough constraints to find satisfying assignment

Incremental Translation to SAT

We do not have to encode *all* the constraints

Satisfiable Formulas

Just enough constraints to find satisfying assignment

Unsatisfiable Formulas

We may hit an unsatisfiable core quickly

Algorithm

procedure pb-sat(P)
    $C \leftarrow \{c \in P : c \text{ is a PB-clause}\}$
    $P \leftarrow \{p \in P : p \text{ is not a PB-clause}\}$
    while true **do**
        $A, U \leftarrow \text{sat}(C)$
        if $A = UNSAT$ **then return** $UNSAT$
        $P \leftarrow \text{simplify}(P, U)$
        if A satisfies P **then return** A
        $P' \leftarrow \{p \in P : p \text{ falsified by } A\}$
        if $P' = \emptyset$ **then** $P' \leftarrow \text{select}(P)$
        for all $p \in P'$ **do** $C \leftarrow C \wedge \text{translate}(p)$
        $P \leftarrow P \setminus P'$

Algorithm

procedure pb-sat(P)

$C \leftarrow \{c \in P : c$ is a PB-clause$\}$

$P \leftarrow \{p \in P : p$ is not a PB-clause$\}$

while true **do**

$A, U \leftarrow$ sat(C)

if $A = UNSAT$ **then return** $UNSAT$

$P \leftarrow$ simplify(P, U)

if A satisfies P **then return** A

$P' \leftarrow \{p \in P : p$ falsified by $A\}$

if $P' = \emptyset$ **then** $P' \leftarrow$ select(P)

for all $p \in P'$ **do** $C \leftarrow C \wedge$ translate(p)

$P \leftarrow P \setminus P'$

Algorithm

procedure pb-sat(P)
$C \leftarrow \{c \in P : c \text{ is a PB-clause}\}$
$P \leftarrow \{p \in P : p \text{ is not a PB-clause}\}$
while true **do**
$A, U \leftarrow \text{sat}(C)$
if $A = UNSAT$ **then return** $UNSAT$
$P \leftarrow \text{simplify}(P, U)$
if A satisfies P **then return** A
$P' \leftarrow \{p \in P : p \text{ falsified by } A\}$
if $P' = \emptyset$ **then** $P' \leftarrow \text{select}(P)$
for all $p \in P'$ **do** $C \leftarrow C \wedge \text{translate}(p)$
$P \leftarrow P \setminus P'$

Algorithm

procedure pb-sat(P)

$C \leftarrow \{c \in P : c \text{ is a PB-clause}\}$

$P \leftarrow \{p \in P : p \text{ is not a PB-clause}\}$

while true **do**

- $A, U \leftarrow \text{sat}(C)$
- **if** $A = \mathit{UNSAT}$ **then return** UNSAT
- $P \leftarrow \text{simplify}(P, U)$
- **if** A satisfies P **then return** A
- $P' \leftarrow \{p \in P : p \text{ falsified by } A\}$
- **if** $P' = \emptyset$ **then** $P' \leftarrow \text{select}(P)$
- **for all** $p \in P'$ **do** $C \leftarrow C \wedge \text{translate}(p)$
- $P \leftarrow P \setminus P'$

returns a *partial* assignment (A) and a set of units (U)

Algorithm

procedure pb-sat(P)

  $C \leftarrow \{c \in P : c$ is a PB-clause$\}$

  $P \leftarrow \{p \in P : p$ is not a PB-clause$\}$

  while true **do**

    $A, U \leftarrow \text{sat}(C)$

    if $A = \mathit{UNSAT}$ **then return** UNSAT

    $P \leftarrow \text{simplify}(P, U)$

    if A satisfies P **then return** A

    $P' \leftarrow \{p \in P : p$ falsified by $A\}$

    if $P' = \emptyset$ **then** $P' \leftarrow \text{select}(P)$

    for all $p \in P'$ **do** $C \leftarrow C \wedge \text{translate}(p)$

    $P \leftarrow P \setminus P'$

Algorithm

```
procedure pb-sat(P)
    C ← {c ∈ P : c is a PB-clause}
    P ← {p ∈ P : p is not a PB-clause}
    while true do
        A, U ← sat(C)
        if A = UNSAT then return UNSAT
        P ← simplify(P, U)
        if A satisfies P then return A
        P′ ← {p ∈ P : p falsified by A}
        if P′ = ∅ then P′ ← select(P)
        for all p ∈ P′ do C ← C ∧ translate(p)
        P ← P \ P′
```

Algorithm

procedure pb-sat(P)

  $C \leftarrow \{c \in P : c$ is a PB-clause$\}$

  $P \leftarrow \{p \in P : p$ is not a PB-clause$\}$

  while true **do**

    $A, U \leftarrow$ sat(C)

    if $A = UNSAT$ **then return** $UNSAT$

    $P \leftarrow$ simplify(P, U)

    if A satisfies P **then return** A

    $P' \leftarrow \{p \in P : p$ falsified by $A\}$

    if $P' = \emptyset$ **then** $P' \leftarrow$ select(P)

    for all $p \in P'$ **do** $C \leftarrow C \wedge$ translate(p)

    $P \leftarrow P \setminus P'$

Algorithm

procedure pb-sat(P)
  $C \leftarrow \{c \in P : c$ is a PB-clause$\}$
  $P \leftarrow \{p \in P : p$ is not a PB-clause$\}$
  while true **do**
    $A, U \leftarrow \text{sat}(C)$
    if $A = \mathit{UNSAT}$ **then return** UNSAT
    $P \leftarrow \text{simplify}(P, U)$
    if A satisfies P **then return** A
    $P' \leftarrow \{p \in P : p$ falsified by $A\}$
    if $P' = \emptyset$ **then** $P' \leftarrow \text{select}(P)$
    for all $p \in P'$ **do** $C \leftarrow C \wedge \text{translate}(p)$
    $P \leftarrow P \setminus P'$

Algorithm

procedure pb-sat(P)
  $C \leftarrow \{c \in P : c$ is a PB-clause$\}$
  $P \leftarrow \{p \in P : p$ is not a PB-clause$\}$
  while true **do**
    $A, U \leftarrow \text{sat}(C)$
    if $A = \mathit{UNSAT}$ **then return** UNSAT
    $P \leftarrow \text{simplify}(P, U)$
    if A satisfies P **then return** A
    $P' \leftarrow \{p \in P : p$ falsified by $A\}$
    if $P' = \emptyset$ **then** $P' \leftarrow \text{select}(P)$
    for all $p \in P'$ **do** $C \leftarrow C \wedge \text{translate}(p)$
    $P \leftarrow P \setminus P'$

Algorithm

procedure pb-sat(P)
  $C \leftarrow \{c \in P : c$ is a PB-clause$\}$
  $P \leftarrow \{p \in P : p$ is not a PB-clause$\}$
  while true **do**
    $A, U \leftarrow$ sat(C)
    if $A = UNSAT$ **then return** $UNSAT$
    $P \leftarrow$ simplify(P, U)
    if A satisfies P **then return** A
    $P' \leftarrow \{p \in P : p$ falsified by $A\}$
    if $P' = \emptyset$ **then** $P' \leftarrow$ select(P)
    for all $p \in P'$ **do** $C \leftarrow C \wedge$ translate(p)
    $P \leftarrow P \setminus P'$

Simplification

$$2x_1 + 2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

Simplification

$$2x_1 + 2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

Simplification

$$2x_1 + 2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

$$2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

Simplification

$$2x_1 + 2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

$$2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

Simplification

$$2x_1 + 2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

$$2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

$$x_3 + x_4 \geq 2 \qquad \neg x_1 \quad x_2$$

Simplification

$$2x_1 + 2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

$$2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

$$x_3 + x_4 \geq 2 \qquad \neg x_1 \quad x_2$$

Simplification

$$2x_1 + 2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

$$2x_2 + x_3 + x_4 \geq 4 \qquad \neg x_1 \quad x_2$$

$$x_3 + x_4 \geq 2 \qquad \neg x_1 \quad x_2$$

$$\neg x_1 \quad x_2 \quad x_3 \quad x_4$$

Algorithm

procedure pb-sat(P)

$C \leftarrow \{c \in P : c$ is a PB-clause$\}$

$P \leftarrow \{p \in P : p$ is not a PB-clause$\}$

while true **do**

$A, U \leftarrow \text{sat}(C)$

if $A = \mathit{UNSAT}$ **then return** UNSAT

$P \leftarrow \text{simplify}(P, U)$

if A satisfies P **then return** A

$P' \leftarrow \{p \in P : p$ falsified by $A\}$

if $P' = \emptyset$ **then** $P' \leftarrow \text{select}(P)$

for all $p \in P'$ **do** $C \leftarrow C \wedge \text{translate}(p)$

$P \leftarrow P \setminus P'$

Extracting More Units

$$A \models C$$

Extracting More Units

$$\{ \quad x, \quad \neg y, \quad \neg z, \quad w, \quad \ldots\} \quad \models C$$

Extracting More Units

$$\{ \quad x, \quad \neg y, \quad \neg z, \quad w, \quad \ldots\} \models C$$

U does not say anything about x, y, z, or w!

Extracting More Units

$$\{ \quad x, \quad \neg y, \quad \neg z, \quad w, \quad \ldots\} \quad \models C$$

$$\begin{aligned} \text{candidates} &: \quad \{x,\ \neg y,\ \neg z,\ w,\ \ldots\} \\ \text{units} &: \quad U \end{aligned}$$

Extracting More Units

$$\{ \quad x, \quad \neg y, \quad \neg z, \quad w, \quad \ldots\} \models C$$

candidates : $\{x, \neg y, \neg z, w, \ldots\}$ $\quad$ $\text{sat}(C \wedge \neg x)$?

units : U

Extracting More Units

$$\{\quad x,\ \ \neg y,\ \ \neg z,\ \ w,\ \ \ldots\} \models C$$
$$\{\ \neg x,\ \ \neg y,\ \ z,\ \ \neg w,\ \ \ldots\} \models C \wedge \neg x$$

candidates : $\{x,\ \neg y,\ \neg z,\ w,\ \ldots\}$ sat$(C \wedge \neg x)$?

units : U

Extracting More Units

$$\{\ x,\ \neg y,\ \neg z,\ w,\ \ldots\} \models C$$
$$\{\ \neg x,\ \neg y,\ z,\ \neg w,\ \ldots\} \models C \wedge \neg x$$

$$\text{candidates}: \quad \{x,\ \neg y,\ \neg z,\ w,\ \ldots\}$$
$$\text{units}: \quad U$$

Extracting More Units

$$\{ \quad x, \quad \neg y, \quad \neg z, \quad w, \quad \ldots\} \quad \models C$$

$$\{ \quad \neg x, \quad \neg y, \quad z, \quad \neg w, \quad \ldots\} \quad \models C \wedge \neg x$$

$$\text{candidates}: \quad \{\neg y, \ \ldots\}$$

$$\text{units}: \quad U$$

Extracting More Units

$$\{\quad x, \quad \neg y, \quad \neg z, \quad w, \quad \ldots\} \quad \models C$$
$$\{\quad \neg x, \quad \neg y, \quad z, \quad \neg w, \quad \ldots\} \quad \models C \wedge \neg x$$

candidates : $\{\neg y, \ \ldots\}$ sat$(C \wedge y)$?

units : U

Extracting More Units

$$\begin{array}{lccccc}\{ & x, & \neg y, & \neg z, & w, & \ldots\} & \models C \\ \{ & \neg x, & \neg y, & z, & \neg w, & \ldots\} & \models C \wedge \neg x \\ & & & & & & \models \neg(C \wedge y)\end{array}$$

candidates : $\{\neg y, \ \ldots\}$ $\quad$ $\text{sat}(C \wedge y)$?

units : U

Extracting More Units

$$\begin{array}{llllll} \{ & x, & \neg y, & \neg z, & w, & \ldots\} & \models C \\ \{ & \neg x, & \neg y, & z, & \neg w, & \ldots\} & \models C \wedge \neg x \\ & & & & & & \models \neg(C \wedge y) \end{array}$$

$$\begin{aligned} \text{candidates}: &\quad \{\ldots\} \\ \text{units}: &\quad U \cup \{\neg y\} \end{aligned}$$

Extracting More Units

$$\begin{array}{llllll} \{ & x, & \neg y, & \neg z, & w, & \ldots\} & \models C \\ \{ & \neg x, & \neg y, & z, & \neg w, & \ldots\} & \models C \wedge \neg x \\ & & & & & & \models \neg(C \wedge y) \end{array}$$

$$\begin{aligned} \text{candidates}: &\quad \{\ldots\} \\ \text{units}: &\quad U \cup \{\neg y\} \end{aligned}$$

SAT queries cost! limit a resource (*e.g.*, decisions)

Experiments: Industrial Instances [1]

MiniSat+

- Default configuration takes more than a day
- BDDs: blow-up with 96GB of RAM
- Human intervention: 91 minutes (11 minutes for SAT solving)

[1] CAV 2011

Experiments: Industrial Instances [1]

MiniSat+

- Default configuration takes more than a day
- BDDs: blow-up with 96GB of RAM
- Human intervention: 91 minutes (11 minutes for SAT solving)

PB-SAT

- 19 seconds; 9 seconds for SAT solving (PicoSAT [2])
- Only BDDs for the translation
- less than 2GB of RAM

[1] CAV 2011
[2] Thanks Armin!

Conclusions

We have extended the reach of SAT-based approaches to pseudo-Boolean solving.

Thank you!

Questions?

PB Competition Benchmarks

486 Decision Instances

	bsolo
solved	430

PB Competition Benchmarks

486 Decision Instances

	bsolo	PB-SAT
solved	430	402

PB Competition Benchmarks

486 Decision Instances

	bsolo	PB-SAT
solved	430	402

A Twist: Linear Relaxations

procedure pb-sat$^R(P)$
  if the relaxation of P is infeasible **then**
    return *UNSAT*
  else
    return pb-sat(P)

PB Competition Benchmarks

486 Decision Instances

	bsolo	PB-SAT
solved	430	402

A Twist: Linear Relaxations

procedure pb-sat$^R(P)$
  if the relaxation of P is infeasible **then**
    return *UNSAT*
  else
    return pb-sat(P)

variables in $[0, 1]$; simplex

PB Competition Benchmarks

486 Decision Instances

	bsolo	PB-SAT
solved	430	402

A Twist: Linear Relaxations

procedure pb-sat$^R(P)$
  if the relaxation of P is infeasible **then**
    return *UNSAT*
  else
    return pb-sat(P)

PB Competition Benchmarks

486 Decision Instances

	bsolo	PB-SAT	PB-SAT^R
solved	430	402	431

A Twist: Linear Relaxations

procedure pb-sat$^R(P)$
    if the relaxation of P is infeasible **then**
        return *UNSAT*
    else
        return pb-sat(P)

Experiments: PB Competition

486 Decision Problems

	CPLEX	bsolo	wbo	SAT4J	MS+[2]	**PB-SAT**	VPS[3]
solved	416	430	397	398	399	**402**	465
avg. time	135.8	38.0	70.3	67.8	83.1	**67.7**	-

939 Optimization Problems

	CPLEX	bsolo	wbo	SAT4J	**PB-SAT**	VPS
solved	676	580	579	542	**540**	792
avg. time	30.8	50.2	48.8	25.7	**81.3**	-

[2] with PicoSAT as the SAT solver

[3] Virtual Portfolio Solver

Extracting More Units: The Algorithm

procedure more-units(C)
    $A, U \leftarrow$ sat(C)
    if $A = \mathit{UNSAT}$ **then return** UNSAT
    $\alpha \leftarrow \{A\}$
    for all $l \in U$ **do** $C \leftarrow C \wedge l$
    for all variables v s.t. $v \notin U \wedge \neg v \notin U$ **do**
        if $\forall A_1, A_2 \in \alpha : A_1(v) = A_2(v)$ **then**
            $l \leftarrow$ polarity(A', v) for some $A' \in \alpha$
            $B \leftarrow$ sat-limited($C \wedge \neg l, R$)
            if $B = \mathit{UNSAT}$ **then**
                $U \leftarrow U \cup \{l\}$
                $C \leftarrow C \wedge l$
            else $\alpha \leftarrow \alpha \cup \{B\}$
    return pick(α), U