Scheduling and SAT

Emmanuel Hebrard

LAAS-CNRS

Toulouse

Outline

1. Introduction
2. Scheduling and SAT Encoding
3. Scheduling and SAT Heuristics
4. Scheduling and SAT Hybrids
5. Conclusion

Outline

Scheduling with Boolean Satisfiability

Scheduling with Boolean Satisfiability

- Number of hits for the Google query "Scheduling problem" with ...

Scheduling with Boolean Satisfiability

- Number of hits for the Google query "Scheduling problem" with ...

130,000

"Mixed Integer Programming" OR "Integer Linear Programming"

"Constraint Programming"

"Boolean Satisfiability"

Scheduling with Boolean Satisfiability

- Number of hits for the Google query "Scheduling problem" with ...

130,000

"Mixed Integer Programming"

OR

"Integer Linear Programming"

60,000

"Constraint Programming"

"Boolean Satisfiability"

Scheduling with Boolean Satisfiability

- Number of hits for the Google query "Scheduling problem" with ...

130,000

"Mixed Integer Programming"

OR

"Integer Linear Programming"

60,000

"Constraint Programming"

21,000

"Boolean Satisfiability"

Scheduling with Boolean Satisfiability

- Important theoretical results
  - [Cook-Levin] theorem: "First" NP-complete problem
  - [Schaefer]'s dichotomy theorem
- Efficient algorithms (CDCL)
- Successful in Circuit design, Model checking, Planning, ...

Scheduling with Boolean Satisfiability

- Important theoretical results
  - [Cook-Levin] theorem: "First" NP-complete problem
  - [Schaefer]'s dichotomy theorem
- Efficient algorithms (CDCL)
- Successful in Circuit design, Model checking, Planning, ...

Association of scheduling and SAT not as natural as MIP or CP

Scheduling with Boolean Satisfiability

- Important theoretical results
  - [Cook-Levin] theorem: “First” NP-complete problem
  - [Schaefer]’s dichotomy theorem
- Efficient algorithms (CDCL)
- Successful in Circuit design, Model checking, Planning, ...

Association of scheduling and SAT not as natural as MIP or CP

- Somewhat counter-intuitive (Boolean vs. Range, logical operator)

Scheduling with Boolean Satisfiability

- Important theoretical results
  - [Cook-Levin] theorem: "First" NP-complete problem
  - [Schaefer]'s dichotomy theorem
- Efficient algorithms (CDCL)
- Successful in Circuit design, Model checking, Planning, ...

Association of scheduling and SAT not as natural as MIP or CP

- Somewhat counter-intuitive (Boolean vs. Range, logical operator)
  - Apparent issue, the numerical aspect can often be avoided

Scheduling with Boolean Satisfiability

- Important theoretical results
  - [Cook-Levin] theorem: "First" NP-complete problem
  - [Schaefer]'s dichotomy theorem
- Efficient algorithms (CDCL)
- Successful in Circuit design, Model checking, Planning, ...

Association of scheduling and SAT not as natural as MIP or CP

- Somewhat counter-intuitive (Boolean vs. Range, logical operator)
  - Apparent issue, the numerical aspect can often be avoided
- Efficiency? SAT Solvers have not always been good

Scheduling with Boolean Satisfiability

- Important theoretical results
  - [Cook-Levin] theorem: “First” NP-complete problem
  - [Schaefer]'s dichotomy theorem
- Efficient algorithms (CDCL)
- Successful in Circuit design, Model checking, Planning, ...

Association of scheduling and SAT not as natural as MIP or CP

- Somewhat counter-intuitive (Boolean vs. Range, logical operator)
  - Apparent issue, the numerical aspect can often be avoided
- Efficiency? SAT Solvers have not always been good
  - They have made huge progress in the past 10 years

Scheduling with Boolean Satisfiability

- Important theoretical results
  - [Cook-Levin] theorem: “First” NP-complete problem
  - [Schaefer]'s dichotomy theorem
- Efficient algorithms (CDCL)
- Successful in Circuit design, Model checking, Planning, ...

Association of scheduling and SAT not as natural as MIP or CP

- Somewhat counter-intuitive (Boolean vs. Range, logical operator)
  - Apparent issue, the numerical aspect can often be avoided
- Efficiency? SAT Solvers have not always been good
  - They have made huge progress in the past 10 years

- All the approaches discussed here were developed in the last 5 years

Scheduling with Boolean Satisfiability

- Important theoretical results
  - [Cook-Levin] theorem: “First” NP-complete problem
  - [Schaefer]’s dichotomy theorem
- Efficient algorithms (CDCL)
- Successful in Circuit design, Model checking, Planning, ...

Association of scheduling and SAT not as natural as MIP or CP

- Somewhat counter-intuitive (Boolean vs. Range, logical operator)
  - Apparent issue, the numerical aspect can often be avoided
- Efficiency? SAT Solvers have not always been good
  - They have made huge progress in the past 10 years

- All the approaches discussed here were developed in the last 5 years
- Recent progress in SAT algorithms opens new research opportunities

Scheduling Problems

Terminology

- Tasks (preemptive, non-preemptive)
- Resources (disjunctive, cumulative, reservoir,...)
- Objectives (makespan, tardiness, flow time,...)
- Side constraints (precedence, time windows, time lags,...)

Scheduling Problems

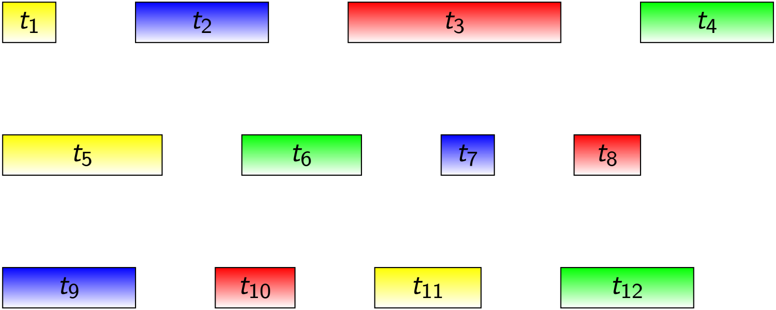
Terminology

- Tasks (preemptive, non-preemptive)
- Resources (disjunctive, cumulative, reservoir,...)
- Objectives (makespan, tardiness, flow time,...)
- Side constraints (precedence, time windows, time lags,...)

Tip of the iceberg

- SAT-based methods have been applied to a very small subset scheduling problems.
  - Minimization of makespan for non-preemptive tasks and disjunctive resources

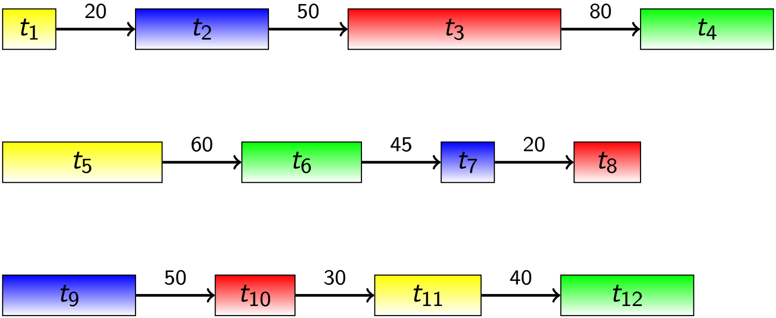
Jobshop Scheduling Problem



Problem description

- A set of non-preemptive tasks

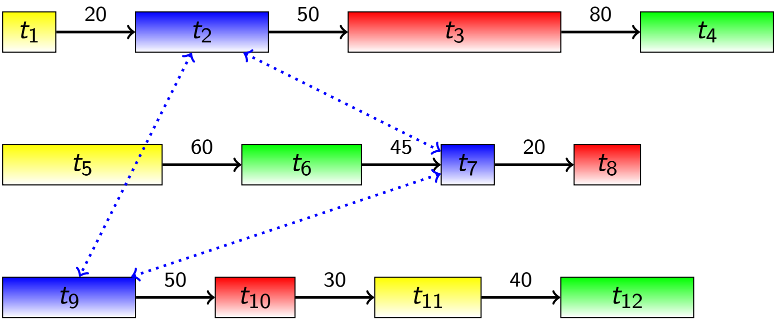
Jobshop Scheduling Problem



Problem description

- A set of non-preemptive tasks
- Organized in jobs (sequences)

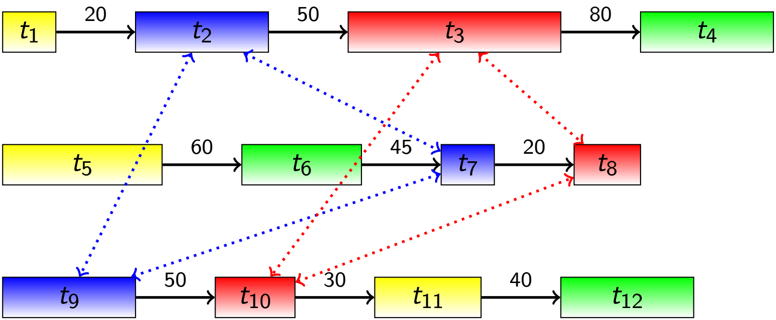
Jobshop Scheduling Problem



Problem description

- A set of non-preemptive tasks
- Organized in jobs (sequences)
- Requiring one of m disjunctive resources

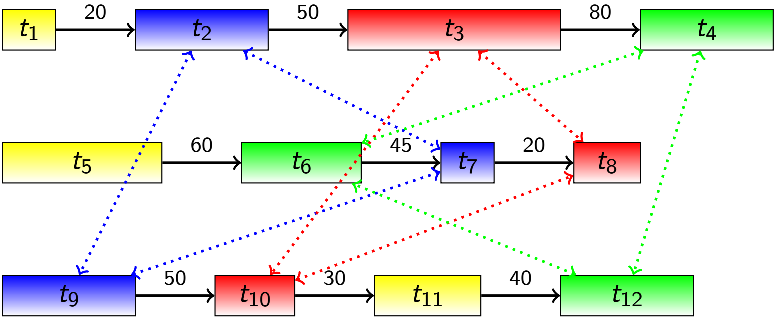
Jobshop Scheduling Problem



Problem description

- A set of non-preemptive tasks
- Organized in jobs (sequences)
- Requiring one of m disjunctive resources

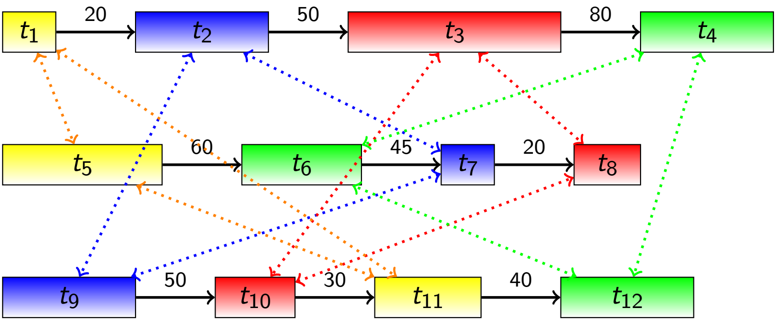
Jobshop Scheduling Problem



Problem description

- A set of non-preemptive tasks
- Organized in jobs (sequences)
- Requiring one of m disjunctive resources

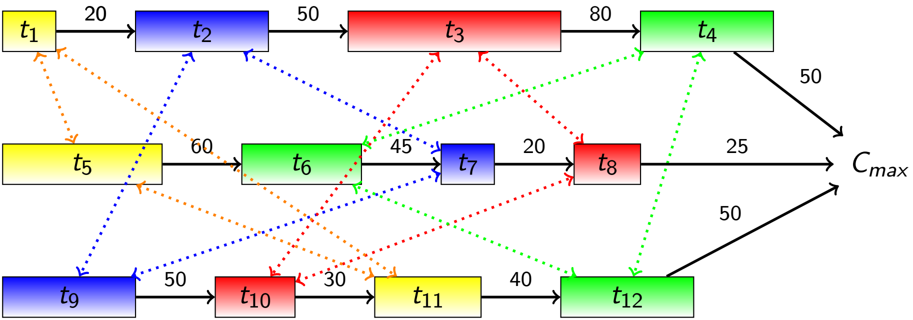
Jobshop Scheduling Problem



Problem description

- A set of non-preemptive tasks
- Organized in jobs (sequences)
- Requiring one of m disjunctive resources

Jobshop Scheduling Problem



Problem description

- A set of non-preemptive tasks
- Organized in jobs (sequences)
- Requiring one of m disjunctive resources
- Objective: minimize the total duration (C_{max})

Boolean Satisfiability (SAT)

Problem

- Boolean variables (atoms)
- Propositional logic formula (often CNF)

Boolean Satisfiability (SAT)

Problem

- Boolean variables (atoms)
- Propositional logic formula (often CNF)
- Literals: $a, \overline{a}$

Boolean Satisfiability (SAT)

Problem

- Boolean variables (atoms)
- Propositional logic formula (often CNF)
- Literals: $a, \overline{a}$
- Clauses: $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{b})$, $(b \vee \overline{c} \vee g)$

Boolean Satisfiability (SAT)

Problem

- Boolean variables (atoms)
- Propositional logic formula (often CNF)
- Literals: $a, \overline{a}$
- Clauses: $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{b})$, $(b \vee \overline{c} \vee g)$
- Solution: assignment of atoms satisfying all clauses

Boolean Satisfiability (SAT)

Problem

- Boolean variables (atoms)
- Propositional logic formula (often CNF)
- Literals: $a, \overline{a}$
- Clauses: $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{b})$, $(b \vee \overline{c} \vee g)$
- Solution: assignment of atoms satisfying all clauses

Algorithms

Boolean Satisfiability (SAT)

Problem

- Boolean variables (atoms)
- Propositional logic formula (often CNF)
- Literals: $a, \overline{a}$
- Clauses: $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{b})$, $(b \vee \overline{c} \vee g)$
- Solution: assignment of atoms satisfying all clauses

Algorithms

- Stochastic local search (GSAT, WalkSat,...)

Boolean Satisfiability (SAT)

Problem

- Boolean variables (atoms)
- Propositional logic formula (often CNF)
- Literals: $a, \overline{a}$
- Clauses: $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{b})$, $(b \vee \overline{c} \vee g)$
- Solution: assignment of atoms satisfying all clauses

Algorithms

- Stochastic local search (GSAT, WalkSat,...)
- Survey propagation

Boolean Satisfiability (SAT)

Problem

- Boolean variables (atoms)
- Propositional logic formula (often CNF)
- Literals: $a, \overline{a}$
- Clauses: $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{b})$, $(b \vee \overline{c} \vee g)$
- Solution: assignment of atoms satisfying all clauses

Algorithms

- Stochastic local search (GSAT, WalkSat,...)
- Survey propagation
- DPLL: Tree search + Unit propagation

Boolean Satisfiability (SAT)

Problem

- Boolean variables (atoms)
- Propositional logic formula (often CNF)
- Literals: $a, \overline{a}$
- Clauses: $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{f} \vee g)$, $(\overline{a} \vee \overline{b})$, $(b \vee \overline{c} \vee g)$
- Solution: assignment of atoms satisfying all clauses

Algorithms

- Stochastic local search (GSAT, WalkSat,...)
- Survey propagation
- DPLL: Tree search + Unit propagation
- CDCL: Conflict Driven Clause learning

Conflict Driven Clause Leaning (CDCL)

Conflict Driven Clause Leaning (CDCL)

“Evolved” from DPLL

- Turning point: clause learning ([GRASP] then [Chaff])
  - First SAT-Solver competition in 2002

Conflict Driven Clause Leaning (CDCL)

"Evolved" from DPLL

- Turning point: clause learning ([GRASP] then [Chaff])
  - First SAT-Solver competition in 2002

- Dive in the "search tree" (make decisions)
  - Unit propagate: if a must be true, then $\overline{a}$ cannot satisfy a clause

Conflict Driven Clause Leaning (CDCL)

"Evolved" from DPLL

- Turning point: clause learning ([GRASP] then [Chaff])
  - First SAT-Solver competition in 2002

- Dive in the "search tree" (make decisions)
  - Unit propagate: if a must be true, then $\overline{a}$ cannot satisfy a clause
  - $\overline{a} \vee b \vee \overline{c}$ effectively becomes $b \vee \overline{c}$
    - ★ continue until a fix point is reached

Conflict Driven Clause Leaning (CDCL)

"Evolved" from DPLL

- Turning point: clause learning ([GRASP] then [Chaff])
  - First SAT-Solver competition in 2002

- Dive in the "search tree" (make decisions)
  - Unit propagate: if a must be true, then $\overline{a}$ cannot satisfy a clause
  - $\overline{a} \vee b \vee \overline{c}$ effectively becomes $b \vee \overline{c}$
    - ★ continue until a fix point is reached
- Until reaching a conflicts (dead-end)
  - Extract a learned clause
  - Backjump several levels and unit-propagate the learned clause

Conflict Driven Clause Leaning (CDCL)

"Evolved" from DPLL

- Turning point: clause learning ([GRASP] then [Chaff])
  - First SAT-Solver competition in 2002

- Dive in the "search tree" (make decisions)
  - Unit propagate: if a must be true, then $\overline{a}$ cannot satisfy a clause
  - $\overline{a} \vee b \vee \overline{c}$ effectively becomes $b \vee \overline{c}$
    - ⋆ continue until a fix point is reached
- Until reaching a conflicts (dead-end)
  - Extract a learned clause
  - Backjump several levels and unit-propagate the learned clause
- Adaptive branching heuristics (weight conflicting literals)

Conflict Driven Clause Leaning (CDCL)

"Evolved" from DPLL

- Turning point: clause learning ([GRASP] then [Chaff])
  - First SAT-Solver competition in 2002

- Dive in the "search tree" (make decisions)
  - Unit propagate: if a must be true, then $\overline{a}$ cannot satisfy a clause
  - $\overline{a} \vee b \vee \overline{c}$ effectively becomes $b \vee \overline{c}$
    - ★ continue until a fix point is reached
- Until reaching a conflicts (dead-end)
  - Extract a learned clause
  - Backjump several levels and unit-propagate the learned clause
- Adaptive branching heuristics (weight conflicting literals)
- And also: restart, simplify the clause base, forget clauses, etc.

CDCL: Example

	f		

$\overline{a} \vee \overline{f} \vee g$

$\overline{a} \vee \overline{b} \vee \overline{h}$

$a \vee c$

$a \vee \overline{i} \vee \overline{l}$

$a \vee \overline{k} \vee \overline{j}$

$b \vee d$

$b \vee g \vee \overline{n}$

$b \vee \overline{f} \vee n \vee k$

$\overline{c} \vee k$

$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$

$c \vee l$

$d \vee \overline{k} \vee l$

$d \vee \overline{g} \vee l$

$\overline{g} \vee n \vee o$

$h \vee \overline{o} \vee \overline{j} \vee n$

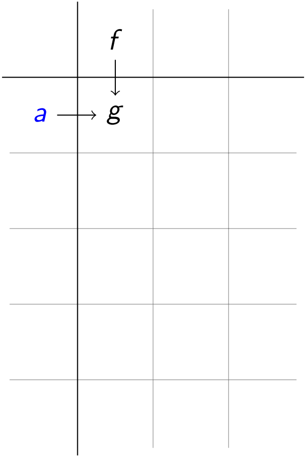
$\overline{i} \vee j$

$\overline{d} \vee \overline{l} \vee \overline{m}$

$\overline{e} \vee m \vee \overline{n}$

$\overline{f} \vee h \vee i$

CDCL: Example



$\overline{a} \vee \overline{f} \vee g$

$\overline{a} \vee \overline{b} \vee \overline{h}$

$a \vee c$

$a \vee \overline{i} \vee \overline{l}$

$a \vee \overline{k} \vee \overline{j}$

$b \vee d$

$b \vee g \vee \overline{n}$

$b \vee \overline{f} \vee n \vee k$

$\overline{c} \vee k$

$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$

$c \vee l$

$d \vee \overline{k} \vee l$

$d \vee \overline{g} \vee l$

$\overline{g} \vee n \vee o$

$h \vee \overline{o} \vee \overline{j} \vee n$

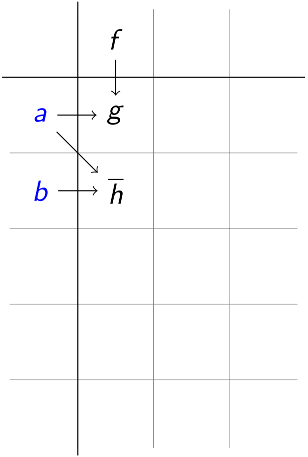
$\overline{i} \vee j$

$\overline{d} \vee \overline{l} \vee \overline{m}$

$\overline{e} \vee m \vee \overline{n}$

$\overline{f} \vee h \vee i$

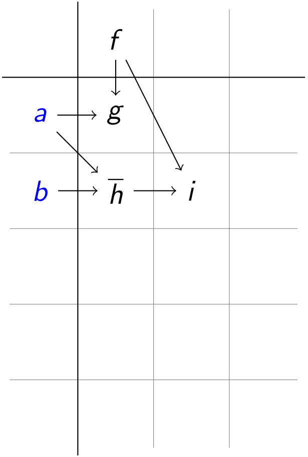
CDCL: Example



$\bar{a} \vee \bar{f} \vee g$
$\bar{a} \vee \bar{b} \vee \bar{h}$
$a \vee c$
$a \vee \bar{i} \vee \bar{l}$
$a \vee \bar{k} \vee \bar{j}$
$b \vee d$
$b \vee g \vee \bar{n}$
$b \vee \bar{f} \vee n \vee k$
$\bar{c} \vee k$
$\bar{c} \vee \bar{k} \vee \bar{i} \vee l$

$c \vee h \vee n \vee \bar{m}$
$c \vee l$
$d \vee \bar{k} \vee l$
$d \vee \bar{g} \vee l$
$\bar{g} \vee n \vee o$
$h \vee \bar{o} \vee \bar{j} \vee n$
$\bar{i} \vee j$
$\bar{d} \vee \bar{l} \vee \bar{m}$
$\bar{e} \vee m \vee \bar{n}$
$\bar{f} \vee h \vee i$

CDCL: Example



$\overline{a} \vee \overline{f} \vee g$

$\overline{a} \vee \overline{b} \vee \overline{h}$

$a \vee c$

$a \vee \overline{i} \vee \overline{l}$

$a \vee \overline{k} \vee \overline{j}$

$b \vee d$

$b \vee g \vee \overline{n}$

$b \vee \overline{f} \vee n \vee k$

$\overline{c} \vee k$

$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$

$c \vee l$

$d \vee \overline{k} \vee l$

$d \vee \overline{g} \vee l$

$\overline{g} \vee n \vee o$

$h \vee \overline{o} \vee \overline{j} \vee n$

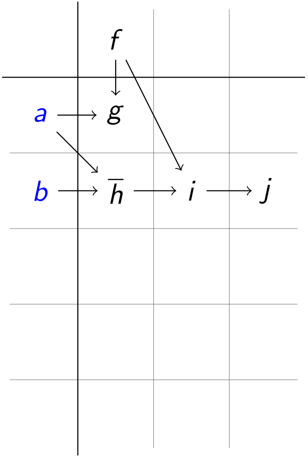
$\overline{i} \vee j$

$\overline{d} \vee \overline{l} \vee \overline{m}$

$\overline{e} \vee m \vee \overline{n}$

$\overline{f} \vee h \vee i$

CDCL: Example



$\overline{a} \lor \overline{f} \lor g$

$\overline{a} \lor \overline{b} \lor \overline{h}$

$a \lor c$

$a \lor \overline{i} \lor \overline{l}$

$a \lor \overline{k} \lor \overline{j}$

$b \lor d$

$b \lor g \lor \overline{n}$

$b \lor \overline{f} \lor n \lor k$

$\overline{c} \lor k$

$\overline{c} \lor \overline{k} \lor \overline{i} \lor l$

$c \lor h \lor n \lor \overline{m}$

$c \lor l$

$d \lor \overline{k} \lor l$

$d \lor \overline{g} \lor l$

$\overline{g} \lor n \lor o$

$h \lor \overline{o} \lor \overline{j} \lor n$

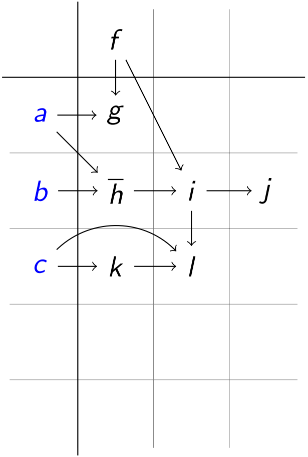
$\overline{i} \lor j$

$\overline{d} \lor \overline{l} \lor \overline{m}$

$\overline{e} \lor m \lor \overline{n}$

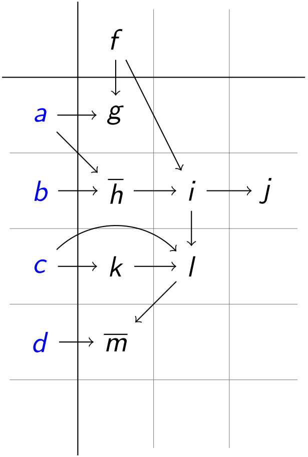
$\overline{f} \lor h \lor i$

CDCL: Example



$\overline{a} \vee \overline{f} \vee g$	$c \vee h \vee n \vee \overline{m}$
$\overline{a} \vee \overline{b} \vee \overline{h}$	$c \vee l$
$a \vee c$	$d \vee \overline{k} \vee l$
$a \vee \overline{i} \vee \overline{l}$	$d \vee \overline{g} \vee l$
$a \vee \overline{k} \vee \overline{j}$	$\overline{g} \vee n \vee o$
$b \vee d$	$h \vee \overline{o} \vee \overline{j} \vee n$
$b \vee g \vee \overline{n}$	$\overline{i} \vee j$
$b \vee \overline{f} \vee n \vee k$	$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{c} \vee k$	$\overline{e} \vee m \vee \overline{n}$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$	$\overline{f} \vee h \vee i$

CDCL: Example



$\bar{a} \lor \bar{f} \lor g$

$\bar{a} \lor \bar{b} \lor \bar{h}$

$a \lor c$

$a \lor \bar{i} \lor \bar{l}$

$a \lor \bar{k} \lor \bar{j}$

$b \lor d$

$b \lor g \lor \bar{n}$

$b \lor \bar{f} \lor n \lor k$

$\bar{c} \lor k$

$\bar{c} \lor \bar{k} \lor \bar{i} \lor l$

$c \lor h \lor n \lor \bar{m}$

$c \lor l$

$d \lor \bar{k} \lor l$

$d \lor \bar{g} \lor l$

$\bar{g} \lor n \lor o$

$h \lor \bar{o} \lor \bar{j} \lor n$

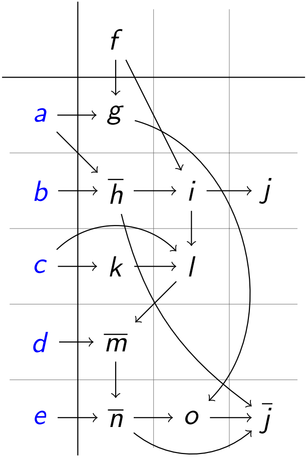
$\bar{i} \lor j$

$\bar{d} \lor \bar{l} \lor \bar{m}$

$\bar{e} \lor m \lor \bar{n}$

$\bar{f} \lor h \lor i$

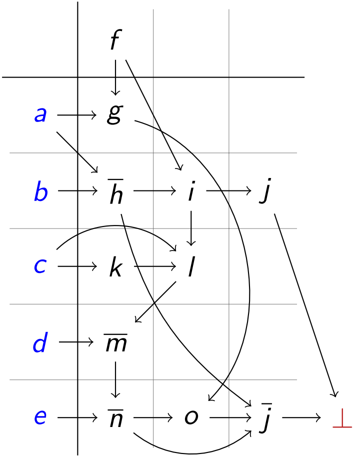
CDCL: Example



$\bar{a} \vee \bar{f} \vee g$
$\bar{a} \vee \bar{b} \vee \bar{h}$
$a \vee c$
$a \vee \bar{i} \vee \bar{l}$
$a \vee \bar{k} \vee \bar{j}$
$b \vee d$
$b \vee g \vee \bar{n}$
$b \vee \bar{f} \vee n \vee k$
$\bar{c} \vee k$
$\bar{c} \vee \bar{k} \vee \bar{i} \vee l$

$c \vee h \vee n \vee \bar{m}$
$c \vee l$
$d \vee \bar{k} \vee l$
$d \vee \bar{g} \vee l$
$\bar{g} \vee n \vee o$
$h \vee \bar{o} \vee \bar{j} \vee n$
$\bar{i} \vee j$
$\bar{d} \vee \bar{l} \vee \bar{m}$
$\bar{e} \vee m \vee \bar{n}$
$\bar{f} \vee h \vee i$

CDCL: Example



$\bar{a} \vee \bar{f} \vee g$

$\bar{a} \vee \bar{b} \vee \bar{h}$

$a \vee c$

$a \vee \bar{i} \vee \bar{l}$

$a \vee \bar{k} \vee \bar{j}$

$b \vee d$

$b \vee g \vee \bar{n}$

$b \vee \bar{f} \vee n \vee k$

$\bar{c} \vee k$

$\bar{c} \vee \bar{k} \vee \bar{i} \vee l$

$c \vee h \vee n \vee \bar{m}$

$c \vee l$

$d \vee \bar{k} \vee l$

$d \vee \bar{g} \vee l$

$\bar{g} \vee n \vee o$

$\boxed{h \vee \bar{o} \vee \bar{j} \vee n}$

$\bar{i} \vee j$

$\bar{d} \vee \bar{l} \vee \bar{m}$

$\bar{e} \vee m \vee \bar{n}$

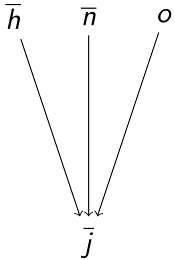
$\bar{f} \vee h \vee i$

CDCL: Example

$$
\begin{tikzcd}
\overline{h} \arrow[dr] & \overline{n} \arrow[d] & o \arrow[dl] \\
& \bar{j} &
\end{tikzcd}
$$

$$(h \vee \overline{o} \vee \bar{j} \vee n)$$

CDCL: Example



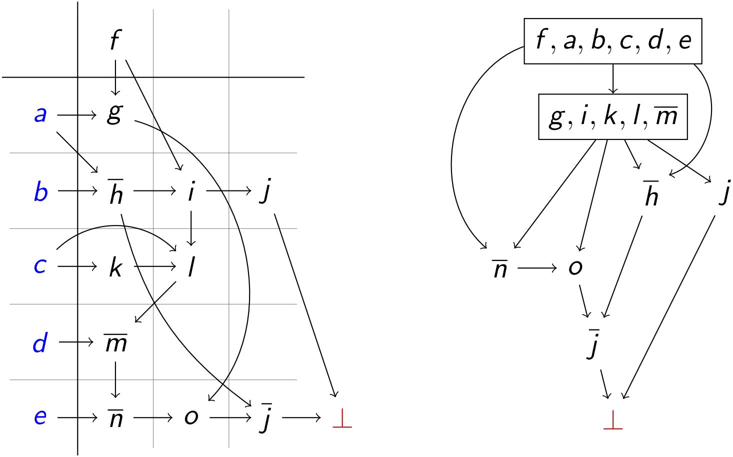
$$(h \vee \overline{o} \vee \bar{j} \vee n)$$

CDCL: Example

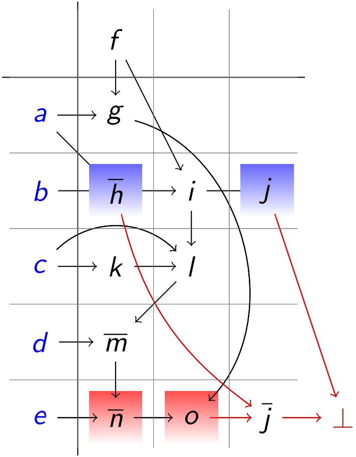
$$
\begin{tikzcd}
\overline{h} \arrow[dr] & \overline{n} \arrow[d] & o \arrow[dl] \\
& \bar{j} &
\end{tikzcd}
$$

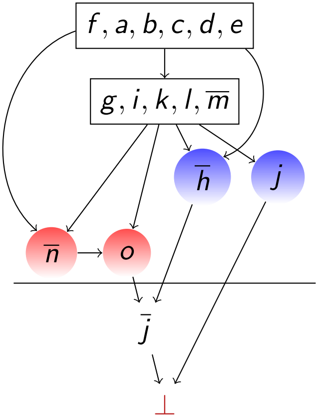
$$(h \vee \overline{o} \vee \bar{j} \vee n)$$
$$\equiv$$
$$(\overline{h} \wedge o \wedge \overline{n}) \rightarrow \bar{j}$$

CDCL: Example

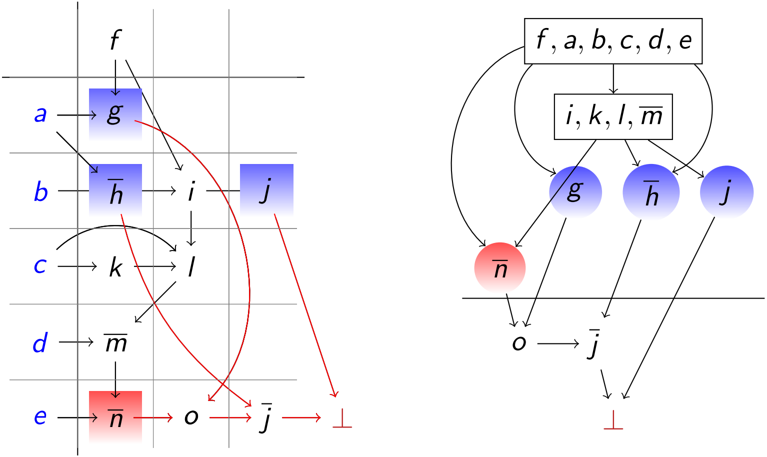


CDCL: Example

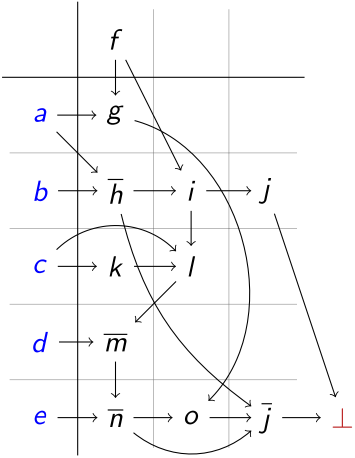




CDCL: Example



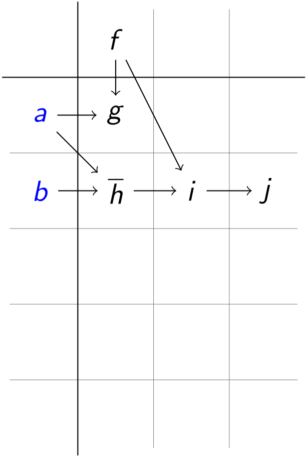
CDCL: Example



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

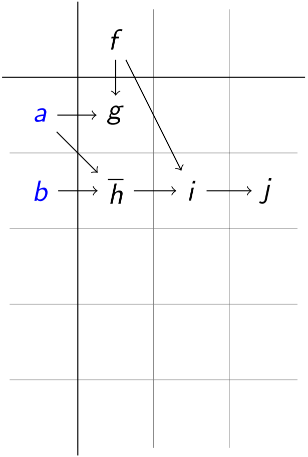
CDCL: Example



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

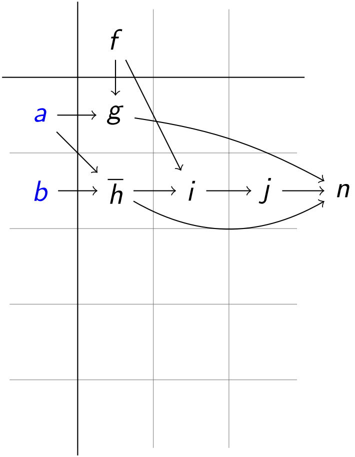
CDCL: Example



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$
$\boxed{\overline{g} \vee h \vee \overline{j} \vee n}$

CDCL: Example



$\overline{a} \lor \overline{f} \lor g$

$\overline{a} \lor \overline{b} \lor \overline{h}$

$a \lor c$

$a \lor \overline{i} \lor \overline{l}$

$a \lor \overline{k} \lor \overline{j}$

$b \lor d$

$b \lor g \lor \overline{n}$

$b \lor \overline{f} \lor n \lor k$

$\overline{c} \lor k$

$\overline{c} \lor \overline{k} \lor \overline{i} \lor l$

$c \lor h \lor n \lor \overline{m}$

$c \lor l$

$d \lor \overline{k} \lor l$

$d \lor \overline{g} \lor l$

$\overline{g} \lor n \lor o$

$h \lor \overline{o} \lor \overline{j} \lor n$

$\overline{i} \lor j$

$\overline{d} \lor \overline{l} \lor \overline{m}$

$\overline{e} \lor m \lor \overline{n}$

$\overline{f} \lor h \lor i$

$\boxed{\overline{g} \lor h \lor \overline{j} \lor n}$

Adaptive heuristics

- Variable State Independent Decaying Sum (VSIDS)
  - Idea ([Chaff]) weight literals in learned conflicts
  - Decay: favor newer weights

Adaptive heuristics

- Variable State Independent Decaying Sum (VSIDS)
  - Idea ([Chaff]) weight literals in learned conflicts
  - Decay: favor newer weights
- Weighted degree heuristic
  - On a failure: weight the constraint propagated last

Adaptive heuristics

- Variable State Independent Decaying Sum (VSIDS)
  - Idea ([Chaff]) weight literals in learned conflicts
  - Decay: favor newer weights
- Weighted degree heuristic
  - On a failure: weight the constraint propagated last
- Activity Based Search
  - On a succes: weight the variables whose domain has changed

Outline

1 Introduction

2 **Scheduling and SAT Encoding**
- Formulation into SAT
- Scheduling by encoding into SAT

3 Scheduling and SAT Heuristics

4 Scheduling and SAT Hybrids

5 Conclusion

CNF encoding

- The way we encode problems into SAT has a huge impact on efficiency
  - Encoding of Planning problems
  - Encoding of CSP (Direct, Log, AC-encoding)
  - Encoding of Pseudo-Boolean (Adder, Sorter)

CNF encoding

- The way we encode problems into SAT has a huge impact on efficiency
  - Encoding of Planning problems
  - Encoding of CSP (Direct, Log, AC-encoding)
  - Encoding of Pseudo-Boolean (Adder, Sorter)
- All encodings are based on CSP formulations
  - Some Boolean variables (e.g., relative orders of tasks)
  - Start time variables (Integer variables)

CNF encoding

- The way we encode problems into SAT has a huge impact on efficiency
  - Encoding of Planning problems
  - Encoding of CSP (Direct, Log, AC-encoding)
  - Encoding of Pseudo-Boolean (Adder, Sorter)
- All encodings are based on CSP formulations
  - Some Boolean variables (e.g., relative orders of tasks)
  - Start time variables (Integer variables)
- Integer variables and precedence constraints

Direct Encoding

Domain

- An atom i_v for each pair $(x_i, v \in D(x_i))$
  - $i_v \Leftrightarrow x_i = v$ $\quad \begin{array}{ll} x_i = 1: & 1000 \\ x_i = 2: & 0100 \\ x_i = 3: & 0010 \\ x_i = 4: & 0001 \end{array}$
- Must take at least a value: $i_1 \vee i_2 \vee \ldots \vee i_n$
- Must take at most one value: $\bigwedge_{v \neq w \in D(x_i)} \overline{i_v} \vee \overline{i_w}$

Direct Encoding

Domain

- An atom i_v for each pair $(x_i, v \in D(x_i))$
  - $i_v \Leftrightarrow x_i = v$ $\quad \begin{array}{ll} x_i = 1: & 1000 \\ x_i = 2: & 0100 \\ x_i = 3: & 0010 \\ x_i = 4: & 0001 \end{array}$
- Must take at least a value: $i_1 \vee i_2 \vee \ldots \vee i_n$
- Must take at most one value: $\bigwedge_{v \neq w \in D(x_i)} \overline{i_v} \vee \overline{i_w}$

Complexity

- $O(n^2)$ space: $n(n-1)/2$ binary clauses and one n-ary clause.
- There are different ways to encode the constraints.

Constraints: Tuple Encoding

Example of constraint: $x_i < x_j$

x_j \ x_i	1	2	3	4
1	$\overline{i_1} \vee \overline{j_1}$	$\overline{i_2} \vee \overline{j_1}$	$\overline{i_3} \vee \overline{j_1}$	$\overline{i_4} \vee \overline{j_1}$
2		$\overline{i_2} \vee \overline{j_2}$	$\overline{i_3} \vee \overline{j_2}$	$\overline{i_4} \vee \overline{j_2}$
3			$\overline{i_3} \vee \overline{j_3}$	$\overline{i_4} \vee \overline{j_3}$
4				$\overline{i_4} \vee \overline{j_3}$

Constraints: Tuple Encoding

Example of constraint: $x_i < x_j$

x_j \ x_i	1	2	3	4
1	$\overline{i_1} \vee \overline{j_1}$	$\overline{i_2} \vee \overline{j_1}$	$\overline{i_3} \vee \overline{j_1}$	$\overline{i_4} \vee \overline{j_1}$
2		$\overline{i_2} \vee \overline{j_2}$	$\overline{i_3} \vee \overline{j_2}$	$\overline{i_4} \vee \overline{j_2}$
3			$\overline{i_3} \vee \overline{j_3}$	$\overline{i_4} \vee \overline{j_3}$
4				$\overline{i_4} \vee \overline{j_3}$

Costly (in space) and weak (in propagation)

- $O(n^2)$ binary clauses.
- $\overline{i_4}(x_i \neq 4)$ and $\overline{j_1}(x_j \neq 1)$ are inconsistent, but not unit propagated.

Constraints: AC Encoding [Kasif 90]

Example of constraint: $x_i < x_j$

assignment	atom		support
$x_i = 1$	$\overline{i_1}$	$\vee$	$j_2 \vee j_3 \vee j_4$
$x_i = 2$	$\overline{i_2}$	$\vee$	$j_3 \vee j_4$
$x_i = 3$	$\overline{i_3}$	$\vee$	j_4
$x_i = 4$	$\overline{i_4}$	$\vee$	$\perp$

Constraints: AC Encoding [Kasif 90]

Example of constraint: $x_i < x_j$

assignment	atom		support
$x_i = 1$	$\overline{i_1}$	$\vee$	$j_2 \vee j_3 \vee j_4$
$x_i = 2$	$\overline{i_2}$	$\vee$	$j_3 \vee j_4$
$x_i = 3$	$\overline{i_3}$	$\vee$	j_4
$x_i = 4$	$\overline{i_4}$	$\vee$	$\perp$

Same space complexity, better propagation

- $O(n)$ n-ary clauses
- $\overline{i_4}(x_i \neq 4)$ and $\overline{j_1}(x_j \neq 1)$ are unit clauses.

Order Encoding [Crawford & Backer 94]

Domain

- An atom i_v for each pair $(x_i, v \in D(x_i))$
  - $i_v \Leftrightarrow x_i \leq v$ $\quad \begin{array}{ll} x_i = 1: & 1111 \\ x_i = 2: & 0111 \\ x_i = 3: & 0011 \\ x_i = 4: & 0001 \end{array}$
- Bound propagation:
  - If $x_i \leq v$ then $x_i \leq v + 1$
  - $\bigwedge_{v \in D(x_i)} \overline{i_v} \vee i_{v+1}$

Order Encoding [Crawford & Backer 94]

Domain

- An atom i_v for each pair $(x_i, v \in D(x_i))$
  - $i_v \Leftrightarrow x_i \leq v$
    $$\begin{aligned} x_i = 1&: \quad 1111 \\ x_i = 2&: \quad 0111 \\ x_i = 3&: \quad 0011 \\ x_i = 4&: \quad 0001 \end{aligned}$$
- Bound propagation:
  - If $x_i \leq v$ then $x_i \leq v + 1$
  - $\bigwedge_{v \in D(x_i)} \overline{i_v} \vee i_{v+1}$

Complexity

- $O(n)$ space ($n - 1$ binary clauses)

Constraints: BC Encoding

Example of constraint: $x_i < x_j$

relation	clause
$x_i > 0 \Rightarrow x_j > 1$	$\perp \vee \overline{j_1}$
$x_i > 1 \Rightarrow x_j > 2$	$i_1 \vee \overline{j_2}$
$x_i > 2 \Rightarrow x_j > 3$	$i_2 \vee \overline{j_3}$
$x_i > 3 \Rightarrow x_j > 4$	$i_3 \vee \perp$

Constraints: BC Encoding

Example of constraint: $x_i < x_j$

relation	clause
$x_i > 0 \Rightarrow x_j > 1$	$\perp \vee \overline{j_1}$
$x_i > 1 \Rightarrow x_j > 2$	$i_1 \vee \overline{j_2}$
$x_i > 2 \Rightarrow x_j > 3$	$i_2 \vee \overline{j_3}$
$x_i > 3 \Rightarrow x_j > 4$	$i_3 \vee \perp$

Better complexity and same propagation on some linear constraints

- $O(n)$ space (n binary clauses)
- $i_3 (x_i \leq 3)$ and $\overline{j_1} (x_j > 1)$ are unit clauses.

Log Encoding [Walsh 00]

Domain

- An atom i_k for each value in $[1, \ldots, \lfloor \log_2 ub \rfloor]$ (assuming $D(x_)) = [0, \ldots, ub]$)
  - $\sum_{k=1}^{ub} 2^k * i_k = v \Leftrightarrow x_i = v$

$x_i = 0$:	00
$x_i = 1$:	01
$x_i = 2$:	10
$x_i = 3$:	11

- For interval domains, no need for extra clauses

Log Encoding [Walsh 00]

Domain

- An atom i_k for each value in $[1, \ldots, \lfloor \log_2 ub \rfloor]$ (assuming $D(x_) = [0, \ldots, ub]$)
  - $\sum_{k=1}^{ub} 2^k * i_k = v \Leftrightarrow x_i = v$

$$
\begin{array}{ll}
x_i = 0: & 00 \\
x_i = 1: & 01 \\
x_i = 2: & 10 \\
x_i = 3: & 11
\end{array}
$$

- For interval domains, no need for extra clauses

Complexity

- $O(\log_2 n)$ space

Propagation

- Encoding constraints is trickier, and less powerful

Other Encodings

Many more!

- Mix of direct and order encoding [lazy-FD, Numberjack]
- Mix of AC and log encoding [Gavanelli 2007]
- Mix of order and log encoding [Sugar, Tamura et al. 2006]

Other Encodings

Many more!

- Mix of direct and order encoding [lazy-FD, Numberjack]
- Mix of AC and log encoding [Gavanelli 2007]
- Mix of order and log encoding [Sugar, Tamura et al. 2006]
  - Log encoding in a base B and order encoding inside a digit
  - Excellent results on scheduling benchmarks! (with CDCL solvers)

Order Encoding, Now and Then

Progress of SAT solvers

- From a few hundreds variables in the 90's to millions now

Order Encoding, Now and Then

Progress of SAT solvers

- From a few hundreds variables in the 90's to millions now

[Crawford & Backer 94]

- Instances from Sadeh, with 10 jobs, 5 operations each (45m cutoff)
- Tableau solved 90% of the instances (about 2 min when it did)

Order Encoding, Now and Then

Progress of SAT solvers

- From a few hundreds variables in the 90's to millions now

[Crawford & Backer 94]

- Instances from Sadeh, with 10 jobs, 5 operations each (45m cutoff)
- Tableau solved 90% of the instances (about 2 min when it did)

[Tamura, Tanjo & Banbara]

- Same instances used during the CSP Solver Competition
- Similar model, hardware of course incomparable, MiniSat

Order Encoding, Now and Then

Progress of SAT solvers

- From a few hundreds variables in the 90's to millions now

[Crawford & Backer 94]

- Instances from Sadeh, with 10 jobs, 5 operations each (45m cutoff)
- Tableau solved 90% of the instances (about 2 min when it did)

[Tamura, Tanjo & Banbara]

- Same instances used during the CSP Solver Competition
- Similar model, hardware of course incomparable, MiniSat
- The hardest instance requires a few 100s conflicts at the most

Closing the Open Shop

Instances

- [Gueret & Prins]: hard for local search, extremely easy for SAT/CP
- [Taillard]: Large, but relatively easy
- [Brucker]: Three open instances

Closing the Open Shop

Instances

- [Gueret & Prins]: hard for local search, extremely easy for SAT/CP
- [Taillard]: Large, but relatively easy
- [Brucker]: Three open instances

results

- All instances solved and proved optimal
  - The two hardest instances were decomposed into 120 subproblems, and required up to 13h to solve

Closing the Open Shop

Instances

- [Gueret & Prins]: hard for local search, extremely easy for SAT/CP
- [Taillard]: Large, but relatively easy
- [Brucker]: Three open instances

results

- All instances solved and proved optimal
  - The two hardest instances were decomposed into 120 subproblems, and required up to 13h to solve
- First approach to close the open shop!

Solving vs. Encoding

Solving vs. Encoding

- [Tamura et al.]'s encoding is better than order encoding

Solving vs. Encoding

- [Tamura et al.]'s encoding is better than order encoding
  - However, the huge difference with respect to [Crawford & Backer 94] is due to the solver

Solving vs. Encoding

- [Tamura et al.]'s encoding is better than order encoding
  - However, the huge difference with respect to [Crawford & Backer 94] is due to the solver
- It is now possible to efficiently solve some scheduling problem simply by formulating it as a CNF formula

Outline

A SAT-like Approach [Grimes & Hebrard 09]

- CSP Solver Competition: scheduling benchmarks

A SAT-like Approach [Grimes & Hebrard 09]

- CSP Solver Competition: scheduling benchmarks
  - Some hard instances
  - Generic format (XCSP), the notions of resource is lost, no global constraint

A SAT-like Approach [Grimes & Hebrard 09]

- CSP Solver Competition: scheduling benchmarks
  - Some hard instances
  - Generic format (XCSP), the notions of resource is lost, no global constraint
  - Yet many solvers solved them ([Sugar], [Choco], [Mistral])

A SAT-like Approach [Grimes & Hebrard 09]

- CSP Solver Competition: scheduling benchmarks
  - Some hard instances
  - Generic format (XCSP), the notions of resource is lost, no global constraint
  - Yet many solvers solved them ([Sugar], [Choco], [Mistral])
- Experiment with Weighted degree [Boussemart et al. 04]

A SAT-like Approach [Grimes & Hebrard 09]

- CSP Solver Competition: scheduling benchmarks
  - Some hard instances
  - Generic format (XCSP), the notions of resource is lost, no global constraint
  - Yet many solvers solved them ([Sugar], [Choco], [Mistral])
- Experiment with Weighted degree [Boussemart et al. 04]
  - Similar simple model in [Mistral], same observation [Grimes]

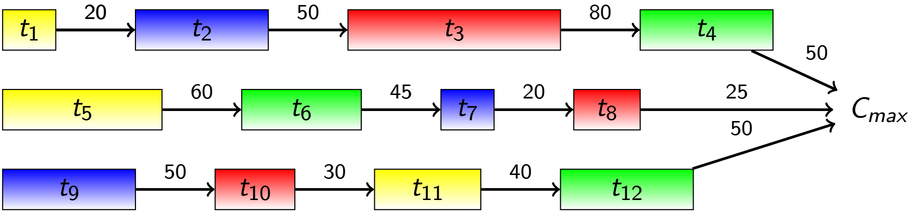
A SAT-like Approach [Grimes & Hebrard 09]

- CSP Solver Competition: scheduling benchmarks
  - Some hard instances
  - Generic format (XCSP), the notions of resource is lost, no global constraint
  - Yet many solvers solved them ([Sugar], [Choco], [Mistral])
- Experiment with Weighted degree [Boussemart et al. 04]
  - Similar simple model in [Mistral], same observation [Grimes]
  - Open shop instances closed by [Tamura et al.] can be solved to optimality in a few minutes

A SAT-like Approach [Grimes & Hebrard 09]

- CSP Solver Competition: scheduling benchmarks
  - Some hard instances
  - Generic format (XCSP), the notions of resource is lost, no global constraint
  - Yet many solvers solved them ([Sugar], [Choco], [Mistral])
- Experiment with Weighted degree [Boussemart et al. 04]
  - Similar simple model in [Mistral], same observation [Grimes]
  - Open shop instances closed by [Tamura et al.] can be solved to optimality in a few minutes
- Are adaptive heuristics all that we need to solve disjunctive scheduling problems?

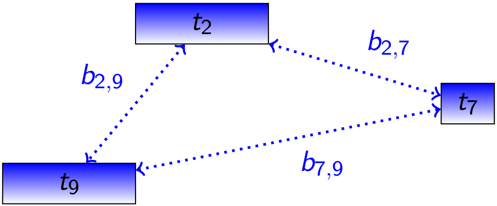
Constraint Model



Model

- A Variable for the start time of each task: $t_i \in [0, \ldots, C_{max}]$.
  - Precedence constraints: $t_i + p_i \leq t_{i+1}$.

Constraint Model



Model

- A Variable for the start time of each task: $t_i \in [0, \ldots, C_{max}]$.
  - Precedence constraints: $t_i + p_i \leq t_{i+1}$.
- A Boolean Variable standing for the relative order of each pair of conflicting tasks (disjunct):
  - Binary Disjunctive constraints: $b_{ij} = \begin{cases} 0 \Leftrightarrow t_i + p_i \leq t_j \\ 1 \Leftrightarrow t_j + p_j \leq t_i \end{cases}$

Search Strategy

Search Strategy

Adaptive heuristic

- Branch on Boolean variables only (order tasks on machines)
- Minimum domain over weighted degree [Boussemart et al. 04]

Search Strategy

Adaptive heuristic

- Branch on Boolean variables only (order tasks on machines)
- Minimum domain over weighted degree [Boussemart et al. 04]

Guided search

- Follow the branch corresponding to the best solution [Beck 07]
- $\simeq$ phase-saving heuristic in SAT [Pipatsrisawat & Darwiche 07]

Search Strategy

Adaptive heuristic

- Branch on Boolean variables only (order tasks on machines)
- Minimum domain over weighted degree [Boussemart et al. 04]

Guided search

- Follow the branch corresponding to the best solution [Beck 07]
- $\simeq$ phase-saving heuristic in SAT [Pipatsrisawat & Darwiche 07]

Restarts

- Geometric [Walsh 99], nogoods on restarts [Lecoutre et al. 07]

Search Strategy

Adaptive heuristic

- Branch on Boolean variables only (order tasks on machines)
- Minimum domain over weighted degree [Boussemart et al. 04]

Guided search

- Follow the branch corresponding to the best solution [Beck 07]
- $\simeq$ phase-saving heuristic in SAT [Pipatsrisawat & Darwiche 07]

Restarts

- Geometric [Walsh 99], nogoods on restarts [Lecoutre et al. 07]

- Almost no problem specific method

CP or SAT?

- Many similarities with SAT:
  - Search variables are Boolean
  - Propagation is very basic
  - SAT-based search strategies

CP or SAT?

- Many similarities with SAT:
  - Search variables are Boolean
  - Propagation is very basic
  - SAT-based search strategies

Some differences

- Faster propagation, but no clause learning

CP or SAT?

- Many similarities with SAT:
  - Search variables are Boolean
  - Propagation is very basic
  - SAT-based search strategies

Some differences

- Faster propagation, but no clause learning
- Restarts + weighted degree “simulates” CDCL behavior?

Experiment on Jobshop and Variants

Experiment on Jobshop and Variants

- Sequence-dependent setup times
    - Transition between tasks on a machine

Experiment on Jobshop and Variants

- Sequence-dependent setup times
  - Transition between tasks on a machine
  - Add the transition times in the disjunct

Experiment on Jobshop and Variants

- Sequence-dependent setup times
  - Transition between tasks on a machine
  - Add the transition times in the disjunct
- Maximum time lags
  - Maximum duration between consecutive tasks in a job

Experiment on Jobshop and Variants

- Sequence-dependent setup times
  - Transition between tasks on a machine
  - Add the transition times in the disjunct
- Maximum time lags
  - Maximum duration between consecutive tasks in a job
  - Precedences with negative durations

Experiment on Jobshop and Variants

- Sequence-dependent setup times
  - Transition between tasks on a machine
  - Add the transition times in the disjunct
- Maximum time lags
  - Maximum duration between consecutive tasks in a job
  - Precedences with negative durations
- Just in Time scheduling
  - Penalties for earliness and tardiness of each job

Experiment on Jobshop and Variants

- Sequence-dependent setup times
  - Transition between tasks on a machine
  - Add the transition times in the disjunct
- Maximum time lags
  - Maximum duration between consecutive tasks in a job
  - Precedences with negative durations
- Just in Time scheduling
  - Penalties for earliness and tardiness of each job
  - Simple decomposition to express the new objective

Experimental Protocol

- This simple model was run on several standard benchmarks
  - 1 hour cutoff
  - 10 random runs, we take the best
- Best known results on each benchmark (LS, CP, MIP)
  - The cutoff may be different
  - The hardware is different

Experimental Protocol

- This simple model was run on several standard benchmarks
  - 1 hour cutoff
  - 10 random runs, we take the best
- Best known results on each benchmark (LS, CP, MIP)
  - The cutoff may be different
  - The hardware is different
- Average % deviation (with respect to a method M in $\{MIP, CP, LS\}$)
  - $$100 \times \sum_{\text{instance } x} \frac{M\ objective(x) - SAT\ objective(x)}{\#instances \times best\ objective(x)}$$

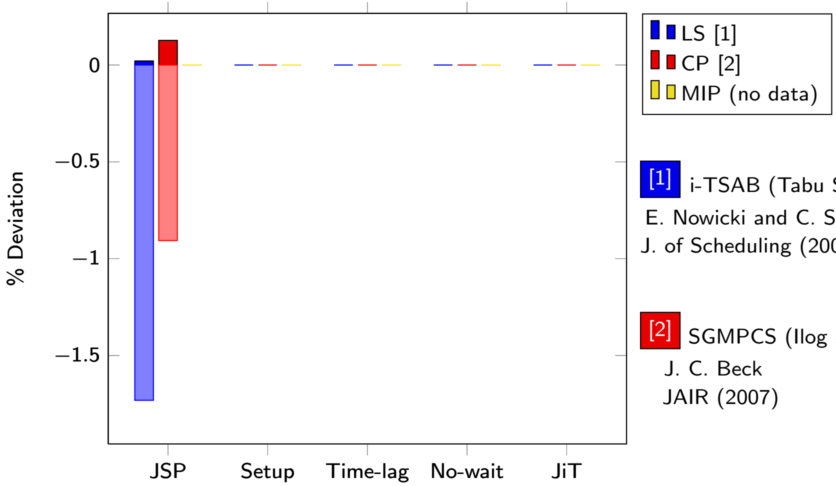
Experimental Protocol

- This simple model was run on several standard benchmarks
  - 1 hour cutoff
  - 10 random runs, we take the best
- Best known results on each benchmark (LS, CP, MIP)
  - The cutoff may be different
  - The hardware is different
- Average % deviation (with respect to a method M in $\{MIP, CP, LS\}$)
  - $$100 \times \sum_{\text{instance } x} \frac{M\ objective(x) - SAT\ objective(x)}{\#instances \times best\ objective(x)}$$
  - Negative: how much worse than M (when it is)

Experimental Protocol

- This simple model was run on several standard benchmarks
  - 1 hour cutoff
  - 10 random runs, we take the best
- Best known results on each benchmark (LS, CP, MIP)
  - The cutoff may be different
  - The hardware is different
- Average % deviation (with respect to a method M in $\{MIP, CP, LS\}$)
  - $$100 \times \sum_{\text{instance } x} \frac{M\ objective(x) - SAT\ objective(x)}{\#instances \times best\ objective(x)}$$
  - Negative: how much worse than M (when it is)
  - Positive: how much better than M (when it is)

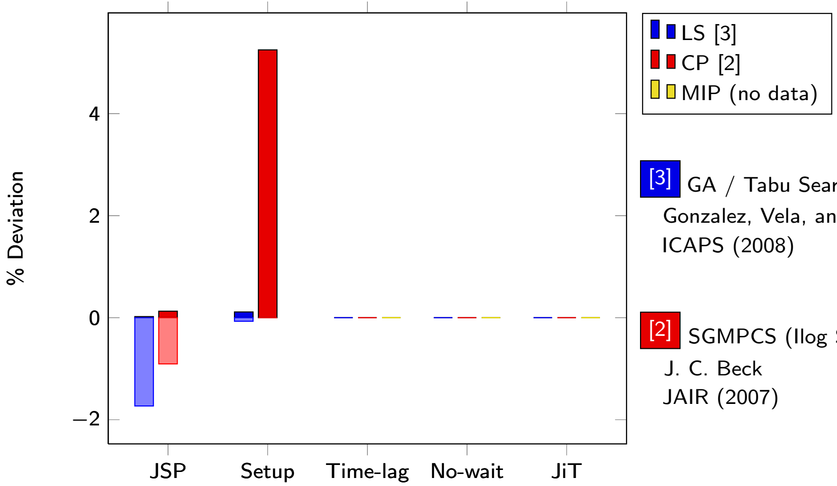
Jobshop - C_{max} - Taillard



[1] i-TSAB (Tabu Search)
E. Nowicki and C. Smutnicki
J. of Scheduling (2005)

[2] SGMPCS (Ilog Scheduler)
J. C. Beck
JAIR (2007)

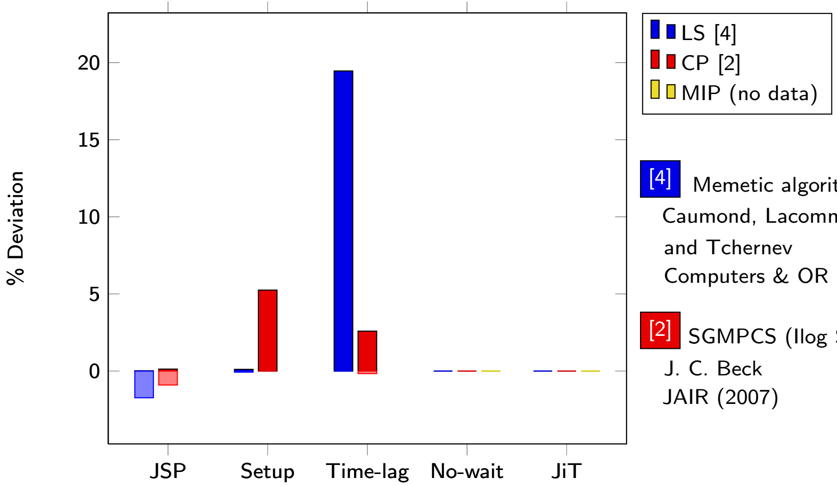
Jobshop with setup times - C_{max} - Brucker & Thiele



[3] GA / Tabu Search
Gonzalez, Vela, and Varela
ICAPS (2008)

[2] SGMPCS (Ilog Scheduler)
J. C. Beck
JAIR (2007)

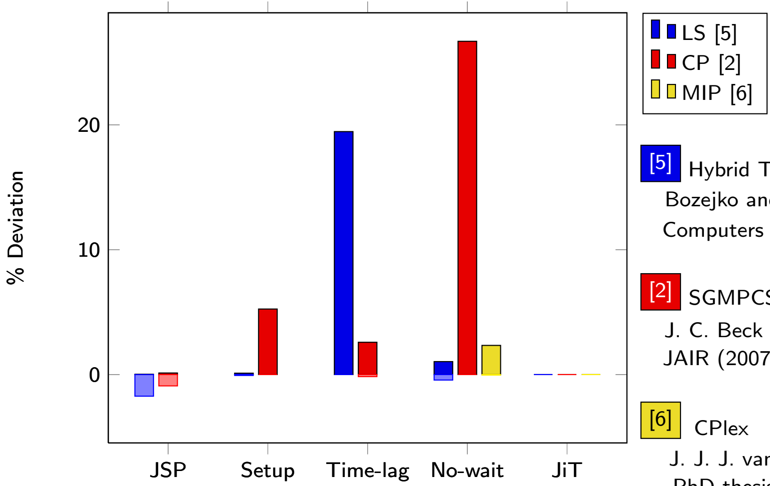
Jobshop with time lags - C_{max} - Lawrence (modified)



[4] Memetic algorithm
Caumond, Lacomme and Tchernev
Computers & OR (2008)

[2] SGMPCS (Ilog Scheduler)
J. C. Beck
JAIR (2007)

"No-wait" Jobshop - C_{max} - Lawrence

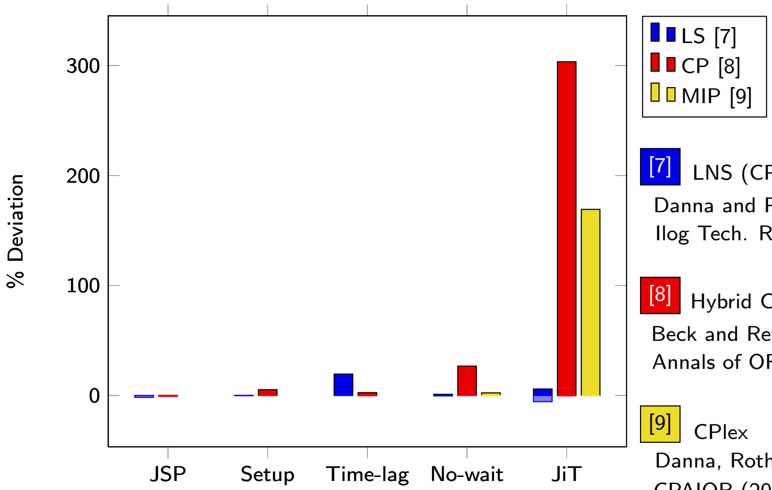


[5] Hybrid Tabu Search
Bozejko and Makuchowski
Computers & IE (2009)

[2] SGMPCS (Ilog Scheduler)
J. C. Beck
JAIR (2007)

[6] CPlex
J. J. J. van den Broek
PhD thesis (2009)

Jobshop - earliness/tardiness - Beck & Refalo; Morton & Pentico



[7] LNS (CP/LP)
Danna and Peron
Ilog Tech. Report (2003)

[8] Hybrid CP/LP
Beck and Refalo
Annals of OR (2003)

[9] CPlex
Danna, Rothberg, Le Pape
CPAIOR (2003)

SAT Strategies

- Often comparable or better than the state of the art

SAT Strategies

- Often comparable or better than the state of the art
  - On benchmarks that are more favorable?

SAT Strategies

- Often comparable or better than the state of the art
  - On benchmarks that are more favorable?
  - On benchmarks that received less attention?

SAT Strategies

- Often comparable or better than the state of the art
  - On benchmarks that are more favorable?
  - On benchmarks that received less attention?
- Adaptive heuristics are extremely powerful

SAT Strategies

- Often comparable or better than the state of the art
  - On benchmarks that are more favorable?
  - On benchmarks that received less attention?
- Adaptive heuristics are extremely powerful
  - Effective at detecting bottlenecks

SAT Strategies

- Often comparable or better than the state of the art
  - On benchmarks that are more favorable?
  - On benchmarks that received less attention?
- Adaptive heuristics are extremely powerful
  - Effective at detecting bottlenecks
  - Often better than dedicated CP approaches to prove optimality

SAT Strategies

- Often comparable or better than the state of the art
  - On benchmarks that are more favorable?
  - On benchmarks that received less attention?
- Adaptive heuristics are extremely powerful
  - Effective at detecting bottlenecks
  - Often better than dedicated CP approaches to prove optimality
    - Even this "pseudo" learning helps!

Outline

SAT Hybrids

- Pure reformulation is surprisingly efficient

SAT Hybrids

- Pure reformulation is surprisingly efficient
- However, simply using an adaptive heuristic and restart seems at least as good

SAT Hybrids

- Pure reformulation is surprisingly efficient
- However, simply using an adaptive heuristic and restart seems at least as good

Hybridization

SAT Hybrids

- Pure reformulation is surprisingly efficient
- However, simply using an adaptive heuristic and restart seems at least as good

Hybridization

- SAT-based learning AND CP-based propagation

SAT Hybrids

- Pure reformulation is surprisingly efficient
- However, simply using an adaptive heuristic and restart seems at least as good

Hybridization

- SAT-based learning AND CP-based propagation
  - What is the best tradeoff?

SAT Hybrids

- Pure reformulation is surprisingly efficient
- However, simply using an adaptive heuristic and restart seems at least as good

Hybridization

- SAT-based learning AND CP-based propagation
  - What is the best tradeoff?
  - Does there need to be a tradeoff?

SAT Hybrids

- Pure reformulation is surprisingly efficient
- However, simply using an adaptive heuristic and restart seems at least as good

Hybridization

- SAT-based learning AND CP-based propagation
  - What is the best tradeoff?
  - Does there need to be a tradeoff?

- Lazy Clause Generation
- SAT Modulo Theories

Lazy Clause Generation [Ohrimenko, Stuckey & Codish 07] - [Feydy & Stuckey 09]

Architecture

- Channel a CP and SAT representations
  - Search and propagation in CP
  - Efficient domain representation and propagators

Lazy Clause Generation [Ohrimenko, Stuckey & Codish 07] - [Feydy & Stuckey 09]

Architecture

- Channel a CP and SAT representations
  - Search and propagation in CP
  - Efficient domain representation and propagators
    - Produce clauses to explain the pruning
    - Just enough to extract a conflict

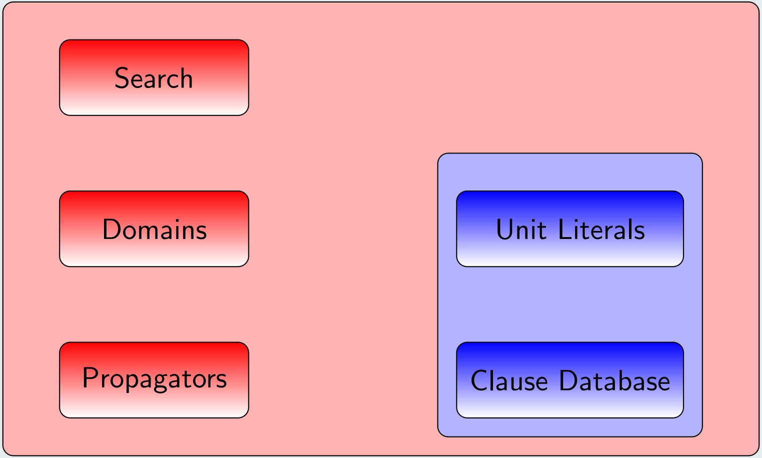
Lazy Clause Generation [Ohrimenko, Stuckey & Codish 07] - [Feydy & Stuckey 09]

Architecture

- Channel a CP and SAT representations
  - Search and propagation in CP
  - Efficient domain representation and propagators
    - Produce clauses to explain the pruning
    - Just enough to extract a conflict
  - The SAT formulation is generated lazily (learned during search)

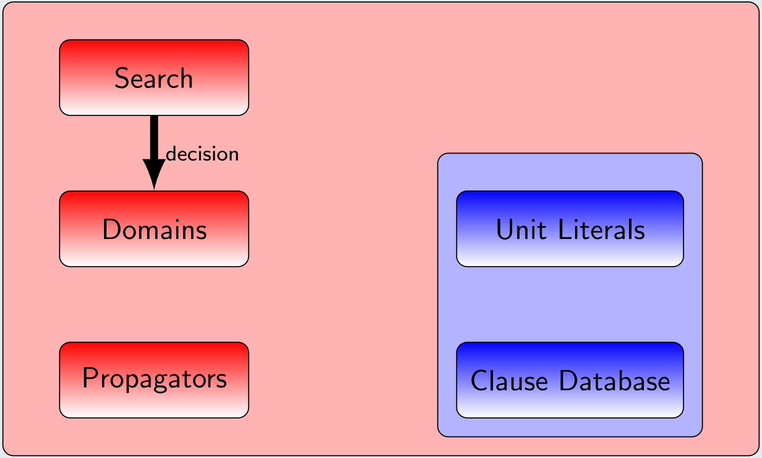
Lazy Clause Generation

Architecture



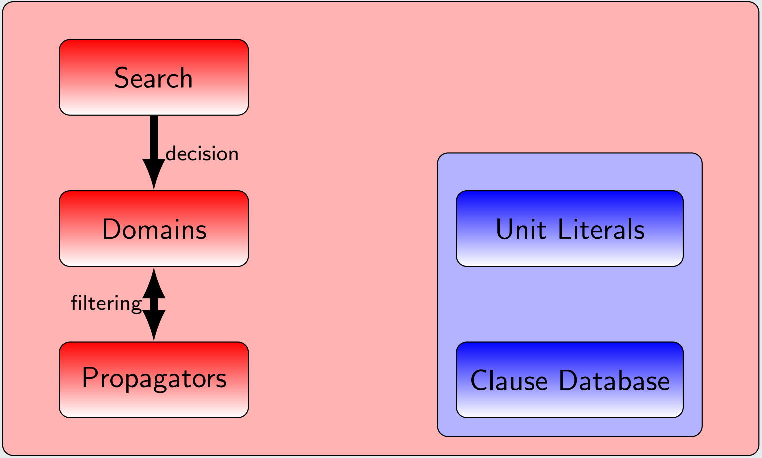
Lazy Clause Generation

Architecture



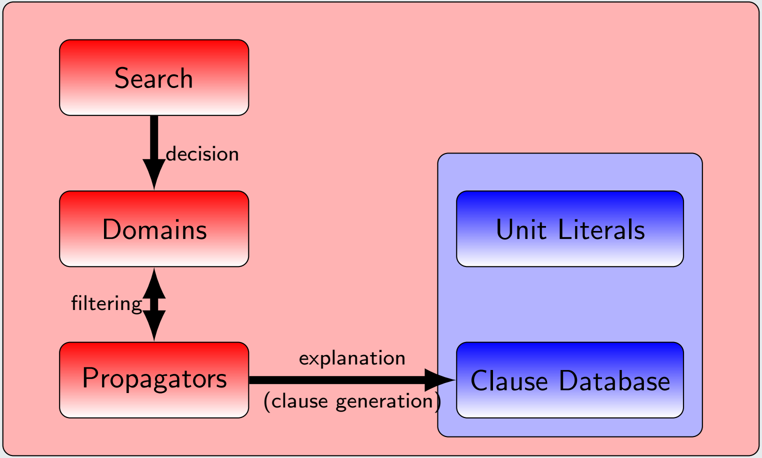
Lazy Clause Generation

Architecture



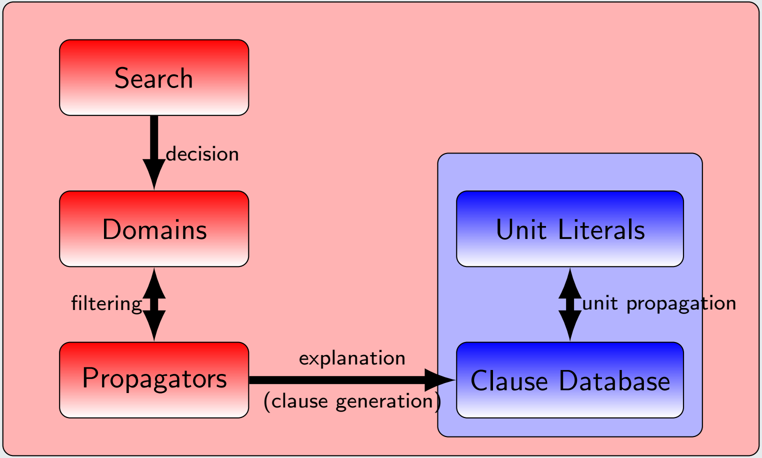
Lazy Clause Generation

Architecture



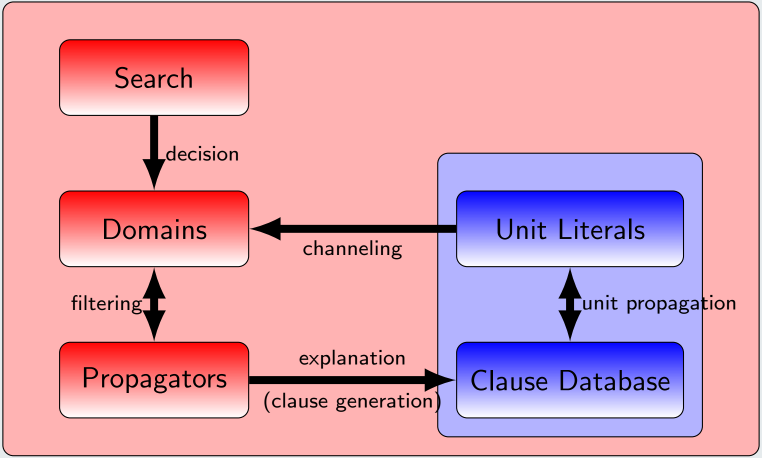
Lazy Clause Generation

Architecture



Lazy Clause Generation

Architecture



Lazy-FD: Example

$x_i < x_j$

- Initial representation

	CP view	SAT view
$D(x_i)$	$\{1, \ldots, 4\}$	$\overline{i_1} \vee i_2, \overline{i_2} \vee i_3$
$D(x_j)$	$\{2, \ldots, 5\}$	$\overline{j_2} \vee j_3, \overline{j_3} \vee j_4$
constraint	$x_i < x_j$	
constraint	$C(x_i, x_k, \ldots)$	

Lazy-FD: Example

$x_i < x_j$

- Some constraint reduces the domain of x_i to $\{2, \ldots, 5\}$

	CP view	SAT view
$D(x_i)$	$\{2, \ldots, 4\}$	$\overline{i_1} \vee i_2, \overline{i_2} \vee i_3$
$D(x_j)$	$\{2, \ldots, 5\}$	$\overline{j_2} \vee j_3, \overline{j_3} \vee j_4$
constraint	$x_i < x_j$	
constraint	$C(x_i, x_k, \ldots)$	

Lazy-FD: Example

$x_i < x_j$

- An explanation clause $T \vee \overline{i_1}$ is produced, and the unit literal $\overline{i_1}$ is propagated

	CP view	SAT view
$D(x_i)$	$\{2, \ldots, 4\}$	$\overline{i_1} \vee i_2, \overline{i_2} \vee i_3$
$D(x_j)$	$\{2, \ldots, 5\}$	$\overline{j_2} \vee j_3, \overline{j_3} \vee j_4$
constraint	$x_i < x_j$	
constraint	$C(x_i, x_k, \ldots)$	$T \vee \overline{i_1}$

Lazy-FD: Example

$x_i < x_j$

- The propagator for $x_i < x_j$ is triggered and reduces the domain of x_j

	CP view	SAT view
$D(x_i)$	$\{2, \ldots, 4\}$	$\overline{i_1} \vee i_2, \overline{i_2} \vee i_3$
$D(x_j)$	$\{3, \ldots, 5\}$	$\overline{j_2} \vee j_3, \overline{j_3} \vee j_4$
constraint	$x_i < x_j$	
constraint	$C(x_i, x_k, \ldots)$	$T \vee \overline{i_1}$

Lazy-FD: Example

$x_i < x_j$

- An explanation clause is also produced

	CP view	SAT view
$D(x_i)$	$\{2, \ldots, 4\}$	$\overline{i_1} \vee i_2, \overline{i_2} \vee i_3$
$D(x_j)$	$\{3, \ldots, 5\}$	$\overline{j_2} \vee j_3, \overline{j_3} \vee j_4$
constraint	$x_i < x_j$	$i_i \vee \overline{j_2}$
constraint	$C(x_i, x_k, \ldots)$	$T \vee \overline{i_1}$

Results on Resource Constrained Project Scheduling Problem (RCPSP) [Schutt, Feydy, Stuckey & Wallace 09]

Resource Constrained Project Scheduling Problem (RCPSP)

- Cumulative resources, each task has a demand r_k for the resource k

Results on Resource Constrained Project Scheduling Problem (RCPSP) [Schutt, Feydy, Stuckey & Wallace 09]

Resource Constrained Project Scheduling Problem (RCPSP)

- Cumulative resources, each task has a demand r_k for the resource k

Model

- Formulated using sums on the order encoding
- A fixed number of runs with a dedicated heuristic, then VSIDS (adaptive heuristic)

Results on Resource Constrained Project Scheduling Problem (RCPSP) [Schutt, Feydy, Stuckey & Wallace 09]

Resource Constrained Project Scheduling Problem (RCPSP)

- Cumulative resources, each task has a demand r_k for the resource k

Model

- Formulated using sums on the order encoding
- A fixed number of runs with a dedicated heuristic, then VSIDS (adaptive heuristic)

Results

- Favorable comparison with state of the art approaches
  - MCS (implemented on top of Ilog-Scheduler [Laborie 05])
  - CP approach by [Liess & Michelon 08]
  - MIP approach by [Koné et al.]

Results on Resource Constrained Project Scheduling Problem (RCPSP) [Schutt, Feydy, Stuckey & Wallace 09]

Resource Constrained Project Scheduling Problem (RCPSP)

- Cumulative resources, each task has a demand r_k for the resource k

Model

- Formulated using sums on the order encoding
- A fixed number of runs with a dedicated heuristic, then VSIDS (adaptive heuristic)

Results

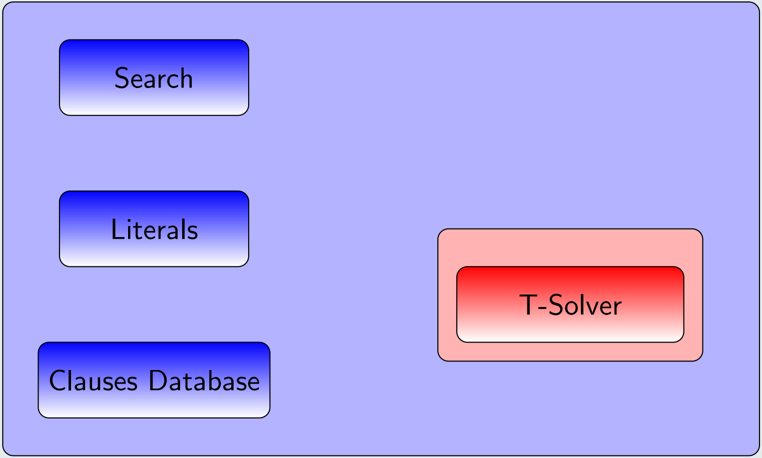
- Favorable comparison with state of the art approaches
  - MCS (implemented on top of Ilog-Scheduler [Laborie 05])
  - CP approach by [Liess & Michelon 08]
  - MIP approach by [Koné et al.]
- 54 open instances closed!

SAT Modulo Theories (SMT)

- Framework to hybridize dedicated solvers (Theories, or T-Solvers) with CDCL solvers
  - T-Solver view: a set of propositions each represented by a literal in F
  - CDCL-Solver view: a CNF formula F partially representing the problem
- CDCL-Solver makes decisions and analyzes the conflicts
- T-Solver detects conflicts and/or propagates and generates explanation clauses

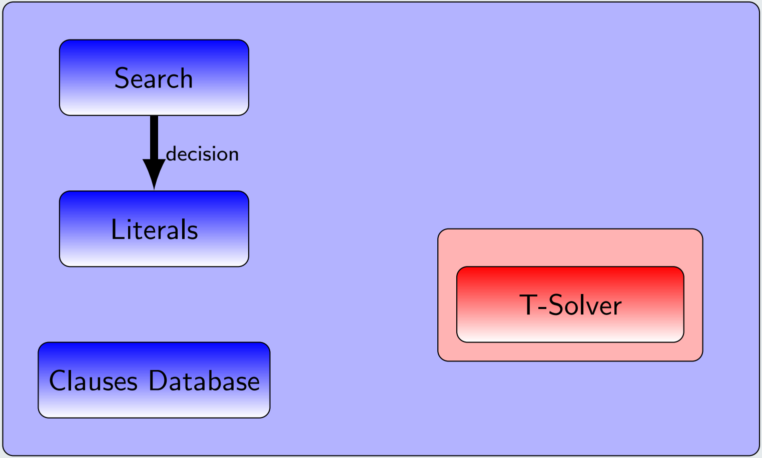
SMT Solver

Architecture



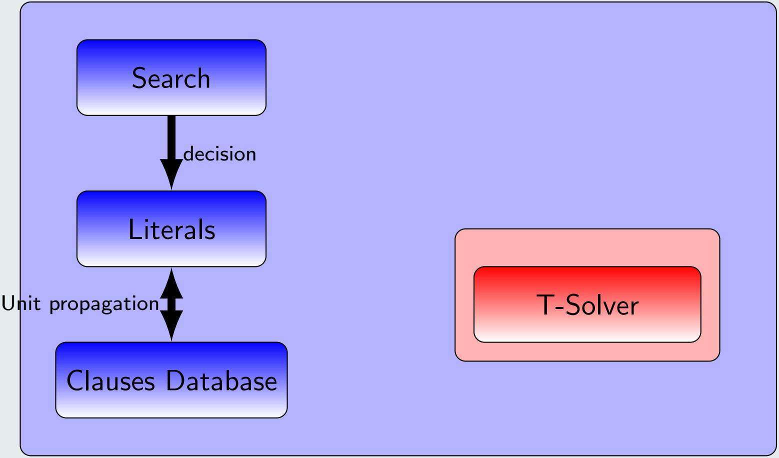
SMT Solver

Architecture



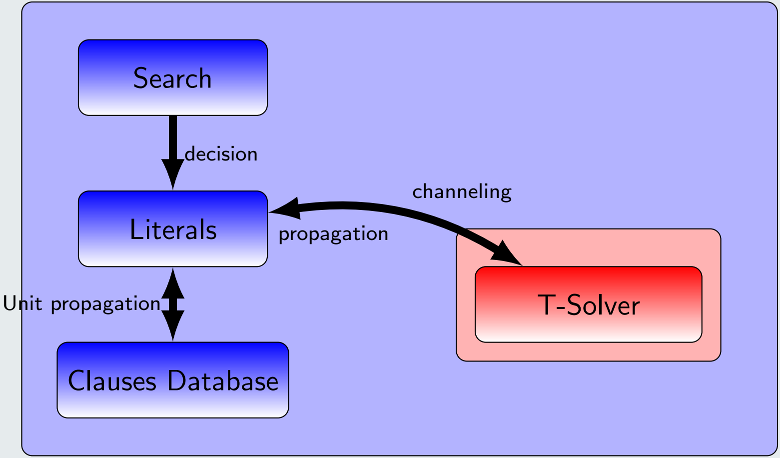
SMT Solver

Architecture



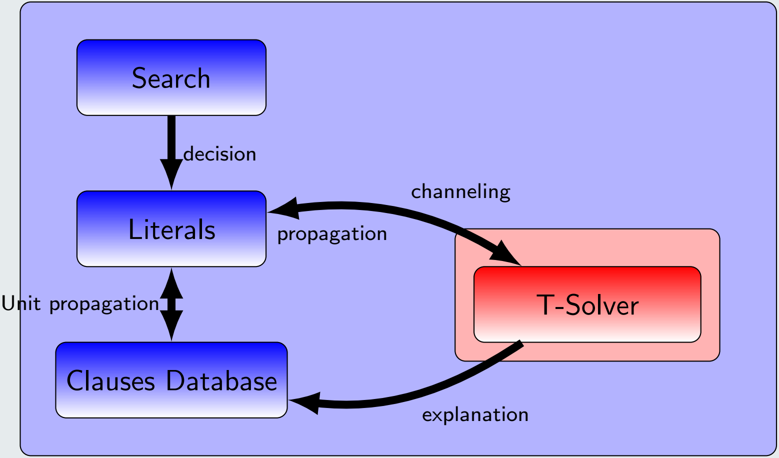
SMT Solver

Architecture



SMT Solver

Architecture



Several Theories

T-Solvers

- Linear Real Arithmetic,
- Arrays,
- Bit-Vectors,
- Equality with Uninterpreted Functions,

Several Theories

T-Solvers

- Linear Real Arithmetic,
- Arrays,
- Bit-Vectors,
- Equality with Uninterpreted Functions,
- Difference Logic (i.e. formulas contain atoms of the form $x - y \leq k$).

Several Theories

T-Solvers

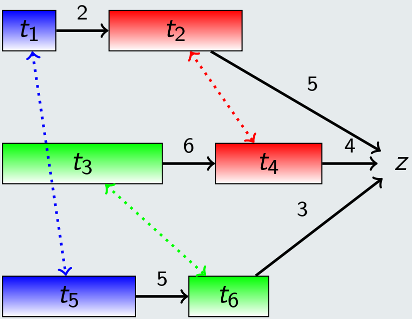
- Linear Real Arithmetic,
- Arrays,
- Bit-Vectors,
- Equality with Uninterpreted Functions,
- Difference Logic (i.e. formulas contain atoms of the form $x - y \leq k$).

SMT for scheduling

- Satisfiability Modulo Difference Logic.

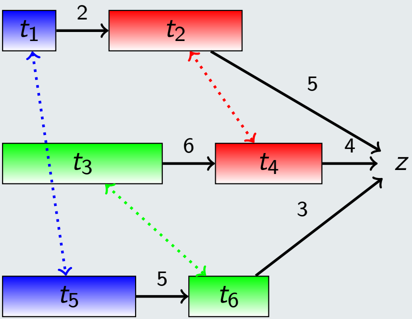
Example: Jobshop Scheduling

problem



Example: Jobshop Scheduling

problem

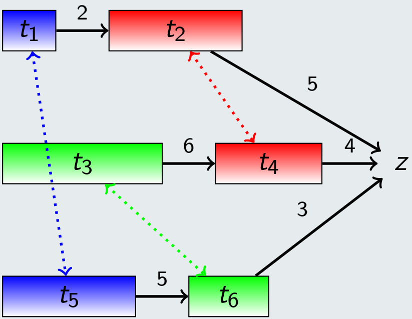


T-Solver view

$$
\begin{array}{ll}
s_1 - s_2 \leq -2 & z - a \leq 15 \\
s_3 - s_4 \leq -4 & \\
s_5 - s_6 \leq -5 & \\
s_2 - z \leq -5 & l_{1\prec 5} \Leftrightarrow s_1 - s_5 \leq -2 \\
s_4 - z \leq -4 & l_{5\prec 1} \Leftrightarrow s_5 - s_1 \leq -5 \\
s_6 - z \leq -3 & l_{2\prec 4} \Leftrightarrow s_2 - s_4 \leq -5 \\
a - s_1 \leq 0 & l_{4\prec 2} \Leftrightarrow s_4 - s_2 \leq -4 \\
a - s_3 \leq 0 & l_{3\prec 6} \Leftrightarrow s_3 - s_6 \leq -6 \\
a - s_5 \leq 0 & l_{6\prec 3} \Leftrightarrow s_6 - s_3 \leq -3
\end{array}
$$

Example: Jobshop Scheduling

problem



T-Solver view

$s_1 - s_2 \leq -2$
$s_3 - s_4 \leq -4$
$s_5 - s_6 \leq -5$
$s_2 - z \leq -5$
$s_4 - z \leq -4$
$s_6 - z \leq -3$
$a - s_1 \leq 0$
$a - s_3 \leq 0$
$a - s_5 \leq 0$

$z - a \leq 15$

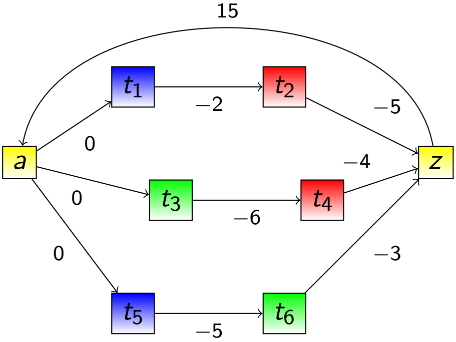
$l_{1 \prec 5} \Leftrightarrow s_1 - s_5 \leq -2$
$l_{5 \prec 1} \Leftrightarrow s_5 - s_1 \leq -5$
$l_{2 \prec 4} \Leftrightarrow s_2 - s_4 \leq -5$
$l_{4 \prec 2} \Leftrightarrow s_4 - s_2 \leq -4$
$l_{3 \prec 6} \Leftrightarrow s_3 - s_6 \leq -6$
$l_{6 \prec 3} \Leftrightarrow s_6 - s_3 \leq -3$

CDCL-Solver view

$l_{1 \prec 5} \vee l_{5 \prec 1}$
$l_{2 \prec 4} \vee l_{4 \prec 2}$
$l_{3 \prec 6} \vee l_{6 \prec 3}$

Example: Jobshop Scheduling

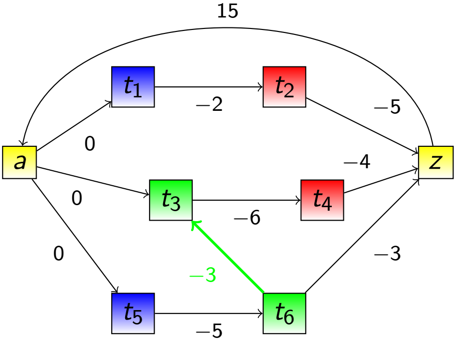
- Reasoning: detection of negative cycles ([Bellman-Ford])



$$l_{1\prec 5} \Leftrightarrow s_1 - s_5 \leq -2$$
$$l_{5\prec 1} \Leftrightarrow s_5 - s_1 \leq -5$$
$$l_{2\prec 4} \Leftrightarrow s_2 - s_4 \leq -5$$
$$l_{4\prec 2} \Leftrightarrow s_4 - s_2 \leq -4$$
$$l_{3\prec 6} \Leftrightarrow s_3 - s_6 \leq -6$$
$$l_{6\prec 3} \Leftrightarrow s_6 - s_3 \leq -3$$

Example: Jobshop Scheduling

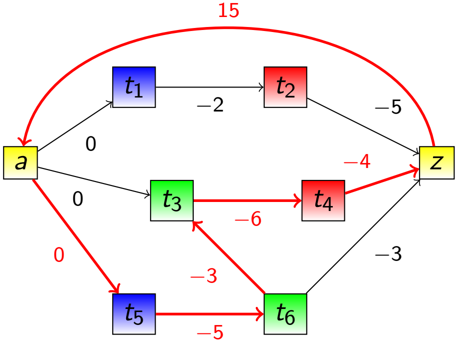
- Reasoning: detection of negative cycles ([Bellman-Ford])



$$
\begin{aligned}
l_{1\prec 5} &\Leftrightarrow s_1 - s_5 \leq -2\\
l_{5\prec 1} &\Leftrightarrow s_5 - s_1 \leq -5\\
l_{2\prec 4} &\Leftrightarrow s_2 - s_4 \leq -5\\
l_{4\prec 2} &\Leftrightarrow s_4 - s_2 \leq -4\\
l_{3\prec 6} &\Leftrightarrow s_3 - s_6 \leq -6\\
l_{6\prec 3} &\Leftrightarrow s_6 - s_3 \leq -3
\end{aligned}
$$

Example: Jobshop Scheduling

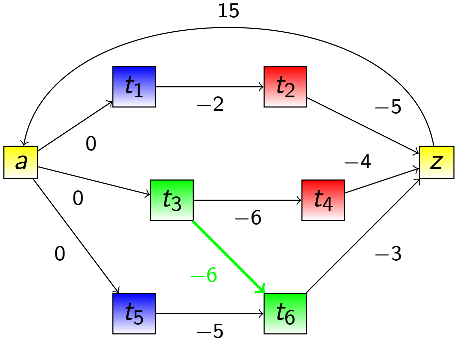
- Reasoning: detection of negative cycles ([Bellman-Ford])



$$
\begin{aligned}
l_{1\prec 5} &\Leftrightarrow s_1 - s_5 \leq -2\\
l_{5\prec 1} &\Leftrightarrow s_5 - s_1 \leq -5\\
l_{2\prec 4} &\Leftrightarrow s_2 - s_4 \leq -5\\
l_{4\prec 2} &\Leftrightarrow s_4 - s_2 \leq -4\\
l_{3\prec 6} &\Leftrightarrow s_3 - s_6 \leq -6\\
l_{6\prec 3} &\Leftrightarrow s_6 - s_3 \leq -3
\end{aligned}
$$

Example: Jobshop Scheduling

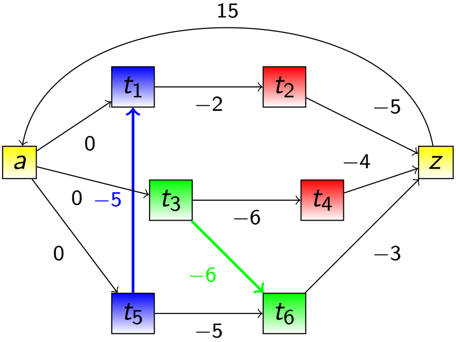
- Reasoning: detection of negative cycles ([Bellman-Ford])



$l_{1\prec 5} \Leftrightarrow s_1 - s_5 \leq -2$
$l_{5\prec 1} \Leftrightarrow s_5 - s_1 \leq -5$
$l_{2\prec 4} \Leftrightarrow s_2 - s_4 \leq -5$
$l_{4\prec 2} \Leftrightarrow s_4 - s_2 \leq -4$
$l_{3\prec 6} \Leftrightarrow s_3 - s_6 \leq -6$
$l_{6\prec 3} \Leftrightarrow s_6 - s_3 \leq -3$

Example: Jobshop Scheduling

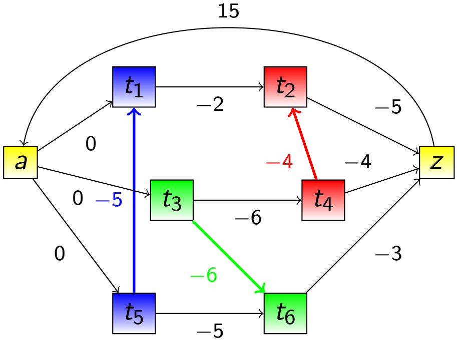
- Reasoning: detection of negative cycles ([Bellman-Ford])



$l_{1\prec 5} \Leftrightarrow s_1 - s_5 \leq -2$
$l_{5\prec 1} \Leftrightarrow s_5 - s_1 \leq -5$
$l_{2\prec 4} \Leftrightarrow s_2 - s_4 \leq -5$
$l_{4\prec 2} \Leftrightarrow s_4 - s_2 \leq -4$
$l_{3\prec 6} \Leftrightarrow s_3 - s_6 \leq -6$
$l_{6\prec 3} \Leftrightarrow s_6 - s_3 \leq -3$

Example: Jobshop Scheduling

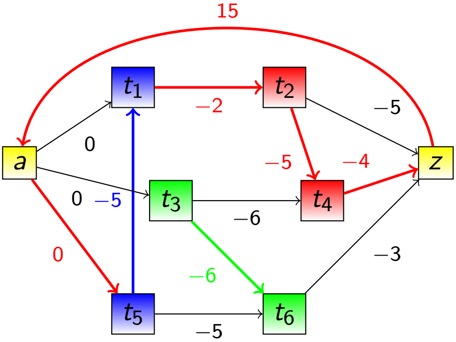
- Reasoning: detection of negative cycles ([Bellman-Ford])



$$l_{1\prec 5} \Leftrightarrow s_1 - s_5 \leq -2$$
$$l_{5\prec 1} \Leftrightarrow s_5 - s_1 \leq -5$$
$$l_{2\prec 4} \Leftrightarrow s_2 - s_4 \leq -5$$
$$l_{4\prec 2} \Leftrightarrow s_4 - s_2 \leq -4$$
$$l_{3\prec 6} \Leftrightarrow s_3 - s_6 \leq -6$$
$$l_{6\prec 3} \Leftrightarrow s_6 - s_3 \leq -3$$

Example: Jobshop Scheduling

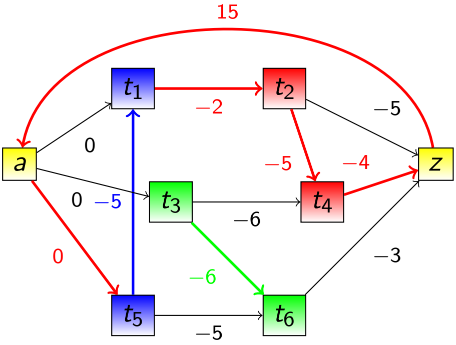
- Reasoning: detection of negative cycles ([Bellman-Ford])



$$l_{1\prec 5} \Leftrightarrow s_1 - s_5 \leq -2$$
$$l_{5\prec 1} \Leftrightarrow s_5 - s_1 \leq -5$$
$$l_{2\prec 4} \Leftrightarrow s_2 - s_4 \leq -5$$
$$l_{4\prec 2} \Leftrightarrow s_4 - s_2 \leq -4$$
$$l_{3\prec 6} \Leftrightarrow s_3 - s_6 \leq -6$$
$$l_{6\prec 3} \Leftrightarrow s_6 - s_3 \leq -3$$

Example: Jobshop Scheduling

- Reasoning: detection of negative cycles ([Bellman-Ford])



$$l_{1\prec 5} \Leftrightarrow s_1 - s_5 \leq -2$$
$$l_{5\prec 1} \Leftrightarrow s_5 - s_1 \leq -5$$
$$l_{2\prec 4} \Leftrightarrow s_2 - s_4 \leq -5$$
$$l_{4\prec 2} \Leftrightarrow s_4 - s_2 \leq -4$$
$$l_{3\prec 6} \Leftrightarrow s_3 - s_6 \leq -6$$
$$l_{6\prec 3} \Leftrightarrow s_6 - s_3 \leq -3$$

Learned clause

- $\overline{l_{5\prec 1}} \vee \overline{l_{2\prec 4}}$

Results on Resource Constrained Project Scheduling Problem (RCPSP) [Ansótegui et al. 11]

Results on Resource Constrained Project Scheduling Problem (RCPSP) [Ansótegui et al. 11]

Two fomulations

- Time encoding
- Task encoding

Results on Resource Constrained Project Scheduling Problem (RCPSP) [Ansótegui et al. 11]

Two fomulations

- Time encoding
- Task encoding

Results

- More robust than lazy-FD

Results on Resource Constrained Project Scheduling Problem (RCPSP) [Ansótegui et al. 11]

Two fomulations

- Time encoding
- Task encoding

Results

- More robust than lazy-FD
- State of the art for RCPSP!

Outline

Conclusion

Conclusion

- Scheduling with SAT is not as bad as it sounds

Conclusion

- Scheduling with SAT is not as bad as it sounds
- Generic algorithms can sometimes be difficult to match
    - Adaptive heuristics
    - Clause learning

Conclusion

- Scheduling with SAT is not as bad as it sounds
- Generic algorithms can sometimes be difficult to match
    - Adaptive heuristics
    - Clause learning
- Nogood learning [Schiex & Verfaillie 93] and explanation for global constraints [Rochart & Jussien 03], disjunctive resource [Vilím 05]?
    - Somehow it does not have the same impact as in SAT

Conclusion

- Scheduling with SAT is not as bad as it sounds
- Generic algorithms can sometimes be difficult to match
  - Adaptive heuristics
  - Clause learning
- Nogood learning [Schiex & Verfaillie 93] and explanation for global constraints [Rochart & Jussien 03], disjunctive resource [Vilím 05]?
  - Somehow it does not have the same impact as in SAT
- Hybridization (learning + dedicated reasoning) is the way to go
  - SAT Modulo Theories?
  - CDCL with global constraints and integer domains?
  - Explanation algorithms for global constraints?