Planning as Satisfiability

Overview

- Observation
  - solvers are developed for many NP-complete classes of problems
  - **progress is not uniform** (reasons?)
- Progress in solving SAT is probably most prominent
- Idea (Kautz & Selman, 91-96):
  - **Maybe we should teach SAT solvers to solve planning?**
  - Problem: Strips planning is PSPACE-complete
  - Solution: Bounded-Strips planning is in NP

Planning as Satisfiability

Transform Planning into a series of SATs

Given $\Pi = (P, A, I, G)$:

$b = 0$

while TRUE **do**

$\Phi(\Pi, b)$:= a CNF that is satisfiable iff
there exists a plan with b steps

if DPLL$(\Phi(\Pi, b), \emptyset)$ **then**

output Plan encoded by a satisfying assignment

$b := b + 1$

Questions

- What notion of “steps” can we use (time/action)?
- What do we know about the plan found?
- What should be the connection between the set of plans for Π and the set of satisfying assignments of $\Phi(\Pi, b)$?
- What can we say about the completeness of the algorithm?

Strips Encodings

How to encode b-step Strips plan existence as a CNF?

Many possible answers. All (in use to date) share:

- *Time steps* $0 \leq t \leq b$
- Fact variables p_t: is p TRUE or FALSE at t?
- Action variables a_t: is a applied at t or not?

- The size of the encoding grows linearly in b

The Linear Encoding, I

Sequential planning

- Problem $\Pi = (P, A, I, G)$, time steps $0 \leq t \leq b$
- Decision variables

  p_t — for all $p \in P, 0 \leq t \leq b$

  a_t — for all $a \in A, 0 \leq t \leq b - 1$
- Initial State Clauses: "Initial state holds at time 0"

  for all $p \in P$: $\{p_0\}$ if $p \in I$, and $\{\neg p_0\}$, otherwise
- Goal Clauses: "Goals satisfied a time b"

  for all $p \in G$: $\{p_b\}$

The Linear Encoding, II

Sequential planning

- Action Precondition Clauses:
  "action implies its preconditions"

  for all $a \in A, p \in pre(a), 0 \leq t \leq b-1$: $\{\neg a_t, p_t\}$

- Action Effect Clauses:
  "action implies its add/delete effects"

  for all $a \in A, p \in add(a), 0 \leq t \leq b-1$: $\{\neg a_t, p_{t+1}\}$
  for all $a \in A, p \in del(a), 0 \leq t \leq b-1$: $\{\neg a_t, \neg p_{t+1}\}$

The Linear Encoding, III

Sequential planning

- Positive Frame Axioms:
  "if a is applied and $p \notin del(a)$ was true, then p is still true"

  for all $a \in A, p \notin del(a), 0 \leq t \leq b-1$: $\{\neg a, \neg p_t, p_{t+1}\}$

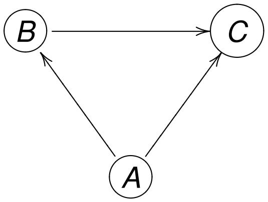
- Negative Frame Axioms:
  "if a is applied and $p \notin add(a)$ was false, then p is still false"

  for all $a \in A, p \notin add(a), 0 \leq t \leq b-1$: $\{\neg a, p_t, \neg p_{t+1}\}$

- Linearity (Exclusion) Constraints:
  "apply exactly one action at each time step"

  for all $a, a' \in A, 0 \leq t \leq b-1$: $\{\neg a, \neg a'_t\}$
  for all $0 \leq t \leq b-1$: A_t (*do we really need them?*)

Example



- $P = \{A, B, C, visB, visC\}$, $I = \{A\}$, $G = \{visB, visC\}$
- Actions

$$\begin{aligned} drAB &= \{\{A\}, \{B, visB\}, \{A\}\} \\ drAC &= \{\{A\}, \{C, visC\}, \{A\}\} \\ drBC &= \{\{B\}, \{C, visC\}, \{B\}\} \end{aligned}$$

Blackboard: Linear encoding for $b = 1$

A Basic Parallel Encoding, I

Parallel planning

- Problem $\Pi = (P, A, I, G)$, noops-extended actions A^N, time steps $0 \leq t \leq b$
- Decision variables

  p_t — for all $p \in P, 0 \leq t \leq b$
  a_t — for all $a \in A^N, 0 \leq t \leq b - 1$
- Initial State Clauses: "Initial state holds at time 0"

  for all $p \in P$: $\{p_0\}$ if $p \in I$, and $\{\neg p_0\}$, otherwise
- Goal Clauses: "goal satisfied at time b"

  for all $p \in G$: $\{p_b\}$

A Basic Parallel Encoding, II

Parallel planning

- Action Precondition Clauses:
  "action implies its preconditions"

  for all $a \in A^N, p \in pre(a), 0 \leq t \leq b-1$: $\{\neg a_t, p_t\}$

- Action Interference Clauses:
  "do not apply interfering actions in the same time step"

  for all $a, a' \in A^N, a \not| a', 0 \leq t \leq b-1$: $\{\neg a_t, \neg a'_t\}$

- Fact Achievement Clauses:
  "fact implies disjunction of its achievers"

  for all $p \in P, 1 \leq t \leq b$: $\{\neg p_t\} \cup \{a_{t-1} | p \in add(a)\}$

A Basic Parallel Encoding, II

Parallel planning

- Action Precondition Clauses:
  "action implies its preconditions"
  for all $a \in A^N, p \in pre(a), 0 \leq t \leq b-1$: $\{\neg a_t, p_t\}$

- Action Interference Clauses:
  "do not apply interfering actions in the same time step"
  for all $a, a' \in A^N, a \not| a', 0 \leq t \leq b-1$: $\{\neg a_t, \neg a'_t\}$

- Fact Achievement Clauses:
  "fact implies disjunction of its achievers"
  for all $p \in P, 1 \leq t \leq b$: $\{\neg p_t\} \cup \{a_{t-1} | p \in add(a)\}$

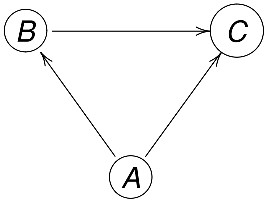
Do we need anything else?

Linear vs. Parallel Encodings

- Optimal parallel plans are often shorter than optimal sequential plans
- Linearity constraints typically dominate the linear encodings

So in parallel planning-as-SAT we (typically) need fewer iterations and (always) consider smaller formulas!

Example



- $P = \{A, B, C, visB, visC\}$, $I = \{A\}$, $G = \{visB, visC\}$
- Actions

$$drAB = \{\{A\}, \{B, visB\}, \{A\}\}$$
$$drAC = \{\{A\}, \{C, visC\}, \{A\}\}$$
$$drBC = \{\{B\}, \{C, visC\}, \{B\}\}$$

Blackboard: Basic parallel encoding for $b = 1$

2-planning graphs extend 1-planning graphs by keeping track of **mutex pairs**; pairs that cannot be **simultaneously** achieved in i steps:

- **action pair mutex** at i if actions interfere or their preconditions mutex at i
- **atom pair mutex** at i if all supporting action pairs are mutex at $i-1$
- a **set of atoms** C is **mutex** at i if it contains a mutex pair at i

Resulting graph:

- $P_0 = \{p \in I\}$
- $A_i = \{a \in A^N \mid Prec(a) \subseteq P_i \text{ and not mutex at } i\}$
- $P_{i+1} = \{p \in Add(a) \mid a \in A_i\}$,
  with sets of action/atom mutex pairs defined as above.

The Planning Graph Based Encoding, I

- Problem $\Pi = (P, A, I, G)$, noops-extended actions A^N, time steps $0 \leq t \leq b$
- Fact layers $P_{(t)}$, action layers $A_{(t)}$, fact mutexes (layers) $EP_{(t)}$, action mutexes (layers) $EA_{(t)}$

- Decision variables

  p_t — for all $p \in P, 1 \leq t \leq b$
  a_t — for all $a \in A^N, 0 \leq t \leq b - 1$

- Goal Clauses: "specify goal values"

  for all $p \in G$: $\{p_b\}$

- Action Precondition Clauses:
  "action implies its preconditions"

  for all $a \in A^N, p \in pre(a), 1 \leq t \leq b - 1$: $\{\neg a_t, p_t\}$

The Planning Graph Based Encoding, II

- Action Mutex Clauses: "do not apply mutex actions in the same time step"

  for all $0 \leq t \leq b-1, a, a' \in A_{(t)}, \{a, a'\} \in EA_{(t)}$: $\{\neg a_t, \neg a'_t\}$

- Fact Achievement Clauses:
  "fact implies disjunction of its achievers"

  for all $p \in P, 1 \leq t \leq b$: $\{\neg p_t\} \cup \{a_{t-1} | p \in add(a)\}$

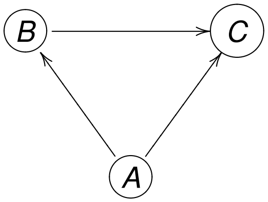
- Fact Mutex Clauses:
  "do not make two mutex facts TRUE"

  for all $1 \leq t \leq b, p, p' \in P_{(t)}, \{p, p'\} \in EP_{(t)}$: $\{\neg p_t, \neg p'_t\}$

Basic Parallel vs. PG-Based Encoding, I

- PG-Based Encoding == Basic Parallel Encoding pruned and enhanced by information contained in 2-Planning Graph
- Pruned: **less** decision variables p_t and a_t, less redundant exclusion clauses
  - Example: We dont need vars for the initial facts since $pre(a) \subseteq I$ holds anyway for all $a \in A_{(0)}$
- Enhanced: **more** non-trivial (temporal) exclusion clauses $\{\neg a_t, \neg a'_t\}$ and $\{\neg p_t, \neg p'_t\}$

Example



- $P = \{A, B, C, visB, visC\}$, $I = \{A\}$, $G = \{visB, visC\}$
- Actions

$$
\begin{aligned}
drAB &= \{\{A\}, \{B, visB\}, \{A\}\} \\
drAC &= \{\{A\}, \{C, visC\}, \{A\}\} \\
drBC &= \{\{B\}, \{C, visC\}, \{B\}\}
\end{aligned}
$$

Blackboard: PG-based encoding for $b = 1$

Basic Parallel vs. PG-Based Encoding, I (Recall)

- PG-Based Encoding == Basic Parallel Encoding pruned and enhanced by information contained in 2-Planning Graph
- Pruned: **less** decision variables p_t and a_t, less redundant exclusion clauses
  - Example: We dont need vars for the initial facts since $pre(a) \subseteq I$ holds anyway for all $a \in A_{(0)}$
- Enhanced: **more** non-trivial (temporal) exclusion clauses $\{\neg a_t, \neg a'_t\}$ and $\{\neg p_t, \neg p'_t\}$

Basic Parallel vs. PG-Based Encoding, II

- All new clauses (the pruned $\{\neg p_t\}$ and $\{\neg a_t\}$, and all new exclusion clauses) **follow** from the Basic Parallel CNF Φ

- By constructing 2-planning graph and basic our SAT encoding on it ...
  - . . . we do some of the reasoning devoted to the SAT solver with a specialized algorithm instead
  - But why this part of work and not all the work?

Basic Parallel vs. PG-Based Encoding, II

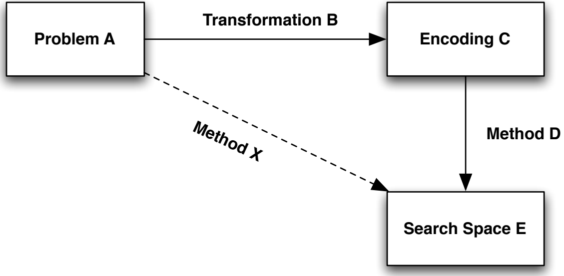
- All new clauses (the pruned $\{\neg p_t\}$ and $\{\neg a_t\}$, and all new exclusion clauses) **follow** from the Basic Parallel CNF Φ

- By constructing 2-planning graph and basic our SAT encoding on it ...
  - ... we do some of the reasoning devoted to the SAT solver with a specialized algorithm instead
  - But why this part of work and not all the work?

- Potentially exponential savings
  - suppose (since) the SAT solver uses, in constraint propagation, 1-Resolution only
  - for exclusion relations we need 2-Resolution! [Brafman, IJCAI-1999, JAIR-2001]

- *What sort of resolution do we need to capture k-planning graphs in the constraint propagation procedure?*

In Front of the Curtains



- What are A, B, C, D, E in our case?
- What is X?

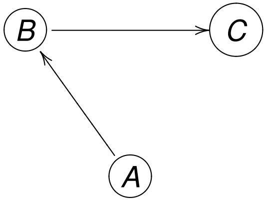
A Very Simple Encoding

Use a 1-planning graph

- Problem $\Pi = (P, A, I, G)$, noops-extended actions A^N, time steps $0 \leq t \leq b$, action layers $A_{(t)}$
- Decision variables: a_t — for all $0 \leq t \leq b - 1$ and $a \in A_{(t)}$
- Goal Clauses: "at least one achiever"
  - for all $p \in G$: $\{a_{b-1} | a \in A_{(b-1)}, g \in add(a)\}$
- Action Precondition Clauses:
  "action implies disjunction of its precondition achievers"

  for all $1 \leq t \leq b - 1, a \in A_{(t)}, p \in pre(a)$:
  $\{\neg a_t\} \cup \{a'_{t-1} | a' \in A_{(t-1)}, p \in add(a')\}$
- Action Interference Clauses: as in basic parallel encoding

Example



- $P = \{A, B, C\}$, $I = \{A\}$, $G = \{C\}$
- Actions

$$drAB = \{\{A\}, \{B\}, \{A\}\}$$
$$drBC = \{\{B\}, \{C\}, \{B\}\}$$

Blackboard: “Very simple” encoding for $b = 2$

Reminder: DPLL

bool **DPLL** (Φ, partial assignment ω)

$(\Phi', \omega') :=$ *unit-propagation*(Φ, ω)
if Φ' contains empty clause **then return** FALSE
select a variable v not assigned by ω'
if no such variable exists **then return** TRUE
if DPLL$(\Phi', \omega' \cup \{v := 1\})$ **then return** TRUE
if DPLL$(\Phi', \omega' \cup \{v := 0\})$ **then return** TRUE
return FALSE

Behind the Curtains, Unit Propagation, I

propagate $a_t = TRUE$

set a IN at t

if $t > 0$ **then forall** $p \in pre(a)$

  if all $a' \in A_{(t-1)}, p \in add(a')$ are OUT at $t - 1$ **then fail**

  if all $a' \in A_{(t-1)}, p \in add(a')$ are OUT at $t - 1$, except a''

    then propagate a'' IN at $t - 1$

forall $a' \in A_{(t)}$ that interfere with a

  propagate a' OUT at t

Behind the Curtains, Unit Propagation, II

propagate $a_t = FALSE$

set a OUT at t

if $t = b - 1$ **then forall** $g \in add(a) \cap G$

if all $a' \in A_{(t)}, g \in add(a')$ are OUT at t **then fail**

if all $a' \in A_{(t)}, g \in add(a')$ are OUT at t, except a''

then propagate a'' IN at $t - 1$

if $t < b - 1$ **then**

???

Behind the Curtains, DPLL

- DPLL makes **commitments** of the form
  "I will/won't apply action *a* at time *t*"
- The search state is a sequence of such commitments

d0 "I will move the truck from *x* to *y* at time 17"

d1 UP: "truck at *x* at time 17", "truck at *y* at time 18"

d1 "I will sell the truck at time 7"

d2 UP: "no truck at time 8, . . . , 25"

d2 FALSE

d1 "I will not sell the truck at time 7"

Behind the Curtains, DPLL

- DPLL makes **commitments** of the form
  "I will/won't apply action *a* at time *t*"
- The search state is a sequence of such commitments

d0 "I will move the truck from *x* to *y* at time 17"
d1   UP: "truck at *x* at time 17", "truck at *y* at time 18"
d1   "I will sell the truck at time 7"
d2   UP: "no truck at time 8, $\ldots$, 25"
d2   FALSE
d1   "I will not sell the truck at time 7"

- The order of commitments in the sequence is **independent** of the time steps *t*
- ... this is why we also call this undirected search

Branching in Planning: A Big Picture

- **Forward**: state-space; extend plan head, totally (possibly weakly) ordered
- **Backward**: regression-space; extend plan tail; totally (possibly weakly) ordered
- **Temporal**: for action *a* and time *i*, create splits $a[i] = \mathit{TRUE}$ / $a[i] = \mathit{FALSE}$
- **POCL**: Partial Order Causal Link Planning
  - next ...

Literature

Planning-as-SAT and Temporal Branching

- H. Kautz and B. Selman, Planning as Satisfiability, ECAI-92
- H. Kautz and B. Selman, Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search, AAAI-96
- H. Kautz, D. McAllester, and B. Selman, Encoding Plans in Propositional Logic, KR-96
- H. Kautz and B. Selman, Unifying SAT-based and Graph-based Planning, IJCAI-99
- J. Rintanen, A Planning Algorithm not Based on Directional Search, KR-98
- J. Hoffmann and H. Geffner, Branching Matters: Alternative Branching in Graphplan, ICAPS-03
- R. Brafman, Reachability, Relevance, Resolution, and the Planning as Satisfiability Approach, IJCAI-99, JAIR-01