Australian National University

CP'18 Workshop on Constraints & AI Planning

Planning with State and Trajectory Constraints

Planning with constraints

$$s_0 \dashrightarrow \boxed{\textit{act 1}} \dashrightarrow s_1 \dashrightarrow \boxed{\textit{act 2}} \dashrightarrow \dots \dashrightarrow \boxed{\textit{act n}} \dashrightarrow s_n \models G$$

- Action pre- and post-conditions impose a fundamental constraint on consecutive states.
- What if we need additional constraints
  - on some or all states traversed by the plan; or
  - on the state sequence?

* Part 1: Planning with state constraints.
  > (Benders'-like) Decomposition: state constraint satisfiability decided by a "black box" solver, planning by heuristic state-space search.

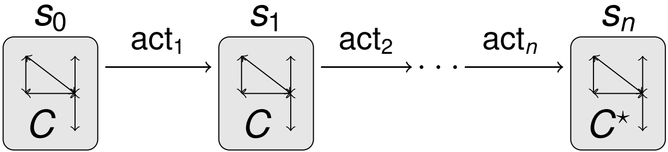
  > Constraint-aware heuristics.
* Part 2: Planning with trajectory constraints.
  > (LTL-like) plan constraints.

  > Plan constraint-specific propagation.

Australian National University

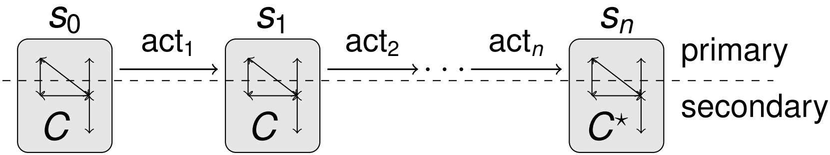
Part 1

State constraints



* Constraints on the valuations of state variables that must be satisfied
  > in every state;
  > in the goal state; or
  > as part of an action's precondition.

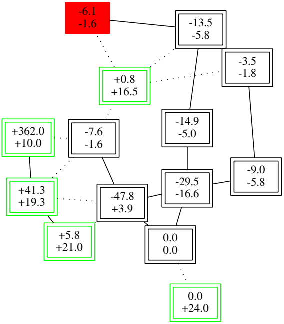
State constraints



* Partition state variables into:
  > *primary*: subject to action effects, inertia.
  > *secondary*: determined by constraints only.
* Secondary variables may be of any type.
* Primary/secondary variables are linked by (switched) constraints.

Example: Power Network Re-Configuration

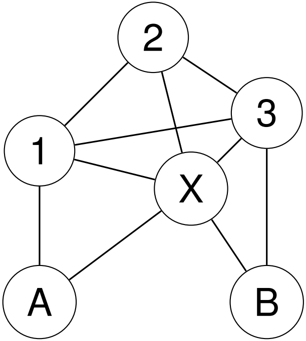
- Discrete actions (e.g., open/close switch).
- Power flow is a (non-linear) function of the global state.
- System stabilises after each discrete action.
- Not all states are valid.



- Primary: proposition y_{ij} for line (i, j) on/off.
- Secondary: generation (controllable), voltage & power flow, bus fed status.
- Constraints:
  - Power flow equations
  - Generation and line limits
  - Goal: target buses fed
- Switched constraints:
  - $y_{ij} \rightarrow p_{ij} = \bar{g}_{ij}(v_{Ri}^2 + v_{Ii}^2) - \bar{g}_{ij}(v_{Ri}v_{Rj} + v_{Ii}v_{Ij}) - \bar{b}_{ij}(v_{Ii}v_{Rj} - v_{Ri}v_{Ij})$
  - $y_{ij} \rightarrow p_{ji} = \ldots$
  - $\neg y_{ij} \rightarrow p_{ij} = q_{ij} = p_{ji} = q_{ji} = 0$

Example: Vehicle routing

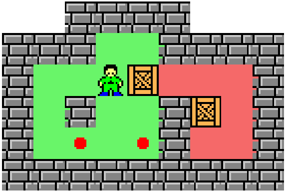
- Trucks carry goods from depots to customers.
- Capacity constraints.
- Possibility of trans-shipments.
- Time constraints are not modelled.



- Primary: $\mathrm{at}_v \in \mathrm{Loc}$, $\mathrm{visit}_{v,l} \in \{T, F\}$, $\mathrm{xs}_{v,v'} \in \{T, F\}$.
  - > Move v to l sets $\mathrm{visit}_{v,l} := T$
  - > Move v to X first sets $\mathrm{xs}_{v',v} := T$ if $\mathrm{visit}_{v',X}$, and $\mathrm{xs}_{v,v'} := T$ if $\mathrm{at}_{v'} = X$
- Goal constraints:
  - > $\sum_v x_{v,s,l} = \bar{q}_{s,i}$, $s \in \mathrm{Dep}$, $l \in \mathrm{Cust}$
  - > $\sum_s x_{v,s,l} = \sum_{v'} y_{v',v}$, $\mathrm{dep}(v) \neq s$, $\mathrm{dep}(v') = s$
  - > $\sum_{s,l} x_{v,s,l} + \sum_{v'} y_{v,v'} \leq \bar{c}_v$
  - > $\neg\mathrm{visit}_{v,l} \rightarrow x_{v,s,l} = 0$
  - > $\neg\mathrm{xs}_{v,v'} \rightarrow y_{v,v'} = 0$
  - > $\mathrm{at}_v = dep(v)$.

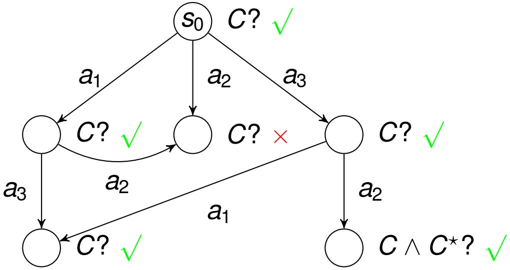
Example: Axioms

* PDDL axioms are logical state constraints, connecting primary and secondary (derived) finite-domain variables.
* Domain axioms are a stratified ASP theory.



$$\text{reach}_l \leftarrow \text{at}_l$$
$$\text{reach}_l \leftarrow \text{reach}_{l'} \wedge \text{adj}_{l',l} \wedge \text{not blocked}_l$$
$$\text{blocked}_l \leftarrow \text{stone}_l$$
$$\text{blocked}_l \leftarrow \text{wall}_l$$

Search with state constraints

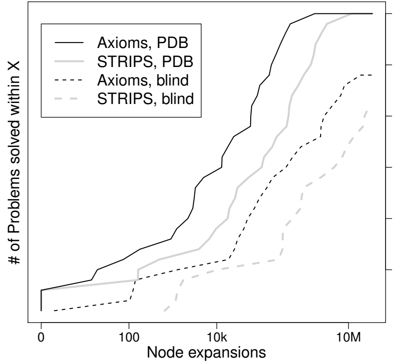


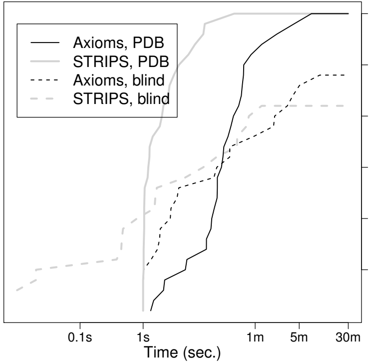
- A* search over the primary state-space, but also solving a CSP (of some kind) for every state.
- If it's "Benders'-like", where are the cuts?

Complexity

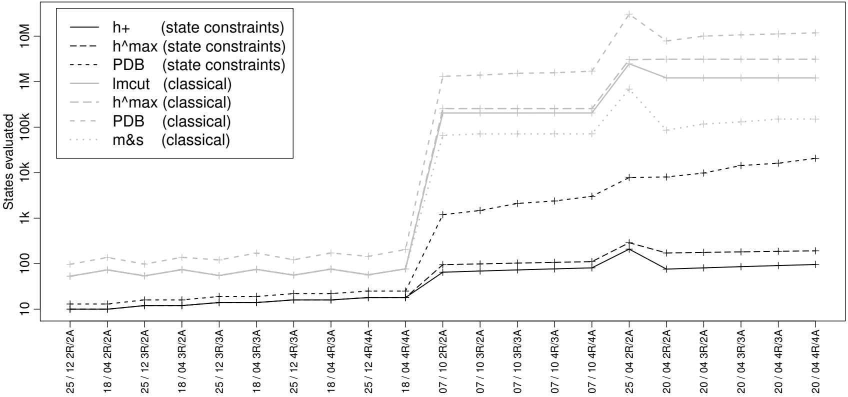
* Observation (due to Peter Jonsson): (ground) classical planning remains in PSPACE even if deciding the validity of a transition (s, a, s') is PSPACE-hard.
* Secondary variables are “existential”
  > Satisfiability is a function of the primary state.
  > Can be reduced to a plain STRIPS/SAS problem, with exponential blow-up.
* Switched linear constraints can encode arbitrary formulas over the primary state – blow-up is unavoidable (Nebel, 2000).

Sokoban (with and without axioms)





Vehicle routing (no trans-shipment, but two goods types)



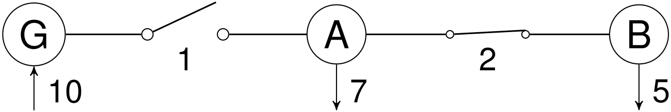
Heuristics

* Classical planning heuristics applied to the primary state only are blind to implicit preconditions – even goals – imposed by state constraints.
* How to make them constraint-aware?
* Admissible planning heuristics are generally based on optimally solving a problem relaxation.
  > Monotone (“delete-free”) relaxation.
  > Abstraction.

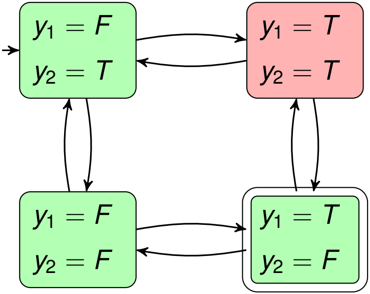
Constraints on relaxed states

* Monotone relaxation:
  > State variables have a set of values;
  > action effects add values to this set.
* Abstraction
  > Projection onto a subset of state variables.
  > Other variables are ignored; they can be assumed to have any value.
* A relaxed (abstract) state s^+ corresponds to a *set* of states; constraints are satisfiable in s^+ iff satisfiable in any state in this set.

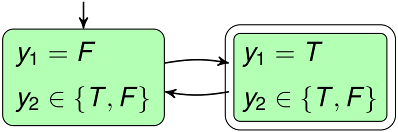
Example



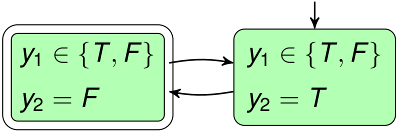
$$
\begin{array}{rcrcl}
y_1 & \rightarrow & f_G & = & f_A \\
y_2 & \rightarrow & f_A & = & f_B \\
 & & f_G & = & 1 \\
\neg y_1 & \rightarrow & p_{GA} & = & p_{AG} \;=\; 0 \\
\neg y_2 & \rightarrow & p_{AB} & = & p_{BA} \;=\; 0 \\
 & & p_{GA} + p_{BA} & = & 7f_A + p_{AG} + p_{AB} \\
 & & p_{AB} & = & 5f_B + p_{BA} \\
 & & p_{GA} & \leq & 10 \\
\text{Goal:} & & f_A & = & 1
\end{array}
$$



Project on $\{y_1\}$



Project on $\{y_2\}$



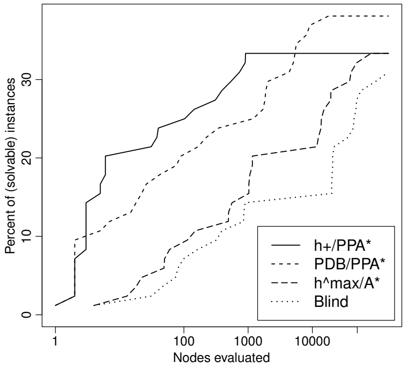
Weaker (tractable) relaxations

- Deciding constraint satisfiability in a relaxed state s^+ requires solving a CSP over both primary (discrete) and secondary variables.
- Relaxation is sound (admissible) as long as s^+ is declared invalid/non-goal *only* if constraints are unsatisfiable in *all* corresponding states.
  - Apply only (tractable) constraint propagation (Francés & Geffner).
  - Discard all switched constraints whose triggering condition is not necessarily true in the relaxed state.

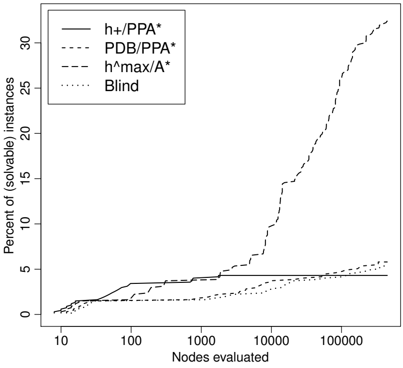
Results

Node (heuristic) evaluations

PSR

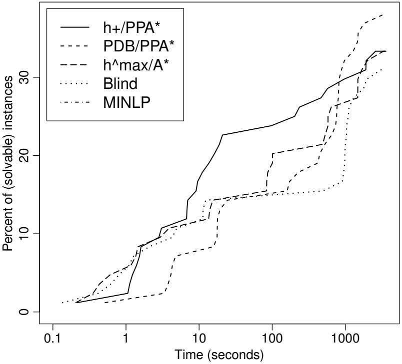


Vehicle routing

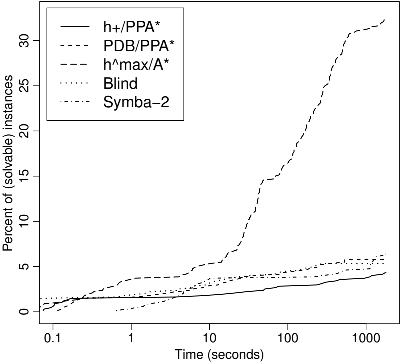


Runtime

PSR



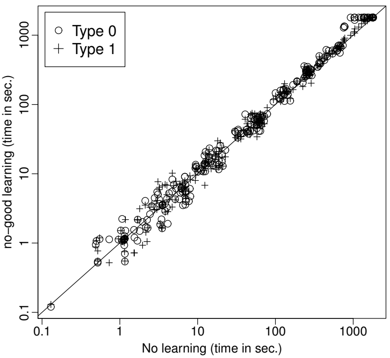
Vehicle routing



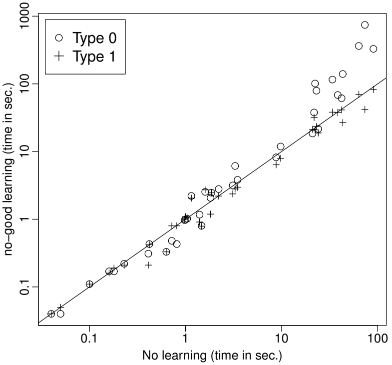
No-good learning

- Type 0
  - When encountering an invalid state *s*, extract a (small) condition ϕ on *s* primary that is sufficient to make *C* unsatisifable.
  - Test no-goods in future states to avoid calling constraint solver.
- Type 1
  - Regress ϕ through the action that led to *s*, and test before generating successor states.
- Note: Did not consider learning *goal* conditions.

PSR



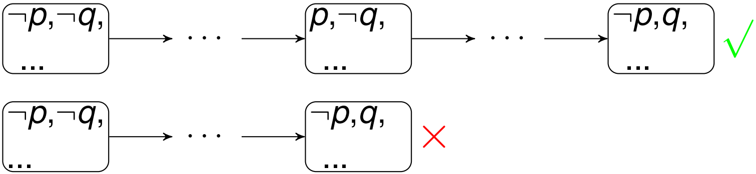
Hydraulic Blocksworld



Part 2

Trajectory constraints

* Constraints on the sequence of states visited by the plan.
* For example, "*p* must hold sometime before *q*":



* Plan constraints (introduced in PDDL3) are a limited subset of Linear Temporal Logic (LTL).

Plan constraints (PDDL3 & extra)

Fα	α true in final state
Aα	α true in all states
Eα	α true in some states
α SB β	α true in some state strictly before the first state where β true, or β never true
α SA β	α true in some state after last state where β true, or $\alpha \wedge \beta$ in that state
AMOα	α true in at most one contiguous subseq of states

Nα	$\neg E\alpha$
α NA β	α false in every state after first state where β true

where α and β are *state* formulas.

Example: Story variations

* Encode the events of a story (*The Illiad*) as actions.
* The original story is one possible plan.
  > C_T: A set of trajectory constraints that are true of this plan.
  > C_F: A set of trajectory constraints that are false of this plan.
* Sample $S \subset C_T$ and $D \subset C_F$, try to find a plan that satisfies $S \cup D$. Repeat many times.
* How to (quickly) filter out unsatisfiable (w.r.t. planning problem) constraint sets?

(achilles-and-agamemnon-quarrel)
(zeus-tricks-agamemnon-into-attacking-troy)
(issues-a-challenge paris)
(challenge-taken-up menelaus paris)
(single-combat-ends-divine-intervention paris menelaus)
(athena-tricks-tojans-into-breaking-peace)
(trojans-driven-back-to-walls-of-troy)
(hector-talks-with-his-wife)
. . .
(battle-begins-at-dawn)
(achaeans-driven-back-to-plain)
(achaeans-driven-back-to-wall)
(battle-ends-at-nightfall)
. . .
(priam-holds-funeral-for-hector)

In C_T: E (wounded hector)
E (dead sarpedon)
(trojans-losing) SB (night)
(fighting hector) SB (night)
(night) SB (achaeans-losing)
(not (wounded hector)) SA (wounded hector)
...

In C_F: A (not (dead sarpedon))
(achaeans-losing) SB (fighting hector)
(night) SB (trojans-losing)
AMO (battle-at-walls-of-troy)
...

Example: Bounding preferences

* Given a set of of weighted soft trajectory constraints, what is the max weight subset that is simultaneously satisfiable by any plan?
* To compute an upper bound,
  * find unsatisfiable subsets of constraints; and
  * solve a weighted hitting set problem.
* How to (quickly) determine if constraint subsets are unsatisfiable (w.r.t. planning problem)?

Plan constraint propagation

- Rules for inferring new constraints – or a contradiction – from sets of constraints.
- Rules form an algorithm that is (almost) polynomial in the size of the constraint set.
- Satisfiability w.r.t. planning problem: Extract from the problem plan constraints that are necessarily true of every executable action sequence, and test in conjunction.
- Resulting test is *fast* and *sound*, but not complete.

Extraction

- Constraint extraction from the problem can use a variety of relaxations (e.g., delete-free).
  - E.g., p SB q if p is a (causal) landmark of q.
- Conditional constraints:
  - Nα and pre(a) $\rightarrow \alpha$ or eff(a) $\rightarrow \alpha$ means a can not be part of any plan (Da, "disallow a").
  - Disallowing more actions can make more constraints hold: e.g., disallowing $\{a \mid p \in \text{add}(a)\} - \{a \mid q \in \text{pre}(a)\}$ implies q SB p.

Propagation Algorithm

* PROPAGATE(C, X)
  > Input: Sets of trajectory constraints (C) and conditional constraints (X).
  > Returns: Contradiction, or extended set of constraints (optionally: proof).
* Inferred constraints only over state formulas present in the input.
* Separate procedure for checking contradiction with AMO constraints.

1. Transitivity over SB and $\rightarrow$:
   - α SB β and β SB γ implies α SB γ.
   - α SB β and $\gamma \rightarrow \beta$ implies α SB γ.
   - $\beta \rightarrow \alpha$ and β SB γ implies α SB γ.
2. Aα and mutex(α, β) implies Nβ.
3. α SB β and β SB α implies Nα.
4. α SB β and Nα implies Nβ.
5. α SB β and β NA α implies Nβ.
6. Aα and del(a) negates α implies Da.
7. Nα and pre(a) $\rightarrow \alpha$ or add(a) $\rightarrow \alpha$ implies Da.
8. For $\langle \varphi, A \rangle \in X$, when D$a$ for all $a \in A$ assert φ.

∗ Rules 1–7 are iterated until fixpoint.

8. EG, where G is the goal.

9. α SB β and Eβ implies Eα.

10. $\alpha \rightarrow \beta$ and Eα implies Eβ.

∗ Rules 9–10 are iterated until fixpoint.

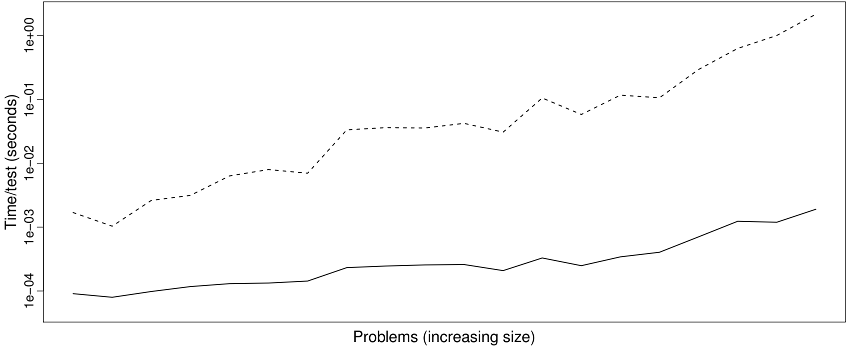
11. Eα and Nα is a contradiction.

12. Eα, Eβ, α NA β and β NA α is a contradiction.

13. CheckAMO(C, D).

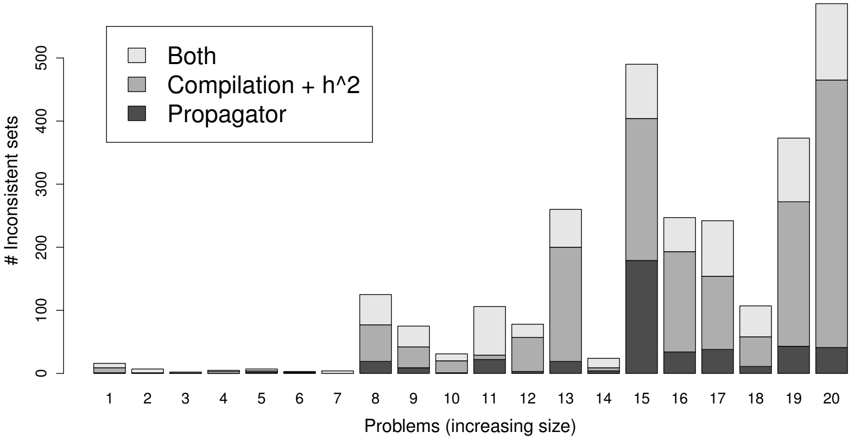
* AMOα implies plan can include:
  > at most one action changing α from true to false (ActChF(α));
  > at most one action changing α from false to true (ActChT(α)); none if α initially true.
* Find a set of atoms R' such that E$p \in C$ $\forall p \in R'$ and no action adds more than one atom in R'.
* Find a set $S = \{A_1, \ldots, A_n\}$ such that $\{a \notin D \mid p \in R', p \in \text{add}(a)\}$ is covered by $\bigcup_{A_i \in S} A_i$ and each $A_i =$ ActChF(α) or $A_i =$ ActChT(α)) for some AMO$\alpha \in C$.
* If $|R'| > |S|$, this is a contradiction.

Results: Runtime

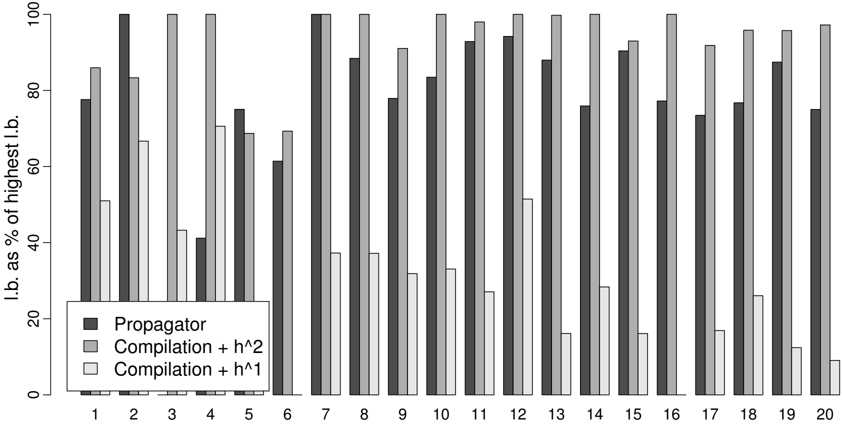


* Alternative: Compiling constraints into planning problem and checking unsolvability with h^2.

Results: Inconsistent sets



Results: Bounds



Conclusions

* “Pinching one idea is plagiarism. Pinching two is research.”
* Combining solvers
  > Interfaces: what is required of/by the “other”?
* ...and combining ideas.