Quality of Solutions to IPC5 Problems – Preliminary Results and Observations

P@trik Haslum

NICTA & ANU

ICAPS'07 Workshop on the Planning Competition

Australian Government

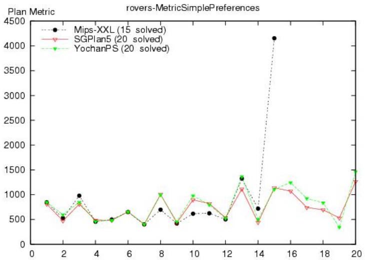
Department of Communications, Information Technology and the Arts

Australian Research Council

NICTA Members

NICTA Partners

Motivation



Plan Quality, Rovers MSP:
All planners are roughly equal – but are they equally *good* or equally *bad*?

- 5th IPC: emphasis on plan quality in evaluation.
- But: optimal solutions (or good bounds) not known, so only relative quality compared.
- Find optimal solutions and/or good quality bounds, using domain-specific methods, for some IPC-5 domains.

Domains Considered

IPC5 Classification

- Propositional:
  - Openstacks
- Metric/Temporal:
  - Openstacks Time
  - Openstacks MetricTime
- Simple Preferences:
  - Openstacks SP
  - Rovers MSP
- Qualitative Preferences:
  - Openstacks QP
  - Rovers QP

Classification by Objective Fn.

- Plan cost (1-objective):
  - Openstacks (# actions)
  - Openstacks Time (makespan)
- Plan cost (2-objective trade-off):
  - Openstacks MetricTime
- End-state value ("soft goals"):
  - Openstacks SP
- Plan cost/goal-value trade-off:
  - Openstacks QP
  - Rovers MSP
- Trajectory preferences:
  - Rovers QP

Conclusions

1. There isn't enough data to support that many conclusions.
2. The quality of plans produced by (some) competitors appears somewhat "accidental".
3. Domain and problem hardness:
    1. 2-objective trade-off functions appear more difficult to optimise.
    2. Relative plan quality does not appear to correlate with planner run-time.

Competing Planners by Domain

	Openstacks					Rovers	
	P	SP	QP	T	MT	MSP	QP
Downward'04-SA	$\surd$						
FDP	$\surd$						
HPlan-P			$\surd$				$\surd$
IPPLAN-G1SC	$\surd$						
MaxPlan	$\surd$						
MIPS-BDD	$\surd$						
MIPS-XXL	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
SGPlan_5	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
$\text{Yochan}^{\mathcal{PS}}$		$(\surd)$		$\surd$		$\surd$	

Conclusions

1. There isn't enough data to support that many conclusions.
2. The quality of plans produced by (some) competitors appears somewhat "accidental".
3. Domain and problem hardness:
   1. 2-objective trade-off functions appear more difficult to optimise.
   2. Relative plan quality does not appear to correlate with planner run-time.

The "Min Max Open Stacks" Problem

- Set of products to be made in sequence.
- Set of orders, each requesting a subset of products.
- An order is open from when the first requested product is made to when the last requested product is made: during this time, it uses a stack.
- Objective: sequence making of products to minimise the maximum number of stacks in use at any point.
- Trivial upper bound: # orders (one stack per order).
- Problem is NP-hard, and equivalent to several graph theory problems (*e.g.*, pathwidth).
- Constraint Modelling Challenge 2005 problem:
  - Large library of problem instances.
  - Several solvers, and data on their performance.

Openstacks: Example

sequence:	2	3	4	5	1	|	1	2	3	5	4
order 1 ($\{1,2\}$):	X	–	–	–	X	|	X	X			
order 2 ($\{1,3\}$):		X	–	–	X	|	X	–	X		
order 3 ($\{2,4\}$):	X	–	X			|		X	–	–	X
order 4 ($\{3,5\}$):		X	–	X		|			X	X	
order 5 ($\{4,5\}$):			X	X		|				X	X
# open stacks:	2	4	5	4	2	|	2	3	3	3	2

The Openstacks Domain

- PDDL encoding of the open stacks problem.
- Actions `(make-product` *p*`)`, `(start-order` *o*`)` and `(ship-order` *o*`)` must each be done exactly once:
  - `(start-order` *o*`)` before `(make-product` *p*`)` when *o* includes *p*,
  - `(make-product` *p*`)` before `(ship-order` *o*`)` when *o* includes *p*.
- How to count current/max number of stacks in use?
  - Stacks are a resource: `start-order` takes 1, `ship-order` returns 1...
  - 4 different formulations (only 1 used in IPC5).
- Problem set: 25 selected – for variety – from CMC library, plus 5 trivially small instances.

The Openstacks Domain

- "Plain" Formulation:
  - Propositional counter for # free stacks.
    ((stacks-avail n0), (stacks-avail n1), ...)
  - Action `open-new-stack` creates one (free) stack.
  - max # stacks in use
    = # `open-new-stack` actions in plan
    = plan length − (problem-dependent) constant.
- "Sequenced" Formulation (IPC5 Propositional):
  - However, min # actions objective can't be specified in "propositional PDDL"; default is "(total-time)".
  - Forced sequentiality: # actions equals # "time steps".
  - Larger plan length constant.

The Openstacks Domain

- "Numeric" Formulation:
  - Fluents track current and max # stacks in use:

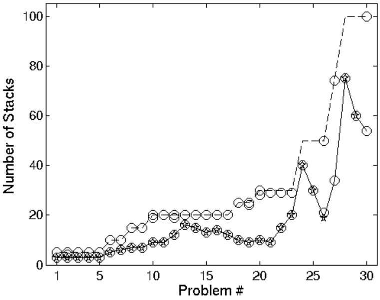
```
(and (increase (stacks-in-use) 1)
     (when (>= (stacks-in-use) (max-in-use))
           (increase (max-in-use) 1)))
```

  - `(:metric minimize (max-in-use))`
- "Preferences" Formulation:
  - Propositional counter for current # stacks in use.
  - PDDL3 trajectory preferences:

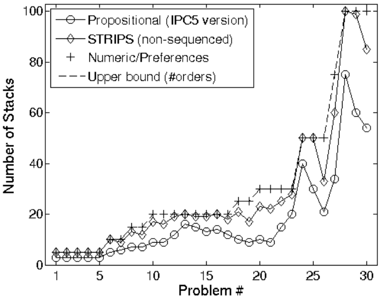
```
(and (preference p1
      (always (not (stacks-in-use n1))))
     (preference p2
      (always (not (stacks-in-use n2)))) ...)
```

  - `(:metric minimize (+ (is-violated p1) ...))`

Openstacks: Plan Quality



Competitor plans (○), best known (—) and upper bounds (- -). A star indicates solution is optimal.

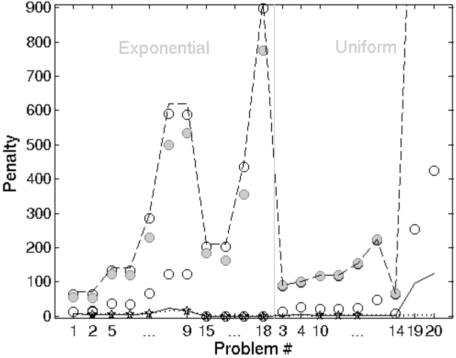


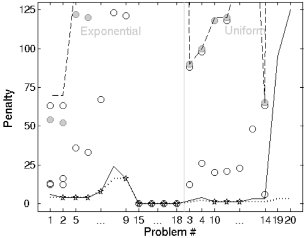
Plans found by SGPlan$_5$ on different domain formulations.

The Openstacks SP Domain

- Like Openstacks, but max # stacks in use is fixed and goals are soft: orders may be shipped without all requested products, but incur a penalty for missing products.
- Objective: minimise total penalty.
- Two formulations:
  - With conditional effects (used in IPC5):
    If *p* made while *o* is open, then *p* is "delivered" to *o*.
  - Without conditional effects:
    Explicit action `(deliver` *p* *o*`)` must take place while *o* is open and *p* is made (split `make-product` action).
- Problem instances:
  - Based on 20 selected CMC problems.
  - Max # stacks fixed slightly below the (believed-to-be) minimum, to force selection of requests to satisfy.

Openstacks SP: Plan Quality





Closeup of "lower" region of the graph.

- In IPC5 formulation (with c.e.), SGPlan_5 consistently best.
- In non-c.e. formulation, SGPlan_5 consistently finds plans of worst possible quality!

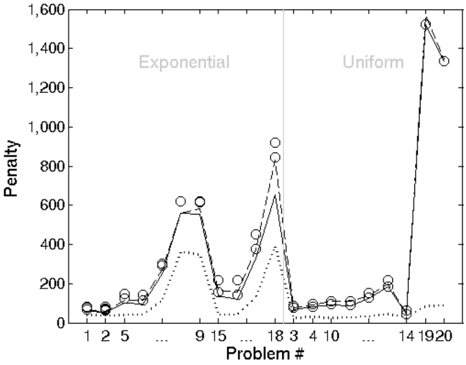
Conclusions

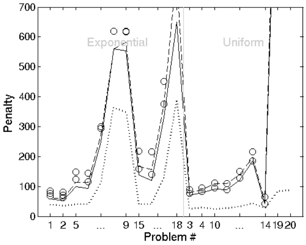
1. There isn't enough data to support that many conclusions.
2. The quality of plans produced by (some) competitors appears somewhat "accidental".
3. Domain and problem hardness:
   1. 2-objective trade-off functions appear more difficult to optimise.
   2. Relative plan quality does not appear to correlate with planner run-time.

The Openstacks QP Domain

- Combines the objectives of the Openstacks and Openstacks SP domains: minimise sum of
  - penalty for unsatisfied product requests, plus
  - max # stacks used times (problem-specific) price / stack.
- IPC5 formulation uses:
  - conditional effects (as in Openstacks SP),
  - trajectory preferences to track max # stacks used.
- Aimed to set price / stack so "extreme" plans have equal value...
  - however, turned out stacks are somewhat "overpriced";
  - a simple, greedy single-stack construction finds plans of quality close to best known – and often better than competitors' – plans.

Openstacks QP: Plan Quality





Closeup of "lower" region of the graph.

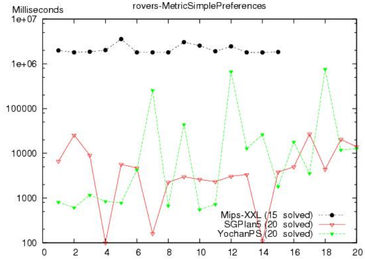
Competitor plans (○), best known (—), upper (- -) and lower (· · ·) bounds.

Conclusions

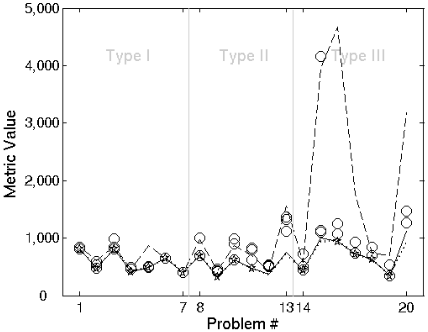
1. There isn't enough data to support that many conclusions.
2. The quality of plans produced by (some) competitors appears somewhat "accidental".
3. Domain and problem hardness:
   1. 2-objective trade-off functions appear more difficult to optimise.
   2. Relative plan quality does not appear to correlate with planner run-time.

Rovers MSP: CPU Time *vs.* Plan Quality

NICTA



CPU time taken by planners in the competition.



Competitor plans (○), best known (—), upper (- -) and lower (· · ·) bounds. A star indicates solution is optimal.

Lessons Learned

- A lot of work (and CPU time!) invested, for questionable "science return"...
- Specifics of problem instances matter!
  - Properties / "biases" of optimal solutions (*e.g.*, "overpriced" stacks in Openstacks QP).
  - Instances with unintended "flaws" (*e.g.*, Openstacks SP `p15`–`p18`).
- Encourage coverage!
  - Offer domains in different formulations.
  - Make coverage part of competition evaluation criteria.

All Results & Additional Resources

http://users.rsise.anu.edu.au/~patrik/ipc5.html