A*mbush family: A* Variations for Ambush Behavior and Path Diversity Generation

Kelwin Fernández, Glebys González and Carolina Chang

Universidad Simón Bolívar
Caracas, Venezuela

Motivation

- STRIPS is PSPACE-complete, **no known algorithm** that **automatically** generates a STRIPS problem from an arbitrary PSPACE problem.
- General Problem Solver for PSPACE problems:
  - Described in a high-level declarative language
  - Taking advantage of current and future planning technology
- Design **benchmark problems for planning** to evaluate heuristics and algorithms.

Idea

First step to build such a tool: translating **NP problems** encoded as SO∃ sentences into planning. The output planning task should be **no more difficult** to solve than the original problem.

Case study: SAT

Predefined relations (signature)

- $P(x, y)$: The variable x appears **positive** in clause y.
- $N(x, y)$: The variable x appears **negative** in clause y.

Existentially quantified relations

- $T(x)$: The variable x is set to $true$.

Formula

- Find which variables should be assigned to **true** in order to satisfy a CNF formula.
- In **every** clause **at least one** variable is satisfied. A variable x is *satisfied* inside a clause y if its truth value T is coherent with its sign, i.e. if it appears as positive, $T(x)$ should be true, if it appears as negative, $T(x)$ should be false.

$$(\exists T^1)(\forall y)(\exists x)[(P(x, y) \land T(x)) \lor (N(x, y) \land \neg T(x))]$$

Finite structures (example)

- Objects are numbered from `zero` to `max`
- A structure holds the information required to specify a **problem instance**

$$\underbrace{(x_0 \lor \neg x_1 \lor x_2)}_{\text{clause 0}} \land \underbrace{(\neg x_0 \lor \neg x_2)}_{\text{clause 1}} \land \underbrace{(\neg x_0 \lor x_1)}_{\text{clause 2}}$$

$$\Downarrow$$

```
(?P zero zero)  (?N obj1 zero)
(?N zero obj1)  (?N max obj1)
(?N zero max)   (?P obj1 max)
(?P max zero)
```

Domain translation

```
(define (domain SAT)
  (:constants zero max)
  (:predicates
    (holds_and_2 ?x ?y) (holds_and_6 ?x0 ?x1)
    (holds_exists_8 ?x0) (holds_forall_9 ?x0)
    (holds_or_7 ?x0 ?x1) (holds_goal)
    (N ?x ?y) (P ?x ?y) (T ?x) (not-T ?x)
    (suc ?x ?y)
  )
  (:action set_T_true
    :parameters (?x)
    :precondition (and (guess) (not-T ?x))
    :effect (and (T ?x) (not (not-T ?x))))

  (:action prove_forall_9_1
    :precondition (and (proof)
                       (holds_exists_8 zero))
    :effect (holds_forall_9 zero))

  (:action prove_forall_9_2
    :parameters (?y1 ?y2)
    :precondition (and (proof)
                       (suc ?y1 ?y2)
                       (holds_forall_9 ?y1)
                       (holds_exists_8 ?y2))
    :effect (holds_forall_9 ?y2))

  (:action prove_exists_8
    :parameters (?y ?x)
    :precondition (and (proof)
                       (holds_or_7 ?y ?x))
    :effect (holds_exists_8 ?y))

  (:action prove_or_7_0
    :parameters (?y ?x)
    :precondition (and (proof)
                       (holds_and_2 ?y ?x))
    :effect (holds_or_7 ?y ?x))

  (:action prove_or_7_1
    :parameters (?y ?x)
    :precondition (and (proof)
                       (holds_and_6 ?y ?x))
    :effect (holds_or_7 ?y ?x))

  (:action prove_and_2
    :parameters (?y ?x)
    :precondition (and (proof)
                       (P ?x ?y) (T ?x))
    :effect (holds_and_2 ?y ?x))

  (:action prove_and_6
    :parameters (?y ?x)
    :precondition (and (proof)
                       (N ?x ?y) (not-T ?x))
    :effect (holds_and_6 ?y ?x))

  (:action prove-goal
    :precondition (holds_forall_9 max)
    :effect (holds_goal))

  (:action begin-proof
    :precondition (guess)
    :effect (and (proof) (not (guess)))) )
```

Solution (example)

$$S = \{T(x_0), T(x_1)\}$$

Reduction

- If the first-order structure (**instance**) satisfies the property expressed by the SO∃ formula (**domain**), the plan is a step-by-step proof of this fact
- Actions to guess the truth value of the second-order relations
- Actions to prove the logical operators
- Guess truth values first, then attempt to build the proof

Experiments

We modeled, translated and attempted to solve several NP-complete problems using our tool and state-of-the-art planners.

Experimental results

- The time limit was 30 minutes, and the memory limit was 1 GB
- The M and Mp planners by Jussi Rintanen performed best
- 1,614 instances out of 1,920 were solved

	N^*/N	#pos.	#neg.	avg. time
SAT				
uf20	40/40	40	0	1.7
uf50	40/40	40	0	146.7
uf75	15/40	15	0	362.1
uuf50	40/40	0	40	548.5
uuf75	1/40	0	1	1,746.4
Clique				
25-3	40/40	30	10	111.9
25-4	40/40	18	22	231.0
25-5	39/40	10	29	387.5
25-6	36/40	8	28	394.1
Hamiltonian Path				
30	22/40	20	2	629.1
3-dimensional Matching				
20	13/40	13	0	1,191.0
25	0/40	0	0	—
3-colorability				
50	40/40	1	39	196.7
k-Colorability				
20-2	40/40	3	37	254.9
20-3	40/40	12	28	395.9
20-4	40/40	20	20	497.3
25-2	0/40	0	0	—
25-3	0/40	0	0	—
25-4	0/40	0	0	—
Total	1,614/1,920	706	908	180.9

Chromatic numbers

Using the k-colorability domain provided by the tool, we determined the chromatic number χ of random graphs.

		k-colorability						
instance	χ	1	2	3	4	5	6	7
10-0.75-1	5	2	2	6	101	**3**		
10-0.75-2	5	1	2	2	6	**4**		
10-0.85	7	2	2	3	6	4	1,265	**4**
15-0.25	2	27	**62**					
15-0.60	5	27	29	54	118	**72**		
15-0.70	6	28	28	33	47	329	**67**	
20-0.10	3	214	350	**705**				
20-0.25	4	211	272	1,261	**837**			

Discussion

- New way of expressing decision problems to solve them using planning technology
- Automatic, efficient reductions from declaratively-expressed PSPACE problems into STRIPS?
- Computation of of transitive closures for predicates definable in second-order logic
- Target other complexity classes first? SO∃∀