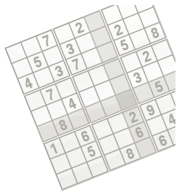


Constraint Programming

Roman Barták
Department of Theoretical Computer Science and Mathematical Logic

Modelling Planning Problems

Planning Problem

We will deal with **classical AI planning**

- looking for the (shortest) sequence of actions (a **plan**) transferring the initial state of the world to a state satisfying some goal condition
- **state** is described using a set of **multi-valued variables**
- (grounded) **action** is specified by:
  - **precondition** (required values of certain state variables before action execution)
  - **effect** (changed values of certain state variables after action execution)

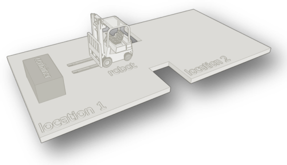
Example Problem

State Variables

rloc ∈ {loc1,loc2} ;; robot's location
cpos ∈ {loc1,loc2,r} ;; container's position

Actions

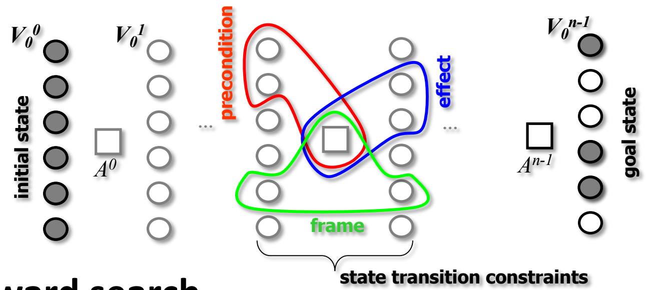
move(r, loc1, loc2) ;; robot r at location loc1 moves to location loc2
  Precond: *rloc* = loc1
  Effects: *rloc* ← loc2
move(r, loc2, loc1) ;; robot r at location loc2 moves to location loc1
  Precond: *rloc* = loc2
  Effects: *rloc* ← loc1
load(r, c, loc1) ;; robot r loads container c at location loc1
  Precond: *rloc* = loc1, *cpos* = loc1
  Effects: *cpos* ← r
load(r, c, loc2) ;; robot r loads container c at location loc2
  Precond: *rloc* = loc2, *cpos* = loc2
  Effects: *cpos* ← r
unload(r, c, loc1) ;; robot r unloads container c at location loc1
  Precond: *rloc* = loc1, *cpos* = r
  Effects: *cpos* ← loc1
unload(r, c, loc2) ;; robot r unloads container c at location loc2
  Precond: *rloc* = loc2, *cpos* = r
  Effects: *cpos* ← loc2



Core Modeling Approach

Iterative extension of the plan length

Formulating the problem of finding a plan of a given length as a CSP



Backward search

- instantiation of action variables
- only actions relevant to the (sub)goal are tried

Ghallab et al. (2004)

Straightforward Model

original formulation

- action constraints

  $A^s = move21 \rightarrow rloc^s = loc2$
  $A^s = move21 \rightarrow rloc^{s+1} = loc1$

  $A^s = act \rightarrow \mathrm{Pre}(act)^s$, $\forall act \in \mathrm{Dom}(A^s)$
  $A^s = act \rightarrow \mathrm{Eff}(act)^{s+1}$, $\forall act \in \mathrm{Dom}(A^s)$
- frame constraint

  $A^s = move21 \rightarrow cpos^s = cpos^{s+1}$

  $A^s \in \mathrm{NonAffAct}(V_i) \rightarrow V_i^s = V_i^{s+1}$, $\forall i \in \langle 0, v\text{-}1\rangle$

problems

- disjunctive constraints do no propagate well
  - ↳ **do not prune well the search space**
- a huge number of constraints (depend on the number of actions)
  - ↳ **the propagation loop takes a lot of time**

Barták & Toropila (2008)

Model Reformulation

idea

- encapsulate the logical constraints into a table constraint describing allowed tuples of values
- be careful about the size of the table!

reformulated straightforward model

- action constraint = a single table

A^s	$rloc^s$	$cpos^s$	$rloc^{s+1}$	$cpos^{s+1}$
move21	loc2		loc1	
move12	loc1		loc2	
load1	loc1	loc1		r
…				

- frame constraint

  $A^s \in \mathrm{NonAffAct}(V_i) \rightarrow V_i^s = V_i^{s+1}$, $\forall i \in \langle 0, v\text{-}1\rangle$

Lopez & Bacchus (2003)

CSP-PLAN

idea

- focus on modeling the reason for the value of a state variable (effect and frame constraints are merged)

original model

- precondition constraint

  $A^s = act \rightarrow \mathrm{Pre}(act)^s$, $\forall act \in \mathrm{Dom}(A^s)$
- successor state constraint

  $V_i^s = val \leftrightarrow A^{s-1} \in C(i,val) \vee (V_i^{s-1} = val \wedge A^{s-1} \in N(i))$
  - $C(i,val)$ = the set of actions containing $V_i \leftarrow val$ among their effects
  - $N(i) = \mathrm{NonAffAct}(V_i)$

reformulated model

- use a single table constraint to describe preconditions
- use ternary table constraints to describe successor state constraints (one table per state variable)

CSP-PLAN Constraints

Table for precondition constraint

A^s	$rloc^s$	$cpos^s$
1: move(r, loc1, loc2)	loc1	{…}
2: move(r, loc2, loc1)	loc2	{…}
3: load(r, c, loc1)	loc1	loc1
4: load(r, c, loc2)	loc2	loc2
5: unload(r, c, loc1))	loc1	r
6: unload(r, c, loc2)	loc2	r

Tables for successor state constraints

A^s	$rloc^s$	$rloc^{s+1}$
2	{…}	loc1
1	{…}	loc2
{3,4,5,6}	loc1	loc1
{3,4,5,6}	loc2	loc2

A^s	$cpos^s$	$cpos^{s+1}$
5	{…}	loc1
6	{…}	loc2
{3,4}	{…}	r
{1,2}	loc1	loc1
{1,2}	loc2	loc2
{1,2}	r	r

Model Comparison

The total **number of constraints**

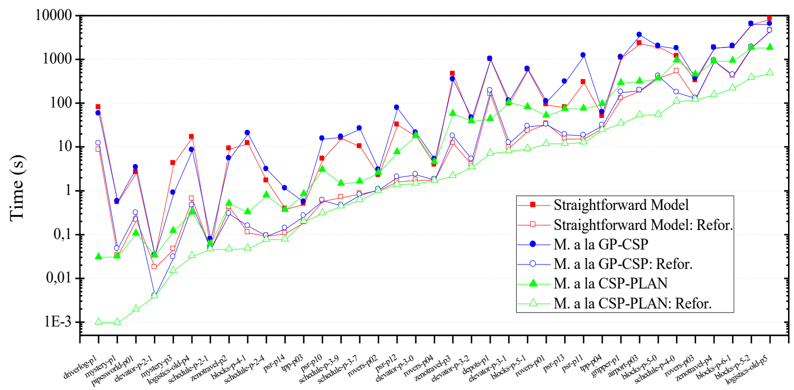
	original	**reformulated**
straightforward	n(ap+ae+v)	n(1+v)
GP-CSP	n(ap+ae+3v)	n(1+3v)
CSP-Plan	n(ap+vd)	n(1+v)

n - number of actions in the plan
a - number of grounded actions in the problem
v - number of multi-valued variables
p - average number of preconditions per action
e - average number of effects per action

Barták (2011)

Model Comparison

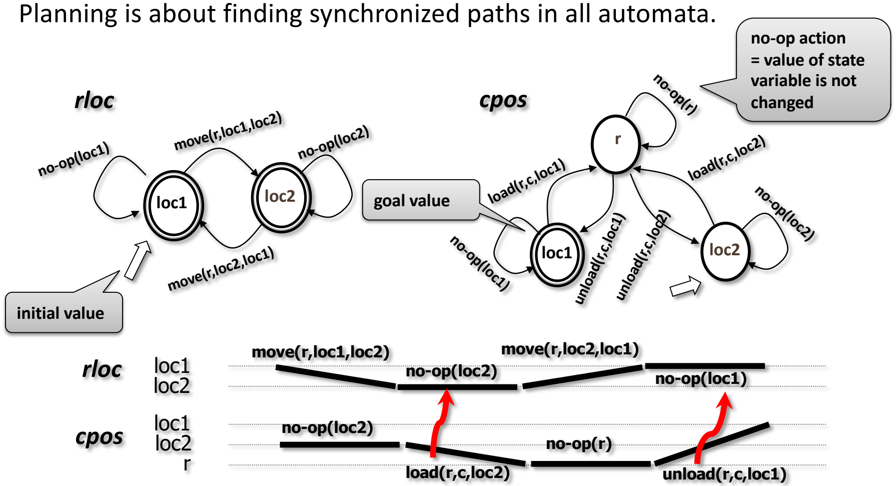
The **runtime** to solve selected problems from IPC 1-5 (logarithmic scale)



Barták (2011)

Timelines

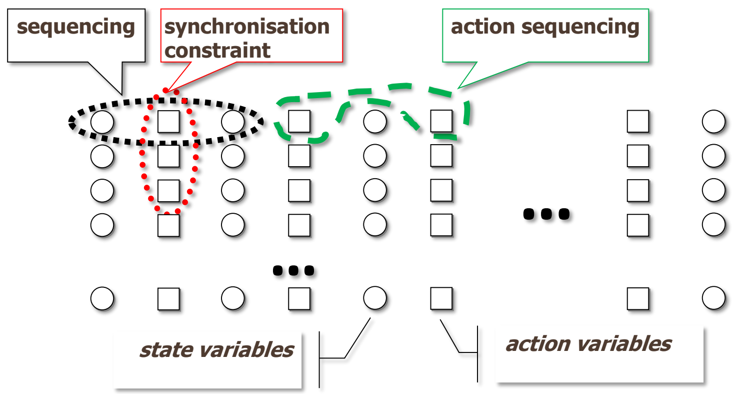
Planning can also be seen as **synchronized changes of state variables**.
Evolution of each variable is described using finite state automaton.
Planning is about finding synchronized paths in all automata.



PaP: Constraint Model

timeline model

state and action variables organized to „layers“



Barták (2011)

Search Strategy

a more or less standard CP labeling procedure instantiating (by the search algorithm) only the **action variables**

– the state variables are instantiated by inference

- **variable selection**
  – dom heuristic (only variables with real action in their domain are assumed)
- **value selection** (in two steps)
  – split the domain into no-op actions (explored first) and real actions
  – domains with real actions only are enumerated then

Barták (2011)

Summary Results (#solved problems)

planning domain	SeP	PaP
airport (15)	4	**6**
blocks (16)	**7**	**7**
depots (10)	**2**	**2**
driverlog (15)	4	**12**
elevator (30)	**30**	27
freecell (10)	1	**3**
openstacks (7)	**5**	0
rovers (10)	4	**6**
tpp (15)	4	**8**
zenotravel (15)	6	**11**

problems from International Planning Competition, runtime limit 30 minutes

Barták (2011)

Detailed Results (runtime)

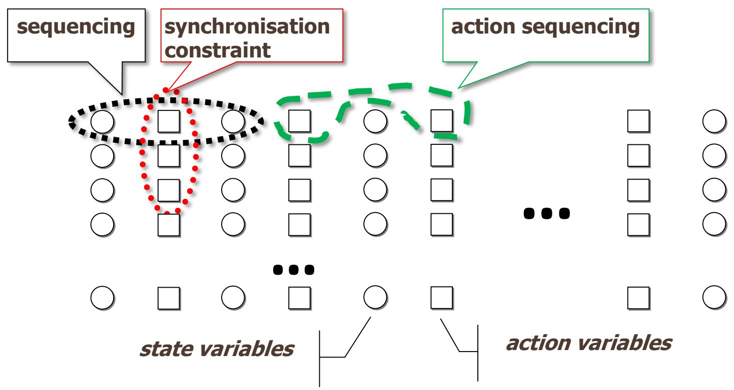
problem	plan length SeP	plan length PaP par	plan length PaP seq	runtime (ms) SeP	runtime (ms) PaP
zenotravel-p01	1	1	1	**10**	20
zenotravel-p02	6	5	6	60	**50**
zenotravel-p03	6	5	9	300	**130**
zenotravel-p04	8	5	11	970	**130**
zenotravel-p05	11	5	14	153 990	**240**
zenotravel-p06	11	5	12	530 390	**510**
zenotravel-p07	≥12	6	16	-	**560**
zenotravel-p08	≥10	5	15	-	**1 690**
zenotravel-p09	≥11	6	24	-	**145 760**
zenotravel-p10	≥12	6	24	-	**252 040**
zenotravel-p11	≥9	6	16	-	**41 780**

Barták (2011)

PaPaP2: Constraint Model

timeline model

state and action variables organized to „layers“



Summary Results (#solved problems)

planning domain	SeP	PaP	PaP-2
airport (15)	4	6	8
depots (10)	2	2	2
driverlog (15)	4	12	13
elevator (30)	30	27	30
freecell (10)	1	3	3
openstacks (7)	5	0	0
rovers (10)	4	6	7
tpp (15)	4	8	8
zenotravel (15)	6	11	12

Detailed Results (runtime)

problem	plan length			runtime (ms)		
	SeP	PaP		SeP	PaP	PaP-2
		par	seq			
zenotravel-p01	1	1	1	10	20	30
zenotravel-p02	6	5	6	60	50	60
zenotravel-p03	6	5	9	300	130	140
zenotravel-p04	8	5	11	970	130	160
zenotravel-p05	11	5	14	153 990	240	320
zenotravel-p06	11	5	12	530 390	510	350
zenotravel-p07	≥12	6	16	-	560	440
zenotravel-p08	≥10	5	15	-	1 690	1 340
zenotravel-p09	≥11	6	24	-	145 760	2 260
zenotravel-p10	≥12	6	24	-	252 040	8 400
zenotravel-p11	≥9	6	16	-	41 780	3 250
zenotravel-p12	≥11	6	24	-	-	5 930

Conclusions

Take away messages:

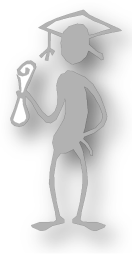
- constraint modeling is critical for efficiency
- models that prune more values are (usually) better
- search strategies can describe specific solving procedures

Can we do even better?

- Yes, definitely (for example see paper "Transition Constraints for Parallel Planning", AAAI 15)

Constraints elsewhere in planning?

- solving specific sub-problems
- temporal and resource reasoning



Course summary

Constraint satisfaction is a technology for **declarative solving combinatorial (optimization) problems**.

Constraint modeling

- describing problems as constraint satisfaction problems (variables, domains, constraints)

Constraint satisfaction

- local search techniques
- combination of depth-first search with inference (constraint propagation/consistency techniques)
- ad-hoc algorithms encoded in global constraints
- soft constraints to express preferences

It is easy to model problems in terms of a CSP
… but it is complicated to design solvable models.