Automated Action Planning

Explicit Planning Task Structure: Hybrid Abstraction/Relaxation Heuristics

Carmel Domshlak

Automated Action Planning
— Explicit Planning Task Structure: Hybrid Abstraction/Relaxation Heuristics

Finite Domain Representation (FDR) Language

Definition (FDR planning tasks)

An FDR planning task is a tuple $\langle V, A, I, G \rangle$

- V is a finite set of state variables with finite domains $dom(v_i)$
- initial state I is a complete assignment to V
- goal G is a partial assignment to V
- A is a finite set of actions a specified via pre(a) and eff(a), both being partial assignments to V

Definition (BDR planning tasks)

BDR planning tasks are FDR planning tasks with only boolean state variables.

Planning as State-Space Heuristic Search

Heuristic functions

What? Something that can be solved in polynomial time to assist us in solving our planning task

How? Solutions to simplifications of the planning task

Window of opportunity for computational tractability!

Structural fragments

Reminder: What are syntactic restrictions?

Fragment of tasks $\xleftarrow{\text{def}}$ restr. on action description

What are structural restrictions?

Fragment of task $\xleftarrow{\text{def}}$ restr. on interactions between actions

Graphical Structures as Problem Abstractions

- ⊛ Why graphs?
  1. Cognitively convenient
  2. Come with a rich math and CS toolbox

- Graphical representations/abstractions of comp. problems
  1. CSP: Constraint networks, junction trees, ...
  2. Probabilistic reasoning: BNs, DBNs, Markov nets, ...
  3. Preferential reasoning: GAI-nets, xCP-nets, ...

- Graphical views in planning?
  - Yes, we have!
  - Today: causal graphs & domain transition graphs
    ⊛ Why these?
  - More to be studied, and even to be discovered/suggested

Graphical Abstractions of Action Interactions

Causal Graphs

In the context of an FDR planning task $\Pi = \langle V, A, I, G \rangle$:

Definition (causal graph)

The causal graph $CG(\Pi)$ of Π is a digraph over nodes V.
An arc (v, v') is in $CG(\Pi)$ iff $v \neq v'$ and there exists an action $a \in A$ such that

$$(v, v') \in V(\mathsf{eff}(a)) \cup V(\mathsf{pre}(a)) \ \times \ V(\mathsf{eff}(a)),$$

that is, both $\mathsf{eff}(a)[v']$ and either $\mathsf{pre}(a)[v]$ or $\mathsf{eff}(a)[v]$ are specified.

Notation: $\mathsf{succ}(v)$ and $\mathsf{pred}(v)$ are immediate successors and predecessors of v in $CG(\Pi)$.

Graphical Abstractions of Action Interactions

Domain Transition Graphs

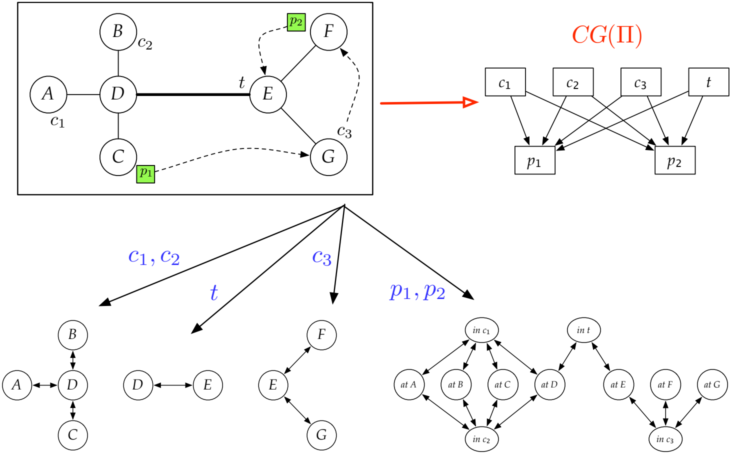
In the context of an FDR planning task $\Pi = \langle V, A, I, G \rangle$:

Definition (domain transition graph)

The domain transition graph $DTG(v, \Pi)$ of a variable $v \in V$ is an arc-labeled digraph over the nodes $dom(v)$.
An arc (d, d') labeled with $\in A$ is in the graph iff

1. $\mathrm{eff}(a)[v] = d'$, and
2. either $\mathrm{pre}(a)[v] = d$, or $v \notin V(\mathrm{pre}(a))$.

Example


A
B
C
D
E
F
G
c1
c2
c3
t
p1
p2
CG(Π)
c1, c2
t
c3
p1, p2
in c1
in t
at A
at B
at C
at D
at E
at F
at G
in c2
in c3

Computational Tractability as a Function of Causal Graph Form

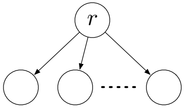
1. From BDR to FDR
2. From severe structural restrictions to their generalizations
3. For simplicity, assume all actions have the same cost (relevant only for optimization)

BDR Forks

⊛ Informal discussion

PlanGen is easy

- r's capabilities: 0, 1, or ∞ changes.
- All leafs are binary $\leadsto$ r changes ≤ 2.
- Given a workload of r, succ(r) are independent.



PlanMinGen is easy

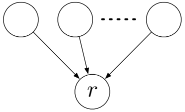
- Given root's workload, all leafs are independent.
- Optimize over all three cases of workload for root.

BDR Inverted Forks

⊛ Informal discussion

PlanGen is easy

- $\mathrm{pred}(r)$ are independent.
- if not trivial, r should change exactly once.
- find action a changing r to $G[r]$ such that, for each $v \in \mathrm{pred}(r)$, $G[v]$ reachable from $I[v]$ via $\mathrm{pre}(a)[v]$.



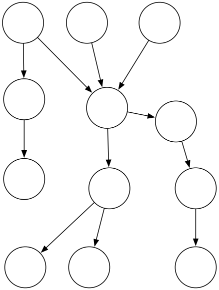
PlanMinGen is easy

- Optimize over all actions changing r to $G[r]$.

So far so good! What next?

Generalizing causal graph fragments

1. Forks $\Longrightarrow$ Directed Trees
2. Inverted Forks $\Longrightarrow$ Directed Inverted Trees
3. Directed Trees + Directed Inverted Trees $\Longrightarrow$ Polytrees



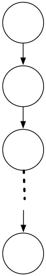
BDR Chains

⊛ Informal discussion

PlanGen is easy

loop

- iteratively eliminate leafs consistent with G
- change the lowest var that can be changed



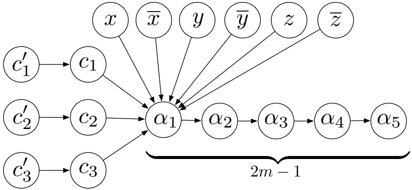
PlanMinGen is easy

- No choices $\rightsquigarrow$ Optimal.
- Same algorithm works for directed trees!
  What about choices? They are $\forall$, not $\exists$.

BDR Polytrees

NO... *PlanGen* is NP-complete [GJ08]

Elegant reduction from 3SAT (m clauses, n vars)



- Note that the proof kills directed inverted trees as well ...
- Can we push further with fixed in-degree?
  ⊛ Various alternative generalizations of polytrees.
- [BD03] For DP singly connected causal graphs, NP-complete starting (at most) in-degree 6.

FDR and Causal Graph Topology

PlanGen looks bad

- Forks $\rightsquigarrow$ NP-complete [DD01]
- Inverted Forks $\rightsquigarrow$ NP-complete [DD01]
- Chains $\rightsquigarrow$ NP-complete [GJ07]
- ⊛ Can we expect for any good news?

FDR and Causal Graph Topology

No, we can't.

Theorem (Chen & Gimenez classification [CG08])

Let $\mathcal{C}$ be a set of directed graphs, and $\mathbf{\Pi}^{\mathcal{C}}$ be the class of planning tasks Π with $CG(\Pi) \in \mathcal{C}$.

- *If the size of all connected components in graphs of $\mathcal{C}$ is bounded by a constant, then PlanGen for $\mathbf{\Pi}^{\mathcal{C}}$ is polynomial-time solvable.*
- *Otherwise, PlanExt for $\mathbf{\Pi}^{\mathcal{C}}$ is not polynomial-time decidable (unless* W[1] $\subseteq$ nu-FPT*)*

Why "unless W[1] $\subseteq$ nu-FPT" and not, say, "unless P $=$ NP"?

Situation Assessment

1. Looking at out benchmarks, natural state variables tend to be non-binary, and even parametric (wrt domain).
2. With binary state variables, we get messy causal graphs.
3. With finite-domain state variables, causal graph is irrelevant.
4. Q: *Have we wasted our time?* Maybe. Maybe not.

The Journey Continues!

Major conclusion so far

Causal graphs are too coarse to provide an effective tractability-oriented abstraction

Possible direction from here

- Look for additional constraints on top of the causal graph

The Journey Continues!

Major conclusion so far

Causal graphs are too coarse to provide an effective tractability-oriented abstraction

Reminder: *PlanGen* looks bad

- Chains $\rightsquigarrow$ NP-complete
- Forks $\rightsquigarrow$ NP-complete
- Inverted Forks $\rightsquigarrow$ NP-complete

Note: all three are easy for BDR!
What about non-binary, yet still small, $O(1)$, domains?

Back to Chains

What happens with chain-structured tasks if $|dom(v)| = O(1)$ for all vars?

- 2001/DD $|dom(v) = 3| \mapsto$ Optimal plans can be exponentially long
- 2002/BD $|dom(v)| = 2 \mapsto$ Polynomial-time solvable
- 2009/GJ $|dom(v)| = 5 \mapsto$ NP-complete

Back to Chains

What happens with chain-structured tasks if $|dom(v)| = O(1)$ for all vars?

2001/DD $|dom(v) = 3| \mapsto$ Optimal plans can be exponentially long

2002/BD $|dom(v)| = 2 \mapsto$ Polynomial-time solvable

2009/GJ $|dom(v)| = 5 \mapsto$ NP-complete

⊛ Was it worth it? Why should we care? Where is practice?

- curiosity
- distilling "sources of complexity"
  (to know what precisely should be avoided)

Tractable Cases of Planning with Forks

[KD08]

Theorem (forks)

PlanMinGen for fork structured problems with root $r \in V$ is polynomial time solvable if

(i) $|dom(r)| = 2$, *or*

(ii) *for all $v \in V$, we have* $|dom(v)| = O(1)$,

Theorem (inverted forks)

PlanMinGen for inverted fork structured problems with root $r \in V$ is polynomial time solvable if $|dom(r)| = O(1)$.

Theorem (inverted forks)

Theorem (inverted forks)

PlanMinGen for inverted fork structured problems with root $r \in V$ is polynomial time solvable if $|dom(r)| = O(1)$.

Proof sketch (Construction)

(1) Create all $\Theta(d^d)$ cycle-free paths from $s^0[r]$ to $G[r]$ in $DTG(r, \Pi)$.

(2) For each $u \in \mathrm{pred}(r)$, and each $x, y \in dom(u)$, compute the cost-minimal path from x to y in $DTG(u, \Pi)$.

(3) For each path in $DTG(r, \Pi)$ generated in step (1), construct a plan for Π based on that path for r, and the shortest paths computed in (2).

(4) Take minimal cost plan from (3).

Putting things together

Major conclusion so far

Causal graphs are too coarse to provide an effective tractability-oriented abstraction

What about tasks with (some) domains of size $O(1)$?

- Chains $\rightsquigarrow$ NP-complete for $dom(v) > 4$. Open for 3 and 4.
- Forks $\rightsquigarrow$ P for $dom(r) = 2$, and for $dom(v) = O(1)$.
- Inverted Forks $\rightsquigarrow$ P for $dom(r) = O(1)$

Can we use these results in practice?

Let us step aside and recall abstraction heuristics.

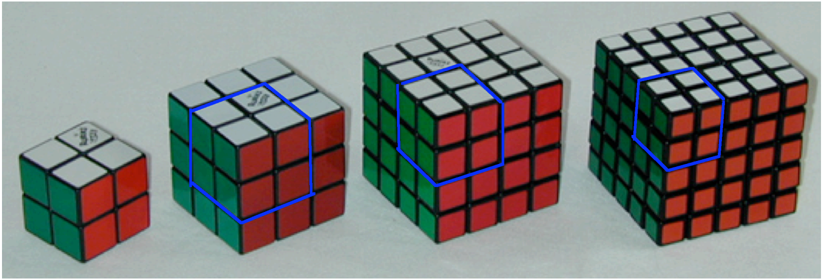
Limitations of Explicit Abstractions

Both PDBs and merge-and-shrink are explicit abstractions: abstract spaces are searched exhaustively

PDBs dimensionality $= O(1)$, size of the abstract space is $O(1)$

M&S dimensionality $= \Theta(|V|)$, size of the abstract space is $O(1)$

$\rightsquigarrow$ (often) price in heuristic accuracy in long-run



Abstractions: Extending the definition

Definition (abstraction, abstraction mapping)

Let $\mathcal{T} = \langle S, L, T, I, G, \mathcal{C} \rangle$ and $\mathcal{T}' = \langle S', L', T', I', G', \mathcal{C}' \rangle$ be transition systems with the same label set $L = L'$, $\mathcal{C} : S \rightarrow \mathbb{R}^{0+}$, $\mathcal{C}' : S' \rightarrow \mathbb{R}^{0+}$, and let $\alpha : S \rightarrow S'$.

We say that $\mathcal{T}'$ is an abstraction of $\mathcal{T}$ with abstraction mapping α if

- for all $s \in I$, we have $\alpha(s) \in I'$,
- for all $s \in G$, we have $\alpha(s) \in G'$, and
- for all $\langle s, l, t \rangle \in T$, we have $\langle \alpha(s), l, \alpha(t) \rangle \in T'$ $h^*(\alpha(s), \alpha(t)) \leq \mathcal{C}(l)$.

Structural Abstraction Heuristics: Main Idea

Objective (departing from PDBs)

Instead of perfectly reflecting a few state variables,
reflect many (up to $\Theta(|V|)$) state variables, BUT

- ♠ guarantee abstract space can be searched (implicitly) in poly-time

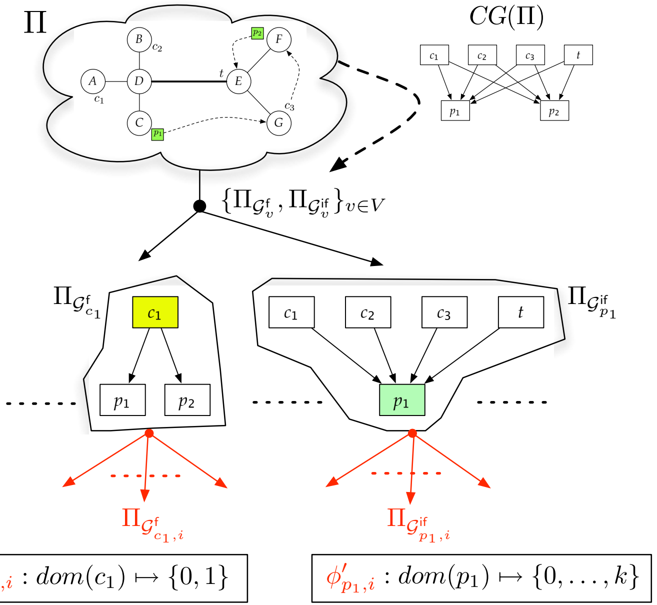
How

Abstracting Π by an instance of a tractable fragment of cost-optimal planning

- ☹ can our islands of tractability help us here?

Here Come the Forks!

Mixing Causal-Graph & Variable-Domain Decompositions



Planning / Logistics-00

Expanded nodesExpanded nodes and Time Shall we redefine the notion of success?...No.
Implicit abstraction databases!

#	h^*	HHH_{10^5}		$h^{\mathcal{F}}$			$h^{\mathcal{FI}}$ + opt	
		nodes	*time*	*nodes*	*time*	♠	*nodes*	*time*
01	20	21	*0.05*	21	*10.49*	*0.27*	21	*20.82*
02	19	20	*0.04*	20	*10.4*	*0.27*	20	*20.36*
03	15	16	*0.05*	16	*5.18*	*0.27*	16	*10.85*
04	27	28	*0.33*	28	*22.81*	*0.33*	28	*47.42*
05	17	18	*0.34*	18	*11.72*	*0.33*	18	*21.63*
06	8	9	*0.33*	9	*2.99*	*0.33*	9	*8.89*
07	25	26	*1.11*	26	*26.88*	*0.41*	26	*53.81*
08	14	15	*1.12*	15	*10.37*	*0.43*	15	*21.19*
09	25	26	*1.14*	26	*27.78*	*0.41*	26	*51.52*
10	36	37	*4.55*	37	*426.07*	*3.96*	37	*973.46*
11	44	2460	*4.65*	1689	*14259.8*	*4.25*	45	*1355.23*
12	31	32	*6.5*	32	*374.48*	*4.68*	32	*876.9*
13	44	7514	*6.84*	45	*702.29*	*4.63*	45	*1621.74*
14	36	37	*8.94*	37	*474.8*	*5.12*	37	*1153.85*
15	30	31	*8.84*	31	*448.86*	*5.12*	31	*1052.46*
16	45	29319	*17.35*	46	*3517.25*	*24.73*	46	*7635.96*
17	42	1561610	*45.61*	43	*3297.69*	*24.13*	43	*7192.51*
18	48	199428	*24.95*	697		*24.73*	49	*10014.3*
19	60			21959		*33.61*	61	*15625.5*
20	42	6095	*24.9*	43	*4325.45*	*29.61*	43	*9470.85*
21	68			106534		*61.54*	69	*22928.4*