Complexity Analysis in Planning

From Theory of Practice to Practice of Theory

Carmel Domshlak

Technion, Israel

What this talk is about?

1. What this talk is [not] about
2. Preliminaries
3. Search for/with tractability I: Syntax
4. Search for/with tractability II: Structure
5. Bridging between the islands I: Heuristic ensembles
6. Bridging between the islands II: Systems of systems
7. What next?

Why complexity?

- understand the problem
- know what is not possible
- find interesting subproblems
- distinguish essential features from syntactic sugar

Complexity in Planning, by Malte Helmert

Previous summer school, ICAPS-2009

MH focused on

- central complexity results
- expressivity vs. complexity tradeoff
- methodology for complexity analysis of planning formalisms

Complexity in Planning, by Malte Helmert

Previous summer school, ICAPS-2009

MH focused on

- central complexity results
- expressivity vs. complexity tradeoff
- methodology for complexity analysis of planning formalisms

CD will (try to focus) on something else

- Improving on MH is known to be 2-EXP-hard
- Great slides by MH with pointers to literature are online
- My objective today is a bit different

Goal of the Tutorial is Practice of Theory

Focus on computational tractability (CT)

- present major approaches to search for CT
- connect between CT and wider complexity analysis
- connect between CT and empirical progress

Goal of the Tutorial is Practice of Theory

Focus on computational tractability (CT)

- present major approaches to search for CT
- connect between CT and wider complexity analysis
- connect between CT and empirical progress

Disclaimer

- Not a comprehensive overview
  (or anything else, for that matter).
- Very subjective, and (hopefully) somewhat provocative.
- Stresses just one aspect of the story;
  many other aspects are also important.

What Do We Mean by "Computational Tractability"?

Given a problem Π, ability to solve in polynomial time something useful for solving Π.

1. Ability to solve something in polynomial time.
2. Given a problem Π, ability to solve in polynomial time something useful for solving Π.
3. For a formalism F (model + language), find tractable fragments of F

$\rightsquigarrow$ *Useful?*

What Do We Mean by "Computational Tractability"?

Given a problem Π, ability to solve in polynomial time something useful for solving Π.

1. Ability to solve something in polynomial time.
2. Given a problem Π, ability to solve in polynomial time something useful for solving Π.
3. For a formalism F (model + language), find tractable fragments of F

$\rightsquigarrow$ *Useful?*

What Do We Mean by "Computational Tractability"?

Given a problem Π, ability to solve in polynomial time something useful for solving Π.

1. Ability to solve something in polynomial time.
2. Given a problem Π, ability to solve in polynomial time something useful for solving Π.
3. For a formalism F (model + language), find tractable fragments of F

$\rightsquigarrow$ *Useful?*

What Do We Mean by "Computational Tractability"?

Given a problem Π, ability to solve in polynomial time something useful for solving Π.

1. Ability to solve something in polynomial time.
2. Given a problem Π, ability to solve in polynomial time something useful for solving Π.
3. For a formalism F (model + language), find tractable fragments of F

$\rightsquigarrow$ *Useful?*

Why Computational Tractability?

Bylander, 1994

If the relationship between intelligence and computation is taken seriously, then intelligence cannot be explained by intractable theories because no intelligent creature has the time to perform intractable computations. Nor can intractable theories provide any guarantees about the performance of engineering systems.

Why Computational Tractability?

Bylander, 1994

If the relationship between intelligence and computation is taken seriously, then intelligence cannot be explained by intractable theories because no intelligent creature has the time to perform intractable computations. Nor can intractable theories provide any guarantees about the performance of engineering systems.

- Point 1 is logical but vague (and thus misleading?)
  - What is the definition of "intractable theory"?
  - "Every science has a big lie. The big lie of complexity is worst case analysis." [C. Papadimitriou]
  - Still, worst case intractability severely limits us algorithmically
- Point 2 is a serious concern.

Some conclusions on Why Computational Tractability?

Concrete applications

- building systems with worst-case guarantees
- building new search guidance mechanisms
- combining a set of search guidance mechanisms
- checking whether new developments any needed (*)

What this talk is about?

1. What this talk is [not] about
2. Preliminaries
3. Search for/with tractability I: Syntax
4. Search for/with tractability II: Structure
5. Bridging between the islands I: Heuristic ensembles
6. Bridging between the islands II: Systems of systems
7. What next?

Model of Deterministic Planning
Transition systems

Definition (deterministic transition system)

A deterministic transition system is $\langle S, I, A, G \rangle$ where

- S is a finite set of states (the state space),
- $I \in S$ is initial state,
- actions $a \in A$ (with $a \subseteq S \times S$) are partial functions,
- $G \subseteq S$ is a finite set of goal states.

Definition (plan)

A plan for $\langle S, I, A, G \rangle$ is a sequence $\pi = \langle a_1, \ldots, a_n \rangle$ of actions from A such that $a_n(a_{n-1}(\ldots a_1(I) \ldots)) \in G$.

$\star$ (Shortest) path finding in digraph.

Finite Domain Representation (FDR) Language

Also known as $\textsc{sas}^+$

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

- An action a is applicable in a state $s \in dom(V)$ iff $s[v] = \text{pre}(a)[v]$ whenever pre$(a)[v]$ is specified
- Applying an applicable action a changes the value of each variable v to eff$(a)[v]$ if eff$(a)[v]$ is specified.
- *Induced deterministic transition system is straightforward.*

Boolean Domain Representation (BDR) Language

Also known as STRIPS with negative preconditions

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.

Major complexity classes

$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} = \mathsf{NPSPACE} \subseteq \mathsf{EXP} \subseteq \mathsf{NEXP} \subseteq \cdots$

- P, NP, and beyond NP: membership and hardness proofs
- Higher up $\rightsquigarrow$ rarer and smaller islands of tractability
- Higher up $\rightsquigarrow$ more sophistication needed to compete with humans?
- So which floor is FDR?

Major complexity classes

$P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXP \subseteq NEXP \subseteq \cdots$

- P, NP, and beyond NP: membership and hardness proofs
- Higher up $\rightsquigarrow$ rarer and smaller islands of tractability
- Higher up $\rightsquigarrow$ more sophistication needed to compete with humans?
- So which floor is FDR?

Computational Tasks

1. *PlanExt* — is the task solvable?
2. *PlanMin* — what is the cost of the optimal plan?
3. *PlanGen* — generate a plan for the task
4. *PlanMinGen* — generate an optimal plan for the task

Connections and relevance.

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!

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!

Heuristics Toolbox

Just 15 years ago

Nothing, but "STRIPS heuristic" (missing goals counting).

- HSP is considered natural yet hopeless approach to planning *(cf. R&N, ed1)*.
- Surprising, given successes of HS in AI back then ...

Heuristics Toolbox

Just 15 years ago

Nothing, but "STRIPS heuristic" (missing goals counting).

In (just) 15 years

HSP is considered a leading approach to planning *(cf. R&N, ed3)*.

1. Delete relaxation $\rightsquigarrow h_{\max}, h_{\text{add}}, h_{\text{FF}}, \ldots$
2. Critical paths/trees $\rightsquigarrow h^m, \ldots$
3. Landmarks $\rightsquigarrow h^{\mathsf{LAMA}}, h^{\mathsf{L}}, h^{\mathsf{LM\text{-}cut}}, \ldots$
4. Abstractions

   $\rightsquigarrow$ PDBs, m&s, fork decompositions ...

Heuristics Toolbox

Just 15 years ago

Nothing, but "STRIPS heuristic" (missing goals counting).

In (just) 15 years

HSP is considered a leading approach to planning *(cf. R&N, ed3)*.

1. Delete relaxation $\leadsto h_{\max}, h_{\text{add}}, h_{\text{FF}}, \ldots$
2. Critical paths/trees $\leadsto h^m, \ldots$
3. Landmarks $\leadsto h^{\mathsf{LAMA}}, h^{\mathsf{L}}, h^{\mathsf{LM\text{-}cut}}, \ldots$
4. Abstractions
   $\leadsto$ PDBs, m&s, fork decompositions ...

Related to our agenda today?

What this talk is about?

1. What this talk is [not] about
2. Preliminaries
3. Search for/with tractability I: Syntax
4. Search for/with tractability II: Structure
5. Bridging between the islands I: Heuristic ensembles
6. Bridging between the islands II: Systems of systems
7. What next?

Syntactic fragments

What are syntactic restrictions?

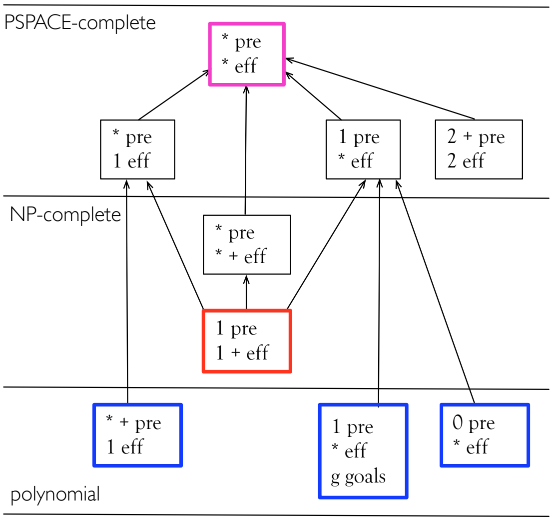
Fragment of tasks $\stackrel{\text{def}}{\longleftarrow}$ restrictions on action description (preconditions and effects)

1. Attack *a la* Erol, Nao, & Subrahmanian, and Bylander
   Restrictions on individual actions
2. Attack *a la* Bäckström, Klein, & Nebel
   Restrictions on action set as a whole

Note:

- Membership can be verified offline
- Membership can be verified in polynomial time (?)

Bylander's Map of BDR
PlanExt



NP-completeness of BDR$^1_{1+}$

Membership in NP by monotonicity of state updates.

Hardness by reduction from 3SAT. Let F be a 3CNF formula with n clauses over variables $U = \{u_1, \ldots, u_m\}$. An equivalent BDR$^1_{1+}$ task can be constructed as follows.

- State variables $V = \{c_1, \ldots, c_n,\ \ t_1, \ldots, t_m,\ \ f_1, \ldots, f_m\}$.
- Initial state $I = \emptyset$ (all vars set to *false*).
- Goal $G = \bigwedge_{i=1}^{n} c_i$.
- Actions
  1. For each u_i, two actions: $\neg f_i \Rightarrow t_i$ and $\neg t_i \Rightarrow f_i$
  2. For $1 \leq j \leq n$,
     - if j-th clause contains u_i, then action $t_i \Rightarrow c_j$
     - if j-th clause contains $\overline{u_i}$, then action $f_i \Rightarrow c_j$

⊛ Suggests why HSP for STRIPS planning was stuck

NP-completeness of BDR^1_{1+}

Membership in NP by monotonicity of state updates.

Hardness by reduction from 3SAT. Let F be a 3CNF formula with n clauses over variables $U = \{u_1, \ldots, u_m\}$. An equivalent BDR^1_{1+} task can be constructed as follows.

- State variables $V = \{c_1, \ldots, c_n, \;\; t_1, \ldots, t_m, \;\; f_1, \ldots, f_m\}$.
- Initial state $I = \emptyset$ (all vars set to *false*).
- Goal $G = \bigwedge_{i=1}^{n} c_i$.
- Actions
  1. For each u_i, two actions: $\neg f_i \Rightarrow t_i$ and $\neg t_i \Rightarrow f_i$
  2. For $1 \leq j \leq n$,
     - if j-th clause contains u_i, then action $t_i \Rightarrow c_j$
     - if j-th clause contains $\overline{u_i}$, then action $f_i \Rightarrow c_j$

⊛ Suggests why HSP for STRIPS planning was stuck

Islands of Tractability

BDR_1^+

- How? Dedicated algorithm, forward + backward search. Search for an intermediate state that can be reached with only positive-effect actions, and from which the goal can be reach with only negative-effect actions.
- Example: Blocksworld. ⊛ General practice?

Islands of Tractability

BDR_1^+

- How? Dedicated algorithm, forward + backward search.
- Example: Blocksworld. ⊛ General practice?

BDR^1 limited to $g = O(1)$ goals

- How? Exhaustive search through a "small" search space. A single goal cannot expand into multiple sub-goals.
- ⊛ Rings familiar? (Hint: critical-path heuristics h^m)

Islands of Tractability

BDR_1^+

- How? Dedicated algorithm, forward + backward search.
- Example: Blocksworld. ⊛ General practice?

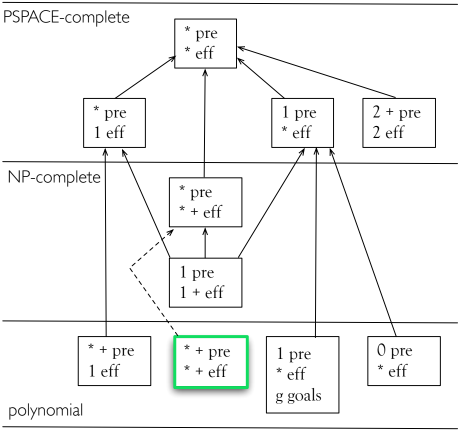
BDR^1 limited to $g = O(1)$ goals

- How? Exhaustive search through a "small" search space.
- ⊛ Rings familiar? (Hint: critical-path heuristics h^m)

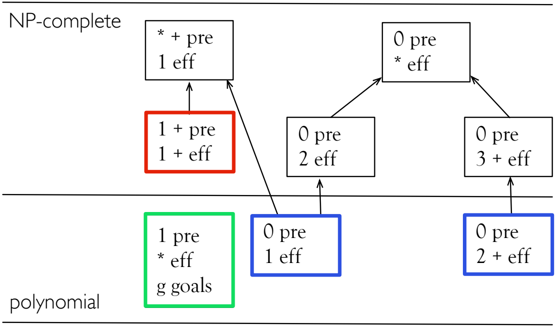
BDR^0

- How? Simple means-end analysis.
- ⊛ An advanced variant of "STRIPS heuristic" (missing goals counting).

BDR$^+_+$ is in P


PSPACE-complete
* pre
* eff
* pre
1 eff
1 pre
* eff
2 + pre
2 eff
NP-complete
* pre
* + eff
1 pre
1 + eff
* + pre
1 eff
* + pre
* + eff
1 pre
* eff
g goals
0 pre
* eff
polynomial

Bylander's Map of BDR
PlanMin



- The islands are getting smaller and rarer ...

NP-completeness of *PlanMin* for BDR^{1+}_{1+}

And for BDR^{0}_{3+}

Let F be a 3CNF formula with n clauses over variables $U = \{u_1, \ldots, u_m\}$. Construct a BDR^{1+}_{1+} task as follows.

- State variables $V = \{c_i\}_1^n \cup \{t_j, f_j, v_j\}_1^m$.
- Initial state $I = \emptyset$ (all vars set to *false*).
- Goal $G = \bigwedge_{i=1}^n c_i \wedge \bigwedge_{j=1}^m v_j$.
- Actions
    1. For each u_i, four actions: $\Rightarrow t_i$, $\Rightarrow f_i$, $f_i \Rightarrow v_i$ and $t_i \Rightarrow v_i$
    2. For $1 \leq j \leq n$,
        - if j-th clause contains u_i, then action $t_i \Rightarrow c_j$
        - if j-th clause contains $\overline{u_i}$, then action $f_i \Rightarrow c_j$

$\leadsto$ Task has a plan of length $2m + n$ iff F is satisfiable.

⊛ Hardness of BDR^{0}_{3+} by a simple reduction from Set Cover; both reductions prove hardness of h^+.

NP-completeness of *PlanMin* for BDR^{1+}_{1+}

And for BDR^{0}_{3+}

Let F be a 3CNF formula with n clauses over variables $U = \{u_1, \dots, u_m\}$. Construct a BDR^{1+}_{1+} task as follows.

- State variables $V = \{c_i\}_1^n \cup \{t_j, f_j, v_j\}_1^m$.
- Initial state $I = \emptyset$ (all vars set to *false*).
- Goal $G = \bigwedge_{i=1}^n c_i \wedge \bigwedge_{j=1}^m v_j$.
- Actions
  1. For each u_i, four actions: $\Rightarrow t_i$, $\Rightarrow f_i$, $f_i \Rightarrow v_i$ and $t_i \Rightarrow v_i$
  2. For $1 \leq j \leq n$,
     - if j-th clause contains u_i, then action $t_i \Rightarrow c_j$
     - if j-th clause contains $\overline{u_i}$, then action $f_i \Rightarrow c_j$

$\rightsquigarrow$Task has a plan of length $2m + n$ iff F is satisfiable.

$\circledast$ Hardness of BDR^{0}_{3+} by a simple reduction from Set Cover; both reductions prove hardness of h^+.

Revisiting the Heuristics Toolbox

Developments of the last 15 years

1. Delete relaxation $\rightsquigarrow h_{\max}, h_{\text{add}}, h_{\text{FF}}, \ldots$
2. Critical paths/trees $\rightsquigarrow h^m$
3. Landmarks $\rightsquigarrow h^{\mathsf{LAMA}}, h^{\mathsf{L}}, h^{\mathsf{LM\text{-}cut}}, \ldots$
4. Abstractions

   $\rightsquigarrow$ PDBs, m&s, fork decompositions ...

Syntactic fragments

What are syntactic restrictions?

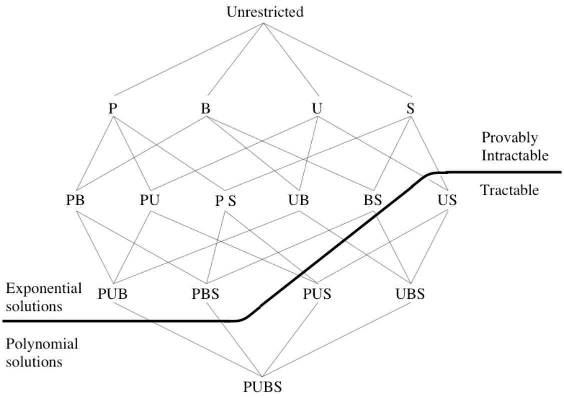
Fragment of tasks $\overset{\text{def}}{\longleftarrow}$ restrictions on action description (preconditions and effects)

1. Attack *a la* Erol, Nao, & Subrahmanian and Bylander
   Restrictions on individual actions.
   - Restrictions are natural and "easy to think in terms of"
   - Computational tractability is rare already for BDR
   - Some (2?) islands of tractability are extremely helpful in practice!
2. Attack *a la* Bäckström, Klein, & Nebel
   Restrictions on action set as a whole

The toolbox of four restrictions

1. Post-uniqueness: For each value of each state variable, there is at most one action achieving that value.
   - Strong condition: desired effects determine achievers.
2. Single-valuedness: If two actions are preconditioned by the value of some $v \in V$, and neither change its value, then they both are preconditioned by the *same value* of v.
   - Generalizes "positive preconditions".
     Example: If some action requires lights on, then no action requires lights off without turning them on.
3. Unariness (FDR_1)
4. Binariness (BDR)

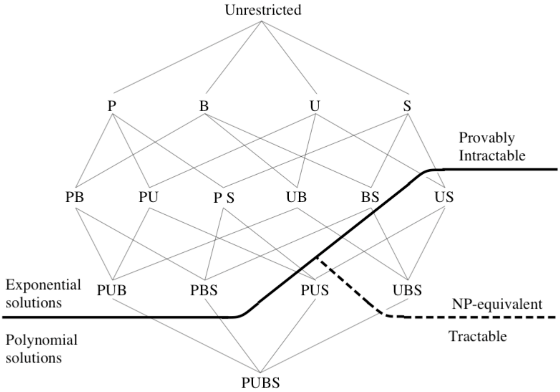
Bäckström & Nebel's Map of FDR
PlanGen



- US = certain generalization of BDR_1^+ to FDR

⊛ System design? Possible (in, e.g., automated control). Heuristics-oriented relaxations? At least not yet.

Bäckström & Nebel's Map of FDR
PlanMinGen



⊛ The already small island is getting smaller ...

Two quotes from the summary of Bäckström & Nebel (1995)

- *The most surprising result for us was that post-uniqueness of operators, which appears to be a very strong restriction, does not guarantee tractability if considered in isolation.*
  - Start with a (combinatorially) simple fragment. Then either climb to harder fragments, or you just saved yourself a lot of time.
- *This should not discourage us, however. It means that we have to start considering alternative restrictions, or combinations of less restricted variants of [our] restrictions.*
  - And this is one thing you can do with the time you just saved for yourself ☺

Two quotes from the summary of Bäckström & Nebel (1995)

- *The most surprising result for us was that post-uniqueness of operators, which appears to be a very strong restriction, does not guarantee tractability if considered in isolation.*
  - ⊛ Start with a (combinatorially) simple fragment. Then either climb to harder fragments, or you just saved yourself a lot of time.

- *This should not discourage us, however. It means that we have to start considering alternative restrictions, or combinations of less restricted variants of [our] restrictions.*
  - ⊛ And this is one thing you can do with the time you just saved for yourself ☺

What this talk is about?

1. What this talk is [not] about
2. Preliminaries
3. Search for/with tractability I: Syntax
4. Search for/with tractability II: Structure
5. Bridging between the islands I: Heuristic ensembles
6. Bridging between the islands II: Systems of systems
7. What next?

Structural fragments

Reminder: What are syntactic restrictions?

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

What are structural restrictions?

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

1. Attack *a la* Jonsson & Bäckström
   Restrictions on interaction between values of individual state variables
2. The causal graph journey
   Restrictions on interaction between variables

Note:

- Membership can be verified offline
- Membership can be verified in polynomial time (?)

Graphical Structures as Problem Abstractions

- General methodology:
  1. Project planning task on some of its aprehendable aspects
  2. Play with various constraints on these aspects
     - ⊛ Syntactic fragmentation was precisely about that
- 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 Structures as Problem Abstractions

- General methodology:
  1. Project planning task on some of its aprehendable aspects
  2. Play with various constraints on these aspects
     - ⊛ Syntactic fragmentation was precisely about that

- 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 Structures as Problem Abstractions

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

- 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

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 $CG(\Pi)$.

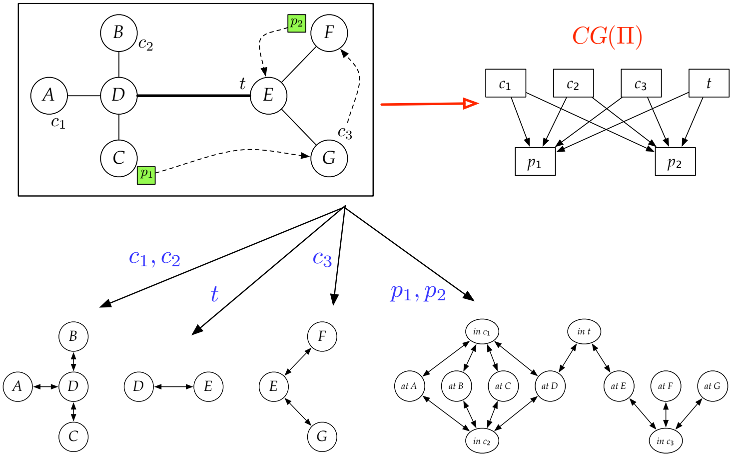
In the context of an FDR planning task $\Pi = \langle V, A, I, G \rangle$:

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 (ϑ, ϑ') labeled with $\in A$ is in the graph iff

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

Example



B
c_2
p_2
F
A
D
t
E
c_1
c_3
C
p_1
G
$CG(\Pi)$
c_1
c_2
c_3
t
p_1
p_2
c_1, c_2
t
c_3
p_1, p_2
B
A
D
C
D
E
F
E
G
in c_1
in t
at A
at B
at C
at D
at E
at F
at G
in c_2
in c_3

Computational Tractability as a Function of Causal Graph Form

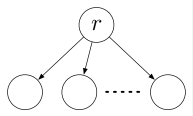
1. From BDR to FDR
2. From severe structural restrictions to their generalizations
3. For simplicity, assume all actions have the same cost (relevant only for optimization)

BDR Forks

- Informal discussion

PlanGen is easy

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

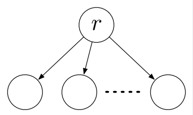


BDR Forks

⊛ 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

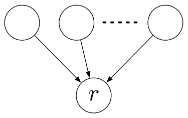
- Given root's workload, all leafs are independent.
- Optimize over all three cases of workload for root.

BDR Inverted Forks

⊛ Informal discussion

PlanGen is easy

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

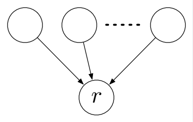


BDR Inverted Forks

⊛ Informal discussion

PlanGen is easy

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



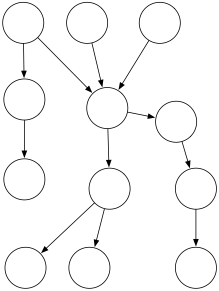
PlanMinGen is easy

- 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



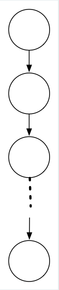
BDR Chains

⊛ Informal discussion

PlanGen is easy [BD03/BBDHP02]

loop

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



BDR Chains

⊛ Informal discussion

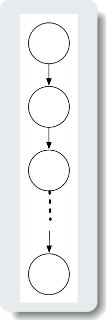
PlanGen is easy [BD03/BBDHP02]

loop

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

PlanMinGen is easy [KD08]

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

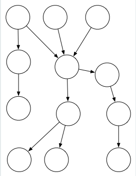


BDR Polytrees: Take I

Boooooooooooom!

PlanGen is easy
for graphs with fixed in-degree k [BD03]

- Lemma: If causal graph is DP singly connected, then no variable should change value more than $|V|$ times.
- BDR $\rightsquigarrow$ # value changes = sequence of value changes
- Algorithm:
  1. Top-down: Given parents' sequences of doable & possibly-needed value changes, determine var's sequence of such value changes.
  2. Toolbox: edge graphs
- Complexity: $O(|V|^{2k+3})$. (Aha ...)



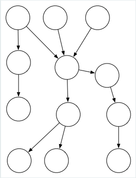
BDR Polytrees: Take I

PlanGen is easy
for graphs with fixed in-degree k [BD03]

- Complexity: $O(|V|^{2k+3})$. (Aha ...)

PlanMinGen is easy
(for graphs with fixed in-degree k) [KD08]

- Not the same algorithm!



PlanMinGen for BDR Polytrees

1. Compile Π into an equivalent constraint optimization problem COP_Π such that
   (I) COP_Π can be constructed in time polynomial in $||\Pi||$,
   (II) cost network of COP_Π = unoriented $CG(\Pi)$ (aka *tree*)
2. Solve COP_Π using linear-time algorithm for constraint optimization over trees.

- Methodology generalizes beyond trees (via tree decompositions of graphs).
- Step (I) can be challenging.

PlanMinGen for BDR Polytrees

1. Compile Π into an equivalent constraint optimization problem COP_{Π} such that
   (I) COP_{Π} can be constructed in time polynomial in $||\Pi||$,
   (II) cost network of COP_{Π} = unoriented $CG(\Pi)$ (aka *tree*)
2. Solve COP_{Π} using linear-time algorithm for constraint optimization over trees.

- Methodology generalizes beyond trees (via tree decompositions of graphs).
- Step (I) can be challenging.

BDR Polytrees: Take II

⊛ Everything so far was so good!
Can you get rid of the fixed in-degree assumption, please?

Looks like there is a promise ...

1. Compile Π into an equivalent constraint optimization problem COP_Π such that
   - (I) COP_Π can be constructed in time polynomial in $||\Pi||$,
   - (II) cost network of COP_Π = unoriented $CG(\Pi)$ (aka *tree*)
2. Solve COP_Π using linear-time algorithm for constraint optimization over trees.

BDR Polytrees: Take II

⊛ Everything so far was so good!
Can you get rid of the fixed in-degree assumption, please?

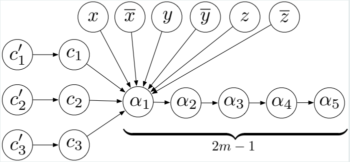
Looks like there is a promise ...

1. Compile Π into an equivalent constraint optimization problem COP_{Π} such that
   (I) COP_{Π} can be constructed in time polynomial in $||\Pi||$,
   (II) cost network of COP_{Π} = unoriented $CG(\Pi)$ (aka *tree*)
2. Solve COP_{Π} using linear-time algorithm for constraint optimization over trees.

BDR Polytrees: Take II

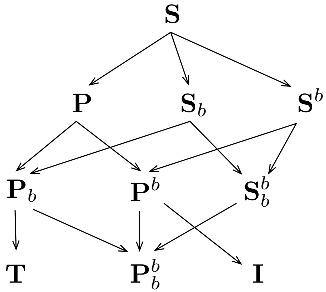
NO. . . *PlanGen* is NP-complete [GJ08]

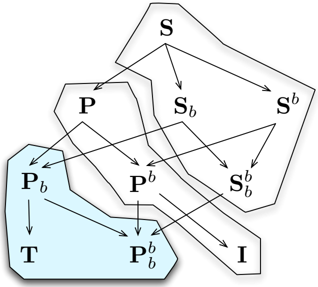
Elegant reduction from 3SAT (m clauses, n vars)



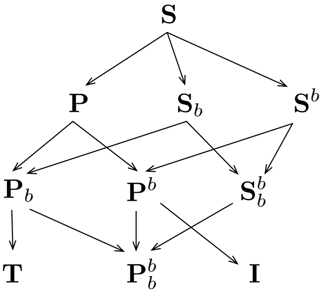
- Note that the proof kills directed inverted trees as well ...
- Can we push further with fixed in-degree?
  ⊛ Various alternative generalizations of polytrees.
- [BD03] For DP singly connected causal graphs, NP-complete starting (at most) in-degree 6.

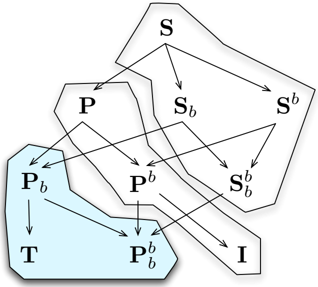
Wrapping-up the Tango of BDR and Causal Graphs





From BDR to FDR





And that is with binary variables only.
What about general finite domains?

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

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 $\mathsf{W}[1] \subseteq \mathsf{nu\text{-}FPT}$)*

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

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?*

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 directions from here

- Look for a different abstraction (later)
- Look for additional constraints on top of the causal graph (now)

Causal Graph and Reversibility

Definition (reversibility)

Π is *reversible* if for any state s reachable from the initial state, the initial state can be reached from s.

- Feature present in many benchmark domains!
- Membership test (?)

Causal Graph and Reversibility

Theorem (Chen & Gimenez classification [CG08])

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

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

Causal Graph and Reversibility

Theorem (Chen & Gimenez classification [CG08])

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

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

- The algorithm for the tractable case is easy (right?)
- Why and what is this "under succinct plan representation"?

Causal Graph and Reversibility

Theorem (Chen & Gimenez classification [CG08])

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

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

- Already exploited in embedded planning! [WN97]
- Inspired the original "causal graph heuristic" of Fast Downward. [H05]
- Close connection to HTN planning.

The Journey Continues!

Major conclusion so far

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

The Journey Continues!

Major conclusion so far

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

Possible directions from here

- Look for a different abstraction (later)
- Look for additional constraints on top of the causal graph
  - Complex state-space properties (e.g., reversibility)
  - Simple state-space properties? (What is simple?)

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?

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
- 2007/GJ $|dom(v)| = \Theta(|V|) \mapsto$ NP-complete
- 2008/GJ $|dom(v)| = 7 \mapsto$ NP-complete
- 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

2007/GJ $|dom(v)| = \Theta(|V|) \mapsto$ NP-complete

2008/GJ $|dom(v)| = 7 \mapsto$ NP-complete

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
- 2007/GJ $|dom(v)| = \Theta(|V|) \mapsto$ NP-complete
- 2008/GJ $|dom(v)| = 7 \mapsto$ NP-complete
- 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
- 2007/GJ $|dom(v)| = \Theta(|V|) \mapsto$ NP-complete
- 2008/GJ $|dom(v)| = 7 \mapsto$ NP-complete
- 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
- 2007/GJ $|dom(v)| = \Theta(|V|) \mapsto$ NP-complete
- 2008/GJ $|dom(v)| = 7 \mapsto$ NP-complete
- 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
- 2007/GJ $|dom(v)| = \Theta(|V|) \mapsto$ NP-complete
- 2008/GJ $|dom(v)| = 7 \mapsto$ NP-complete
- 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
- 2007/GJ $|dom(v)| = \Theta(|V|) \mapsto$ NP-complete
- 2008/GJ $|dom(v)| = 7 \mapsto$ NP-complete
- 2009/GJ $|dom(v)| = 5 \mapsto$ NP-complete

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

- curiosity (and with that, de facto judgements are problematic)

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
- 2007/GJ $|dom(v)| = \Theta(|V|) \mapsto$ NP-complete
- 2008/GJ $|dom(v)| = 7 \mapsto$ NP-complete
- 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)
- something else (TBP)

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 $u \in \text{pred}(r)$, 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.

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

Can we use these results in practice?

Let us step aside and recall abstraction heuristics.

Abstracting a transition system

Abstracting a transition system means dropping some distinctions between states, while preserving the transition behaviour as much as possible.

- An abstraction of a transition system $\mathcal{T}$ is defined by an abstraction mapping α that defines which states of $\mathcal{T}$ should be distinguished and which ones should not.
- From $\mathcal{T}$ and α, we compute an abstract transition system $\mathcal{T}'$ which is similar to $\mathcal{T}$, but smaller.
- The abstract goal distances (goal distances in $\mathcal{T}'$) are used as heuristic estimates for goal distances in $\mathcal{T}$.

Computing the abstract transition system

Given $\mathcal{T}$ and α, how do we compute $\mathcal{T}'$?

Requirement

We want to obtain an admissible heuristic.
Hence, $h^*(\alpha(s))$ (in the abstract state space $\mathcal{T}'$) should never overestimate $h^*(s)$ (in the concrete state space $\mathcal{T}$).

An easy way to achieve this is to ensure that all solutions in $\mathcal{T}$ also exist in $\mathcal{T}'$:

- If s is a goal state in $\mathcal{T}$, then $\alpha(s)$ is a goal state in $\mathcal{T}'$.
- If $\mathcal{T}$ has a transition from s to t, then $\mathcal{T}'$ has a transition from $\alpha(s)$ to $\alpha(t)$.

Practical requirements for abstractions

To be useful in practice, an abstraction heuristic must be efficiently computable. This gives us two requirements for α:

1. For a given state s, the abstract state $\alpha(s)$ must be efficiently computable.
2. For a given abstract state $\alpha(s)$, the abstract goal distance $h^*(\alpha(s))$ must be efficiently computable.

Canonical approach: explicit abstractions

- pattern database heuristics
- merge-and-shrink abstractions

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/potentially) price in heuristic accuracy in long-run

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

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

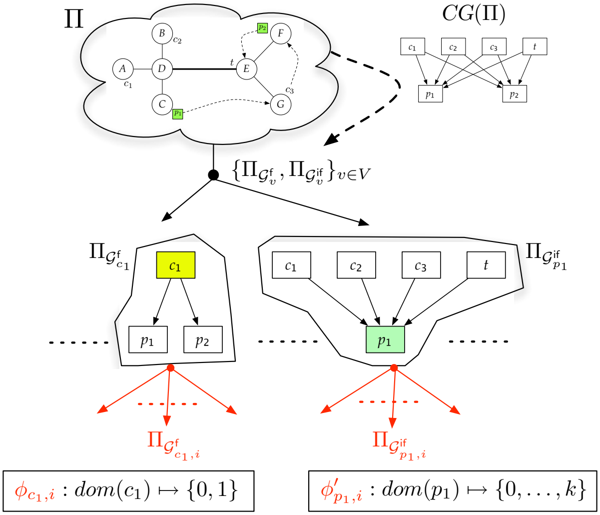
How

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



+ ensuring proper action cost partitioning

Planning / Logistics-00
Expanded nodes

#	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*	21	*20.82*
02	19	20	*0.04*	20	*10.4*	20	*20.36*
03	15	16	*0.05*	16	*5.18*	16	*10.85*
04	27	28	*0.33*	28	*22.81*	28	*47.42*
05	17	18	*0.34*	18	*11.72*	18	*21.63*
06	8	9	*0.33*	9	*2.99*	9	*8.89*
07	25	26	*1.11*	26	*26.88*	26	*53.81*
08	14	15	*1.12*	15	*10.37*	15	*21.19*
09	25	26	*1.14*	26	*27.78*	26	*51.52*
10	36	37	*4.55*	37	*426.07*	37	*973.46*
11	44	2460	*4.65*	1689	*14259.8*	45	*1355.23*
12	31	32	*6.5*	32	*374.48*	32	*876.9*
13	44	7514	*6.84*	45	*702.29*	45	*1621.74*
14	36	37	*8.94*	37	*474.8*	37	*1153.85*
15	30	31	*8.84*	31	*448.86*	31	*1052.46*
16	45	29319	*17.35*	46	*3517.25*	46	*7635.96*
17	42	1561610	*45.61*	43	*3297.69*	43	*7192.51*
18	48	199428	*24.95*			49	*10014.3*
19	60					61	*15625.5*
20	42	6095	*24.9*	43	*4325.45*	43	*9470.85*
21	68					69	*22928.4*

Planning / Logistics-00
Expanded nodes and Time

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

Planning / Logistics-00

Shall we redefine the notion of success?...

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

Planning / Logistics-00

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

Looking around

Tractable fragments are ...

1. rare, but still exist
2. key to heuristic engineering
3. based on very different sets of restrictions

Given a problem to solve, how shall we choose between

1. different heuristics/fragments?
2. different instances of a single heuristic/fragment?

It is generally not necessary to commit to a single heuristic/fragment.

Looking around

Tractable fragments are ...

1. rare, but still exist
2. key to heuristic engineering
3. based on very different sets of restrictions

Given a problem to solve, how shall we choose between

1. different heuristics/fragments?
2. different instances of a single heuristic/fragment?

It is generally not necessary to commit to a single heuristic/fragment.

What this talk is about?

1. What this talk is [not] about
2. Preliminaries
3. Search for/with tractability I: Syntax
4. Search for/with tractability II: Structure
5. Bridging between the islands I: Heuristic ensembles
6. Bridging between the islands II: Systems of systems
7. What next?

Combining multiple admissible heuristics

Maximizing several heuristics:

- By computing the maximum of several admissible heuristics, we obtain another admissible heuristic which dominates the component heuristics.

Adding several heuristics:

- In some cases, we can even compute the sum of individual estimates and still stay admissible.
- Summation often leads to much higher estimates than maximization, so it is important to understand when it is admissible.

Additive sets of heuristics

Theorem (action cost partitioning)

Let $\Pi, \Pi_1, \ldots, \Pi_k$ *be planning tasks, identical except for the operator costs* $cost, cost_1, \ldots, cost_k$*. Let* $\{h_i\}_{i=1}^k$ *be a set of* arbitrary *admissible heuristic functions for* $\{\Pi_i\}_{i=1}^k$*, respectively.*

If holds $cost(o) \geq \sum_{i=1}^k cost_i(o)$ *for all operators* o*, then* $\sum_{i=1}^k h_i$ *is an admissible heuristic for* Π*.*

Observations

- Generalizes action counting orthogonality
- No idea what partition is better? $\rightsquigarrow$ Uniform partition?

Additive sets of heuristics

Theorem (action cost partitioning)

Let $\Pi, \Pi_1, \ldots, \Pi_k$ be planning tasks, identical except for the operator costs $cost, cost_1, \ldots, cost_k$. Let $\{h_i\}_{i=1}^k$ be a set of arbitrary *admissible heuristic functions for $\{\Pi_i\}_{i=1}^k$, respectively.*

If holds $cost(o) \geq \sum_{i=1}^k cost_i(o)$ for all operators o, then $\sum_{i=1}^k h_i$ is an admissible heuristic for Π.

Observations

- Generalizes action counting orthogonality
- No idea what partition is better? $\rightsquigarrow$ Uniform partition?

Additive sets of heuristics

Theorem (action cost partitioning)

Let $\Pi, \Pi_1, \ldots, \Pi_k$ be planning tasks, identical except for the operator costs $cost, cost_1, \ldots, cost_k$. Let $\{h_i\}_{i=1}^k$ be a set of arbitrary *admissible heuristic functions for $\{\Pi_i\}_{i=1}^k$, respectively.*

If holds $cost(o) \geq \sum_{i=1}^k cost_i(o)$ for all operators o, then $\sum_{i=1}^k h_i$ is an admissible heuristic for Π.

Observations

- Generalizes action counting orthogonality
- No idea what partition is better? $\rightsquigarrow$ Uniform partition?
- Still, how to choose among the alternative cost partitions?

Optimal action cost partitioning for abstractions

Problem statement

Given

1. a (costs attached) transition system $\mathcal{T}$,
2. a set of (costs attached) abstractions $\{\mathcal{T}_i\}_{i=1}^k$ of $\mathcal{T}$ with abstraction mappings $\{\alpha_i\}_{i=1}^k$, respectively, and
3. a state s in $\mathcal{T}$,

determine optimal additive heuristic for $\mathcal{T}$ on the basis of $\{\mathcal{T}_i\}_{i=1}^k$, that is

$$h_{\mathsf{opt}}(s) = \max_{\{cost_i\}} \sum_{i=1}^{k} h_i^*(\alpha_i(s)).$$

Problems on the way

Optimal additive heuristic for $\mathcal{T}$ on the basis of $\{\mathcal{T}_i\}_{i=1}^k$

$$h_{\mathsf{opt}}(s) = \max_{\{cost_i\}} \sum_{i=1}^{k} h_i^*(\alpha_i(s)).$$

Challenges

1. Infinite space of alternative choices $\{cost_i\}_{i=1}^k$
2. The optimal choice is state-dependent
3. The process is fully unsupervised

The LP trick

Main Idea

Instead of, given an action cost partition $\{cost_i\}_{i=1}^k$, independently searching each abstraction $\mathcal{T}_i$ using dynamic programming

1. compile SSSP problem over each $\mathcal{T}_i$ into a linear program $\mathscr{L}_i$ with action costs being free variables
2. combine $\mathscr{L}_1, \ldots, \mathscr{L}_k$ with additivity constraints $cost(o) \geq \sum_{i=1}^k cost_i(a)$
3. solution of the joint LP $\rightsquigarrow h_{\text{opt}}(s)$

The LP trick

Main Idea

Instead of, given an action cost partition $\{cost_i\}_{i=1}^{k}$, independently searching each abstraction $\mathcal{T}_i$ using dynamic programming

1. compile SSSP problem over each $\mathcal{T}_i$ into a linear program $\mathscr{L}_i$ with action costs being free variables
2. combine $\mathscr{L}_1, \dots, \mathscr{L}_k$ with additivity constraints $cost(o) \geq \sum_{i=1}^{k} cost_i(a)$
3. solution of the joint LP $\rightsquigarrow h_{\mathsf{opt}}(s)$

Single-Source Shortest Paths: LP Formulation

LP formulation

Given: digraph $G = (N, E)$, source node $v \in N$

LP variables: $d(v') \leadsto$ shortest-path length from v to v'

LP:

$$\max_{\vec{d(\cdot)}} \sum_{v'} d(v')$$

$$\text{s.t. } d(v) = 0$$

$$d(v'') \leq d(v') + w(v', v''), \;\; \forall (v', v'') \in E$$

Step 1: Compile each SSSP over $\mathcal{T}_i$ into $\mathscr{L}_i$

LP formulation

Given: abstraction $\mathcal{T}_i$, state s of concrete system $\mathcal{T}$

LP variables: $\{d(s') \mid s' \in S_i\} \cup \{d(G_i)\} \cup \{cost(o, i)\}$

LP:

$$\max \quad d(G_i)$$

$$\text{s.t.} \quad \begin{cases} d(s') \leq d(s'') + cost(o, i), & \forall \langle s', o, s'' \rangle \in \mathcal{T}_i \\ d(s') = 0, & s' = \alpha_i(s) \\ d(G_i) \leq d(s'), & s' \in G(i) \end{cases}$$

Step 2: Properly combine $\{\mathscr{L}_i\}_{i=1}^k$

LP formulation

Given: abstractions $\{\mathcal{T}_i\}_{i=1}^k$ state s of $\mathcal{T}$

LP variables: $\bigcup_{i=1}^k \{d(s') \mid s' \in S_i\} \cup \{d(G_i)\} \cup \{cost(o,i)\}$

LP:

$$\max \sum_{i=1}^{k} d(G_i)$$

$$\text{s.t. } \forall i \begin{cases} d(s') \leq d(s'') + cost(o,i), & \forall \langle s', o, s'' \rangle \in \mathcal{T}_i \\ d(s') = 0, & s' = \alpha_i(s) \\ d(G_i) \leq d(s'), & s' \in G(i) \end{cases}$$

$$\forall o \in O : \quad cost(o) \geq \sum_{i=1}^{k} cost(o,i)$$

Optimizing Action-Cost Partitioning: Generalization

General theory of LP-optimizable ensembles
of additive heuristic functions

- Warning: Any reduction to LP is not enough
  $\rightsquigarrow$ requires (surprising) relation between polyhedron and planning problem

Optimizing Action-Cost Partitioning: Generalization

General theory of LP-optimizable ensembles
of additive heuristic functions

- Warning: Any reduction to LP is not enough
- Works as above for
  - projection and variable-domain abstraction (PDB) heuristics
  - constrained PDBs heuristics (Haslum *et al.*, 2005)
  - merge-and-shrink abstractions (Helmert *et al.*, 2007)
- Suitable poly-size LPs $\mathscr{L}_i$ exist also for
  - fork-decomposition heuristics
  - tree-COP reducible fragments of tractable cost-optimal planning (from Katz & D, 2007)
  - ...

Optimizing Action-Cost Partitioning: Generalization

General theory of LP-optimizable ensembles
of additive heuristic functions

- Warning: Any reduction to LP is not enough
- Works as above for
  - projection and variable-domain abstraction (PDB) heuristics
  - constrained PDBs heuristics (Haslum *et al.*, 2005)
  - merge-and-shrink abstractions (Helmert *et al.*, 2007)
- Suitable poly-size LPs $\mathscr{L}_i$ exist also for
  - fork-decomposition heuristics
  - tree-COP reducible fragments of tractable cost-optimal planning (from Katz & D, 2007)
  - ...

LP for Inverted Forks (1)

Given: problem Π, state s, goal G

Variables

$$\overrightarrow{x} = \{h^*\} \cup \bigcup_{\substack{v \in V' \setminus \{r\}, \\ \vartheta, \vartheta' \in dom(v)}} \{d(v, \vartheta, \vartheta')\}.$$

$d(v, \vartheta, \vartheta') \rightsquigarrow$ cost of the cheapest sequence of actions *affecting* v that changes its value from ϑ to ϑ'

Objective

$$\max \; \{h^*\}$$

LP for Inverted Forks (2)

Given: problem Π, state s, goal G

Constraints (I)

For each simple path $\langle a_1 \cdot \ldots \cdot a_m \rangle$ from $s[r]$ to $G[r]$ in *DTG*(r, Π),

$$h^* \leq \sum_{v \in V \setminus \{r\}} d(v, s_0[v], s_1[v]) + \sum_{i=1}^{m} \left(\mathcal{C}(a_i) + \sum_{v \in V' \setminus \{r\}} d(v, s_i[v], s_{i+1}[v]) \right)$$

where

$$s_i[v] = \begin{cases} s[v], & i = 0 \\ G[v], & i = m+1, \text{ and } G[v] \text{ is specified} \\ \mathsf{pre}(a_i)[v], & 1 \leq i \leq m, \text{ and } \mathsf{pre}(a_i)[v] \text{ is specified} \\ s_{i-1}[v], & \text{otherwise} \end{cases}$$

Semantics: *The cost of solving the problem is not greater than the cost of any cycle-free path of r plus sums of costs of reaching the prevail conditions of actions on this path and reaching the goal afterwards.*

LP for Inverted Forks (3)

Given: problem Π, state s, goal G

Constraints (II)

For each $v \in V \setminus \{r\}$, $\vartheta \in dom(v)$,

$$d(v, \vartheta, \vartheta) = 0$$

For each v-changing action $a \in A$,

$$d(v, \vartheta, \mathsf{post}(a)[v]) \leq d(v, \vartheta, \mathsf{pre}(a)[v]) + \mathcal{C}(a)$$

Semantics: *Shortest-path constraints.*

What this talk is about?

1. What this talk is [not] about
2. Preliminaries
3. Search for/with tractability I: Syntax
4. Search for/with tractability II: Structure
5. Bridging between the islands I: Heuristic ensembles
6. Bridging between the islands II: Systems of systems
7. What next?

Planning for Automated Control

- We have discussed composing islands of tractability within heuristics
- Next: composing islands of tractability in industrial systems

Motivation

Observations

- Automated planning is generally hard
- Bing (27/5/2011)

"automated planning"	11M
"ai planning"	18M
"strips planning"	73M
"classical planning"	85M
"multi[-]agent planning"	

Motivation

Observations

- Automated planning is generally hard
- Bing (27/5/2011)

"automated planning"	11M
"ai planning"	18M
"strips planning"	73M
"classical planning"	85M
"multi[-]agent planning"	110M

- Paradox?
  - Yes (you cannot lose weight by eating more)
  - Not necessarily, if these works assume some sort of simple agents (plus something else)
  - Formal analysis?

Motivation

Logistics planning

Deliver packages using vehicles (trucks, airplanes, ships) operating in/between different countries/regions/cities

- Classical benchmark for "single-agent" planning
- Classic example of a distributed system $\rightsquigarrow$ vehicle = agent

(Informal) Question

Can we exploit the fact that the domain describes a naturally distributed system to make planning more efficient?

(Ultimate) Answer

YES, we can solve distributed components independently

Motivation

Logistics planning

Deliver packages using vehicles (trucks, airplanes, ships) operating in/between different countries/regions/cities

- Classical benchmark for "single-agent" planning
- Classic example of a distributed system $\rightsquigarrow$ vehicle = agent

(Informal) Question

Can we exploit the fact that the domain describes a naturally distributed system to make planning more efficient?

(Ultimate) Answer

YES, we can solve distributed components independently

Basic Motivation/Intuition

k-agents MA Systems (Logistics domain example)

Fully decoupled

Vehicles are a priori responsible for different packages

Same as planning k times for a single agent
$\leadsto$ linear time-complexity growth
($\exp(k)$ time-complexity reduction)

Fully coupled

Vehicles have to move the same packages and maybe coordinate on loads/unloads

Same as planning for a single "k-times larger" agent
$\leadsto$ $\exp(k)$ time-complexity growth
(no reduction in time-complexity)

Basic Motivation/Intuition
k-agents MA Systems (Logistics domain example)

Fully decoupled

Same as planning k times for a single agent
$\rightsquigarrow$ linear time-complexity growth
($\exp(k)$ time-complexity reduction)

Fully coupled

Same as planning for a single "k-times larger" agent
$\rightsquigarrow$ $\exp(k)$ time-complexity growth
(no reduction in time-complexity)

Loosely coupled

Somewhere in between depending on the "level" of coupling?

"Loose Coupling" is a Loose Concept

Questions

1. How to measure the coupling level of a MA system?

2. Is there an algorithm that
   1. can handle any "coupling level", yet
   2. is guaranteed to benefit from lower "coupling level"

How to measure coupling of a MA system?
Multiagent = Distributed = Modular = ...

⊛ Let us use this illustration to establish intuitions. Ideas?

Next

- Formal measure of coupling level by a combination of
  1. a measure of a MA system's inherent coupling level
  2. a measure of a problem's coupling level
- An algorithm that scales
  - exponentially with coupling level
  - polynomially with the number of agents
- Based on a very simple model
  $\rightsquigarrow$ a minimal extension of FDR to MA systems

Agent Actions

Logistics planning

Deliver packages using vehicles (trucks, airplanes, ships) operating in/between different countries/regions/cities

- Actions $\texttt{move}(v, from, to), \texttt{load}(p, v, at), \texttt{unload}(p, v, at)$
- Agents: vehicles
- Vehicle agent actions:
  moving it, loading into it, unloading from it

From FDR to MA-FDR

Everything is the same, except that
actions are partitioned between the agents

Our Focus Here

Input Planning problem for a set of k collaborative agents

Question To what extent is planning for such a MA system harder than solving individual planning problems of each of the agents in isolation?

Approach Theoretical. Try to formulate an algorithm that is tractable under reasonable conditions.

Centralized Planning for MA Systems

Our Focus Here

Input Planning problem for a set of k collaborative agents

Question To what extent is planning for such a MA system harder than solving individual planning problems of each of the agents in isolation?

Approach Theoretical. Try to formulate an algorithm that is tractable under reasonable conditions.

Solving MA-FDR Problems

Standard Approaches

1. Compile into a single-agent FDR problem
   - ☹ Lose all structure and obtain k-times larger "agent"
   - ☹ Worst-case complexity exponential in description size or shortest plan (depending on search strategy)
2. Try to solve as much as possible locally and compose the resulting individual agent plans
   - ☹ What can we say about it?

Solving MA-FDR Problems

Standard Approaches

1. Compile into a single-agent FDR problem
   - ☹ Lose all structure and obtain k-times larger "agent"
   - ☹ Worst-case complexity exponential in description size or shortest plan (depending on search strategy)

2. Try to solve as much as possible locally and compose the resulting individual agent plans
   - ☹ What can we say about it?

Main Ideas

A New Graphical Model

Potential (positive and negative) interactions between the agents' individual abilities (= actions)

System coupling-level

Define an interaction graph of the system

Nodes = agents

Edges = agents may need (coordinate with) each other

Parameter $\omega \rightsquigarrow$ tree-width of interaction graph

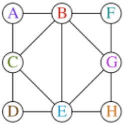
Main Ideas

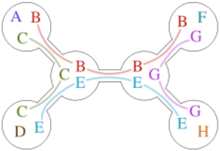
A New Graphical Model

Potential (positive and negative) interactions between the agents' individual abilities (= actions)

System coupling-level

Parameter $\omega \rightsquigarrow$ tree-width of interaction graph





Main Ideas

System coupling-level

Parameter $\omega \rightsquigarrow$ tree-width of interaction graph

Problem coupling-level

Some problems require more coordination than others!

Parameter $\delta \rightsquigarrow$ minmax number of times a single agent needs some other agent to do something for it

Main Ideas

System coupling-level

Parameter $\omega \rightsquigarrow$ tree-width of interaction graph

Problem coupling-level

Parameter $\delta \rightsquigarrow$ minmax number of times a single agent needs some other agent to do something for it

Algorithm

- Fix the agents' commitments to each other
  $\rightsquigarrow$ *careful selection of language matters!*
- Let each agent independently plan "in-between" commitments
- Use iterative deepening to extend the number of per-agent commitments if needed

A Closer Look at Agent Actions

Private vs. Non-Private

Private: affect and depend only on that agent

Non-Private: all the rest

Logistic planning

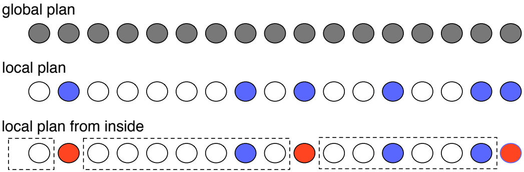
- Move actions are private
  (influence and influenced only by vehicle location)
- Loading into/unloading from a vehicle is non-private
  $\leadsto$ unless the package location is private to the vehicle!

A Closer Look at Agent Subplans

Private vs. Non-Private

Private affect and depend only on that agent

Non-Private all the rest



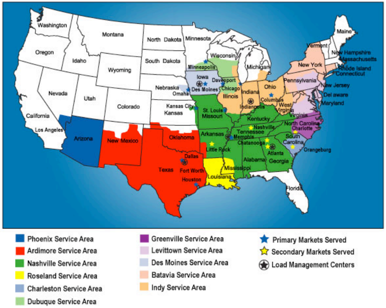
- non-private actions in the plan $\rightsquigarrow$ coordination points
- arbitrarily long sequences of private actions between adjacent non-private actions

Example: Logistics

Logistics

- imagine vehicles moving on a large map
- each vehicle has a service region

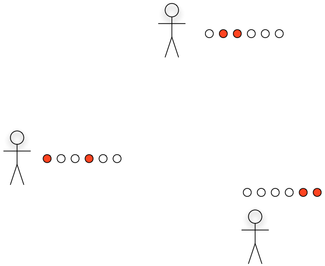
⇝ between each load/unload action, there are multiple `move` actions by the vehicle



Main Idea

"Algorithm"

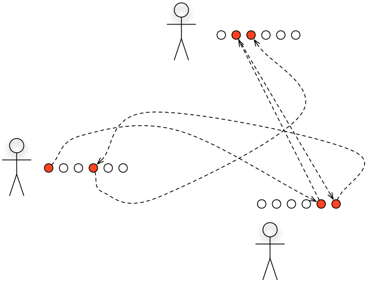
1. Find a good choice of coordination points
2. Solve k local planning problems over the private actions of the agents only



Main Idea

"Algorithm"

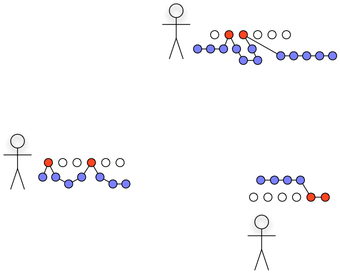
1. Find a good choice of coordination points
2. Solve k local planning problems over the private actions of the agents only



Main Idea

"Algorithm"

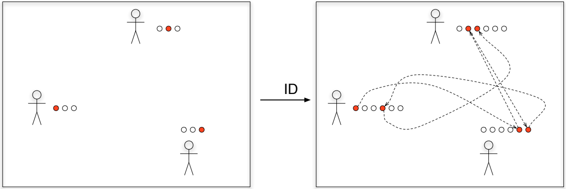
1. Find a good choice of coordination points
2. Solve k local planning problems over the private actions of the agents only



Main Idea

"Algorithm"

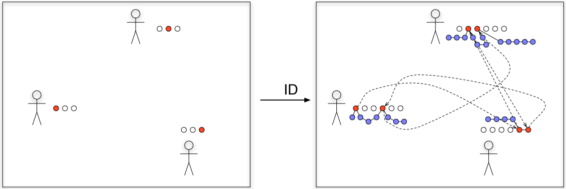
1. Find a good choice of coordination points
   - Iterative deepening on δ — # of coord-points per agent
   - For each choice of δ
     - Define a CSP whose solutions are consistent assignments to the coordination points
2. Solve k local planning problems over the private actions



Main Idea

"Algorithm"

1. Find a good choice of coordination points
2. Solve k local planning problems over the private actions
   - purely independent phase $\rightsquigarrow$ unary constraints
   - can be reduced to regular FDR planning



Complexity

The complexity is derived from

1. number of agents (k)
2. complexity of local planning (M)
3. number of "coordination" CSPs we have to solve ($\leadsto \delta$)
4. solving each "coordination" CSP (?)

Complexity

The complexity is derived from

1. number of agents (k)
2. complexity of local planning (M)
3. number of "coordination" CSPs we have to solve ($\rightsquigarrow \delta$)
4. solving each "coordination" CSP

$$O\left(k \cdot \left(\exp\left(\omega\delta + \omega + \delta\right) + M \cdot \exp\left(\delta\right)\right)\right)$$

$M =$ complexity of planning for a focused module

$=?$ tractable

Complexity

The complexity is derived from

1. number of agents (k)
2. complexity of local planning (M)
3. number of "coordination" CSPs we have to solve ($\leadsto \delta$)
4. solving each "coordination" CSP

$$O\left(k \cdot \left(\exp\left(\omega\delta + \omega + \delta\right) + M \cdot \exp\left(\delta\right)\right)\right)$$

$M =$ complexity of planning for a focused module

$=?$ tractable

Intermediate Summary

- Formal measure of coupling level by a combination of
  - δ problem-specific #times an agent needs assistance
  - ω the inherent coupling level of the system
- Planning complexity polynomial in the number of agents (for fixed coupling level)
- "Coordination complexity" is not affected by the length of the local plans
- Generating fully distributed algorithm conceptually easy
  - Use distributed CSP
  - Local planning is already distributed

Practice and Extensions

1. Can we really exploit these theoretical guarantees in practice?
2. Can we say something intelligent for self-interested agents?
3. Can we improve the theoretical upper bound?

Practice and Extensions

1. Can we really exploit these theoretical guarantees in practice?
   - Nissim, Brafman, & Domshlak. *A General, Fully Distributed Multi-Agent Planning Algorithm.* AAMAS-2010.
2. Can we say something intelligent for self-interested agents?
3. Can we improve the theoretical upper bound?

Practice and Extensions

1. Can we really exploit these theoretical guarantees in practice?
   - Nissim, Brafman, & Domshlak. *A General, Fully Distributed Multi-Agent Planning Algorithm.* AAMAS-2010.
2. Can we say something intelligent for self-interested agents?
   - Brafman, Domshlak, Engel, & Tennenholtz. *Planning Games.* IJCAI-2009.
   - Brafman, Domshlak, Engel, & Tennenholtz. *Transferable Utility Planning Games.* AAAI-2010.
3. Can we improve the theoretical upper bound?

Practice and Extensions

1. Can we really exploit these theoretical guarantees in practice?
   - Nissim, Brafman, & Domshlak. *A General, Fully Distributed Multi-Agent Planning Algorithm.* AAMAS-2010.
2. Can we say something intelligent for self-interested agents?
   - Brafman, Domshlak, Engel, & Tennenholtz. *Planning Games.* IJCAI-2009.
   - Brafman, Domshlak, Engel, & Tennenholtz. *Transferable Utility Planning Games.* AAAI-2010.
3. Can we improve the theoretical upper bound?
   - Remains open question.

What this talk is about?

1. What this talk is [not] about
2. Preliminaries
3. Search for/with tractability I: Syntax
4. Search for/with tractability II: Structure
5. Bridging between the islands I: Heuristic ensembles
6. Bridging between the islands II: Systems of systems
7. What next?

What next?
This is just a short list of obvious things

1. Some fascinating problems are still open
   - what happens with chain causal graphs and ternary domains?!
2. Novel combinations of syntax and structure and ???.
   - action k-dependence [KD08,GJ09]
3. Novel graphical/??? structures.
   - interaction networks [CG10]
   - refinements of causal graph [BD08]
4. CT in more complex formalisms?
   - M. Helmert. *Decidability and undecidability results for planning with numerical state variables.* AIPS-2002.
5. Exploitation of CT in modular/hierarchical/??? systems.
6. New algorithmic ideas for domain-independent heuristics!

What next?

This is just a short list of obvious things

1. Some fascinating problems are still open
   - what happens with chain causal graphs and ternary domains?!
2. Novel combinations of syntax and structure and ???.
   - action k-dependence [KD08,GJ09]
3. Novel graphical/??? structures.
   - interaction networks [CG10]
   - refinements of causal graph [BD08]
4. CT in more complex formalisms?
   - M. Helmert. *Decidability and undecidability results for planning with numerical state variables.* AIPS-2002.
5. Exploitation of CT in modular/hierarchical/??? systems.
6. New algorithmic ideas for domain-independent heuristics!

What next?

This is just a short list of obvious things

1. Some fascinating problems are still open
   - what happens with chain causal graphs and ternary domains?!
2. Novel combinations of syntax and structure and ???.
   - action k-dependence [KD08,GJ09]
3. Novel graphical/??? structures.
   - interaction networks [CG10]
   - refinements of causal graph [BD08]
4. CT in more complex formalisms?
   - M. Helmert. *Decidability and undecidability results for planning with numerical state variables.* AIPS-2002.
5. Exploitation of CT in modular/hierarchical/??? systems.
6. New algorithmic ideas for domain-independent heuristics!

What next?

This is just a short list of obvious things

1. Some fascinating problems are still open
   - what happens with chain causal graphs and ternary domains?!
2. Novel combinations of syntax and structure and ???.
   - action k-dependence [KD08,GJ09]
3. Novel graphical/??? structures.
   - interaction networks [CG10]
   - refinements of causal graph [BD08]
4. CT in more complex formalisms?
   - M. Helmert. *Decidability and undecidability results for planning with numerical state variables.* AIPS-2002.
5. Exploitation of CT in modular/hierarchical/??? systems.
6. New algorithmic ideas for domain-independent heuristics!

What next?

This is just a short list of obvious things

1. Some fascinating problems are still open
   - what happens with chain causal graphs and ternary domains?!
2. Novel combinations of syntax and structure and ???.
   - action k-dependence [KD08,GJ09]
3. Novel graphical/??? structures.
   - interaction networks [CG10]
   - refinements of causal graph [BD08]
4. CT in more complex formalisms?
   - M. Helmert. *Decidability and undecidability results for planning with numerical state variables.* AIPS-2002.
5. Exploitation of CT in modular/hierarchical/??? systems.
6. New algorithmic ideas for domain-independent heuristics!

What next?

This is just a short list of obvious things

1. Some fascinating problems are still open
   - what happens with chain causal graphs and ternary domains?!
2. Novel combinations of syntax and structure and ???.
   - action k-dependence [KD08,GJ09]
3. Novel graphical/??? structures.
   - interaction networks [CG10]
   - refinements of causal graph [BD08]
4. CT in more complex formalisms?
   - M. Helmert. *Decidability and undecidability results for planning with numerical state variables.* AIPS-2002.
5. Exploitation of CT in modular/hierarchical/??? systems.
6. New algorithmic ideas for domain-independent heuristics!

Syntactic properties and planning complexity

- T. Bylander. *The computational complexity of propositional STRIPS planning.* AIJ, 1994.
- K. Erol, D. S. Nau, & V. S. Subrahmanian. *Complexity, decidability and undecidability results for domain-independent planning.* AIJ, 1995.
- C. Bäckström, & B. Nebel. *Complexity results for SAS^+ planning.* Computational Intelligence, 1995.

Mixed syntactic/structural restrictions

- M. Katz, & C. Domshlak. *New Islands of Tractability of Cost-Optimal Planning.* JAIR, 2008.
- O. Giménez, & A. Jonsson. *The Influence of k-Dependence on the Complexity of Planning.* ICAPS-2009.

Structural properties and planning complexity

- P. Jonsson, & C. Bäckström. *State-variable planning under structural restrictions: Algorithms and complexity.* AIJ, 1998.
- P. Jonsson, & C. Bäckström. *Tractable plan existence does not imply tractable plan generation.* Annals of Math. and AI, 1998.
- C. Domshlak, & Y. Dinitz. *Multi-agent off-line coordination: Structure and complexity.* ECP-2001.
- R. I. Brafman, & C. Domshlak. *Structure and complexity in planning with unary operators.* JAIR, 2003.
- M. Katz, & C. Domshlak. *New Islands of Tractability of Cost-Optimal Planning.* JAIR, 2008.

Structural properties and planning complexity

- H. Chen, & Omer Giménez. *Causal graphs and structurally restricted planning.* J. of Comp. and System Sciences, 2010.
- O. Giménez, & A. Jonsson. *The complexity of planning problems with simple causal graphs.* JAIR, 2008.
- O. Giménez, & A. Jonsson. *Planning over chain causal graphs for variables with domains of size 5 is NP-hard.* JAIR, 2009.

Mixed syntactic/structural restrictions

- M. Katz, & C. Domshlak. *New Islands of Tractability of Cost-Optimal Planning.* JAIR, 2008.
- O. Giménez, & A. Jonsson. *The Influence of k-Dependence on the Complexity of Planning*. ICAPS-2009.

Heuristic Ensembles

- M. Katz, & C. Domshlak. *Optimal admissible composition of abstraction heuristics*. AIJ, 2010.
- M. Helmert, & C. Domshlak. *Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?*. ICAPS-2010.

Plan-space properties and planning complexity

- H. Chen, & O. Giménez. *Act local, think global: Width notions for tractable planning.* ICAPS-2007.
- R. I. Brafman, & C. Domshlak. *Factored planning: How, When, and When Not.* AAAI-2006.
- R. I. Brafman, & C. Domshlak. *From One to Many: Planning for Loosely Coupled Multi-Agent Systems*. ICAPS-2008.