Distributed Cost-Optimal Planning

Loïg Jezequel

ENS Cachan Bretagne

November 13, 2012
Planning problems: overview

Goal

Find a *plan*: a sequence of actions (with minimal cost) moving the system from its initial state to one of its goal states.
Planning problems: search in graphs

Initial state of the resources

Goal state of the resources

Action

New state of the resources
Planning problems: resolution

Heuristic search

- A*-like algorithms: Hart et al. 1968
- Various heuristics: Bonet and Geffner 2001, Helmert et al. 2007, Karpas and Domshlak 2009, ...

Parallelism of actions (concurrency)

- GRAPHPLAN: Blum and Furst 1995
- Petri net unfolding: Hickmott et al. 2007, Bonet et al. 2008

Split problems into subproblems (Factored planning)

Amir and Engelhardt 2003, Brafman and Domshlak 2006, Brafman and Domshlak 2008
Factored planning: principles

Each component is a planning problem with its own resources and actions.

Goal

Find a set of compatible local plans: they can be interleaved into a global plan.
Factored planning: principles

Each component is a planning problem with its own resources and actions

The components interact by resources and/or actions

Goal

Find a set of compatible local plans: they can be interleaved into a global plan
# Factored planning: our contribution

Prior to this thesis, reasoning on the number of synchronizations:
- Absence of solution *can not be detected*
- Cost-optimality of plans *can not be achieved*

## Our contribution

Two new approaches to *factored planning*, allowing to find *cost-optimal* plans with *distributed* algorithms

### Top-down approach

Successive restrictions of the sets of local plans

### Bottom-up approach

Progressive construction of a local plan per component
Top-down approach
Factored cost-optimal planning using message passing algorithms
Centralized planning problem = weighted automaton

Set of actions $\Sigma$
- The words are the plans
- The words with minimal cost are the cost-optimal plans

Goal
Find a minimal cost word in a weighted automaton
Factored planning problem

Components are weighted automata

They interact by their shared actions: formalization using the notion of *synchronous product*

**Goal**

In $A = A_1 \times \cdots \times A_n$, find a tuple $(w_1, \ldots, w_n)$ of words which are all *compatible* and *minimize* the sum of their cost, *without computing* $A$. 
Factored planning problem: example

Centralized plans: $\beta d\gamma$ and $d\beta\gamma$

Factored/distributed/concurrent plan: $(\beta, \beta\gamma, d\gamma)$
Projection: from global plans to local plans

Projection reduces a **global plan** to the actions of a **particular component**

\( \Pi_{\Sigma'} \) corresponds to:

1. Replace each action not in \( \Sigma' \) by \( \varepsilon \)
2. Perform \( \varepsilon \)-reduction (to the left)
3. (Minimize)

- Diagrams showing the transition from the global plan to the local plan, labeled with state transitions and corresponding actions.
MPA: computing $\Pi_{\Sigma_i}(\mathcal{A})$ without computing $\mathcal{A}$

Central properties of the projection of $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$

1. any *cost-optimal* word $w$ of $\mathcal{A}$ can be projected into a *cost-optimal* word $w_i$ of $\Pi_{\Sigma_i}(\mathcal{A})$, moreover $c(w) = c_i(w_i)$
2. any *cost-optimal* word $w_i$ of $\Pi_{\Sigma_i}(\mathcal{A})$ is the projection of a *cost-optimal* word $w$ of $\mathcal{A}$, moreover $c_i(w_i) = c(w)$

Consequence

Taking the minimal cost word in each $\Pi_{\Sigma_i}(\mathcal{A})$ gives a cost-optimal global plan (hypothesis: it is unique)

Building the $\Pi_{\Sigma_i}(\mathcal{A})$ by local computations

Successive refinements of the $\mathcal{A}_i$ from the constraints imposed by their neighbours
How to get the $\Pi_{\Sigma_i}(A)$: the message passing algorithms

**Fundamental property (conditional independence)**

$$\Pi_{\Sigma_1 \cap \Sigma_2}(A_1 \times A_2) \equiv \Pi_{\Sigma_1 \cap \Sigma_2}(A_1) \times \Pi_{\Sigma_1 \cap \Sigma_2}(A_2)$$

**Application:**

$$\begin{array}{c}
A_1 \quad \Sigma_1 \cap \Sigma_2 \quad A_2 \quad \Sigma_2 \cap \Sigma_3 \quad A_3
\end{array}$$

$$\Pi_{\Sigma_1}(A) = \Pi_{\Sigma_1}(A_1 \times A_2 \times A_3) \equiv \Pi_{\Sigma_1}(A_1) \times \Pi_{\Sigma_1}(A_2 \times A_3) \equiv A_1 \times \Pi_{\Sigma_1 \cap \Sigma_2}(A_2 \times A_3) \equiv A_1 \times \Pi_{\Sigma_1 \cap \Sigma_2}(A_2 \times \Pi_{\Sigma_2 \cap \Sigma_3}(A_3))$$
How to get the $\Pi_{\Sigma_i}(\mathcal{A})$: the message passing algorithms

**Fundamental property (conditional independence)**

$$\Pi_{\Sigma_1 \cap \Sigma_2}(\mathcal{A}_1 \times \mathcal{A}_2) \equiv \mathcal{L} \Pi_{\Sigma_1 \cap \Sigma_2}(\mathcal{A}_1) \times \Pi_{\Sigma_1 \cap \Sigma_2}(\mathcal{A}_2)$$

**Application:**

$$\Pi_{\Sigma_1}(\mathcal{A}) = \Pi_{\Sigma_1}(\mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3)$$

$$\equiv \mathcal{L} \Pi_{\Sigma_1}(\mathcal{A}_1) \times \Pi_{\Sigma_1}(\mathcal{A}_2 \times \mathcal{A}_3)$$

$$\equiv \mathcal{L} \mathcal{A}_1 \times \Pi_{\Sigma_1 \cap \Sigma_2}(\mathcal{A}_2 \times \mathcal{A}_3)$$

$$\equiv \mathcal{L} \mathcal{A}_1 \times \Pi_{\Sigma_1 \cap \Sigma_2}(\mathcal{A}_2 \times \Pi_{\Sigma_2 \cap \Sigma_3}(\mathcal{A}_3))$$
Example

Messages from $A_1$ to $A_3$:

$\Pi_{\alpha,\beta} (A_I)$  $A_2 \land \Pi_{\alpha,\beta} (A_I)$  $\Pi_{\gamma} (A_2 \land \Pi_{\alpha,\beta} (A_I))$  $A_3 \land \Pi_{\gamma} (A_2 \land \Pi_{\alpha,\beta} (A_I))$
Example

Messages from $A_3$ to $A_1$:
Example

Updated components:

\[ \Pi_{\alpha, \beta, a, b} (A) \]

\[ \Pi_{\alpha, \beta, \gamma, c} (A) \]

\[ \Pi_{\gamma, d} (A) \]
Message passing algorithms: main results generalized

If $A = A_1 \times \cdots \times A_n$ has a tree shaped interaction graph, the message passing algorithm converges and returns $A'_i \equiv \prod_{\Sigma_i}(A)$ for each $A_i$.

---

Distoplan: presentation

Distoplan

C++ implementation of the message passing on weighted automata, on top of openFST\(^2\) and the HSP*’s parser\(^3\)

A benchmark: philosophers from IPC4

\[ p_1 \cup f_4 \cup f_1 \cup p_4 \cup p_2 \cup f_3 \cup f_2 \cup p_3 \]

\(^2\)http://www.openfst.org/

\(^3\)Patrik Haslum, 4\(^{th}\) IPC Booklet, 2004
Distoplan: presentation

Distoplan

C++ implementation of the message passing on weighted automata, on top of openFST\(^2\) and the HSP*'s parser\(^3\)

A benchmark: philosophers from IPC4

\(^2\)http://www.openfst.org/

\(^3\)Patrik Haslum, 4\(^{th}\) IPC Booklet, 2004
Conclusions

- Practical solving of planning problems
- Can be more efficient than centralized search
- Difficulty: find decompositions

---

4 Eric Fabre, Loïg Jezequel, Patrik Haslum, and Sylvie Thiébaux, 
Cost-Optimal Factored Planning: Promises and Pitfalls, ICAPS 2010
Extension 1: read arcs in networks of automata\(^5\)

\(^5\)Loïg Jezequel and Eric Fabre, *Networks of Automata with Read Arcs: A Tool for Distributed Planning*, IFAC World Congress 2011
Extension 1: read arcs in networks of automata

The message passing algorithms extend to this setting with minor modifications.

Read arcs
Automata → automata with readings/writings on transitions

Theorem

Loïg Jezequel and Eric Fabre, *Networks of Automata with Read Arcs: A Tool for Distributed Planning*, IFAC World Congress 2011
Extension 2: turbo planning

Starting point

When interaction graphs contain cycles:

Existing solution

Tree decomposition of graphs:
- Not all parameters taken into account
- Tree-width can be huge

---

Extension 2: Turbo planning

Starting point
When interaction graphs contain cycles:

Turbo planning
Ignore cycles and perform approximate planning

Result: \( A'_i \) such that \( \mathcal{L}(\Pi_{\Sigma_i}(A)) \subseteq \mathcal{L}(A'_i) \subseteq \mathcal{L}(A_i) \)

---

\(^6\)Loïg Jezequel and Eric Fabre, *Turbo Planning*, WODES 2012
Extension 2: convergence issues of turbo planning

As a constraint solving problem
- Convergence in (possibly) infinite time
- Convergence in finite time for words of small length

As an optimization problem
- Costs diverge in general
- Normalization:
  - Costs of optimal paths stabilize
  - Costs of other paths still diverge
Extension 2: experiments on turbo planning

Problem shapes

Results
- Fast convergence (always less than 5 iterations)
- Promising quality of the solutions found (70% of solutions cost less than 10% more than the optimal)
Extension 2: experiments on turbo planning

Problem shapes

Results
- Fast convergence (always less than 5 iterations)
- Promising quality of the solutions found (70% of solutions cost less than 10% more than the optimal)

Open question
Theoretical explanation of this efficiency
Bottom-up approach
Cost-optimal planning using a distributed version of A*
A*: a best-first search algorithm

Rank of a node

Most promising node: \( s^* = \arg\min_s (g(s) + h(s)) \)
Always expand from \( s^* \) first
A*: a best-first search algorithm

**Rank of a node**

Most promising node: \( s^* = \text{argmin}_s (g(s) + h(s)) \)

Always expand from \( s^* \) first
A*: a best-first search algorithm

Termination
When $s_f$ is the most promising node
A#: intuition

Goal
A pair of compatible paths
A#: intuition

Goal
A pair of compatible paths

Idea
Parallel constrained searches

Agent $\varphi$
$A^*$ with information from $\overline{\varphi}$
Compatible final states

The problem

- Two automata (not sharing actions)
- A colouring function on final states

**Goal:** find a path in each automaton such that:

- They both reach final states of the same colour
- The sum of their costs is minimal among such paths

![Diagram of automata and paths](image-url)
Compatible final states: ranking

\[ \phi \]

Locally

One heuristic \( h \) per color

\[ g(s_1) \quad g(s_2) \]

\[ s_1 \quad s_2 \]

\[ h(s_1, \text{blue}) \quad h(s_2, \text{blue}) \]

\[ h(s_1, \text{green}) \quad h(s_2, \text{green}) \]

\[ s_f \]
Compatible final states: ranking

Locally
One heuristic $h$ per color

Externally
information $\overline{H}$ from $\overline{\varphi}$

Rank(s)
$$g(s) + \min_c (h(s, c) + \overline{H}(c))$$
**Planning problems**

**Message passing algorithms**

**A#: a distributed A***

**Conclusion**

**Compatible final states: termination**

\[ h(s_1, \text{blue}) \]

\[ h(s_2, \text{blue}) \]

\[ h(s_1, \text{green}) \]

\[ h(s_2, \text{green}) \]

\[ g(s_1) \]

\[ g(s_2) \]

\[ s_1 \]

\[ s_2 \]

\[ s_f \]

\[ \varphi \]

**Achieved best costs** for colors

Termination

A goal with color \( c_s \) such that:

\[ g(s) + \overline{G}(c_s) \] lowest rank
Compatible final states: results\textsuperscript{7}

Theorem

When executed by $\varphi$ on any CFS problem:

- A# terminates,
- A# is sound,
- A# is complete,

assuming that $\varphi$ has access to $\overline{G}$ and $\overline{H}$

\textsuperscript{7}Loïg Jezequel and Eric Fabre, \textit{A#: A Distributed Version of A* for Factored Planning}, CDC 2012
Compatible final states: results\textsuperscript{7}

Theorem

When executed by $\varphi$ on any CFS problem:

- $A\#$ terminates,
- $A\#$ is sound,
- $A\#$ is complete,

assuming that $\varphi$ has access to $\overline{G}$ and $\overline{H}$

Theorem

$\overline{G}$ and $\overline{H}$ can be computed by $\overline{\varphi}$ along its own execution of $A\#$

\textsuperscript{7}Loïg Jezequel and Eric Fabre, $A\#$: A Distributed Version of A* for Factored Planning, CDC 2012
A#: from CFS to factored planning

CCP and factored planning with two components

- Colour = sequence of (shared) actions
- Number of colours cannot be bounded locally

**Consequence:** computation of $h$, $H$, and $G$ more difficult
A#: from CFS to factored planning

**CCP and factored planning with two components**

- Colour = sequence of (shared) actions
- Number of colours cannot be bounded locally

**Consequence:** computation of $h$, $H$, and $G$ more difficult

\[
s_1 \xrightarrow{g(s_1)} s_1 \xrightarrow{\alpha} s'_1 \xrightarrow{h(s'_1, \alpha)} s_f(\alpha)
\]
\[
s_1 \xrightarrow{g(s_2)} s_2 \xrightarrow{\beta} s''_1 \xrightarrow{h(s''_1, \beta)} s_f(\beta)
\]
\[
h(s_2) \xrightarrow{-} s_f
\]

**Theorem: termination**

As soon as the considered factored planning problem has a solution, A# terminates
A#: extension to larger interaction graphs

From the point of view of agent $\varphi_i$, any factored planning problem has two components:

$$A_i \quad \Pi_{k \neq i} A_k$$
A#: extension to larger interaction graphs

From the point of view of agent $\varphi_i$, any factored planning problem has two components:

$$A_i \quad \Pi_{k \neq i} A_k$$

Theorem

If the interaction graph is a tree $\overline{H}$ and $\overline{G}$ can be constructed using only information (messages) from the neighbours of $\varphi_i$.

![Diagram](attachment:image.png)
Conclusion and perspectives
Conclusion

Main contribution

Two planning algorithms allowing: distributed planning and cost-optimal planning

First approach

- Message passing algorithms + weighted automata calculus
- Implementation in Distoplan
- Reading variables
- Turbo planning (approximate methods for factored planning)

Second approach

- Distributed version of A*
- Proof of validity and implementability
One needs *benchmarks* for factored planning

How to *automatically decompose* planning problems?

Is it possible to benefit from *local concurrency*?

*When/why* turbo planning works?