Master in Computer Science speciality Research in Computer Science
University of Rennes 1

Distributed Optimal Planning in Large Distributed Systems

Loïg JEZEQUEL
ENS Cachan Bretagne

Advisor:

Eric Fabre
INRIA Rennes - Bretagne Atlantique

Introduction: planning problem

Planning problem

- From an initial state, reach a goal state, by choosing and scheduling actions
- Useful in many domains:
  - Artificial intelligence (actions of a robot)
  - Industrial production (scheduling of tasks)

Three levels of difficulty

1. Is there a solution ?
2. Give a solution
3. What is the best solution ?

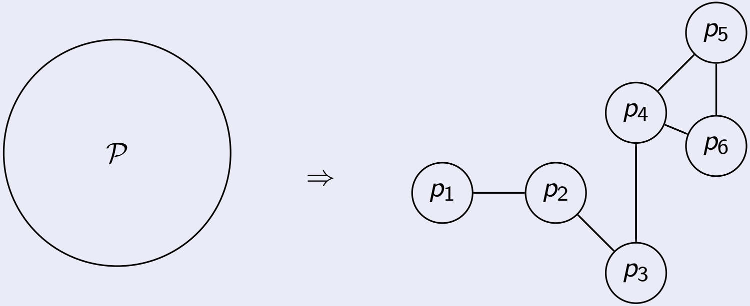
Solutions

- Sequence of actions
- Partial order of actions

Modular planning

Factored planning

Exploits the weak dependencies between variables



Challenges

- Find the factors (not explored here)
- Find local plans that combine into an (optimal) global plan

Outline

1. Modular optimal planning formalism
2. Message passing algorithm for modular optimal planning
3. Example

Automated planning

Automaton

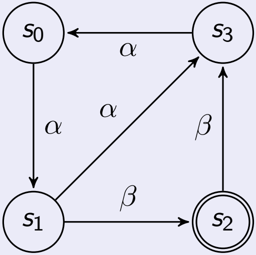
$\mathcal{A} = (S, T, \rightarrow, s_0, \lambda, \Lambda, F)$

Planning

Find a path which reaches the goal

Example

- $S = \{s_0, s_1, s_2, s_3\}$
- $T = \{t_1, t_2, t_3, t_4, t_5\}$
- $\Lambda = \{\alpha, \beta\}$
- $\lambda(t_1) = \lambda(t_3) = \lambda(t_5) = \alpha$
  $\lambda(t_2) = \lambda(t_4) = \beta$
- $F = \{s_2\}$

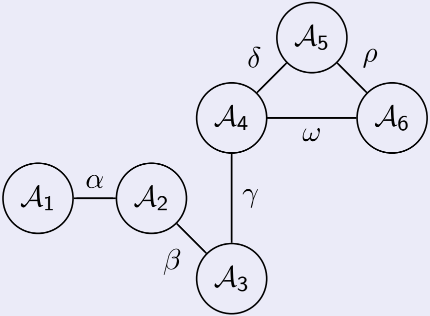


Modular planning

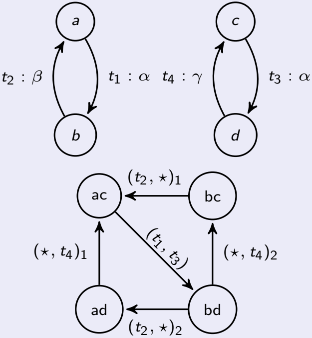
Planning problem

- Network of automata
- Composition by synchronous product
- Local/global plans

Network of automata



Synchronous product

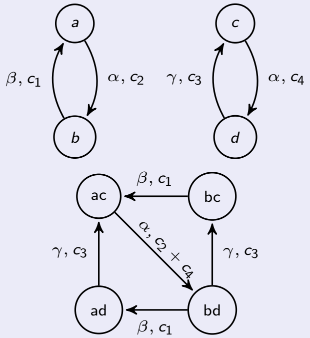


Modular optimal planning (1)

Weighted automaton

$\mathcal{A} = (S, T, \rightarrow, s_0, \lambda, \Lambda, F, c, c_i, c_F)$

Synchronous product



The problem

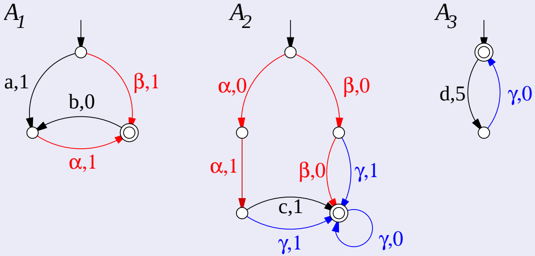
Given $\mathcal{A} = \mathcal{A}_1 \| \ldots \| \mathcal{A}_n$ find an accepted word w_i in each $\mathcal{A}_i$ such that:

- there is an accepted word w in $\mathcal{A}$ such that $\forall i,\ w_{|\Lambda_i} = w_i$,
- this w is optimal in $\mathcal{A}$,

without computing $\mathcal{A}$ nor $\mathcal{L}(\mathcal{A})$

Modular optimal planning (2)

Network of weighted automata



Two approaches

- Specialization of the message passing algorithm
  - find all the optimal plans (top-down approach)
- Distributed A^* algorithm
  - find one optimal plan (bottom-up approach)

Outline

Weighted languages

Definition

- set of couples (u, w) where u is a word and w a cost
- from (non-deterministic) WA: accepted words associated to minimal costs

Intuition

All the valid plans in an automaton associated with their optimal cost (a plan is a word of the language)

Operations

Projection natural projection + cost minimisation:
$P_{\{\alpha\}}(\{(\alpha\beta, 1), (\beta\alpha, 2), (\alpha\alpha, 3)\}) = \{(\alpha, 1), (\alpha\alpha, 3)\}$

Composition synchronous product of weighted languages (cost added when interleaving two words):
$\mathcal{L}(\mathcal{A}_1) \times \mathcal{L}(\mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1 \times \mathcal{A}_2)$

Toward the message passing algorithm

From $\mathcal{L} = \mathcal{L}_1 \times \cdots \times \mathcal{L}_n$ consider the $\mathcal{L}'_i = P_{\Lambda_i}(\mathcal{L})$. They have the following properties:

- if (u_i, c) is optimal in $\mathcal{L}'_i$, then there is u such that $u_{|\Lambda_i} = u_i$ and (u, c) is optimal in $\mathcal{L}$
- if (u, c) is optimal in $\mathcal{L}$, then there is u_i such that $u_i = u_{|\Lambda_i}$ and (u_i, c) is optimal in $\mathcal{L}'_i$

Objective

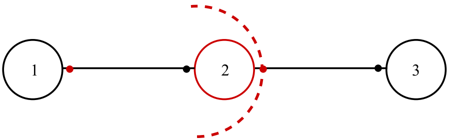
1. compute the $\mathcal{L}'_i$ (without computing $\mathcal{L}$)
2. find optimal local words in these $\mathcal{L}'_i$
3. interlace them to construct an optimal global plan

Axiom needed

$$\forall \Lambda_3 \supseteq \Lambda_1 \cap \Lambda_2,\ P_{\Lambda_3}(\mathcal{L}_1 \times \mathcal{L}_2) = P_{\Lambda_3}(\mathcal{L}_1) \times P_{\Lambda_3}(\mathcal{L}_2)$$

Message passing algorithm[1]

Based on the *communication graph*



The message passing algorithm

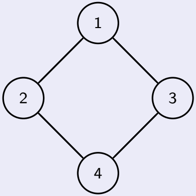
$\mathcal{M}_{i,j} \leftarrow \mathbb{I}, \forall (i,j) \in \mathcal{G}_{\mathcal{L}}$
until stability of messages **do**
  select an edge (i,j)
  $\mathcal{M}_{i,j} \leftarrow P_{\Lambda_i \cap \Lambda_j}\left(\mathcal{L}_i \times \left(\bigtimes_{k \in \mathcal{N}(i) \setminus j} \mathcal{M}_{k,i}\right)\right)$
done
$\mathcal{L}'_i \leftarrow \mathcal{L}_i \times \left(\bigtimes_{k \in \mathcal{N}(i)} \mathcal{M}_{k,i}\right), \forall i \in \{1, \ldots, n\}$

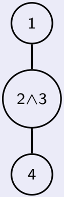
[1] Eric Fabre. *Bayesian Networks of Dynamic Systems*. HDR Thesis, 2007.

Drawbacks

Limitations due to the MPA

Communication graphs have to be trees





Limitations due to languages

Weighted languages are (potentially) infinite sets: necessity of a finite representation

Solution

Our weighted languages are ‘regular’... we will use weighted automata in our computations

Weighted automata[2]

Projection (on Σ)

1. replace any symbol from Σ by ϵ (keeping costs)
2. ϵ-removal
3. determinisation
4. minimisation

Composition

1. synchronous product
2. minimisation

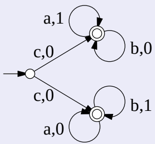
Validity

These operations are just implementations of the operations on weighted languages: they verify needed axioms

[2] Mehryar Mohri. *Weighted automata algorithms*. Springer, 2009.

Remarks on determinisation

- To use non-deterministic automata would work, but a part of optimisation is performed in determinisation
- Determinisation may have exponential cost, but it often reduces the size of the WA
- Not all WA are determinisable, but we can determinise partially



Experimental results

Random problems[a] of 1 to 50 components with 2 to 10 states and 5 minutes time limit for solving

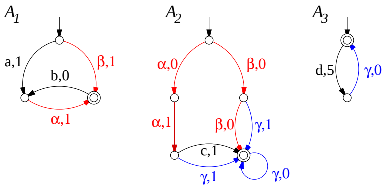
- 80% solved using determinisation
- 50% solved without determinisation (no more than 10 components)

[a] Bonet et al. *Directed unfolding of petri nets*. ToPNoC, 2008.

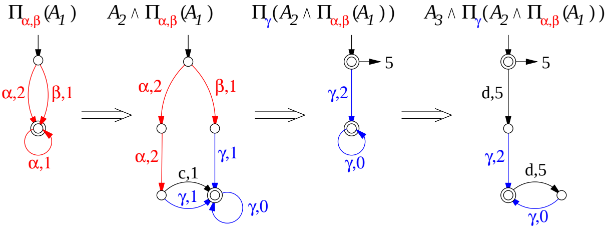
Outline

1. Modular optimal planning formalism
2. Message passing algorithm for modular optimal planning
3. Example

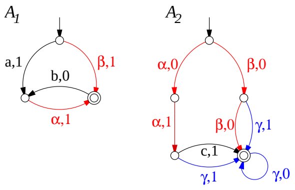
Example (1)



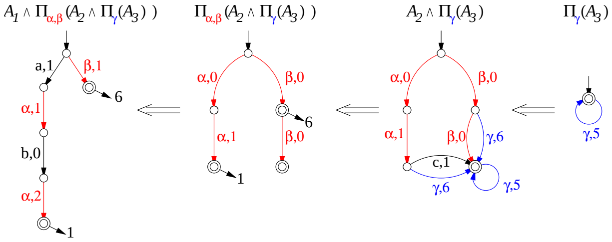
Messages from left to right:



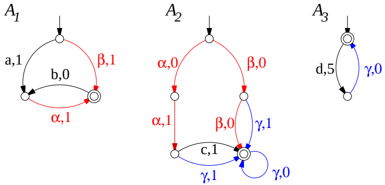
Example (2)



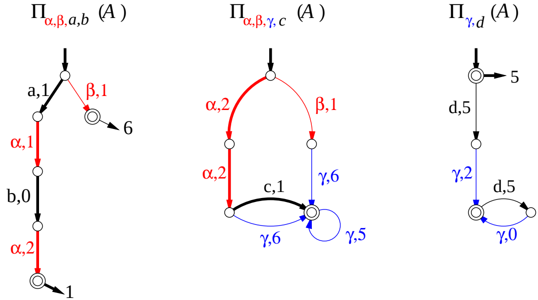
Messages from right to left:



Example (3)



Reduced components and optimal plans:



Conclusion and further work

Conclusion

- To our knowledge: the first method to perform factored optimal planning
- Some drawbacks:
  - Undeterminisable weighted automata
  - Communication graphs which are not trees

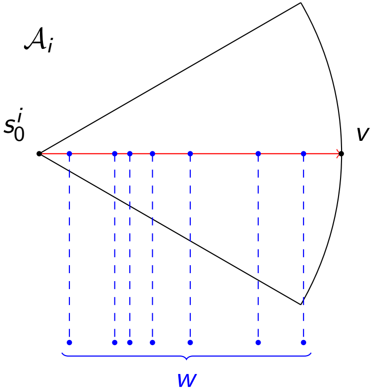
Further work

- Partial determinisation
- Turbo algorithms
- Partially ordered local solutions
- ...

Distributed A^*: context and principle

Context

- a network of automata: $\mathcal{A} = \mathcal{A}_1 \| \ldots \| \mathcal{A}_2$
- a collection of agents: $\varphi_1, \ldots, \varphi_n$, one for each automaton
- communication between agents by shared memory



φ_i maintains a queue Q_i

$$(v, w) \in Q_i$$

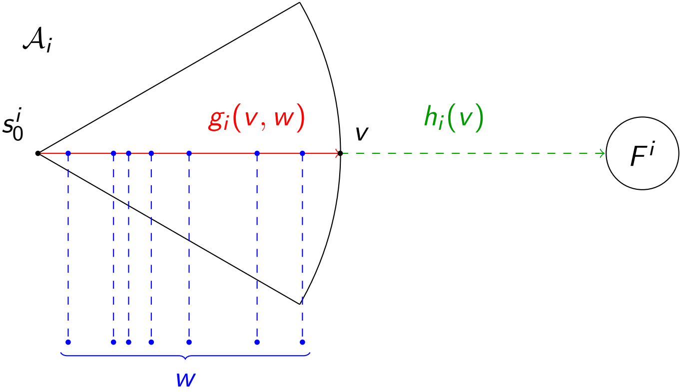
Ordering of Q_i

Much promising elements have to be at the head of Q_i

Distributed A^*: local cost

For $v \in S_i$ and w a synchronization word:

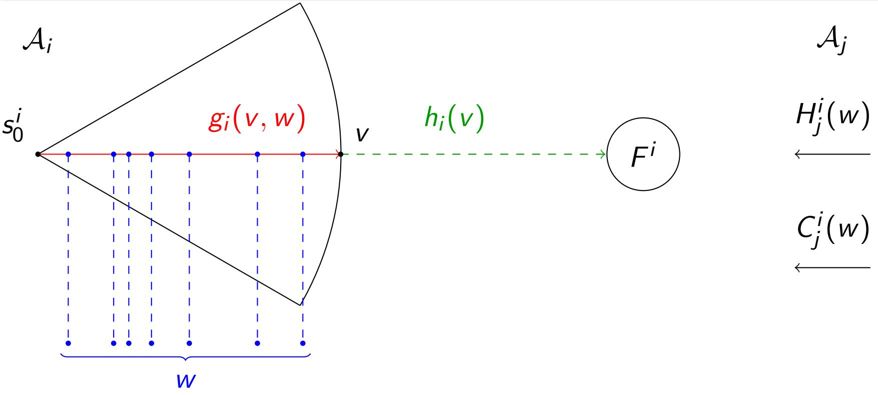
- $g_i(v, w)$ is the best cost known to reach v from s_i^0 with synchronization word w
- $h_i(v)$ is a lower bound on the cost of a path from v to F_i



Distributed A^*: global cost

For w a synchronization word:

- $H_j^i(w)$ is a lower bound on the cost of accepted paths in $\mathcal{A}_j$ that could be consistant with w
- $C_j^i(w)$ will eventually be the optimal cost of an accepted path in $\mathcal{A}_j$ which is consistant with w



Distributed A^*: intuition

Ranking cost of (v, w)

$$g(v, w) + h(v) + \sum_{j \in \mathcal{N}(i)} H^i_j(w)$$

Ordering of Q_i

couples (v, w) are ordered by ranking cost

Termination and validity

- special element $o(w)$ – with optimal cost for w as ranking cost – to terminate (when final state reached globally)
- special element $\tilde{o}(w)$ to ensure validity (when final state reached locally but not globally)