Turbo Planning

Eric FABRE and Loïg JEZEQUEL

INRIA Rennes Bretagne Atlantique, ENS Cachan Bretagne

WODES 2012

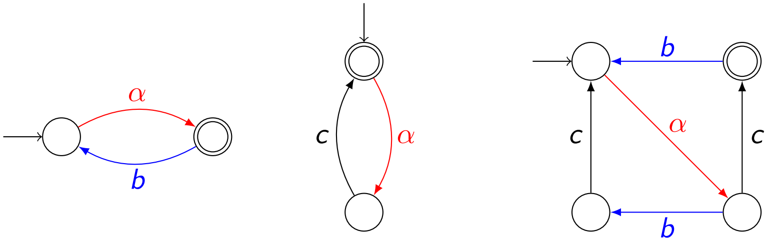
Outline

1. Message passing algorithm, Motivations
   - Planning problem
   - Message passing algorithm
   - Motivations

Our representation of planning problems

Network of automata

$\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$.



Goal

Find a tuple $(w_1, \ldots, w_n)$ of words that interleave into a word of $\mathcal{A}$.

Proposed resolution method [CDC09]

Idea

For each $\mathcal{A}_i$ compute an $\mathcal{A}'_i$ such that $\mathcal{L}(\mathcal{A}'_i) = \Pi_{\Sigma_i}(\mathcal{L}(\mathcal{A}))$.

Method

Use a message passing algorithm which progressively refines $\mathcal{A}_i$ by removing "bad" words (i.e do not fit with words of its neighbors).
Convergence: no more word can be removed (stability).

Condition for convergence

Convergence is ensured as soon as the graph of interaction between the $\mathcal{A}_i$ is a tree.

Why using turbo methods

Problem

The MPA only works on trees.

Existing solution

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

Proposed solution

- Turbo methods:
  - promising results in many domains

Outline

2 Principle of turbo methods

- What is computed
- Solution extraction

What is computed

Idea

Run MPA on non-tree interaction graphs.

Result after MPA convergence

From $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ one gets some $\mathcal{A}''_i$ such that:

$$\mathcal{L}(\mathcal{A}'_i) \subseteq \mathcal{L}(\mathcal{A}''_i) \subseteq \mathcal{L}(\mathcal{A}_i).$$

Extracting solutions on trees

Extracting a solution of $\mathcal{A}$ from the $\mathcal{A}'_i$ is straightforward with tree shaped interaction graphs.

1 let w_i be a word in some $\mathcal{A}'_i$

2 let w_j be a word compatible with w_i in some $\mathcal{A}'_j$ neighbor of $\mathcal{A}_i$

3 let w_k be a word compatible with w_i and w_j in some $\mathcal{A}'_k$ neighbor of $\mathcal{A}_i$ or $\mathcal{A}_j$

...

n+1 $(w_1, \ldots, w_n)$ can be interleaved into a word in $\mathcal{A}$

Extracting solutions in general

In general: extracting a solution from the $\mathcal{A}''_i$ in an interaction graph with cycles is more difficult than from the $\mathcal{A}'_i$ in a tree-shaped interaction graph.

May require backtracking.

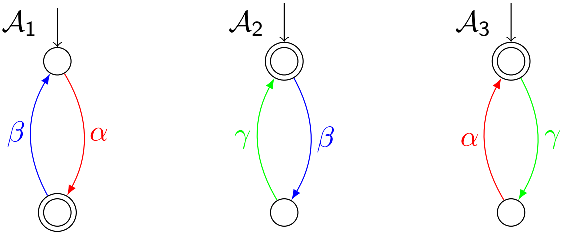
Our hope

Not much backtracking in general.

Outline

3. Turbo for constraint solving
   - Deciding convergence
   - Experimental results

No convergence in general



Condition for deciding convergence

Distance between automata

$$d(\mathcal{A}_1, \mathcal{A}_2) = \sum_{n=0}^{\infty} \frac{1}{2^n} \mathbf{1}_{\mathcal{L}_n(\mathcal{A}_1) \neq \mathcal{L}_n(\mathcal{A}_2)}$$

Condition for deciding convergence

$$d(\mathcal{A}_i^k, \mathcal{A}_i^{k+1}) \leq \epsilon$$

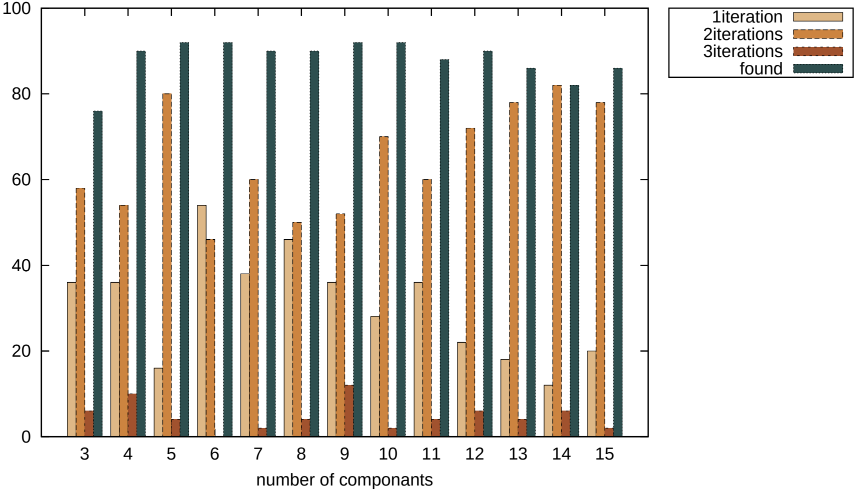
Always stops:

- updating $\mathcal{A}_i$ only removes words
- the number w such that $|w| \leq k$ is bounded

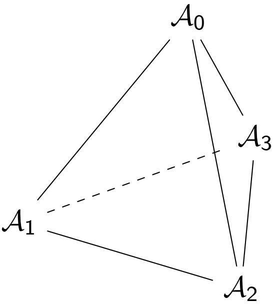
Experimental setting

- randomly generated automata
- two different shapes for interaction graphs
- selection of 50 difficult problems
- only problems with solutions
- no backtracking

Automata on circles: results



Automata on a tetrahedron



1 iteration: 2%
2 iterations: 52%
3 iterations: 42%
4 iterations: 4%

found: 85%

Outline

4 Turbo for optimization
- Normalization
- Experimental results

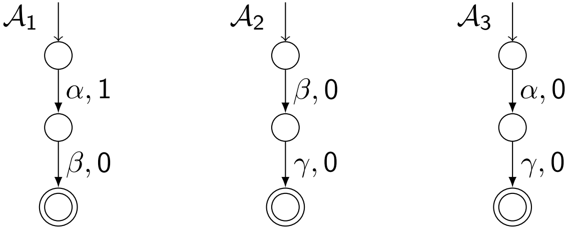
The problem

From now on we consider weighted automata.

Objective

Find close-to-optimal solutions: $(w_1, \ldots, w_n)$ minimizing $\Sigma_i c(w_i)$.

Necessity of normalization



Normalization in practice

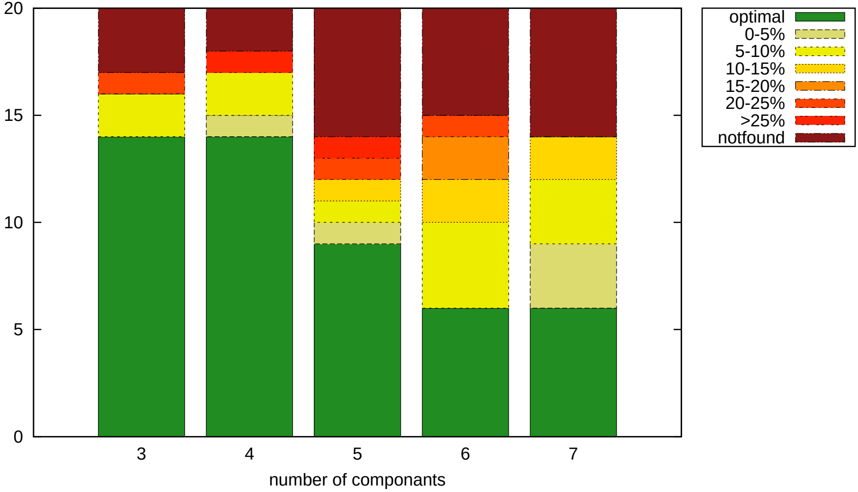
Two possible ways of normalizing:

- multiplicative normalization: $c' = c \times N$
  - easy to perform: multiply the cost of each transition by N
  - may change the difference between costs of paths
- additive normalization: $c' = c + N$
  - preserves the difference between costs of paths

A possible normalization constant:

- minimal cost of a path minus one

Automata on circles (20 problems per circle size)



Automata on a tetrahedron (50 problems)

found	opt	0-5%	5-10%	10-15%	15-20%	>20%
34	17	0	7	3	4	3

Conclusion

Summary of this work:

- experimental study of the use of turbo algorithms in planning
- methods for deciding convergence
- methods for normalization when dealing with costs

Outcome:

- approximate methods (in particular turbo algorithms) seem to be promising for factored planning

Further work:

- other normalization constants
- other distances between automata
- use on real planning problems