Exploring diversity of maximum a posteriori solutions in Markov random fields

Application to computational protein design

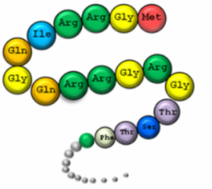
Manon Ruffini
PhD Student
Supervisors: Thomas Schiex (MIAT) and Sophie Barbe (LISBP)

INRA MIAT/LISBP

December 6th, 2018

Proteins - Folding problem

Most active molecules of life



Amino Acid sequence

→

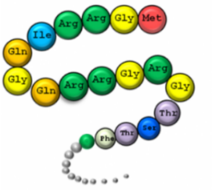
3D structure
(backbone + side-chains)

→

Reaction catalysis,
DNA replication,
molecule transport,
...

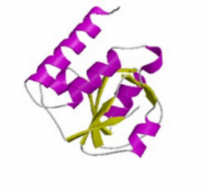
Function

Most active molecules of life



Amino Acid sequence

→

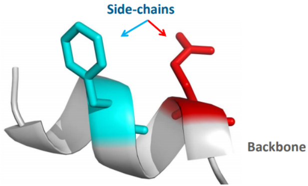


3D structure
(backbone + side-chains)

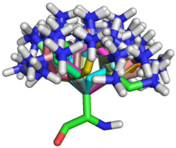
→

Reaction catalysis,
DNA replication,
molecule transport,
...

Function



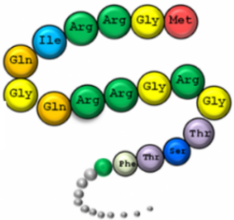
Side chains have different conformations (rotamers)



Change function

Function →

→

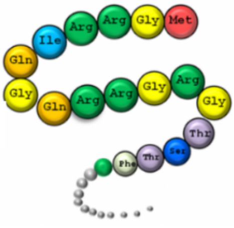


Protein Design : Inverse folding problem

Computational Protein Design

Function →

→



Protein Design : Inverse folding problem

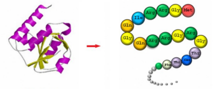
N positions 20 amino acids per positions (≈ 400 rotamers)	⇒	Search space = 20^N sequences (≈ 400^N conformations)

→ Need for computational methods to explore and prune the search space

Inverse folding problem

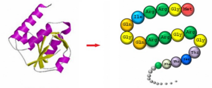
We have :

- 3D Structure (protein backbone)
- Rotamer library (amino acids and all their conformations)
- Energy function E



We have :

- 3D Structure (protein backbone)
- Rotamer library (amino acids and all their conformations)
- Energy function E



Minimum energy = Maximum stability

Given $\mathbf{x}$ sequence of rotamers (amino acids + conformations) :

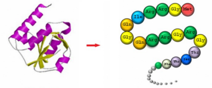
$$p(\mathbf{x}) = \exp(-\beta E(\mathbf{x}))$$

$\beta = \frac{1}{k_B T}$, k_B Boltzmann constant

Inverse folding problem

We have :

- 3D Structure (protein backbone)
- Rotamer library (amino acids and all their conformations)
- Energy function E



Minimum energy = Maximum stability

Given $\mathbf{x}$ sequence of rotamers (amino acids + conformations) :

$$p(\mathbf{x}) = \exp(-\beta E(\mathbf{x}))$$

$\beta = \frac{1}{k_B T}$, k_B Boltzmann constant

We want :

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} p(\mathbf{x}) = \arg\min_{\mathbf{x}} E(\mathbf{x})$$

Cost Function Networks

Cost Function Network $(X, D, \mathcal{E})$

- $X = (X_1, \ldots, X_n)$ set of variables, each with domain $D_i \in D$
- $\mathcal{E}$ set of unary and binary cost functions $\begin{array}{cccc} E_i & : & D_i & \rightarrow & [|0, \infty|] \\ E_{i,j} & : & D_i \times D_j & \rightarrow & [|0, \infty|] \end{array}$
- Cost of a solution $\mathbf{x} = x_1 \ldots x_n$:

$$E(\mathbf{x}) = E_\varnothing + \sum_{1 \leqslant i \leqslant n} E_i(x_i) + \sum_{1 \leqslant i < j \leqslant n} E_{ij}(x_i, x_j)$$

D. Allouche et al. (2014) Computational protein design as an optimization problem. In : Artif. Intell. Vol. 212. pp 59-79.

Cost Function Networks

Cost Function Network $(X, D, \mathcal{E})$

- $X = (X_1, \ldots, X_n)$ set of variables, each with domain $D_i \in D$
- $\mathcal{E}$ set of unary and binary cost functions $\begin{array}{cccc} E_i & : & D_i & \rightarrow & [|0,\infty|] \\ E_{i,j} & : & D_i \times D_j & \rightarrow & [|0,\infty|] \end{array}$
- Cost of a solution $\mathbf{x} = x_1 \ldots x_n$:

$$E(\mathbf{x}) = E_\emptyset + \sum_{1 \leqslant i \leqslant n} E_i(x_i) + \sum_{1 \leqslant i < j \leqslant n} E_{ij}(x_i, x_j)$$

≡ Markov Random Field

- $X = (X_1, \ldots, X_n)$ set of variables, each with domain $D_i \in D$
- Φ set of unary and binary potentials $\begin{array}{ccc} \varphi_i & = & e^{-\beta E_i} \\ \varphi_{i,j} & = & e^{-\beta E_{ij}} \end{array}$
- Potential of a solution $\mathbf{x} = x_1 \ldots x_n$:

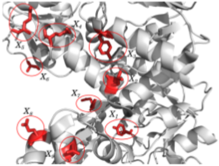
$$p(\mathbf{x}) = \varphi_\emptyset \times \prod_{1 \leqslant i \leqslant n} \varphi_i(x_i) \times \prod_{1 \leqslant i < j \leqslant n} \varphi_{ij}(x_i, x_j)$$

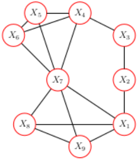
Cost Function Networks

Cost Function Network $(X, D, \mathscr{E})$

- $X = (X_1, \ldots, X_n)$ set of variables, each with domain $D_i \in D$
- $\mathscr{E}$ set of unary and binary cost functions $\begin{array}{cccc} E_i & : & D_i & \rightarrow & [|0,\infty|] \\ E_{i,j} & : & D_i \times D_j & \rightarrow & [|0,\infty|] \end{array}$
- Cost of a solution $\mathbf{x} = x_1 \ldots x_n$:

$$E(\mathbf{x}) = E_{\varnothing} + \sum_{1 \leqslant i \leqslant n} E_i(x_i) + \sum_{1 \leqslant i < j \leqslant n} E_{ij}(x_i, x_j)$$



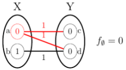


One variable per position
Domain = available rotamers
Cost functions = unary and binary energy terms

Global Minimum Energy Conformation (GMEC) : $\mathbf{x}^* = \arg\min_{\mathbf{x} \in D^X} E(\mathbf{x})$

⚠NP-complete

D. Allouche et al. (2014) Computational protein design as an optimization problem. In : Artif. Intell. Vol. 212. pp 59-79.



toulbar2

An exact solver for cost function networks

Home | Download | Documentation | Publications

NEWS: ***new toulbar2 source repository on GitHub***
NEWS: ***talk on toulbar2 latest algorithmic features at ISMP 2018***

Presentation

toulbar2 is an open-source C++ solver for cost function networks. It solves various combinatorial optimization problems.

The constraints and objective function are factorized in local functions on discrete variables. Each function returns a cost (a finite positive integer) for any assignment of its variables. Constraints are represented as functions with costs in {0,∞} where ∞ is a large integer representing forbidden assignments. toulbar2 looks for a non-forbidden assignment of all variables that minimizes the sum of all functions.

Its engine uses a hybrid best-first branch-and-bound algorithm exploiting soft arc consistencies. It incorporates a parallel variable neighborhood search method for better performances. See Publications.

toulbar2 won several competitions on Max-CSP (CPAI08) and probabilistic graphical models (UAI 2008, 2010, 2014 MAP task).

Authors

toulbar2 was originally developped by Toulouse (INRA MIAT) and Barcelona (UPC, IIIA-CSIC) teams, hence the name of the solver. Additional global cost functions were provided by the *Chinese University of Hong Kong* and Caen University (GREYC). It also includes codes from Marseille University (LSIS, tree decomposition heuristics) and *Ecole des Ponts ParisTech* (CERMICS/LIGM, INCOP local search solver).

A Python interface is available in NumberJack (Insight - University College Cork). A portfolio approach dedicated to UAI format instances is available here.

toulbar2 is currently maintained by Simon de Givry and hosted on GitHub.

Citations

Multi-Language Evaluation of Exact Solvers in Graphical Model Discrete Optimization
Barry Hurley, Barry O'Sullivan, David Allouche, George Katsirelos, Thomas Schiex, Matthias Zytnicki, Simon de Givry
Constraints, 21(3):413-434, 2016

Tractability-preserving Transformations of Global Cost Functions
David Allouche, Christian Bessiere, Patrice Boizumault, Simon de Givry, Patricia Gutierrez, Jimmy HM. Lee, Ka Lun Leung, Samir Loudni, Jean-Philippe Métivier, Thomas Schiex, Yi Wu

http ://www7.inra.fr/mia/T/toulbar2/

Exploring diversity of solutions

Global Minimum Energy Conformation (GMEC) : $\mathbf{x}^* = \arg\min_{\mathbf{x} \in D^X} E(\mathbf{x})$

BUT :

- Energy terms = approximations
- Energy fails at representing desirable properties other than stability
- There might be a better backbone for **x**

Global Minimum Energy Conformation (GMEC) : $\mathbf{x}^* = \arg\min_{\mathbf{x} \in D^X} E(\mathbf{x})$

BUT :

- Energy terms = approximations
- Energy fails at representing desirable properties other than stability
- There might be a better backbone for **x**

⟶ Set of **diverse** and **good quality** solutions
The sequence with the best properties is kept

Outline

Outline

Expressing diversity

$(X, D, \mathscr{E})$ cost function network

Expressing diversity : Hamming distance

Let $\mathbf{x} = x_1 \dots x_n$ and $\mathbf{x}' = x'_1 \dots x'_n$ be two solutions.

$$d(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} \mathbb{1}_{aa(x_i) \neq aa(x'_i)}$$

→ number of mutations (substitutions only) between $\mathbf{x}$ and $\mathbf{x}'$

Let $\{\mathbf{x}\} = \{\mathbf{x}^1, \dots, \mathbf{x}^M\}$ be a set of M solutions :

$$\Delta(\{\mathbf{x}\}) = \min\left\{ d(\mathbf{x}^i, \mathbf{x}^j) \;\middle|\; i \neq j \in [|1, M|] \right\}$$

Expressing diversity

$(X, D, \mathscr{E})$ cost function network

Expressing diversity : Hamming distance

Let $\mathbf{x} = x_1 \dots x_n$ and $\mathbf{x}' = x'_1 \dots x'_n$ be two solutions.

$$d(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} \mathbb{1}_{aa(x_i) \neq aa(x'_i)}$$

→ number of mutations (substitutions only) between $\mathbf{x}$ and $\mathbf{x}'$

Let $\{\mathbf{x}\} = \{\mathbf{x}^1, \dots, \mathbf{x}^M\}$ be a set of M solutions :

$$\Delta(\{\mathbf{x}\}) = \min \left\{ d(\mathbf{x}^i, \mathbf{x}^j) \,\middle|\, i \neq j \in [|1, M|] \right\}$$

dDistantMSet

$$\{\mathbf{x}\} = \arg\min \left\{ \sum_{j=1}^{M} E(\mathbf{x}^j) \,\middle|\, \Delta(\{\mathbf{x}\}) \geqslant d \right\}$$

E. Hebrard et al. (2005) Finding diverse and similar solutions in constraint programming. AAAI. Vol. 5.

A. KIRILLOV et al. (2015) Inferring M-best diverse labelings in a single one. In : Proceedings of the IEEE International Conference on Computer Vision. p. 1814-1822.

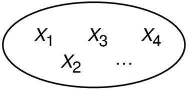
$$\{\mathbf{x}\} = \arg\min \left\{ \sum_{j=1}^{M} E(\mathbf{x}^j) \,\middle|\, \Delta(\{\mathbf{x}\}) \geqslant d \right\}$$

E. Hebrard et al. (2005) Finding diverse and similar solutions in constraint programming. AAAI. Vol. 5.

A. KIRILLOV et al. (2015) Inferring M-best diverse labelings in a single one. In : Proceedings of the IEEE International Conference on Computer Vision. p. 1814-1822.

*d*DistantMSet

$$\{\mathbf{x}\} = \arg\min \left\{ \sum_{j=1}^{M} E(\mathbf{x}^j) \;\middle|\; \Delta(\{\mathbf{x}\}) \geqslant d \right\}$$

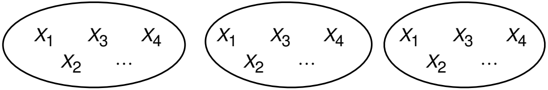


E. Hebrard et al. (2005) Finding diverse and similar solutions in constraint programming. AAAI. Vol. 5.

A. KIRILLOV et al. (2015) Inferring M-best diverse labelings in a single one. In : Proceedings of the IEEE International Conference on Computer Vision. p. 1814-1822.

*d*DistantMSet

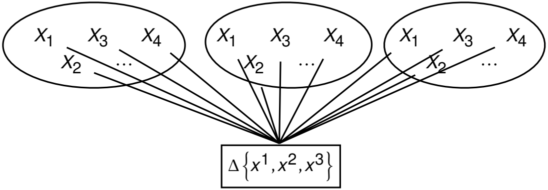
$$\{\mathbf{x}\} = \arg\min \left\{ \sum_{j=1}^{M} E(\mathbf{x}^j) \;\middle|\; \Delta(\{\mathbf{x}\}) \geqslant d \right\}$$



E. Hebrard et al. (2005) Finding diverse and similar solutions in constraint programming. AAAI. Vol. 5.

A. KIRILLOV et al. (2015) Inferring M-best diverse labelings in a single one. In : Proceedings of the IEEE International Conference on Computer Vision. p. 1814-1822.

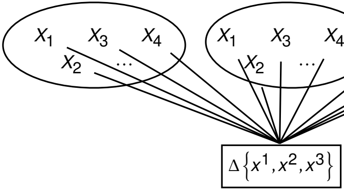
$$\{\mathbf{x}\} = \arg\min \left\{ \sum_{j=1}^{M} E(\mathbf{x}^j) \,\middle|\, \Delta(\{\mathbf{x}\}) \geqslant d \right\}$$



E. Hebrard et al. (2005) Finding diverse and similar solutions in constraint programming. AAAI. Vol. 5.

A. KIRILLOV et al. (2015) Inferring M-best diverse labelings in a single one. In : Proceedings of the IEEE International Conference on Computer Vision. p. 1814-1822.

$$\{\mathbf{x}\} = \arg\min \left\{ \sum_{j=1}^{M} E(\mathbf{x}^j) \,\middle|\, \Delta(\{\mathbf{x}\}) \geqslant d \right\}$$



$$E_\Delta(\delta) = \begin{cases} 0 & \text{if } \delta \geqslant d \\ \infty & \text{otherwise} \end{cases}$$

E. Hebrard et al. (2005) Finding diverse and similar solutions in constraint programming. AAAI. Vol. 5.

A. KIRILLOV et al. (2015) Inferring M-best diverse labelings in a single one. In : Proceedings of the IEEE International Conference on Computer Vision. p. 1814-1822.

dDISTANTMSET - Greedy approximation

Assume $\mathbf{s}^1, \ldots, \mathbf{s}^{M-1}$ are computed solutions :

$$\mathbf{x}^M = \arg\min_{\mathbf{x}} \{E(\mathbf{x})\} \text{ s.t. } \begin{cases} d(\mathbf{x}, \mathbf{s}^1) \geqslant d \\ \vdots \\ d(\mathbf{x}, \mathbf{s}^{M-1}) \geqslant d \end{cases}$$

$\equiv$ Find optimum in cost function network with additional diversity constraints

$\longrightarrow$ How do we express the diversity constraints ?

D. Batra et al. (2012). Diverse m-best solutions in markov random fields. In European Conference on Computer Vision (pp. 1-16). Springer, Berlin, Heidelberg.

Outline

Lagrangian Relaxation

$$(P) \qquad E^* = \min_{\mathbf{x}} \{ E(\mathbf{x}) \} \qquad \text{s.t.} \qquad \begin{cases} d(\mathbf{x}, \mathbf{s}^1) \geqslant d \\ \vdots \\ d(\mathbf{x}, \mathbf{s}^{M-1}) \geqslant d \end{cases}$$

Lagrangian Relaxation

$$(P) \qquad E^* = \min_{\mathbf{x}} \{ E(\mathbf{x}) \} \qquad \text{s.t.} \quad \begin{cases} d(\mathbf{x}, \mathbf{s}^1) \geq d \\ \vdots \\ d(\mathbf{x}, \mathbf{s}^{M-1}) \geq d \end{cases}$$

$$q(\lambda) = \min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \sum_{j=1}^{M-1} \lambda_j \left(d(\mathbf{x}, \mathbf{s}^j) - d \right) \right\}$$

$$(D) \qquad Q^* = \max_{\lambda} q(\lambda)$$

$q(\lambda)$ = Optimum CFN with unary penalties

$$q(\lambda) = \min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \sum_{i=1}^{n} \left(\sum_{j=1}^{M-1} \lambda_j \mathbb{1}_{aa(x_i) \neq aa(s_i^j)} \right) - \sum_{j=1}^{M-1} \lambda_j d \right\}$$

Lagrangian Relaxation

$$(P) \qquad E^* = \min_{\mathbf{x}} \{ E(\mathbf{x}) \} \qquad \text{s.t.} \qquad \begin{cases} d(\mathbf{x}, \mathbf{s}^1) \geqslant d \\ \vdots \\ d(\mathbf{x}, \mathbf{s}^{M-1}) \geqslant d \end{cases}$$

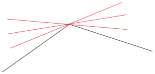
$$q(\lambda) = \min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \sum_{j=1}^{M-1} \lambda_j \left(d(\mathbf{x}, \mathbf{s}^j) - d \right) \right\}$$

$$(D) \qquad Q^* = \max_{\lambda} q(\lambda)$$

$q(\lambda)$ = Optimum CFN with unary penalties

$$q(\lambda) = \min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \sum_{i=1}^{n} \left(\sum_{j=1}^{M-1} \lambda_j \mathbb{1}_{aa(x_i) \neq aa(s_i^j)} \right) - \sum_{j=1}^{M-1} \lambda_j d \right\}$$

q : concave, piecewise differentiable $\longrightarrow$ **supergradient ascent**



D. Batra et al. (2012). Diverse m-best solutions in markov random fields. In European Conference on Computer Vision (pp. 1-16). Springer, Berlin, Heidelberg.

Supergradient ascent on q

$$Q^* = \max_{\lambda} q(\lambda) = \max_{\lambda} \min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \sum_{j=1}^{M-1} \lambda_i \left(d(\mathbf{x}, \mathbf{s}^j) - d \right) \right\}$$

Idea :

- λ too small → diversity constraint not satisfied
- λ too high → problem too constrained : we might miss good quality solutions

Supergradient ascent on q

$$Q^* = \max_{\lambda} q(\lambda) = \max_{\lambda} \min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \sum_{j=1}^{M-1} \lambda_i \left(d(\mathbf{x}, \mathbf{s}^j) - d \right) \right\}$$

Idea :

- λ too small → diversity constraint not satisfied
- λ too high → problem too constrained : we might miss good quality solutions

Iteratively

1. Set $\lambda = \lambda_0$
2. Solve $q(\lambda) = \min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \sum_{j=1}^{M-1} \lambda_j \left(d(\mathbf{x}, \mathbf{s}^j) - d \right) \right\}$
3. Adjust λ
4. Go to step 2 until stopping criterion is reached

Supergradient ascent on q

$$Q^* = \max_{\lambda} q(\lambda) = \max_{\lambda} \min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \sum_{j=1}^{M-1} \lambda_i \left(d(\mathbf{x}, \mathbf{s}^j) - d \right) \right\}$$

Idea :

- λ too small → diversity constraint not satisfied
- λ too high → problem too constrained : we might miss good quality solutions

Iteratively

1. Set $\lambda = \lambda_0$
2. Solve $q(\lambda) = \min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \sum_{j=1}^{M-1} \lambda_j \left(d(\mathbf{x}, \mathbf{s}^j) - d \right) \right\}$
3. Adjust λ
4. Go to step 2 until stopping criterion is reached

? Step 3 : Stepsize to adjust λ ? (several strategies investigated)

? Step 4 : Stopping criterion ?

? Correctness ? (duality gap)

Lagrangian relaxation

- Stepsize $\begin{cases} \text{Constant} & \\ \text{Square summable, non summable} & \frac{1}{k} \\ \text{Non summable diminishing} & \frac{1}{\sqrt{k}} \\ \text{Polyak} & \frac{q_{best}^k - q(\lambda^k) + \frac{1}{\sqrt{k}}}{||u_k||_2^2} \end{cases}$
- Stopping criterion : empirical

Duality gap

The Hamming dissimilarity does not lead to a tight Lagrangian relaxation and may leave a duality gap.

Lagrangian relaxation

- Stepsize

Stepsize	
Constant	
Square summable, non summable	$\frac{1}{k}$
Non summable diminishing	$\frac{1}{\sqrt{k}}$
Polyak	$\frac{q_{best}^k - q(\lambda^k) + \frac{1}{\sqrt{k}}}{\|\|u_k\|\|_2^2}$

- Stopping criterion : empirical

Duality gap

The Hamming dissimilarity does not lead to a tight Lagrangian relaxation and may leave a duality gap.

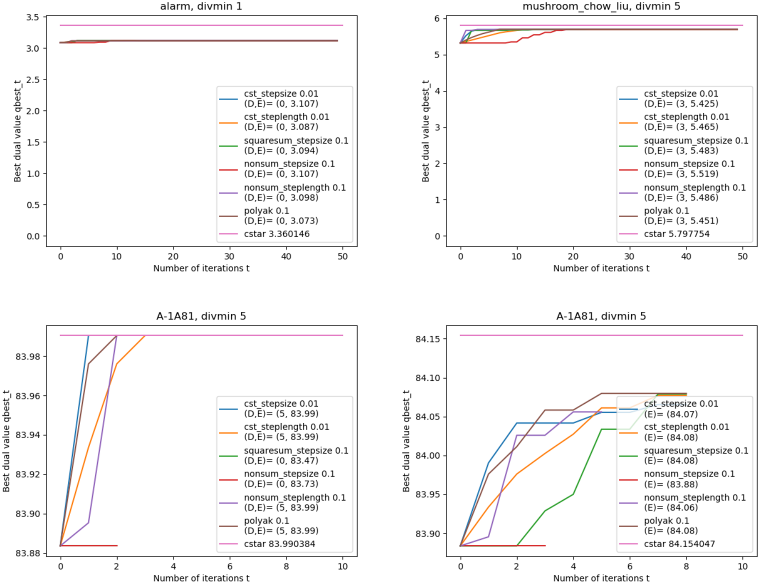
Datasets

- Bayesian Networks *(alarm)*
- Tree Structured Networks *(mushroom)*
- CPD instances *(A-1A81)*

S. Boyd et al. Subgradients. Lecture notes for EE364b, Stanford University, Spring 2014-15

D. Batra et al. (2012). Diverse m-best solutions in markov random fields. In European Conference on Computer Vision (pp. 1-16). Springer, Berlin, Heidelberg.

Preliminary results



Outline

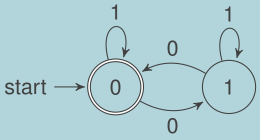
Weighted Regular Constraint (WREGULAR)

Constraint based on the membership in a regular language, defined by an automaton.

Automaton

A deterministic finite state automaton is a quintuple $(\Sigma, S, s_0, \delta, F)$ where :

- Σ is the input alphabet
- Q is a finite set of states
- $s_0 \in Q$ is an initial state
- $\delta : Q \times \Sigma \rightarrow Q$ is the state-transition function
- $F \subset Q$ is a set of final states



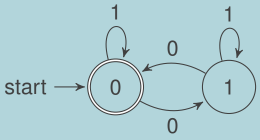
Weighted Regular Constraint (WREGULAR)

Constraint based on the membership in a regular language, defined by an automaton.

Automaton

A deterministic finite state automaton is a quintuple $(\Sigma, S, s_0, \delta, F)$ where :

- Σ is the input alphabet
- Q is a finite set of states
- $s_0 \in Q$ is an initial state
- $\delta : Q \times \Sigma \rightarrow Q$ is the state-transition function
- $F \subset Q$ is a set of final states

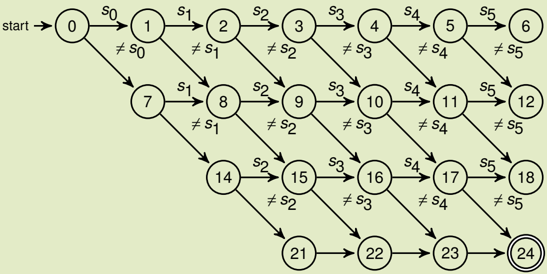


A word $\mathbf{x} \in \Sigma^*$ is accepted by the automaton if there exists a set of transitions from the initial state s_0 to a final state f labeled by the letters of $\mathbf{x}$.

In ToulBar2 :

- Global constraint described by an automaton
- Alphabet = domain values
- Costs on initial state, transition and final states

Ex : Diversity $\geqslant 3$ from $\mathbf{s} = s_0 \dots s_5$



All costs $= 0$

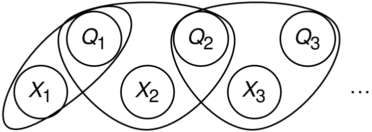
Decomposition of the regular constraint : Counting variables

For a solution $\mathbf{s}^j$ and minimum diversity value d :

- Set of additional variables $Q_1, \ldots, Q_n$, with $Q_i = d(x_1 \ldots x_i, s_1^j \ldots s_i^j)$ $\quad D_{Q_i} = \{0, d\}$
- Additional cost functions to ensure

$$Q_i = Q_{i-1} + \mathbb{1}_{aa(x_i) \neq aa(s_i^j)}$$

- Unary cost on Q_n



$$E_{Q_n}(q_n) = \begin{cases} 0 & \text{if } q_n = d \\ \infty & \text{otherwise} \end{cases}$$

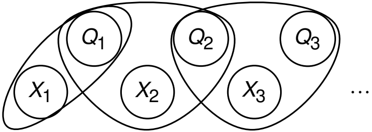
Decomposition of the regular constraint : Counting variables

For a solution $\mathbf{s}^j$ and minimum diversity value d :

- Set of additional variables $Q_1, \ldots, Q_n$, with $Q_i = d(x_1 \ldots x_i, s_1^j \ldots s_i^j)$ $\quad D_{Q_i} = \{0, d\}$
- Additional cost functions to ensure

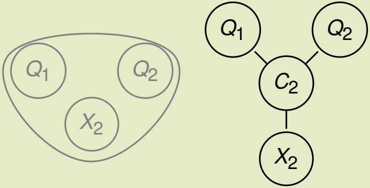
$$Q_i = Q_{i-1} + \mathbb{1}_{aa(x_i) \neq aa(s_i^j)}$$

- Unary cost on Q_n



$$E_{Q_n}(q_n) = \begin{cases} 0 & \text{if } q_n = d \\ \infty & \text{otherwise} \end{cases}$$

From ternary to binary functions : Hidden variable representation

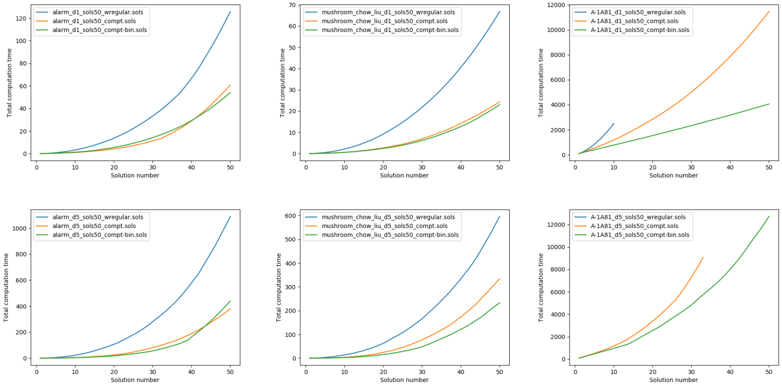


$$D_{C_i} = \left\{ \begin{cases} (q, 0, q) \\ (q, 1, q+1) \end{cases} \middle| \, q \in D_{Q_i} \right\}$$

Bessièreet al. (2011). Decomposing global cost functions. Soft'11 - Principles and Practice of Constraint Programming (pp. 16-30).

J.Larrosa, R. Dechter (2000). On the dual representation of non-binary semiring-based CSPs. In CP'2000 workshop on soft constraints.

Preliminary results



Outline

(X, D, E) CFN ; $\mathbf{s} = s_1 \dots s_n$ solution

MDD — Multi-valued version of BDD

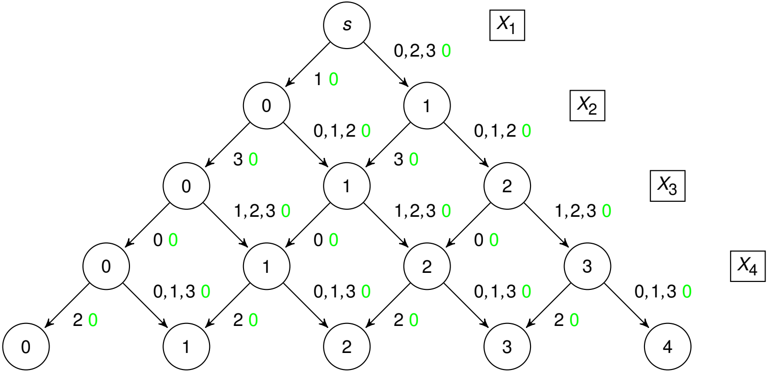
- Layered automaton
- One layer L_i per variable X_i
- Edge labeled v_i from L_i to L_{i+1} = Assignment of X_i to v_i
- Each node u in layer L_i has a state $I_u = d(x_1 \dots x_{i-1}, s_1 \dots s_{i-1})$
- Weights on edges

Diversity from several solutions $(\mathbf{s}^j)_j$:

$$I_u = \left(d(x_1 \dots x_{i-1}, s_1^j \dots s_{i-1}^j)\right)_j$$

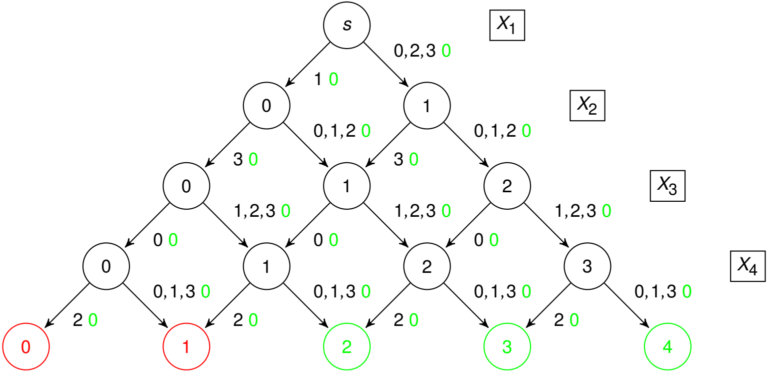
Example

$X = (X_1, X_2, X_3, X_4)$
$\forall i, D_i = \{0, 1, 2, 3\}$
$\mathbf{s} = 1\ 3\ 0\ 2 \quad d = 2$



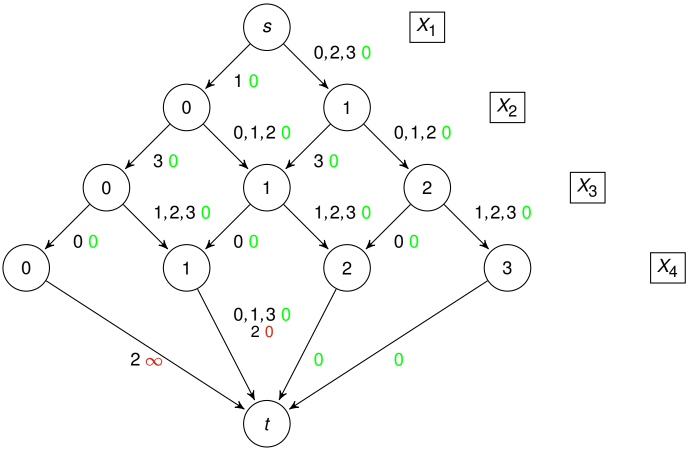
Example

$X = (X_1, X_2, X_3, X_4)$
$\forall i, D_i = \{0,1,2,3\}$
$\mathbf{s} = 1\ 3\ 0\ 2 \quad d = 2$



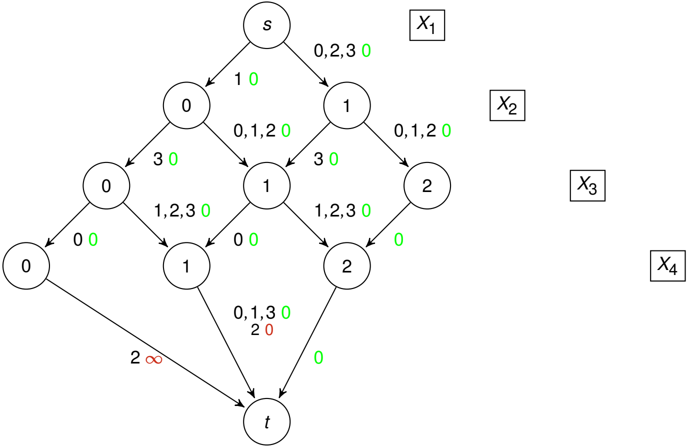
Example

$X = (X_1, X_2, X_3, X_4)$
$\forall i, D_i = \{0, 1, 2, 3\}$
$\mathbf{s} = 1\ 3\ 0\ 2 \quad d = 2$



Example

$X = (X_1, X_2, X_3, X_4)$
$\forall i, D_i = \{0, 1, 2, 3\}$
$\mathbf{s} = 1\ 3\ 0\ 2 \quad d = 2$

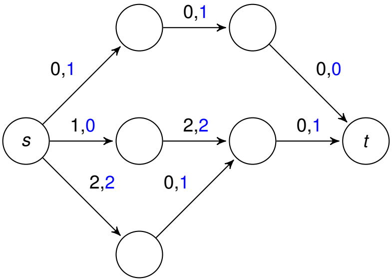


$\emptyset$-IC

The MDD cost function is said to be strongly $\emptyset$-inverse consistent (strongly $\emptyset$-IC) if there exists a tuple $\mathbf{x} \in D_X$ such that

$$MDD(\mathbf{x}) + \sum_{i=1}^{n} E_i(x_i) = 0$$

For u node in layer L_i, $\alpha^+[u]$ = smallest path weight (with unary costs) from s to u

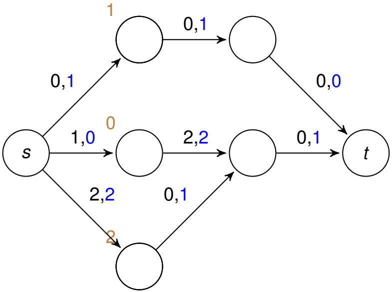


$\emptyset$-IC

The MDD cost function is said to be strongly $\emptyset$-inverse consistent (strongly $\emptyset$-IC) if there exists a tuple $\mathbf{x} \in D_X$ such that

$$MDD(\mathbf{x}) + \sum_{i=1}^{n} E_i(x_i) = 0$$

For u node in layer L_i, $\alpha^+[u]$ = smallest path weight (with unary costs) from s to u

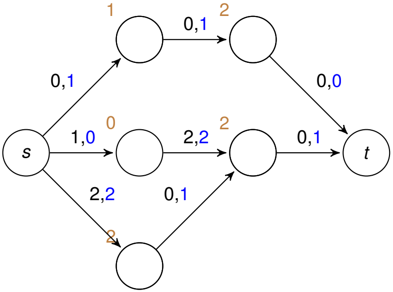


$\emptyset$-IC

The MDD cost function is said to be strongly $\emptyset$-inverse consistent (strongly $\emptyset$-IC) if there exists a tuple $\mathbf{x} \in D_X$ such that

$$MDD(\mathbf{x}) + \sum_{i=1}^{n} E_i(x_i) = 0$$

For u node in layer L_i, $\alpha^+[u]$ = smallest path weight (with unary costs) from s to u

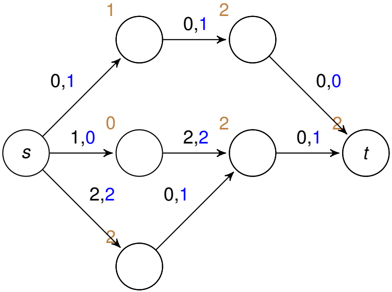


$\emptyset$-IC

The MDD cost function is said to be strongly $\emptyset$-inverse consistent (strongly $\emptyset$-IC) if there exists a tuple $\mathbf{x} \in D_X$ such that

$$MDD(\mathbf{x}) + \sum_{i=1}^{n} E_i(x_i) = 0$$

For u node in layer L_i, $\alpha^+[u]$ = smallest path weight (with unary costs) from s to u

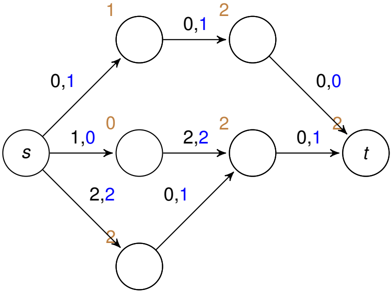


$\emptyset$-IC

The MDD cost function is said to be strongly $\emptyset$-inverse consistent (strongly $\emptyset$-IC) if there exists a tuple $\mathbf{x} \in D_X$ such that

$$MDD(\mathbf{x}) + \sum_{i=1}^{n} E_i(x_i) = 0$$

For u node in layer L_i, $\alpha^+[u]$ = smallest path weight (with unary costs) from s to u



Equivalence preserving transformation

$$\forall \mathbf{x}, MDD(\mathbf{x}) = MDD(\mathbf{x}) - 2$$

$$E_\emptyset = E_\emptyset + 2$$

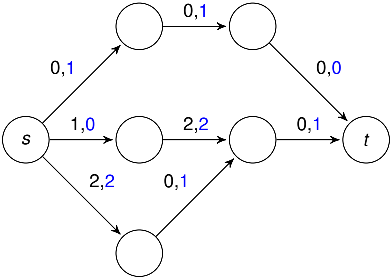
Cooper, M. C., De Givry, S., Sánchez, M., Schiex, T., Zytnicki, M., Werner, T. (2010). Soft arc consistency revisited. Artificial Intelligence, 174(7-8), 449-478.

AC

The MDD cost function is arc consistent if for all $X_i \in X$ and all $v_i \in D_{X_i}$, there exists a tuple $\mathbf{x}$ such that $\mathbf{x}[i] = v_i$ and $MDD(\mathbf{x}) = 0$.

For u node in layer L_i,	$\alpha[u]$	smallest path weight from s to u
	$\beta[u]$	smallest path weight from u to t

Example : $X_i = X_2; v_2 = 0$

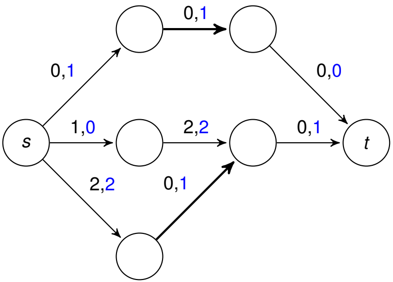


AC

The MDD cost function is arc consistent if for all $X_i \in X$ and all $v_i \in D_{X_i}$, there exists a tuple $\mathbf{x}$ such that $\mathbf{x}[i] = v_i$ and $MDD(\mathbf{x}) = 0$.

For u node in layer L_i,	$\alpha[u]$	smallest path weight from s to u
	$\beta[u]$	smallest path weight from u to t

Example : $X_i = X_2; v_2 = 0$



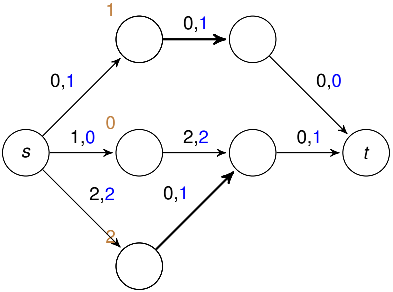
AC

The MDD cost function is arc consistent if for all $X_i \in X$ and all $v_i \in D_{X_i}$, there exists a tuple $\mathbf{x}$ such that $\mathbf{x}[i] = v_i$ and $MDD(\mathbf{x}) = 0$.

For u node in layer L_i,

$\alpha[u]$	smallest path weight from s to u
$\beta[u]$	smallest path weight from u to t

Example : $X_i = X_2; v_2 = 0$

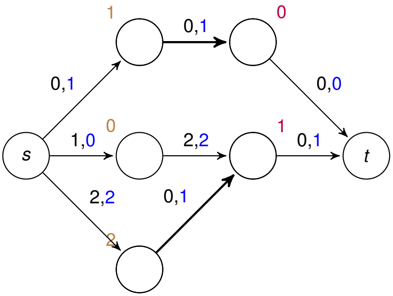


AC

The MDD cost function is arc consistent if for all $X_i \in X$ and all $v_i \in D_{X_i}$, there exists a tuple $\mathbf{x}$ such that $\mathbf{x}[i] = v_i$ and $MDD(\mathbf{x}) = 0$.

For u node in layer L_i, $\alpha[u]$ smallest path weight from s to u; $\beta[u]$ smallest path weight from u to t

Example : $X_i = X_2; v_2 = 0$

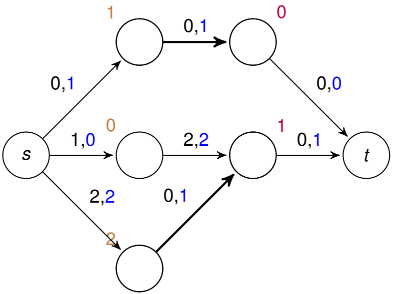


AC

The MDD cost function is arc consistent if for all $X_i \in X$ and all $v_i \in D_{X_i}$, there exists a tuple $\mathbf{x}$ such that $\mathbf{x}[i] = v_i$ and $MDD(\mathbf{x}) = 0$.

For u node in layer L_i, $\alpha[u]$ smallest path weight from s to u; $\beta[u]$ smallest path weight from u to t

Example : $X_i = X_2; v_2 = 0$



$$\min\{E(\mathbf{x}) \mid x_2 = 0\} = \min(1+1+0, 2+1+1) = 2$$

$$\forall \mathbf{x} \text{ s.t. } x_2 = 0, MDD(\mathbf{x}) = MDD(\mathbf{x}) - 2$$

$$E_2(0) = E_2(0) + 2$$

Cooper, M. C., De Givry, S., Sánchez, M., Schiex, T., Zytnicki, M., Werner, T. (2010). Soft arc consistency revisited. Artificial Intelligence, 174(7-8), 449-478.

Complexity

$(X, D, \mathscr{E})$ CFN ; $\mathbf{s}^1, \ldots, \mathbf{s}^M$ M solutions

The Div_{min} constraint

$$\text{Div}_{\text{min}}(\mathbf{x}, \mathbf{s}^1, \ldots, \mathbf{s}^M, d) = \begin{cases} 0 & \text{if } \left(\min_{1 \leqslant j \leqslant m} d(\mathbf{x}, \mathbf{s}^j)\right) \geqslant d \\ \top & \text{otherwise} \end{cases}$$

Complexity

$(X, D, \mathscr{E})$ CFN ; $\mathbf{s}^1, \dots, \mathbf{s}^M$ M solutions

The Div_{min} constraint

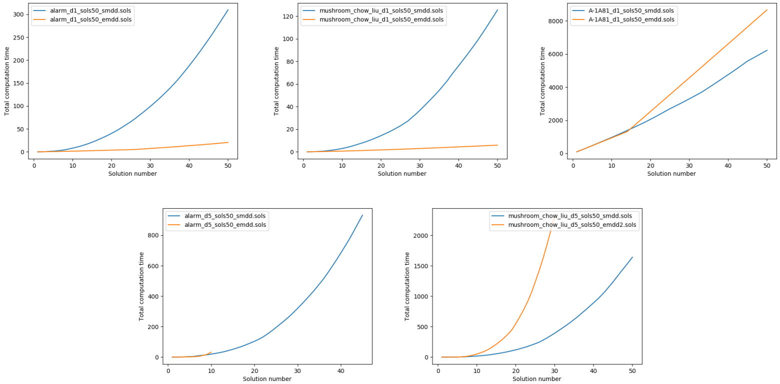
$$\text{Div}_{\text{min}}(\mathbf{x}, \mathbf{s}^1, \dots, \mathbf{s}^M, d) = \begin{cases} 0 & \text{if } \left(\min_{1 \leqslant j \leqslant m} d(\mathbf{x}, \mathbf{s}^j) \right) \geqslant d \\ \top & \text{otherwise} \end{cases}$$

$\emptyset$-IC is **NP-hard** to propagate on Div_{min}.

AC is **NP-hard** to propagate on Div_{min}.

E. Hebrard et al. (2005) Finding diverse and similar solutions in constraint programming. AAAI. Vol. 5.

Preliminary results



smmd One MDD constraint per solution

emdd One MDD constraint for all solutions

MDD width

With m solutions and diversity d, maximum MDD width = $(d+1)^m$

10 solutions, $d = 5 \Longrightarrow \equiv 60$ million nodes per layer !

- 1 MDD per solution = small width
- 1 MDD for all solutions = better propagation

MDD width

With m solutions and diversity d, maximum MDD width = $(d+1)^m$

10 solutions, $d = 5 \Longrightarrow \equiv 60$ million nodes per layer !

- 1 MDD per solution = small width
- 1 MDD for all solutions = better propagation

$\Longrightarrow$ Relaxation !

Relaxed MDD

Forall $\mathbf{x}$,

$$MDD_{relax}(\mathbf{x}) \leqslant MDD(\mathbf{x}) \quad (1)$$

If $\#(L_i) > w_{\max}$, we merge nodes and adjust weights to satisfy (1).

Merging strategies :

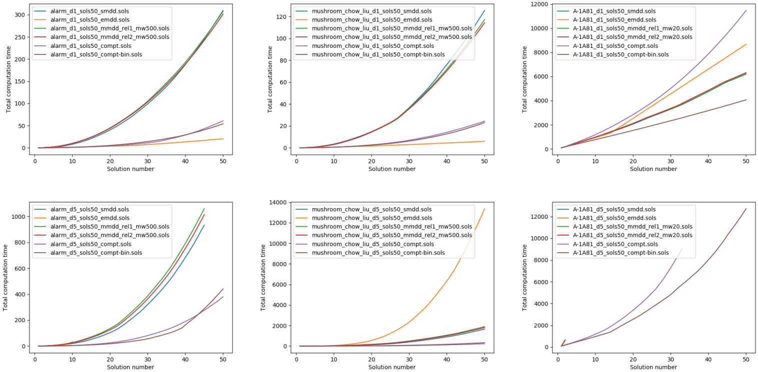
*rel*1 Random nodes are merged

*rel*2 Nodes u with smallest $div = \sum_{d^j \in I_u} d^j$ are merged

---

Bergman, D., Cire, A. A., Van Hoeve, W. J., Hooker, J. (2016). Decision diagrams for optimization. Springer International Publishing.

Preliminary results



time limit = 600 seconds

What now ?

Methods

- Lagrangian relaxation → duality gap !
- MDD constraint
  - 1 constraint per solution
  - 1 constraint for all → exponential !
  - Relaxation ?
- Regular constraint
  - REGULAR → worse than MDD
  - **Binary decomposition**

To do

- Smaller search tree with relaxed MDD ?
- Better propagations on MDD constraint
- Use dissimilarity matrix in diversity measure

What now ?

Methods

- Lagrangian relaxation → duality gap !
- MDD constraint
  - 1 constraint per solution
  - 1 constraint for all → exponential !
  - Relaxation ?
- Regular constraint
  - REGULAR → worse than MDD
  - **Binary decomposition**

To do

- Smaller search tree with relaxed MDD ?
- Better propagations on MDD constraint
- Use dissimilarity matrix in diversity measure

Thank you !