Tutorial on

Inference and Learning in Discrete Graphical Models: Theory and Practice

ICCV 2015, Santiago de Chile

Bogdan Savchynskyy*, Jörg Hendrik Kappes*, Thorsten Beier, Sebastian Nowozin, and Carsten Rother

December 12th, 2015 (full day)

About Us

Jörg Herndrik Kappes

Bogdan Savchynskyy

Heidelberg Collaboratory for Image Processing

COMPUTER VISION LAB DRESDEN

Tutorial on
Inference and Learning in Discrete Graphical Models:
Theory and Practice

Thorsten Beier

Carsten Rother

Sebastian Nowozin

Microsoft Research Cambridge

Schedule

- (08:30-08:40) **Opening**
- (08:40-09:30) **Discrete Graphical Models** (50 min)
  - Applications in Computer Vision (20 min)
  - Definitions and Notation (20 min)
  - Overview of Existing Software-packages (10 min)
- (09:30-10:00) **Inference in Discrete Graphical Models I** (150 min)
  - Exact Inference Methods (50 min)
- (10:00-10:30) **Coffee Break**
- (10:30-12:30) **Inference in Discrete Graphical Models II**
  - ... Exact Inference Methods
  - Inference Methods based on Relaxations (40 min)
  - Partial Optimality (10 min)
  - Approximative and Move Making Methods (40 min)
  - Meta-Methods : Combining Methods to get a better overall performance (10 min)
- (12:15-14:00) **Lunch**
- (14:00-15:00) **From Benchmarks to the Current Limits**
  - Insights from Benchmark Studies (20 min)
  - How to deal with Huge Models and Higher-order Potentials? (20 min)
  - Models with Discrete and Continuous Variables (20 min)
- (15:00-15:30) **Coffee Break**
- (15:30-16:20) **Learning in Discrete Graphical Models** (50 min)
  - Problem Setting
  - Maximum Likelihood based Methods
  - Prediction-based Parameter Learning Methods
- (16:20-16:30) **Closing**

Good to know . . .

Good to know ...

- Slides will be online.

Good to know . . .

- Slides will be online.
-  point You to references with further informations.

Good to know . . .

- Slides will be online.
-  point You to references with further informations.
- Formulas are only important for the tutorial if we mention them.

Good to know ...

- Slides will be online.
-  point You to references with further informations.
- Formulas are only important for the tutorial if we mention them.
- If You have questions **please** ask.

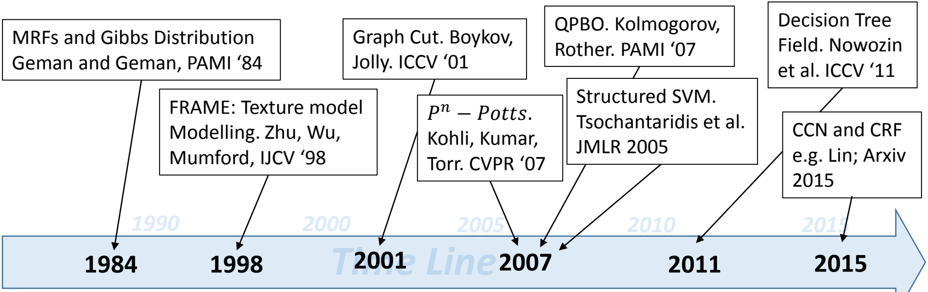
Good to know . . .

- Slides will be online.
-  point You to references with further informations.
- Formulas are only important for the tutorial if we mention them.
- If You have questions **please** ask.

- It is more important to get the concepts than the details!

Applications for Graphical Models in Computer Vision

Learning and Inference in Graphical Models



Markov Random Fields (Theory and Practice)

Local inference (ICM, Simulated Annealing, Gibbs sampler)

Global inference (Graph Cut, Message Passing, move making)

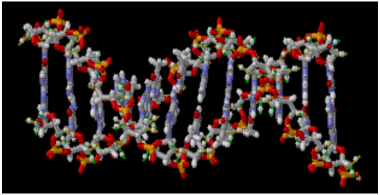
Inference in Complex models (higher- order, non-submodular, etc)
Learning (struct SVM, etc.)

Learning of Complex Models

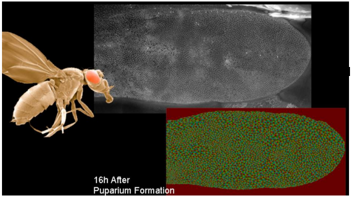
Challenging Inference & Deep Learning

<u>My personal, Computer Vision-Based View</u>

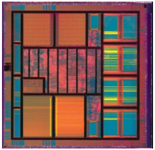
Applications of Graphical Models – outside CV



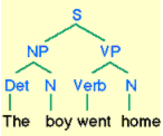
Label Chemical Structures



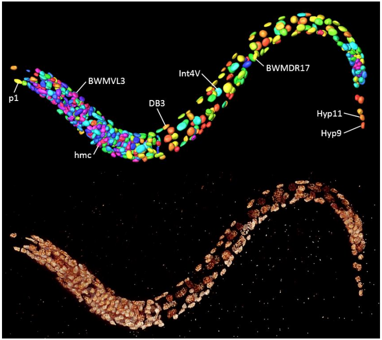
Flywing partitioning and tracking



VLSI integrated-circuit design



Speech recognition



Semantic segmentation of the worm

Applications of Graphical Models – inside CV

Semantic segmentation

Pose Estimation

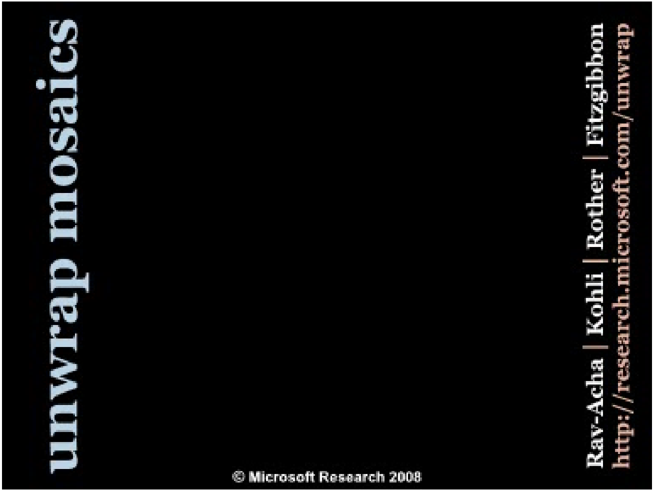
Depth Estimation

Debluring and Denoising

Motion Estimation and Tracking

Image In-paiting

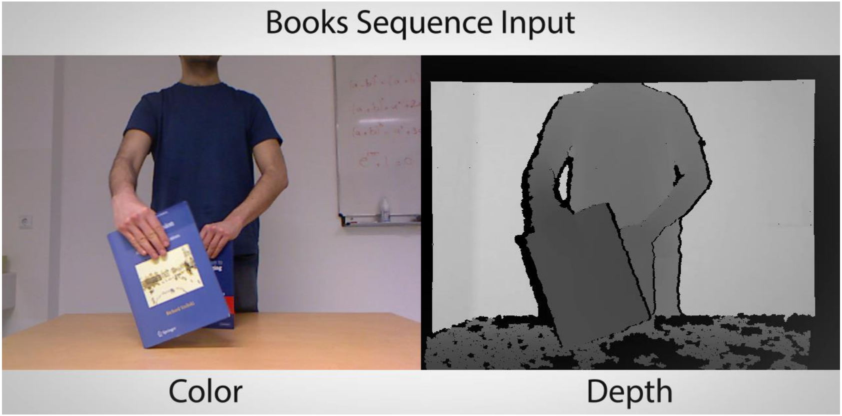
Graphical Models in Computer Vision: A Success Story



- Video Segmentation
- Dense Discrete-Continuous Optimization

Video Enhancement
[Rav-Acha et al. Siggraph 2008]

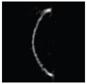
Graphical Models in Computer Vision: A Success Story



- Sparse Graph matching
- 6D Dense Continuous Motion

Scene Flow Estimation
[Abu Alhaija et. al. GCPR 2015]

Graphical Models in Computer Vision: A Success Story



Input Blur

Input

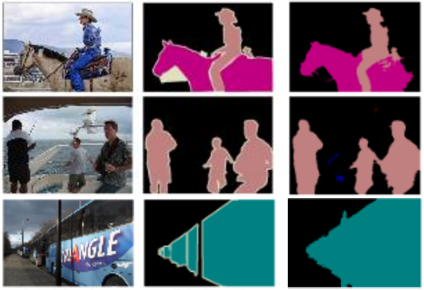
Output

- Learning Gaussian Markov Random Fields
- State-of-the art deconvolution

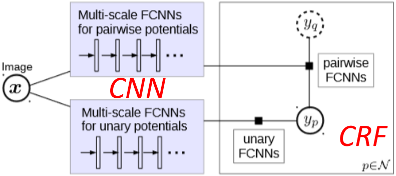
Image Deconvolution
[Schmidt et al. CVPR 2013]

Graphical Models in Computer Vision: A Success Story

Current Leader of Semantic Segmentation Challenge VOC2012



(a) Testing (b) Truth (d) Full model



[Efficient Piecewise Training of Deep Structured Models for Semantic Segmentation, Lin, Shen Reid, Hegel, Arxiv 2015]

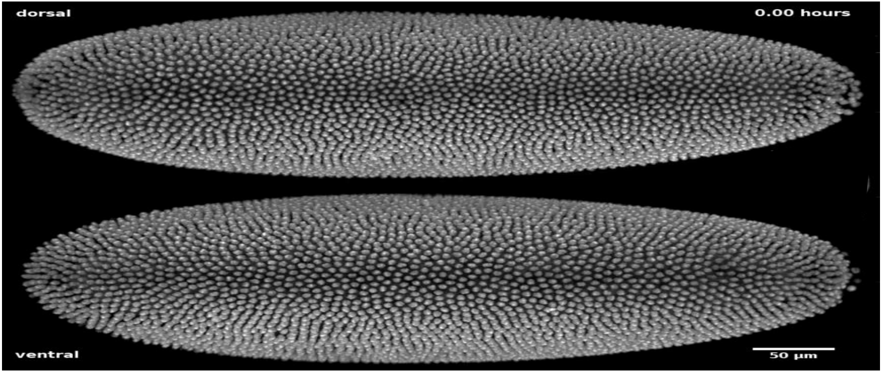
Open Challenges

1. Inference in Large and Complex Models
2. Exact inference
3. Fast Inference
4. Continuous Variables and Mixed Models
5. Deep Learning and Graphical Models

Open Challenges

1. Inference in Large and Complex Models
2. Exact inference
3. Fast Inference
4. Continuous Variables and Mixed Models
5. Deep Learning and Graphical Models

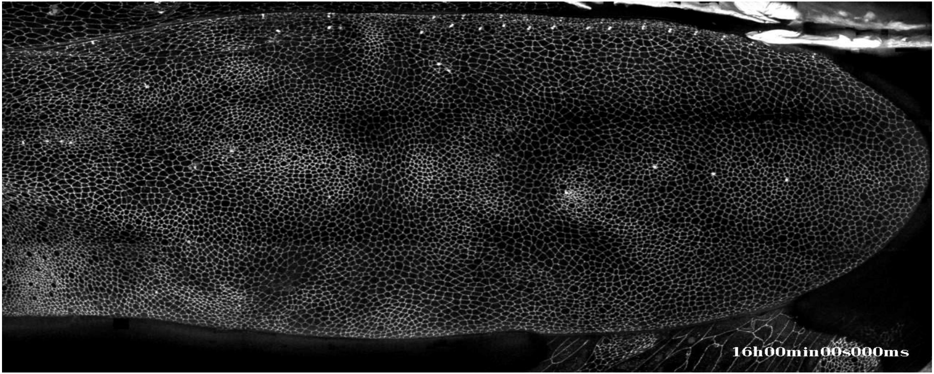
1. Large Scale Models



Drosophila Embryo, Keller Lab, Janelia, US
e.g. 2,880 x 250MB stack with 100K cells

Track each individual cell perfectly (99.9% accuracy needed!)

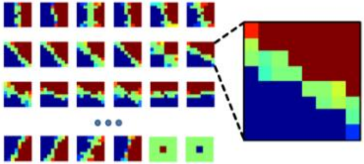
1. Large Scale Models

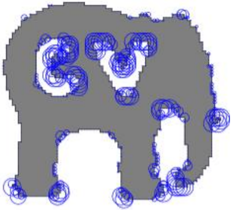


Fly Wing, Eaton Lab, MPI-CBG, Dresden

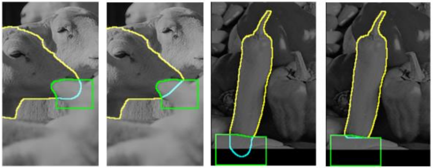
Track each individual cell perfectly (99.9 % accuracy needed!)

1. Complex Models





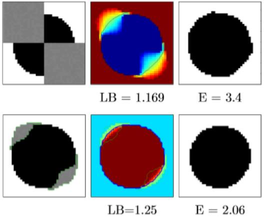
Higher-order Potentials penalizing high Curvature



Curvature Model | Pairwise MRF | Curvature Model | Pairwise MRF

- Right model … but inference too hard!

    1. TRW-S
    2. TRW-S (hard constraints)
    3. Block-ICM



Open Challenges

1. Inference in Large and Complex Models
2. Exact inference
3. Fast Inference
4. Continuous Variables and Mixed Models
5. Deep Learning and Graphical Models

2. Exact Inference

Input:
Image sequence

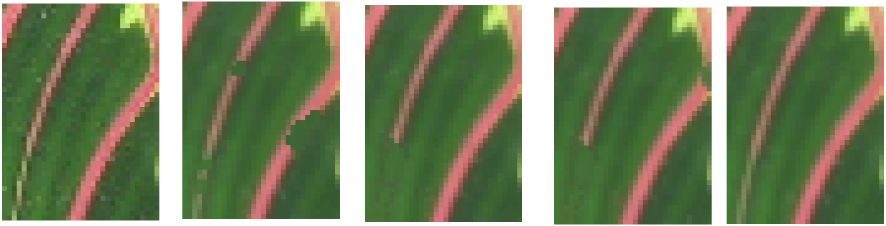
[Data courtesy from Oliver Woodford]

Output: New view

Model: Minimize a binary 4-connected pair-wise graph
(choose a colour-mode at each pixel)

[Fitzgibbon et al. ICCV '03]

2. Exact Inference



Ground Truth

Graph Cut with truncation [Rother et al. '05] **(approximate solution)**

Belief Propagation **(approximate solution)**

ICM, Simulated Annealing **(approximate solution)**

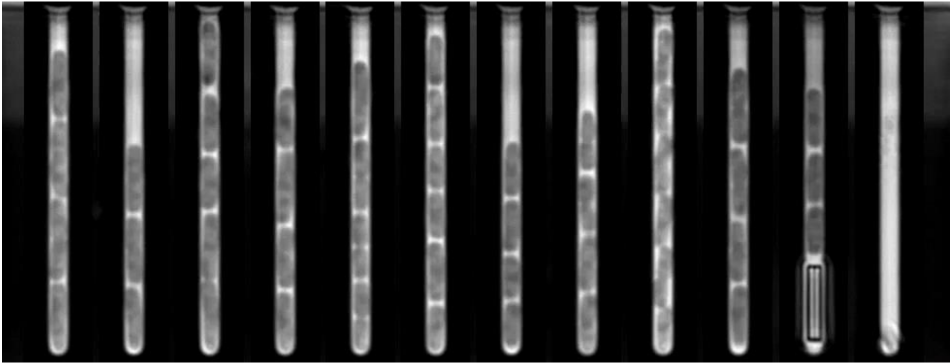
QPBOP [Boros et al. '06; Rother et al. '07] **(exact solution)**

Why is the result not perfect?
Model or Optimization

Open Challenges

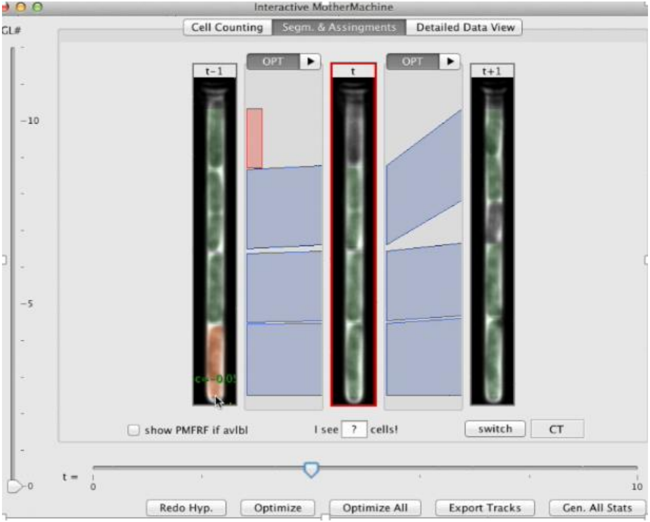
1. Inference in Large and Complex Models
2. Exact inference
3. Fast Inference
4. Continuous Variables and Mixed Models
5. Deep Learning and Graphical Models

2. Fast Inference



1D cell tracking

3. Fast Inference



- Human in-the Loop
- Deep Learning

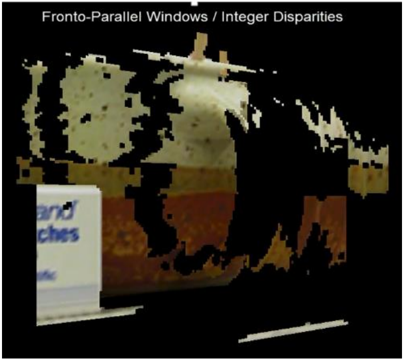
Joint Segmentation and Tracking
[Jug et al. BAMBI (MICCAI) 2014]

Open Challenges

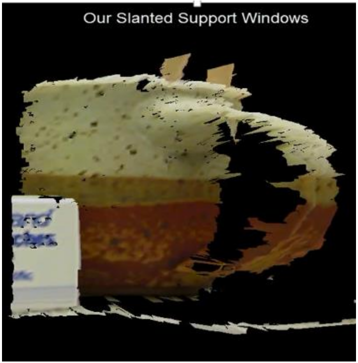
1. Inference in Large and Complex Models
2. Exact inference
3. Fast Inference
4. Continuous Variables and Mixed Models
5. Deep Learning and Graphical Models

4. Continuous Variables Models

Stereo Matching



Discrete Variables



Continuous Variables

[Bleyer, Rhemann, Rother. BMVC '2011]

Open Challenges

1. Inference in Large and Complex Models
2. Exact inference
3. Fast Inference
4. Continuous Variables and Mixed Models
5. Deep Learning and Graphical Models

5. Deep Learning and Structured Models

Input

Ground Truth | CNN Trained separately | + CRF Trained separately **(87.6%)** | CNN Trained jointly | + CRF Trained jointly **(89.0%)**

- CRF gives CNN additional regularization
- What is the optimal combination of CRFs and CNNs?

ML Learning of a generic CNN-CRF model
[Kirrilov et al. arxiv 2015]

Summary

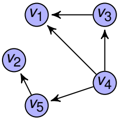
1. Graphical Models are everywhere in Vision
2. Many interesting open challenges
3. Enjoy the Tutorial!

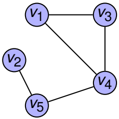
Definitions and Notation

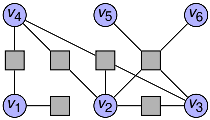
Graphical Models

Definition: Graphical Model

A **graphical model** is a model for which a *graph* denotes some *structure* between variables (represented by its nodes).



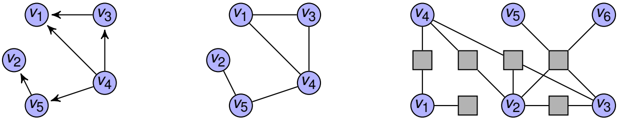




Graphical Models

Definition: Graphical Model

A **graphical model** is a model for which a *graph* denotes some *structure* between variables (represented by its nodes).



$$\textcircled{v_i} \Leftrightarrow x_i \in X_i = \begin{cases} \{0, 1, 2, \ldots, L_i\} & \textit{Discrete Variable} \\ \text{or} & \\ \Omega \subset \mathbb{R} & \textit{Continuous Variable} \end{cases}$$

Graphical Models: Probabilistic Graphical Model

Definition: Probabilistic Graphical Model

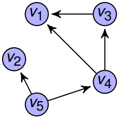
A **probabilistic graphical model** is a probabilistic model for which a *graph* $G = (V, E \subset V \times V)$ denotes the conditional independence *structure* between random variables (represented by its nodes).

Graphical Models: Probabilistic Graphical Model

Definition: Probabilistic Graphical Model

A **probabilistic graphical model** is a probabilistic model for which a *graph* $G = (V, E \subset V \times V)$ denotes the conditional independence *structure* between random variables (represented by its nodes).

Bayesian Network



$G \Rightarrow$ Markov Properties (MP) , e.g. $X_v \perp\!\!\!\perp X_{V\setminus \mathrm{de}(v)} | X_{\mathrm{pa}(v)}$

$G \Rightarrow$ Factorization , $P(X) = \prod_{v\in V} P(X_v | X_{\mathrm{pa}(v)})$

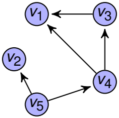
G has to be a directed acyclic graph (DAG)

Graphical Models: Probabilistic Graphical Model

Definition: Probabilistic Graphical Model

A **probabilistic graphical model** is a probabilistic model for which a *graph* $G = (V, E \subset V \times V)$ denotes the conditional independence *structure* between random variables (represented by its nodes).

Bayesian Network

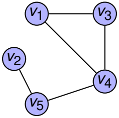


$G \Rightarrow$ Markov Properties (MP) , e.g. $X_v \perp\!\!\!\perp X_{V\setminus \mathrm{de}(v)} | X_{\mathrm{pa}(v)}$

$G \Rightarrow$ Factorization , $P(X) = \prod_{v\in V} P(X_v|X_{\mathrm{pa}(v)})$

G has to be a directed acyclic graph (DAG)

Markov Random Field (MRF)



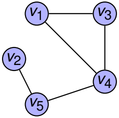
$G \Rightarrow$ Markov Properties (MP) , e.g.[1] $X_u \perp\!\!\!\perp X_v | X_{V\setminus\{u,v\}} \Leftrightarrow (uv) \notin E$

MP $\overset{*}{\Rightarrow}$ Factorization , $P(X = x) \propto \prod_{C\in \mathrm{cliques}(G)} \varphi_C(x_C)$

[1] Pairwise Markov property
$*$ P has to be a positive density or G has to be chordal

Graphical Models: MRF vs. CRF

Markov Random Field (MRF)

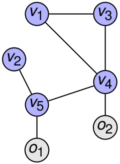


$G \Rightarrow$ Markov Properties (MP) , e.g. $X_u \perp\!\!\!\perp X_v | X_{V\setminus\{u,v\}} \Leftrightarrow (uv) \notin E$

MP $\overset{*}{\Rightarrow}$ Factorization , $P(X=x) \propto \prod_{C \in \text{cliques}(G)} \varphi_C(x_C)$

$*$ P has to be a positive density or G has to be chordal

Conditional Random Field (CRF)



$G \Rightarrow$ Markov Properties (MP) , e.g.[1] $X_u \perp\!\!\!\perp X_v | X_{V\setminus\{u,v\}} \Leftrightarrow (uv) \notin E$

MP $\overset{*}{\Rightarrow}$ Factorization , $P(X=x|D=d) \propto \prod\limits_{C \in \text{cliques}(G)} \varphi_C(x_{H(C)}, d_{O(C)})$

1 Pairwise Markov property
$*$ P has to be a positive density or G has to be chordal

Graphical Models: Factor Graph Model

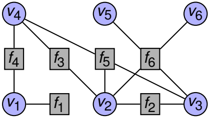
Definition: Factor Graph (Graphical) Model

A **factor graph model** is a model for which a *factor graph* $G = (V, F, E \subset V \times F)$ denotes the *factorization* of the objective function over its variables (represented by its nodes).

Graphical Models: Factor Graph Model

Definition: Factor Graph (Graphical) Model

A **factor graph model** is a model for which a *factor graph* $G = (V, F, E \subset V \times F)$ denotes the *factorization* of the objective function over its variables (represented by its nodes).



$G = (V, F, E)$

$G \Rightarrow$ Factorization: $\quad func(x) = \bigotimes_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

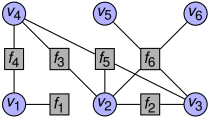
e.g. $\quad P(X = x) = \prod_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

$\quad J(x) = \sum_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

Graphical Models: Factor Graph Model

Definition: Factor Graph (Graphical) Model

A **factor graph model** is a model for which a *factor graph* $G = (V, F, E \subset V \times F)$ denotes the *factorization* of the objective function over its variables (represented by its nodes).



$G = (V, F, E)$

$G \Rightarrow$ Factorization: $\quad func(x) = \bigotimes_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

e.g. $\quad P(X = x) = \prod_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

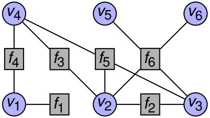
$\quad J(x) = \sum_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

$$func(x) = \varphi_{f_1}(x_{ne(f_1)}) \otimes \varphi_{f_2}(x_{ne(f_2)}) \otimes \varphi_{f_3}(x_{ne(f_3)}) \otimes \varphi_{f_4}(x_{ne(f_4)}) \otimes \varphi_{f_5}(x_{ne(f_5)}) \quad \otimes \varphi_{f_6}(x_{ne(f_6)})$$

Graphical Models: Factor Graph Model

Definition: Factor Graph (Graphical) Model

A **factor graph model** is a model for which a *factor graph* $G = (V, F, E \subset V \times F)$ denotes the *factorization* of the objective function over its variables (represented by its nodes).



$G = (V, F, E)$

$G \Rightarrow$ Factorization: $\quad func(x) = \bigotimes_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

e.g. $\quad P(X = x) = \prod_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

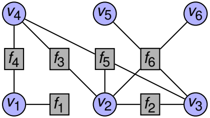
$\quad J(x) = \sum_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

$$
\begin{aligned}
func(x) &= \varphi_{f_1}(x_{ne(f_1)}) \otimes \varphi_{f_2}(x_{ne(f_2)}) \otimes \varphi_{f_3}(x_{ne(f_3)}) \otimes \varphi_{f_4}(x_{ne(f_4)}) \otimes \varphi_{f_5}(x_{ne(f_5)}) \otimes \varphi_{f_6}(x_{ne(f_6)}) \\
&= \varphi_{f_1}(x_1) \otimes \varphi_{f_2}(x_2, x_3) \otimes \varphi_{f_3}(x_2, x_4) \otimes \varphi_{f_4}(x_1, x_4) \otimes \varphi_{f_5}(x_2, x_3, x_4) \otimes \varphi_{f_6}(x_2, x_3, x_5, x_6)
\end{aligned}
$$

Graphical Models: Factor Graph Model

Definition: Factor Graph (Graphical) Model

A **factor graph model** is a model for which a *factor graph* $G = (V, F, E \subset V \times F)$ denotes the *factorization* of the objective function over its variables (represented by its nodes).



$G = (V, F, E)$

$G \Rightarrow$ Factorization: $\quad func(x) = \bigotimes_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

e.g. $\quad P(X = x) = \prod_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

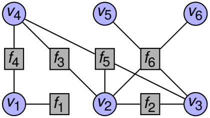
$\quad J(x) = \sum_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

$$
\begin{aligned}
func(x) &= \varphi_{f_1}(x_{ne(f_1)}) \otimes \varphi_{f_2}(x_{ne(f_2)}) \otimes \varphi_{f_3}(x_{ne(f_3)}) \otimes \varphi_{f_4}(x_{ne(f_4)}) \otimes \varphi_{f_5}(x_{ne(f_5)}) \quad \otimes \varphi_{f_6}(x_{ne(f_6)}) \\
&= \varphi_{f_1}(x_1) \quad \otimes \varphi_{f_2}(x_2, x_3) \otimes \varphi_{f_3}(x_2, x_4) \otimes \varphi_{f_4}(x_1, x_4) \otimes \varphi_{f_5}(x_2, x_3, x_4) \otimes \varphi_{f_6}(x_2, x_3, x_5, x_6) \\
J(x) &= \varphi_{f_1}(x_1) \quad + \varphi_{f_2}(x_2, x_3) + \varphi_{f_3}(x_2, x_4) + \varphi_{f_4}(x_1, x_4) + \varphi_{f_5}(x_2, x_3, x_4) \ \varphi_{f_6}(x_2, x_3, x_5, x_6)
\end{aligned}
$$

Graphical Models: Factor Graph Model

Definition: Factor Graph (Graphical) Model

A **factor graph model** is a model for which a *factor graph* $G = (V, F, E \subset V \times F)$ denotes the *factorization* of the objective function over its variables (represented by its nodes).



$G = (V, F, E)$

$G \Rightarrow$ Factorization: $\quad func(x) = \bigotimes_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

e.g. $\quad P(X = x) = \prod_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

$\quad J(x) = \sum_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$

Definition: Order

The **order** of a factor is its degree, i.e.

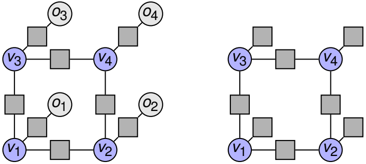
$$o(f) = |\{v \in V | (v, f) \in E\}|$$

The order of a factor graph is the maximal order of its factors, i.e.

$$o(G) = \max_{f \in F} o(f)$$

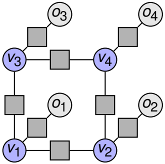
Graphical Models

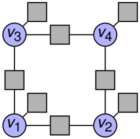
Conditional Factor Graph Model



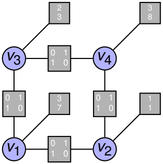
Graphical Models

Conditional Factor Graph Model



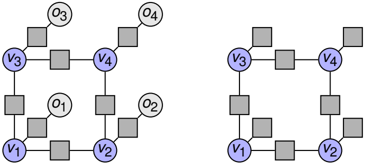


Discrete Factor Graph Model

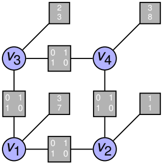


Graphical Models

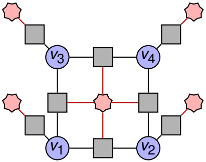
Conditional Factor Graph Model



Discrete Factor Graph Model



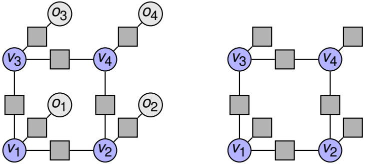
Extended Factor Graph Model



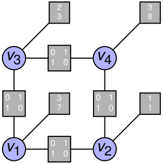
$$G = (V, F, E, \mathcal{F}, \mathcal{E})$$

Graphical Models

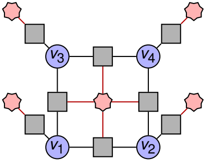
Conditional Factor Graph Model



Discrete Factor Graph Model

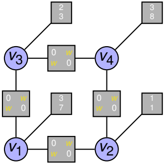


Extended Factor Graph Model



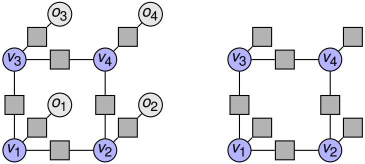
$$G = (V, F, E, \mathcal{F}, \mathcal{E})$$

Weighted Factor Graph Model

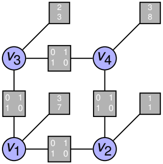


Graphical Models

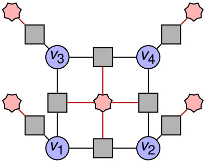
Conditional Factor Graph Model



Discrete Factor Graph Model

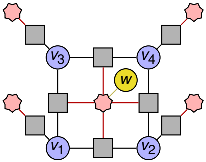


Extended Factor Graph Model



$$G = (V, F, E, \mathcal{F}, \mathcal{E})$$

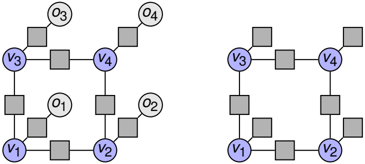
Weighted Factor Graph Model



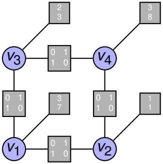
$$G = (V, F, E, \mathcal{F}, \mathcal{E}, W, E_W)$$

Graphical Models

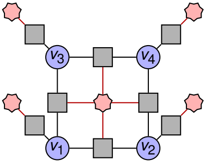
Conditional Factor Graph Model



Discrete Factor Graph Model

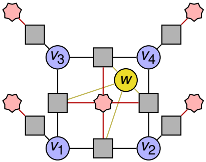


Extended Factor Graph Model



$$G = (V, F, E, \mathcal{F}, \mathcal{E})$$

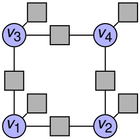
Weighted Factor Graph Model



$$G = (V, F, E, \mathcal{F}, \mathcal{E}, W, E_W)$$

Graphical Models: Shorthands for Factor Graphs

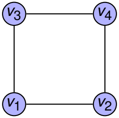
Factor Graph Models



$$G = (V, F, E)$$

$$J(x) = \sum_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$$

Common used Shorthand for 2nd order Factor Graph Models



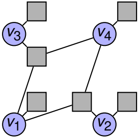
$$G = (V, E \subset V \times V)$$

$$J(x) = \sum_{v \in V} \varphi_v(x_v) + \sum_{e \in E} \varphi_e(x_e)$$

Important: This is not a MRF, because G decodes directly the factorization!

Graphical Models: Shorthands for Factor Graphs

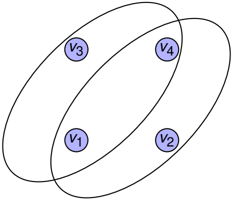
Factor Graph Models



$$G = (V, F, E)$$

$$J(x) = \sum_{f \in F} \varphi_f(x_{\mathrm{ne}(f)})$$

Common used Shorthand for general Factor Graph Models

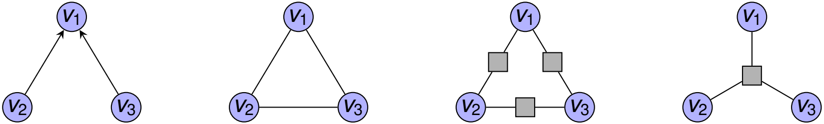


$$G = (V, E \subset 2^V)$$

$$J(x) = \sum_{v \in V} \varphi_v(x_v) + \sum_{C \in E} \varphi_C(x_C)$$

Important: This is not a MRF, because G decodes directly the factorization!

Graphical Models: Factor-Graph Model vs. Markov Random Fields



Factor graph models are more powerful than MRFs/CRFs and Bayesian networks in expressing factorization.

Bayesian networks are more powerful than Factor graph models and MRFs/CRFs in expressing conditional independences.

The terms MRFs/CRFs are often loosely used in computer vision!
Further information: *[Lauritzen, 1996, Koller and Friedman, 2009]*

Graphical Models: Gibbs Distribution

- Given an energy function

$$J(x) = \sum_{C \in \mathcal{C} \subset 2^V} \varphi_C(x_C)$$

Graphical Models: Gibbs Distribution

- Given an energy function

$$J(x) = \sum_{C \in \mathcal{C} \subset 2^V} \varphi_C(x_C)$$

we can define the *Gibbs distribution* for a given *free parameter* β by

$$P(X = x) = p(x) = \frac{1}{Z} \exp(-\beta \cdot J(x)) = \frac{1}{Z} \prod_{C \in \mathcal{C} \subset 2^V} \exp(-\beta \cdot \varphi_C(x_C))$$

Graphical Models: Gibbs Distribution

- Given an energy function

$$J(x) = \sum_{C \in \mathcal{C} \subset 2^V} \varphi_C(x_C)$$

we can define the *Gibbs distribution* for a given *free parameter* β by

$$P(X = x) = p(x) = \frac{1}{Z} \exp(-\beta \cdot J(x)) = \frac{1}{Z} \prod_{C \in \mathcal{C} \subset 2^V} \exp(-\beta \cdot \varphi_C(x_C))$$

where the *partition function* is given by

$$Z = \sum_{x \in \mathcal{X}} \exp(-\beta \cdot J(x)).$$

Graphical Models: Gibbs Distribution

- Given an energy function

$$J(x) = \sum_{C \in \mathcal{C} \subset 2^V} \varphi_C(x_C)$$

we can define the *Gibbs distribution* for a given *free parameter* β by

$$P(X = x) = p(x) = \frac{1}{Z} \exp(-\beta \cdot J(x)) = \frac{1}{Z} \prod_{C \in \mathcal{C} \subset 2^V} \exp(-\beta \cdot \varphi_C(x_C))$$

where the *partition function* is given by

$$Z = \sum_{x \in \mathcal{X}} \exp(-\beta \cdot J(x)).$$

And what is the statistical meaning of $p(x)$?

Graphical Models: Gibbs Distribution

- Given a distribution

$$p(x) \text{ with Markov properties given by } G = (V, E)$$

Graphical Models: Gibbs Distribution

- Given a distribution

$$p(x) \text{ with Markov properties given by } G = (V, E)$$

- If $p(\cdot)$ is **strict positive** ($p(x) > 0$) **or** G is **chordal** it factorize into

$$p(x) = \prod_{C \in \mathcal{C} \subset 2^V} \phi_C(x_C)$$

where $\mathcal{C}$ the maximal cliques in G.

$\rightarrow$ Hammersley-Clifford theorem *[Hammersley and Clifford, 1971]*.

Graphical Models: Gibbs Distribution

- Given a distribution

$$p(x) \text{ with Markov properties given by } G = (V, E)$$

- If $p(\cdot)$ is **strict positive** ($p(x) > 0$) **or** G is **chordal** it factorize into

$$p(x) = \prod_{C \in \mathcal{C} \subset 2^V} \phi_C(x_C)$$

where $\mathcal{C}$ the maximal cliques in G.
$\rightarrow$ Hammersley-Clifford theorem *[Hammersley and Clifford, 1971]*.

- We can define a energy function which has $p(x)$ as Gibbs distribution

$$J(x) = -\frac{1}{\beta}\log(p(x)) = \sum_{C \in \mathcal{C} \subset 2^V} -\frac{1}{\beta}\log(\phi_C(x_C)).$$

Graphical Models: Inference

Graphical Models: Inference

Probabilistic Inference

$$p(x_i) = \sum_{x' \in \mathcal{X}, x'_i = x_i} p(x') \qquad \text{Marginals}$$

$$p_{\max}(x_i) = \max_{x' \in \mathcal{X}, x'_i = x_i} p(x') \qquad \text{Max-Marginals}$$

$$\tilde{x} \sim p(x) \qquad \text{Sampling}$$

$$Z = \sum_{x' \in \mathcal{X}} \exp(-J(x')) \qquad \text{Partition Function}$$

MAP Inference and Energy Minimization

$$x^* \in \operatorname*{arg\,max}_{x \in \mathcal{X}} p(x) \qquad \text{Most Likely Explanation}$$

$$x^* \in \operatorname*{arg\,min}_{x \in \mathcal{X}} J(x) \qquad \text{Configuration with the Lowest Energy}$$

Graphical Models: Inference

Using MAP-inference to approximate marginals

- Perturb and MAP
  - Expected MAP solution of perturbed model $\Rightarrow$ marginals
  - *[Papandreou and Yuille, 2011, Hazan et al., 2013]*
- Frank-Wolf Algorithm
  - Perturbed MAP-problems show up in Frank-Wolf algorithm for calculating marginals
  - *[R. Krishnan, 2015]*

Graphical Models: Inference

MAP Inference and Energy Minimization

$$x^* \in \underset{x \in \mathcal{X}}{\arg\max}\, p(x|d)$$ Most Likely Explanation

$$x^* \in \underset{x \in \mathcal{X}}{\arg\min}\, J(x|d)$$ Configuration with the Lowest Energy

data (d)

modeling/learning $\longrightarrow$

$J(x|d)$
or
$p(x|d)$

model

MAP inference $\longrightarrow$

labeling (x^*)

Graphical Models: Inference

MAP Inference and Energy Minimization

$$x^* \in \arg\max_{x \in \mathcal{X}} p(x|d) \qquad \text{Most Likely Explanation}$$

$$x^* \in \arg\min_{x \in \mathcal{X}} J(x|d) \qquad \text{Configuration with the Lowest Energy}$$

data (d)

modeling/learning $\longrightarrow$

$J(x|d)$ or $p(x|d)$

model

MAP inference $\longrightarrow$

labeling (x^*)

Graphical Models: Inference

MAP Inference and Energy Minimization

$$x^* \in \arg\max_{x \in \mathcal{X}} p(x|d) \qquad \text{Most Likely Explanation}$$

$$x^* \in \arg\min_{x \in \mathcal{X}} J(x|d) \qquad \text{Configuration with the Lowest Energy}$$

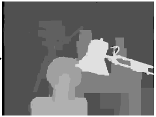
data (d)

modeling/learning →

$J(x|d)$
or
$p(x|d)$

model

MAP inference →



labeling (x^*)

MAP Inference and Energy Minimization

$$x^* \in \arg\max_{x \in \mathcal{X}} p(x|d) \qquad \text{Most Likely Explanation}$$

$$x^* \in \arg\min_{x \in \mathcal{X}} J(x|d) \qquad \text{Configuration with the Lowest Energy}$$

data (d)

modeling/learning $\longrightarrow$

$J(x|d)$
or
$p(x|d)$

model

MAP inference $\longrightarrow$

labeling (x^*)

Graphical Models: Inference

MAP Inference and Energy Minimization

$$x^* \in \arg\max_{x \in \mathcal{X}} p(x|d) \qquad \text{Most Likely Explanation}$$

$$x^* \in \arg\min_{x \in \mathcal{X}} J(x|d) \qquad \text{Configuration with the Lowest Energy}$$

data (d)

modeling/learning $\longrightarrow$

$J(x|d)$ or $p(x|d)$

model

MAP inference $\longrightarrow$

labeling (x^*)

MAP Inference and Energy Minimization

$$x^* \in \arg\max_{x \in \mathcal{X}} p(x|d) \qquad \text{Most Likely Explanation}$$

$$x^* \in \arg\min_{x \in \mathcal{X}} J(x|d) \qquad \text{Configuration with the Lowest Energy}$$

$\xrightarrow{\text{modeling/learning}}$

$J(x|d)$
or
$p(x|d)$

$\xrightarrow{\text{MAP inference}}$

data (d) | model | labeling (x^*)

Graphical Models: Learning

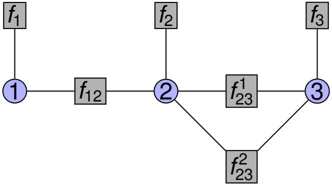
Graphical Models: Learning

Graphical Models: Learning

1. Select (Learn) the (number of) variables and labels and their meaning

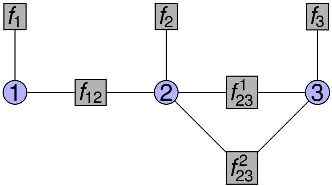
Graphical Models: Learning

1. Select (Learn) the (number of) variables and labels and their meaning
2. Select (Learn) the structure of the model



Graphical Models: Learning

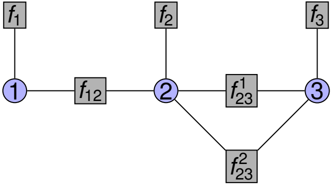
1. Select (Learn) the (number of) variables and labels and their meaning
2. Select (Learn) the structure of the model
3. Learn independently **local** potentials/feature-functions
   - physical priors
   - handcrafted features
   - output of random forest or any local model from pattern recognition textbook
   - output of CNNs



$$J(x) = \sum_{i=1}^{3} \varphi_{f_i}(x_i) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f^1_{23}}(x_2, x_3) + \varphi_{f^2_{23}}(x_2, x_3)$$

Graphical Models: Learning

1. Select (Learn) the (number of) variables and labels and their meaning
2. Select (Learn) the structure of the model
3. Learn independently **local** potentials/feature-functions
   - physical priors
   - handcrafted features
   - output of random forest or any local model from pattern recognition textbook
   - output of CNNs
4. Learn **global** parameters that adjust local terms **(optional)**

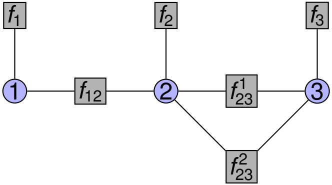


$$J(x) = \sum_{i=1}^{3} \varphi_{f_i}(x_i) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f^1_{23}}(x_2, x_3) + \varphi_{f^2_{23}}(x_2, x_3)$$

$$J(x) = \sum_{i=1}^{3} w_{f_i} \bar{\varphi}_{f_i}(x_i) + w_{f_{12}} \bar{\varphi}_{f_{12}}(x_1, x_2) + w_{f^1_{12}} \bar{\varphi}_{f^1_{23}}(x_2, x_3) + w_{f^2_{12}} \bar{\varphi}_{f^2_{23}}(x_2, x_3)$$

Graphical Models: Learning

1. Select (Learn) the (number of) variables and labels and their meaning
2. Select (Learn) the structure of the model
3. Learn independently **local** potentials/feature-functions
   - physical priors
   - handcrafted features
   - output of random forest or any local model from pattern recognition textbook
   - output of CNNs
4. Learn **global** parameters that adjust local terms **(optional)**



$$J(x) = \sum_{i=1}^{3} \varphi_{f_i}(x_i) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_{23}^1}(x_2, x_3) + \varphi_{f_{23}^2}(x_2, x_3)$$

$$J(x) = \sum_{i=1}^{3} w_{f_i} \bar{\varphi}_{f_i}(x_i) + w_{f_{12}} \bar{\varphi}_{f_{12}}(x_1, x_2) + w_{f_{12}^1} \bar{\varphi}_{f_{23}^1}(x_2, x_3) + w_{f_{12}^2} \bar{\varphi}_{f_{23}^2}(x_2, x_3)$$

Graphical Models: Learning

Problem Setting

Given some training-data $(d^1, \ldots, d^N)$ and ground truth configuration $(x^{GT;1}, \ldots, x^{GT;N})$ we would like to learn the optimal model parameters.

Graphical Models: Learning

Problem Setting

Given some training-data $(d^1, \dots, d^N)$ and ground truth configuration $(x^{GT;1}, \dots, x^{GT;N})$ we would like to learn the optimal model parameters.

Maximum Likelihood Learning

$$w^* \in \arg\max_w \prod_i p(x^{GT;i}|w, d^i))$$

Maximum Margin Learning

$$w^* \in \arg\min_w \sum_i \Delta(x^{GT;i}, \arg\max_{x \in X} p(x|w, d^i)) + \lambda \|w\|_2^2$$

Software Packages

Software Packages

Library	Authors	Language	Last Updated	License	Model	Inference	Learning
OpenGM2	Andres, Beier, Kappes	C++, MatLab, Python	2015	MIT	DFGM	(P), E	(ML, MM)
BNT	Murphy	MatLab	2007	GPL 2	BN	P	ML
PMTK	Dunham,Murphy	MatLab, (Python)	2011	MIT	DFGM,MRF	P, (E)	ML
UGM	Schmidt	MatLab	2015 (2011)	BSD-2	DFGM	P, E	ML
Darwin	Gould	c++	2015	BSD-2	DFGM	P, E	-
FastInf	Jaimovich, Meshi, et al.	c++	2011	GPL 3	DFGM	P, (E)	ML
Infer.NET	Bronskill, Guiver, et al.	C++,C#	2014	MSR-LA	BN	P, (E)	ML
libDAI	Mooij	C++, Python, Matlab	2015 (2010)	BSD 2	DFGM	P,(E)	-
JGMT	Domke	Matlab	2014	MIT	DFGM^{P}	P	ML
grante	Nowozin	C++,Matlab wrappers		MSR-LA	DFGM	P, (E)	ML, MM
Factorie	UMass	Scala (Java)		Apache	FGM	P	ML
pystruct	Müller	Python	2015	BSD	DFGM	(E)	MM
mrf-lib	Szeliski et al.	C++	2012	-	DFGM^{PS}	E	-
gco-lib	Veksler, Delong	C++, MatLab, Python	2014	research only	DFGM^{P}	E	-
mplp	Globerson, Sontag, Choe, Li	C++	2014	GPL	DFGM	E	-
SRMP (TRWS)	Kolmogorov	C++	2014	GPL	DFGM	E	-
DAOOPT	Otten	C++	2012	GPL	DFGM	E	-
...							

Models: discrete factor graph models **(DFG)**, Bayesian Nets **(BN)**, ^{P} only pairwise, ^{S} restiction on the graph structure
Models: probabilistic inference **(P)**, energy minimization **(E)**
Learning: maximum likelihood based **(ML)**, max margin based **(MM)**

Software Packages

Library	Authors	Language	Last Updated	License	Model	Inference	Learning
OpenGM2	Andres, Beier, Kappes	C++, MatLab, Python	2015	MIT	DFGM	(P), E	(ML, MM)
BNT	Murphy	MatLab	2007	GPL 2	BN	P	ML
PMTK	Dunham,Murphy	MatLab, (Python)	2011	MIT	DFGM,MRF	P, (E)	ML
UGM	Schmidt	MatLab	2015 (2011)	BSD-2	DFGM	P, E	ML
Darwin	Gould	c++	2015	BSD-2	DFGM	P, E	-
FastInf	Jaimovich, Meshi, et al.	c++	2011	GPL 3	DFGM	P, (E)	ML
Infer.NET	Bronskill, Guiver, et al.	C++,C#	2014	MSR-LA	BN	P, (E)	ML
libDAI	Mooij	C++, Python, Matlab	2015 (2010)	BSD 2	DFGM	P,(E)	-
JGMT	Domke	Matlab	2014	MIT	DFGM^{P}	P	ML
grante	Nowozin	C++,Matlab wrappers		MSR-LA	DFGM	P, (E)	ML, MM
Factorie	UMass	Scala (Java)		Apache	FGM	P	ML
pystruct	Müller	Python	2015	BSD	DFGM	(E)	MM
mrf-lib	Szeliski et al.	C++	2012	-	DFGM^{PS}	E	-
gco-lib	Veksler, Delong	C++, MatLab, Python	2014	research only	DFGM^{P}	E	-
mplp	Globerson, Sontag, Choe, Li	C++	2014	GPL	DFGM	E	-
SRMP (TRWS)	Kolmogorov	C++	2014	GPL	DFGM	E	-
DAOOPT	Otten	C++	2012	GPL	DFGM	E	-
...							

Models: discrete factor graph models **(DFG)**, Bayesian Nets **(BN)**, ^{P} only pairwise, ^{S} restiction on the graph structure
Models: probabilistic inference **(P)**, energy minimization **(E)**
Learning: maximum likelihood based **(ML)**, max margin based **(MM)**

Which Library is the Best?

Software Packages

Library	Authors	Language	Last Updated	License	Model	Inference	Learning
OpenGM2	Andres, Beier, Kappes	C++, MatLab, Python	2015	MIT	DFGM	(P), E	(ML, MM)
BNT	Murphy	MatLab	2007	GPL 2	BN	P	ML
PMTK	Dunham,Murphy	MatLab, (Python)	2011	MIT	DFGM,MRF	P, (E)	ML
UGM	Schmidt	MatLab	2015 (2011)	BSD-2	DFGM	P, E	ML
Darwin	Gould	c++	2015	BSD-2	DFGM	P, E	-
FastInf	Jaimovich, Meshi, et al.	c++	2011	GPL 3	DFGM	P, (E)	ML
Infer.NET	Bronskill, Guiver, et al.	C++,C#	2014	MSR-LA	BN	P, (E)	ML
libDAI	Mooij	C++, Python, Matlab	2015 (2010)	BSD 2	DFGM	P,(E)	-
JGMT	Domke	Matlab	2014	MIT	DFGM^{P}	P	ML
grante	Nowozin	C++,Matlab wrappers		MSR-LA	DFGM	P, (E)	ML, MM
Factorie	UMass	Scala (Java)		Apache	FGM	P	ML
pystruct	Müller	Python	2015	BSD	DFGM	(E)	MM
mrf-lib	Szeliski et al.	C++	2012	-	DFGM^{PS}	E	-
gco-lib	Veksler, Delong	C++, MatLab, Python	2014	research only	DFGM^{P}	E	-
mplp	Globerson, Sontag, Choe, Li	C++	2014	GPL	DFGM	E	-
SRMP (TRWS)	Kolmogorov	C++	2014	GPL	DFGM	E	-
DAOOPT	Otten	C++	2012	GPL	DFGM	E	-
...							

Models: discrete factor graph models **(DFG)**, Bayesian Nets **(BN)**, ^{P} only pairwise, ^{S} restiction on the graph structure
Models: probabilistic inference **(P)**, energy minimization **(E)**
Learning: maximum likelihood based **(ML)**, max margin based **(MM)**

Which Library is the Best? Depends on Your Goal!

Software Packages

Library	Authors	Language	Last Updated	License	Model	Inference	Learning
OpenGM2	Andres, Beier, Kappes	C++, MatLab, Python	2015	MIT	DFGM	(P), E	(ML, MM)
BNT	Murphy	MatLab	2007	GPL 2	BN	P	ML
PMTK	Dunham,Murphy	MatLab, (Python)	2011	MIT	DFGM,MRF	P, (E)	ML
UGM	Schmidt	MatLab	2015 (2011)	BSD-2	DFGM	P, E	ML
Darwin	Gould	c++	2015	BSD-2	DFGM	P, E	-
FastInf	Jaimovich, Meshi, et al.	c++	2011	GPL 3	DFGM	P, (E)	ML
Infer.NET	Bronskill, Guiver, et al.	C++,C#	2014	MSR-LA	BN	P, (E)	ML
libDAI	Mooij	C++, Python, Matlab	2015 (2010)	BSD 2	DFGM	P,(E)	-
JGMT	Domke	Matlab	2014	MIT	DFGM^P	P	ML
grante	Nowozin	C++,Matlab wrappers		MSR-LA	DFGM	P, (E)	ML, MM
Factorie	UMass	Scala (Java)		Apache	FGM	P	ML
pystruct	Müller	Python	2015	BSD	DFGM	(E)	MM
mrf-lib	Szeliski et al.	C++	2012	-	DFGM^{PS}	E	-
gco-lib	Veksler, Delong	C++, MatLab, Python	2014	research only	DFGM^P	E	-
mplp	Globerson, Sontag, Choe, Li	C++	2014	GPL	DFGM	E	-
SRMP (TRWS)	Kolmogorov	C++	2014	GPL	DFGM	E	-
DAOOPT	Otten	C++	2012	GPL	DFGM	E	-
...							

Models: discrete factor graph models **(DFG)**, Bayesian Nets **(BN)**, ^{P} only pairwise, ^{S} restiction on the graph structure
Models: probabilistic inference **(P)**, energy minimization **(E)**
Learning: maximum likelihood based **(ML)**, max margin based **(MM)**

Which Library is the Best? Depends on Your Goal!

functionality

efficiency (time and memory)

generality / flexibility

usability

Software Packages

Library	Authors	Language	Last Updated	License	Model	Inference	Learning
OpenGM2	Andres, Beier, Kappes	C++, MatLab, Python	2015	MIT	DFGM	(P), E	(ML, MM)
BNT	Murphy	MatLab	2007	GPL 2	BN	P	ML
PMTK	Dunham,Murphy	MatLab, (Python)	2011	MIT	DFGM,MRF	P, (E)	ML
UGM	Schmidt	MatLab	2015 (2011)	BSD-2	DFGM	P, E	ML
Darwin	Gould	c++	2015	BSD-2	DFGM	P, E	-
FastInf	Jaimovich, Meshi, et al.	c++	2011	GPL 3	DFGM	P, (E)	ML
Infer.NET	Bronskill, Guiver, et al.	C++,C#	2014	MSR-LA	BN	P, (E)	ML
libDAI	Mooij	C++, Python, Matlab	2015 (2010)	BSD 2	DFGM	P,(E)	-
JGMT	Domke	Matlab	2014	MIT	DFGM^{P}	P	ML
grante	Nowozin	C++,Matlab wrappers		MSR-LA	DFGM	P, (E)	ML, MM
Factorie	UMass	Scala (Java)		Apache	FGM	P	ML
pystruct	Müller	Python	2015	BSD	DFGM	(E)	MM
mrf-lib	Szeliski et al.	C++	2012	-	DFGM^{PS}	E	-
gco-lib	Veksler, Delong	C++, MatLab, Python	2014	research only	DFGM^{P}	E	-
mplp	Globerson, Sontag, Choe, Li	C++	2014	GPL	DFGM	E	-
SRMP (TRWS)	Kolmogorov	C++	2014	GPL	DFGM	E	-
DAOOPT	Otten	C++	2012	GPL	DFGM	E	-
...							

Models: discrete factor graph models **(DFG)**, Bayesian Nets **(BN)**, ^{P} only pairwise, ^{S} restiction on the graph structure
Models: probabilistic inference **(P)**, energy minimization **(E)**
Learning: maximum likelihood based **(ML)**, max margin based **(MM)**

Which Library is the Best? Depends on Your Goal!

functionality

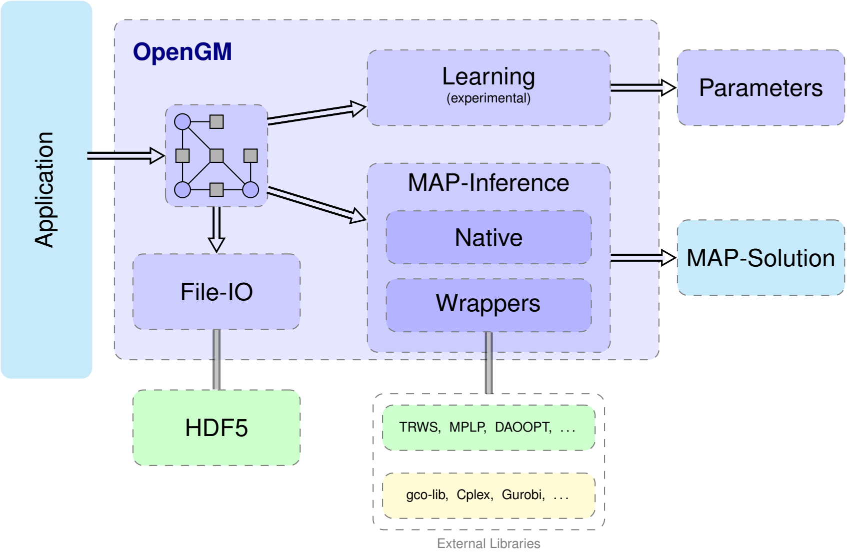
generality / flexibility

CONFLICTING!

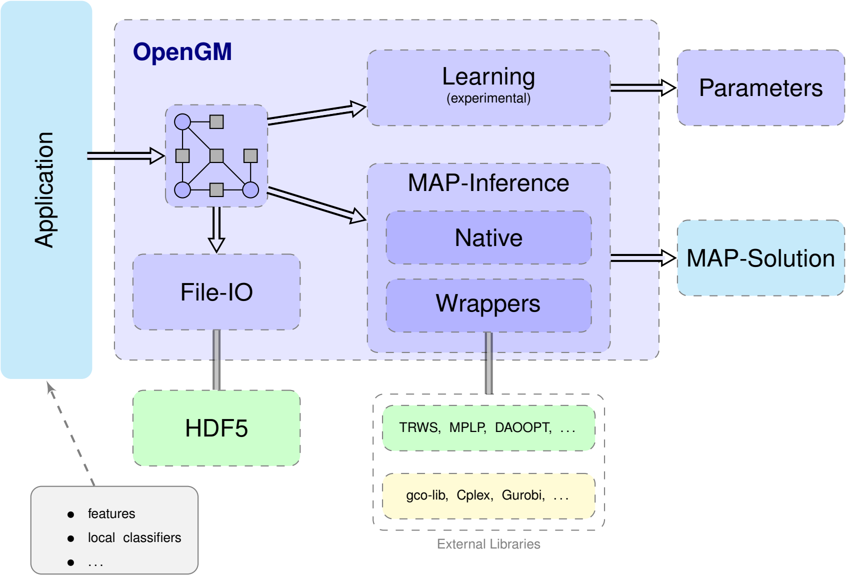
efficiency (time and memory)

usability

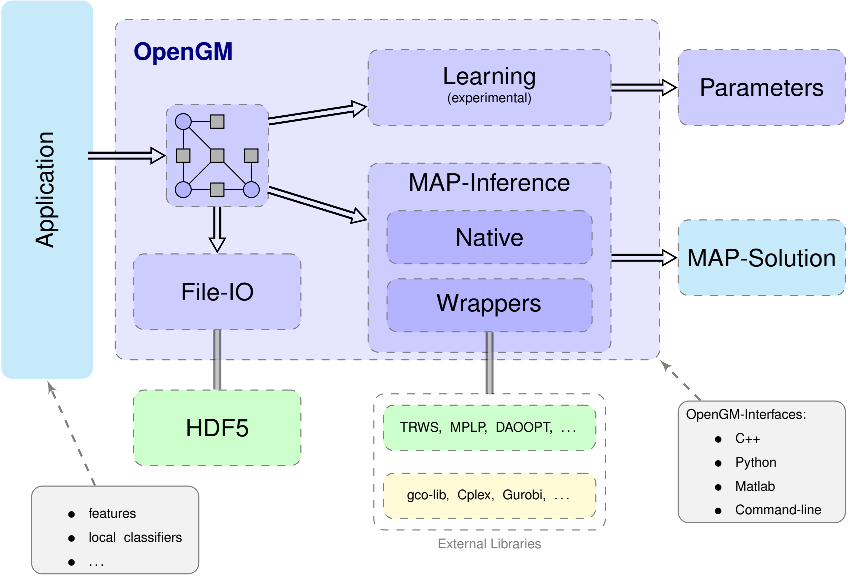
How to use OpenGM



How to use OpenGM

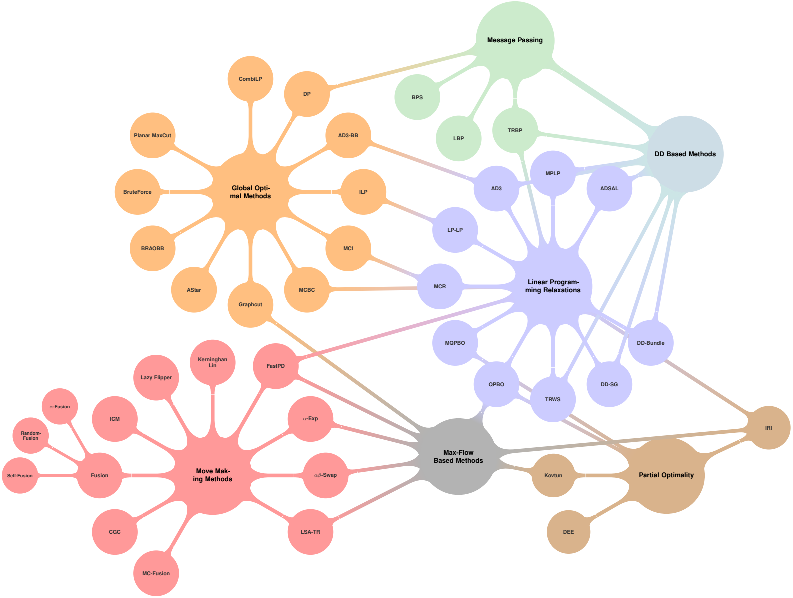


How to use OpenGM



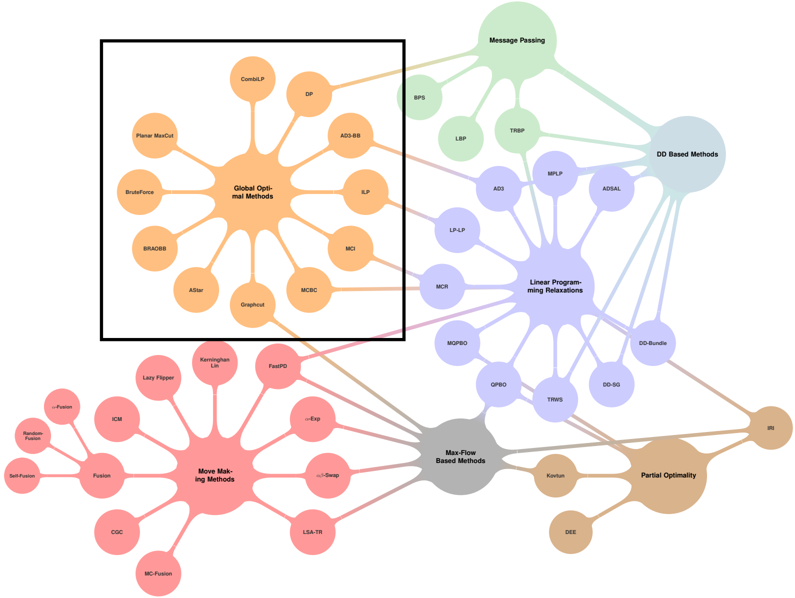
Inference Methods for Energy Minimization with Discrete Graphical Models

Energy Minimization Methods available in OpenGM



Exact Inference Methods for Energy Minimization with Discrete Graphical Models

Exact Inference Methods



Exact Inference Methods for Energy Minimization with Discrete Graphical Models

$$\arg\min_{x \in \mathcal{X}} \sum_{f \in F} \varphi_f(x_{ne(f)})$$

Exact Inference Methods for Energy Minimization with Discrete Graphical Models

$$\arg\min_{x \in \mathcal{X}} \sum_{f \in F} \varphi_f(x_{ne(f)})$$

Stop! This problem is **NP-hard**, so exact inference is **in general** not tractable!

Exact Inference Methods for Energy Minimization with Discrete Graphical Models

$$\arg\min_{x \in \mathcal{X}} \sum_{f \in F} \varphi_f(x_{ne(f)})$$

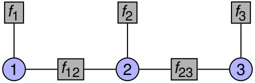
	Problem Restriction	**Runtime Complexity**
Dynamic Programming		
→ Viterbi Algorithm	acyclic structure	polynomial
→ Junction Tree Algorithm	limited tree-width	polynomial in the tree-width
∼ Loopy Belief Propagation*	no, but not exact	polynomial in the model order per iteration
Reduction to		
→ Max Flow	pairwise submodular	polynomial
→ Submodular Minimization	submodular	polynomial
→ Perfect Matching	(outer) planar binary	polynomial
→ Multicut / Multiway Cut Problem	Potts-(like) functions	exponential in the worst case
Reduction to a Search Problem		
→ Brute Force Search	no	exponential
→ Best First Search	no	exponential in the worst case
Integer Linear Program	no	exponential in the worst case

*** Loopy Belief Propagation is not exact method, but explained here due to its relation to dynamic programming!**

Exact Inference by Dynamic Programming (aka Viterbi algorithm)

Exact Inference Methods: Dynamic Programming on a Chain

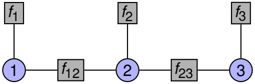
Optimization Problem



$$\arg\min_{x_1,x_2,x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_{23}}(x_2, x_3)$$

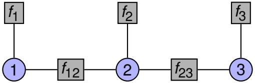
Exact Inference Methods: Dynamic Programming on a Chain

Optimization Problem



$$\arg\min_{x_1,x_2,x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1,x_2) + \varphi_{f_{23}}(x_2,x_3)$$

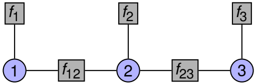
Optimization by Dynamic Programming



$$\min_{x_1} \varphi_{f_1}(x_1) + \min_{x_2} \varphi_{f_{12}}(x_1,x_2) + \varphi_{f_2}(x_2) + \min_{x_3} \varphi_{f_{23}}(x_2,x_3) + \varphi_{f_3}(x_3)$$

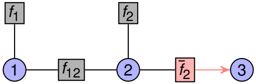
Exact Inference Methods: Dynamic Programming on a Chain

Optimization Problem



$$\arg\min_{x_1,x_2,x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1,x_2) + \varphi_{f_{23}}(x_2,x_3)$$

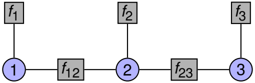
Optimization by Dynamic Programming



$$\min_{x_1} \varphi_{f_1}(x_1) + \min_{x_2} \varphi_{f_{12}}(x_1,x_2) + \varphi_{f_2}(x_2) + {\color{red}\varphi_{\bar{f}_2}(x_2)}$$

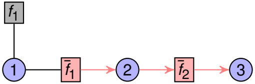
Exact Inference Methods: Dynamic Programming on a Chain

Optimization Problem



$$\arg\min_{x_1,x_2,x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_{23}}(x_2, x_3)$$

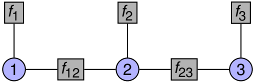
Optimization by Dynamic Programming



$$\min_{x_1} \varphi_{f_1}(x_1) + \color{red}{\varphi_{\bar{f}_1}(x_1)}$$

Exact Inference Methods: Dynamic Programming on a Chain

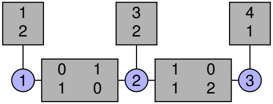
Optimization Problem



$$\arg\min_{x_1,x_2,x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_{23}}(x_2, x_3)$$

Exact Inference Methods: Dynamic Programming on a Chain

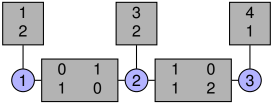
Optimization Problem



$$\arg\min_{x_1,x_2,x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_{23}}(x_2, x_3)$$

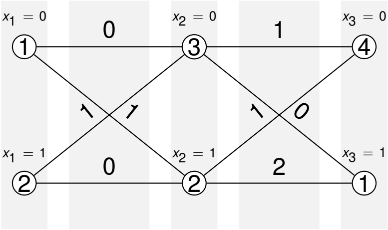
Exact Inference Methods: Dynamic Programming on a Chain

Optimization Problem



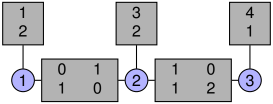
$$\arg\min_{x_1,x_2,x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_{23}}(x_2, x_3)$$

Optimization by Dynamic Programming



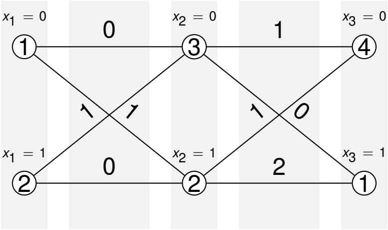
Exact Inference Methods: Dynamic Programming on a Chain

Optimization Problem



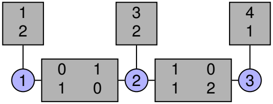
$$\arg \min_{x_1, x_2, x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_{23}}(x_2, x_3)$$

Optimization by Dynamic Programming



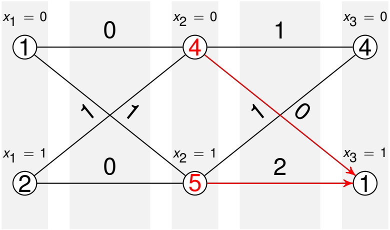
Exact Inference Methods: Dynamic Programming on a Chain

Optimization Problem



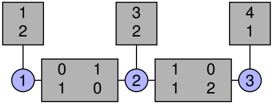
$$\arg \min_{x_1, x_2, x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_{23}}(x_2, x_3)$$

Optimization by Dynamic Programming



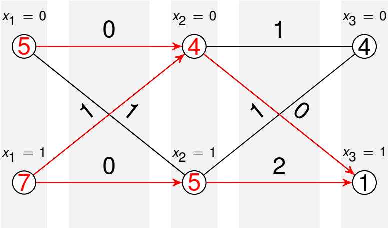
Exact Inference Methods: Dynamic Programming on a Chain

Optimization Problem



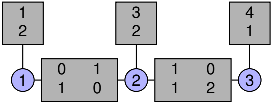
$$\arg\min_{x_1, x_2, x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_{23}}(x_2, x_3)$$

Optimization by Dynamic Programming



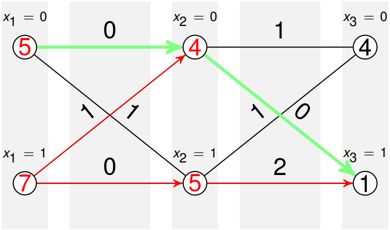
Exact Inference Methods: Dynamic Programming on a Chain

Optimization Problem



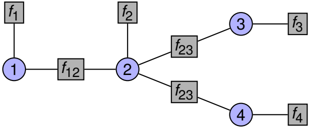
$$\arg\min_{x_1,x_2,x_3} \varphi_{f_1}(x_1) + \varphi_{f_2}(x_2) + \varphi_{f_3}(x_3) + \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_{23}}(x_2, x_3)$$

Optimization by Dynamic Programming



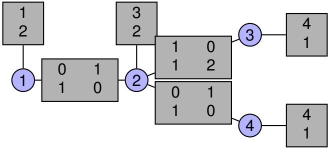
Exact Inference Methods: Dynamic Programming on a Tree

Factor Graph Model



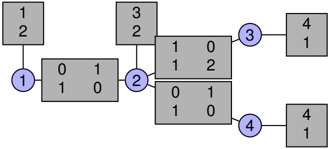
Exact Inference Methods: Dynamic Programming on a Tree

Factor Graph Model

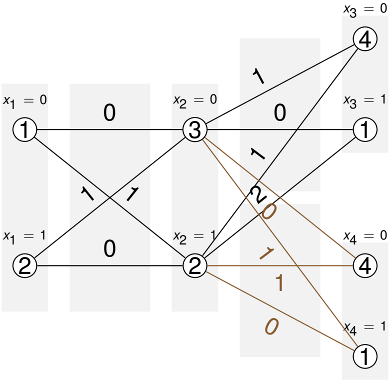


Exact Inference Methods: Dynamic Programming on a Tree

Factor Graph Model

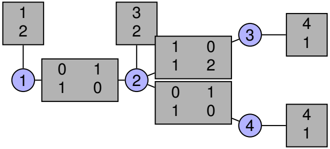


Optimization by Dynamic Programming

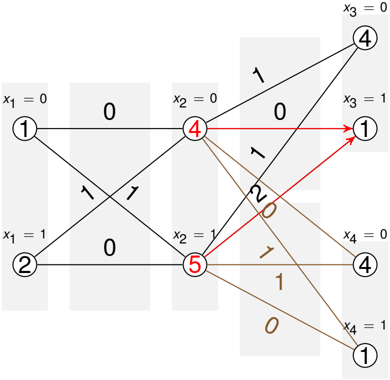


Exact Inference Methods: Dynamic Programming on a Tree

Factor Graph Model

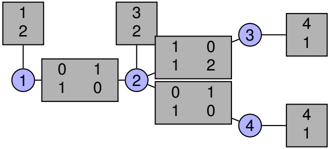


Optimization by Dynamic Programming

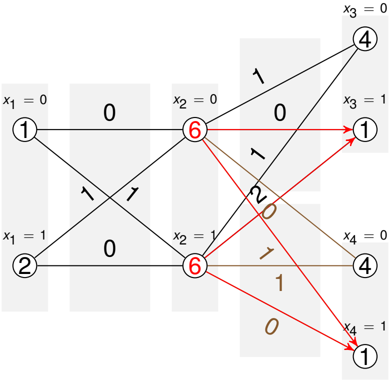


Exact Inference Methods: Dynamic Programming on a Tree

Factor Graph Model

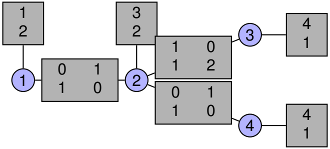


Optimization by Dynamic Programming

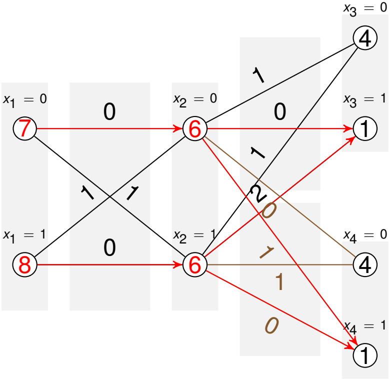


Exact Inference Methods: Dynamic Programming on a Tree

Factor Graph Model

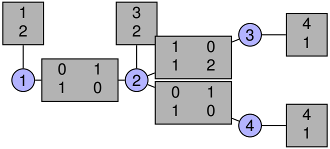


Optimization by Dynamic Programming

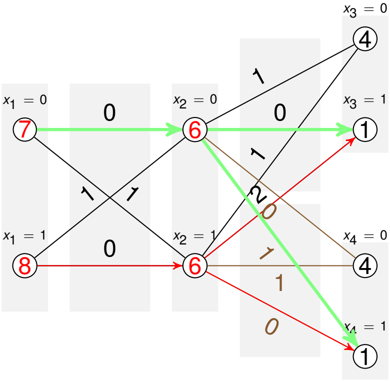


Exact Inference Methods: Dynamic Programming on a Tree

Factor Graph Model

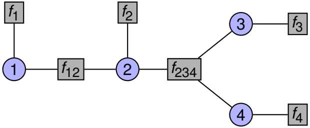


Optimization by Dynamic Programming

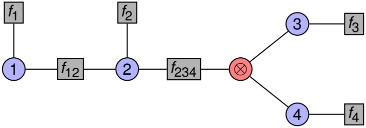


Exact Inference Methods: Dynamic Programming on a Higher-Order Tree

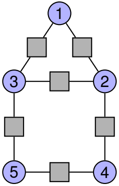
Factor Graph Model



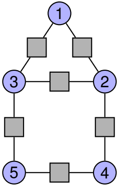
Algorithmic Idea



Exact Inference Methods: Dynamic Programming on General Graphs

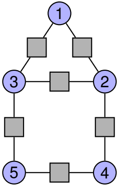


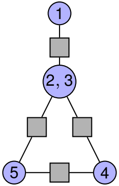
Exact Inference Methods: Dynamic Programming on General Graphs



Why did't You merge node 2 and 3
...

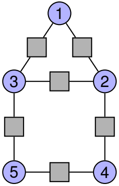
Exact Inference Methods: Dynamic Programming on General Graphs

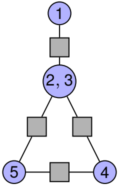




Why did't You merge node 2 and 3
...

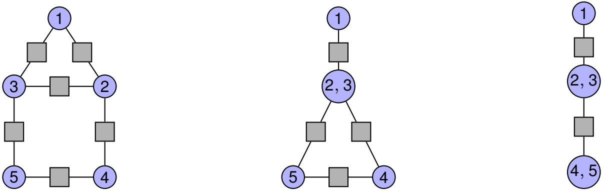
Exact Inference Methods: Dynamic Programming on General Graphs





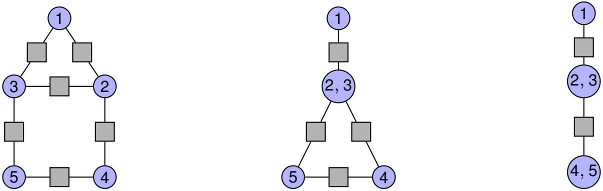
Why did't You merge node 2 and 3
…and node 4 and 5 …

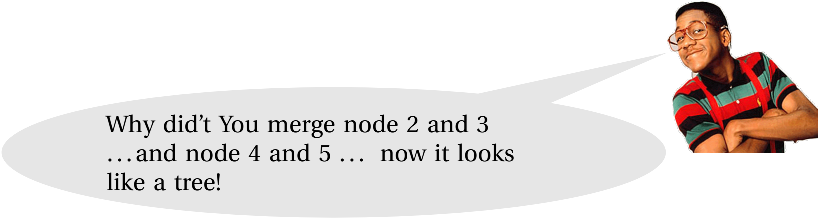
Exact Inference Methods: Dynamic Programming on General Graphs



Why did't You merge node 2 and 3
…and node 4 and 5 …

Exact Inference Methods: Dynamic Programming on General Graphs

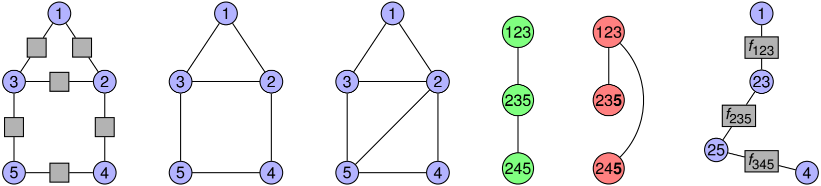




Exact Inference Methods: Dynamic Programming on General Graphs

Background

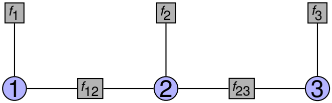
- **Idea:** Reduce inference to dynamic programming on the *Junction Tree*
- The junction tree is a cluster tree that fulfills the *Running Intersection Property*. It exists if and only if the node-adjacency graph is chordal.
- Finding the triangulation that generates a graph with the lowest clique-size is NP-hard.



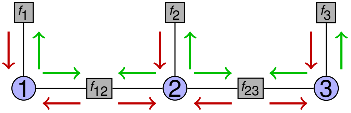
[Lauritzen, 1996, Koller and Friedman, 2009]

(Loopy) Belief Propagation
(Not always exact, but most often useful)

Belief Propagation



Belief Propagation

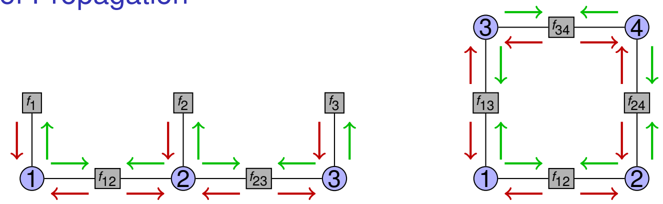


Messages

$$m_{v \to f}(x_v) = \sum_{f' \in \mathrm{ne}(v) \setminus \{f\}} m_{f' \to v}(x_v)$$

$$m_{f \to v}(x_v) = \min_{x_{\mathrm{ne}(f) \setminus \{v\}} \in X_{\mathrm{ne}(f) \setminus \{v\}}} \varphi_f(x_{\mathrm{ne}(f)}) + \sum_{u \in \mathrm{ne}(f) \setminus \{v\}} m_{u \to f}(x_v)$$

Belief Propagation

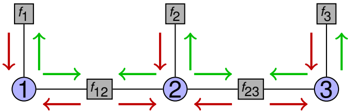


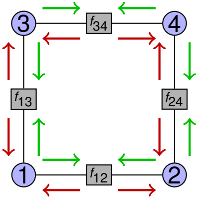
Messages

$$m_{v\to f}(x_v) = \sum_{f'\in \mathrm{ne}(v)\setminus\{f\}} m_{f'\to v}(x_v)$$

$$m_{f\to v}(x_v) = \min_{x_{\mathrm{ne}(f)\setminus\{v\}}\in X_{\mathrm{ne}(f)\setminus\{v\}}} \varphi_f(x_{\mathrm{ne}(f)}) + \sum_{u\in \mathrm{ne}(f)\setminus\{v\}} m_{u\to f}(x_v)$$

Belief Propagation





Messages

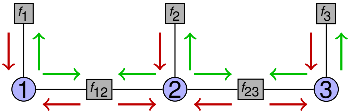
$$m_{v\to f}(x_v) = \sum_{f'\in \mathrm{ne}(v)\setminus\{f\}} m_{f'\to v}(x_v)$$

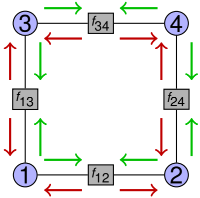
$$m_{f\to v}(x_v) = \min_{x_{\mathrm{ne}(f)\setminus\{v\}}\in X_{\mathrm{ne}(f)\setminus\{v\}}} \varphi_f(x_{\mathrm{ne}(f)}) + \sum_{u\in \mathrm{ne}(f)\setminus\{v\}} m_{u\to f}(x_v)$$

Damping

$$m^{\mathrm{new}}(x_v) = (1-\alpha)\cdot m(x_v)+\alpha\cdot m^{\mathrm{old}}(x_v)$$

Belief Propagation





Messages

$$m_{v\to f}(x_v) = \sum_{f' \in \mathrm{ne}(v)\setminus\{f\}} m_{f'\to v}(x_v)$$

$$m_{f\to v}(x_v) = \min_{x_{\mathrm{ne}(f)\setminus\{v\}} \in X_{\mathrm{ne}(f)\setminus\{v\}}} \varphi_f(x_{\mathrm{ne}(f)}) + \sum_{u \in \mathrm{ne}(f)\setminus\{v\}} m_{u\to f}(x_v)$$

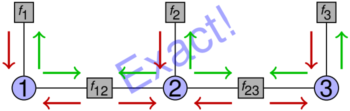
Damping

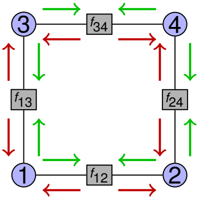
$$m^{\mathrm{new}}(x_v) = (1-\alpha)\cdot m(x_v) + \alpha \cdot m^{\mathrm{old}}(x_v)$$

Normalization

$$m^{\mathrm{new}}(x_v) = m(x_v) - \min_{x'_v \in X_v} m(x'_v)$$

Belief Propagation





Messages

$$m_{v\to f}(x_v) = \sum_{f'\in \mathrm{ne}(v)\setminus\{f\}} m_{f'\to v}(x_v)$$

$$m_{f\to v}(x_v) = \min_{x_{\mathrm{ne}(f)\setminus\{v\}}\in X_{\mathrm{ne}(f)\setminus\{v\}}} \varphi_f(x_{\mathrm{ne}(f)}) + \sum_{u\in \mathrm{ne}(f)\setminus\{v\}} m_{u\to f}(x_v)$$

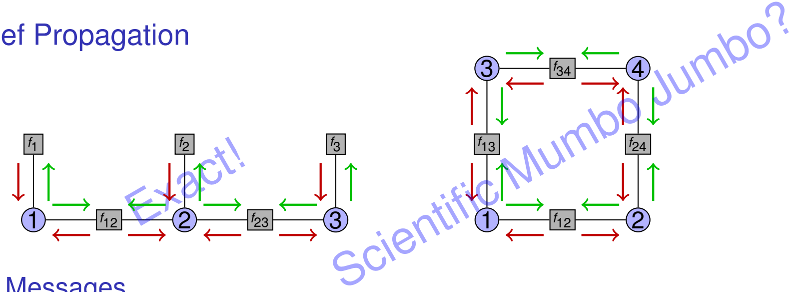
Damping

$$m^{\mathrm{new}}(x_v) = (1-\alpha)\cdot m(x_v) + \alpha\cdot m^{\mathrm{old}}(x_v)$$

Normalization

$$m^{\mathrm{new}}(x_v) = m(x_v) - \min_{x'_v\in X_v} m(x'_v)$$

Belief Propagation



Messages

$$m_{v\to f}(x_v) = \sum_{f'\in \mathrm{ne}(v)\setminus\{f\}} m_{f'\to v}(x_v)$$

$$m_{f\to v}(x_v) = \min_{x_{\mathrm{ne}(f)\setminus\{v\}}\in X_{\mathrm{ne}(f)\setminus\{v\}}} \varphi_f(x_{\mathrm{ne}(f)}) + \sum_{u\in \mathrm{ne}(f)\setminus\{v\}} m_{u\to f}(x_v)$$

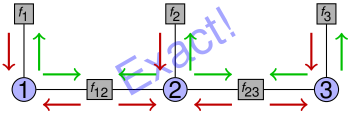
Damping

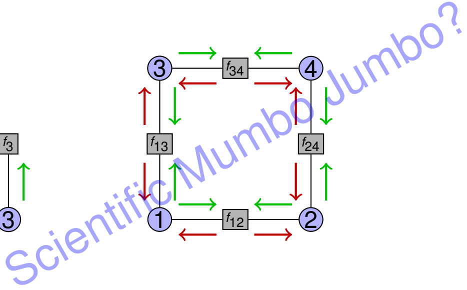
$$m^{\mathrm{new}}(x_v) = (1-\alpha)\cdot m(x_v)+\alpha\cdot m^{\mathrm{old}}(x_v)$$

Normalization

$$m^{\mathrm{new}}(x_v) = m(x_v) - \min_{x'_v\in X_v} m(x'_v)$$

Belief Propagation





Messages

$$m_{v \to f}(x_v) = \sum_{f' \in \mathrm{ne}(v) \setminus \{f\}} m_{f' \to v}(x_v)$$

$$m_{f \to v}(x_v) = \min_{x_{\mathrm{ne}(f) \setminus \{v\}} \in X_{\mathrm{ne}(f) \setminus \{v\}}} \varphi_f(x_{\mathrm{ne}(f)}) + \sum_{u \in \mathrm{ne}(f) \setminus \{v\}} m_{u \to f}(x_v)$$

Damping

$$m^{\mathrm{new}}(x_v) = (1-\alpha) \cdot m(x_v) + \alpha \cdot m^{\mathrm{old}}(x_v)$$

Normalization

$$m^{\mathrm{new}}(x_v) = m(x_v) - \min_{x'_v \in X_v} m(x'_v)$$

Works surprisingly good for cyclic models...

Belief Propagation

Message Schedule

- Parallel $\rightarrow$ Loopy Belief Propagation (LBP)
  [Pearl, 1988]
- Sequential $\rightarrow$ Sequential Belief Propagation (BPS)
  [Felzenszwalb and Huttenlocher, 2006]
- Informed $\rightarrow$ Residual Belief Propagation (RBP)
  [Elidan et al., 2006]

Belief Propagation

Message Schedule

- Parallel $\rightarrow$ Loopy Belief Propagation (LBP)
  [Pearl, 1988]
- Sequential $\rightarrow$ Sequential Belief Propagation (BPS)
  [Felzenszwalb and Huttenlocher, 2006]
- Informed $\rightarrow$ Residual Belief Propagation (RBP)
  [Elidan et al., 2006]

Message Update Rules (from a theoretical point of view)

- Original Updates: Optimize a non-convex objective function
  $\rightarrow$ lack of convergence
- Modified Updates: Optimize a convex objective function
  $\rightarrow$ we will come back to this point later

Belief Propagation

Message Schedule

- Parallel → Loopy Belief Propagation (LBP)
  [Pearl, 1988]
- Sequential → Sequential Belief Propagation (BPS)
  [Felzenszwalb and Huttenlocher, 2006]
- Informed → Residual Belief Propagation (RBP)
  [Elidan et al., 2006]

Message Update Rules (from a theoretical point of view)

- Original Updates: Optimize a non-convex objective function
  → lack of convergence
- Modified Updates: Optimize a convex objective function
  → we will come back to this point later

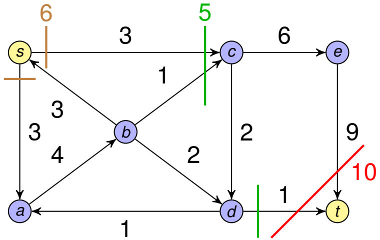
[Kschischang et al., 2001] → generalization to other semi-rings

Reduction to Min-st-Cut/Max-Flow Problem (aka GraphCut)

Minimal st-Cut Problem

Definition: Min-st-Cut Problem

Given a weighted directed graph $G = (V, E, w)$ with a source-node $s \in V$ and a sink-node $t \in V$. Find the subset of edges $E' \subset E$ with the minimal edge-weight $\sum_{e \in E'} w(e)$ such that no path from s to t exists in $G' = (V, E \setminus E')$.



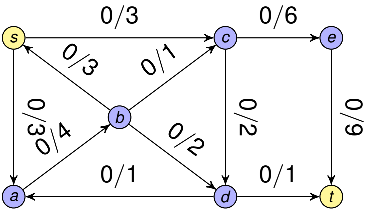
Remark

If the edge-weights are non-negative the Min-st-Cut problem is efficiently solvable.

Max Flow Problem

Definition: Max-Flow Problem

Given a weighted directed graph $G = (V, E, w)$ with a source-node $s \in V$ and a sink-node $t \in V$. Find the maximal flow passing from s to t, where a positive flow smaller than $w(e)$ flow can be passed over the edge e and conservation holds.



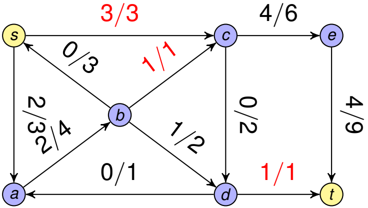
Max-Flow Min-Cut Theorem

The maximum value of an st flow with capacity $w(e)$ is equal to the minimum st cuts with edge weights $w(e)$.

Max Flow Problem

Definition: Max-Flow Problem

Given a weighted directed graph $G = (V, E, w)$ with a source-node $s \in V$ and a sink-node $t \in V$. Find the maximal flow passing from s to t, where a positive flow smaller than $w(e)$ flow can be passed over the edge e and conservation holds.



Max-Flow Min-Cut Theorem

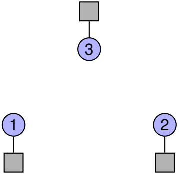
The maximum value of an st flow with capacity $w(e)$ is equal to the minimum st cuts with edge weights $w(e)$.

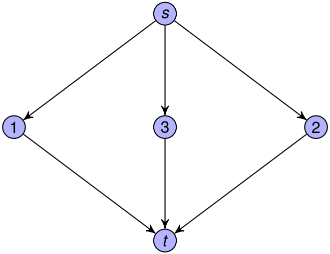
Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms

Reformulation:



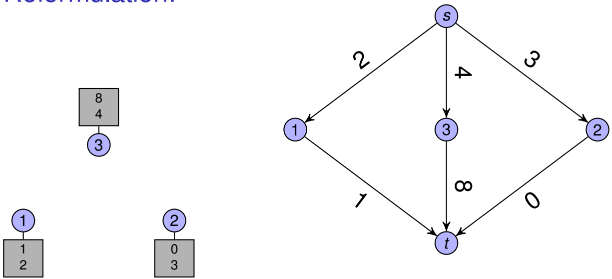


Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms

Reformulation:

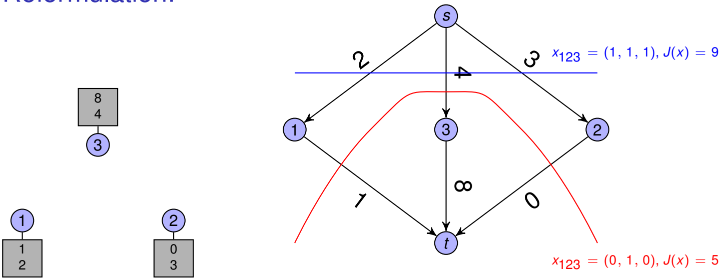


Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms

Reformulation:

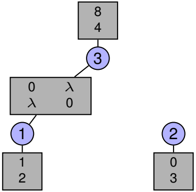


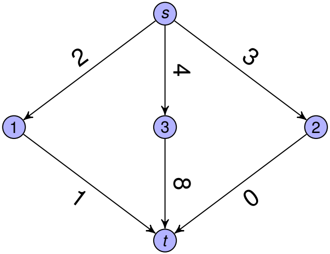
Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms

Reformulation:



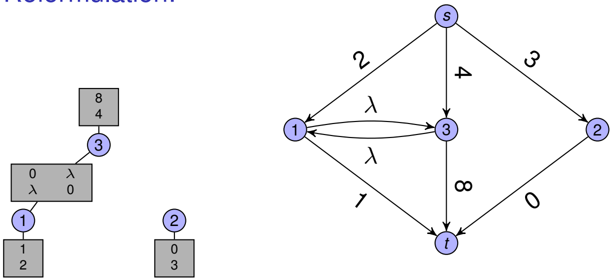


Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms

Reformulation:

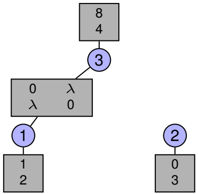


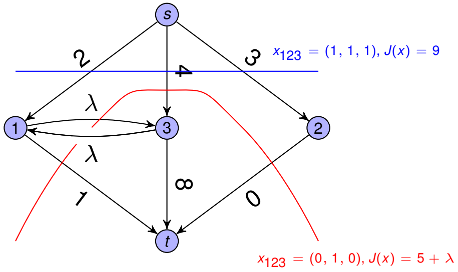
Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms

Reformulation:



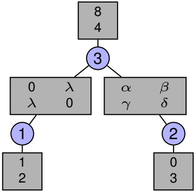


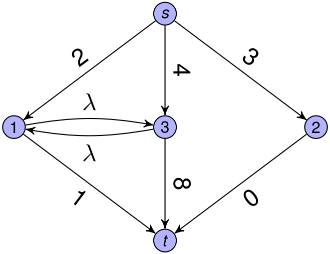
Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms

Reformulation:





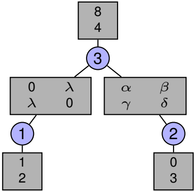
$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} \alpha & \alpha \\ \alpha & \alpha \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \gamma - \alpha & \gamma - \alpha \end{pmatrix} + \begin{pmatrix} 0 & \delta - \gamma \\ 0 & \delta - \gamma \end{pmatrix} + \begin{pmatrix} 0 & \beta + \gamma - \alpha - \delta \\ 0 & 0 \end{pmatrix}$$

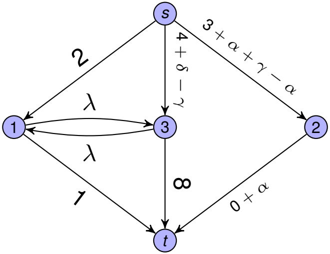
Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms

Reformulation:





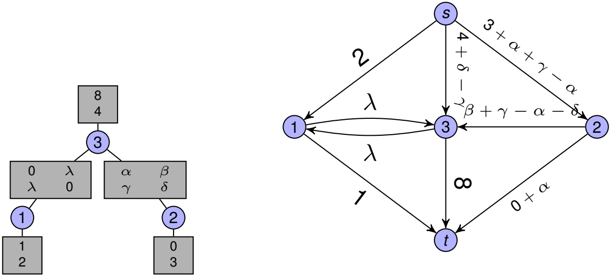
$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} \alpha & \alpha \\ \alpha & \alpha \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \gamma - \alpha & \gamma - \alpha \end{pmatrix} + \begin{pmatrix} 0 & \delta - \gamma \\ 0 & \delta - \gamma \end{pmatrix} + \begin{pmatrix} 0 & \beta + \gamma - \alpha - \delta \\ 0 & 0 \end{pmatrix}$$

Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms ($\beta + \gamma - \alpha - \delta \geq 0$)

Reformulation:



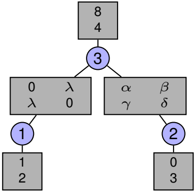
$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} \alpha & \alpha \\ \alpha & \alpha \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \gamma - \alpha & \gamma - \alpha \end{pmatrix} + \begin{pmatrix} 0 & \delta - \gamma \\ 0 & \delta - \gamma \end{pmatrix} + \begin{pmatrix} 0 & \beta + \gamma - \alpha - \delta \\ 0 & 0 \end{pmatrix}$$

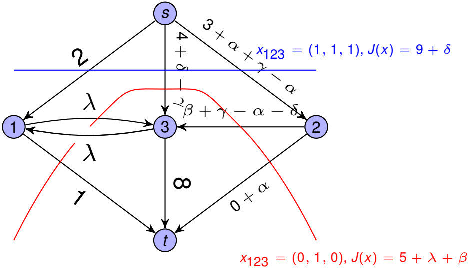
Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms ($\beta + \gamma - \alpha - \delta \geq 0$)

Reformulation:





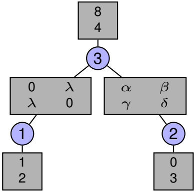
$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} \alpha & \alpha \\ \alpha & \alpha \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \gamma - \alpha & \gamma - \alpha \end{pmatrix} + \begin{pmatrix} 0 & \delta - \gamma \\ 0 & \delta - \gamma \end{pmatrix} + \begin{pmatrix} 0 & \beta + \gamma - \alpha - \delta \\ 0 & 0 \end{pmatrix}$$

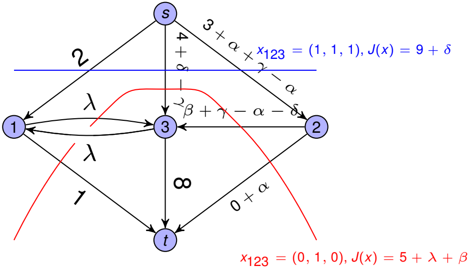
Exact Inference Methods: Reformulate into Min-st-Cut

Requirements:

- Binary label-space
- Regular/submodular pairwise terms ($\beta + \gamma - \alpha - \delta \geq 0$)

Reformulation:





$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} \alpha & \alpha \\ \alpha & \alpha \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \gamma - \alpha & \gamma - \alpha \end{pmatrix} + \begin{pmatrix} 0 & \delta - \gamma \\ 0 & \delta - \gamma \end{pmatrix} + \begin{pmatrix} 0 & \beta + \gamma - \alpha - \delta \\ 0 & 0 \end{pmatrix}$$

binary *[Greig et al., 1989, Boykov and Kolmogorov, 2004]* multilabel *[Ishikawa, 2003]*

Reduction to Submodular Minimization

Submodular Minimization

$$\arg\min_{S \in 2^V} f(S)$$

Submodular Function

If V is a finite set, a submodular function is a set function $f : 2^V \to \mathbb{R}$, where 2^V denotes the power set of V, for which for every $S, T \subseteq V$ the inequality $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ holds.

Minimizing Submodular Function

Finding the subset $S \subset V$ that minimize a submodular function is computable in polynomial time *[Schrijver, 2000, Iyer et al., 2013]*

Relation to Binary Graphical Models

Let S be the set of variables taking label 0 and $V \setminus S$ the set of labels taking label 1.

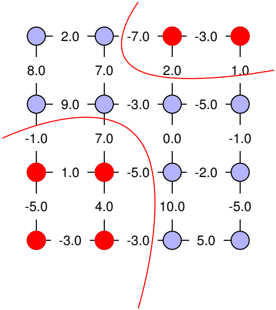
ICML-2013 Tutorial by Stefanie Jegelka and Andreas Krause (`http://submodularity.org/`)

Reduction to Perfect Matching via Max-Cut on Planar Graphs

Max Cut Problem

Definition:

Given a weighted undirected graph $G = (V, E, w)$ the Max-Cut problem is to find a cut $E' \subset E$, that divide V into two sets, that the sum of the weights of cut edges is maximized.

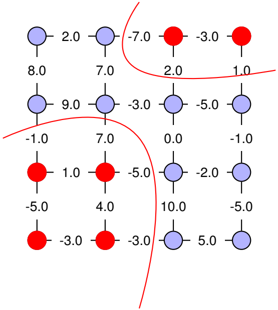


Example for a (not maximal) cut

Max Cut Problem

Definition:

Given a weighted undirected graph $G = (V, E, w)$ the Max-Cut problem is to find a cut $E' \subset E$, that divide V into two sets, that the sum of the weights of cut edges is maximized.



Example for a (not maximal) cut

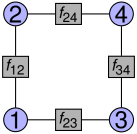
Remark:

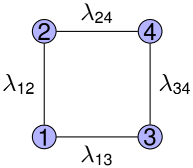
For planar graphs the Max Cut problem can be efficiently solved *[Kasteleyn, 1961, Fisher, 1961] [Globerson and Jaakkola, 2006, Schraudolph and Kamenetsky, 2008]* by a reduction to a Perfect Matching problem, e.g. Blossom V *[Kolmogorov, 2009]*, on some expanded dual graph.

Reduction to a Max Cut Problem

$$x_i \in \{0,1\} \qquad \varphi_{f_{ij}}(x_i,x_j) = -\lambda_{ij}\mathbb{I}(x_i \neq x_j) \qquad \varphi_{u_i}(x_i) = -\alpha_i\mathbb{I}(x_i = 0)$$

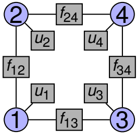
Planar Potts Model without Unaries

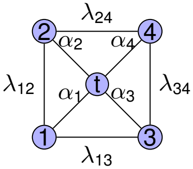




x_i = cluster number of node i

Outer-Planar Potts Model with Unaries





$$x_i = \begin{cases} 1 & \text{if the edge}(ti) \text{ is not cut} \\ 0 & \text{otherwise} \end{cases}$$

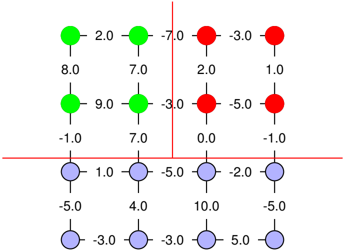
[Schraudolph and Kamenetsky, 2008]

Reduction to Minimal Multicut Problem

Minimal Multicut Problem

Definition:

Given a weighted undirected graph $G = (V, E, w)$ the Minimal Multicut problem is to find a cut $E' \subset E$, that divide V into a unknown number of sets, that the sum of the weights of cut edges is minimized.

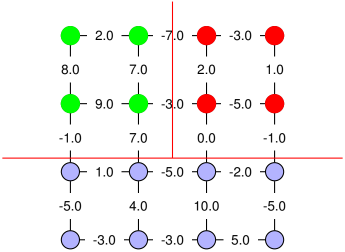


Example for a (not minimal) multicut

Minimal Multicut Problem

Definition:

Given a weighted undirected graph $G = (V, E, w)$ the Minimal Multicut problem is to find a cut $E' \subset E$, that divide V into a unknown number of sets, that the sum of the weights of cut edges is minimized.



Example for a (not minimal) multicut

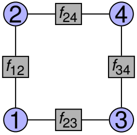
Remark:

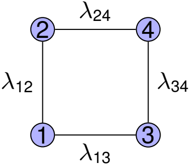
The above multicut is not a cut, since the clustering is not 2-colorable!

Reduction to a Minimal Multicut Problem

$$x_i \in \{0, \dots, L\} \qquad \varphi_{f_{ij}}(x_i, x_j) = \lambda_{ij}\mathbb{I}(x_i \neq x_j) \qquad \varphi_{u_i}(x_i) = \alpha_{i;x_i}$$

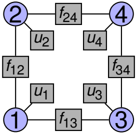
Potts Model with $L = |V|$ and without Unaries $\rightarrow$ Multicut

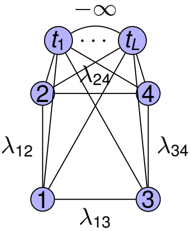




$x_i =$ cluster number of node i

Potts Model with Unaries $\rightarrow$ Multiway Cut





$$\begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} w_{t_1 i} \\ \vdots \\ \vdots \\ w_{t_L i} \end{pmatrix} = \begin{pmatrix} \alpha_{i;1} - \alpha_{i;0} \\ \vdots \\ \vdots \\ \alpha_{i;L} - \alpha_{i;0} \end{pmatrix}$$

$$x_i = \begin{cases} l & \text{if the edge}(t_l i) \text{ is not cut} \\ 0 & \text{otherwise} \end{cases}$$

Reduction to Multicut/Multiway Cut Problem

Reduction to Max Flow

Label-space:	$x_i \in \{0, 1\}$
Unary terms:	arbitrary
Pairwise terms:	$\varphi_{f_{ij}}(x_i, x_j) = \lambda_{ij}\mathbb{I}(x_i \neq x_j) \qquad \lambda_{ij} \in \mathbb{R}^+$ (submodular)
Problem structure:	arbitrary
Runtime complexity:	polynomial

Reduction to Multicut/Multiway Cut Problem

Reduction to Max Flow

Label-space:	$x_i \in \{0, 1\}$
Unary terms:	arbitrary
Pairwise terms:	$\varphi_{f_{ij}}(x_i, x_j) = \lambda_{ij}\mathbb{I}(x_i \neq x_j) \qquad \lambda_{ij} \in \mathbb{R}^+$ (submodular)
Problem structure:	arbitrary
Runtime complexity:	polynomial

Reduction to Max Cut

Label-space:	$x_i \in \{0, 1\}$
Unary terms:	none[1] or arbitrary[2]
Pairwise terms:	$\varphi_{f_{ij}}(x_i, x_j) = \lambda_{ij}\mathbb{I}(x_i \neq x_j) \qquad \lambda_{ij} \in \mathbb{R}$
Problem structure:	planar[1] or outer planar[2]
Runtime complexity:	polynomial

Reduction to Multicut/Multiway Cut Problem

Reduction to Max Flow

Label-space:	$x_i \in \{0, 1\}$		
Unary terms:	arbitrary		
Pairwise terms:	$\varphi_{f_{ij}}(x_i, x_j) = \lambda_{ij}\mathbb{I}(x_i \neq x_j)$	$\lambda_{ij} \in \mathbb{R}^+$	(submodular)
Problem structure:	arbitrary		
Runtime complexity:	polynomial		

Reduction to Max Cut

Label-space:	$x_i \in \{0, 1\}$	
Unary terms:	none[1] or arbitrary[2]	
Pairwise terms:	$\varphi_{f_{ij}}(x_i, x_j) = \lambda_{ij}\mathbb{I}(x_i \neq x_j)$	$\lambda_{ij} \in \mathbb{R}$
Problem structure:	planar[1] or outer planar[2]	
Runtime complexity:	polynomial	

Reduction to Multicut / Multiway Cut

Label-space:	$x_i \in \{0, \dots, L\}$	
Unary terms:	none or arbitrary	
Pairwise terms:	$\varphi_{f_{ij}}(x_i, x_j) = \lambda_{ij}\mathbb{I}(x_i \neq x_j)$	$\lambda_{ij} \in \mathbb{R}$
Problem structure:	arbitrary	
Runtime complexity:	exponential in the worst case	

Reduction to Multicut/Multiway Cut Problem

Reduction to Max Flow

Label-space:	$x_i \in \{0, 1\}$
Unary terms:	arbitrary
Pairwise terms:	$\varphi_{f_{ij}}(x_i, x_j) = \lambda_{ij}\mathbb{I}(x_i \neq x_j) \qquad \lambda_{ij} \in \mathbb{R}^+$ (submodular)
Problem structure:	arbitrary
Runtime complexity:	polynomial

Reduction to Max Cut

Label-space:	$x_i \in \{0, 1\}$
Unary terms:	none[1] or arbitrary[2]
Pairwise terms:	$\varphi_{f_{ij}}(x_i, x_j) = \lambda_{ij}\mathbb{I}(x_i \neq x_j) \qquad \lambda_{ij} \in \mathbb{R}$
Problem structure:	planar[1] or outer planar[2]
Runtime complexity:	polynomial

Reduction to Multicut / Multiway Cut

Label-space:	$x_i \in \{0, \ldots, L\}$
Unary terms:	none or arbitrary
Pairwise terms:	$\varphi_{f_{ij}}(x_i, x_j) = \lambda_{ij}\mathbb{I}(x_i \neq x_j) \qquad \lambda_{ij} \in \mathbb{R}$
Problem structure:	arbitrary
Runtime complexity:	exponential in the worst case, but often tractable in practice

Solving a Multicut/Multiway Cut Problem by an (I)LP

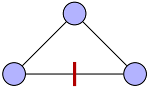
LP Formulation

$$\min_{y \in [0,1]^E} \sum_{e \in E} w_e \cdot y_e$$
$$s.t. \quad A y \leq b$$

Solving a Multicut/Multiway Cut Problem by an (I)LP

LP Formulation

$$\min_{y \in [0,1]^E} \sum_{e \in E} w_e \cdot y_e$$
$$s.t. \quad A\,y \le b$$



This is no valid multicut! $\Rightarrow Ay \le b$

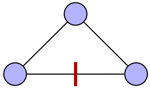
Solving a Multicut/Multiway Cut Problem by an (I)LP

LP Formulation

$$\min_{y \in [0,1]^E} \sum_{e \in E} w_e \cdot y_e$$

$$s.t. \quad \boxed{A\,y \leq b}$$

exponential size



This is no valid multicut! $\Rightarrow Ay \leq b$

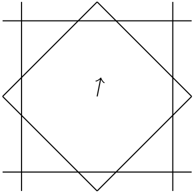
Solving a Multicut/Multiway Cut Problem by an (I)LP

LP Formulation

$$\min_{y \in [0,1]^E} \sum_{e \in E} w_e \cdot y_e$$

$$s.t. \quad \underbrace{\boxed{A\,y \leq b}}_{\text{exponential size}}$$

Idea of Cutting-Plane Methods



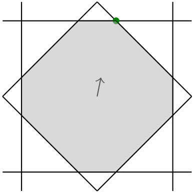
Solving a Multicut/Multiway Cut Problem by an (I)LP

LP Formulation

$$\min_{y \in [0,1]^E} \sum_{e \in E} w_e \cdot y_e$$

$$s.t. \quad \underbrace{\boxed{A y \leq b}}_{\text{exponential size}}$$

Idea of Cutting-Plane Methods



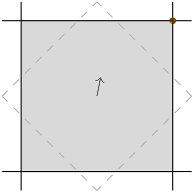
Solving a Multicut/Multiway Cut Problem by an (I)LP

LP Formulation

$$\min_{y \in [0,1]^E} \sum_{e \in E} w_e \cdot y_e$$

$$s.t. \quad \underbrace{\boxed{A\,y \leq b}}_{\text{exponential size}}$$

Idea of Cutting-Plane Methods



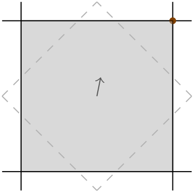
Solving a Multicut/Multiway Cut Problem by an (I)LP

LP Formulation

$$\min_{y \in [0,1]^E} \sum_{e \in E} w_e \cdot y_e$$

$$s.t. \quad \underbrace{\boxed{A\,y \leq b}}_{\text{exponential size}}$$

Idea of Cutting-Plane Methods



- **Separation Procedure:**
  Find violated constraints

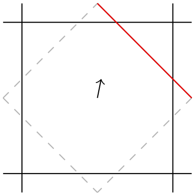
Solving a Multicut/Multiway Cut Problem by an (I)LP

LP Formulation

$$\min_{y \in [0,1]^E} \sum_{e \in E} w_e \cdot y_e$$

$$s.t. \quad \underbrace{\boxed{A y \leq b}}_{\text{exponential size}}$$

Idea of Cutting-Plane Methods



- **Separation Procedure:** Find violated constraints

Solving a Multicut/Multiway Cut Problem by an (I)LP

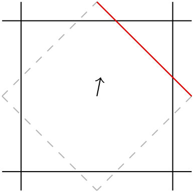
LP Formulation

$$\min_{y \in [0,1]^E} \sum_{e \in E} w_e \cdot y_e$$

$$s.t. \quad \boxed{Ay \leq b}$$

exponential size

Idea of Cutting-Plane Methods



- **Separation Procedure:**
  Find violated constraints
- The *separation procedure* is as hard as the *original problem* **or** the *original problem* is as hard as the *separation procedure*

Ellipsoid method for combinatorial optimization
[Grötschel et al.*, 1981]*

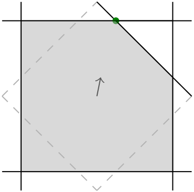
Solving a Multicut/Multiway Cut Problem by an (I)LP

LP Formulation

$$\min_{y \in [0,1]^E} \sum_{e \in E} w_e \cdot y_e$$

$$s.t. \quad \underbrace{\boxed{A y \leq b}}_{\text{exponential size}}$$

Idea of Cutting-Plane Methods



- **Separation Procedure:**
  Find violated constraints
- The *separation procedure* is as hard as the *original problem* **or** the *original problem* is as hard as the *separation procedure*

Ellipsoid method for combinatorial optimization
[Grötschel et al.*, 1981]*

Solving a Multicut/Multiway Cut Problem by an (I)LP

LP Formulation

$$\min_{y\in[0,1]^E} \sum_{e\in E} w_e \cdot y_e$$

$$s.t. \quad Ay \leq b$$

exponential size

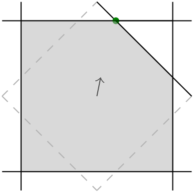
ILP Formulation

$$\min_{y\in[0,1]^E} \sum_{e\in E} w_e \cdot y_e$$

$$s.t. \quad \tilde{A}y \leq \tilde{b}$$

$$y \in \{0,1\}^E$$

Idea of Cutting-Plane Methods



- **Separation Procedure:**
  Find violated constraints
- The *separation procedure* is as hard as the *original problem* **or** the *original problem* is as hard as the *separation procedure*

Ellipsoid method for combinatorial optimization
[Grötschel et al.*, 1981]*

Solving a Multicut/Multiway Cut Problem by an (I)LP

LP Formulation

$$\min_{y\in[0,1]^E} \sum_{e\in E} w_e \cdot y_e$$

$$s.t. \quad Ay \leq b$$

exponential size

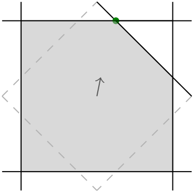
ILP Formulation

$$\min_{y\in[0,1]^E} \sum_{e\in E} w_e \cdot y_e$$

$$s.t. \quad \tilde{A}y \leq \tilde{b}$$

$$y \in \{0,1\}^E$$

Idea of Cutting-Plane Methods



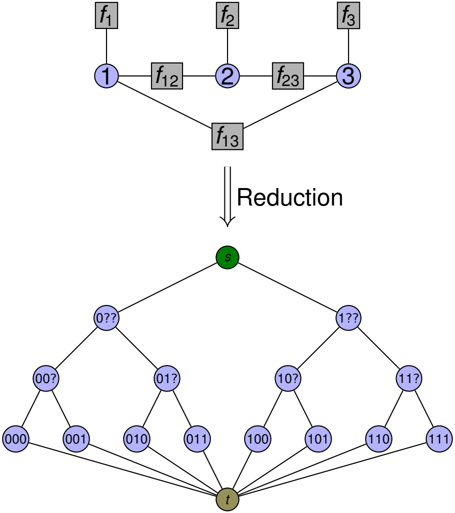
- **Separation Procedure:**
  Find violated constraints
- The *separation procedure* is as hard as the *original problem* **or** the *original problem* is as hard as the *separation procedure*

  Ellipsoid method for combinatorial optimization [Grötschel et al., 1981]

[Chopra and Rao, 1993, Kappes et al., 2011, Kappes et al., 2015b]

Reduction to Shortest Path Search

Reduction to Shortest Path Search



$X = \{0, 1\}^3$

edge weights

$\varphi_{f_1}(x_1)$

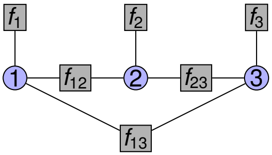
$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

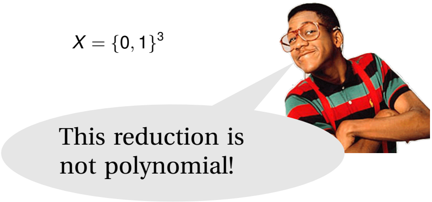
0

$\min_{x \in X} \sum_{f \in F} \varphi_f(x) \Leftrightarrow$ Finding shortest path from s to t

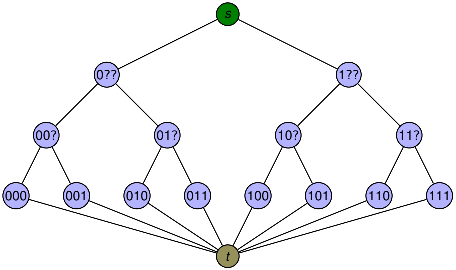
Reduction to Shortest Path Search



$X = \{0, 1\}^3$



$\Downarrow$ Reduction



edge weights

$\varphi_{f_1}(x_1)$

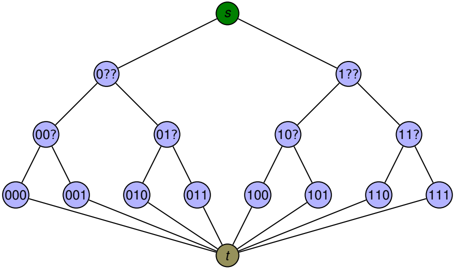
$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

$\min_{x \in X} \sum_{f \in F} \varphi_f(x) \Leftrightarrow$ Finding shortest path from s to t

Reduction to Shortest Path Search



edge weights

$\varphi_{f_1}(x_1)$

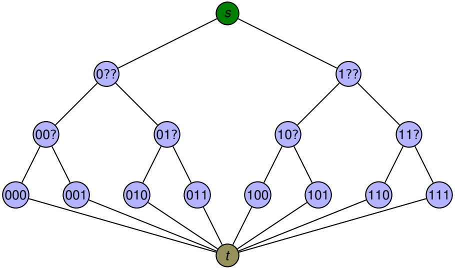
$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

Useful Ideas

Reduction to Shortest Path Search



edge weights

$\varphi_{f_1}(x_1)$

$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

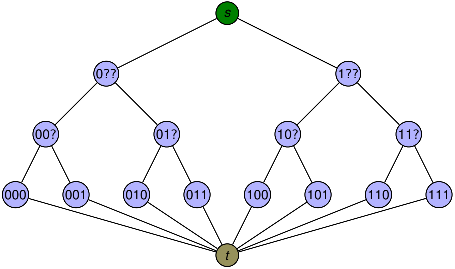
$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

Useful Ideas

- Use an implicit representation of the graph.

Reduction to Shortest Path Search



edge weights

$\varphi_{f_1}(x_1)$

$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

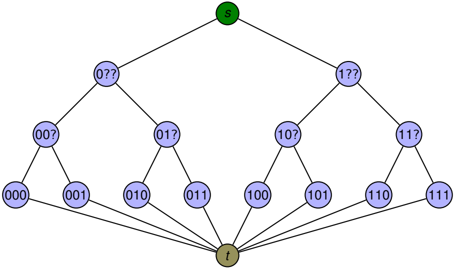
$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

Useful Ideas

- Use an implicit representation of the graph.
- Use Best First Search Methods, e.g. A^*
  $\rightarrow$ underestimate minimal future cost on the path to the node t by:

Reduction to Shortest Path Search



edge weights

$\varphi_{f_1}(x_1)$

$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

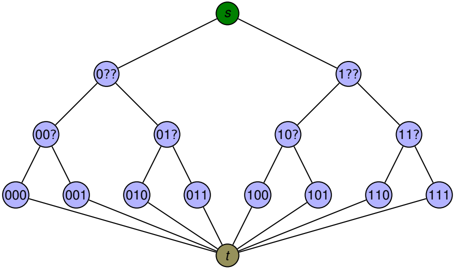
$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

Useful Ideas

- Use an implicit representation of the graph.
- Use Best First Search Methods, e.g. A^*
  $\rightarrow$ underestimate minimal future cost on the path to the node t by:
  - Tree-based heuristics *[Bergtholdt et al., 2010]*

Reduction to Shortest Path Search



edge weights

$\varphi_{f_1}(x_1)$

$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

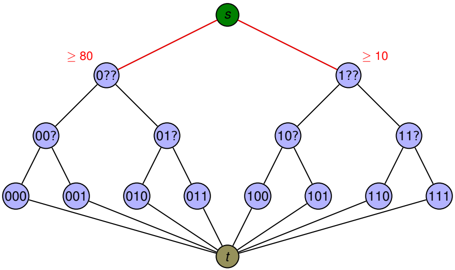
$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

Useful Ideas

- Use an implicit representation of the graph.
- Use Best First Search Methods, e.g. A^*
  $\rightarrow$ underestimate minimal future cost on the path to the node t by:
  - Tree-based heuristics *[Bergtholdt* et al.*, 2010]*
  - LP-based heuristics *[Schlesinger, 2009]*

Reduction to Shortest Path Search



edge weights

$\varphi_{f_1}(x_1)$

$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

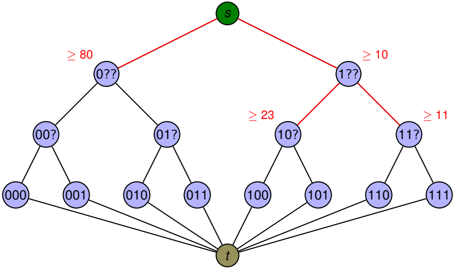
$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

Useful Ideas

- Use an implicit representation of the graph.
- Use Best First Search Methods, e.g. A^*
  $\rightarrow$ underestimate minimal future cost on the path to the node t by:
  - Tree-based heuristics *[Bergtholdt et al., 2010]*
  - LP-based heuristics *[Schlesinger, 2009]*

Reduction to Shortest Path Search



edge weights

$\varphi_{f_1}(x_1)$

$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

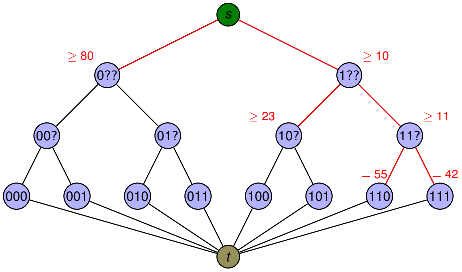
$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

Useful Ideas

- Use an implicit representation of the graph.
- Use Best First Search Methods, e.g. A^*
  $\rightarrow$ underestimate minimal future cost on the path to the node t by:
  - Tree-based heuristics *[Bergtholdt et al., 2010]*
  - LP-based heuristics *[Schlesinger, 2009]*

Reduction to Shortest Path Search



edge weights

$\varphi_{f_1}(x_1)$

$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

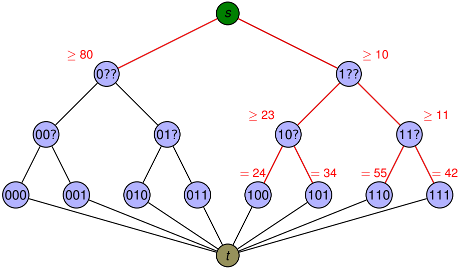
$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

Useful Ideas

- Use an implicit representation of the graph.
- Use Best First Search Methods, e.g. A^*
  $\rightarrow$ underestimate minimal future cost on the path to the node t by:
  - Tree-based heuristics [Bergtholdt et al., 2010]
  - LP-based heuristics [Schlesinger, 2009]

Reduction to Shortest Path Search



edge weights

$\varphi_{f_1}(x_1)$

$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

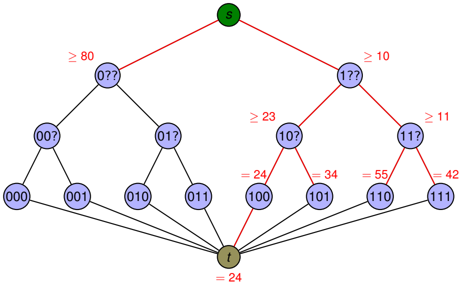
$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

Useful Ideas

- Use an implicit representation of the graph.
- Use Best First Search Methods, e.g. A^*
  $\rightarrow$ underestimate minimal future cost on the path to the node t by:
  - Tree-based heuristics [Bergtholdt et al., 2010]
  - LP-based heuristics [Schlesinger, 2009]

Reduction to Shortest Path Search



edge weights

$\varphi_{f_1}(x_1)$

$\varphi_{f_2}(x_2) + \varphi_{f_{12}}(x_1, x_2)$

$\varphi_{f_3}(x_3) + \varphi_{f_{13}}(x_1, x_3) + \varphi_{f_{23}}(x_2, x_3)$

0

Useful Ideas

- Use an implicit representation of the graph.
- Use Best First Search Methods, e.g. A^*
  $\rightarrow$ underestimate minimal future cost on the path to the node t by:
  - Tree-based heuristics *[Bergtholdt et al., 2010]*
  - LP-based heuristics *[Schlesinger, 2009]*

MAP Inference as Integer Linear Program

MAP inference as Integer Linear Program

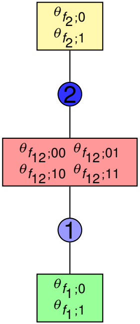
$$\begin{aligned} \min_{\mu} & \langle \theta, \mu \rangle \\ s.t. \quad & A\mu \leq b \\ & \mu \in \{0, 1\}^N \end{aligned}$$

MAP inference as Integer Linear Program

$$\min_{\mu} \langle \theta, \mu \rangle$$
$$s.t. \quad A\mu \leq b$$
$$\mu \in \{0,1\}^N$$

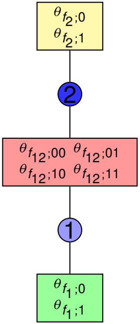
MAP inference as Integer Linear Program

$$\min_{\mu} \langle \theta, \mu \rangle$$
$$s.t. \quad A\mu \leq b$$
$$\mu \in \{0,1\}^N$$



MAP inference as Integer Linear Program

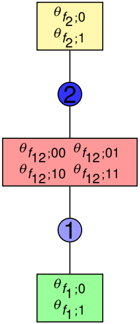
$$\min_{\mu} \langle \theta, \mu \rangle$$
$$s.t. \quad A\mu \leq b$$
$$\mu \in \{0,1\}^N$$



$$\theta \quad \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & \theta_{f_1;0} & \theta_{f_1;1} & \theta_{f_2;0} & \theta_{f_2;1} & \theta_{f_{12};00} & \theta_{f_{12};01} & \theta_{f_{12};10} & \theta_{f_{12};11} \\ \hline \end{array}$$

MAP inference as Integer Linear Program

$$\min_{\mu} \langle \theta, \mu \rangle$$
$$s.t. \quad A\mu \leq b$$
$$\mu \in \{0,1\}^N$$



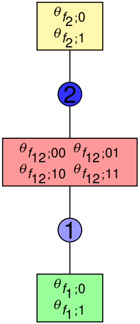
μ	$\mu_{1;0}$	$\mu_{1;1}$	$\mu_{2;0}$	$\mu_{2;1}$	$\mu_{f_1;0}$	$\mu_{f_1;1}$	$\mu_{f_2;0}$	$\mu_{f_2;1}$	$\mu_{f_{12};00}$	$\mu_{f_{12};01}$	$\mu_{f_{12};10}$	$\mu_{f_{12};11}$
θ	0	0	0	0	$\theta_{f_1;0}$	$\theta_{f_1;1}$	$\theta_{f_2;0}$	$\theta_{f_2;1}$	$\theta_{f_{12};00}$	$\theta_{f_{12};01}$	$\theta_{f_{12};10}$	$\theta_{f_{12};11}$

MAP inference as Integer Linear Program

$$\min_{\mu} \langle \theta, \mu \rangle$$

$$s.t. \quad A\mu \leq b$$

$$\mu \in \{0,1\}^N$$



μ	$\mu_{1;0}$	$\mu_{1;1}$	$\mu_{2;0}$	$\mu_{2;1}$	$\mu_{f_1;0}$	$\mu_{f_1;1}$	$\mu_{f_2;0}$	$\mu_{f_2;1}$	$\mu_{f_{12};00}$	$\mu_{f_{12};01}$	$\mu_{f_{12};10}$	$\mu_{f_{12};11}$
θ	0	0	0	0	$\theta_{f_1;0}$	$\theta_{f_1;1}$	$\theta_{f_2;0}$	$\theta_{f_2;1}$	$\theta_{f_{12};00}$	$\theta_{f_{12};01}$	$\theta_{f_{12};10}$	$\theta_{f_{12};11}$

$\forall v \in V$:

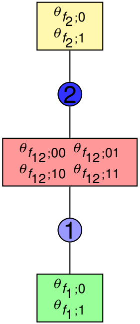
$$\sum_{x_v \in X_v} \mu_{v;x_c} = 1$$

$\forall f \in F, v \in ne(f)$:

$$\sum_{x_{ne(f)\setminus v} \in X_{ne(f)\setminus v}} \mu_{f;x_{ne(f)}} = \mu_{v;x_v}$$

MAP inference as Integer Linear Program

$$\min_{\mu} \langle \theta, \mu \rangle$$
$$s.t. \quad A\mu \leq b$$
$$\mu \in \{0,1\}^N$$



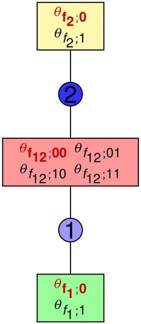
μ	$\mu_{1;0}$	$\mu_{1;1}$	$\mu_{2;0}$	$\mu_{2;1}$	$\mu_{f_1;0}$	$\mu_{f_1;1}$	$\mu_{f_2;0}$	$\mu_{f_2;1}$	$\mu_{f_{12};00}$	$\mu_{f_{12};01}$	$\mu_{f_{12};10}$	$\mu_{f_{12};11}$
θ	0	0	0	0	$\theta_{f_1;0}$	$\theta_{f_1;1}$	$\theta_{f_2;0}$	$\theta_{f_2;1}$	$\theta_{f_{12};00}$	$\theta_{f_{12};01}$	$\theta_{f_{12};10}$	$\theta_{f_{12};11}$

$\forall v \in V:$
$$\sum_{x_v \in X_v} \mu_{v;x_c} = 1 \qquad \overset{\mu \in \{0,1\}^N}{\Rightarrow} \qquad \sum_{x_v \in X_v} \mathbb{I}(\mu_{v;x_v} = 1) = 1$$

$\forall f \in F, v \in ne(f):$
$$\sum_{x_{ne(f)\setminus v} \in X_{ne(f)\setminus v}} \mu_{f;x_{ne(f)}} = \mu_{v;x_v} \qquad \overset{\mu \in \{0,1\}^N}{\Rightarrow} \qquad \mu_{f;x_{ne(f)}} = \prod_{v \in ne(f)} \mu_{v,x_v}$$

MAP inference as Integer Linear Program

$$\min_{\mu} \langle \theta, \mu \rangle$$
$$s.t. \quad A\mu \leq b$$
$$\mu \in \{0,1\}^N$$



$x_1 = 0, \quad x_2 = 0$

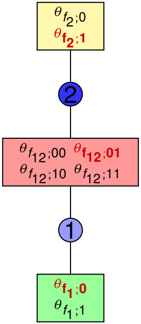
	1	0	**1**	0	**1**	0	**1**	0	**1**	0	0	0
μ	$\mu_{1;0}$	$\mu_{1;1}$	$\mu_{2;0}$	$\mu_{2;1}$	$\mu_{f_1;0}$	$\mu_{f_1;1}$	$\mu_{f_2;0}$	$\mu_{f_2;1}$	$\mu_{f_{12};00}$	$\mu_{f_{12};01}$	$\mu_{f_{12};10}$	$\mu_{f_{12};11}$
θ	0	0	0	0	$\theta_{f_1;0}$	$\theta_{f_1;1}$	$\theta_{f_2;0}$	$\theta_{f_2;1}$	$\theta_{f_{12};00}$	$\theta_{f_{12};01}$	$\theta_{f_{12};10}$	$\theta_{f_{12};11}$

$\forall v \in V$:
$$\sum_{x_v \in X_v} \mu_{v;x_c} = 1 \qquad \overset{\mu \in \{0,1\}^N}{\Rightarrow} \qquad \sum_{x_v \in X_v} \mathbb{I}(\mu_{v;x_v} = 1) = 1$$

$\forall f \in F, v \in ne(f)$:
$$\sum_{x_{ne(f)\setminus v} \in X_{ne(f)\setminus v}} \mu_{f;x_{ne(f)}} = \mu_{v;x_v} \qquad \overset{\mu \in \{0,1\}^N}{\Rightarrow} \qquad \mu_{f;x_{ne(f)}} = \prod_{v \in ne(f)} \mu_{v,x_v}$$

MAP inference as Integer Linear Program

$$\min_{\mu} \langle \theta, \mu \rangle$$
$$s.t. \quad A\mu \leq b$$
$$\mu \in \{0,1\}^N$$



$x_1 = 0, \quad x_2 = 1$

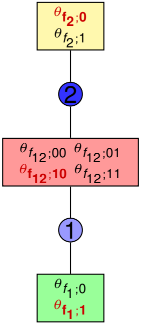
	1	0	0	**1**	**1**	0	0	**1**	0	**1**	0	0
μ	$\mu_{1;0}$	$\mu_{1;1}$	$\mu_{2;0}$	$\mu_{2;1}$	$\mu_{f_1;0}$	$\mu_{f_1;1}$	$\mu_{f_2;0}$	$\mu_{f_2;1}$	$\mu_{f_{12};00}$	$\mu_{f_{12};01}$	$\mu_{f_{12};10}$	$\mu_{f_{12};11}$
θ	0	0	0	0	$\theta_{\mathbf{f_1};\mathbf{0}}$	$\theta_{f_1;1}$	$\theta_{f_2;0}$	$\theta_{\mathbf{f_2};\mathbf{1}}$	$\theta_{f_{12};00}$	$\theta_{\mathbf{f_{12}};\mathbf{01}}$	$\theta_{f_{12};10}$	$\theta_{f_{12};11}$

$\forall v \in V:$
$$\sum_{x_v \in X_v} \mu_{v;x_c} = 1 \qquad \overset{\mu\in\{0,1\}^N}{\Rightarrow} \qquad \sum_{x_v \in X_v} \mathbb{I}(\mu_{v;x_v} = 1) = 1$$

$\forall f \in F, v \in ne(f):$
$$\sum_{x_{ne(f)\setminus v} \in X_{ne(f)\setminus v}} \mu_{f;x_{ne(f)}} = \mu_{v;x_v} \qquad \overset{\mu\in\{0,1\}^N}{\Rightarrow} \qquad \mu_{f;x_{ne(f)}} = \prod_{v\in ne(f)} \mu_{v,x_v}$$

MAP inference as Integer Linear Program

$$\min_{\mu} \langle \theta, \mu \rangle$$
$$s.t. \quad A\mu \leq b$$
$$\mu \in \{0,1\}^N$$



$x_1 = 1, \quad x_2 = 0$

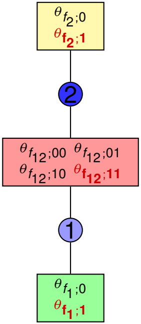
	0	1	1	0	0	1	1	0	0	0	1	0
μ	$\mu_{1;0}$	$\mu_{1;1}$	$\mu_{2;0}$	$\mu_{2;1}$	$\mu_{f_1;0}$	$\mu_{f_1;1}$	$\mu_{f_2;0}$	$\mu_{f_2;1}$	$\mu_{f_{12};00}$	$\mu_{f_{12};01}$	$\mu_{f_{12};10}$	$\mu_{f_{12};11}$
θ	0	0	0	0	$\theta_{f_1;0}$	$\theta_{f_1;1}$	$\theta_{f_2;0}$	$\theta_{f_2;1}$	$\theta_{f_{12};00}$	$\theta_{f_{12};01}$	$\theta_{f_{12};10}$	$\theta_{f_{12};11}$

$\forall v \in V$:
$$\sum_{x_v \in X_v} \mu_{v;x_c} = 1 \quad \overset{\mu \in \{0,1\}^N}{\Rightarrow} \quad \sum_{x_v \in X_v} \mathbb{I}(\mu_{v;x_v} = 1) = 1$$

$\forall f \in F, v \in ne(f)$:
$$\sum_{x_{ne(f)\setminus v} \in X_{ne(f)\setminus v}} \mu_{f;x_{ne(f)}} = \mu_{v;x_v} \quad \overset{\mu \in \{0,1\}^N}{\Rightarrow} \quad \mu_{f;x_{ne(f)}} = \prod_{v \in ne(f)} \mu_{v,x_v}$$

MAP inference as Integer Linear Program

$$\min_{\mu} \langle \theta, \mu \rangle$$
$$s.t. \quad A\mu \leq b$$
$$\mu \in \{0,1\}^N$$



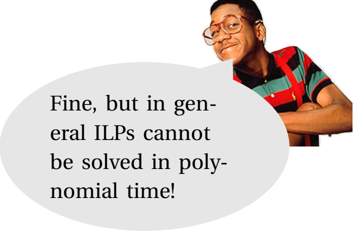
$x_1 = 1, \quad x_2 = 1$

	0	**1**	0	**1**	0	**1**	0	**1**	0	0	0	**1**
μ	$\mu_{1;0}$	$\mu_{1;1}$	$\mu_{2;0}$	$\mu_{2;1}$	$\mu_{f_1;0}$	$\mu_{f_1;1}$	$\mu_{f_2;0}$	$\mu_{f_2;1}$	$\mu_{f_{12};00}$	$\mu_{f_{12};01}$	$\mu_{f_{12};10}$	$\mu_{f_{12};11}$
θ	0	0	0	0	$\theta_{f_1;0}$	$\theta_{f_1;1}$	$\theta_{f_2;0}$	$\theta_{f_2;1}$	$\theta_{f_{12};00}$	$\theta_{f_{12};01}$	$\theta_{f_{12};10}$	$\theta_{f_{12};11}$

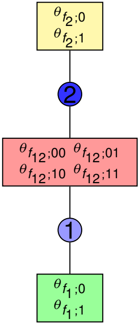
$\forall v \in V:$
$$\sum_{x_v \in X_v} \mu_{v;x_c} = 1 \qquad \overset{\mu \in \{0,1\}^N}{\Rightarrow} \qquad \sum_{x_v \in X_v} \mathbb{I}(\mu_{v;x_v} = 1) = 1$$

$\forall f \in F, v \in ne(f):$
$$\sum_{x_{ne(f)\setminus v} \in X_{ne(f)\setminus v}} \mu_{f;x_{ne(f)}} = \mu_{v;x_v} \qquad \overset{\mu \in \{0,1\}^N}{\Rightarrow} \qquad \mu_{f;x_{ne(f)}} = \prod_{v \in ne(f)} \mu_{v,x_v}$$

MAP inference as Integer Linear Program



$$\min_{\mu} \langle \theta, \mu \rangle$$
$$s.t. \quad A\mu \leq b$$
$$\mu \in \{0,1\}^N$$



$x_1 = 1, \quad x_2 = 1$

	0	1	0	1	0	1	0	1	0	0	0	1
μ	$\mu_{1;0}$	$\mu_{1;1}$	$\mu_{2;0}$	$\mu_{2;1}$	$\mu_{f_1;0}$	$\mu_{f_1;1}$	$\mu_{f_2;0}$	$\mu_{f_2;1}$	$\mu_{f_{12};00}$	$\mu_{f_{12};01}$	$\mu_{f_{12};10}$	$\mu_{f_{12};11}$
θ	0	0	0	0	$\theta_{f_1;0}$	$\theta_{f_1;1}$	$\theta_{f_2;0}$	$\theta_{f_2;1}$	$\theta_{f_{12};00}$	$\theta_{f_{12};01}$	$\theta_{f_{12};10}$	$\theta_{f_{12};11}$

$\forall v \in V :$
$$\textstyle\sum_{x_v \in X_v} \mu_{v;x_c} = 1 \quad \overset{\mu \in \{0,1\}^N}{\Rightarrow} \quad \sum_{x_v \in X_v} \mathbb{I}(\mu_{v;x_v} = 1) = 1$$

$\forall f \in F, v \in ne(f) :$
$$\textstyle\sum_{x_{ne(f)\setminus v} \in X_{ne(f)\setminus v}} \mu_{f;x_{ne(f)}} = \mu_{v;x_v} \quad \overset{\mu \in \{0,1\}^N}{\Rightarrow} \quad \mu_{f;x_{ne(f)}} = \prod_{v \in ne(f)} \mu_{v,x_v}$$

MAP inference as Integer Linear Program

Reasons why ILPs had been ignored

- Worst case complexity is exponential
- ILPs are often very memory consuming
- Good ILP-solvers are expensive

MAP inference as Integer Linear Program

Reasons why ILPs had been ignored

- Worst case complexity is exponential
- ILPs are often very memory consuming
- Good ILP-solvers are expensive

Reasons why ILPs are Relevant

- Worst case complexity does not always matter!
- Highly optimized commercial solvers (e.g. CPLEX, Gurobi) are freely available for academic use.
- ILPs can compute the global optimal solution.
- No limitations on the model . . . if we ignore memory requirements.
- Always a good baseline for small problems.

MAP inference as Integer Linear Program

Reasons why ILPs had been ignored

- Worst case complexity is exponential
- ILPs are often very memory consuming
- Good ILP-solvers are expensive

Reasons why ILPs are Relevant

- Worst case complexity does not always matter!
- Highly optimized commercial solvers (e.g. CPLEX, Gurobi) are freely available for academic use.
- ILPs can compute the global optimal solution.
- No limitations on the model . . . if we ignore memory requirements.
- Always a good baseline for small problems.

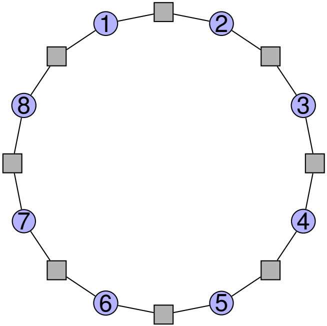
How ILP-solvers works

- Branch and Bound
- Searching for generic constraints
- The *black magic* is to combine all of this

Finally some Simple Tricks ...

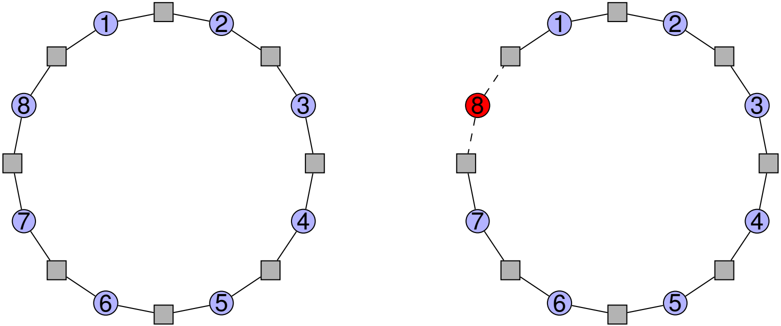
Tricks to make Inference more Tractable

Ring



Tricks to make Inference more Tractable

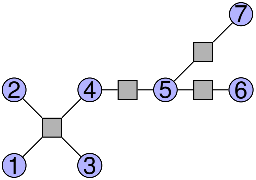
Ring



- Solve the acyclic problem for each possible labeling of x_8.
- Select the best solution from all these problems.

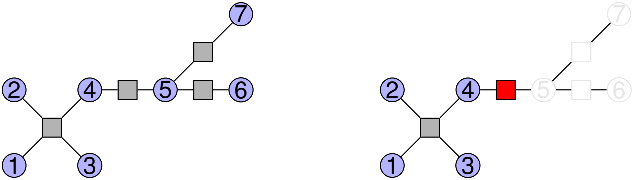
Tricks to make Inference more Tractable

Partially Acyclic Graph



Tricks to make Inference more Tractable

Partially Acyclic Graph

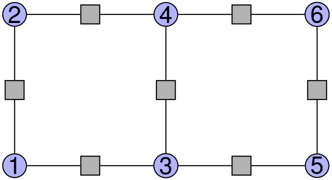


- Presolve acyclic substructures by dynamic programming.
- Solve core problem, with acyclic part replaced by unary factor
- Calculate labeling for acyclic part as for dynamic programming given the solution of the core problem.

[Kappes et al.*, 2013]*

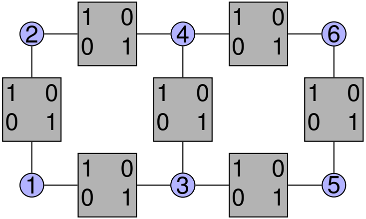
Tricks to make Inference more Tractable

Permuted Submodular



Tricks to make Inference more Tractable

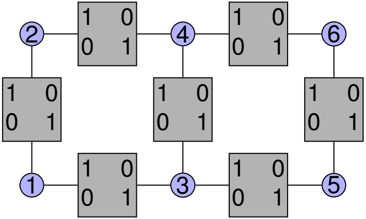
Permuted Submodular



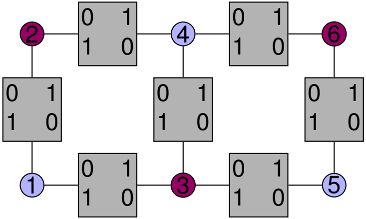
not regular $\stackrel{?}{\rightarrow}$ not submodular

Tricks to make Inference more Tractable

Permuted Submodular



not regular $\overset{?}{\rightarrow}$ not submodular

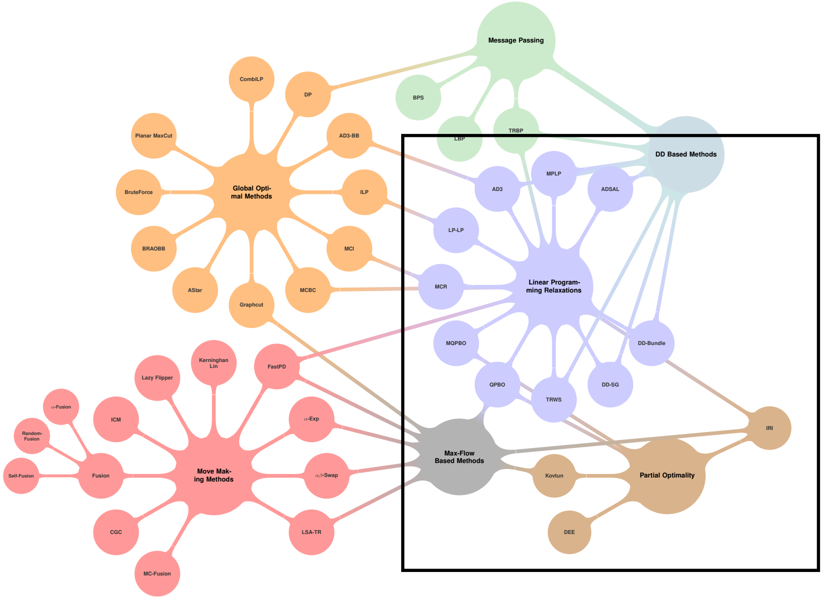


regular $\rightarrow$ submodular

- **The reformulation is variant to label permutation $\rightarrow$ the order of labels matters!**
- Reorder labels
- Solve max-flow or submodular-minimization problem
- Undo reorder for solution

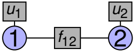
[Schlesinger, 2007, Swoboda et al., 2013]

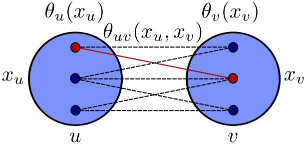
Inference Methods based on Relaxations


Message Passing
BPS
LBP
TRBP
DD Based Methods
CombiLP
DP
Planar MaxCut
AD3-BB
BruteForce
Global Optimal Methods
ILP
BRAOBB
MCI
AStar
Graphcut
MCBC
AD3
MPLP
ADSAL
LP-LP
MCR
Linear Programming Relaxations
MQPBO
DD-Bundle
QPBO
TRWS
DD-SG
IRI
Kerninghan Lin
Lazy Flipper
FastPD
α-Fusion
ICM
α-Exp
Random-Fusion
Self-Fusion
Fusion
Move Making Methods
αβ-Swap
Max-Flow Based Methods
Kovtun
Partial Optimality
CGC
LSA-TR
DEE
MC-Fusion

MAP Inference as Integer Linear Program

$$\text{(arg)}\min_{x\in\mathcal{X}} \sum_{v\in V} \theta_v(x_v) + \sum_{uv\in E} \theta_{uv}(x_u, x_v)$$

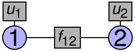


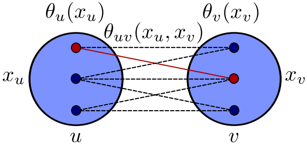


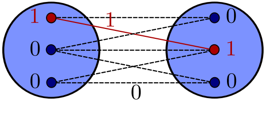
MAP Inference as Integer Linear Program

$$(\text{arg})\min_{x\in\mathcal{X}}\sum_{v\in V}\theta_v(x_v)+\sum_{uv\in E}\theta_{uv}(x_u,x_v) \;=\; (\text{arg})\min_{x\in\mathcal{X}}\langle\theta,\delta(x)\rangle$$

$$\delta(x) = \underbrace{100}_{u}\,\underbrace{010000000}_{uv}\,\underbrace{010}_{v}$$



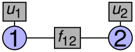


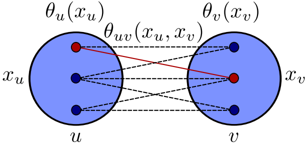


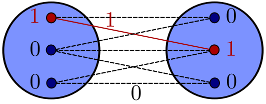
MAP Inference as Integer Linear Program

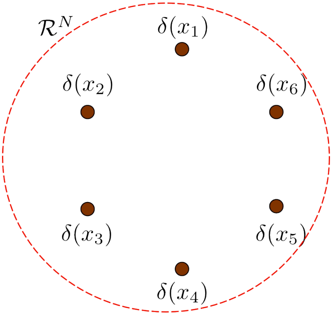
$$(\text{arg})\min_{x\in\mathcal{X}} \sum_{v\in V}\theta_v(x_v)+\sum_{uv\in E}\theta_{uv}(x_u,x_v) \;=\; (\text{arg})\min_{x\in\mathcal{X}} \langle\theta,\delta(x)\rangle$$

$$\delta(x) = \underbrace{100}_{u}\,\underbrace{010000000}_{uv}\,\underbrace{010}_{v}$$







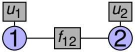


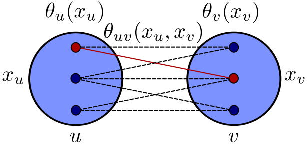
$$(\text{arg})\quad \min_{x\in\mathcal{X}} \langle\theta,\delta(x)\rangle$$

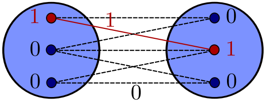
MAP Inference as Integer Linear Program

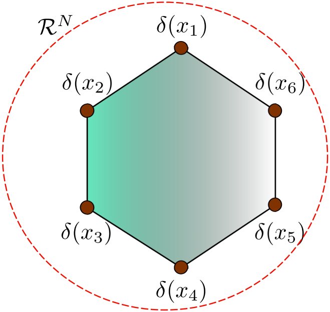
$$\text{(arg)}\min_{x\in\mathcal{X}} \sum_{v\in V} \theta_v(x_v) + \sum_{uv\in E} \theta_{uv}(x_u,x_v) \quad = \quad \text{(arg)}\min_{x\in\mathcal{X}} \langle \theta, \delta(x)\rangle$$

$$\delta(x) = \underbrace{100}_{u}\,\underbrace{010000000}_{uv}\,\underbrace{010}_{v}$$







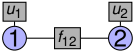


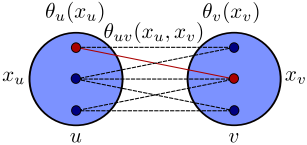
$$\text{(arg)}\min_{\mu\in\text{conv}(\mathcal{X})} \langle \theta, \mu\rangle$$

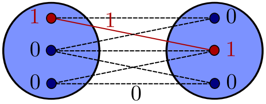
MAP Inference as Integer Linear Program

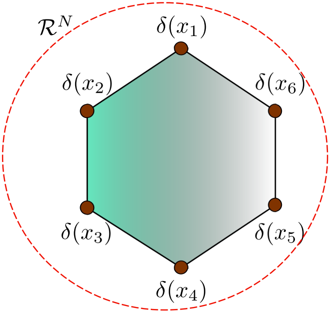
$$\text{(arg)}\min_{x\in\mathcal{X}} \sum_{v\in V} \theta_v(x_v) + \sum_{uv\in E} \theta_{uv}(x_u, x_v) \quad = \quad \text{(arg)}\min_{x\in\mathcal{X}} \langle \theta, \delta(x) \rangle$$

$$\delta(x) = \underbrace{100}_{u}\ \underbrace{010000000}_{uv}\ \underbrace{010}_{v}$$





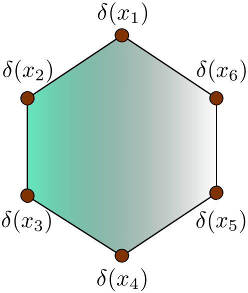




$$\text{(arg)}\min_{\mu\in\text{conv}(\delta(\mathcal{X}))} \langle \theta, \mu \rangle$$

$$\text{conv}(\delta(\mathcal{X})) \Rightarrow A\mu \geq b$$

Relaxed MAP Inference



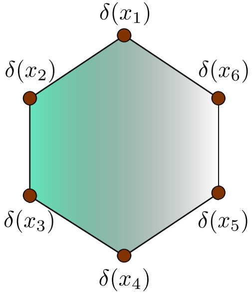
$$\min_{\mu} \langle \theta, \mu \rangle$$

$$\text{s.t. } \mu = \sum_{x \in \mathcal{X}} \alpha_x \delta(x)$$

$$\sum_{x \in \mathcal{X}} \alpha_x = 1$$

$$\alpha_x \geq 0, \; x \in \mathcal{X}$$

Relaxed MAP Inference



$$\min_{\mu} \langle \theta, \mu \rangle$$

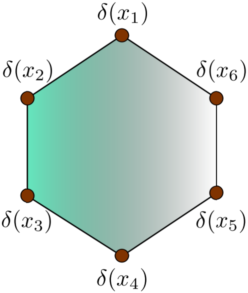
$$\text{s.t. } \mu = \sum_{x \in \mathcal{X}} \alpha_x \delta(x)$$

$$\sum_{x \in \mathcal{X}} \alpha_x = 1$$

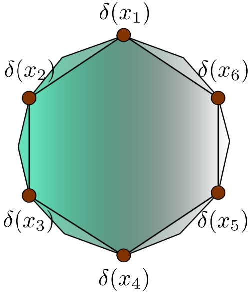
$$\alpha_x \geq 0, \; x \in \mathcal{X}$$

Exponentially many constraints!

Relaxed MAP Inference



Relaxation $\Rightarrow$



$$\min_{\mu} \langle \theta, \mu \rangle$$

$$\text{s.t. } \mu = \sum_{x \in \mathcal{X}} \alpha_x \delta(x)$$

$$\sum_{x \in \mathcal{X}} \alpha_x = 1$$

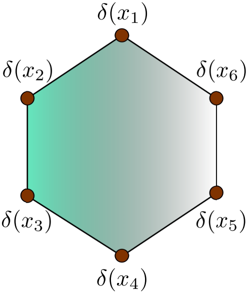
$$\alpha_x \geq 0, \; x \in \mathcal{X}$$

Exponentially many constraints!

$$\min_{\mu} \langle \theta, \mu \rangle$$

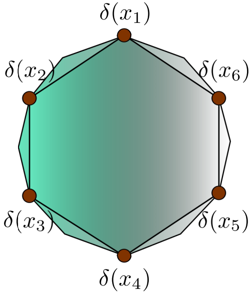
s.t. $A\mu \geq b$, A is small!

Relaxed MAP Inference



Relaxation

$\Rightarrow$



$$\min_{\mu} \langle \theta, \mu \rangle$$

$$\text{s.t. } \mu = \sum_{x \in \mathcal{X}} \alpha_x \delta(x)$$

$$\sum_{x \in \mathcal{X}} \alpha_x = 1$$

$$\alpha_x \geq 0, \ x \in \mathcal{X}$$

Exponentially many constraints!

$$\min_{\mu} \langle \theta, \mu \rangle$$

s.t. $A\mu \geq b$, A is small!

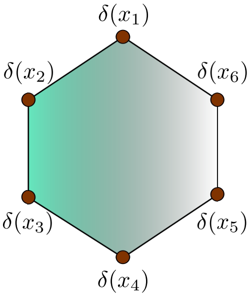
$$\forall v \in V: \quad \sum_{x_v \in X_v} \mu_{v;x_c} = 1$$

$$\forall f \in F, v \in ne(f):$$

$$\sum_{x_{ne(f)\setminus v} \in X_{ne(f)\setminus v}} \mu_{f;x_{ne(f)}} = \mu_{v;x_v}$$

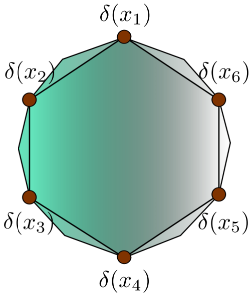
$$\cancel{\mu \in \{0,1\}^N} \ \mu \in [0,1]^N$$

Relaxed MAP Inference



Relaxation

$\Rightarrow$



$$\min_{\mu} \langle \theta, \mu \rangle$$

$$\text{s.t. } \mu = \sum\nolimits_{x \in \mathcal{X}} \alpha_x \delta(x)$$

$$\sum\nolimits_{x \in \mathcal{X}} \alpha_x = 1$$

$$\alpha_x \geq 0, \; x \in \mathcal{X}$$

Exponentially many constraints!

$$\min_{\mu} \langle \theta, \mu \rangle$$

s.t. $A\mu \geq b$, A is small!

$$\forall v \in V : \quad \sum\nolimits_{x_v \in X_v} \mu_{v;x_c} = 1$$

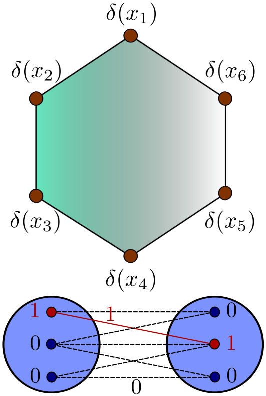
$$\forall f \in F, v \in ne(f) :$$

$$\sum\nolimits_{x_{ne(f)\setminus v} \in X_{ne(f)\setminus v}} \mu_{f;x_{ne(f)}} = \mu_{v;x_v}$$

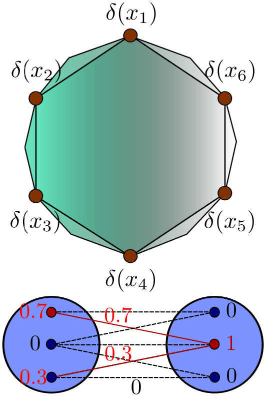
$$\cancel{\mu \in \{0,1\}^N} \; \mu \in [0,1]^N$$

μ - non-relaxed solution $\Leftarrow$ μ - relaxed solution

Relaxed MAP Inference



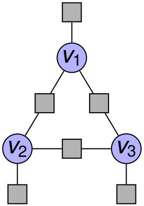
Relaxation
$\Rightarrow$



$\mu \in \{0, 1\}^N$

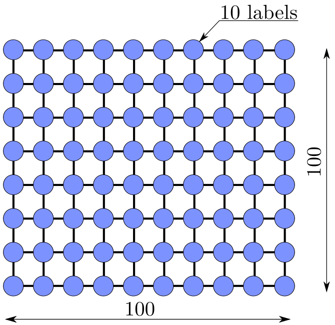
$\mu \in [0, 1]^N$

Local Polytope Complexity



$\lvert L\rvert$	$\lvert X\rvert$	vertices in $\mathbb{L}(M)$
2	8	12
3	27	207
4	64	8.992
5	125	853.725

Why dedicated solvers needed for the relaxed inference?

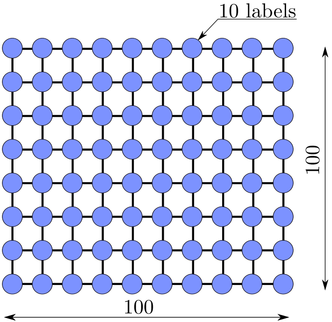


Pascal VOC 2012
semantic segmentation
model $\approx 500 \times 300 \times 21$ labels

$2 \cdot 10^6$ variables

$> 10^9$ variables

Why dedicated solvers needed for the relaxed inference?



$2 \cdot 10^6$ variables

Pascal VOC 2012
semantic segmentation
model $\approx 500 \times 300 \times 21$ labels

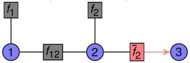
$> 10^9$ variables

Standard LP solvers (simplex, interior point) do not scale well!

Lagrangean (Dual) Decomposition

Certain problems are easily solvable
(e.g. acyclic with dynamic programming):

Optimization by Dynamic Programming

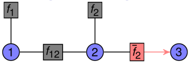


$$\min_{x_1} \varphi_{f_1}(x_1) + \min_{x_2} \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_2}(x_2) + \varphi_{\bar{f}_2}(x_2)$$

Lagrangean (Dual) Decomposition

Certain problems are easily solvable
(e.g. acyclic with dynamic programming):

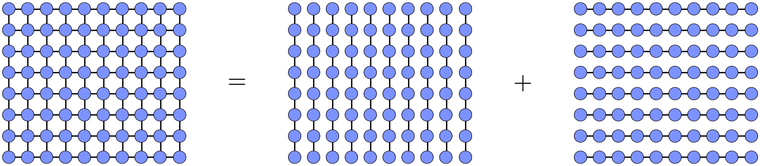
Optimization by Dynamic Programming



$$\min_{x_1} \varphi_{f_1}(x_1) + \min_{x_2} \varphi_{f_{12}}(x_1, x_2) + \varphi_{f_2}(x_2) + \varphi_{\bar{f}_2}(x_2)$$

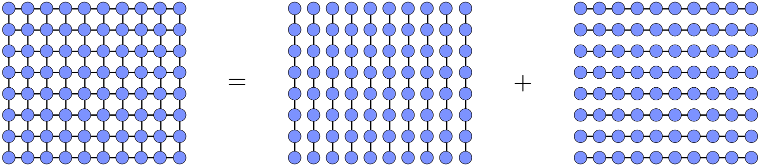
What about decomposing the problem
into solvable subproblems?

Lagrangean (Dual) Decomposition

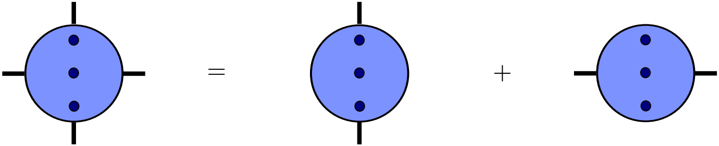


$$E(\theta, x) \quad = \quad E(\theta^c, x) \quad + \quad E(\theta^r, x)$$

Lagrangean (Dual) Decomposition

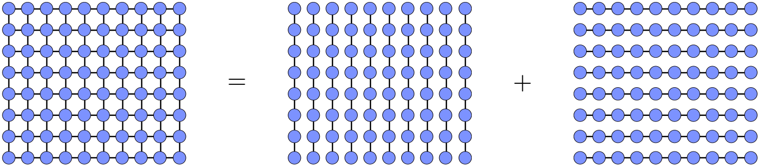


$$E(\theta, x) = E(\theta^c, x) + E(\theta^r, x)$$

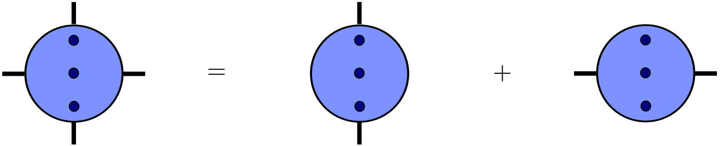


$$\theta_v(x_v) = \frac{\theta_v(x_v)}{2} + \lambda_v(x_v) + \frac{\theta_v(x_v)}{2} - \lambda_v(x_v)$$

Lagrangean (Dual) Decomposition



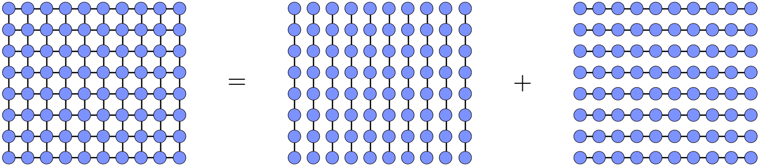
$$E(\theta, x) \quad = \quad E(\theta^c, x) \quad + \quad E(\theta^r, x)$$



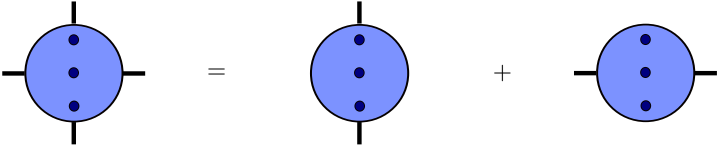
$$\theta_v(x_v) \quad = \quad \frac{\theta_v(x_v)}{2} + \lambda_v(x_v) \quad + \quad \frac{\theta_v(x_v)}{2} - \lambda_v(x_v)$$

$$\min_{x \in \mathcal{X}} E(\theta, x) \quad \overset{\forall \lambda}{\geq} \quad \min_{x^c \in \mathcal{X}} E(\theta^c(\lambda), x^c) \quad + \quad \min_{x^r \in \mathcal{X}} E(\theta^r(\lambda), x^r)$$

Lagrangean (Dual) Decomposition



$$E(\theta, x) = E(\theta^c, x) + E(\theta^r, x)$$



$$\theta_v(x_v) = \frac{\theta_v(x_v)}{2} + \lambda_v(x_v) + \frac{\theta_v(x_v)}{2} - \lambda_v(x_v)$$

$$\min_{x \in \mathcal{X}} E(\theta, x) \geq \max_{\lambda} \left[\min_{x^c \in \mathcal{X}} E(\theta^c(\lambda), x^c) + \min_{x^r \in \mathcal{X}} E(\theta^r(\lambda), x^r) \right]$$

Lower Bound Optimization

$$\theta^c = \frac{\theta_v}{2} + \lambda_v; \quad \theta^r = \frac{\theta_v}{2} - \lambda_v$$

$$\max_{\lambda} \left[\min_{x^c \in \mathcal{X}} E(\theta^c(\lambda), x^c) + \min_{x^r \in \mathcal{X}} E(\theta^r(\lambda), x^r) \right]$$

Lower Bound Optimization

$$\theta^c = \frac{\theta_v}{2} + \lambda_v; \quad \theta^r = \frac{\theta_v}{2} - \lambda_v$$

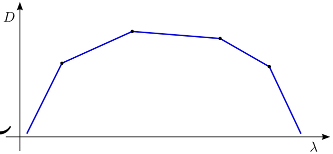
$$\max_{\lambda} \left[\min_{x^c \in \mathcal{X}} \underbrace{E(\theta^c(\lambda), x^c)}_{\text{linear in } \lambda} + \min_{x^r \in \mathcal{X}} \underbrace{E(\theta^r(\lambda), x^r)}_{\text{linear in } \lambda} \right]$$

Lower Bound Optimization

$$\theta^c = \frac{\theta_v}{2} + \lambda_v; \quad \theta^r = \frac{\theta_v}{2} - \lambda_v$$

$$\underbrace{\max_{\lambda} \left[\min_{x^c \in \mathcal{X}} \underbrace{E(\theta^c(\lambda), x^c)}_{\text{linear in } \lambda} + \min_{x^r \in \mathcal{X}} \underbrace{E(\theta^r(\lambda), x^r)}_{\text{linear in } \lambda} \right]}_{D(\lambda)}$$

piece-wise linear, concave

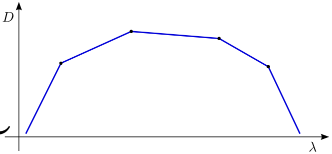


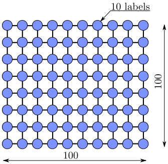
Lower Bound Optimization

$$\theta^c = \frac{\theta_v}{2} + \lambda_v; \quad \theta^r = \frac{\theta_v}{2} - \lambda_v$$

$$\underbrace{\max_{\lambda} \left[\min_{x^c \in \mathcal{X}} \underbrace{E(\theta^c(\lambda), x^c)}_{\text{linear in } \lambda} + \min_{x^r \in \mathcal{X}} \underbrace{E(\theta^r(\lambda), x^r)}_{\text{linear in } \lambda} \right]}_{D(\lambda)}$$

piece-wise linear, concave





variables: $10^5 \ll 10^6$

Pascal VOC

$500 \times 300 \times 21$

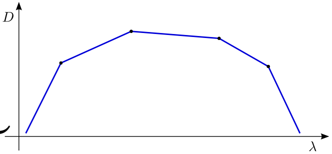
$3 \cdot 10^6 \ll 10^9$

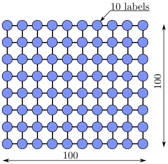
Lower Bound Optimization

$$\theta^c = \frac{\theta_v}{2} + \lambda_v; \quad \theta^r = \frac{\theta_v}{2} - \lambda_v$$

$$\underbrace{\max_{\lambda} \left[\min_{x^c \in \mathcal{X}} \underbrace{E(\theta^c(\lambda), x^c)}_{\text{linear in } \lambda} + \min_{x^r \in \mathcal{X}} \underbrace{E(\theta^r(\lambda), x^r)}_{\text{linear in } \lambda} \right]}_{D(\lambda)}$$

piece-wise linear, concave





\# variables: $10^5 \ll 10^6$

Pascal VOC

$500 \times 300 \times 21$

$3 \cdot 10^6 \ll 10^9$

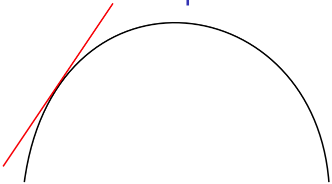
Optimization:

- convex
- non-smooth
- large-scale

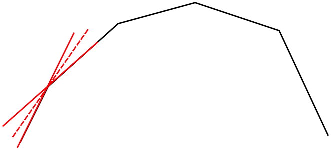
First-Order Convex Optimization

update rule	subproblem	rate	note
sub-gradient	MAP-inference	$O(\frac{1}{\epsilon^2})$	step-size selection
mirror-descent	MAP-inference	$O(\frac{1}{\epsilon^2})$	
bundle	MAP-inference	$O(\frac{1}{\epsilon^2})$	additional QP
coord. ascent	min-marginals	unknown	no optimum guarantee
smooth coord. ascent	probab.-marginals	unknown	exp. operation
smooth acc. ascent	marginalization	$O(\frac{1}{\epsilon})$	exp. operation
proximal (e.g. ADMM)	proximal inference	$O(\frac{1}{\epsilon})$	

First-Order Convex Optimization: Subgradient Method

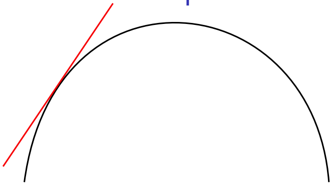


gradient ∇D



subgradient ∂D

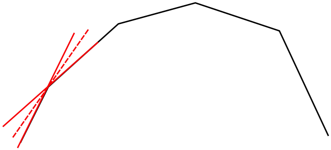
First-Order Convex Optimization: Subgradient Method



gradient ∇D

Gradient ascent:

$$\lambda^{t+1} = \lambda^{t} + \tau \nabla D$$

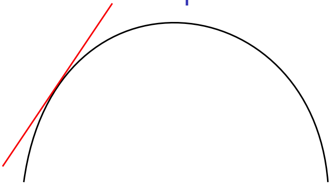


subgradient ∂D

Subgradient method:

$$\lambda^{t+1} = \lambda^{t} + \tau^{t} \partial D$$

First-Order Convex Optimization: Subgradient Method

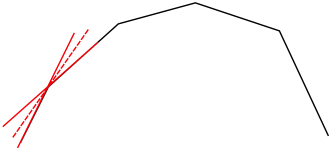


gradient ∇D

Gradient ascent:

$$\lambda^{t+1} = \lambda^t + \tau \, \nabla D$$

$$L \geq \tau > 0$$



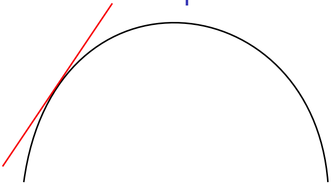
subgradient ∂D

Subgradient method:

$$\lambda^{t+1} = \lambda^t + \tau^t \partial D$$

$$\tau^t > 0, \tau^t \to 0, \; \sum_{t=1}^{\infty} \tau^t = \infty$$

First-Order Convex Optimization: Subgradient Method

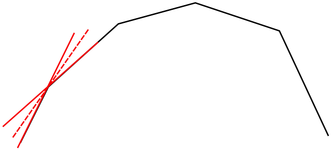


gradient ∇D

Gradient ascent:

$$\lambda^{t+1} = \lambda^t + \tau\,\nabla D$$

$$L \geq \tau > 0$$



subgradient ∂D

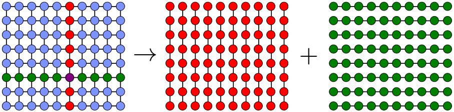
Subgradient method:

$$\lambda^{t+1} = \lambda^t + \tau^t \partial D$$

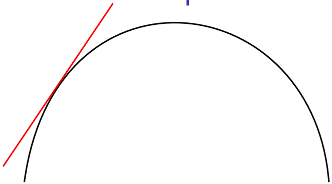
$$\tau^t > 0, \tau^t \to 0, \ \textstyle\sum_{t=1}^{\infty} \tau^t = \infty$$

$$D(\lambda) = \min_{x^c} \left\langle \frac{\theta}{2} + \lambda, \delta(x^c) \right\rangle + \min_{x^r} \left\langle \frac{\theta}{2} - \lambda, \delta(x^r) \right\rangle$$

$$\Rightarrow \partial D(\lambda) = \delta(x^{*c}) - \delta(x^{*r})$$



First-Order Convex Optimization: Subgradient Method

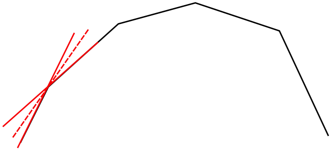


gradient ∇D

Gradient ascent:

$$\lambda^{t+1} = \lambda^t + \tau \nabla D$$

$$L \geq \tau > 0$$



subgradient ∂D

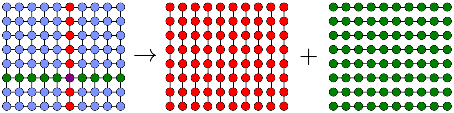
Subgradient method:

$$\lambda^{t+1} = \lambda^t + \tau^t \partial D$$

$$\tau^t > 0, \tau^t \to 0, \ \sum_{t=1}^{\infty} \tau^t = \infty$$

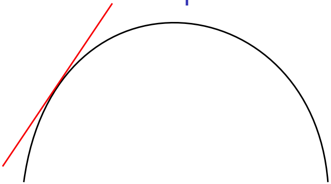
$$D(\lambda) = \min_{x^c} \left\langle \frac{\theta}{2} + \lambda, \delta(x^c) \right\rangle + \min_{x^r} \left\langle \frac{\theta}{2} - \lambda, \delta(x^r) \right\rangle$$

$$\Rightarrow \partial D(\lambda) = \delta(x^{*c}) - \delta(x^{*r})$$



+ based on MAP inference

First-Order Convex Optimization: Subgradient Method

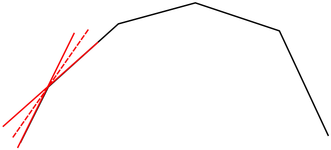


gradient ∇D

Gradient ascent:

$$\lambda^{t+1} = \lambda^t + \tau \, \nabla D$$

$$L \geq \tau > 0$$



subgradient ∂D

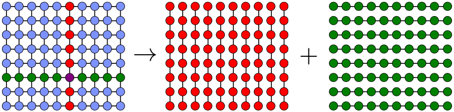
Subgradient method:

$$\lambda^{t+1} = \lambda^t + \tau^t \partial D$$

$$\tau^t > 0, \tau^t \to 0, \; \sum_{t=1}^{\infty} \tau^t = \infty$$

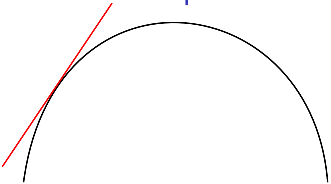
$$D(\lambda) = \min_{x^c} \left\langle \frac{\theta}{2} + \lambda, \delta(x^c) \right\rangle + \min_{x^r} \left\langle \frac{\theta}{2} - \lambda, \delta(x^r) \right\rangle$$

$$\Rightarrow \partial D(\lambda) = \delta(x^{*c}) - \delta(x^{*r})$$



- \+ based on MAP inference
- \- sensible to τ^t

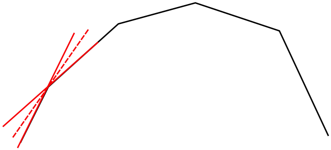
First-Order Convex Optimization: Subgradient Method



gradient ∇D

Gradient ascent:

$$\lambda^{t+1} = \lambda^t + \tau\, \nabla D$$
$$L \geq \tau > 0$$



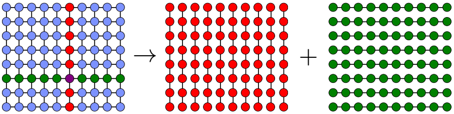
subgradient ∂D

Subgradient method:

$$\lambda^{t+1} = \lambda^t + \tau^t \partial D$$
$$\tau^t > 0, \tau^t \to 0,\ \textstyle\sum_{t=1}^{\infty} \tau^t = \infty$$

$$D(\lambda) = \min_{x^c} \left\langle \frac{\theta}{2} + \lambda, \delta(x^c) \right\rangle + \min_{x^r} \left\langle \frac{\theta}{2} - \lambda, \delta(x^r) \right\rangle$$

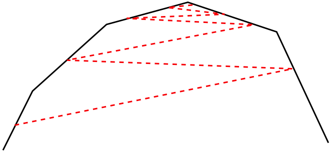
$$\Rightarrow \partial D(\lambda) = \delta(x^{*c}) - \delta(x^{*r})$$



- \+ based on MAP inference
- \- sensible to τ^t
- \- slow: converges as $O(\frac{1}{\epsilon^2})$
  $\epsilon = 0.1 \Rightarrow t = 100$
  $\epsilon = 0.01 \Rightarrow t = 10000$

First-Order Convex Optimization: Bundle Method

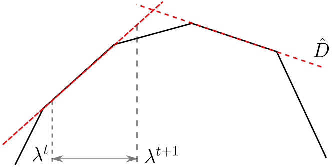
Subgradient:



$$\lambda^{t+1} = \lambda^t + \tau^t \partial D$$
$$\tau^t \rightarrow 0$$

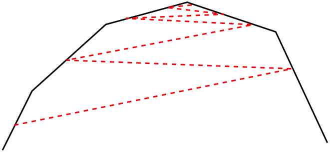
First-Order Convex Optimization: Bundle Method

Bundle:



$$\lambda^{t+1} = \arg\min_{\lambda} \widehat{D}(\lambda) + \gamma^t \|\lambda - \lambda^t\|^2$$
$$\gamma^t \to 0$$

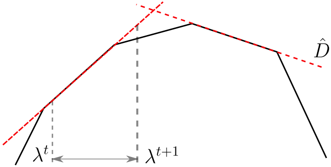
Subgradient:



$$\lambda^{t+1} = \lambda^t + \tau^t \partial D$$
$$\tau^t \to 0$$

First-Order Convex Optimization: Bundle Method

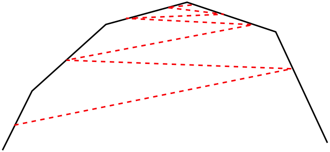
Bundle:



$$\lambda^{t+1} = \arg\min_\lambda \widehat{D}(\lambda) + \gamma^t \|\lambda - \lambda^t\|^2$$
$$\gamma^t \to 0$$

[Kappes et al.*, 2012]*

Subgradient:



$$\lambda^{t+1} = \lambda^t + \tau^t \partial D$$
$$\tau^t \to 0$$

[Storvik and Dahl, 2000, Schlesinger and Giginyak, 2007, Komodakis et al.*, 2011]*

+ based on MAP inference
+ less sensible to γ^t
+ much faster convergence in practice
- can be slow: worst-case complexity $O(\frac{1}{\epsilon^2})$

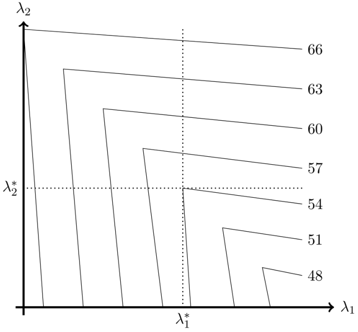
First-Order Convex Optimization: Coordinate Ascent

$$\lambda_i^{t+1} = \arg\min_{\lambda_i} D(\lambda_1^t, \ldots, \lambda_i, \ldots, \lambda_N^t)$$
$$i = 1, \ldots, N$$

First-Order Convex Optimization: Coordinate Ascent

$$\lambda_i^{t+1} = \arg\min_{\lambda_i} D(\lambda_1^t, \ldots, \lambda_i, \ldots, \lambda_N^t)$$

$$i = 1, \ldots, N$$

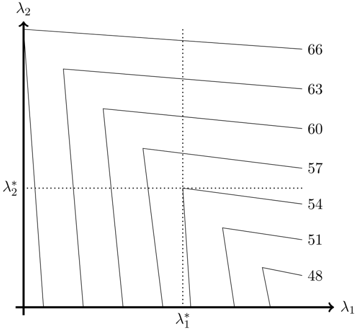


First-Order Convex Optimization: Coordinate Ascent

$$\lambda_i^{t+1} = \arg\min_{\lambda_i} D(\lambda_1^t, \ldots, \lambda_i, \ldots, \lambda_N^t)$$

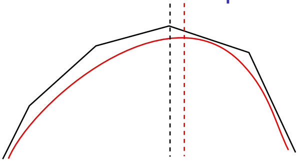
$$i = 1, \ldots, N$$

+ requires MAP - inference (min-marginals)
+ can be very efficiently implemented
- can get stuck
- convergence rate is unknown

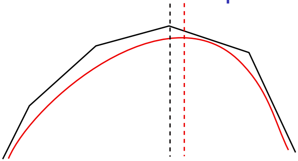


Inference algorithms:
TRW-S: *[Kolmogorov, 2006]*
SRMP: *[Kolmogorov, 2015]*
Max-Sum-Diffusion (MSD): *[Schlesinger and Antoniuk, 2011, Werner, 2007]*
MPLP: *[Globerson and Jaakkola, 2007]*
Norm-Product BP (NPBP): *[Hazan and Shashua, 2010]*

First-Order Convex Optimization: Smoothing



First-Order Convex Optimization: Smoothing

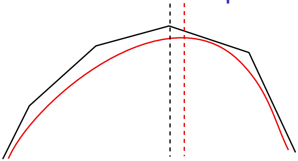


min $\to$ "soft" min

$\min_{x^c} E^c(x^c) \to -T \log \sum_{x^c} \exp(-E^c(x^c)/T)$

MAP-inf. $\to$ Probabilistic inf.

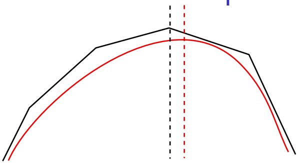
First-Order Convex Optimization: Smoothing



min $\rightarrow$ "soft" min
$\min_{x^c} E^c(x^c) \rightarrow -T \log \sum_{x^c} \exp(-E^c(x^c)/T)$
MAP-inf. $\rightarrow$ Probabilistic inf.

+ accelerated gradient ascent converges as $O(\frac{1}{\epsilon})$
  $\epsilon = 0.1 \Rightarrow t = 10$
  $\epsilon = 0.01 \Rightarrow t = 100$

First-Order Convex Optimization: Smoothing



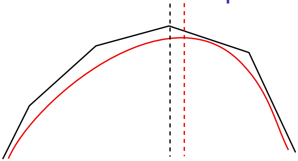
min $\rightarrow$ "soft" min

$\min_{x^c} E^c(x^c) \rightarrow -T \log \sum_{x^c} \exp(-E^c(x^c)/T)$

MAP-inf. $\rightarrow$ Probabilistic inf.

+ accelerated gradient ascent converges as $O(\frac{1}{\epsilon})$
  $\epsilon = 0.1 \Rightarrow t = 10$
  $\epsilon = 0.01 \Rightarrow t = 100$
+ coordinate ascent does not gut stuck!

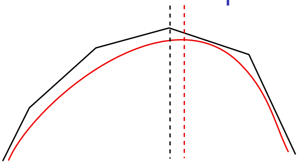
First-Order Convex Optimization: Smoothing



min $\rightarrow$ "soft" min
$\min_{x^c} E^c(x^c) \rightarrow -T \log \sum_{x^c} \exp(-E^c(x^c)/T)$
MAP-inf. $\rightarrow$ Probabilistic inf.

- \+ accelerated gradient ascent converges as $O(\frac{1}{\epsilon})$
  $\epsilon = 0.1 \Rightarrow t = 10$
  $\epsilon = 0.01 \Rightarrow t = 100$
- \+ coordinate ascent does not gut stuck!
- +/- Requires probabilistic inference: tractable for acyclic subproblems

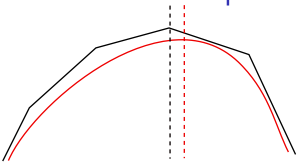
First-Order Convex Optimization: Smoothing



min	$\rightarrow$	"soft" min
$\min_{x^c} E^c(x^c)$	$\rightarrow$	$-T \log \sum_{x^c} \exp(-E^c(x^c)/T)$
MAP-inf.	$\rightarrow$	Probabilistic inf.

- \+ accelerated gradient ascent converges as $O(\frac{1}{\epsilon})$
  $\epsilon = 0.1 \Rightarrow t = 10$
  $\epsilon = 0.01 \Rightarrow t = 100$
- \+ coordinate ascent does not gut stuck!
- +/- Requires probabilistic inference: tractable for acyclic subproblems
- \- Expensive 'exp' and 'log' operations

First-Order Convex Optimization: Smoothing



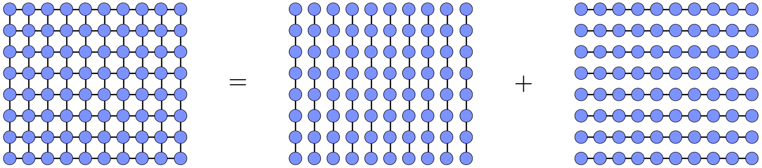
$$\begin{array}{ll} \min & \to \text{"soft" } \min \\ \min_{x^c} E^c(x^c) & \to -T \log \sum_{x^c} \exp(-E^c(x^c)/T) \\ \text{MAP-inf.} & \to \text{Probabilistic inf.} \end{array}$$

- + accelerated gradient ascent converges as $O(\frac{1}{\epsilon})$
  $\epsilon = 0.1 \Rightarrow t = 10$
  $\epsilon = 0.01 \Rightarrow t = 100$
- + coordinate ascent does not gut stuck!
- +/- Requires probabilistic inference: tractable for acyclic subproblems
- \- Expensive 'exp' and 'log' operations

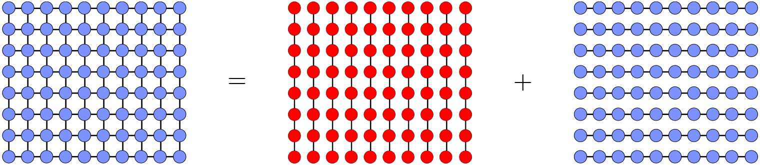
Used to approximate probabilistic inference on the master graph.

Smoothing theory: *[Nesterov, 2005]*
Inference algorithms:
Nesterov: *[Savchynskyy et al., 2011]*
ADSal: *[Savchynskyy et al., 2012]*
See also: *[Johnson et al., 2007, Werner, 2009, Meshi et al., 2012]*

Rounding, Obtaining a Labeling

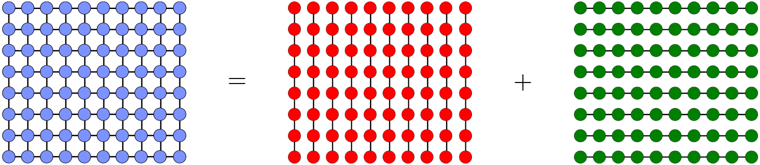


Rounding, Obtaining a Labeling



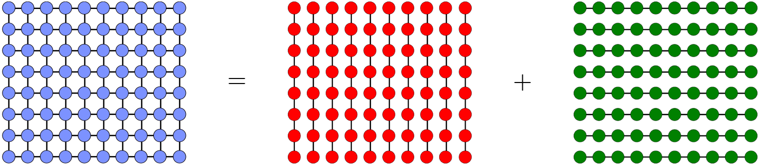
Column- or row-wise labeling as a solution

Rounding, Obtaining a Labeling

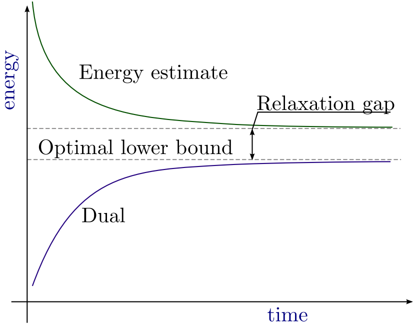


Column- or row-wise labeling as a solution

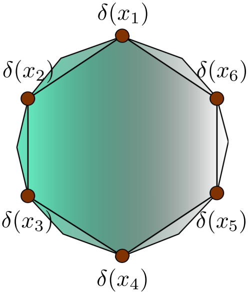
Rounding, Obtaining a Labeling



Column- or row-wise labeling as a solution

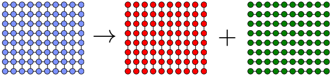


Rounding: Alternatives



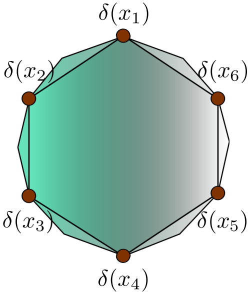
How to round?

- Best of integer solutions, collected over iterations



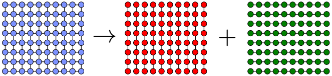
- Local rounding: deterministic/probabilistic: $\mu \geq 0.5$?
  [Ravikumar et al., 2010, Kleinberg and Tardos, 2002]
- Sequential conditional rounding (TRW-S)
  [Kolmogorov, 2006]
- …

Rounding: Alternatives



<u>How to round?</u>

- Best of integer solutions, collected over iterations



- Local rounding: deterministic/probabilistic: $\mu \geq 0.5$?

 *[Ravikumar et al., 2010, Kleinberg and Tardos, 2002]*

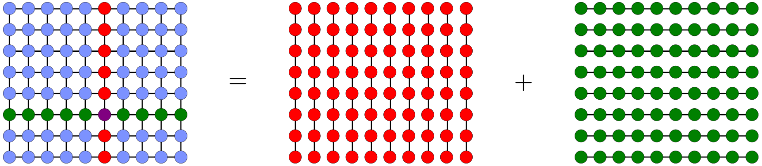
- Sequential conditional rounding (TRW-S)

 *[Kolmogorov, 2006]*

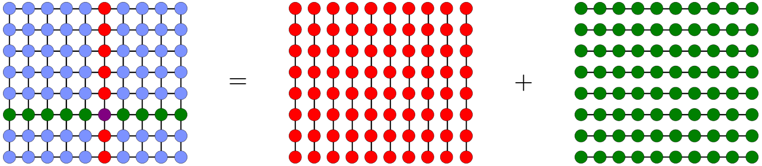
- ...

There is no single best method. Most of methods - heuristics.

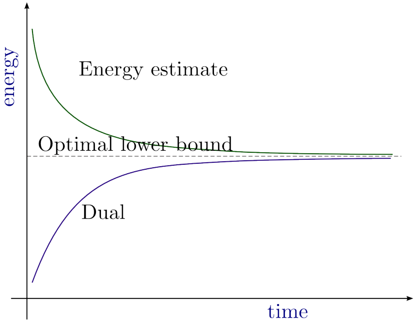
Tree Agreement



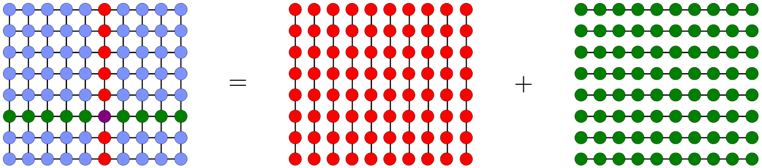
Tree Agreement



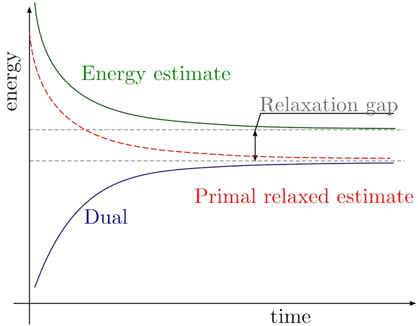
Strong tree agreement in each node $=$ solution of the non-relaxed problem



Tree Agreement



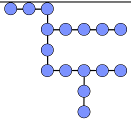
In general: only weak tree agreement holds in the limit



How to obtain the primal relaxed solution: *[Savchynskyy and Schmidt, 2014]*

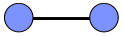
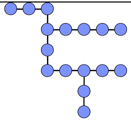
Acyclic Subgraphs ⇒ Local Polytope Relaxation

Acyclic subgraphs for decomposition:

	subgraph	**problematic solvers**
	1-edge graph	no
	chain	proximal inference
	tree	proximal inference

Acyclic Subgraphs ⇒ Local Polytope Relaxation

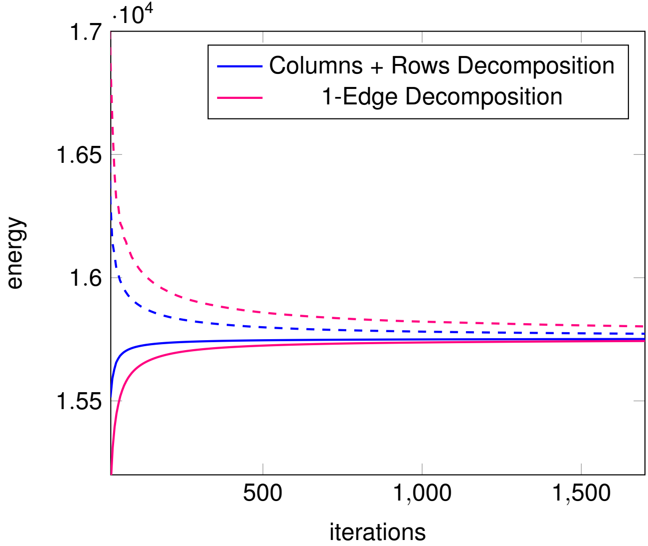
Acyclic subgraphs for decomposition:

	subgraph	**problematic solvers**
	1-edge graph	no
	chain	proximal inference
	tree	proximal inference

Theorem (*[Komodakis* et al.*, 2011]*)

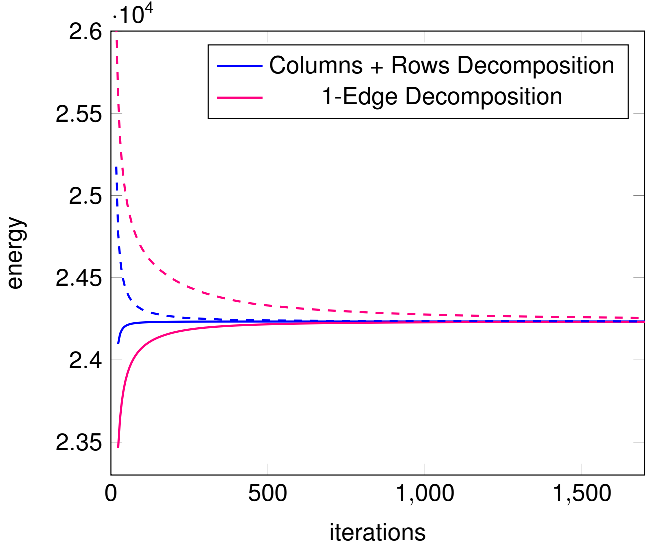
Arbitrary *covering acyclic subgraphs* ⇒ **the same** local polytope *relaxation.*

Which decomposition is better? Smooth. coord. ascent, grid models



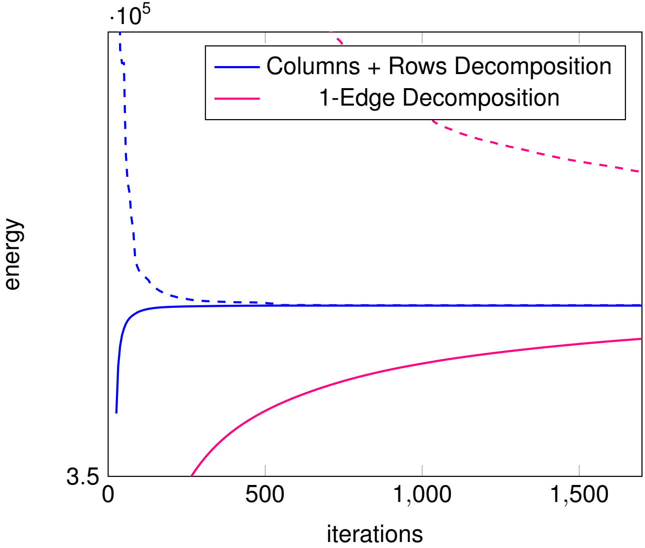
Synthetic uniformly generated 200×100 **grid,** 5 **labels**

Which decomposition is better? Smooth. coord. ascent, grid models



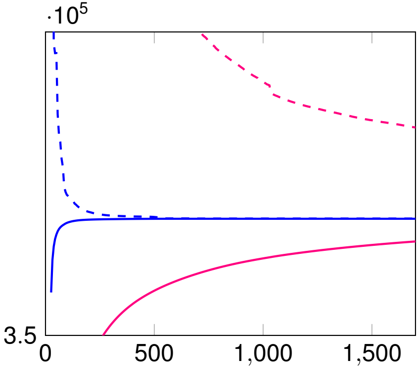
Color segmentation, Potts model, 360 $\times$ 240 **grid,** 12 **labels**

Which decomposition is better? Smooth. coord. ascent, grid models



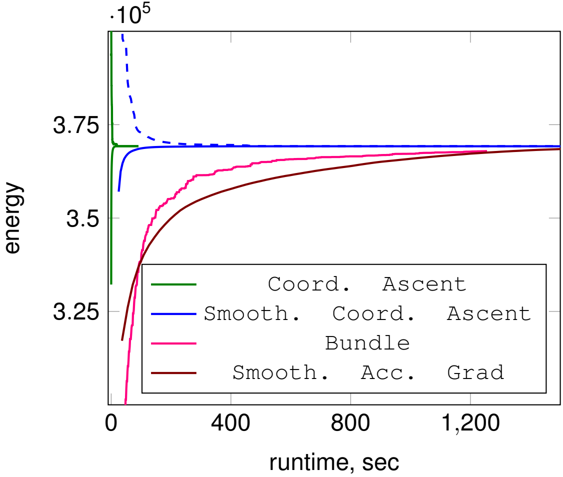
Stereo reconstruction (tsukuba), 384 $\times$ 288 **grid,** 16 **labels**

Which decomposition is better? Smooth. coord. ascent, grid models

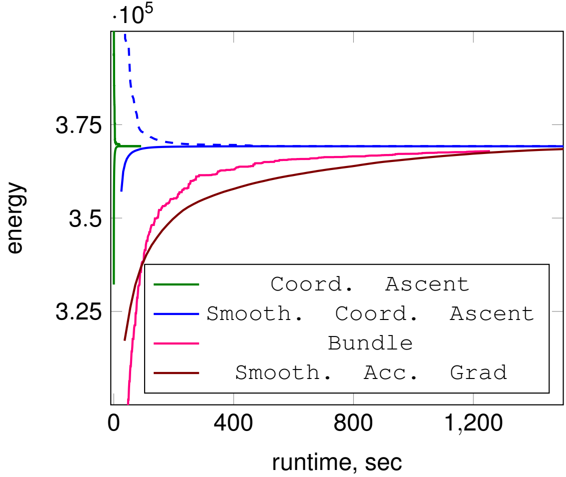


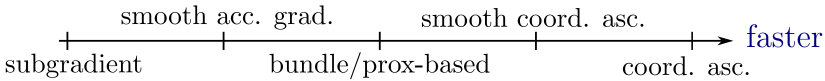
Bigger subproblems $\Rightarrow$ less iterations!

Which update rule is better? Stereo, Columns+Rows decomposition

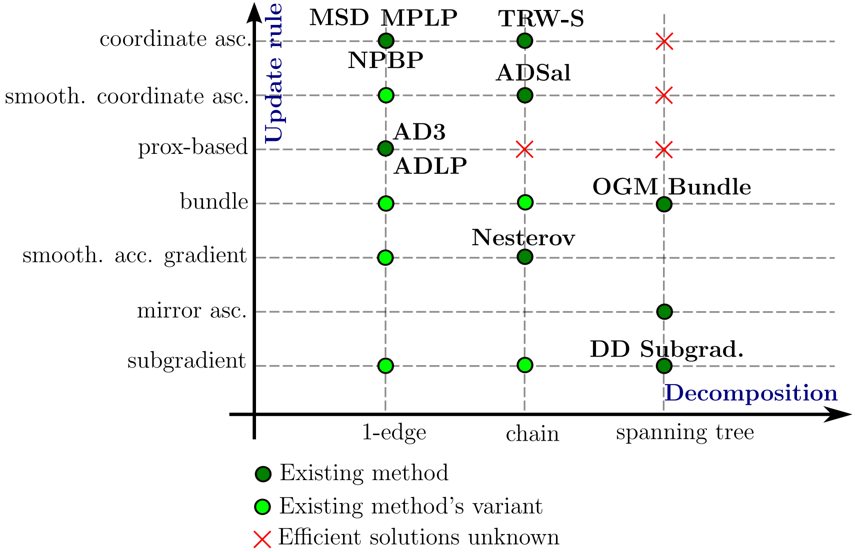


Which update rule is better? Stereo, Columns+Rows decomposition

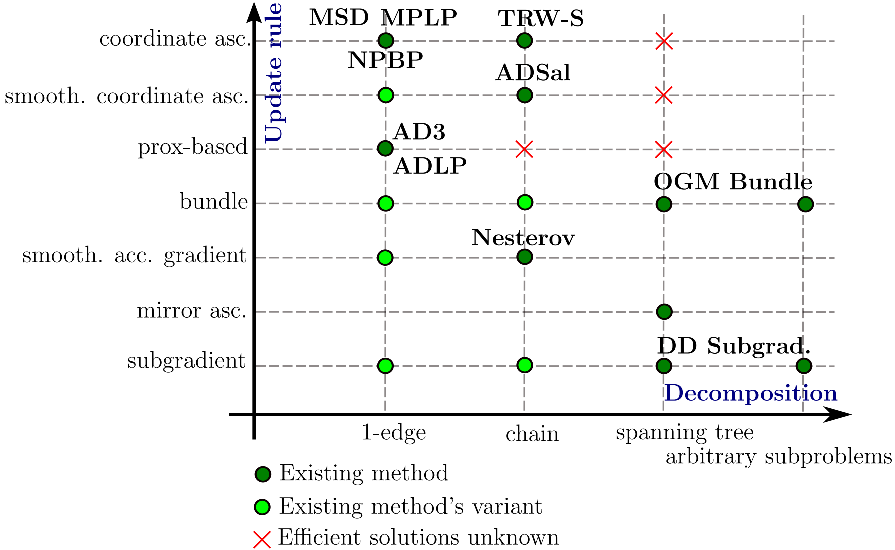




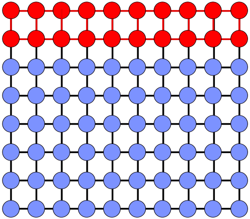
Local Polytope: Algorithm's Overview



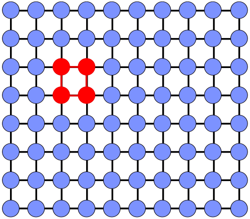
Local Polytope: Algorithm's Overview



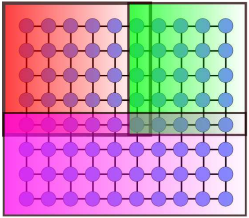
Beyond Acyclic Subproblems



[Komodakis et al., 2011]

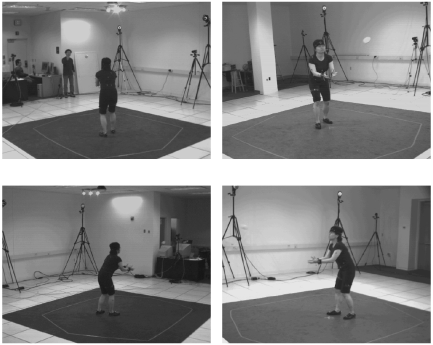


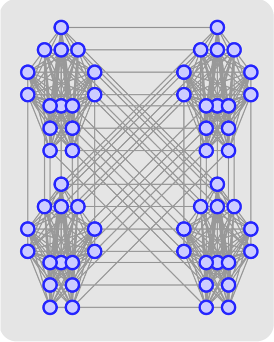
[Sontag et al., 2008]



[Strandmark and Kahl, 2010]

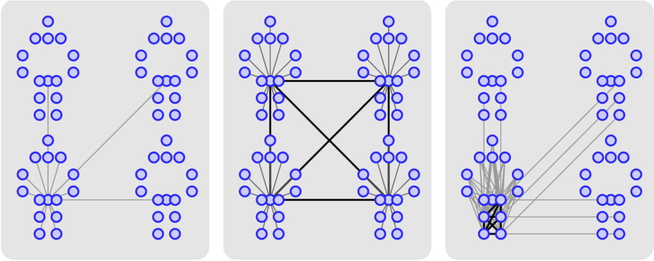
Beyond Acyclic Subproblems



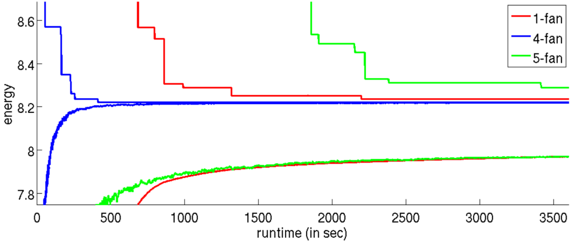


[Kappes et al., *2010]*

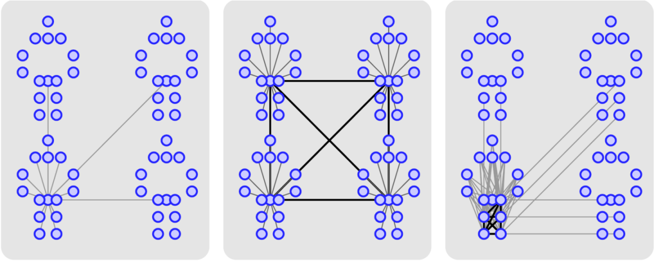
Beyond Acyclic Subproblems



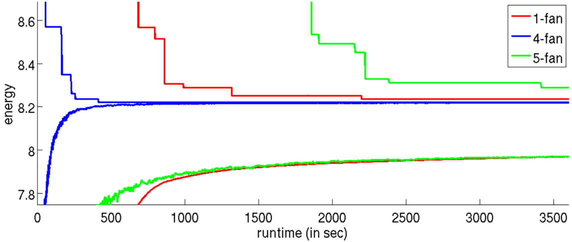
(a) 1-fan (b) 4-fan (c) 5-fan



Beyond Acyclic Subproblems



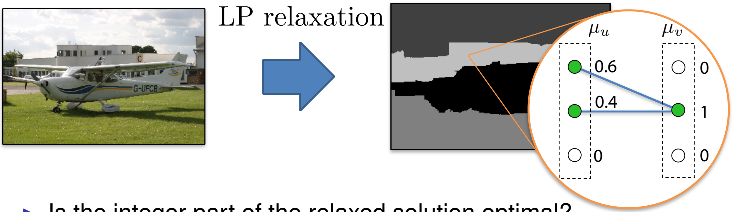
(a) 1-fan (b) 4-fan (c) 5-fan



Hint: Subgraphs, where LP relaxation is loose, are preferable as subproblems.

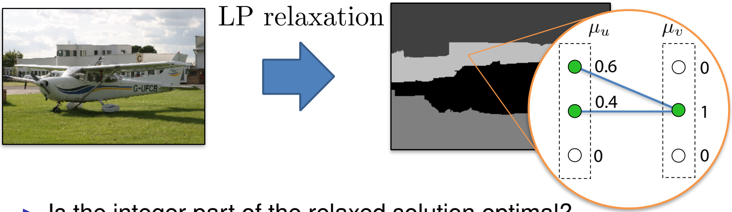
Partial Optimality

Partial Optimality: Definition



- Is the integer part of the relaxed solution optimal?

Partial Optimality: Definition

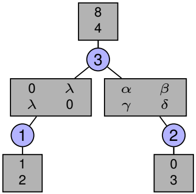


- Is the integer part of the relaxed solution optimal?
- Can we eliminate labels with zero weight?

In general - no, but sometimes - yes.

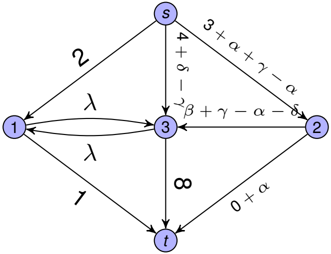
2-Label Local Polytope, QPBO

Recall:



Binary
submodular

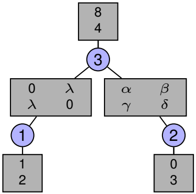
Exact
Reformulation
$\Rightarrow$



Min-st-Cut

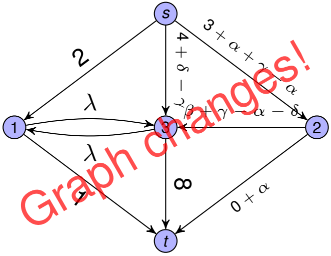
[Boros and Hammer, 2002, Rother et al., 2007a]

2-Label Local Polytope, QPBO



Binary
~~submodular~~

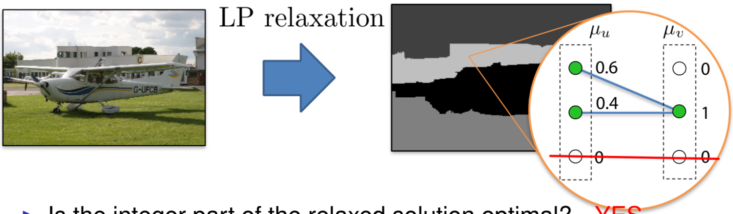
~~Exact~~
~~Reformulation~~
$\Rightarrow$
Local Polytope
Relaxation
(QPBO)



Min-st-Cut

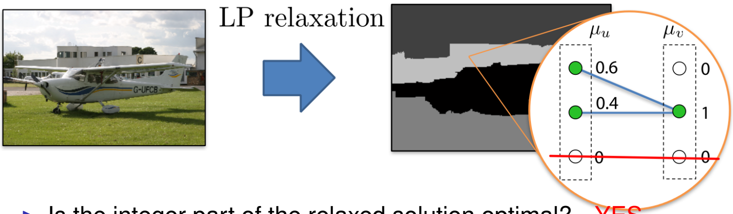
[Boros and Hammer, 2002, Rother et al., 2007a]

2-Label Local Polytope, QPBO



- Is the integer part of the relaxed solution optimal? - YES

2-Label Local Polytope, QPBO

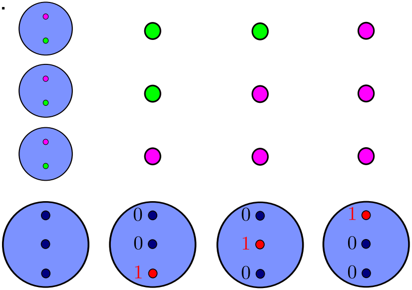


- Is the integer part of the relaxed solution optimal? - YES
- Can we eliminate labels with zero weight? - YES

What about $\geq$ 3 labels?
a) MQPBO

MQPBO-method (submodular relaxation) *[Kohli et al., 2008]* **:**

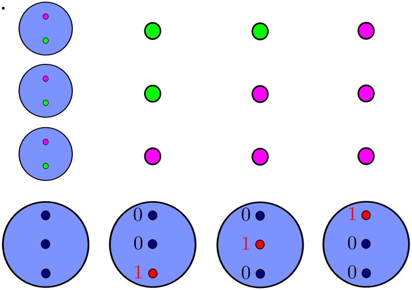
- Convert *n*-label to 2-label problem. ("Battleship" encoding, *[Schlesinger and Flach, 2006]*)
- Apply QPBO.



What about ≥ 3 labels?
a) MQPBO

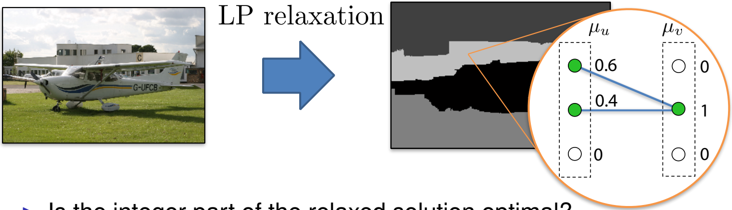
MQPBO-method (submodular relaxation) *[Kohli et al., 2008]* **:**

- Convert *n*-label to 2-label problem. ("Battleship" encoding, *[Schlesinger and Flach, 2006]*)
- Apply QPBO.



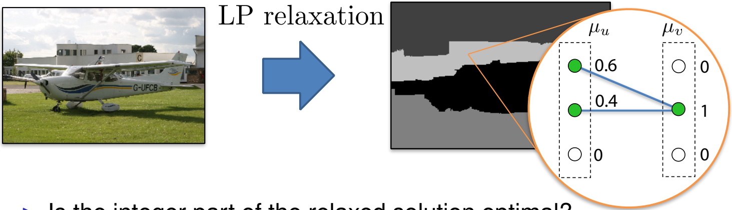
Polytope of the **submodular relaxation** $\supseteq$ local polytope
Depends on the selected order of variables *[Swoboda et al., 2013]*
In practice works up to $3-4$ variables only

What about $\geq$ 3 labels?
b) Arbitrary relaxation



- Is the integer part of the relaxed solution optimal?

What about ≥ 3 labels?
b) Arbitrary relaxation

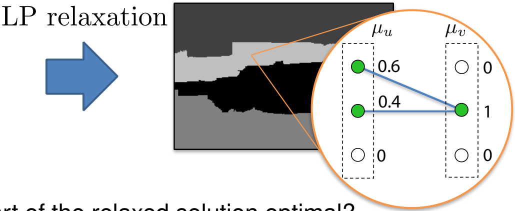


- Is the integer part of the relaxed solution optimal?

integer x and fractional y labelings

Criterion:
Check whether for **all** y the labeling x remains optimal: $\forall y: \; x = \arg\min_{x'} E(x', y)$

What about ≥ 3 labels?
b) Arbitrary relaxation



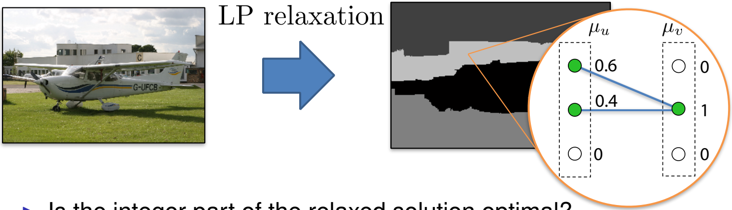
- Is the integer part of the relaxed solution optimal?

integer x and fractional y labelings

Criterion:
Check whether for **all** y the labeling x remains optimal: $\forall y: x = \arg\min_{x'} E(x', y)$

- relaxed inference $\arg\min_{x'} E(x', y)$ is sufficient;

What about ≥ 3 labels?
b) Arbitrary relaxation



- Is the integer part of the relaxed solution optimal?
- Can we eliminate labels with zero weight?

integer x and fractional y labelings

Criterion:
Check whether for **all** y the labeling x remains optimal: $\forall y: x = \arg\min_{x'} E(x', y)$

- relaxed inference $\arg\min_{x'} E(x', y)$ is sufficient;
- efficient procedure exists, for local polytope and a more general question.

Partial Optimality:Potts Models

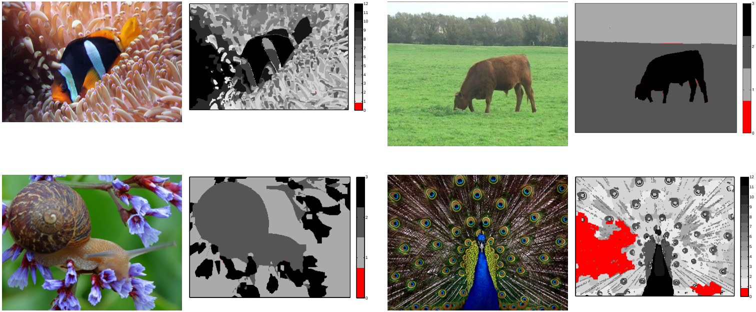
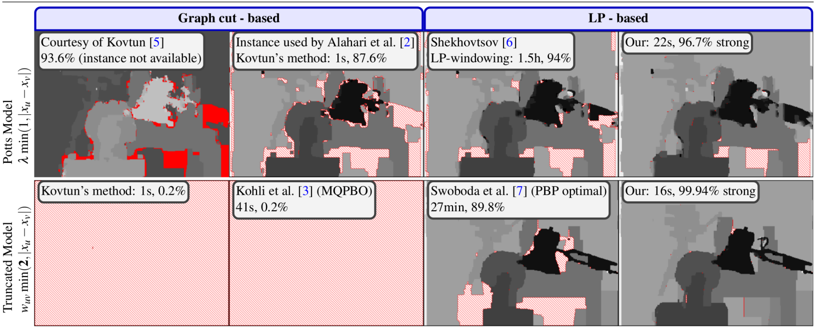


Figure : *[Shekhovtsov et al., 2015]*, Color segmentation, Potts model ($>$ 20 instances)

Progress in Partial Optimality



[Kovtun, 2003]
1s

[Kovtun, 2003], 1s
[Kohli et al., 2008],
41s

[Shekhovtsov, 2014],
1.5h
[Swoboda et al., 2014]
27min

[Shekhovtsov et al., 2015]
22/16 sec

Alternative to Partial Optimality: CombiLP

Partial optimality:

integer x and fractional y labelings

Criterion:
Check whether for **all** y the labeling x remains optimal: $\forall y\colon\ x = \arg\min_{x'} E(x', y)$

Alternative to Partial Optimality: CombiLP

Partial optimality:

integer x and fractional y labelings

Criterion:
Check whether for **all** y the labeling x remains optimal: $\forall y: \ x = \arg\min_{x'} E(x', y)$

CombiLP: [Savchynskyy et al., 2013]

integer x and fractional y labelings

Criterion:
Check whether for **optimal** y^* the labeling x remains optimal: $x = \arg\min_{x'} E(x', y^*)$

Alternative to Partial Optimality: CombiLP

Partial optimality:

Criterion:
Check whether for **all** y the labeling x remains optimal: $\forall y: \; x = \arg\min_{x'} E(x', y)$

integer x and fractional y labelings

CombiLP: [Savchynskyy et al., 2013]

Criterion:
Check whether for **optimal** y^* the labeling x remains optimal: $x = \arg\min_{x'} E(x', y^*)$

integer x and fractional y labelings

+ Weaker requirement $\Rightarrow$ stronger result.

Alternative to Partial Optimality: CombiLP

Partial optimality:

integer x and fractional y labelings

Criterion:
Check whether for **all** y the labeling x remains optimal: $\forall y : x = \arg\min_{x'} E(x', y)$

CombiLP: *[Savchynskyy et al., 2013]*

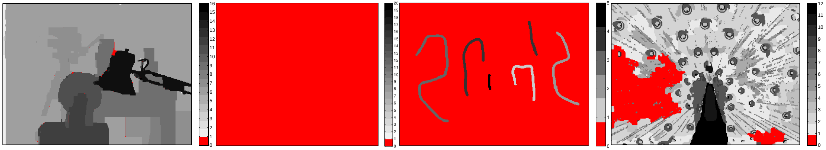
integer x and fractional y labelings

Criterion:
Check whether for **optimal** y^* the labeling x remains optimal: $x = \arg\min_{x'} E(x', y^*)$

+ Weaker requirement $\Rightarrow$ stronger result.
- CombiLP has exponential complexity (Partial optimality - polynomial).

CombiLP vs. Partial Optimality

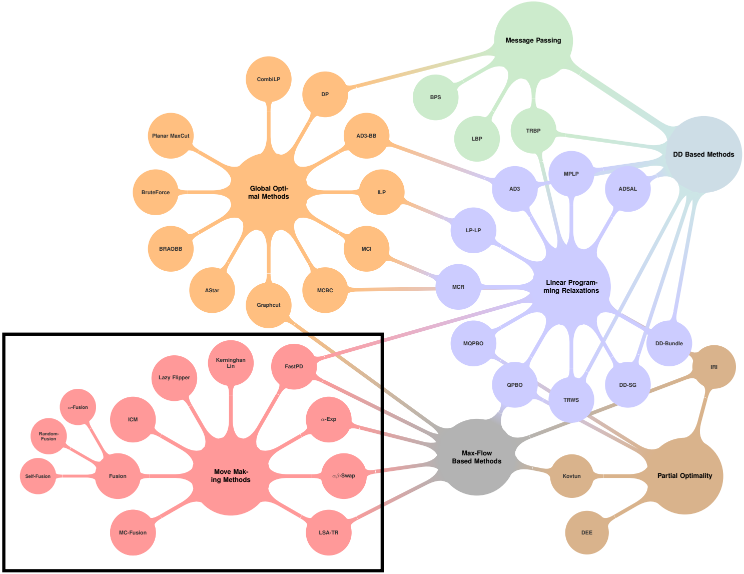
Partial optimality: *[Shekhovtsov et al., 2015]*



CombiLP: *[Savchynskyy et al., 2013]*

Approximative and Move Making Methods

Move Making Methods



Move Making Methods:

- Start from any solution X
- Maintain *current best* feasible solution $\hat{X}$
- Try to improve $\hat{X}$ with a set of *Moves*
- Stop when no more improvement is possible with the set of Moves
- Work in the primal domain

Advantages:

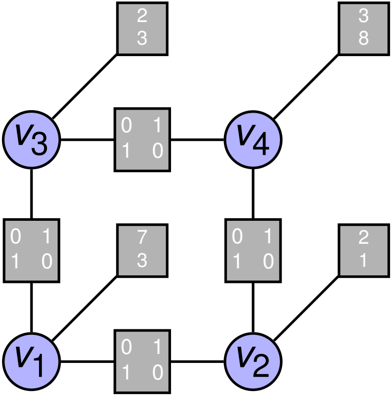
- trivial warm start
- can improve solutions from arbitrary solvers
- fast and scalable

Downsides:

- (often) no lower bound / guarantees
- can get stuck in local minim
- “hope” for a good local minimum

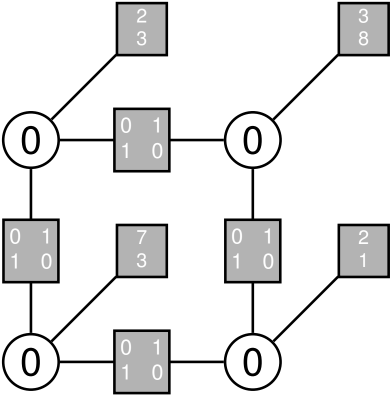
Subgraph Methods

Iterated Conditional Modes *[Besag, 1986]*



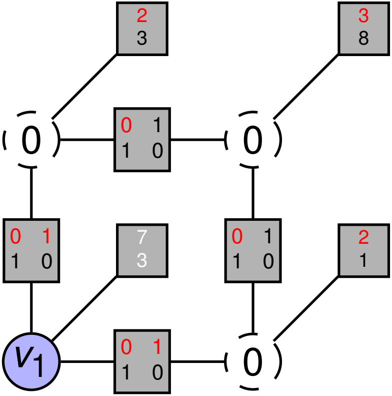
- Start from arbitrary starting point (e.g. $X = 0$)

Iterated Conditional Modes *[Besag, 1986]*



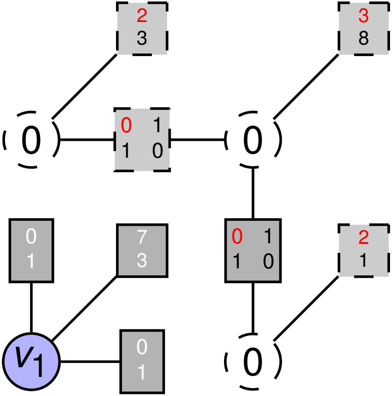
- Start from arbitrary starting point (e.g. $X = 0$)

Iterated Conditional Modes *[Besag, 1986]*



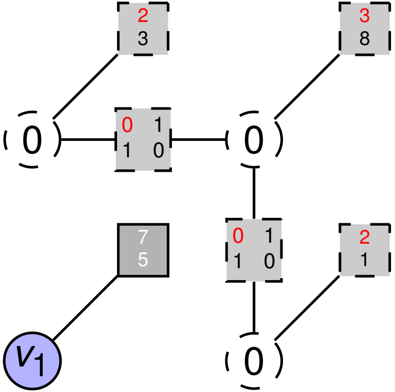
- Start from arbitrary starting point (e.g. $X = 0$)
- Change only a single variable at once

Iterated Conditional Modes *[Besag, 1986]*



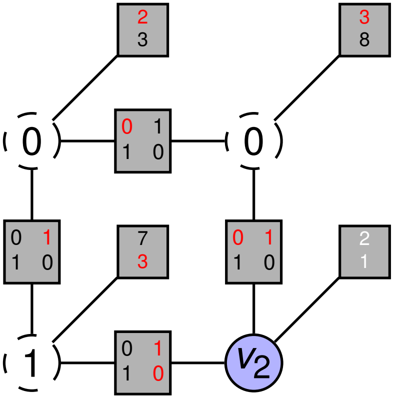
- Start from arbitrary starting point (e.g. $X = 0$)
- Change only a single variable at once

Iterated Conditional Modes *[Besag, 1986]*



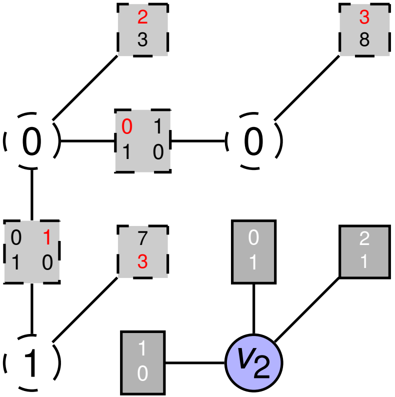
- Start from arbitrary starting point (e.g. $X = 0$)
- Change only a single variable at once
- Change label to local optimal label

Iterated Conditional Modes *[Besag, 1986]*



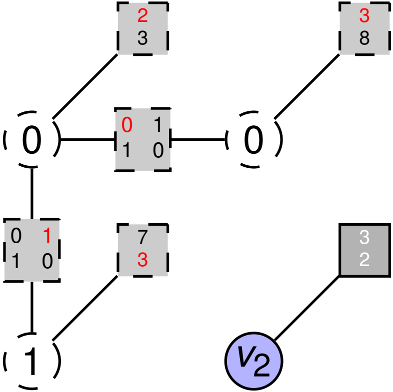
- Start from arbitrary starting point (e.g. $X = 0$)
- Change only a single variable at once
- Change label to local optimal label
- Repeat this for all variables ...

Iterated Conditional Modes *[Besag, 1986]*



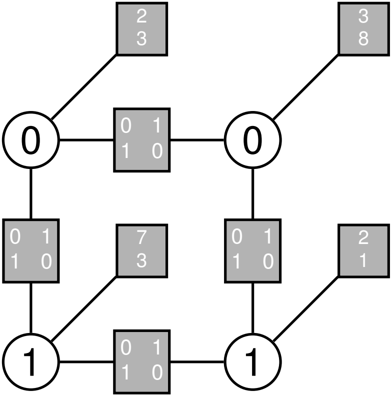
- Start from arbitrary starting point (e.g. $X = 0$)
- Change only a single variable at once
- Change label to local optimal label
- Repeat this for all variables ...

Iterated Conditional Modes *[Besag, 1986]*



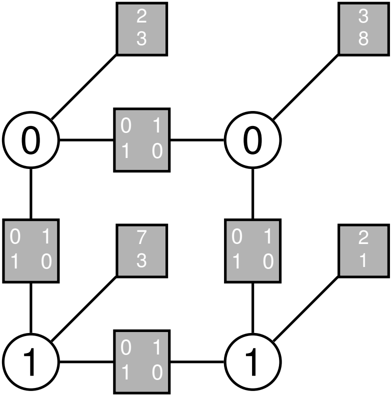
- Start from arbitrary starting point (e.g. $X = 0$)
- Change only a single variable at once
- Change label to local optimal label
- Repeat this for all variables . . .

Iterated Conditional Modes *[Besag, 1986]*

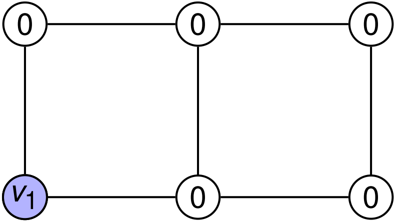


- Start from arbitrary starting point (e.g. $X = 0$)
- Change only a single variable at once
- Change label to local optimal label
- Repeat this for all variables ...

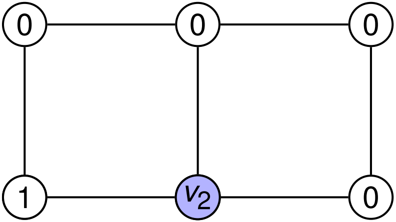
Iterated Conditional Modes *[Besag, 1986]*



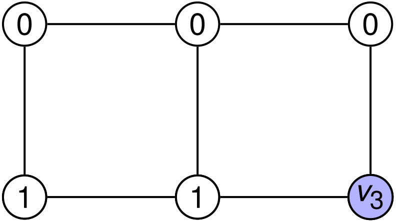
- Start from arbitrary starting point (e.g. $X = 0$)
- Change only a single variable at once
- Change label to local optimal label
- Repeat this for all variables . . .
- . . . until no more improvement is possible



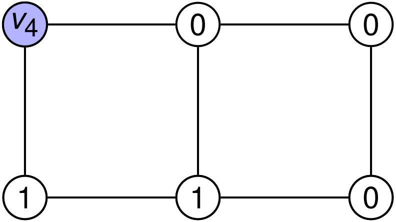
- ICM moves a single variable at once conditioned on the rest



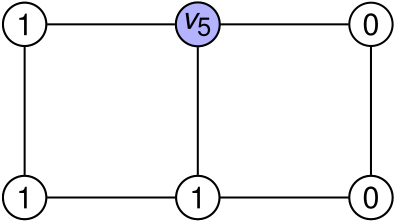
- ICM moves a single variable at once conditioned on the rest



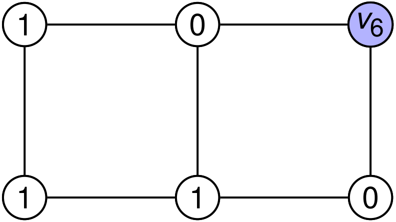
- ICM moves a single variable at once conditioned on the rest



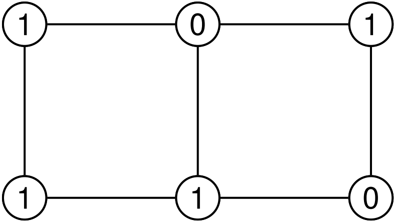
- ICM moves a single variable at once conditioned on the rest



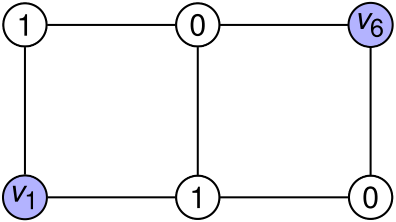
- ICM moves a single variable at once conditioned on the rest



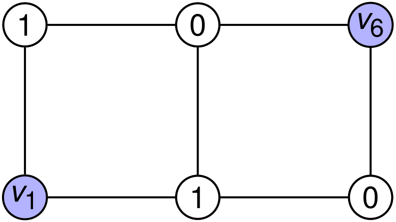
- ICM moves a single variable at once conditioned on the rest



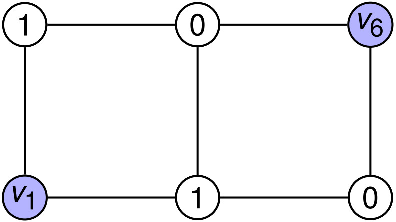
- ICM moves a single variable at once conditioned on the rest
- Obvious extension: Move / optimize multiple variables simultaneously



- ICM moves a single variable at once conditioned on the rest
- Obvious extension: Move / optimize multiple variables simultaneously
- Optimize v_1 and v_6 simultaneously

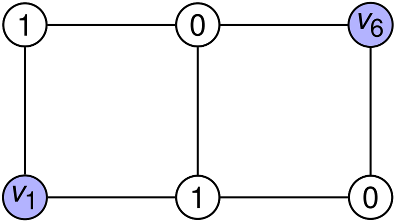


- ICM moves a single variable at once conditioned on the rest
- Obvious extension: Move / optimize multiple variables simultaneously
- Optimize v_1 and v_6 simultaneously

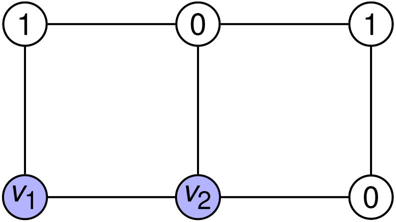


- ICM moves a single variable at
  simultaneously
- Optimize v_1 and v_6 simultaneously

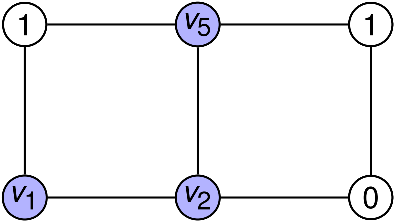
v_1 and v_6 are not connected!
They can be optimized independent!



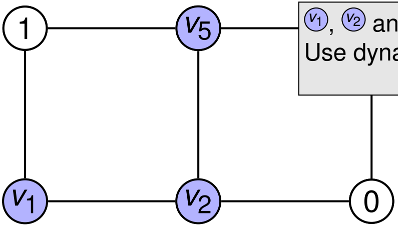
- ICM moves a single variable at once conditioned on the rest
- Obvious extension: Move / optimize multiple variables simultaneously
- Optimize v_1 and v_6 ~~simultaneously~~



- ICM moves a single variable at once conditioned on the rest
- Obvious extension: Move / optimize multiple variables simultaneously
- Optimize v_1 and v_6 ~~simultaneously~~
- Optimize v_1 and v_2 simultaneously



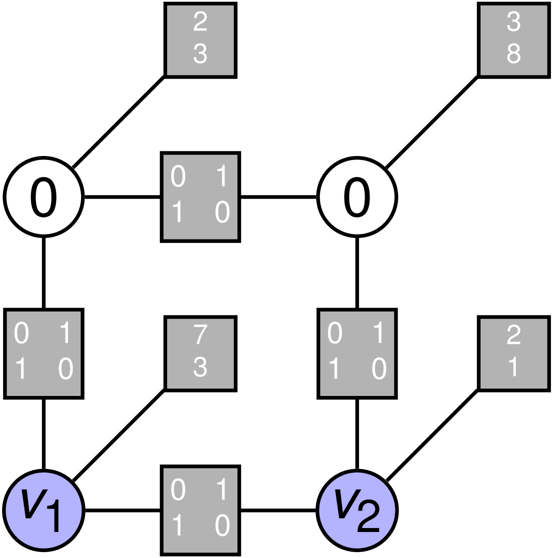
- ICM moves a single variable at once conditioned on the rest
- Obvious extension: Move / optimize multiple variables simultaneously
- Optimize v_1 and v_6 ~~simultaneously~~
- Optimize v_1 and v_2 simultaneously
- Optimize v_1, v_2 and v_5 simultaneously



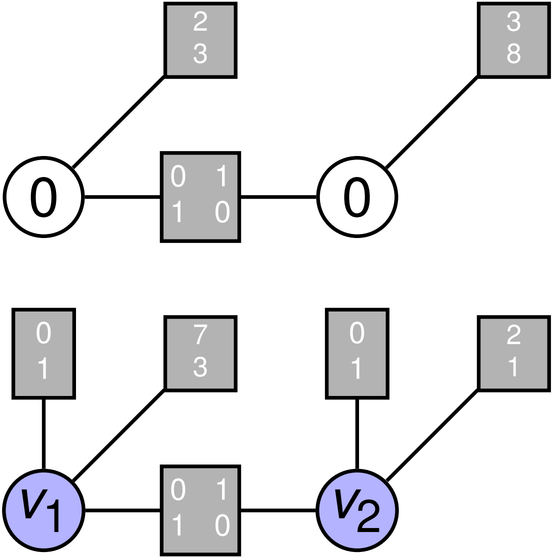
- ICM moves a single variable at
- simultaneously
- Optimize v_1 and ~~simultaneously~~
- Optimize v_1 and v_2 simultaneous
- Optimize v_1, v_2 and v_5 simultaneously

v_1, v_2 and v_2 are a chain!
Use dynamic programing instead of brute force!

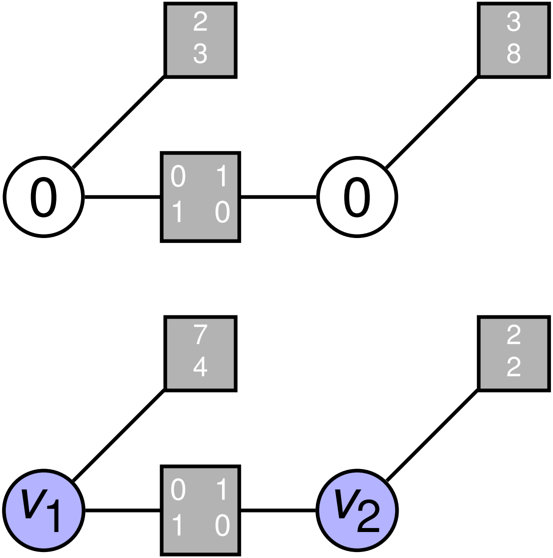
Subgraph Construction



Subgraph Construction

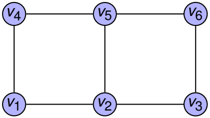


Subgraph Construction

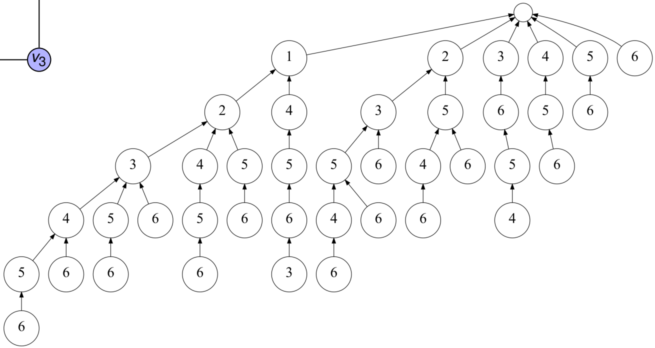


Lazy Flipper *[Andres et al., 2010]*

- systematically optimize all connected subgraphs of size k
- for $k = |V|$ global optimal
- for $k = 1$ equal to ICM *[Besag, 1986]*

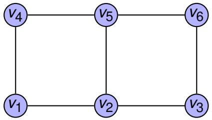


Connected Subgraph Tree:
Maximum Subgraph size $k = 6$

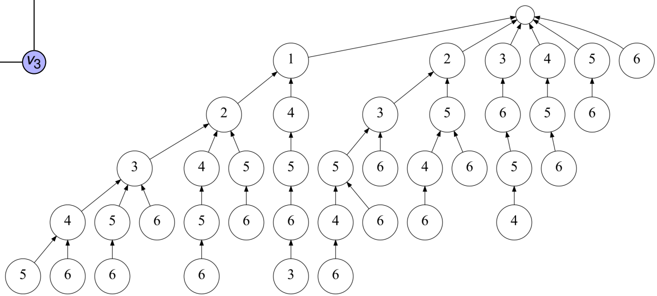


Lazy Flipper *[Andres et al., 2010]*

- systematically optimize all connected subgraphs of size k
- for $k = |V|$ global optimal
- for $k = 1$ equal to ICM *[Besag, 1986]*

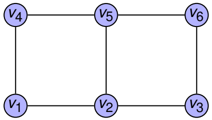


Connected Subgraph Tree:
Maximum Subgraph size $k = 5$

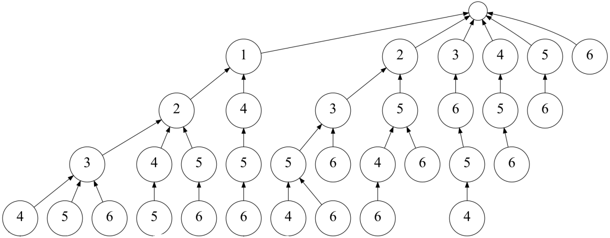


Lazy Flipper *[Andres et al., 2010]*

- systematically optimize all connected subgraphs of size k
- for $k = |V|$ global optimal
- for $k = 1$ equal to ICM *[Besag, 1986]*

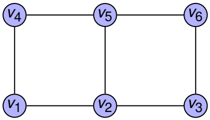


Connected Subgraph Tree:
Maximum Subgraph size $k = 4$

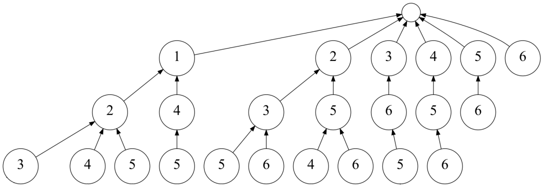


Lazy Flipper *[Andres et al., 2010]*

- systematically optimize all connected subgraphs of size k
- for $k = |V|$ global optimal
- for $k = 1$ equal to ICM *[Besag, 1986]*

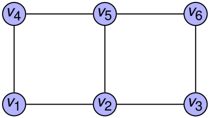


Connected Subgraph Tree:
Maximum Subgraph size $k = 3$

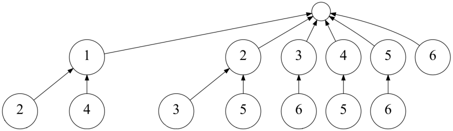


Lazy Flipper *[Andres et al., 2010]*

- systematically optimize all connected subgraphs of size k
- for $k = |V|$ global optimal
- for $k = 1$ equal to ICM *[Besag, 1986]*

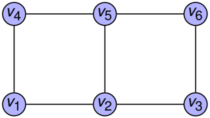


Connected Subgraph Tree:
Maximum Subgraph size $k = 2$

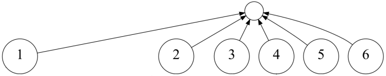


Lazy Flipper *[Andres et al., 2010]*

- systematically optimize all connected subgraphs of size k
- for $k = |V|$ global optimal
- for $k = 1$ equal to ICM *[Besag, 1986]*



Connected Subgraph Tree:
Maximum Subgraph size $k = 1$



Local Rules for Global MAP: When Do They Work ? *[Jung et al., 2009]*

Local, iterative randomized PTAS for MAP:

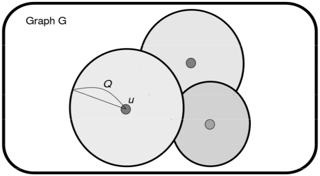


Image Credit: *[Jung et al., 2009]*

- Optimize "ball"-shaped subgraphs around random nodes
- radius of "ball" drawn from truncated geometric distribution Q

Local Rules for Global MAP: When Do They Work ? *[Jung et al., 2009]*

Local, iterative randomized PTAS for MAP:

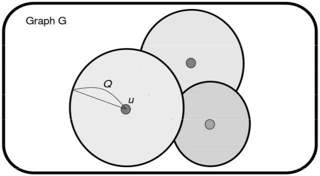


Image Credit: *[Jung et al., 2009]*

- Optimize "ball"-shaped subgraphs around random nodes
- radius of "ball" drawn from truncated geometric distribution Q

Provable $\epsilon -$ *approximation*:

- for certain geometric distribution Q
- on graphs with polynomial growth rate
- with $n\log^2 n$ iterations
- pairwise MRFS

Fusion Move Methods

Fusion Moves *[Lempitsky et al., 2010]*

- Energy function $J(X)$ with large label space $|X|$

$$J(X) = \sum \phi_f(x_{ne(f)})$$

Fusion Moves *[Lempitsky et al., 2010]*

- Energy function $J(X)$ with large label space $|X|$

$$J(X) = \sum \phi_f(x_{ne(f)})$$

- Minimize aux. problems :

$$|\hat{X}| \ll |X|$$

Fusion Moves *[Lempitsky et al., 2010]*

- Energy function $J(X)$ with large label space $|X|$

$$J(X) = \sum \phi_f(x_{ne(f)})$$

- Minimize aux. problems :

$$|\hat{X}| \ll |X|$$

Proposal Generator

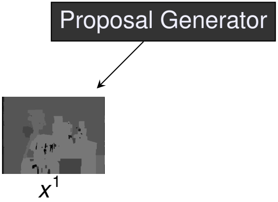
Fusion Moves *[Lempitsky et al., 2010]*

- Energy function $J(X)$ with large label space $|X|$

$$J(X) = \sum \phi_f(x_{ne(f)})$$

- Minimize aux. problems :

$$|\hat{X}| \ll |X|$$



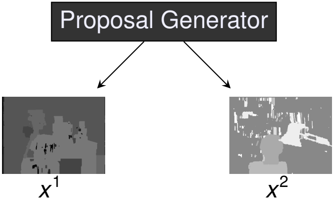
Fusion Moves *[Lempitsky et al., 2010]*

- Energy function $J(X)$ with large label space $|X|$

$$J(X) = \sum \phi_f(x_{ne(f)})$$

- Minimize aux. problems :

$$|\hat{X}| \ll |X|$$



Fusion Moves *[Lempitsky et al., 2010]*

- Energy function $J(X)$ with large label space $|X|$

$$J(X) = \sum \phi_f(x_{ne(f)})$$

- Minimize aux. problems :

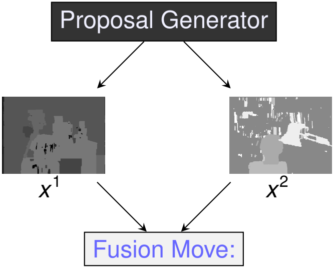
$$|\hat{X}| \ll |X|$$

- Fusion Move:
  - Binary Label Space:

    $$\hat{X} = \{x \in X | \forall i : x_i \in \{x^1, x^2\}\}$$

  - Allowed Moves:

    $$X^{Move} = \left\{ x \in X | J(x) \leq min \left(J(x^1), J(x^2) \right) \right\}$$



Fusion Moves *[Lempitsky et al., 2010]*

- Energy function $J(X)$ with large label space $|X|$

$$J(X) = \sum \phi_f(x_{ne(f)})$$

- Minimize aux. problems :

$$|\hat{X}| \ll |X|$$

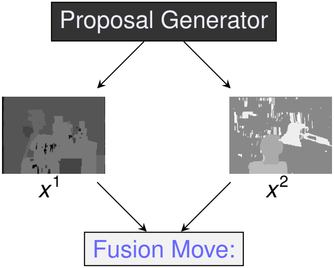
- Fusion Move:
  - Binary Label Space:

    $$\hat{X} = \{x \in X | \forall i : x_i \in \{x^1, x^2\}\}$$

  - Allowed Moves:

    $$X^{Move} = \left\{x \in X | J(x) \le min\left(J(x^1), J(x^2)\right)\right\}$$

  - Optimized with graph-cut/qpbo*[Lempitsky et al., 2010]*, ilp solvers and movemaking methods*[Kappes et al., 2014]*



Fusion Moves *[Lempitsky et al., 2010]*

- Energy function $J(X)$ with large label space $|X|$

$$J(X) = \sum \phi_f(x_{ne(f)})$$

- Minimize aux. problems :

$$|\hat{X}| \ll |X|$$

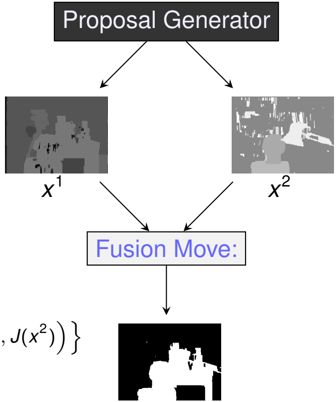
- Fusion Move:
  - Binary Label Space:

    $$\hat{X} = \{x \in X | \forall i : x_i \in \{x^1, x^2\}\}$$

  - Allowed Moves:

    $$X^{Move} = \left\{x \in X | J(x) \leq min\left(J(x^1), J(x^2)\right)\right\}$$

  - Optimized with graph-cut/qpbo *[Lempitsky et al., 2010]*, ilp solvers and movemaking methods *[Kappes et al., 2014]*



Fusion Moves *[Lempitsky et al., 2010]*

- Energy function $J(X)$ with large label space $|X|$

$$J(X) = \sum \phi_f(x_{ne(f)})$$

- Minimize aux. problems :

$$|\hat{X}| \ll |X|$$

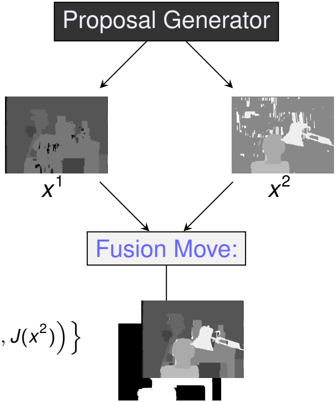
- Fusion Move:
  - Binary Label Space:

    $$\hat{X} = \{x \in X | \forall i : x_i \in \{x^1, x^2\}\}$$

  - Allowed Moves:

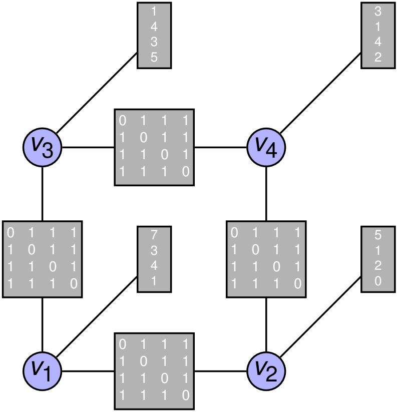
    $$X^{Move} = \left\{ x \in X | J(x) \leq min\left(J(x^1), J(x^2)\right) \right\}$$

  - Optimized with graph-cut/qpbo*[Lempitsky et al., 2010]*, ilp solvers and movemaking methods*[Kappes et al., 2014]*



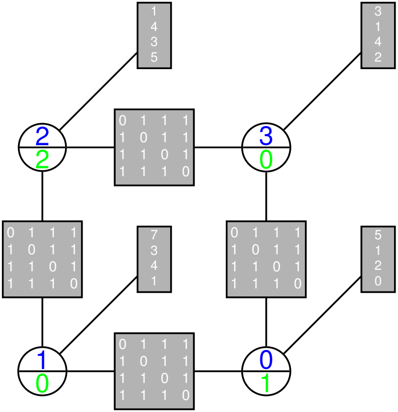
Subproblem Construction

Multi-Label Energy Function:



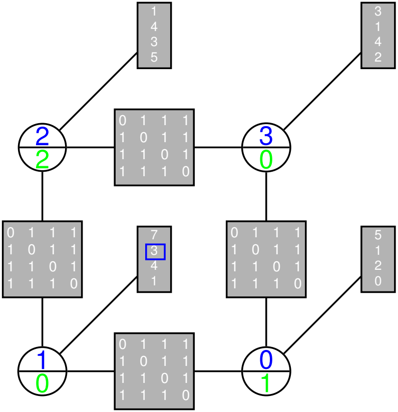
Subproblem Construction

Multi-Label Energy Function:



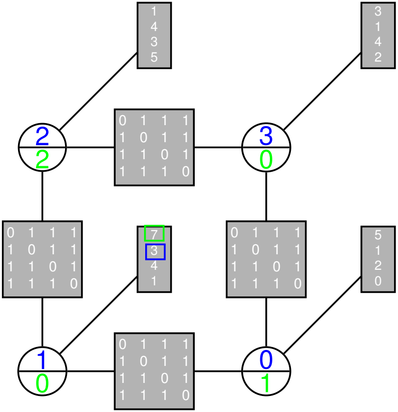
Subproblem Construction

Multi-Label Energy Function:



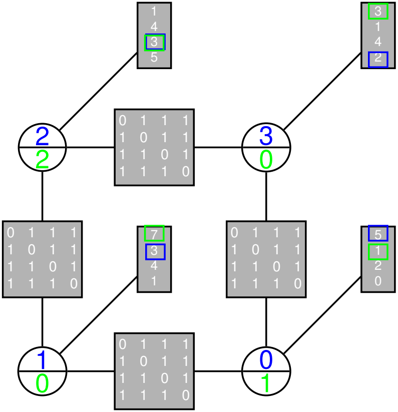
Subproblem Construction

Multi-Label Energy Function:



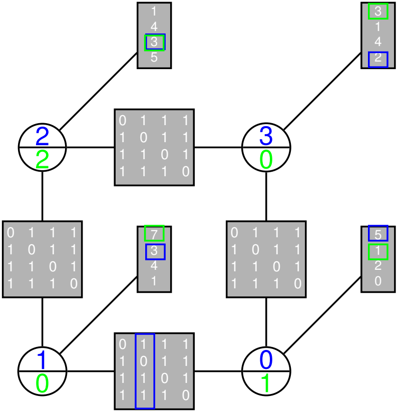
Subproblem Construction

Multi-Label Energy Function:



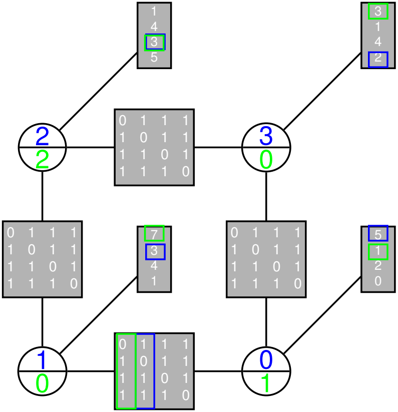
Subproblem Construction

Multi-Label Energy Function:



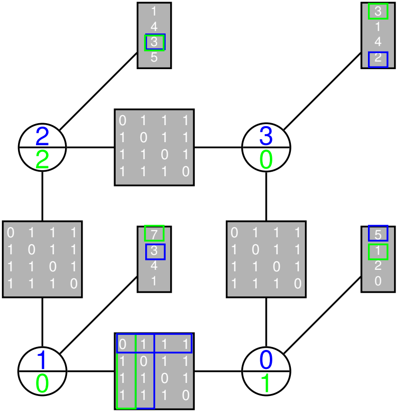
Subproblem Construction

Multi-Label Energy Function:



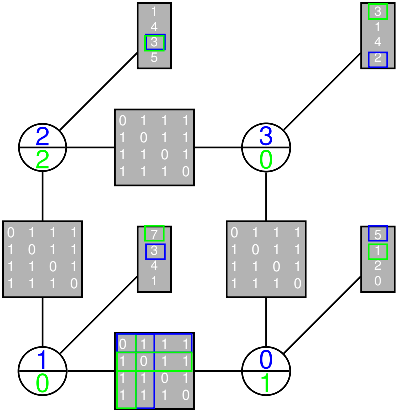
Subproblem Construction

Multi-Label Energy Function:



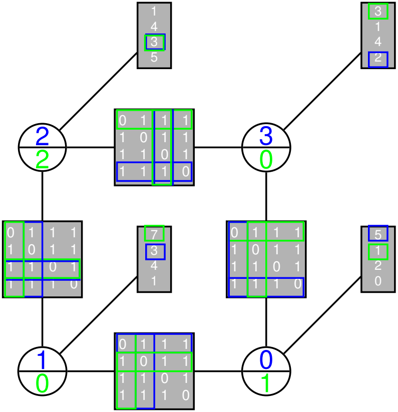
Subproblem Construction

Multi-Label Energy Function:



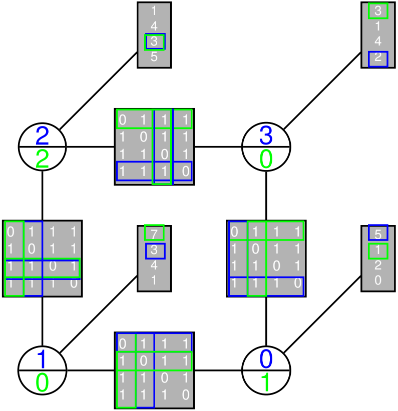
Subproblem Construction

Multi-Label Energy Function:

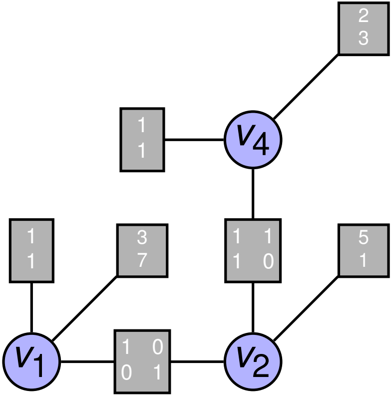


Subproblem Construction

Multi-Label Energy Function:

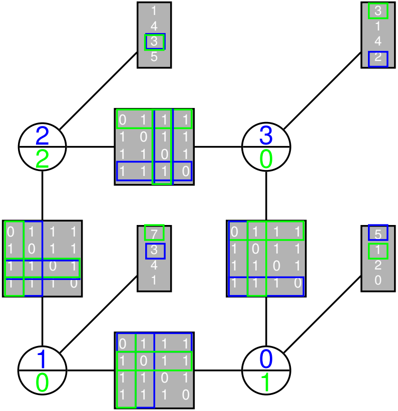


Binary Energy Function:

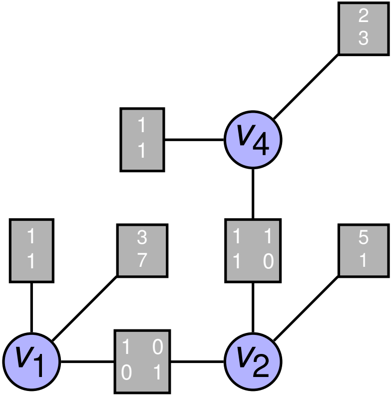


Subproblem Construction

Multi-Label Energy Function:



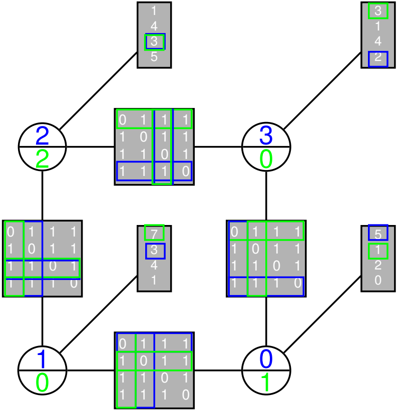
Binary Energy Function:



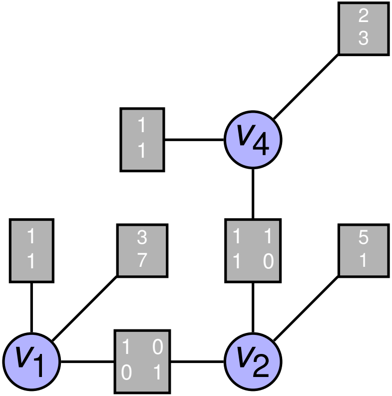
- Binary subproblems are “ordinary” binary graphical models

Subproblem Construction

Multi-Label Energy Function:



Binary Energy Function:



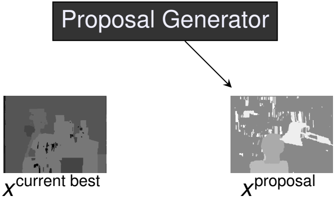
- Binary subproblems are “ordinary” binary graphical models
- Any solver can be used to optimize them

Fusion Move Based Algorithms

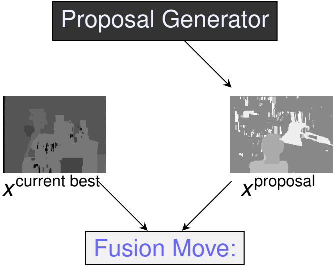
Proposal Generator

$x^{\text{current best}}$

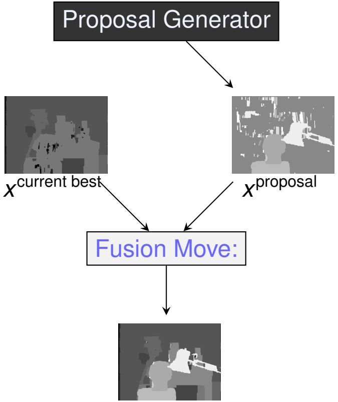
Fusion Move Based Algorithms



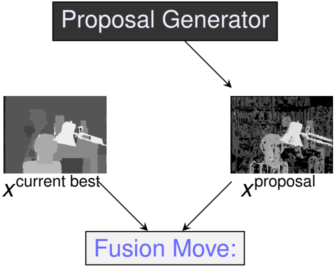
Fusion Move Based Algorithms



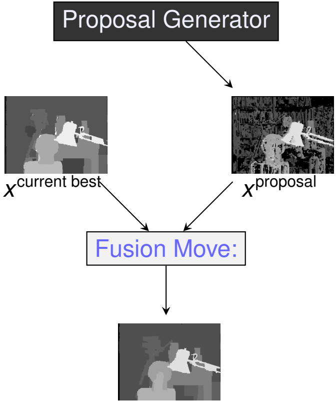
Fusion Move Based Algorithms



Fusion Move Based Algorithms



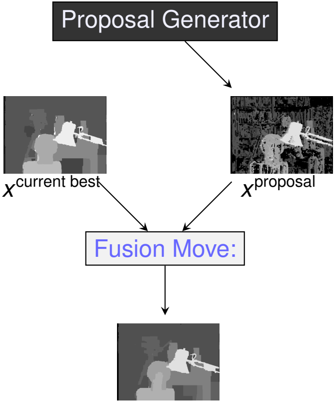
Fusion Move Based Algorithms



Fusion Move Based Algorithms

- α-Expansion:
  [Kolmogorov and Zabih, 2002]

$$x_i^{\text{proposal}} = \alpha$$



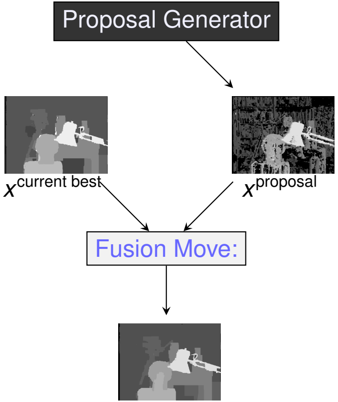
Fusion Move Based Algorithms

- α-Expansion: *[Kolmogorov and Zabih, 2002]*

$$x_i^{\text{proposal}} = \alpha$$

- $\alpha\beta$-Swap: *[Kolmogorov and Zabih, 2002]*

$$x_i^{\text{proposal}} = \begin{cases} \alpha & \text{if } x_i^{\text{current}} = \beta \text{ and } \alpha \in X_i \\ \beta & \text{if } x_i^{\text{current}} = \alpha \text{ and } \beta \in X_i \\ x_i^{\text{current}} & \text{else} \end{cases}$$



Fusion Move Based Algorithms

- α-Expansion:
  [Kolmogorov and Zabih, 2002]

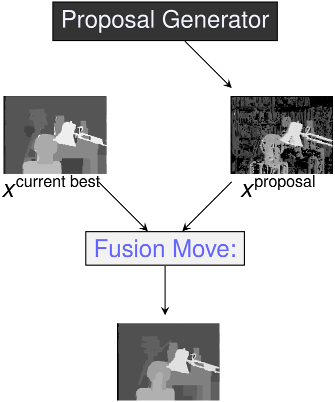
$$x_i^{\text{proposal}} = \alpha$$

- $\alpha\beta$-Swap: *[Kolmogorov and Zabih, 2002]*

$$x_i^{\text{proposal}} = \begin{cases} \alpha & \text{if } x_i^{\text{current}} = \beta \text{ and } \alpha \in X_i \\ \beta & \text{if } x_i^{\text{current}} = \alpha \text{ and } \beta \in X_i \\ x_i^{\text{current}} & \text{else} \end{cases}$$

- Jump-Move: *[Lempitsky et al., 2010]*

$$x_i^{\text{proposal}} = \begin{cases} x_i^{\text{current}} + k & \text{if } x_i^{\text{current}} + k \in X_i \\ x_i^{\text{current}} & \text{else} \end{cases}$$



Fusion Move Based Algorithms

- α-Expansion:
  [Kolmogorov and Zabih, 2002]

$$x_i^{\text{proposal}} = \alpha$$

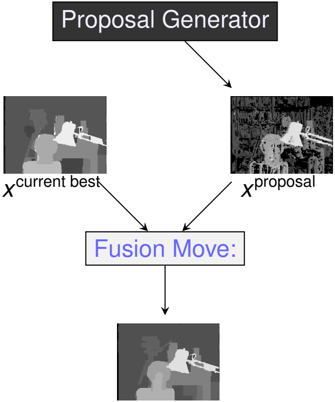
- $\alpha\beta$-Swap:*[Kolmogorov and Zabih, 2002]*

$$x_i^{\text{proposal}} = \begin{cases} \alpha & \text{if } x_i^{\text{current}} = \beta \text{ and } \alpha \in X_i \\ \beta & \text{if } x_i^{\text{current}} = \alpha \text{ and } \beta \in X_i \\ x_i^{\text{current}} & \text{else} \end{cases}$$

- Jump-Move: *[Lempitsky et al., 2010]*

$$x_i^{\text{proposal}} = \begin{cases} x_i^{\text{current}} + k & \text{if } x_i^{\text{current}} + k \in X_i \\ x_i^{\text{current}} & \text{else} \end{cases}$$

- Randomized Proposals
  [Kappes et al., 2014]



Fusion Move Based Algorithms

- α-Expansion: *[Kolmogorov and Zabih, 2002]*

$$x_i^{\text{proposal}} = \alpha$$

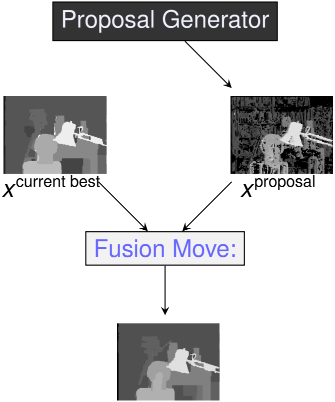
- $\alpha\beta$-Swap:*[Kolmogorov and Zabih, 2002]*

$$x_i^{\text{proposal}} = \begin{cases} \alpha & \text{if } x_i^{\text{current}} = \beta \text{ and } \alpha \in X_i \\ \beta & \text{if } x_i^{\text{current}} = \alpha \text{ and } \beta \in X_i \\ x_i^{\text{current}} & \text{else} \end{cases}$$

- Jump-Move: *[Lempitsky et al., 2010]*

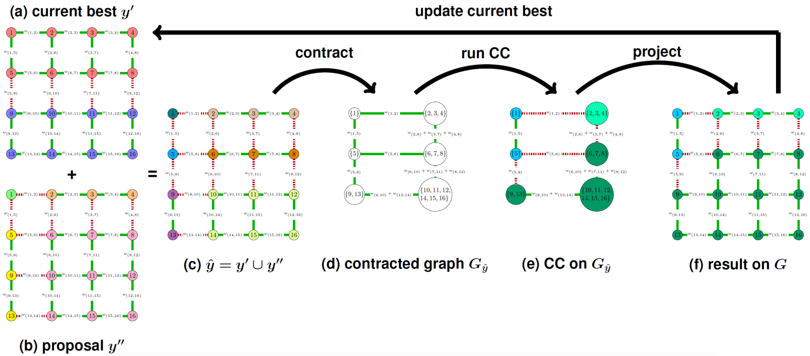
$$x_i^{\text{proposal}} = \begin{cases} x_i^{\text{current}} + k & \text{if } x_i^{\text{current}} + k \in X_i \\ x_i^{\text{current}} & \text{else} \end{cases}$$

- Randomized Proposals *[Kappes et al., 2014]*
- Inference Based Proposals *[Lempitsky et al., 2010, Kappes et al., 2014]*



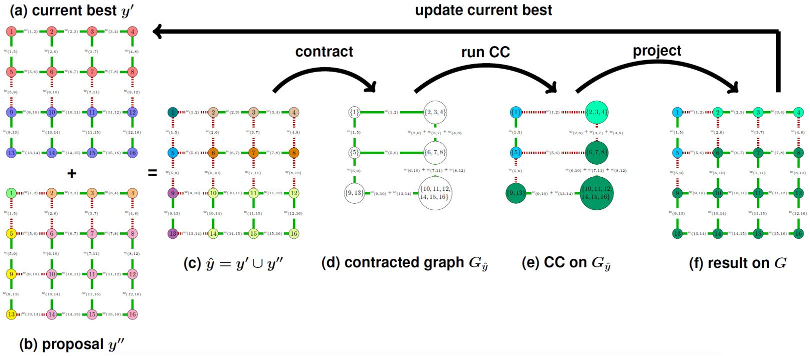
Fusion Moves for Correlation Clustering *[Beier et al., 2015]*

$$J(X) = \sum_{uv \in E} \quad \omega_{uv} \cdot x_u \neq x_v \qquad |X_i| = |V|$$



Fusion Moves for Correlation Clustering *[Beier et al., 2015]*

$$J(X) = \sum_{uv \in E} \quad \omega_{uv} \cdot \underbrace{x_u \neq x_v}_{:= y_e} \qquad |X_i| = |V|$$



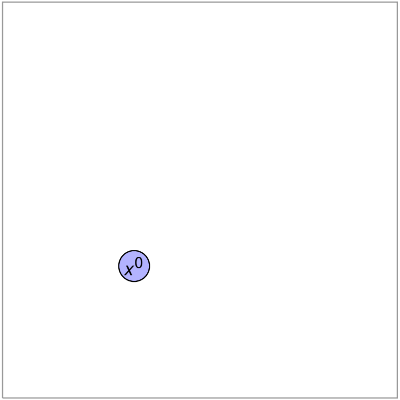
Local Submodular Approximations

Local Submodular Approximations

Non Submodular Energy $J(X)$:

$$J(X) = \sum \varphi_i(x_i) + \sum \varphi_{ij}(x_i, x_j) \quad x_i \in \{0, 1\}$$

$$\tilde{J}^{x^0}(x) \approx J(X); \qquad \tilde{J}^{x^0}(x) \geq J(x); \quad \tilde{J}^{x^0}(x_0) = J(x_0)$$



- Compute submodular approximation $\tilde{J}^{x^0}(x)$ arround current labeling x^0

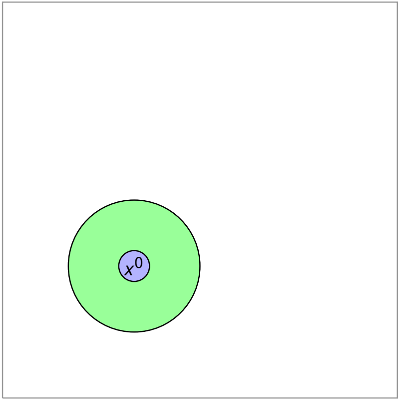
- LSA-TR and LSA-AUX *[Gorelick et al., 2014a]*

Local Submodular Approximations

Non Submodular Energy $J(X)$:

$$J(X) = \sum \varphi_i(x_i) + \sum \varphi_{ij}(x_i, x_j) \quad x_i \in \{0, 1\}$$

$$\tilde{J}^{x^0}(x) \approx J(X); \qquad \tilde{J}^{x^0}(x) \geq J(x); \quad \tilde{J}^{x^0}(x_0) = J(x_0)$$



- Compute submodular approximation $\tilde{J}^{x^0}(x)$ arround current labeling x^0
- Approximation $\tilde{J}^{x^0}(x)$ is only valid close to current labeling

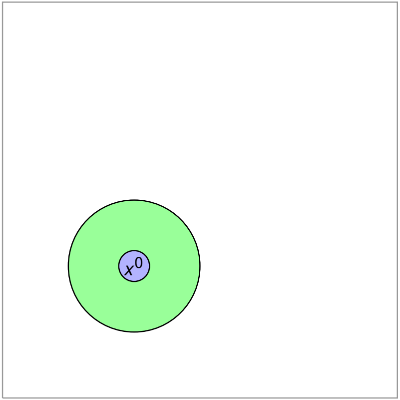
- LSA-TR and LSA-AUX *[Gorelick* et al.*, 2014*a*]*

Local Submodular Approximations

Non Submodular Energy $J(X)$:

$$J(X) = \sum \varphi_i(x_i) + \sum \varphi_{ij}(x_i, x_j) \quad x_i \in \{0, 1\}$$

$$\tilde{J}^{x^0}(x) \approx J(X); \qquad \tilde{J}^{x^0}(x) \geq J(x); \quad \tilde{J}^{x^0}(x_0) = J(x_0)$$



- Compute submodular approximation $\tilde{J}^{x^0}(x)$ arround current labeling x^0
- Approximation $\tilde{J}^{x^0}(x)$ is only valid close to current labeling
- By optimizing $\tilde{J}$ the next solution is generated

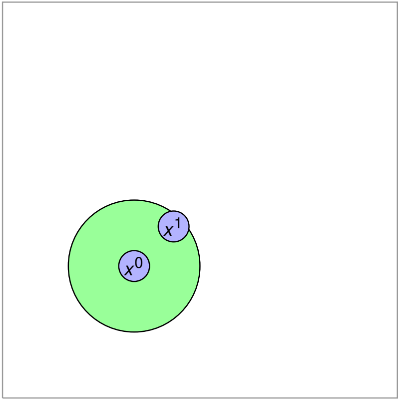
- LSA-TR and LSA-AUX *[Gorelick et al., 2014a]*

Local Submodular Approximations

Non Submodular Energy $J(X)$:

$$J(X) = \sum \varphi_i(x_i) + \sum \varphi_{ij}(x_i, x_j) \quad x_i \in \{0,1\}$$

$$\tilde{J}^{x^0}(x) \approx J(X); \qquad \tilde{J}^{x^0}(x) \geq J(x); \quad \tilde{J}^{x^0}(x_0) = J(x_0)$$



- Compute submodular approximation $\tilde{J}^{x^0}(x)$ arround current labeling x^0
- Approximation $\tilde{J}^{x^0}(x)$ is only valid close to current labeling
- By optimizing $\tilde{J}$ the next solution is generated

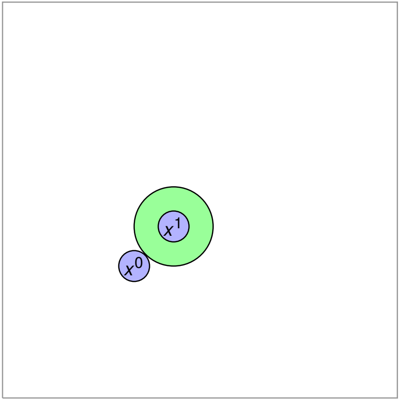
- LSA-TR and LSA-AUX *[Gorelick* et al.*, 2014*a*]*

Local Submodular Approximations

Non Submodular Energy $J(X)$:

$$J(X) = \sum \varphi_i(x_i) + \sum \varphi_{ij}(x_i, x_j) \quad x_i \in \{0, 1\}$$

$$\tilde{J}^{x^0}(x) \approx J(X); \qquad \tilde{J}^{x^0}(x) \geq J(x); \quad \tilde{J}^{x^0}(x_0) = J(x_0)$$



- Compute submodular approximation $\tilde{J}^{x^1}(x)$ arround current labeling x^1
- Approximation $\tilde{J}^{x^1}(x)$ is only valid close to current labeling
- By optimizing $\tilde{J}$ the next solution is generated

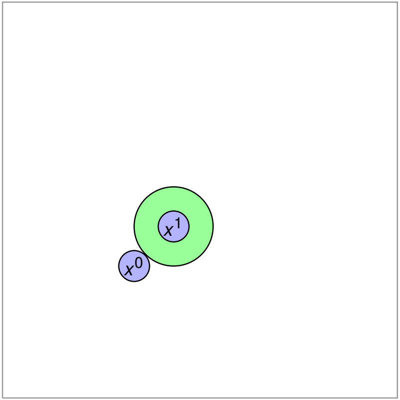
- LSA-TR and LSA-AUX *[Gorelick et al., 2014a]*

Local Submodular Approximations

Non Submodular Energy $J(X)$:

$$J(X) = \sum \varphi_i(x_i) + \sum \varphi_{ij}(x_i, x_j) \quad x_i \in \{0, 1\}$$

$$\tilde{J}^{x^0}(x) \approx J(X); \qquad \tilde{J}^{x^0}(x) \geq J(x); \quad \tilde{J}^{x^0}(x_0) = J(x_0)$$



- Compute submodular approximation $\tilde{J}^{x^1}(x)$ arround current labeling x^1
- Approximation $\tilde{J}^{x^1}(x)$ is only valid close to current labeling
- By optimizing $\tilde{J}$ the next solution is generated

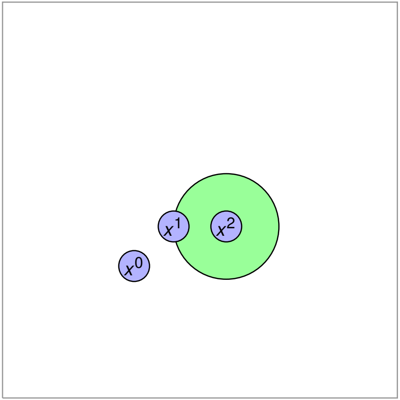
- LSA-TR and LSA-AUX *[Gorelick* et al.*, 2014*a*]*

Local Submodular Approximations

Non Submodular Energy $J(X)$:

$$J(X) = \sum \varphi_i(x_i) + \sum \varphi_{ij}(x_i, x_j) \quad x_i \in \{0, 1\}$$

$$\tilde{J}^{x^0}(x) \approx J(X); \qquad \tilde{J}^{x^0}(x) \geq J(x); \quad \tilde{J}^{x^0}(x_0) = J(x_0)$$



- Compute submodular approximation $\tilde{J}^{x^2}(x)$ arround current labeling x^2
- Approximation $\tilde{J}^{x^2}(x)$ is only valid close to current labeling
- By optimizing $\tilde{J}$ the next solution is generated

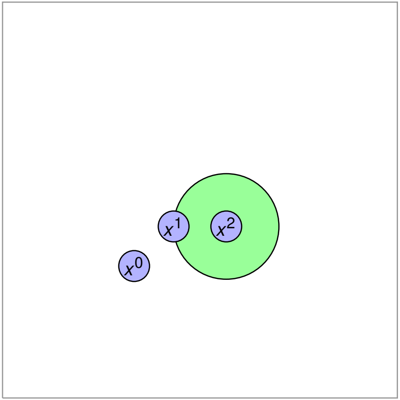
- LSA-TR and LSA-AUX [Gorelick et al., 2014a]

Local Submodular Approximations

Non Submodular Energy $J(X)$:

$$J(X) = \sum \varphi_i(x_i) + \sum \varphi_{ij}(x_i, x_j) \quad x_i \in \{0, 1\}$$

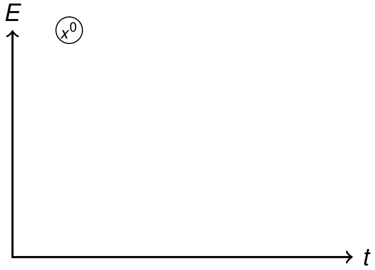
$$\tilde{J}^{x^0}(x) \approx J(X); \qquad \tilde{J}^{x^0}(x) \geq J(x); \quad \tilde{J}^{x^0}(x_0) = J(x_0)$$

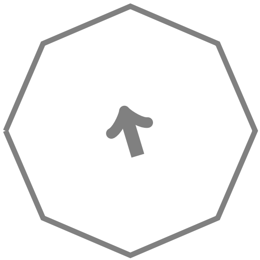


- Compute submodular approximation $\tilde{J}^{x^2}(x)$ arround current labeling x^2
- Approximation $\tilde{J}^{x^2}(x)$ is only valid close to current labeling
- By optimizing $\tilde{J}$ the next solution is generated

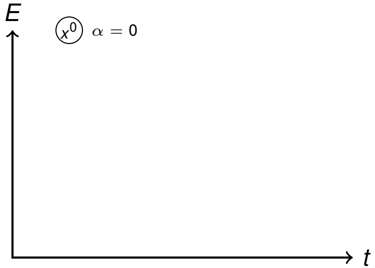
- LSA-TR and LSA-AUX [Gorelick et al., 2014a]

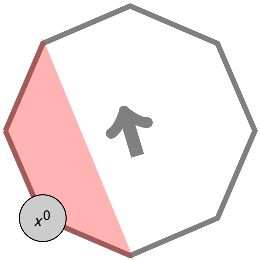
Polyhedral Interpretation





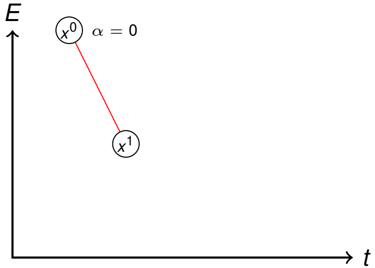
Polyhedral Interpretation

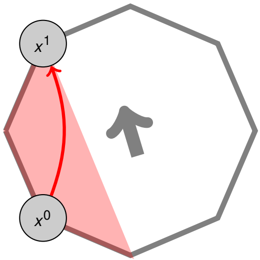




- Each move can be interpreted as an optimization of an inner polytope

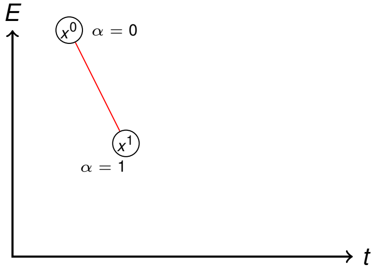
Polyhedral Interpretation

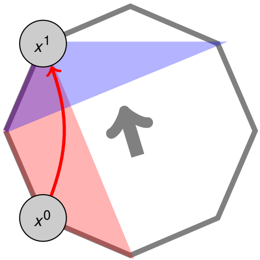




- Each move can be interpreted as an optimization of an inner polytope
- Each inner polytope includes the vertex of the current best solution

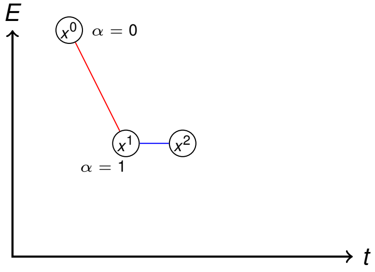
Polyhedral Interpretation

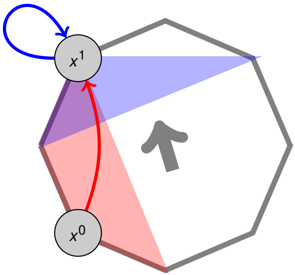




- Each move can be interpreted as an optimization of an inner polytope
- Each inner polytope includes the vertex of the current best solution

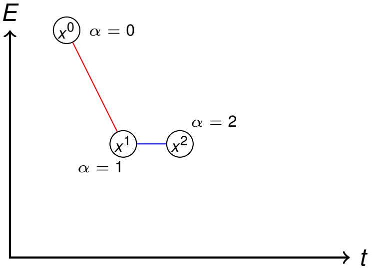
Polyhedral Interpretation

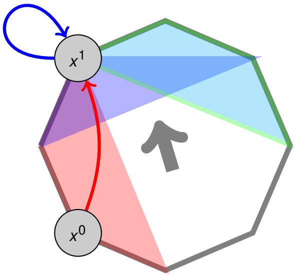




- Each move can be interpreted as an optimization of an inner polytope
- Each inner polytope includes the vertex of the current best solution

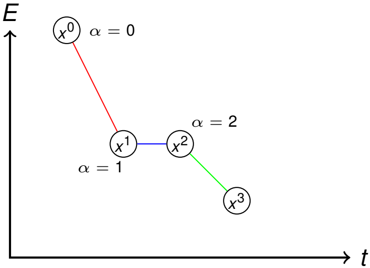
Polyhedral Interpretation

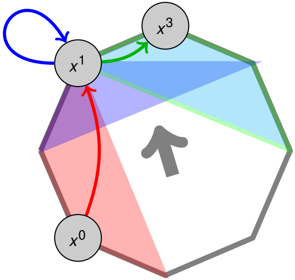




- Each move can be interpreted as an optimization of an inner polytope
- Each inner polytope includes the vertex of the current best solution

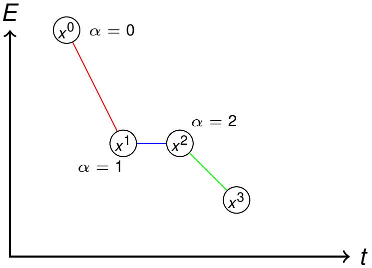
Polyhedral Interpretation

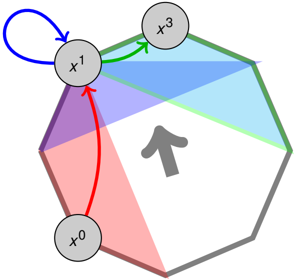




- Each move can be interpreted as an optimization of an inner polytope
- Each inner polytope includes the vertex of the current best solution

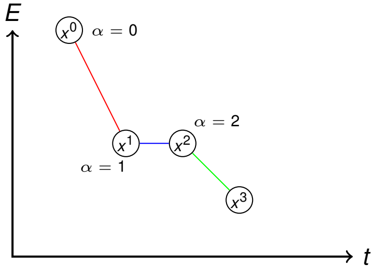
Polyhedral Interpretation

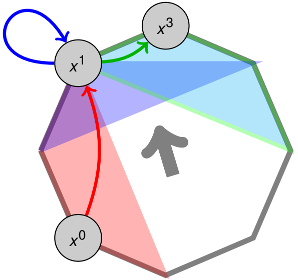




- Each move can be interpreted as an optimization of an inner polytope
- Each inner polytope includes the vertex of the current best solution

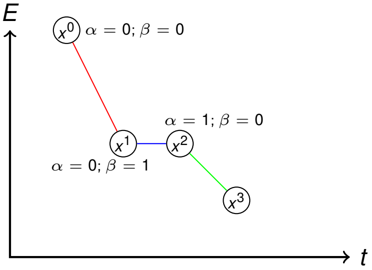
Polyhedral Interpretation

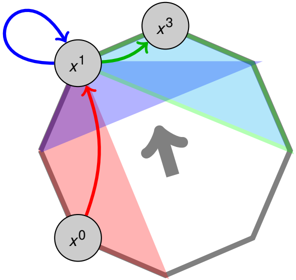




- Each move can be interpreted as an optimization of an inner polytope
- Each inner polytope includes the vertex of the current best solution

Polyhedral Interpretation

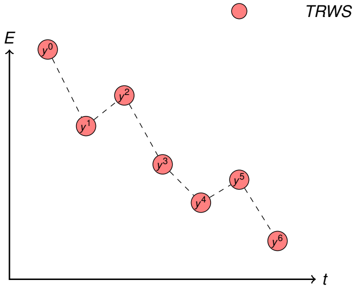




- Each move can be interpreted as an optimization of an inner polytope
- Each inner polytope includes the vertex of the current best solution

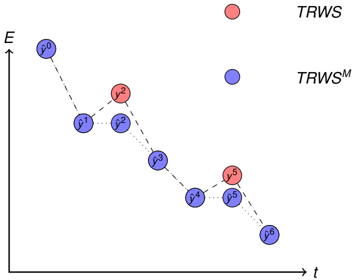
Meta-Methods: Combining Methods to get better overall Performance

Fusion Moves With Inference Based Proposals



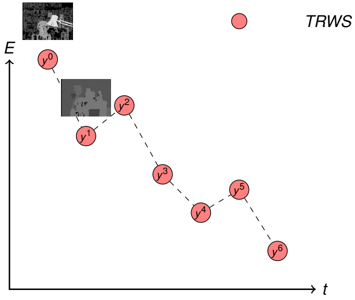
- Some algorithms do not decrease the energy monotonously

Fusion Moves With Inference Based Proposals



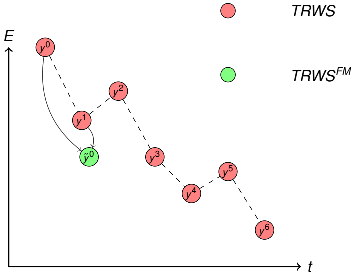
- Some algorithms do not decrease the energy monotonously
- Make monotone by remembering current best solution

Fusion Moves With Inference Based Proposals



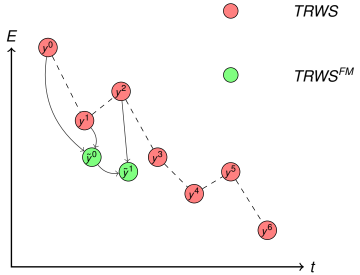
- Some algorithms do not decrease the energy monotonously
- Make monotone by remembering current best solution
- Used generated labels more efficient

Fusion Moves With Inference Based Proposals



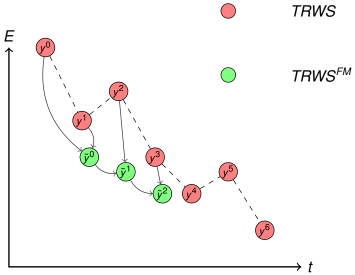
- Some algorithms do not decrease the energy monotonously
- Make monotone by remembering current best solution
- Used generated labels more efficient
- Compute Fusion move between y^0 and y^1

Fusion Moves With Inference Based Proposals



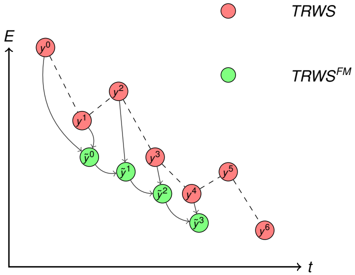
- Some algorithms do not decrease the energy monotonously
- Make monotone by remembering current best solution
- Used generated labels more efficient
- Compute Fusion move between y^0 and y^1

Fusion Moves With Inference Based Proposals



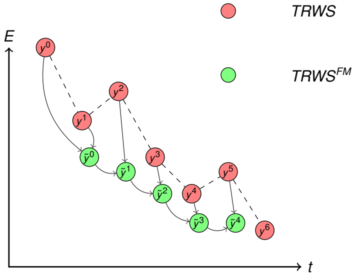
- Some algorithms do not decrease the energy monotonously
- Make monotone by remembering current best solution
- Used generated labels more efficient
- Compute Fusion move between y^0 and y^1

Fusion Moves With Inference Based Proposals



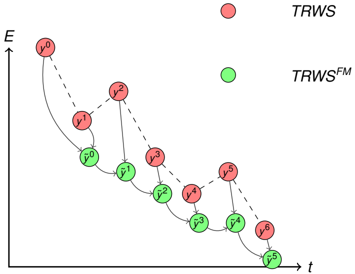
- Some algorithms do not decrease the energy monotonously
- Make monotone by remembering current best solution
- Used generated labels more efficient
- Compute Fusion move between y^0 and y^1

Fusion Moves With Inference Based Proposals



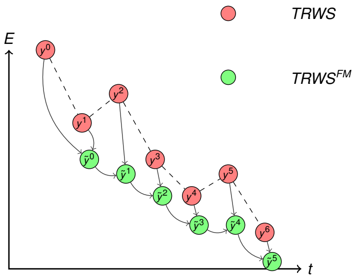
- Some algorithms do not decrease the energy monotonously
- Make monotone by remembering current best solution
- Used generated labels more efficient
- Compute Fusion move between y^0 and y^1

Fusion Moves With Inference Based Proposals



- Some algorithms do not decrease the energy monotonously
- Make monotone by remembering current best solution
- Used generated labels more efficient
- Compute Fusion move between y^0 and y^1

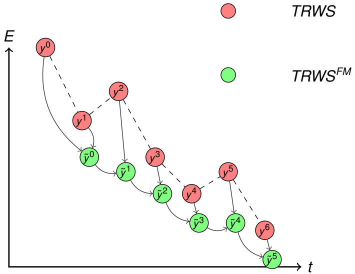
Fusion Moves With Inference Based Proposals



- Some algorithms do not decrease the energy monotonously
- Make monotone by remembering current best solution
- Used generated labels more efficient
- Compute Fusion move between y^0 and y^1

- Proposed by Lempitsky[Lempitsky et al., 2010] with Loopy BP
- Investigated in detail by Kappes and Beier[Kappes et al., 2014] with TRWS Dual Decomposition.

Fusion Moves With Inference Based Proposals



- Some algorithms do not decrease the energy monotonously
- Make monotone by remembering current best solution
- Used generated labels more efficient
- Compute Fusion move between y^0 and y^1

- Proposed by Lempitsky *[Lempitsky et al., 2010]* with Loopy BP
- Investigated in detail by Kappes and Beier *[Kappes et al., 2014]* with TRWS Dual Decomposition.
- OpenGM allows this trick for **all** inference algorithms

Fusion Moves For Parallelization

$Alg1$

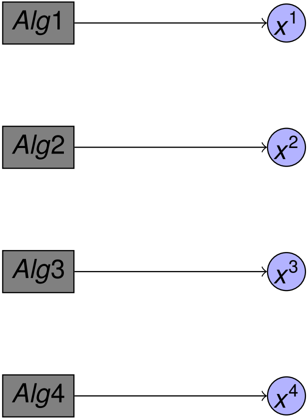
$Alg2$

$Alg3$

$Alg4$

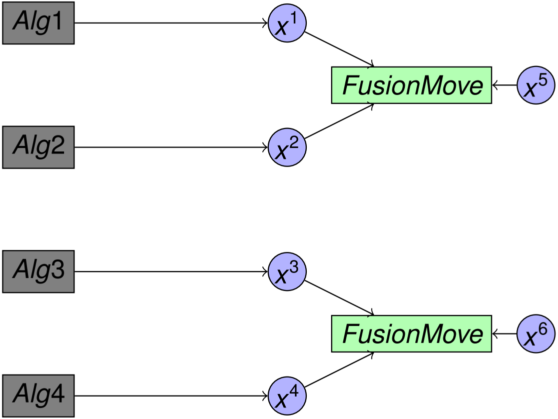
- Run many different algorithms / proposal generators in parallel

Fusion Moves For Parallelization



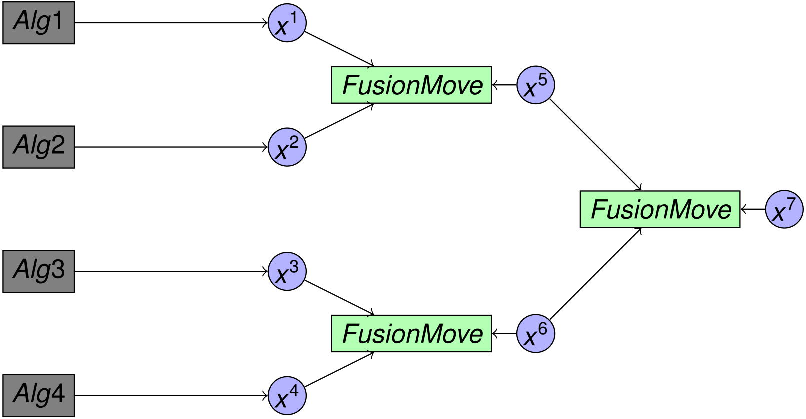
- Run many different algorithms / proposal generators in parallel
- Hierarchically fuse them

Fusion Moves For Parallelization



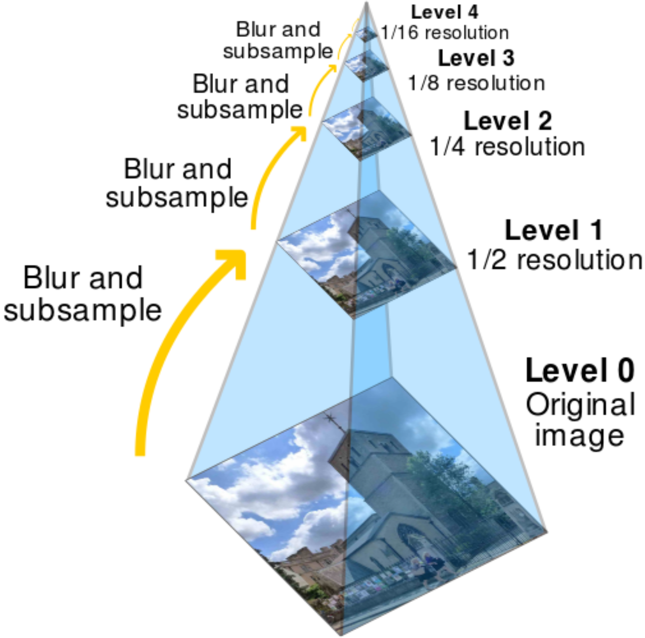
- Run many different algorithms / proposal generators in parallel
- Hierarchically fuse them
- *[Lempitsky* et al.*, 2010]*

Fusion Moves For Parallelization



- Run many different algorithms / proposal generators in parallel
- Hierarchically fuse them
- *[Lempitsky et al., 2010]*

Multi Scale Methods



Multi Scale Methods *[Bagon and Galun, 2012]*

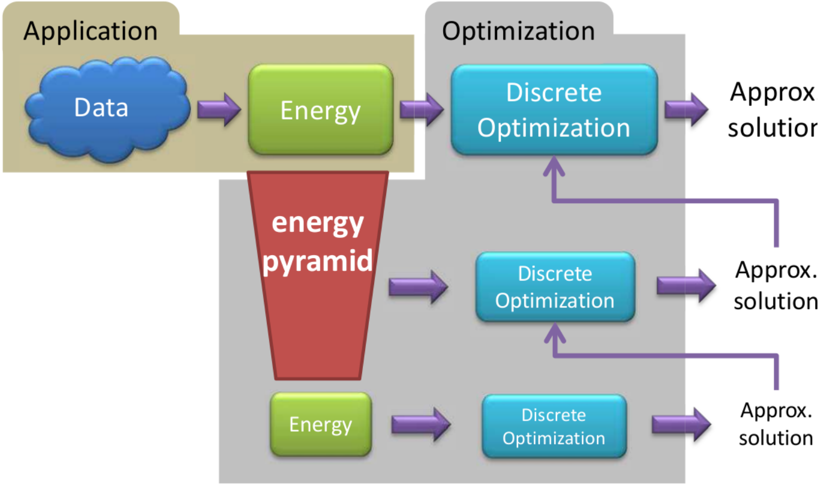
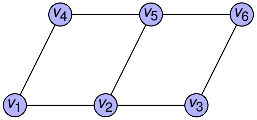


Image Credit: *[Bagon and Galun, 2012]*

Multi Scale Methods

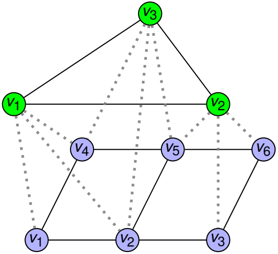
- build energy pyramid



- *[Bagon and Galun, 2012]*

Multi Scale Methods

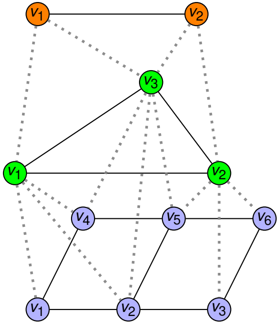
- build energy pyramid



- *[Bagon and Galun, 2012]*

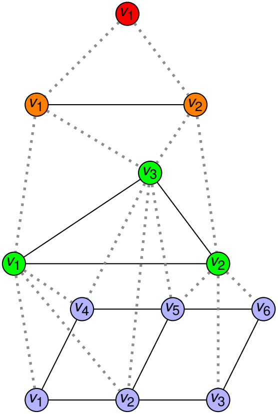
Multi Scale Methods

- build energy pyramid



- [Bagon and Galun, 2012]

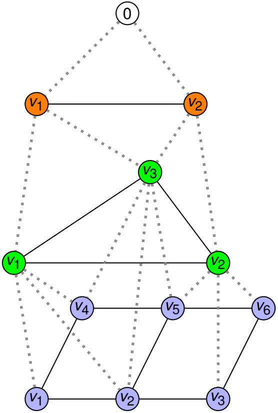
Multi Scale Methods



- build energy pyramid

- [Bagon and Galun, 2012]

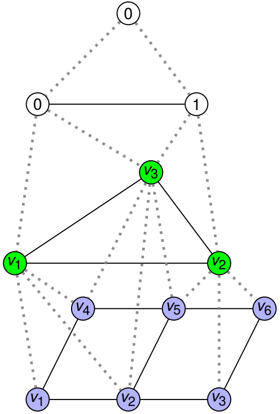
Multi Scale Methods



- build energy pyramid
- Optimize top down

- *[Bagon and Galun, 2012]*

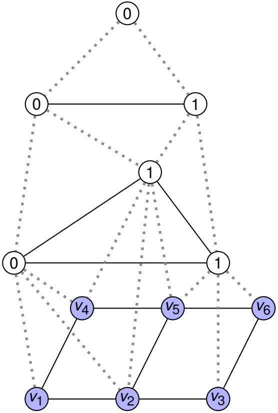
Multi Scale Methods



- build energy pyramid
- Optimize top down
- Warm start with solution from layer above

- *[Bagon and Galun, 2012]*

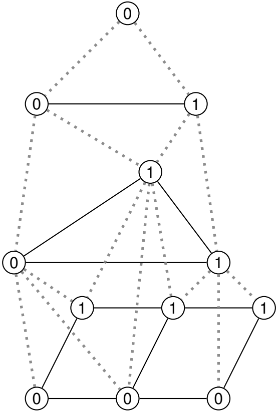
Multi Scale Methods



- build energy pyramid
- Optimize top down
- Warm start with solution from layer above

- [Bagon and Galun, 2012]

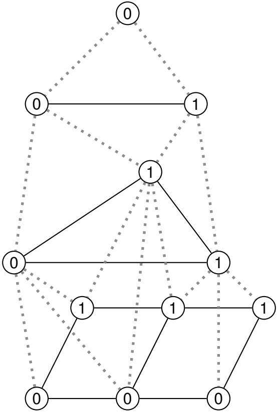
Multi Scale Methods



- build energy pyramid
- Optimize top down
- Warm start with solution from layer above

- *[Bagon and Galun, 2012]*

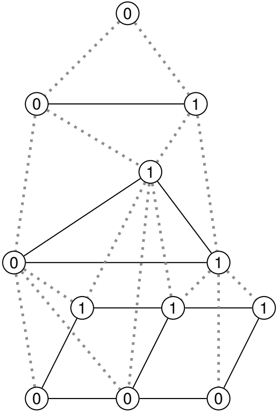
Multi Scale Methods



- build energy pyramid
- Optimize top down
- Warm start with solution from layer above

- *[Bagon and Galun, 2012]*

Multi Scale Methods



- build energy pyramid
- Optimize top down
- Warm start with solution from layer above

ICM Single Scale: ICM Multi Scale:

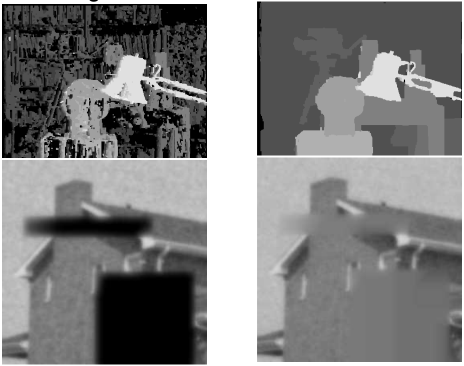
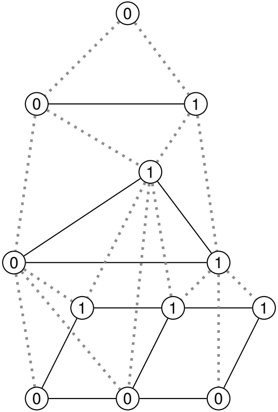


Image Credit: *[Bagon and Galun, 2012]*

- *[Bagon and Galun, 2012]*

Multi Scale Methods



- build energy pyramid
- Optimize top down
- Warm start with solution from layer above

- [Bagon and Galun, 2012]
- [Meir et al., 2015]

Insights from Benchmark Studies

Benchmarks for Graphical Models

- Middlebury MRF [Szeliski et al., 2008]
- Probabilistic Inference Challenge 2011
  `http://www.cs.huji.ac.il/project/PASCAL/`
- OpenGM Benchmark [Kappes et al., 2015]

Benchmarks for Graphical Models

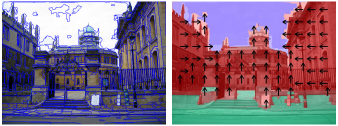
- Middlebury MRF *[Szeliski et al., 2008]*
- Probabilistic Inference Challenge 2011
  `http://www.cs.huji.ac.il/project/PASCAL/`
- OpenGM Benchmark *[Kappes et al., 2015]*

Evaluations from other communities:

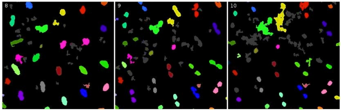
- MAX-CSP 2008 Competition
  `http://www.cril.univ-artois.fr/CPAI08/results/results.php?idev=16`
- Max-SAT Evaluation(s)
  `http://maxsat.ia.udl.cat/introduction/`

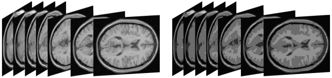
OpenGM Datasets: Overview

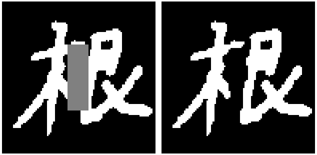

(a) Pixel-based Models


(b) Superpixel-based Models

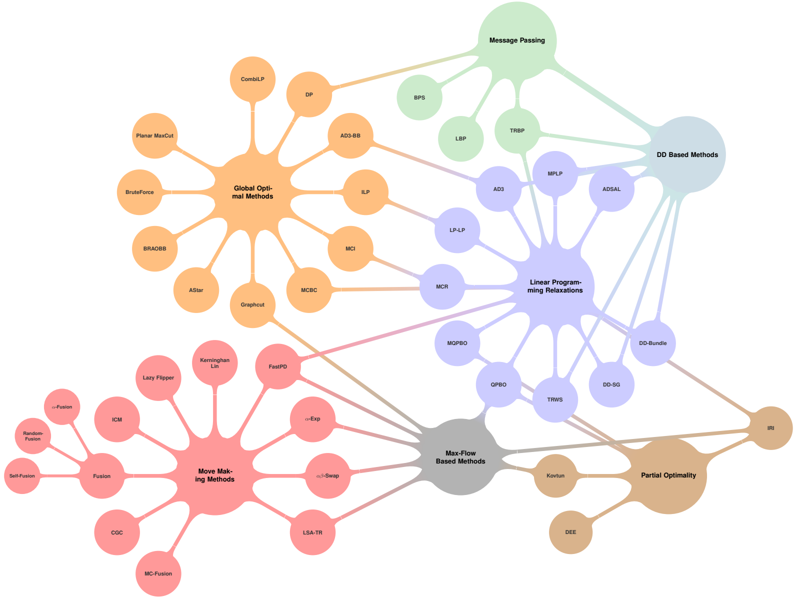

(c) Unsupervised Partitioning

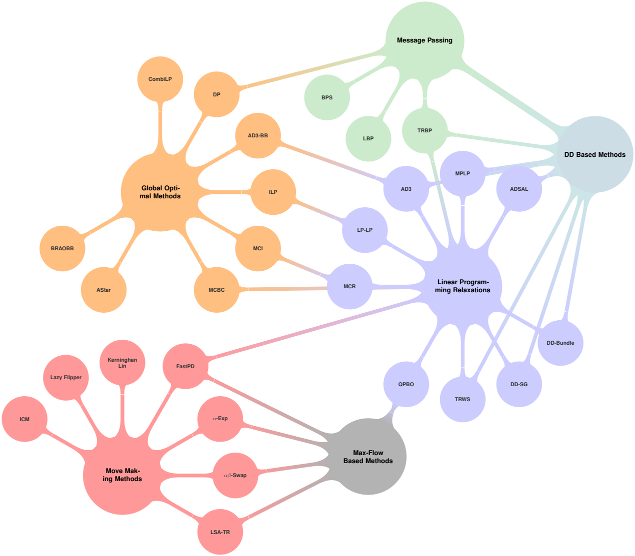

(d) Higher-order Models


(e) Large-scale Models


(f) Small but hard models

. . . 32 datasets, over 2000 problem instances


Message Passing
BPS
LBP
TRBP
DD Based Methods
CombiLP
DP
Planar MaxCut
AD3-BB
BruteForce
Global Optimal Methods
ILP
BRAOBB
MCI
AStar
Graphcut
MCBC
AD3
MPLP
ADSAL
LP-LP
MCR
Linear Programming Relaxations
MQPBO
DD-Bundle
QPBO
TRWS
DD-SG
Kerninghan Lin
Lazy Flipper
FastPD
α-Fusion
ICM
α-Exp
Random-Fusion
Self-Fusion
Fusion
Move Making Methods
αβ-Swap
Max-Flow Based Methods
Kovtun
Partial Optimality
IRI
CGC
LSA-TR
DEE
MC-Fusion


Message Passing
BPS
LBP
TRBP
DD Based Methods
CombiLP
DP
AD3-BB
Global Opti-
mal Methods
ILP
MCI
MCBC
BRAOBB
AStar
AD3
MPLP
ADSAL
LP-LP
MCR
Linear Program-
ming Relaxations
DD-Bundle
DD-SG
TRWS
QPBO
Max-Flow
Based Methods
FastPD
Kerninghan
Lin
Lazy Flipper
ICM
α-Exp
αβ-Swap
Move Mak-
ing Methods
LSA-TR

Typical Result Table

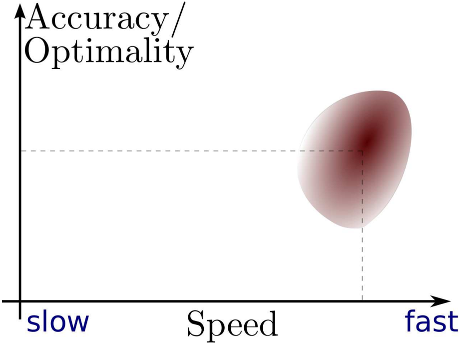
algorithm	runtime	value	bound	mem	best	opt	accuracy
α-Exp-QPBO	**0.01** sec	−866.85	$-\infty$	**0.01**	587	0	0.7694
ogm-LBP-LF2	0.06 sec	−866.76	$-\infty$	**0.01**	576	0	0.7699
ogm-LF-3	0.45 sec	−866.27	$-\infty$	**0.01**	420	0	0.7699
ogm-TRWS-LF2	**0.01** sec	**−866.93**	**−866.93**	**0.01**	714	712	0.7693
BPS-TAB	0.10 sec	−866.73	$-\infty$	**0.01**	566	0	0.7701
ogm-BPS	0.02 sec	−866.77	$-\infty$	**0.01**	585	0	0.7694
ogm-LBP-0.95	0.02 sec	−866.76	$-\infty$	**0.01**	580	0	0.7696
ogm-TRBP-0.95	0.11 sec	−866.84	$-\infty$	**0.01**	644	0	**0.7708**
ogm-TRBPS	0.13 sec	−866.79	$-\infty$	**0.01**	644	0	0.7705
ADDD	0.06 sec	−866.92	**−866.93**	**0.01**	701	697	0.7693
MPLP	0.04 sec	−866.91	**−866.93**	**0.01**	700	561	0.7693
MPLP-C	0.04 sec	−866.92	**−866.93**	**0.01**	710	567	0.7693
ogm-ADSAL	0.04 sec	**−866.93**	**−866.93**	**0.01**	714	712	0.7693
ogm-BUNDLE-H	0.26 sec	**−866.93**	**−866.93**	**0.01**	**715**	673	0.7693
ogm-BUNDLE-A+	0.07 sec	**−866.93**	**−866.93**	**0.01**	**715**	712	0.7693
ogm-LP-LP	0.23 sec	−866.92	**−866.93**	0.05	712	712	0.7693
TRWS-TAB	**0.01** sec	**−866.93**	**−866.93**	**0.01**	714	712	0.7693
BRAOBB-1	17.61 sec	−866.90	$-\infty$	0.27	670	0	0.7688
ADDD-BB	0.11 sec	**−866.93**	**−866.93**	**0.01**	**715**	**715**	0.7693
ogm-CombiLP	0.02 sec	**−866.93**	**−866.93**	0.03	**715**	**715**	0.7693
ogm-ILP	0.17 sec	**−866.93**	**−866.93**	0.09	**715**	**715**	0.7693

Typical Result Table

algorithm	runtime	value	bound	mem	best	opt	accuracy
α-Exp-QPBO	**0.01** sec	−866.85	$-\infty$	**0.01**	587	0	0.7694
ogm-LBP-LF2	0.06 sec	−866.76	$-\infty$	**0.01**	576	0	0.7699
ogm-LF-3	0.45 sec	−866.27	$-\infty$	**0.01**	420	0	0.7699
ogm-TRWS-LF2	**0.01** sec	**−866.93**	**−866.93**	**0.01**	714	712	0.7693
BPS-TAB	0.10 sec	−866.73	$-\infty$	**0.01**	566	0	0.7701
ogm-BPS	0.02 sec	−866.77	$-\infty$	**0.01**	585	0	0.7694
ogm-LBP-0.95	0.02 sec	−866.76	$-\infty$	**0.01**	580	0	0.7696
ogm-TRBP-0.95	0.11 sec	−866.84	$-\infty$	**0.01**	644	0	**0.7708**
ogm-TRBPS	0.13 sec	−866.79	$-\infty$	**0.01**	644	0	0.7705
ADDD	0.06 sec	−866.92	**−866.93**	**0.01**	701	697	0.7693
MPLP	0.04 sec	−866.91	**−866.93**	**0.01**	700	561	0.7693
MPLP-C	0.04 sec	−866.92	**−866.93**	**0.01**	710	567	0.7693
ogm-ADSAL	0.04 sec	**−866.93**	**−866.93**	**0.01**	714	712	0.7693
ogm-BUNDLE-H	0.26 sec	**−866.93**	**−866.93**	**0.01**	**715**	673	0.7693
ogm-BUNDLE-A+	0.07 sec	**−866.93**	**−866.93**	**0.01**	**715**	712	0.7693
ogm-LP-LP	0.23 sec	−866.92	**−866.93**	0.05	712	712	0.7693
TRWS-TAB	**0.01** sec	**−866.93**	**−866.93**	**0.01**	714	712	0.7693
BRAOBB-1	17.61 sec	−866.90	$-\infty$	0.27	670	0	0.7688
ADDD-BB	0.11 sec	**−866.93**	**−866.93**	**0.01**	**715**	**715**	0.7693
ogm-CombiLP	0.02 sec	**−866.93**	**−866.93**	0.03	**715**	**715**	0.7693
ogm-ILP	0.17 sec	**−866.93**	**−866.93**	0.09	**715**	**715**	0.7693

How to select the best method for my problem (without checking all methods)?

Inference Requirements



Model Characterization

Model Characterization:

Size:

- # variables
- # factors
- # labels
- model order

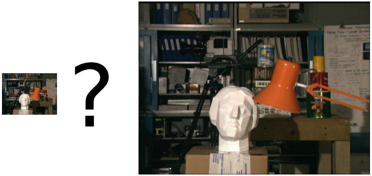
?

Model Characterization

Model Characterization:

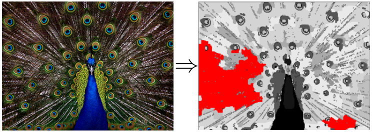
Size:

- # variables
- # factors
- # labels
- model order



Reduction possibility (partial optimality):

- pairwise Potts
- binary pairwise - QPBO
- binary higher order
- pairwise with tight LP relaxation
- low tree-width sub-graphs (junction-tree algorithm)



Model Characterization

Model Characterization:

Size:

- # variables
- # factors
- # labels
- model order

?

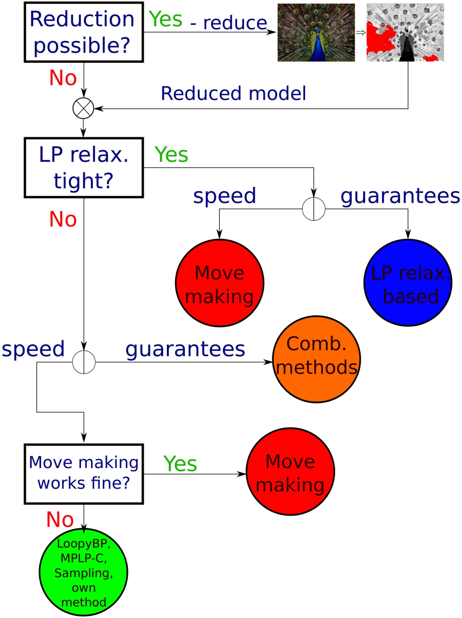
Reduction possibility (partial optimality):

- pairwise Potts
- binary pairwise - QPBO
- binary higher order
- pairwise with tight LP relaxation
- low tree-width sub-graphs (junction-tree algorithm)

⇒

- Tightness of LP relaxation.

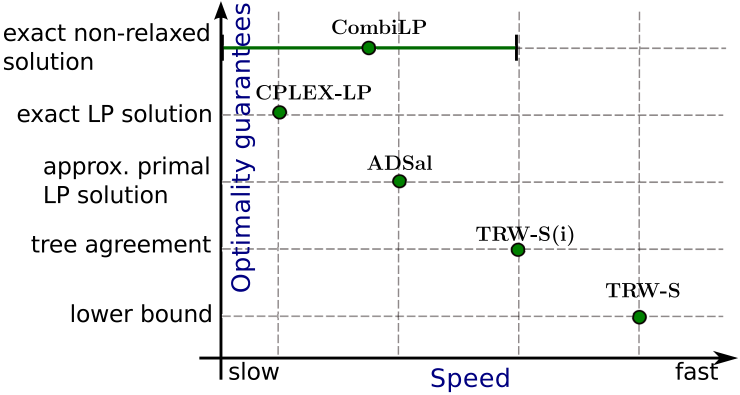
Inference Method Selection



LP-Relaxation Based Methods:

LP relax. tight?

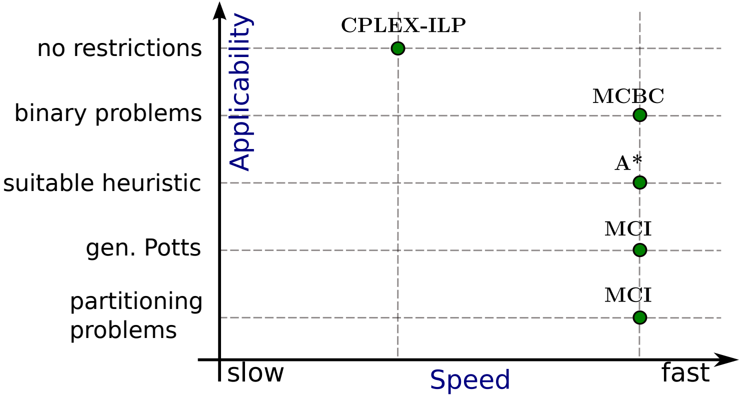
LP relax. based



- Use **SRMP** for higher order instead of TRW-S
- **CPLEX-LP** applicable for any order

Combinatorial Methods:

Comb. methods

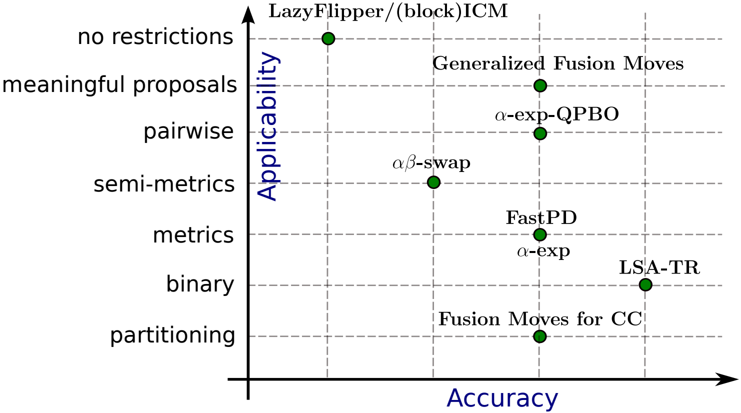


MCI is used with different workflows

Move Making Methods:

Move making works fine?

Move making



FastPD is typically much faster than α-exp

Inference Method Selection

OpenGM2 Library • Contacts • Algorithms • References • **Benchmarks:** CVPR 2013 ARXIV 2014 IJCV 2015 MORE •

OpenGM Benchmark (IJCV 2015)

Benchmark database of discrete energy minimization problems. For further details see

Jörg H. Kappes, Bjoern Andres, Fred A. Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Thorben Kroeger, Bernhard X. Kausler, Jan Lellmann, Bogdan Savchynskyy, Nikos Komodakis, Carsten Rother :
"A Comparative Study of Modern Inference Techniques for Discrete Energy Minimization Problems",
International Journal of Computer Vision 2015,
[publisher], [preprint], [sublementary material 1][sublementary material 2], [bib]

Download Code and Data

- **Optimization methods provided within OpenGM 2.3.3**
- **Links to download the models are given below.**

Models

dataset (zip) model description (pdf) evaluation (html)

		Variables	Labels	Order	Structure	Functions	Instances	Reference	Comment
	**In-Painting (N4)** *J. Lellmann et.al.* *converted by J. Lellmann and J.H. Kappes*	14400	4	2	grid4	potts	2	[57]	
	**In-Painting (N8)** *J. Lellmann et.al.* *converted by J. Lellmann and J.H. Kappes*	14400	4	2	grid8	potts	2	[57]	
	**ColorSegmentation (N4)** *J. Lellmann et.al.* *converted by J. Lellmann and J.H. Kappes*	76800	3-12	2	grid4	potts	2	[57]	
	**ColorSegmentation (N8)** *J. Lellmann et.al.* *converted by J. Lellmann and J.H. Kappes*	76800	3-12	2	grid8	potts	2	[57]	

We are looking for your data and code!

Huge Models and Higher-Order Potentials?

What make problems hard?

What make problems hard?

1. Inherent Combinatorial Complexity
   - LP-relaxations are not tight
   - Local optimal decisions do not lead to global optimal decisions.

What make problems hard?

1. Inherent Combinatorial Complexity
   - LP-relaxations are not tight
   - Local optimal decisions do not lead to global optimal decisions.

2. Higher-Order Factors
   - A factor of order N has L^N entries, which all have to be explored when no additional information is given.

What make problems hard?

1. Inherent Combinatorial Complexity
   - LP-relaxations are not tight
   - Local optimal decisions do not lead to global optimal decisions.

2. Higher-Order Factors
   - A factor of order N has L^N entries, which all have to be explored when no additional information is given.

3. Huge Number of Variables
   - Memory requirements can quickly become very huge

What make problems hard?

1. Inherent Combinatorial Complexity
   - LP-relaxations are not tight
   - Local optimal decisions do not lead to global optimal decisions.

2. Higher-Order Factors
   - A factor of order N has L^N entries, which all have to be explored when no additional information is given.

3. Huge Number of Variables
   - Memory requirements can quickly become very huge

4. Huge Label-Spaces
   - A second-order factor with 10.000 states per variable has 10^8 entries, which all has to be explored when no additional information is given.

Inherent Combinatorial Complexity

Inherent Combinatorial Complexity

When does (real) combinatorial problem show up?

Inherent Combinatorial Complexity

When does (real) combinatorial problem show up?

- When factors/functions encode constraints
- When we learn a model with many parameters $\rightarrow$ over-fitting

$\rightarrow$ When a set of factors/functions conflicting (frustrated cycles)

Inherent Combinatorial Complexity

When does (real) combinatorial problem show up?

- When factors/functions encode constraints
- When we learn a model with many parameters $\rightarrow$ over-fitting

$\rightarrow$ When a set of factors/functions conflicting (frustrated cycles)

Examples

- Clustering (consistency constraint) *[Andres* et al.*, 2011]*
- Graph Matching with weak local assignments (1-to-1 constraint) *[Torresani* et al.*, 2008, Komodakis and Paragios, 2008]*
- Decision Tree Fields *[Nowozin* et al.*, 2011]*
- Vector Compression *[Babenko and Lempitsky, 2014]*

Inherent Combinatorial Complexity

Inherent Combinatorial Complexity

Problem

1. LP-relaxations are not tight
2. Local optimal decisions do not lead to global optimal decisions.

Inherent Combinatorial Complexity

Problem

1. LP-relaxations are not tight
2. Local optimal decisions do not lead to global optimal decisions.

Consequence

1. The rounding problem for methods based on LP-relaxation is very hard. ILP-methods have to deal with weak bounds
2. Move-making methods will walk into wrong directions

Inherent Combinatorial Complexity

Problem

1. LP-relaxations are not tight
2. Local optimal decisions do not lead to global optimal decisions.

Consequence

1. The rounding problem for methods based on LP-relaxation is very hard. ILP-methods have to deal with weak bounds
2. Move-making methods will walk into wrong directions

Ways Out

1. Problem specific Constraints/Separation
   [Nowozin and Lampert, 2009, Kappes et al., 2015b]
2. Make moves over *"meaningful"* subsets
   [Gorelick et al., 2014b, Kappes et al., 2014]
3. Stronger local terms can make the problems easier
4. Try to separate the hard combinatorial parts of the problem
   [Kappes et al., 2013, Kappes et al., 2015b]

Higher-Order Factors

Higher-Order Factors

Complexity of Inference in a Factor Graph Model

$$\Theta\left(\max_{f\in F}\prod_{u\in\mathrm{ne}(f)}|X_u|\right)\approx\Theta(L^{o(G)})$$

where L is the number of labels and $o(G)$ the order of the model

Higher-Order Factors

Complexity of Inference in a Factor Graph Model

$$\Theta\left(\max_{f \in F} \prod_{u \in \mathrm{ne}(f)} |X_u|\right) \approx \Theta(L^{o(G)})$$

where L is the number of labels and $o(G)$ the order of the model

Can we do better?

Higher-Order Factors

Complexity of Inference in a Factor Graph Model

$$\Theta\left(\max_{f\in F}\prod_{u\in\mathrm{ne}(f)}|X_u|\right)\approx\Theta(L^{o(G)})$$

where L is the number of labels and $o(G)$ the order of the model

Can we do better?

Yes, by using structure of functions.

Higher-Order Factors

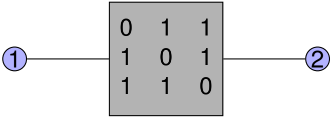
Complexity of Inference in a Factor Graph Model

$$\Theta\left(\max_{f \in F} \prod_{u \in \mathrm{ne}(f)} |X_u|\right) \approx \Theta(L^{o(G)})$$

where L is the number of labels and $o(G)$ the order of the model

Can we do better?

Yes, by using structure of functions.



$\sum_{l \in L} \sum_{l' \in L} \theta_{l\,l'} \cdot \mathbb{I}(x_1 = l \wedge x_2 = l')$

$O(L^2)$

Higher-Order Factors

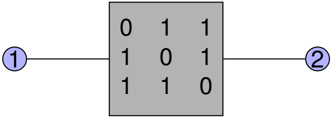
Complexity of Inference in a Factor Graph Model

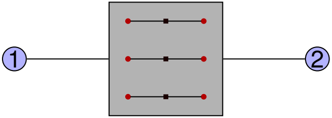
$$\Theta\left(\max_{f\in F}\prod_{u\in \mathrm{ne}(f)}|X_u|\right)\approx\Theta(L^{o(G)})$$

where L is the number of labels and $o(G)$ the order of the model

Can we do better?

Yes, by using structure of functions.





$\sum_{l\in L}\sum_{l'\in L}\theta_{l\,l'}\cdot\mathbb{I}(x_1=l\wedge x_2=l')$

$1-\sum_{l\in L}\mathbb{I}(x_1=l\wedge x_2=l)$

$O(L^2)$

$O(L)$

Reformulation of Higher order Factors

General Way to Reformation in LPs

- $\prod_{v \in \mathrm{ne}(f)} |X_v|$ slack variables $(\sim L^o)$
- $\sum_{v \in \mathrm{ne}(f)} |X_v|$ constraints of size $|X_{\mathrm{ne}(f)\setminus v}|$ $(\sim L \cdot o \times L^{o-1})$

  $L =$ number of labels, $o =$ order of factor

Reformulation of Higher order Factors

General Way to Reformation in LPs

- $\prod_{v \in \mathrm{ne}(f)} |X_v|$ slack variables ($\sim L^o$)
- $\sum_{v \in \mathrm{ne}(f)} |X_v|$ constraints of size $|X_{\mathrm{ne}(f) \setminus v}|$ ($\sim L \cdot o \times L^{o-1}$)

  $L =$ number of labels, $o =$ order of factor

Idea

Replace a higher-order term by some slack variables + additional constraints.

Reformulation of Higher order Factors

General Way to Reformation in LPs

- $\prod_{v \in \mathrm{ne}(f)} |X_v|$ slack variables ($\sim L^o$)
- $\sum_{v \in \mathrm{ne}(f)} |X_v|$ constraints of size $|X_{\mathrm{ne}(f)\setminus v}|$ ($\sim L \cdot o \times L^{o-1}$)

  $L =$ number of labels, $o =$ order of factor

Idea

Replace a higher-order term by some slack variables + additional constraints.

Example: Sparse Function *[Rother et al., 2009, Kappes et al., 2015a]*

$$\varphi(x_{1,2,3,4}) = \begin{cases} \gamma & \text{if } x_{1,2,3,4} = (2,6,2,4) \\ 0 & \text{else} \end{cases}$$

Reformulation of Higher order Factors

General Way to Reformation in LPs

- $\prod_{v\in \text{ne}(f)} |X_v|$ slack variables ($\sim L^o$)
- $\sum_{v\in \text{ne}(f)} |X_v|$ constraints of size $|X_{\text{ne}(f)\setminus v}|$ ($\sim L \cdot o \times L^{o-1}$)

  $L =$ number of labels, $o =$ order of factor

Idea

Replace a higher-order term by some slack variables + additional constraints.

Example: Sparse Function *[Rother et al., 2009, Kappes et al., 2015a]*

$$\varphi(x_{1,2,3,4}) = \begin{cases} \gamma & \text{if } x_{1,2,3,4} = (2,6,2,4) \\ 0 & \text{else} \end{cases}$$

$$= \min_s \gamma \cdot s \quad \text{s.t.} \quad \begin{array}{l} s \leq \mathbb{I}(x_1 = 2) \\ s \leq \mathbb{I}(x_2 = 6) \\ s \leq \mathbb{I}(x_3 = 2) \\ s \leq \mathbb{I}(x_4 = 4) \\ s \geq \mathbb{I}(x_1 = 2) + \mathbb{I}(x_2 = 6) + \mathbb{I}(x_3 = 2) + \mathbb{I}(x_4 = 4) - 3 \end{array}$$

Reformulation of Higher order Factors

General Way to Reformation in LPs

- $\prod_{v \in \mathrm{ne}(f)} |X_v|$ slack variables ($\sim L^o$)
- $\sum_{v \in \mathrm{ne}(f)} |X_v|$ constraints of size $|X_{\mathrm{ne}(f)\setminus v}|$ ($\sim L \cdot o \times L^{o-1}$)

  $L =$ number of labels, $o =$ order of factor

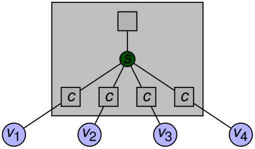
Idea

Replace a higher-order term by some slack variables + additional constraints.

Example: Sparse Function *[Rother et al., 2009, Kappes et al., 2015a]*

$$\varphi(x_{1,2,3,4}) = \begin{cases} \gamma & \text{if } x_{1,2,3,4} = (2,6,2,4) \\ 0 & \text{else} \end{cases}$$

$$= \min_s \gamma \cdot s \quad \text{s.t.} \quad \begin{aligned} s &\leq \mathbb{I}(x_1 = 2) \\ s &\leq \mathbb{I}(x_2 = 6) \\ s &\leq \mathbb{I}(x_3 = 2) \\ s &\leq \mathbb{I}(x_4 = 4) \\ s &\geq \mathbb{I}(x_1 = 2) + \mathbb{I}(x_2 = 6) + \mathbb{I}(x_3 = 2) + \mathbb{I}(x_4 = 4) - 3 \end{aligned}$$



requires that $\gamma \leq 0$

Reformulation of Higher order Factors

General Way to Reformation in LPs

- $\prod_{v\in \mathrm{ne}(f)} |X_v|$ slack variables ($\sim L^o$)
- $\sum_{v\in \mathrm{ne}(f)} |X_v|$ constraints of size $|X_{\mathrm{ne}(f)\setminus v}|$ ($\sim L \cdot o \times L^{o-1}$)

$L =$ number of labels, $o =$ order of factor

Idea

Replace a higher-order term by some slack variables + additional constraints.

Examples: Label Cost *[Delong et al., 2012]*

$$\varphi^l(x_V) = \begin{cases} \gamma_l & \text{if } \exists v \in V : x_v = l \\ 0 & \text{otherwise} \end{cases}$$

$$\varphi(x_V) = \sum_{l \in L} \varphi^l(x_V)$$

Reformulation of Higher order Factors

General Way to Reformation in LPs

- $\prod_{v \in \mathrm{ne}(f)} |X_v|$ slack variables ($\sim L^o$)
- $\sum_{v \in \mathrm{ne}(f)} |X_v|$ constraints of size $|X_{\mathrm{ne}(f) \setminus v}|$ ($\sim L \cdot o \times L^{o-1}$)

  $L =$ number of labels, $o =$ order of factor

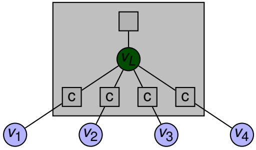
Idea

Replace a higher-order term by some slack variables + additional constraints.

Examples: Label Cost *[Delong et al., 2012]*

$$\varphi^l(x_V) = \begin{cases} \gamma_l & \text{if } \exists v \in V : x_v = l \\ 0 & \text{otherwise} \end{cases}$$

$$\varphi(x_V) = \sum_{l \in L} \varphi^l(x_V)$$



requires that $\gamma_l \geq 0$

Reformulation of Higher order Factors

General Way to Reformation in LPs

- $\prod_{v \in \mathrm{ne}(f)} |X_v|$ slack variables ($\sim L^o$)
- $\sum_{v \in \mathrm{ne}(f)} |X_v|$ constraints of size $|X_{\mathrm{ne}(f)\setminus v}|$ ($\sim L \cdot o \times L^{o-1}$)

  $L =$ number of labels, $o =$ order of factor

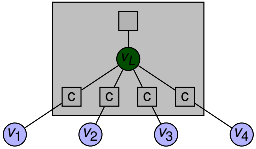
Idea

Replace a higher-order term by some slack variables + additional constraints.

Examples: Label Cost *[Delong et al., 2012]*

$$\varphi^l(x_V) = \begin{cases} \gamma_l & \text{if } \exists v \in V : x_v = l \\ 0 & \text{otherwise} \end{cases}$$

$$\varphi(x_V) = \sum_{l \in L} \varphi^l(x_V)$$



requires that $\gamma_l \geq 0$

For $|L| = 2$ the label costs c_l can be also negative

Reformulation of Higher order Factors

General Way to Reformation in LPs

- $\prod_{v \in \mathrm{ne}(f)} |X_v|$ slack variables $(\sim L^o)$
- $\sum_{v \in \mathrm{ne}(f)} |X_v|$ constraints of size $|X_{\mathrm{ne}(f)\setminus v}|$ $(\sim L \cdot o \times L^{o-1})$

  $L =$ number of labels, $o =$ order of factor

Idea

Replace a higher-order term by some slack variables + additional constraints.

P^N Potts *[Kohli et al., 2009]*

$$\varphi_f(x_{ne(f)}) = \begin{cases} \gamma_k & \text{if } x_v = k, \forall v \in ne(f) \\ 0 & \text{otherwise} \end{cases}$$

Reformulation of Higher order Factors

General Way to Reformation in LPs

- $\prod_{v\in \text{ne}(f)} |X_v|$ slack variables ($\sim L^o$)
- $\sum_{v\in \text{ne}(f)} |X_v|$ constraints of size $|X_{\text{ne}(f)\setminus v}|$ ($\sim L \cdot o \times L^{o-1}$)

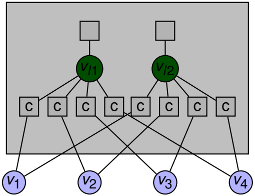
  $L =$ number of labels, $o =$ order of factor

Idea

Replace a higher-order term by some slack variables + additional constraints.

P^N Potts *[Kohli et al., 2009]*

$$\varphi_f(x_{ne(f)}) = \begin{cases} \gamma_k & \text{if } x_v = k, \forall v \in ne(f) \\ 0 & \text{otherwise} \end{cases}$$



requires that $\gamma_k \leq 0, \forall k$

Order Reduction Higher-Order Boolean Functions

$$\varphi : \{0,1\}^N \to \mathbb{R}$$

Order Reduction Higher-Order Boolean Functions

$$\varphi : \{0, 1\}^N \to \mathbb{R}$$

Reducing Negative-Coefficient Monomials *[Freedman and Drineas, 2005]*

$$\varphi(x) = -x_1 \cdots x_N = \min_{y \in \{0,1\}} y \left((N-1) - \sum_{i=1}^{N} x_i \right)$$

Order Reduction Higher-Order Boolean Functions

$$\varphi : \{0,1\}^N \to \mathbb{R}$$

Reducing Negative-Coefficient Monomials *[Freedman and Drineas, 2005]*

$$\varphi(x) = -x_1 \cdots x_N = \min_{y \in \{0,1\}} y \left((N-1) - \sum_{i=1}^{N} x_i \right)$$

Reducing Positive-Coefficient Monomials *[Ishikawa, 2009]*

$$\varphi(x) = x_1 \cdots x_N$$
$$= \begin{cases} \min\limits_{y \in \{0,1\}^N} \sum_{i=1}^{\lfloor N-1/2 \rfloor} y_i \left(1(-\sum_{j=1}^{N} x_j + 2i) \right) + \sum_{i<j} x_i x_j & \text{if } mod(N,2) = 1 \\ \min\limits_{y \in \{0,1\}^N} \sum_{i=1}^{\lfloor N-1/2 \rfloor} y_i \left(2(-\sum_{j=1}^{N} x_j + 2i) \right) + \sum_{i<j} x_i x_j & \text{if } mod(N,2) = 0 \end{cases}$$

Order Reduction Higher-Order Boolean Functions

$$\varphi : \{0,1\}^N \to \mathbb{R}$$

Reducing Negative-Coefficient Monomials *[Freedman and Drineas, 2005]*

$$\varphi(x) = -x_1 \cdots x_N = \min_{y \in \{0,1\}} y \left((N-1) - \sum_{i=1}^{N} x_i \right)$$

Reducing Positive-Coefficient Monomials *[Ishikawa, 2009]*

$$\varphi(x) = x_1 \cdots x_N$$

$$= \begin{cases} \min\limits_{y \in \{0,1\}^N} \sum_{i=1}^{\lfloor N-1/2 \rfloor} y_i \left(1(-\sum_{j=1}^{N} x_j + 2i) \right) + \sum_{i<j} x_i x_j & \text{if } mod(N,2) = 1 \\ \min\limits_{y \in \{0,1\}^N} \sum_{i=1}^{\lfloor N-1/2 \rfloor} y_i \left(2(-\sum_{j=1}^{N} x_j + 2i) \right) + \sum_{i<j} x_i x_j & \text{if } mod(N,2) = 0 \end{cases}$$

[Ishikawa, 2009, Fix et al., 2011, Gallagher et al., 2011]

Huge Number of Variables

Huge Number of Variables

Problems

- Model does not fit in the memory
- Existing solvers does not scale to this problem-size

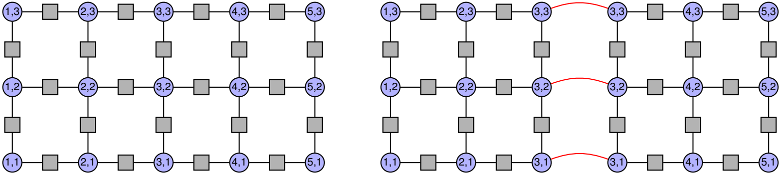
Huge Number of Variables

Problems

- Model does not fit in the memory
- Existing solvers does not scale to this problem-size

Possible Ways Out

- Superpixels/Supervariables
  [Kim et al., 2011]
- Domain-Decomposition (Dual Decomposition)
  [Schwing et al., 2011, Strandmark and Kahl, 2010]



- Block-ICM

Huge Label-Spaces

Huge Label-Spaces

How to deal with huge label-spaces?

Huge Label-Spaces

How to deal with huge label-spaces?

- Buy new hardware

Huge Label-Spaces

How to deal with huge label-spaces?

- Buy new hardware
- Prune the label-space

Huge Label-Spaces

How to deal with huge label-spaces?

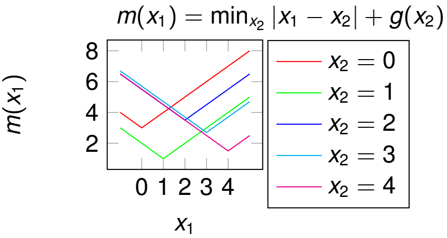
- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*

$$m(x_1) = \min_{x_2} |x_1 - x_2| + g(x_2)$$

Huge Label-Spaces

How to deal with huge label-spaces?

- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*

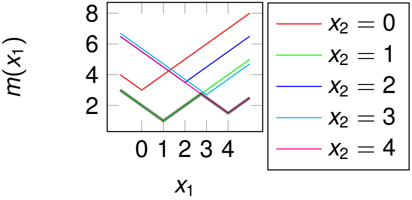


Huge Label-Spaces

How to deal with huge label-spaces?

- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*

$$m(x_1) = \min_{x_2} |x_1 - x_2| + g(x_2)$$



Huge Label-Spaces

How to deal with huge label-spaces?

- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*

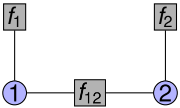
$$x_i \in \{1,\ldots,10\}^3 \qquad\qquad f_{12}(x_1,x_2) = \sum_{i=1}^{3} f_{12}^i(x_{1^i},x_{2^i})$$

Huge Label-Spaces

How to deal with huge label-spaces?

- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*

$$x_i \in \{1, \ldots, 10\}^3 \qquad\qquad f_{12}(x_1, x_2) = \sum_{i=1}^{3} f_{12}^i(x_{1^i}, x_{2^i})$$

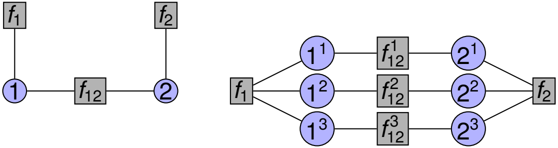


Huge Label-Spaces

How to deal with huge label-spaces?

- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*

$x_i \in \{1, \ldots, 10\}^3$ $\qquad\qquad$ $f_{12}(x_1, x_2) = \sum_{i=1}^{3} f_{12}^{i}(x_{1^i}, x_{2^i})$



Huge Label-Spaces

How to deal with huge label-spaces?

- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*
- Iteratively work on tractable subspaces *[Kappes et al., 2014]*

Huge Label-Spaces

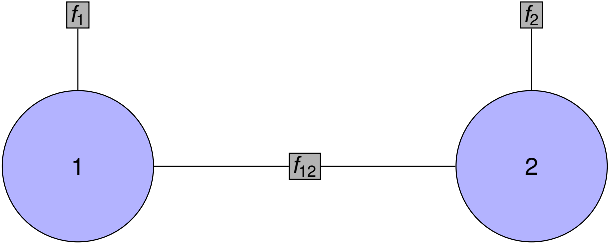
How to deal with huge label-spaces?

- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*
- Iteratively work on tractable subspaces *[Kappes et al., 2014]*
- Particle Inference *[Ihler and McAllester, 2009]*

Huge Label-Spaces

How to deal with huge label-spaces?

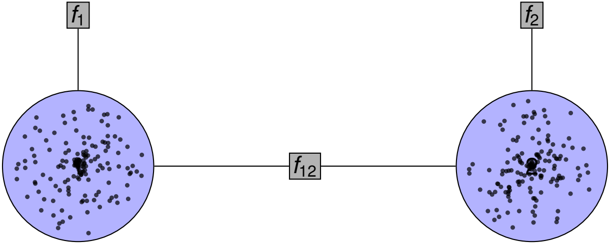
- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*
- Iteratively work on tractable subspaces *[Kappes et al., 2014]*
- Particle Inference *[Ihler and McAllester, 2009]*



Huge Label-Spaces

How to deal with huge label-spaces?

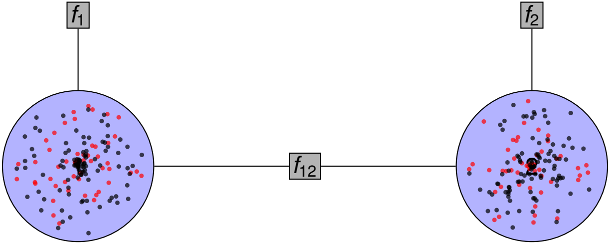
- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*
- Iteratively work on tractable subspaces *[Kappes et al., 2014]*
- Particle Inference *[Ihler and McAllester, 2009]*



Huge Label-Spaces

How to deal with huge label-spaces?

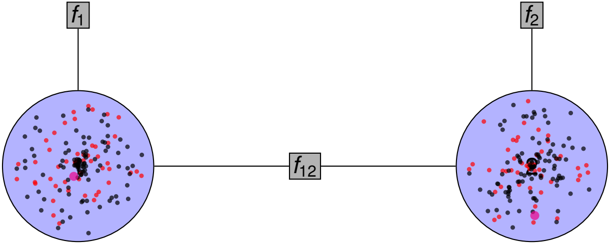
- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*
- Iteratively work on tractable subspaces *[Kappes et al., 2014]*
- Particle Inference *[Ihler and McAllester, 2009]*



Huge Label-Spaces

How to deal with huge label-spaces?

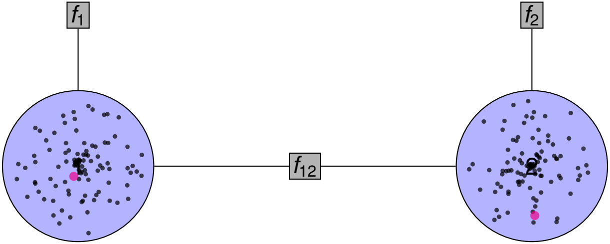
- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*
- Iteratively work on tractable subspaces *[Kappes et al., 2014]*
- Particle Inference *[Ihler and McAllester, 2009]*



Huge Label-Spaces

How to deal with huge label-spaces?

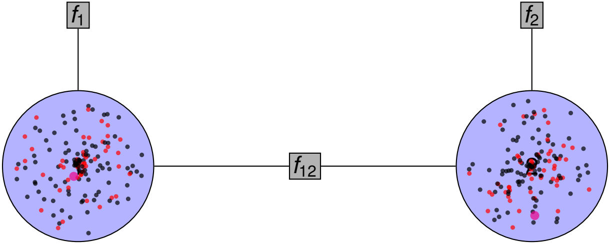
- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*
- Iteratively work on tractable subspaces *[Kappes et al., 2014]*
- Particle Inference *[Ihler and McAllester, 2009]*



Huge Label-Spaces

How to deal with huge label-spaces?

- Buy new hardware
- Prune the label-space
- Efficient updates *[Felzenszwalb and Huttenlocher, 2006]*
- Use structure of label spaces *[Goldluecke and Cremers, 2010]*
- Iteratively work on tractable subspaces *[Kappes et al., 2014]*
- Particle Inference *[Ihler and McAllester, 2009]*



Mixed Models with Continuous Variables

Possible Solvers

- Block ICM (most popular)
- Particle-based Methods
- Others: Discrete-Continuous Fusion Move;
  Gradient Descent with relaxed discrete Variables;
  Variable elimination …

Examples in Computer Vision

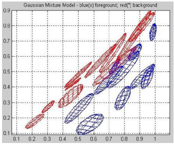
Many Discrete, Few Continuous Variables

Input

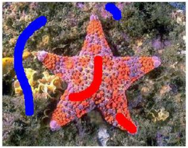
Pose

Segmentation

Segmentation and Human Pose fitting
[Kohli, Rihan, Bray, Torr, IJCV 2008]



GMM

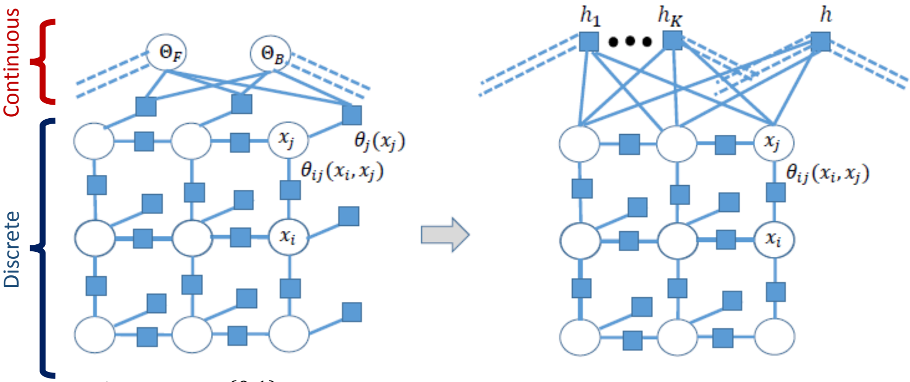


Input

Segmentation

Segmentation and Color Model fitting
GrabCut [Rother, Kolmogorov, Blake 2004]

Illustration Block-ICM



Discrete: $x_i \in \{0,1\}$

Continuous: $\Theta_F \in R^{200}$

(example image segmentation)

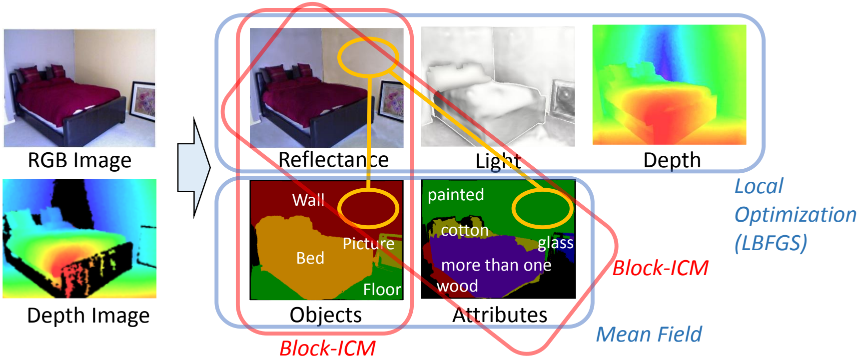
Block-ICM: iteratively update the 2 variable-sets

Example Variable elimination

[Vicente; Kolmogorov, Rother; ICCV 2009]

Examples in Computer Vision

Many Discrete, Many Continuous Variables



Dual Decomposition

Joint Models [Vineet, Rother, Torr, NIPS 2013]

Examples in Computer Vision

Many Continuous Variables

time = 1

time = 2

Motion

Stereo Matching

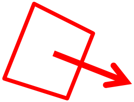
Local stereo matching: check photo-consistency

Stereo Matching

Local stereo matching: check photo-consistency

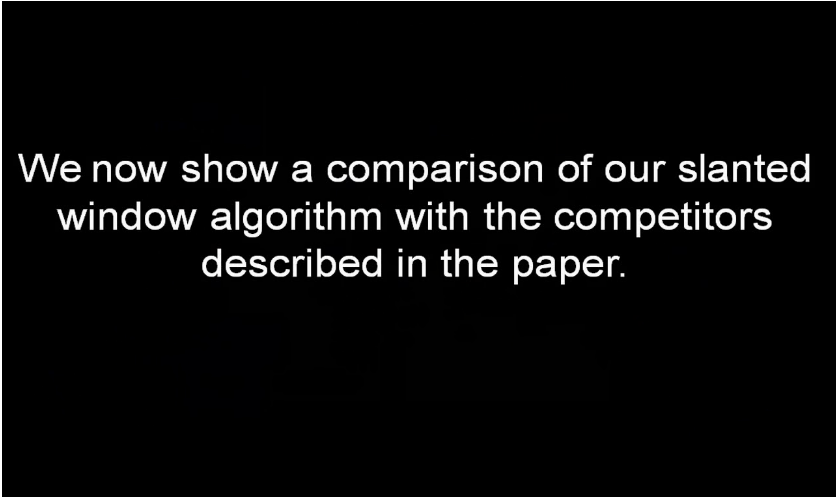
None Front-to-Parallel Surface

None Front-to-Parallel Surface



3 continuous parameters (depth + normal) for each pixel

Motivation

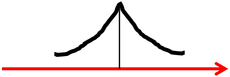

We now show a comparison of our slanted window algorithm with the competitors described in the paper.

PatchMatch Stereo

[Barnes et al. Siggraph 2009]
[Bleyer, Rhemann, Rother. BMVC 2011]

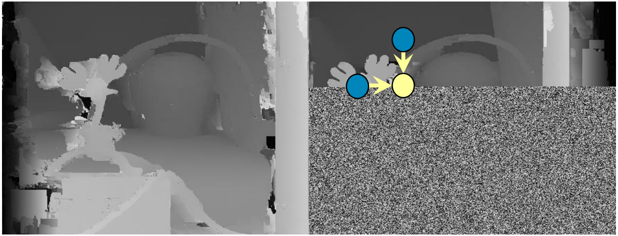
1. Random initialization
2. Go through pixels in sequential order:
    3. Consider solution from left/top neighbour
    4. Sample around current solution

1D example:



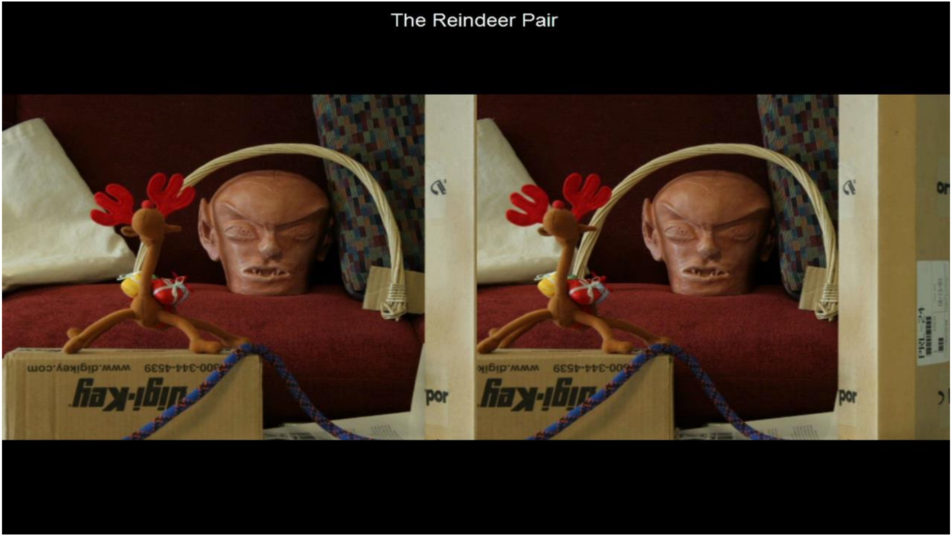
Current solution

Left image



Left and right disparity maps (intermediate step of iteration 1)

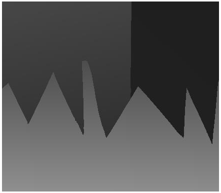
PatchMatch Stereo



Why does it work?

- Random Initialization is in your favour

Left image



Ground truth disparities

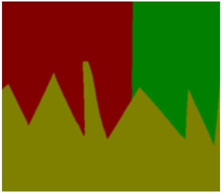
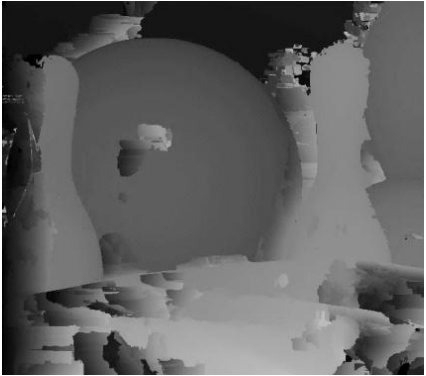


Image consists of 3 planes - ~80.000 guesses for yellow plane

What's Missing



PatchMatch Stereo result

Continuous Variable MRF

Add pairwise terms:

$$E(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \boxed{\sum_{s=1}^{n} \psi_s(\mathbf{u}_s)} + \lambda \boxed{\sum_{s=1}^{n} \left[\sum_{t \in N(s)} \psi_{st}(\mathbf{u}_s, \mathbf{u}_t) \right]}$$

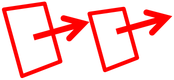
Unary term
(photo-consistency)

Pairwise term
(local curvature)

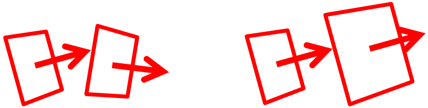
Continuous
3-dimension

Pairwise term

$$\psi_{st}(\boldsymbol{u}_s, \boldsymbol{u}_t) = \underbrace{\omega}_{\beta w_{st}\left(\left|\mathbf{n}_s \cdot (\mathbf{x}_t - \mathbf{x}_s)\right| + \left|\mathbf{n}_t \cdot (\mathbf{x}_s - \mathbf{x}_t)\right|\right)} [\boldsymbol{u}_s \neq \boldsymbol{u}_t]$$



Cost = 0
both planes are aligned in 3D



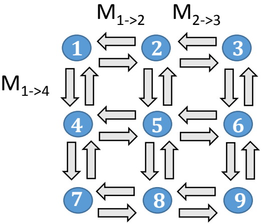
Cost ≠ 0:
local curvature or discontinuity

Solvers

- w/o Pairwise Terms: PatchMatch

- w/ Pairwise Terms (super high-dimensional **u)**:
  - Gradient descent + Fusion move
  - Simulated Annealing
  - Continuous Belief Propagation, e.g. Particle BP

Discrete State BP [Pearl '88]

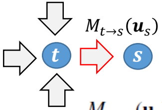


Sequential schedule

Algorithm

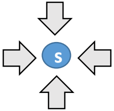
1. Initialize Messages
2. Go over all Messages
   3. Update Message $M_{t\to s}(\boldsymbol{u}_s)$
4. Compute final Output
   $\boldsymbol{u}_s{}^* = \text{argmin } B_s(\boldsymbol{u}_s)$

<u>Step 3: Update Message</u>



$$M_{t\to s}(\mathbf{u}_s) := \min_{\mathbf{u}_t} \ \psi_{st}(\mathbf{u}_s, \mathbf{u}_t) + B_t(\mathbf{u}_t) - M_{s\to t}(\mathbf{u}_t)$$

<u>Step 4: Compute neg-log Belief</u>



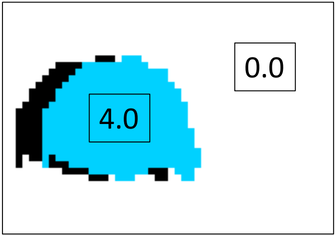
$$B_s(\mathbf{u}_s) := \psi_s(\mathbf{u}_s) + \sum_{t\in N(s)} M_{t\to s}(\mathbf{u}_s)$$

Toy Example – Shift 4.0

Target

Toy Example – Shift 4.0

Source
(shifted 4.0 + noise)



Ground Truth

Toy Example – Shift 4.0

Error: 0.618; Unary only

Ground Truth

Error: 0.251

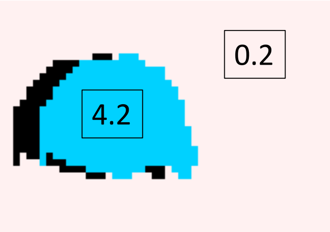
12x12 discrete labels

Toy Example – Shift 4.2

Target

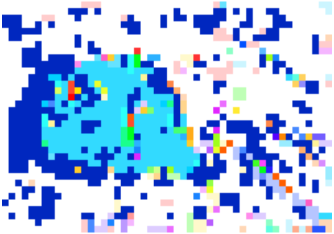
Toy Example – Shift 4.2

Source
(shifted 4.2 + noise)

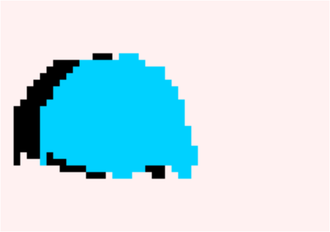


Ground Truth

Toy Example – Shift 4.2



Error: 1.9; unary only

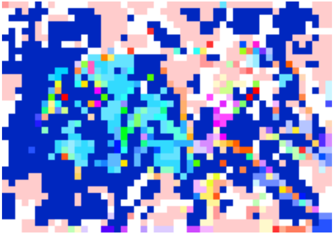


Ground Truth

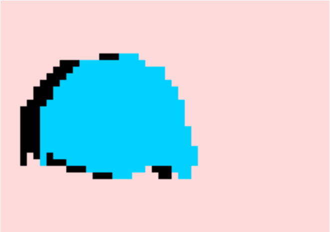
Error: 0.66

12x12 discrete labels

Toy Example – Shift 4.5



Error: 3.46; unary only



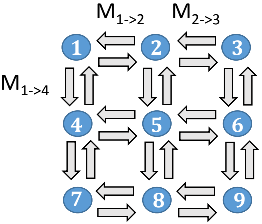
Ground Truth

Error: 5.68

12x12 discrete labels

Max-Product Particle BP

[Kothapa et al. Master thesis]
Sum-Product [Ihler et al. AISTATS '09]



Sequential schedule

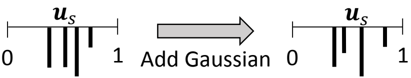
Each pixel has different set of particles:

$B_t(\boldsymbol{u}_t)$, i.e. neg. log Belief $\quad$ $B_s(\boldsymbol{u}_s)$

Algorithm

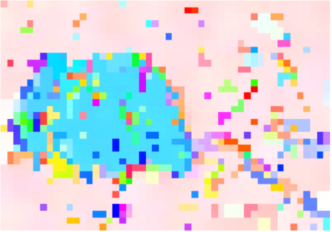
1. Initialize Particles and Messages
2. Go over all Messages
    3. Move Particles $\{\boldsymbol{u}_s\}$
    4. Update Message $M_{t\to s}(\boldsymbol{u}_s)$
5. Compute final Output
    $\boldsymbol{u}_s{}^* = \text{argmin } B_s(\boldsymbol{u}_s)$

Step 3: Move Particles $\{\boldsymbol{u}_s\}$



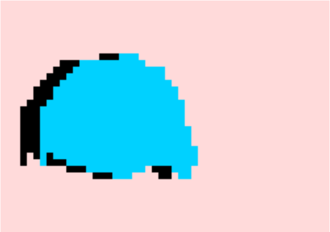
Add Gaussian Noise

Toy Example – Shift 4.5



Energy: 42628 Best initialization

Error: 0.8259

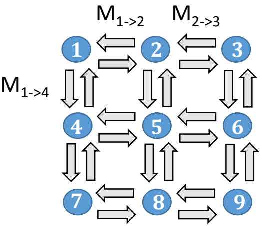


Ground Truth

discrete Error: 5.68

PatchMatch BP

[Besse, Rother, Fitzgibbon, Kautz, BMVC '12]



Sequential schedule

Each pixel has different set of particles:

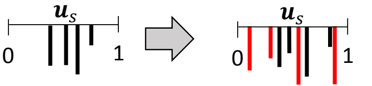
$B_t(\boldsymbol{u}_t)$, i.e. neg. log Belief

$B_s(\boldsymbol{u}_s)$

Algorithm

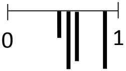
1. Initialize Particles and Messages
2. Go over all Messages
    3. Move and add Particles $\{\boldsymbol{u}_s\}$
    4. Update Message $M_{t\to s}(\boldsymbol{u}_s)$
    5. Take K best Particles wrt Belief $B_s(\boldsymbol{u}_s)$
6. Compute final Output
    $\boldsymbol{u}_s^* = \text{argmin } B_s(\boldsymbol{u}_s)$

Step 3: Move and add Particles $\{\boldsymbol{u}_s\}$



Particles from $\boldsymbol{u}_t$

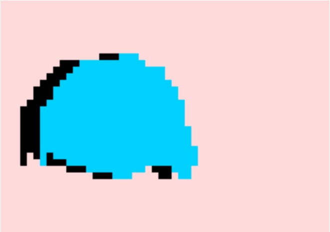
Step 5: Take K best Particles $\{\boldsymbol{u}_s\}$



Good since
$\psi_{st}(\boldsymbol{u}_s, \boldsymbol{u}_t) = \omega\,[\boldsymbol{u}_s \neq \boldsymbol{u}_t]$

Toy Example – Shift 4.5

50 particles Error: 0.4159
Energy: 21959 Random init

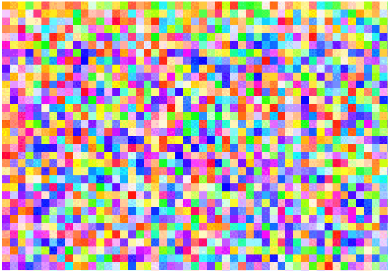


Ground Truth

1 particle Error: 0.3864
Energy: 22593 Random init

Energy: 42628 Particle BP
Error: 0.8259

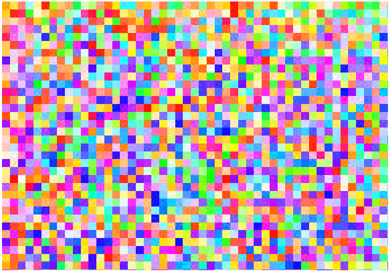
Animation



50 particles

Energy: 21959

Error: 0.4159

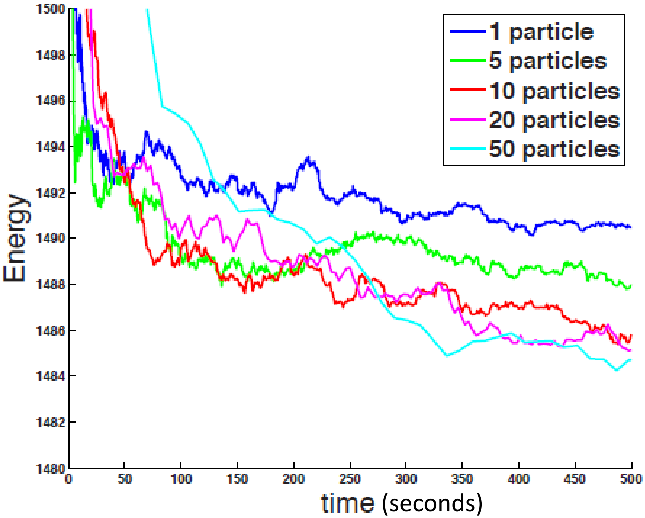


1 particle

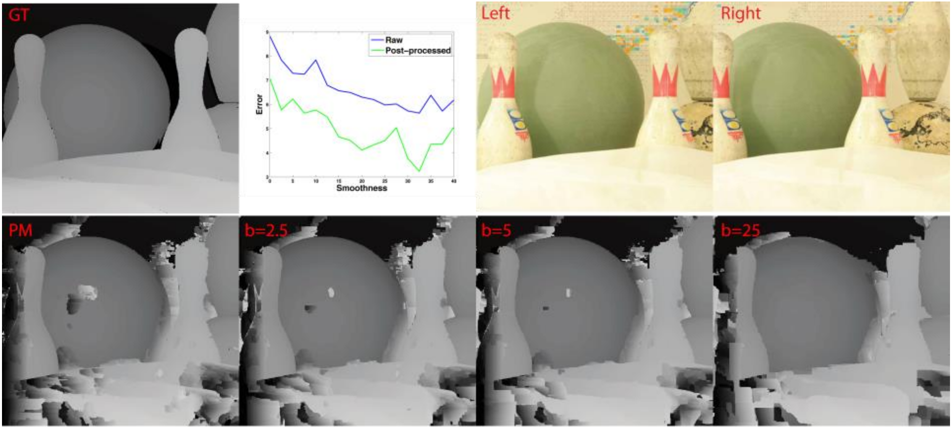
Energy: 22593

Error: 0.3864

Number of Particles



Results



GT
Raw
Post-processed
Error
Smoothness
Left
Right
PM
b=2.5
b=5
b=25

Extension: 6D Scene Flow

[M. Hornacek, A. Fitzgibbon, C. Rother, CVPR 2014]

Left RGBD image

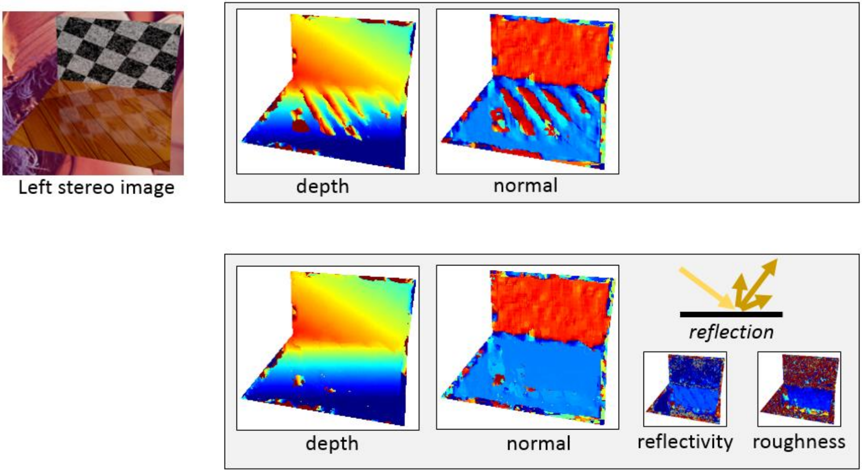
Right RGBD image

Our result

Closest competitor
[Herbst, Ren, Fox, ICRA 2013]

Extension: Reflections on Stereo

[R. Nair, A. Fitzgibbon, D. Kondermann, C. Rother, ICCV 2015]



Related Work

- Different Variant of [Kothapa et al.]. Run full BP and then re-sample particles (no augmentation) [Peng et al. ICML '11]

- Maintain diverse particle set [Pacheco, Zuffi, Black, Sudderth, ICML '14]

- None Particle-based Methods (rarely applied to Computer Vision due to runtime limitations):
  - Nonparametric belief propagation [Sudderth et al. IEEE Intl. Conf. Acoustics, Speech, Signal 2010]
  - Sparse forward-backward [Pal et al. ICASSP 2006]
  - Kernel BP [Song et al. AIStats 2011]
  - Stochastic belief propagate [N. Noorshams and M. J. Wainwright arxiv 2011]

See separate slides in learning.pdf

References I

(2015).
GraphFlow - 6D Large Displacement Scene Flow via Graph Matching.
In GCPR, (Gall, J., Gehler, P. V. and Leibe, B., eds), vol. 9358, of Lecture Notes in Computer Science pp. 285–296, Springer.

Andres, B., Kappes, J. H., Beier, T., Köthe, U. and Hamprecht, F. A. (2011).
Probabilistic Image Segmentation with Closedness Constraints.
In ICCV.

Andres, B., Kappes, J. H., Köthe, U. and Hamprecht, F. A. (2010).
The Lazy Flipper: MAP Inference in Higher-Order Graphical Models by Depth-limited Exhaustive Search.
CoRR *abs/1009.4102*.

Babenko, A. and Lempitsky, V. (2014).
Additive Quantization for Extreme Vector Compression.

Bagon, S. and Galun, M. (2012).
A Unified Multiscale Framework for Discrete Energy Minimization.
CoRR *abs/1204.4867*.

Beier, T., Hamprecht, F. A. and Kappes, J. H. (2015).
Fusion Moves for Correlation Clustering.
In CVPR. Proceedings, in press.

Bergtholdt, M., Kappes, J. H., Schmidt, S. and Schnörr, C. (2010).
A Study of Parts-Based Object Class Detection Using Complete Graphs.
IJCV *87*, 93–117.

Besag, J. (1986).
On the statistical analysis of dirty pictures.
Journal of the Royal Statistical Society. Series B *48*, 259–302.

References II

Bleyer, M., Rhemann, C. and Rother, C. (2011).
PatchMatch Stereo - Stereo Matching with Slanted Support Windows.
In British Machine Vision Conference, BMVC 2011, Dundee, UK, August 29 - September 2, 2011. Proceedings pp. 1–11,.

Boros, E. and Hammer, P. L. (2002).
Pseudo-boolean optimization.
Discrete applied mathematics *123*, 155–225.

Boykov, Y. and Jolly, M. (2001).
Demonstration of Segmentation with Interactive Graph Cuts.
In ICCV p. 741,.

Boykov, Y. and Kolmogorov, V. (2004).
An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision.
IEEE PAMI *26*, 1124–1137.

Chopra, S. and Rao, M. (1993).
The partition problem.
Mathematical Programming *59*, 87–115.

Delong, A., Osokin, A., Isack, H. and Boykov, Y. (2012).
Fast Approximate Energy Minimization with Label Costs.
International Journal of Computer Vision *96*, 1–27.

Elidan, G., McGraw, I. and Koller, D. (2006).
Residual belief propagation: informed scheduling for asynchronous message passing.
In Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI).

Felzenszwalb, P. F. and Huttenlocher, D. P. (2006).
Efficient Belief Propagation for Early Vision.
Int. J. Comput. Vision *70*, 41–54.

References III

Fisher, M. E. (1961).
Statistical Mechanics of Dimers on a Plane Lattice.
Phys. Rev. *124*, 1664–1672.

Fix, A., Gruber, A., Boros, E. and Zabih, R. (2011).
A graph cut algorithm for higher-order Markov Random Fields.
In ICCV.

Freedman, D. and Drineas, P. (2005).
Energy minimization via graph cuts: settling what is possible.
In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on vol. 2, pp. 939–946 vol. 2,.

Gallagher, A. C., Batra, D. and Parikh, D. (2011).
Inference for order reduction in Markov random fields.
In CVPR.

Geman, S. and Geman, D. (1984).
Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images.
IEEE Trans. Pattern Anal. Mach. Intell. *6*, 721–741.

Globerson, A. and Jaakkola, T. (2007).
Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations.
In NIPS.

Globerson, A. and Jaakkola, T. S. (2006).
Approximate inference using planar graph decomposition.
In Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006 pp. 473–480,.

Goldluecke, B. and Cremers, D. (2010).
Convex Relaxation for Multilabel Problems with Product Label Spaces.
In European Conference on Computer Vision.

References IV

Gorelick, L., Boykov, Y., Veksler, O., Ayed, I. B. and Delong, A. (2014a).
Submodularization for Binary Pairwise Energies.
In 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014 pp. 1154–1161,.

Gorelick, L., Veksler, O., Boykov, Y., Ben Ayed, I. and Delong, A. (2014b).
Local Submodular Approximations for Binary Pairwise Energies.
In Computer Vision and Pattern Recognition.

Greig, D. M., Porteous, B. T. and Seheult, A. H. (1989).
Exact maximum a posteriori estimation for binary images.
Journal of the Royal Statistical Society. Series B (Methodological) *51*, 271–279.

Grötschel, M., Lovasz, L. and Schrijver, A. (1981).
The ellipsoid method and its consequences in combinatorial optimization.
Combinatorica *1*, 169–197.

Hammersley, J. M. and Clifford, P. E. (1971).
Markov random fields on finite graphs and lattices.
Unpublished manuscript .

Hazan, T., Maji, S. and Jaakkola, T. S. (2013).
On Sampling from the Gibbs Distribution with Random Maximum A-Posteriori Perturbations.
In Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 1268–1276,.

Hazan, T. and Shashua, A. (2010).
Norm-Product Belief Propagation: Primal-Dual Message-Passing for Approximate Inference.
IEEE Trans. on Inf. Theory, *56*, 6294 –6316.

Ihler, A. and McAllester, D. (2009).
Particle Belief Propagation.
International Conference on Artificial Intelligence and Statistics *5*, 256–263.

References V

Ishikawa, H. (2003).
Exact optimization for Markov random fields with convex priors.
Pattern Analysis and Machine Intelligence, IEEE Transactions on *25*, 1333–1336.

Ishikawa, H. (2009).
Higher-order clique reduction in binary graph cut.
In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on pp. 2993–3000,.

Iyer, R. K., Jegelka, S. and Bilmes, J. A. (2013).
Fast Semidifferential-based Submodular Function Optimization.
In Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013 pp. 855–863,.

Johnson, J. K., Malioutov, D. and Willsky, A. S. (2007).
Lagrangian relaxation for MAP estimation in graphical models.
In 45th Ann. Allerton Conf. on Comm., Control and Comp.

Jug, F., Pietzsch, T., Kainmüller, D., Funke, J., Kaiser, M., van Nimwegen, E., Rother, C. and Myers, G. (2014).
Optimal Joint Segmentation and Tracking of Escherichia Coli in the Mother Machine.
In Bayesian and grAphical Models for Biomedical Imaging - First International Workshop, BAMBI 2014, Cambridge, MA, USA, September 18, 2014, Revised Selected Papers pp. 25–36,.

Jung, K., Kohli, P. and Shah, D. (2009).
Local Rules for Global MAP: When Do They Work ?
In NIPS, (Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I. and Culotta, A., eds), pp. 871–879, Curran Associates, Inc.

Kappes, J. H., Andres, B., Hamprecht, F. A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B. X., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B. and Rother, C. (2015).
A Comparative Study of Modern Inference Techniques for Structured Discrete Energy Minimization Problems.
International Journal of Computer Vision *1*, 1–30.

Kappes, J. H., Beier, T. and Schnörr, C. (2014).
MAP-Inference on Large Scale Higher-Order Discrete Graphical Models by Fusion Moves.
In Computer Vision - ECCV 2014 Workshops - Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part II.

References VI

Kappes, J. H., Savchynskyy, B. and Schnörr, C. (2012).
A Bundle Approach To Efficient MAP-Inference by Lagrangian Relaxation.
In CVPR.

Kappes, J. H., Schmidt, S. and Schnörr, C. (2010).
MRF inference by k-fan decomposition and tight Lagrangian relaxation.
In Computer Vision–ECCV 2010 pp. 735–747. Springer.

Kappes, J. H., Speth, M., Andres, B., Reinelt, G. and Schnörr, C. (2011).
Globally Optimal Image Partitioning by Multicuts.
In EMMCVPR.

Kappes, J. H., Speth, M., Reinelt, G. and Schnörr, C. (2013).
Towards Efficient and Exact MAP-Inference for Large Scale Discrete Computer Vision Problems via Combinatorial Optimization.
In CVPR.

Kappes, J. H., Speth, M., Reinelt, G. and Schnörr, C. (2015a).
Higher-order Segmentation via Multicuts.
CoRR *abs/1305.6387*.

Kappes, J. H., Speth, M., Reinelt, G. and Schnörr, C. (2015b).
Higher-order segmentation via multicuts.
Computer Vision and Image Understanding -, –.

Kasteleyn, P. (1961).
The statistics of dimers on a lattice: I. The number of dimer arrangements on a quadratic lattice.
Physica *27*, 1209 – 1225.

Kim, T., Nowozin, S., Kohli, P. and Yoo, C. (2011).
Variable grouping for energy minimization.
In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on pp. 1913–1920,.

References VII

Kleinberg, J. and Tardos, E. (2002).
Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields.
Journal of the ACM (JACM) *49*, 616–639.

Kohli, P., Kumar, M. P. and Torr, P. H. S. (2007).
P3 & Beyond: Solving Energies with Higher Order Cliques.
In 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA.

Kohli, P., Ladický, L. and Torr, P. H. (2009).
Robust Higher Order Potentials for Enforcing Label Consistency.
Int. J. Comput. Vision *82*, 302–324.

Kohli, P., Shekhovtsov, A., Rother, C., Kolmogorov, V. and Torr, P. (2008).
On partial optimality in multi-label MRFs.
In ICML.

Koller, D. and Friedman, N. (2009).
Probabilistic Graphical Models: Principles and Techniques.
MIT Press.

Kolmogorov, V. (2006).
Convergent Tree-Reweighted Message Passing for Energy Minimization.
PAMI *28*, 1568–1583.

Kolmogorov, V. (2009).
Blossom V: a new implementation of a minimum cost perfect matching algorithm.
Math. Program. Comput. *1*, 43–67.

Kolmogorov, V. (2015).
A new look at reweighted message passing.
Pattern Analysis and Machine Intelligence, IEEE Transactions on *37*, 919–930.

References VIII

Kolmogorov, V. and Rother, C. (2007).
Minimizing Nonsubmodular Functions with Graph Cuts-A Review.
IEEE Trans. Pattern Anal. Mach. Intell. *29*, 1274–1279.

Kolmogorov, V. and Zabih, R. (2002).
What Energy Functions Can Be Minimized via Graph Cuts?
In ECCV.

Komodakis, N. and Paragios, N. (2008).
Beyond Loose LP-Relaxations: Optimizing MRFs by Repairing Cycles.
In ECCV.

Komodakis, N., Paragios, N. and Tziritas, G. (2011).
MRF Energy Minimization and Beyond via Dual Decomposition.
PAMI *33*.

Kovtun, I. (2003).
Partial Optimal Labeling Search for a NP-Hard Subclass of (max,+) Problems.
In DAGM.

Kschischang, F., Frey, B. and Loeliger, H.-A. (2001).
Factor graphs and the sum-product algorithm.
Information Theory, IEEE Transactions on *47*, 498–519.

Lauritzen, S. L. (1996).
Graphical Models.
Oxford University Press, Oxford, UK.

Lempitsky, V. S., Rother, C., Roth, S. and Blake, A. (2010).
Fusion Moves for Markov Random Field Optimization.
IEEE Trans. Pattern Anal. Mach. Intell. *32*, 1392–1405.

References IX

Lin, G., Shen, C., Reid, I. D. and van den Hengel, A. (2015).
Efficient piecewise training of deep structured models for semantic segmentation.
CoRR *abs/1504.01013*.

Meir, O., Galun, M., Yagev, S., Basri, R. and Irad, Y. (2015).
Multiscale Variable-grouping Framework for MRF Energy Minimization.
In Proceedings of the International Conference on Computer Vision (ICCV).

Meshi, O., Globerson, A. and Jaakkola, T. S. (2012).
Convergence Rate Analysis of MAP Coordinate Minimization Algorithms.
In Advances in Neural Information Processing Systems 25, (Pereira, F., Burges, C., Bottou, L. and Weinberger, K., eds), pp. 3014–3022.

Nesterov, Y. (2005).
Smooth minimization of non-smooth functions.
Math. Program. *103*, 127–152.

Nowozin, S. and Lampert, C. (2009).
Global connectivity potentials for random field models.
In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on pp. 818–825,.

Nowozin, S., Rother, C., Bagon, S., Sharp, T., Yao, B. and Kohli, P. (2011).
Decision tree fields.
In ICCV pp. 1668–1675, IEEE.

Papandreou, G. and Yuille, A. (2011).
Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models.
In Computer Vision (ICCV), 2011 IEEE International Conference on pp. 193–200,.

Pearl, J. (1988).
Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

References X

R. Krishnan, S. Lacoste-Julien, D. S. (2015).
Barrier Frank-Wolfe for Marginal Inference.
In NIPS.

Rav-Acha, A., Kohli, P., Rother, C. and Fitzgibbon, A. (2008).
Unwrap mosaics: a new representation for video editing.
ACM Trans. Graph. *27*.

Ravikumar, P., Agarwal, A. and Wainwright, M. J. (2010).
Message-passing for graph-structured linear programs: Proximal methods and rounding schemes.
The Journal of Machine Learning Research *11*, 1043–1080.

Rother, C., Kohli, P., Feng, W. and Jia, J. (2009).
Minimizing sparse higher order energy functions of discrete variables.
In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on pp. 1382–1389,.

Rother, C., Kolmogorov, V., Lempitsky, V. S. and Szummer, M. (2007a).
Optimizing Binary MRFs via Extended Roof Duality.
In CVPR.

Rother, C., Kolmogorov, V., Lempitsky, V. S. and Szummer, M. (2007b).
Optimizing Binary MRFs via Extended Roof Duality.
In CVPR.

Savchynskyy, B., Kappes, J., Schmidt, S. and Schnörr, C. (2011).
A Study of Nesterov's Scheme for Lagrangian Decomposition and MAP Labeling.
In CVPR 2011 pp. 1817–1823,.

Savchynskyy, B., Kappes, J. H., Swoboda, P. and Schnörr, C. (2013).
Global MAP-Optimality by Shrinking the Combinatorial Search Area with Convex Relaxation.
In NIPS.

References XI

Savchynskyy, B. and Schmidt, S. (2014).

Getting Feasible Variable Estimates From Infeasible Ones: MRF Local Polytope Study.
In Advanced Structured Prediction MIT Press.

Savchynskyy, B., Schmidt, S., Kappes, J. H. and Schnörr, C. (2012).

Efficient MRF Energy Minimization via Adaptive Diminishing Smoothing.
In UAI pp. 746–755,.

Schlesinger, D. (2007).

Exact Solution of Permuted Submodular MinSum Problems.
In Energy Minimization Methods in Computer Vision and Pattern Recognition, (Yuille, A., Zhu, S.-C., Cremers, D. and Wang, Y., eds), vol. 4679, of Lecture Notes in Computer Science pp. 28–38. Springer Berlin Heidelberg.

Schlesinger, D. (2009).

General Search Algorithms for Energy Minimization Problems.
In Energy Minimization Methods in Computer Vision and Pattern Recognition, (Cremers, D., Boykov, Y., Blake, A. and Schmidt, F. R., eds), vol. 5681, of Lecture Notes in Computer Science pp. 84–97, Springer.

Schlesinger, D. and Flach, B. (2006).

Transforming an arbitrary minsum problem into a binary one.
TU Dresden, Fak. Informatik.

Schlesinger, M. and Giginyak, V. (2007).

Solution to Structural Recognition (MAX,+)-problems by their Equivalent Transformations. in 2 Parts.
Control Systems and Computers *1-2*.

Schlesinger, M. I. and Antoniuk, K. V. (2011).

Diffusion algorithms and structural recognition optimization problems.
Cybernetics and Systems Analysis *47*, 175–192.

Schmidt, U., Rother, C., Nowozin, S., Jancsary, J. and Roth, S. (2013).

Discriminative Non-blind Deblurring.
In 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013 pp. 604–611,.

References XII

Schraudolph, N. N. and Kamenetsky, D. (2008).
Efficient Exact Inference in Planar Ising Models.
In Advances in Neural Information Processing Systems 21, (Koller, D., Schuurmans, D., Bengio, Y. and Bottou, L., eds), pp. 1417–1424. Curran Associates, Inc.

Schrijver, A. (2000).
A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time.
J. Comb. Theory Ser. B *80*, 346–355.

Schwing, A. G., Hazan, T., Pollefeys, M. and Urtasun, R. (2011).
Distributed Message Passing for Large Scale Graphical Models.
In Proc. CVPR.

Shekhovtsov, A. (2014).
Maximum Persistency in Energy Minimization.
In CVPR.

Shekhovtsov, A., Kohli, P. and Rother, C. (2012).
Curvature Prior for MRF-Based Segmentation and Shape Inpainting.
In Pattern Recognition, (Pinz, A., Pock, T., Bischof, H. and Leberl, F., eds), vol. 7476, of Lecture Notes in Computer Science pp. 41–51. Springer Berlin Heidelberg.

Shekhovtsov, A., Swoboda, P. and Savchynskyy, B. (2015).
Maximum Persistency via Iterative Relaxed Inference with Graphical Models.
In CVPR.

Sontag, D., Meltzer, T., Globerson, A., Weiss, Y. and Jaakkola, T. (2008).
Tightening LP Relaxations for MAP using Message-Passing.
In UAI pp. 503–510,.

Storvik, G. and Dahl, G. (2000).
Lagrangian-based methods for finding MAP solutions for MRF models.
IEEE Transactions on Image Processing *9*, 469–479.

References XIII

Strandmark, P. and Kahl, F. (2010).
Parallel and distributed graph cuts by dual decomposition.
In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on pp. 2085–2092,.

Swoboda, P., Savchynskyy, B., Kappes, J. H. and Schnörr, C. (2013).
Partial Optimality via Iterative Pruning for the Potts Model.
In SSVM.

Swoboda, P., Savchynskyy, B., Kappes, J. H. and Schnörr, C. (2014).
Partial Optimality by Pruning for MAP-inference with General Graphical Models.
In CVPR.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M. and Rother, C. (2008).
A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors.
IEEE PAMI *30*, 1068–1080.

Torresani, L., Kolmogorov, V. and Rother, C. (2008).
Feature Correspondence Via Graph Matching: Models and Global Optimization.
In Proceedings of the 10th European Conference on Computer Vision: Part II ECCV '08 pp. 596–609, Springer-Verlag, Berlin, Heidelberg.

Tsochantaridis, I., Joachims, T., Hofmann, T. and Altun, Y. (2005).
Large Margin Methods for Structured and Interdependent Output Variables.
Journal of Machine Learning Research *6*, 1453–1484.

Werner, T. (2007).
A Linear Programming Approach to Max-Sum Problem: A Review.
PAMI *29*.

Werner, T. (2009).
Revisiting the Decomposition Approach to Inference in Exponential Families and Graphical Models.
Technical report CMP, Czech TU.

References XIV

Zhu, S. C., Wu, Y. N. and Mumford, D. (1998).
Filters, Random Fields and Maximum Entropy (FRAME): Towards a Unified Theory for Texture Modeling.
International Journal of Computer Vision *27*, 107–126.