Optimization of Markov Random Field in Computer Vision

Thalaiyasingam Ajanthan

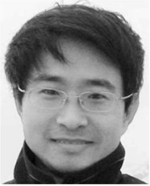
The Australian National University

Data61, CSIRO

Data61, April 2017

Collaborators

Richard Hartley
(primary supervisor)



Hongdong Li
(panel chair)

Mathieu Salzmann
(co-supervisor)

Philip Torr

Pawan Kumar

Alban Desmaison

Rudy Bunel

Outline

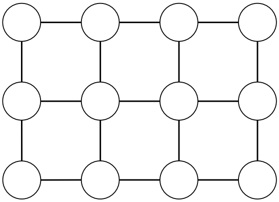
Outline

A Pairwise Markov Random Field

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) \ ,$$

where $x_i \in \mathcal{L}$ for all $i \in \mathcal{V}$.

- θ_i Unary potentials (data)
- θ_{ij} Pairwise potentials (regularizer)
- $\mathcal{V}$ Set of vertices (n)
- $\mathcal{E}$ Set of edges (m)
- $\mathcal{L}$ Set of labels (ℓ)



4-connected

Optimization

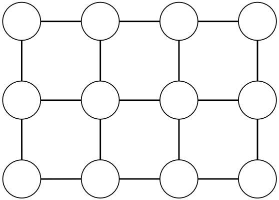
$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{L}^n} E(\mathbf{x}) \ .$$

A Pairwise Markov Random Field

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) \ ,$$

where $x_i \in \mathcal{L}$ for all $i \in \mathcal{V}$.

- θ_i Unary potentials (data)
- θ_{ij} Pairwise potentials (regularizer)
- $\mathcal{V}$ Set of vertices (n)
- $\mathcal{E}$ Set of edges (m)
- $\mathcal{L}$ Set of labels (ℓ)



4-connected

Optimization

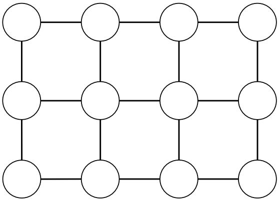
$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{L}^n} E(\mathbf{x}) \ .$$

A Pairwise Markov Random Field

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) \ ,$$

where $x_i \in \mathcal{L}$ for all $i \in \mathcal{V}$.

- θ_i Unary potentials (data)
- θ_{ij} Pairwise potentials (regularizer)
- $\mathcal{V}$ Set of vertices (n)
- $\mathcal{E}$ Set of edges (m)
- $\mathcal{L}$ Set of labels (ℓ)



4-connected

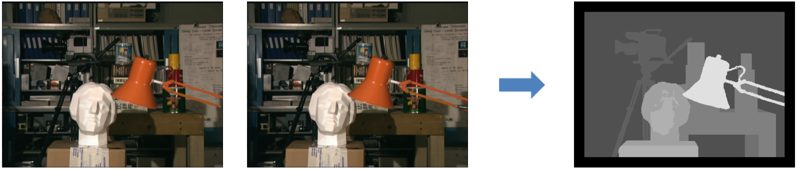
Optimization

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{L}^n} E(\mathbf{x}) \ .$$

Intractable

Computer Vision Applications

Stereo



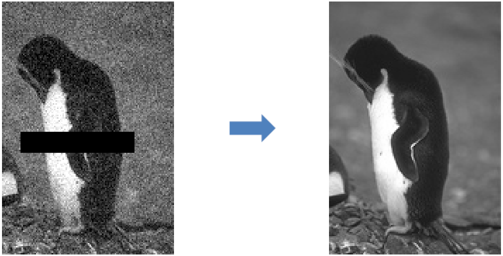
$\mathcal{V}$ Set of pixels

$\mathcal{E}$ 4-connected neighbourhood

$\mathcal{L}$ Set of disparities, $\{0, \ldots, \kappa\}$

Computer Vision Applications

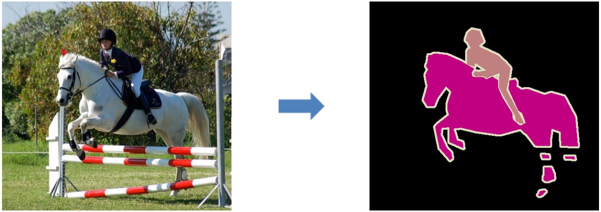
Inpainting



- $\mathcal{V}$ Set of pixels
- $\mathcal{E}$ 4-connected neighbourhood
- $\mathcal{L}$ Set of intensities, $\{0, \ldots, 255\}$

Computer Vision Applications

Segmentation



$\mathcal{V}$ Set of pixels

$\mathcal{E}$ Fully connected neighbourhood

$\mathcal{L}$ Set of object classes

Contributions

Three new algorithms.

Memory Efficient Max Flow (MEMF)

- A max-flow algorithm with $\mathcal{O}(\ell)$ memory reduction for multi-label submodular MRFs.

Iteratively Reweighted Graph Cut (IRGC)

- A move-making algorithm that can handle robust non-convex priors.

Efficient Linear Programming (PROX-LP)

- An LP minimization algorithm for dense CRFs that has linear time iterations.

Contributions

Three new algorithms.

Memory Efficient Max Flow (MEMF)

- A max-flow algorithm with $\mathcal{O}(\ell)$ memory reduction for multi-label submodular MRFs.

Iteratively Reweighted Graph Cut (IRGC)

- A move-making algorithm that can handle robust non-convex priors.

Efficient Linear Programming (PROX-LP)

- An LP minimization algorithm for dense CRFs that has linear time iterations.

Contributions

Three new algorithms.

Memory Efficient Max Flow (MEMF)

- A max-flow algorithm with $\mathcal{O}(\ell)$ memory reduction for multi-label submodular MRFs.

Iteratively Reweighted Graph Cut (IRGC)

- A move-making algorithm that can handle robust non-convex priors.

Efficient Linear Programming (PROX-LP)

- An LP minimization algorithm for dense CRFs that has linear time iterations.

Contributions

Three new algorithms.

Memory Efficient Max Flow (MEMF)

- A max-flow algorithm with $\mathcal{O}(\ell)$ memory reduction for multi-label submodular MRFs.

Iteratively Reweighted Graph Cut (IRGC)

- A move-making algorithm that can handle robust non-convex priors.

Efficient Linear Programming (PROX-LP)

- An LP minimization algorithm for dense CRFs that has linear time iterations.

Outline

Introduction

Minimize

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) \ ,$$

where $x_i \in \mathcal{L} = \{0, 1, \cdots, \ell - 1\}$.

Multi-label submodular

$$\theta_{ij}(\lambda', \mu) + \theta_{ij}(\lambda, \mu') - \theta_{ij}(\lambda, \mu) - \theta_{ij}(\lambda', \mu') \geq 0 \ ,$$

for all $\lambda, \lambda', \mu, \mu'$ where $\lambda < \lambda'$ and $\mu < \mu'$ [Schlesinger-2006].

E.g. θ_{ij} is convex.

Current method

- Ishikawa algorithm [Ishikawa-2003, Schlesinger-2006].

Introduction

Minimize

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) ,$$

where $x_i \in \mathcal{L} = \{0, 1, \cdots, \ell - 1\}$.

Multi-label submodular

$$\theta_{ij}(\lambda', \mu) + \theta_{ij}(\lambda, \mu') - \theta_{ij}(\lambda, \mu) - \theta_{ij}(\lambda', \mu') \geq 0 ,$$

for all $\lambda, \lambda', \mu, \mu'$ where $\lambda < \lambda'$ and $\mu < \mu'$ [Schlesinger-2006].

E.g. θ_{ij} is convex.

Current method

- Ishikawa algorithm [Ishikawa-2003, Schlesinger-2006].

Introduction

Minimize

$$E(\mathbf{x}) = \sum_{i\in\mathcal{V}} \theta_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \theta_{ij}(x_i, x_j) ,$$

where $x_i \in \mathcal{L} = \{0, 1, \cdots, \ell - 1\}$.

Multi-label submodular

$$\theta_{ij}(\lambda', \mu) + \theta_{ij}(\lambda, \mu') - \theta_{ij}(\lambda, \mu) - \theta_{ij}(\lambda', \mu') \geq 0 ,$$

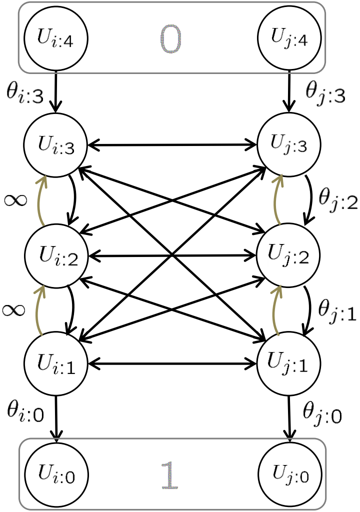
for all $\lambda, \lambda', \mu, \mu'$ where $\lambda < \lambda'$ and $\mu < \mu'$ [Schlesinger-2006].

E.g. θ_{ij} is convex.

Current method

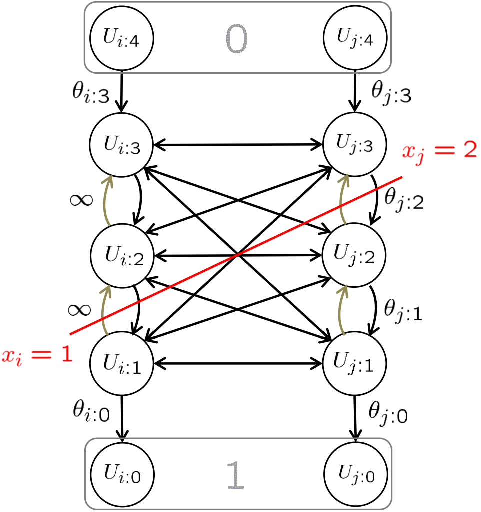
- Ishikawa algorithm [Ishikawa-2003, Schlesinger-2006].

The Ishikawa Algorithm



The Ishikawa graph

The Ishikawa Algorithm



The Ishikawa graph

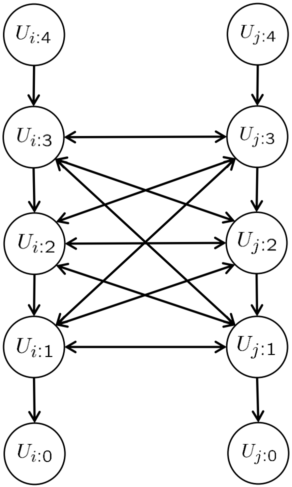
The Ishikawa Algorithm

Drawback

- Huge memory complexity: $\mathcal{O}(m\ell^2)$.

Contribution

- An algorithm with memory complexity $\mathcal{O}(m\ell)$.



The Ishikawa Algorithm

Drawback

- Huge memory complexity: $\mathcal{O}(m\ell^2)$.

E.g. $n = 10^6$, $\ell = 256$
$m \approx 2 \times 10^6$
Edges $\approx 2 \times 10^6 \times 2 \times 256^2$
Memory $\approx$ 1000 GB

$\Downarrow$

Contribution

- An algorithm with memory complexity $\mathcal{O}(m\ell)$.

Memory $\approx$ 4 GB

The Ishikawa Algorithm

Drawback

- Huge memory complexity: $\mathcal{O}(m\ell^2)$.

E.g. $n = 10^6$, $\ell = 256$
$m \approx 2 \times 10^6$
Edges $\approx 2 \times 10^6 \times 2 \times 256^2$
Memory $\approx$ 1000 GB

$\Downarrow$

Contribution

- An algorithm with memory complexity $\mathcal{O}(m\ell)$.

Memory $\approx$ 4 GB

The Ishikawa Algorithm

Drawback

- Huge memory complexity: $\mathcal{O}(m\ell^2)$.

E.g. $n = 10^6$, $\ell = 256$
$m \approx 2 \times 10^6$
Edges $\approx 2 \times 10^6 \times 2 \times 256^2$
Memory $\approx$ 1000 GB

$\Downarrow$

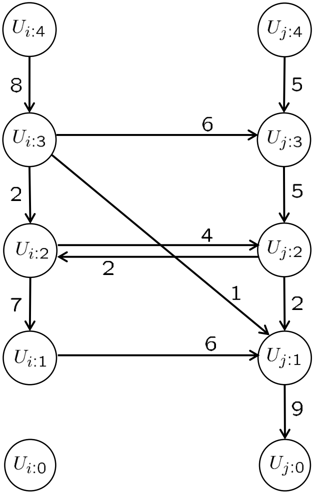
Memory $\approx$ 4 GB

Contribution

- An algorithm with memory complexity $\mathcal{O}(m\ell)$.

Memory reduction: $\mathcal{O}(\ell)$.

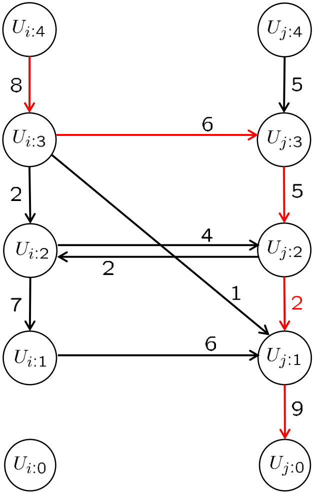
Max Flow on the Ishikawa Graph



Initial Ishikawa graph

Flow $= 0$

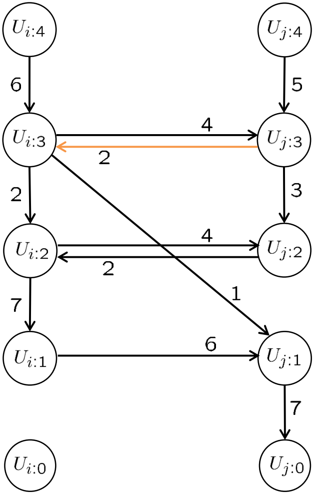
Max Flow on the Ishikawa Graph



Max-flow in progress

Flow = 0

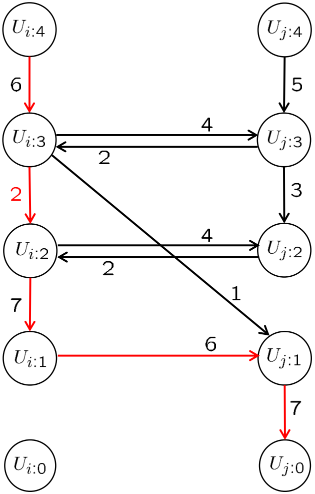
Max Flow on the Ishikawa Graph



Max-flow in progress

Flow = 2

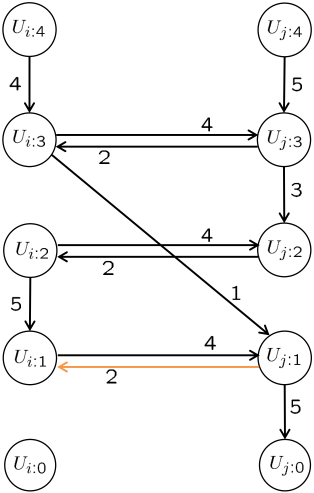
Max Flow on the Ishikawa Graph



Max-flow in progress

Flow = 2

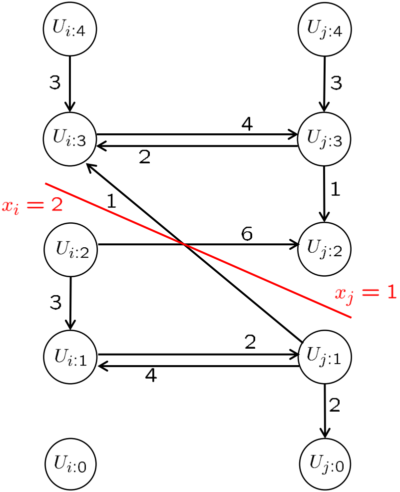
Max Flow on the Ishikawa Graph



Max-flow in progress

Flow $= 4$

Max Flow on the Ishikawa Graph



Min-cut

Flow $= 7$

Memory Efficient Flow Encoding

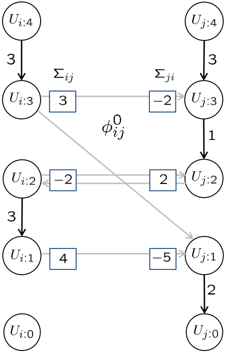
Idea: Don't store the residual graph but exit-flows between each pair of neighbouring columns.

Memory Efficient Flow Encoding

Idea: Don't store the residual graph but exit-flows between each pair of neighbouring columns.

Exit-flow: Given flow ψ, an exit-flow is defined as

$$\Sigma_{ij:\lambda} = \sum_{\mu} \psi_{ij:\lambda\mu} \ .$$

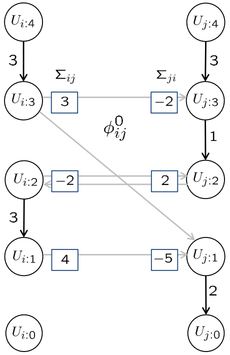


Memory Efficient Flow Encoding

Idea: Don't store the residual graph but exit-flows between each pair of neighbouring columns.

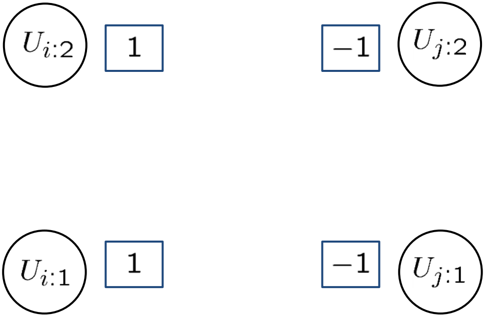
Exit-flow: Given flow ψ, an exit-flow is defined as

$$\Sigma_{ij:\lambda} = \sum_{\mu} \psi_{ij:\lambda\mu} \ .$$



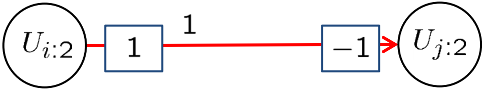
The residual graph can be rapidly computed from the exit-flows.

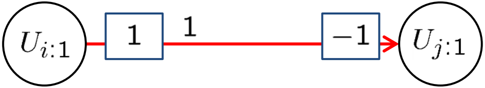
Flow Equivalence - An Example



Exit-flows

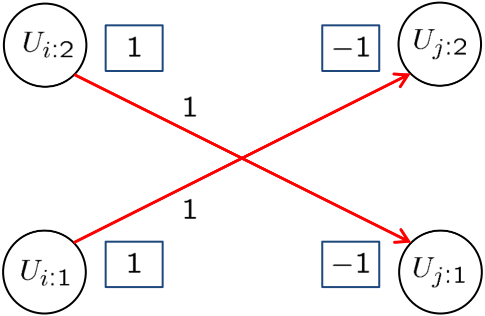
Flow Equivalence - An Example





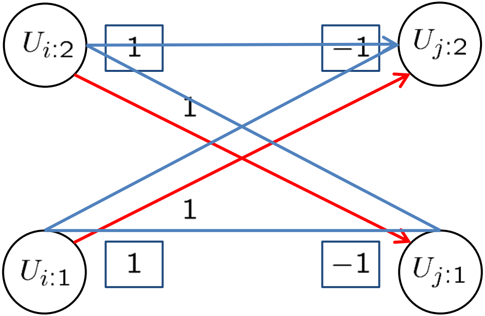
A reconstructed flow

Flow Equivalence - An Example



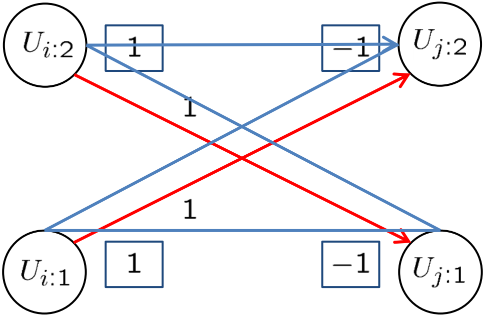
Another reconstructed flow

Flow Equivalence - An Example



Both reconstructions are equivalent

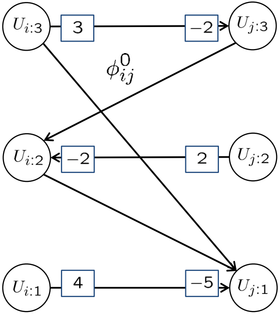
Flow Equivalence - An Example



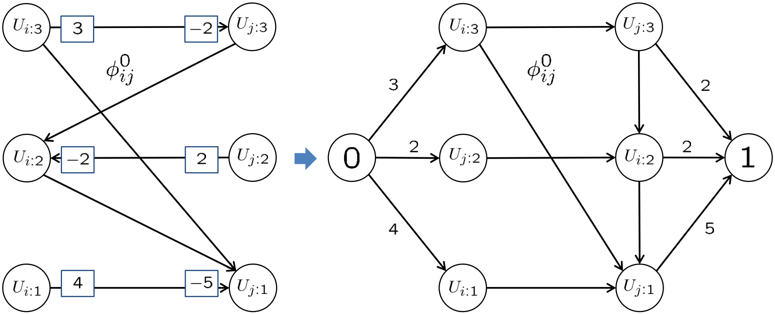
Both reconstructions are equivalent

Flow-loop $\equiv$ reparametrization.

Flow Reconstruction / Computing Residual Edges



Flow Reconstruction / Computing Residual Edges



Flow reconstruction as a small max-flow problem

Memory Efficient Max Flow (MEMF)

Algorithm

Require: ϕ^0 $\triangleright$ Initial Ishikawa capacities

  $\Sigma \leftarrow 0$ $\triangleright$ Initialize exit-flows

  repeat

    $P \leftarrow \text{augmenting-path}(\phi^0, \Sigma)$

    $\Sigma \leftarrow \text{augment}(P, \phi^0, \Sigma)$

  until no augmenting paths possible

Assumption:
ϕ^0 can be stored in an efficient manner.

Memory Efficient Max Flow (MEMF)

Algorithm

Require: ϕ^0 ▷ Initial Ishikawa capacities
  $\Sigma \leftarrow 0$ ▷ Initialize exit-flows
  repeat
    $P \leftarrow \text{augmenting-path}(\phi^0, \Sigma)$
    $\Sigma \leftarrow \text{augment}(P, \phi^0, \Sigma)$
  until no augmenting paths possible

Assumption:
ϕ^0 can be stored in an efficient manner.

Memory complexity: $\mathcal{O}(m\ell)$.

Memory Efficient Max Flow (MEMF)

Algorithm

Require: ϕ^0 ▷ Initial Ishikawa capacities

$\Sigma \leftarrow 0$ ▷ Initialize exit-flows

repeat

$P \leftarrow \text{augmenting-path}(\phi^0, \Sigma)$

$\Sigma \leftarrow \text{augment}(P, \phi^0, \Sigma)$

until no augmenting paths possible

Assumption:

ϕ^0 can be stored in an efficient manner.

Memory complexity: $\mathcal{O}(m\ell)$.

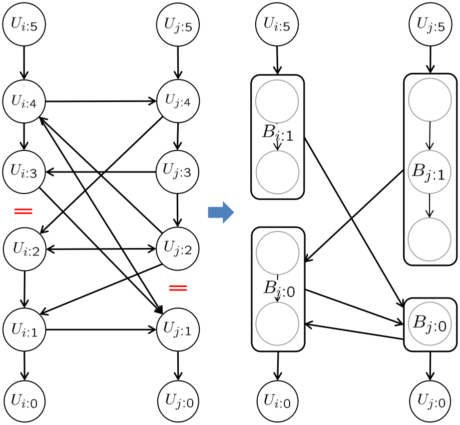
Efficiently Finding an Augmenting Path

Simplified graph

- Unweighted sparse graph.
- Fewer augmenting paths.

Search-tree-recycling

- Good empirical performance.



Simplified graph representation

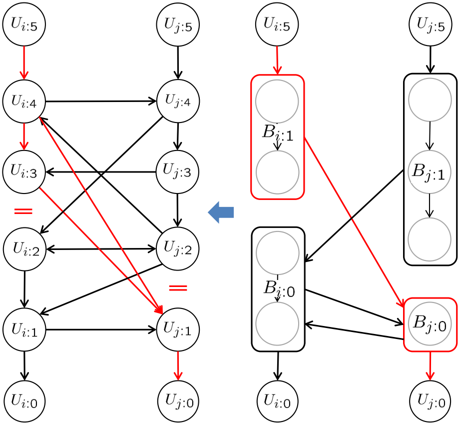
Efficiently Finding an Augmenting Path

Simplified graph

- Unweighted sparse graph.
- Fewer augmenting paths.

Search-tree-recycling

- Good empirical performance.



Simplified graph representation

Efficiently Finding an Augmenting Path

Simplified graph

- Unweighted sparse graph.
- Fewer augmenting paths.

Search-tree-recycling

- Good empirical performance.

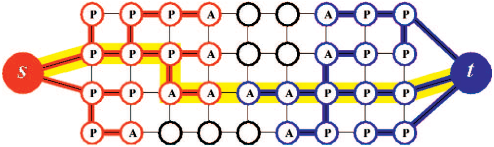
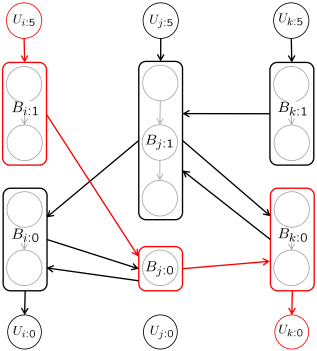


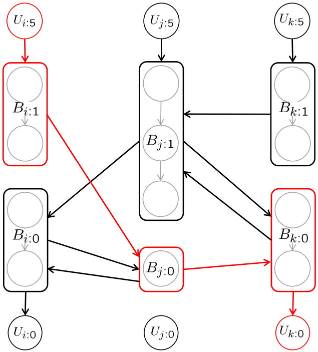
Image from [Boykov-2004]

Augmentation

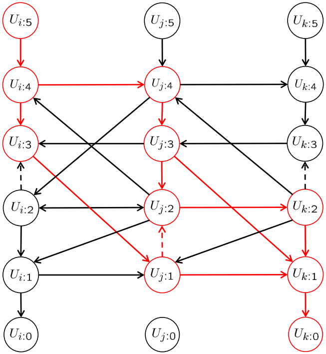


Augmenting path

Augmentation

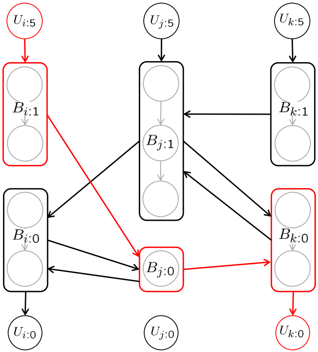


Augmenting path

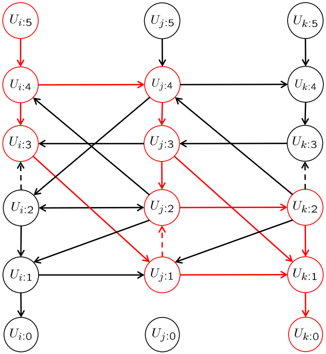


Directed acyclic graph

Augmentation



Augmenting path



Directed acyclic graph

Maximum flow can be pushed using dynamic programming.

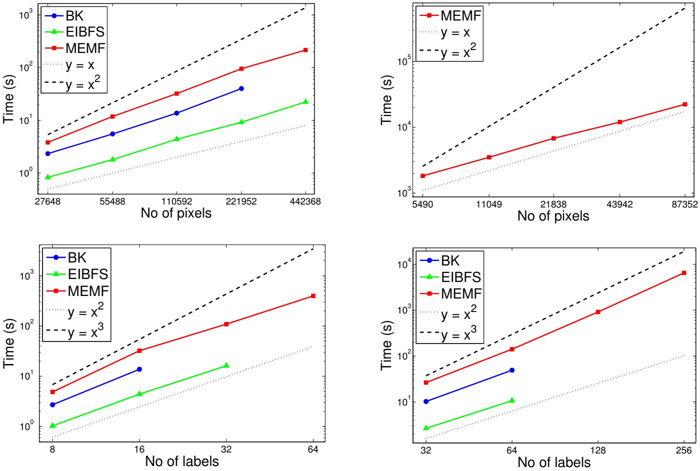
Results

Problem		Memory [MB]			Time [s]		
Name	ℓ	BK	EIBFS	**MEMF**	BK	EIBFS	**MEMF**
Tsukuba	16	3195	2495	211	14	4	30
Venus	20	7626	5907	396	35	9	60
Sawtooth	20	7566	5860	393	31	8	35
Map	30	6454	4946	219	57	9	36
Cones	60	*72303	*55063	1200	-	-	371
Teddy	60	*72303	*55063	1200	-	-	2118
KITTI	40	*88413	*67316	2215	-	-	19008
Penguin	256	*173893	*130728	663	-	-	6835
House	256	*521853	*392315	1986	-	-	9290

Comparison with other max-flow implementations

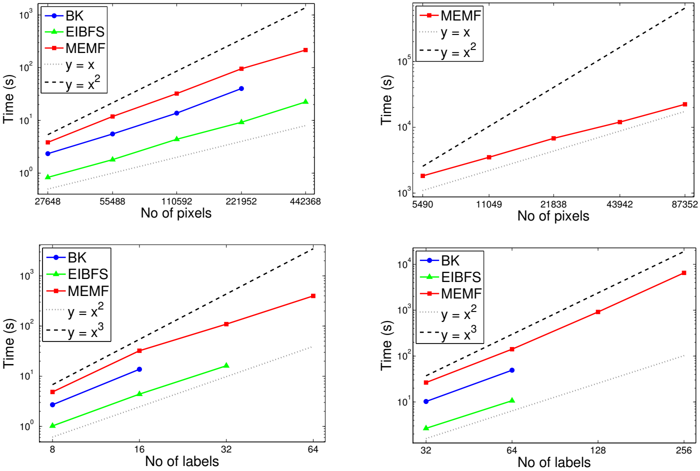
BK [Boykov-2004]
EIBFS [Goldberg-2015]

Empirical Time Complexity



Tsukuba *Penguin*

Empirical Time Complexity



Tsukuba *Penguin*

Empirical time complexity: $\mathcal{O}(n\ell^3)$.

Summary

- We have introduced a memory efficient alternative to the Ishikawa algorithm.

Publication: CVPR, 2016 and submitted to PAMI, 2017

Code: `https://github.com/tajanthan/memf`

Outline

Introduction

Minimize

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}\left(|x_i - x_j|\right) \ ,$$

where $x_i \in \mathcal{L} = \{0, 1, \cdots, \ell - 1\}$.

Graph cut algorithms

- θ_{ij} convex $\Rightarrow$ Ishikawa algorithm [Ishikawa-2003].
- θ_{ij} concave $\Rightarrow$ α-expansion [Boykov-2001].
- θ_{ij} non-convex $\Rightarrow$ IRGC [Ajanthan-2015].

Introduction

Minimize

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}\left(|x_i - x_j|\right) \,,$$

where $x_i \in \mathcal{L} = \{0, 1, \cdots, \ell - 1\}$.

Graph cut algorithms

- θ_{ij} convex $\Rightarrow$ Ishikawa algorithm [Ishikawa-2003].
- θ_{ij} concave $\Rightarrow$ α-expansion [Boykov-2001].
- θ_{ij} non-convex $\Rightarrow$ IRGC [Ajanthan-2015].

Introduction

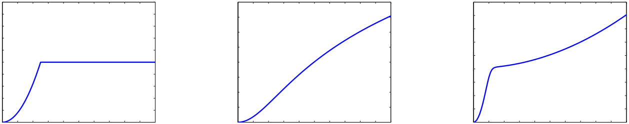
Minimize

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}\left(|x_i - x_j|\right) ,$$

where $x_i \in \mathcal{L} = \{0, 1, \cdots, \ell - 1\}$.

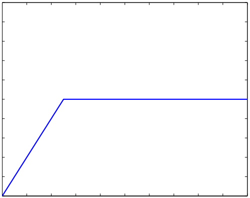
Graph cut algorithms

- θ_{ij} convex $\Rightarrow$ Ishikawa algorithm [Ishikawa-2003].
- θ_{ij} concave $\Rightarrow$ α-expansion [Boykov-2001].
- θ_{ij} non-convex $\Rightarrow$ IRGC [Ajanthan-2015].



Trun. quad. *Cauchy* *Cor. Gauss.*

α-expansion

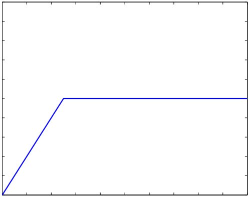


Pairwise potential

Initialization

- Optimal expansion move is found using *max-flow*.

α-expansion

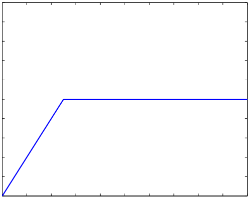


Pairwise potential

Expand green

- Optimal expansion move is found using *max-flow*.

α-expansion

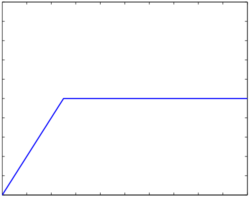


Pairwise potential

Expand dark-brown

- Optimal expansion move is found using *max-flow*.

α-expansion

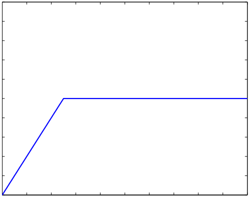


Pairwise potential

Expand light-green

- Optimal expansion move is found using *max-flow*.

α-expansion



Pairwise potential

No expansion possible

- Optimal expansion move is found using *max-flow*.

Iteratively Reweighted Graph Cut (IRGC)

A move-making algorithm

- Minimizes the original MRF energy, by iteratively minimizing a multi-label submodular surrogate energy.
- Monotonic decrease of the original energy.

Special case: Iteratively Reweighted Least Squares (IRLS).

Iteratively Reweighted Graph Cut

Assumption

$$\theta_{ij}\left(|x_i - x_j|\right) = h \circ g\left(|x_i - x_j|\right) .$$

Non-decreasing concave

Convex

Minimize

$$\tilde{E}(\mathbf{x}) = \sum_{i\in\mathcal{V}} \theta_i(x_i) + \sum_{(i,j)\in\mathcal{E}} w_{ij}^t \, g\left(|x_i - x_j|\right) .$$

Depends on the function h and the current labelling $\mathbf{x}^t$.

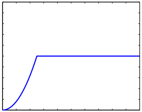
Iterately Reweighted Graph Cut

Assumption

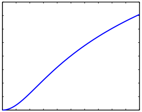
$$\theta_{ij}\left(|x_i - x_j|\right) = h \circ g\left(|x_i - x_j|\right) .$$

Non-decreasing concave

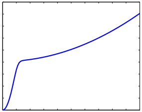
Convex



Trun. quad.



Cauchy



Cor. Gauss.

Minimize

$$\tilde{E}(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} w_{ij}^t \, g\left(|x_i - x_j|\right) .$$

Depends on the function h and the current labelling $\mathbf{x}^t$.

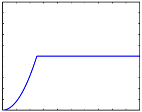
Iteratively Reweighted Graph Cut

Assumption

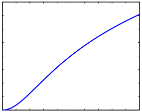
$$\theta_{ij}\left(|x_i - x_j|\right) = h \circ g\left(|x_i - x_j|\right) .$$

Non-decreasing concave

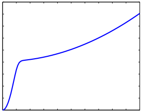
Convex



Trun. quad.



Cauchy



Cor. Gauss.

Minimize

$$\tilde{E}(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} w_{ij}^t \, g\left(|x_i - x_j|\right) .$$

Depends on the function h and the current labelling $\mathbf{x}^t$.

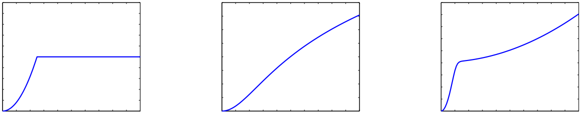
Iterately Reweighted Graph Cut

Assumption

$$\theta_{ij}\left(|x_i - x_j|\right) = h \circ g\left(|x_i - x_j|\right) .$$

Non-decreasing concave

Convex



Trun. quad. *Cauchy* *Cor. Gauss.*

Minimize

$$\tilde{E}(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} w_{ij}^t \, g\left(|x_i - x_j|\right) .$$

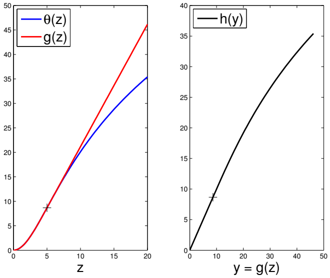
Depends on the function h and the current labelling $\mathbf{x}^t$.

> $\tilde{E}(\mathbf{x})$ is multi-label submodular.

Choice of Functions g and h

$$\theta(z) = h \circ g(z) \; .$$

- Choose g such that the number of edges in the Ishikawa graph is minimized.



θ - Cauchy function

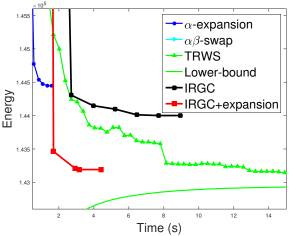
Hybrid Strategy

IRGC + α-expansion

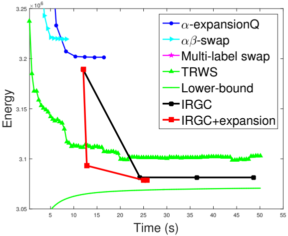
- Updates $\mathbf{x}^t \to \mathbf{x}^{t+1}$ in two steps:
  1. $\mathbf{x}^t \to \mathbf{x}' \Rightarrow$ Ishikawa algorithm.
  2. $\mathbf{x}' \to \mathbf{x}^{t+1} \Rightarrow$ One pass of α-expansion.
- Effective to overcome local minima.

Results

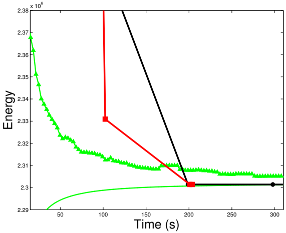
- We evaluated on stereo and inpainting problems.



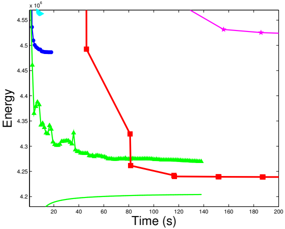
Map, Trunc. linear



Venus, Trunc. quad.



Cones, Cauchy



Penguin, Trunc. quad.

Results

Problem	α-exp. (QPBO)	$\alpha\beta$ swap	TRWS	**Ours** IRGC	**Ours** IRGC+exp.
Map	1.05%	4.59%	0.05%	0.74%	0.17%
Teddy	0.75%	2.40%	0.30%	1.63%	0.21%
Venus	4.26%	4.85%	0.91%	0.35%	0.26%
Sawtooth	3.42%	4.58%	0.65%	0.96%	0.26%
Cones	7.80%	95.11%	0.16%	0.01%	0.01%
Tsukuba	1.99%	3.45%	0.09%	0.47%	0.17%
Penguin	6.71%	8.53%	1.56%	11.72%	0.83%
House	4.59%	3.71%	0.02%	0.01%	0.01%
Average	3.82%	15.90%	0.47%	1.99%	0.24%

Quality of the minimum energies

TRWS [Kolmogorov-2006]

Summary

- We have introduced a move-making algorithm that is effective on multi-label MRFs with non-convex priors.

Publication: CVPR, 2015

Code: `https://github.com/tajanthan/irgc`

Outline

Introduction

Minimize

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) \ ,$$

where $x_i \in \mathcal{L}$, $\mathcal{V} = \{1, \ldots, n\}$ and $\mathcal{E} = \{(i,j) \mid i,j \in \mathcal{V}, i \neq j\}$.

Gaussian pairwise potentials

$$\theta_{ij}(x_i, x_j) = \underbrace{\mathbb{1}[x_i \neq x_j]}_{\text{Label compatibility}} \ \underbrace{\exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right)}_{\text{Pixel compatibility}} ,$$

where $\mathbf{f}_i \in \mathbb{R}^{\mathrm{d}}$.

Why?

- Captures long-range interactions and provides fine grained segmentations [Krähenbühl-2011].

Introduction

Minimize

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) \quad ,$$

where $x_i \in \mathcal{L}$, $\mathcal{V} = \{1, \ldots, n\}$ and $\mathcal{E} = \{(i,j) \mid i,j \in \mathcal{V}, i \neq j\}$.

Gaussian pairwise potentials

$$\theta_{ij}(x_i, x_j) = \underbrace{\mathbb{1}[x_i \neq x_j]}_{\text{Label compatibility}} \underbrace{\exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right)}_{\text{Pixel compatibility}},$$

where $\mathbf{f}_i \in \mathbb{R}^{\mathrm{d}}$.

Why?

- Captures long-range interactions and provides fine grained segmentations [Krähenbühl-2011].

Introduction

Minimize

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \theta_{ij}(x_i, x_j) \quad ,$$

where $x_i \in \mathcal{L}$, $\mathcal{V} = \{1, \ldots, n\}$ and $\mathcal{E} = \{(i,j) \mid i,j \in \mathcal{V}, i \neq j\}$.

Gaussian pairwise potentials

$$\theta_{ij}(x_i, x_j) = \underbrace{\mathbb{1}[x_i \neq x_j]}_{\text{Label compatibility}} \underbrace{\exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right)}_{\text{Pixel compatibility}},$$

where $\mathbf{f}_i \in \mathbb{R}^{\mathrm{d}}$.

Why?

- Captures long-range interactions and provides fine grained segmentations [Krähenbühl-2011].

Dense CRF

$$E(\mathbf{x}) = \sum_{i=1}^{n} \theta_i(x_i) + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbb{1}[x_i \neq x_j] \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right) .$$

Difficulty

- Requires $\mathcal{O}(n^2)$ computations $\Rightarrow$ Infeasible.

Idea

- Approximate using the filtering method [Adams-2010] $\Rightarrow$ $\mathcal{O}(n)$ computations.

Dense CRF

$$E(\mathbf{x}) = \sum_{i=1}^{n} \theta_i(x_i) + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbb{1}[x_i \neq x_j] \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right).$$

Difficulty

- Requires $\mathcal{O}(n^2)$ computations $\Rightarrow$ Infeasible.

Idea

- Approximate using the filtering method [Adams-2010] $\Rightarrow \mathcal{O}(n)$ computations.

Dense CRF

$$E(\mathbf{x}) = \sum_{i=1}^{n} \theta_i(x_i) + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbb{1}[x_i \neq x_j] \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right).$$

Difficulty

- Requires $\mathcal{O}(n^2)$ computations $\Rightarrow$ Infeasible.

Idea

- Approximate using the filtering method [Adams-2010] $\Rightarrow$ $\mathcal{O}(n)$ computations.

Current Algorithms for MAP Inference in Dense CRFs

- Rely on the efficient filtering method [Adams-2010].

Algorithm	Time complexity per iteration	Theoretical bound
Mean Field (MF) [1]	$\mathcal{O}(n)$	No
Quadratic Programming (QP) [2]	$\mathcal{O}(n)$	Yes
Difference of Convex (DC) [2]	$\mathcal{O}(n)$	Yes
Linear Programming (LP) [2]	$\mathcal{O}(n \log(n))$	Yes (best)

Contribution

- LP in $\mathcal{O}(n)$ time per iteration
  $\Rightarrow$ An order of magnitude speedup.

[1] [Krähenbühl-2011]
[2] [Desmaison-2016]

Current Algorithms for MAP Inference in Dense CRFs

- Rely on the efficient filtering method [Adams-2010].

Algorithm	Time complexity per iteration	Theoretical bound
Mean Field (MF) [1]	$\mathcal{O}(n)$	No
Quadratic Programming (QP) [2]	$\mathcal{O}(n)$	Yes
Difference of Convex (DC) [2]	$\mathcal{O}(n)$	Yes
Linear Programming (LP) [2]	$\mathcal{O}(n \log(n))$	Yes (best)

Contribution

- LP in $\mathcal{O}(n)$ time per iteration
  $\Rightarrow$ An order of magnitude speedup.

[1] [Krähenbühl-2011]
[2] [Desmaison-2016]

Current Algorithms for MAP Inference in Dense CRFs

- Rely on the efficient filtering method [Adams-2010].

Algorithm	Time complexity per iteration	Theoretical bound
Mean Field (MF) [1]	$\mathcal{O}(n)$	No
Quadratic Programming (QP) [2]	$\mathcal{O}(n)$	Yes
Difference of Convex (DC) [2]	$\mathcal{O}(n)$	Yes
Linear Programming (LP) [2]	$\mathcal{O}(n \log(n))$	Yes (best)

Contribution

- LP in $\mathcal{O}(n)$ time per iteration
  $\Rightarrow$ An order of magnitude speedup.

[1] [Krähenbühl-2011]
[2] [Desmaison-2016]

E.g. $n = 10^6 \quad \Rightarrow \quad$ 20 times speedup.

LP Relaxation of a Dense CRF

$$y_{i:\lambda} = 1 \quad \Rightarrow \quad x_i = \lambda.$$

$$\min_{\mathbf{y}} \quad \tilde{E}(\mathbf{y}) = \sum_i \sum_\lambda \theta_{i:\lambda}\, y_{i:\lambda} + \sum_{i,j \neq i} \sum_\lambda K_{ij} \frac{|y_{i:\lambda} - y_{j:\mu}|}{2} \;,$$

$$\text{s.t.} \quad \mathbf{y} \in \mathcal{S} = \left\{ \mathbf{y} \;\middle|\; \begin{array}{l} \sum_\lambda y_{i:\lambda} = 1,\ i \in \mathcal{V} \\ y_{i:\lambda} \in [0,1],\ i \in \mathcal{V},\ \lambda \in \mathcal{L} \end{array} \right\} ,$$

where $K_{ij} = \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right)$.

- Provides integrality gap of 2 [Kleinberg-2002].

LP Relaxation of a Dense CRF

$$y_{i:\lambda} = 1 \quad \Rightarrow \quad x_i = \lambda.$$

$$\min_{\mathbf{y}} \quad \tilde{E}(\mathbf{y}) = \sum_{i} \sum_{\lambda} \theta_{i:\lambda}\, y_{i:\lambda} + \sum_{i,j \neq i} \sum_{\lambda} K_{ij} \frac{|y_{i:\lambda} - y_{j:\mu}|}{2} \ ,$$

$$\text{s.t.} \quad \mathbf{y} \in \mathcal{S} = \left\{ \mathbf{y} \;\middle|\; \begin{array}{l} \sum_{\lambda} y_{i:\lambda} = 1,\ i \in \mathcal{V} \\ y_{i:\lambda} \in [0,1],\ i \in \mathcal{V},\ \lambda \in \mathcal{L} \end{array} \right\} \ ,$$

where $K_{ij} = \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right)$.

- Provides integrality gap of 2 [Kleinberg-2002].

LP Relaxation of a Dense CRF

$$y_{i:\lambda} = 1 \quad \Rightarrow \quad x_i = \lambda.$$

$$\min_{\mathbf{y}} \quad \tilde{E}(\mathbf{y}) = \sum_i \sum_\lambda \theta_{i:\lambda}\, y_{i:\lambda} + \sum_{i,j\neq i} \sum_\lambda K_{ij} \frac{|y_{i:\lambda} - y_{j:\mu}|}{2},$$

$$\text{s.t.} \quad \mathbf{y} \in \mathcal{S} = \left\{ \mathbf{y} \;\middle|\; \begin{array}{l} \sum_\lambda y_{i:\lambda} = 1,\ i \in \mathcal{V} \\ y_{i:\lambda} \in [0,1],\ i \in \mathcal{V},\ \lambda \in \mathcal{L} \end{array} \right\},$$

where $K_{ij} = \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right)$.

- Provides integrality gap of 2 [Kleinberg-2002].

Standard solvers would require $\mathcal{O}(n^2)$ variables.

LP Minimization

Current method

- Projected subgradient descent $\Rightarrow$ Too slow.
  - Linearithmic time per iteration.
  - Expensive line search.
  - Requires large number of iterations.

Our algorithm

- Proximal minimization using block-coordinate descent.
  - One block: Significantly smaller subproblems.
  - The other block: Efficient *conditional gradient descent*.
    - Linear time per iteration.
    - Optimal step size.
  - Guarantees optimality and converges faster.

LP Minimization

Current method

- Projected subgradient descent $\Rightarrow$ Too slow.
  - Linearithmic time per iteration.
  - Expensive line search.
  - Requires large number of iterations.

Our algorithm

- Proximal minimization using block-coordinate descent.
  - One block: Significantly smaller subproblems.
  - The other block: Efficient *conditional gradient descent*.
    - Linear time per iteration.
    - Optimal step size.
  - Guarantees optimality and converges faster.

Proximal Minimization of LP (PROX-LP)

$$\begin{aligned} \min_{\mathbf{y}} \quad & \tilde{E}(\mathbf{y}) + \frac{1}{2\eta} \|\mathbf{y} - \mathbf{y}^r\|^2 \ , \\ \text{s.t.} \quad & \mathbf{y} \in \mathcal{S} \ , \end{aligned}$$

where $\eta > 0$ and $\mathbf{y}^r$ is the current estimate.

Why?

- Initialization using MF or DC.
- Smooth dual $\Rightarrow$ Sophisticated optimization.

Approach

- Block-coordinate descent tailored to this problem.

Proximal Minimization of LP (PROX-LP)

$$\min_{\mathbf{y}} \quad \tilde{E}(\mathbf{y}) + \frac{1}{2\eta}\left\|\mathbf{y} - \mathbf{y}^r\right\|^2 \ ,$$
$$\text{s.t.} \quad \mathbf{y} \in \mathcal{S} \ ,$$

where $\eta > 0$ and $\mathbf{y}^r$ is the current estimate.

Why?

- Initialization using MF or DC.
- Smooth dual $\Rightarrow$ Sophisticated optimization.

Approach

- Block-coordinate descent tailored to this problem.

Proximal Minimization of LP (PROX-LP)

$$\begin{aligned}
\min_{\mathbf{y}} \quad & \tilde{E}(\mathbf{y}) + \frac{1}{2\eta} \left\| \mathbf{y} - \mathbf{y}^r \right\|^2 \ , \\
\text{s.t.} \quad & \mathbf{y} \in \mathcal{S} \ ,
\end{aligned}$$

where $\eta > 0$ and $\mathbf{y}^r$ is the current estimate.

Why?

- Initialization using MF or DC.
- Smooth dual $\Rightarrow$ Sophisticated optimization.

Approach

- Block-coordinate descent tailored to this problem.

Dual of the Proximal Problem

$$
\begin{aligned}
\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = {} & \frac{\eta}{2}\|A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u\|^2 \\
& + \langle A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u, \mathbf{y}^r\rangle - \langle \mathbf{1}, \boldsymbol{\beta}\rangle \ ,
\end{aligned}
$$

$$
\text{s.t.} \quad \gamma_{i:\lambda} \geq 0 \qquad \forall\, i \in \mathcal{V} \quad \forall\, \lambda \in \mathcal{L}\ ,
$$

$$
\boldsymbol{\alpha} \in \mathcal{C} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = \frac{K_{ij}}{2}, \forall\, i, j \neq i,\, \lambda \in \mathcal{L} \\ \alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \geq 0, \forall\, i, j \neq i,\, \lambda \in \mathcal{L} \end{array} \right\} .
$$

Block-coordinate descent

- β: Unconstrained $\Rightarrow$ Set derivative to zero.
- γ: Unbounded and separable $\Rightarrow$ Small QP for each pixel.
- $\alpha \in \mathcal{C}$: Compact domain $\Rightarrow$ Conditional gradient descent.

Dual of the Proximal Problem

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{\eta}{2}\|A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u\|^2 + \langle A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u, \mathbf{y}^r\rangle - \langle \mathbf{1}, \boldsymbol{\beta}\rangle \ ,$$

$$\text{s.t.} \quad \gamma_{i:\lambda} \geq 0 \quad \forall\, i \in \mathcal{V} \quad \forall\, \lambda \in \mathcal{L}\ ,$$

$$\boldsymbol{\alpha} \in \mathcal{C} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = \frac{K_{ij}}{2},\ \forall\, i,j \neq i,\ \lambda \in \mathcal{L} \\ \alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \geq 0,\ \forall\, i,j \neq i,\ \lambda \in \mathcal{L} \end{array} \right\} .$$

Block-coordinate descent

- $\boldsymbol{\beta}$: Unconstrained $\Rightarrow$ Set derivative to zero.
- $\boldsymbol{\gamma}$: Unbounded and separable $\Rightarrow$ Small QP for each pixel.
- $\boldsymbol{\alpha} \in \mathcal{C}$: Compact domain $\Rightarrow$ Conditional gradient descent.

Dual of the Proximal Problem

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{\eta}{2}\|A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u\|^2 + \langle A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u, \mathbf{y}^r\rangle - \langle \mathbf{1}, \boldsymbol{\beta}\rangle \ ,$$

$$\text{s.t.} \quad \gamma_{i:\lambda} \geq 0 \quad \forall\, i \in \mathcal{V} \quad \forall\, \lambda \in \mathcal{L} \ ,$$

$$\boldsymbol{\alpha} \in \mathcal{C} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = \frac{K_{ij}}{2}, \forall\, i, j \neq i,\, \lambda \in \mathcal{L} \\ \alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \geq 0, \forall\, i, j \neq i,\, \lambda \in \mathcal{L} \end{array} \right\} .$$

Block-coordinate descent

- $\boldsymbol{\beta}$: Unconstrained $\Rightarrow$ Set derivative to zero.
- $\boldsymbol{\gamma}$: Unbounded and separable $\Rightarrow$ Small QP for each pixel.
- $\boldsymbol{\alpha} \in \mathcal{C}$: Compact domain $\Rightarrow$ Conditional gradient descent.

Dual of the Proximal Problem

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{\eta}{2}\|A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u\|^2 + \langle A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u, \mathbf{y}^r\rangle - \langle \mathbf{1}, \boldsymbol{\beta}\rangle \ ,$$

$$\text{s.t.} \quad \gamma_{i:\lambda} \geq 0 \quad \forall\, i \in \mathcal{V} \quad \forall\, \lambda \in \mathcal{L} \ ,$$

$$\boldsymbol{\alpha} \in \mathcal{C} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = \frac{K_{ij}}{2}, \forall\, i, j \neq i,\, \lambda \in \mathcal{L} \\ \alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \geq 0, \forall\, i, j \neq i,\, \lambda \in \mathcal{L} \end{array} \right\} .$$

Block-coordinate descent

- $\boldsymbol{\beta}$: Unconstrained $\Rightarrow$ Set derivative to zero.
- $\boldsymbol{\gamma}$: Unbounded and separable $\Rightarrow$ Small QP for each pixel.
- $\boldsymbol{\alpha} \in \mathcal{C}$: Compact domain $\Rightarrow$ Conditional gradient descent.

Dual of the Proximal Problem

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{\eta}{2}\|A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u\|^2 + \langle A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u, \mathbf{y}^r\rangle - \langle \mathbf{1}, \boldsymbol{\beta}\rangle \ ,$$

$$\text{s.t.} \quad \gamma_{i:\lambda} \geq 0 \quad \forall\, i \in \mathcal{V} \quad \forall\, \lambda \in \mathcal{L} \ ,$$

$$\boldsymbol{\alpha} \in \mathcal{C} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = \frac{K_{ij}}{2}, \forall\, i, j \neq i,\, \lambda \in \mathcal{L} \\ \alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \geq 0, \forall\, i, j \neq i,\, \lambda \in \mathcal{L} \end{array} \right\} .$$

Block-coordinate descent

- $\boldsymbol{\beta}$: Unconstrained $\Rightarrow$ Set derivative to zero.
- $\boldsymbol{\gamma}$: Unbounded and separable $\Rightarrow$ Small QP for each pixel.
- $\boldsymbol{\alpha} \in \mathcal{C}$: Compact domain $\Rightarrow$ Conditional gradient descent.

Guarantees optimality since g is strictly convex and smooth.

Dual of the Proximal Problem

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{\eta}{2}\|A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u\|^2 + \langle A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\theta}_u, \mathbf{y}^r\rangle - \langle \mathbf{1}, \boldsymbol{\beta}\rangle \ ,$$

$$\text{s.t.} \quad \gamma_{i:\lambda} \geq 0 \quad \forall\, i \in \mathcal{V} \quad \forall\, \lambda \in \mathcal{L}\ ,$$

$$\boldsymbol{\alpha} \in \mathcal{C} = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{array}{l} \alpha^1_{ij:\lambda} + \alpha^2_{ij:\lambda} = \frac{K_{ij}}{2}, \forall\, i, j \neq i,\, \lambda \in \mathcal{L} \\ \alpha^1_{ij:\lambda}, \alpha^2_{ij:\lambda} \geq 0, \forall\, i, j \neq i,\, \lambda \in \mathcal{L} \end{array} \right\} .$$

Block-coordinate descent

- $\boldsymbol{\beta}$: Unconstrained $\Rightarrow$ Set derivative to zero.
- $\boldsymbol{\gamma}$: Unbounded and separable $\Rightarrow$ Small QP for each pixel.
- $\boldsymbol{\alpha} \in \mathcal{C}$: Compact domain $\Rightarrow$ Conditional gradient descent.

Guarantees optimality since g is strictly convex and smooth.

Conditional Gradient Descent

$$\min_{\boldsymbol{\alpha} \in \mathcal{C}} g(\boldsymbol{\alpha}) \ .$$

Requirements

- $g : \mathcal{C} \to \mathbb{R}$ is differentiable.
- $\mathcal{C} \subset \mathbb{R}^{\mathrm{N}}$ is convex and compact.

Conditional gradient (s)

- Minimize the first order Taylor approximation.

In our case

- Linear time conditional gradient computation.
- Optimal step size.

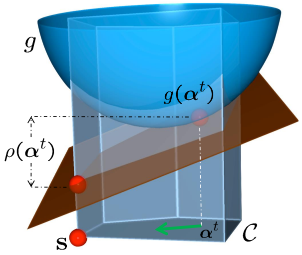


Image from [Lacoste-2012]

Conditional Gradient Descent

$$\min_{\boldsymbol{\alpha} \in \mathcal{C}} g(\boldsymbol{\alpha}) \ .$$

Requirements

- $g : \mathcal{C} \to \mathbb{R}$ is differentiable.
- $\mathcal{C} \subset \mathbb{R}^{\mathrm{N}}$ is convex and compact.

Conditional gradient (s)

- Minimize the first order Taylor approximation.

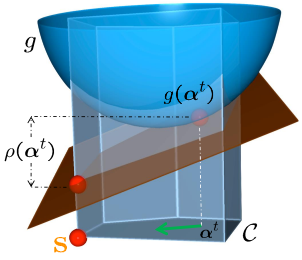


Image from [Lacoste-2012]

In our case

- Linear time conditional gradient computation.
- Optimal step size.

Conditional Gradient Descent

$$\min_{\boldsymbol{\alpha} \in \mathcal{C}} g(\boldsymbol{\alpha}) \ .$$

Requirements

- $g : \mathcal{C} \to \mathbb{R}$ is differentiable.
- $\mathcal{C} \subset \mathbb{R}^{\mathrm{N}}$ is convex and compact.

Conditional gradient (s)

- Minimize the first order Taylor approximation.

In our case

- Linear time conditional gradient computation.
- Optimal step size.

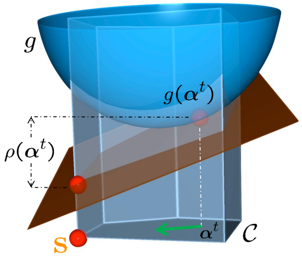


Image from [Lacoste-2012]

Conditional Gradient Computation

$$\forall\, i \in \mathcal{V}, \quad \tilde{s}_i = \sum_j K_{ij}\, \mathbb{1}[y_i \geq y_j]\ ,$$

where $K_{ij} = \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right)$, $y_i \in [0,1]$ and $\mathbf{f}_i \in \mathbb{R}^{\mathrm{d}}$.

Difficulty

- The permutohedral lattice based filtering method of [Adams-2010] cannot handle the ordering constraint.

Current method [Desmaison-2016]

- Repeated application of the original filtering method using a divide-and-conquer strategy $\Rightarrow \mathcal{O}(d^2 n \log(n))$ computations.

Our idea

- Discretize the interval $[0,1]$ to H levels and instantiate H permutohedral lattices $\Rightarrow \mathcal{O}(Hdn)$ computations ($H = 10$).

Conditional Gradient Computation

$$\forall\, i \in \mathcal{V}, \quad \tilde{s}_i = \sum_j K_{ij} \; \mathbb{1}[y_i \geq y_j] \;,$$

where $K_{ij} = \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right)$, $y_i \in [0,1]$ and $\mathbf{f}_i \in \mathbb{R}^{\mathrm{d}}$.

Difficulty

- The permutohedral lattice based filtering method of [Adams-2010] cannot handle the ordering constraint.

Current method [Desmaison-2016]

- Repeated application of the original filtering method using a divide-and-conquer strategy $\Rightarrow \mathcal{O}(d^2 n \log(n))$ computations.

Our idea

- Discretize the interval $[0,1]$ to H levels and instantiate H permutohedral lattices $\Rightarrow \mathcal{O}(Hdn)$ computations ($H = 10$).

Conditional Gradient Computation

$$\forall\, i \in \mathcal{V}, \quad \tilde{s}_i = \sum_j K_{ij} \; \mathbb{1}[y_i \geq y_j] \;,$$

where $K_{ij} = \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right)$, $y_i \in [0,1]$ and $\mathbf{f}_i \in \mathbb{R}^{\mathrm{d}}$.

Difficulty

- The permutohedral lattice based filtering method of [Adams-2010] cannot handle the ordering constraint.

Current method [Desmaison-2016]

- Repeated application of the original filtering method using a divide-and-conquer strategy $\Rightarrow \mathcal{O}(d^2 n \log(n))$ computations.

Our idea

- Discretize the interval $[0,1]$ to H levels and instantiate H permutohedral lattices $\Rightarrow \mathcal{O}(Hdn)$ computations ($H = 10$).

Conditional Gradient Computation

$$\forall\, i \in \mathcal{V}, \quad \tilde{s}_i = \sum_j K_{ij} \; \mathbb{1}[y_i \geq y_j] \;,$$

where $K_{ij} = \exp\left(\frac{-\|\mathbf{f}_i - \mathbf{f}_j\|^2}{2}\right)$, $y_i \in [0, 1]$ and $\mathbf{f}_i \in \mathbb{R}^{\mathrm{d}}$.

Difficulty

- The permutohedral lattice based filtering method of [Adams-2010] cannot handle the ordering constraint.

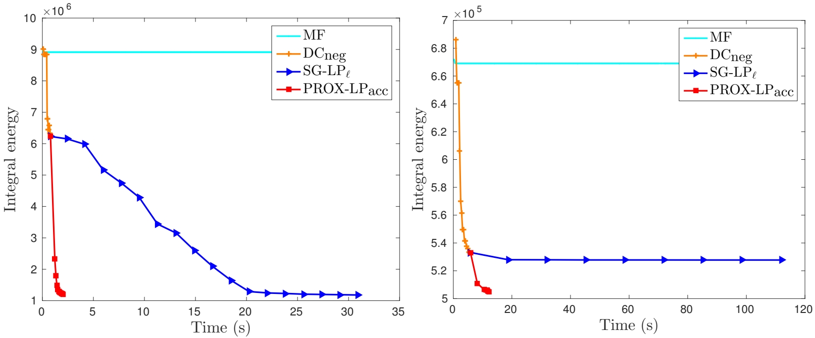
Current method [Desmaison-2016]

- Repeated application of the original filtering method using a divide-and-conquer strategy $\Rightarrow \mathcal{O}(d^2 n \log(n))$ computations.

Our idea

- Discretize the interval $[0, 1]$ to H levels and instantiate H permutohedral lattices $\Rightarrow \mathcal{O}(Hdn)$ computations ($H = 10$).

Segmentation Results



*Energy vs time plot for an image in (**left**) MSRC and (**right**) Pascal*

- Both LP minimization algorithms are initialized with $\mathrm{DC}_{\mathrm{neg}}$.

Segmentation Results

		Avg. E ($\times 10^3$)	Avg. T (s)	Acc.
MSRC	MF5	8078.0	0.2	79.33
	MF	8062.4	0.5	79.35
	DC_{neg}	3539.6	1.3	83.01
	SG-LP_{ℓ}	3335.6	13.6	83.15
	$\textbf{PROX-LP}_{\textbf{acc}}$	1340.0	3.7	84.16
Pascal	MF5	1220.8	0.8	79.13
	MF	1220.8	0.7	79.13
	DC_{neg}	629.5	3.7	80.43
	SG-LP_{ℓ}	617.1	84.4	80.49
	$\textbf{PROX-LP}_{\textbf{acc}}$	507.7	14.7	80.58

Results on the MSRC and Pascal datasets

Segmentation Results

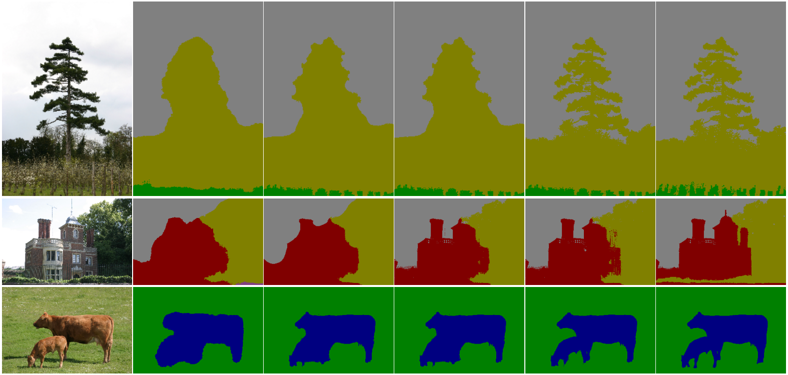
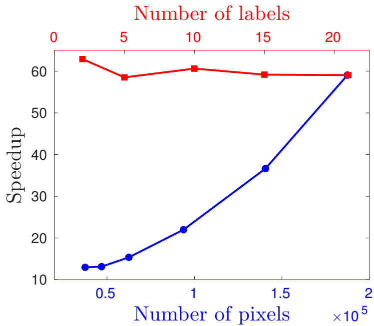


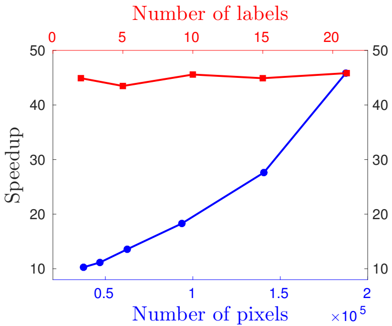
Image MF $\mathrm{DC}_{\mathrm{neg}}$ $\mathrm{SG\text{-}LP}_{\ell}$ $\mathrm{PROX\text{-}LP}_{\mathrm{acc}}$ Ground truth

Qualitative results on MSRC

Modified Filtering Method



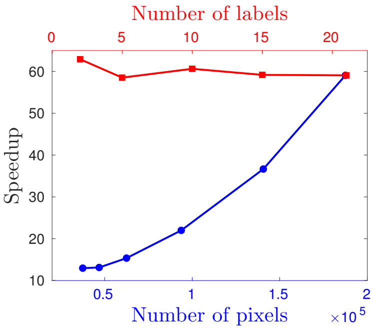
Spatial kernel ($d = 2$)



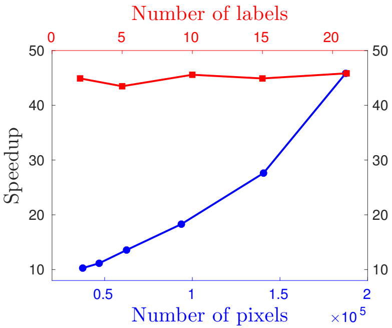
Bilateral kernel ($d = 5$)

Speedup of our modified filtering algorithm on a Pascal image

Modified Filtering Method



Spatial kernel ($d = 2$)



Bilateral kernel ($d = 5$)

Speedup of our modified filtering algorithm on a Pascal image

Speedup is around $45 - 65$ on the standard image.

Summary

- We have introduced the first LP minimization algorithm for dense CRFs whose iterations are linear in the number of pixels and labels.

Publication: CVPR, 2017

Code: `https://github.com/oval-group/DenseCRF`

Outline

Conclusion

- We have introduced three new algorithms for MRF optimization, targeting computer vision applications.

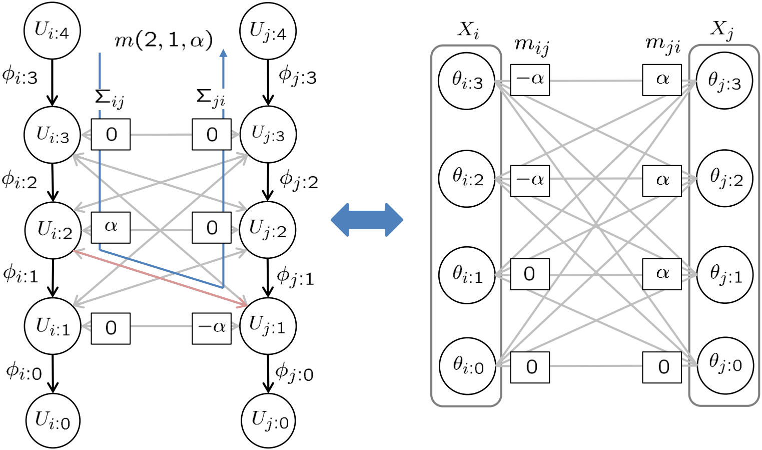
MEMF: A max-flow algorithm with $\mathcal{O}(\ell)$ memory reduction for Ishikawa type graphs.

IRGC: A move-making algorithm that can handle robust non-convex priors.

PROX-LP: An LP minimization algorithm for dense CRFs that has linear time iterations.

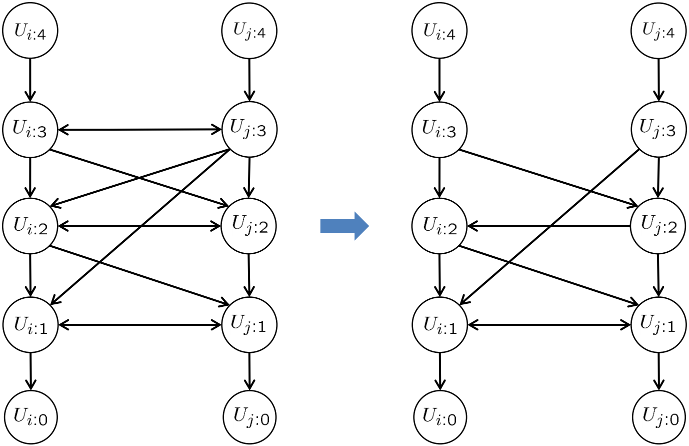
Thank you!

Flow vs Reparametrization



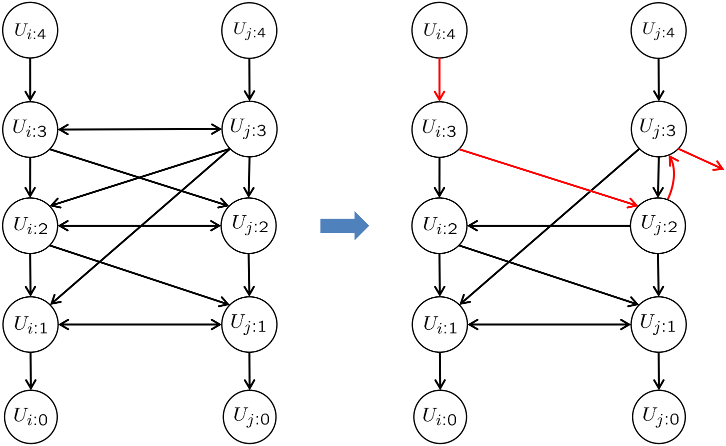
Flow vs reparametrization

Finding an Augmenting Path



Find augmenting paths on a subgraph

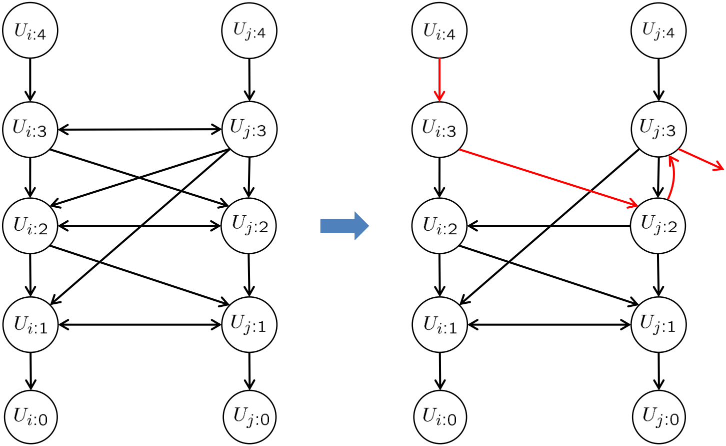
Finding an Augmenting Path



Find augmenting paths on a subgraph

Utilize upward infinite capacity edges in each column.

Finding an Augmenting Path



Find augmenting paths on a subgraph

Overall time complexity: $\mathcal{O}(nm\ell^6)$

Iteratively Reweighted Minimization

- Minimize the original energy $E(\mathbf{x}) = \sum_k h_k \circ f_k(\mathbf{x})$, by iteratively minimizing a surrogate energy $\tilde{E}(\mathbf{x}) = \sum_k w_k f_k(\mathbf{x})$.

Lemma (Monotonic decrease)

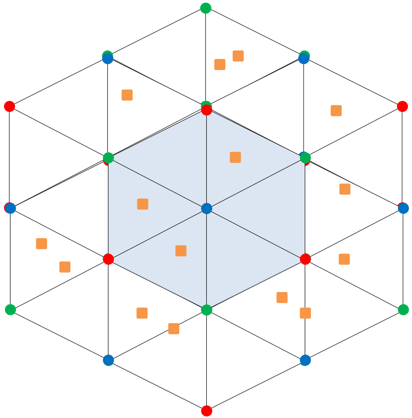
Given a set $\mathcal{X}$, functions $f_k : \mathcal{X} \to \mathcal{D}$ and concave functions $h_k : \mathcal{D} \to \mathbb{R}$, with $\mathcal{D} \subseteq \mathbb{R}$, such that,

$$\sum_k w_k^t f_k(\mathbf{x}^{t+1}) \leq \sum_k w_k^t f_k(\mathbf{x}^t) \ ,$$

where $w_k^t = h_k^s(f_k(\mathbf{x}^t))$ and $\mathbf{x}^t$ is the estimate of $\mathbf{x}$ at iteration t, then

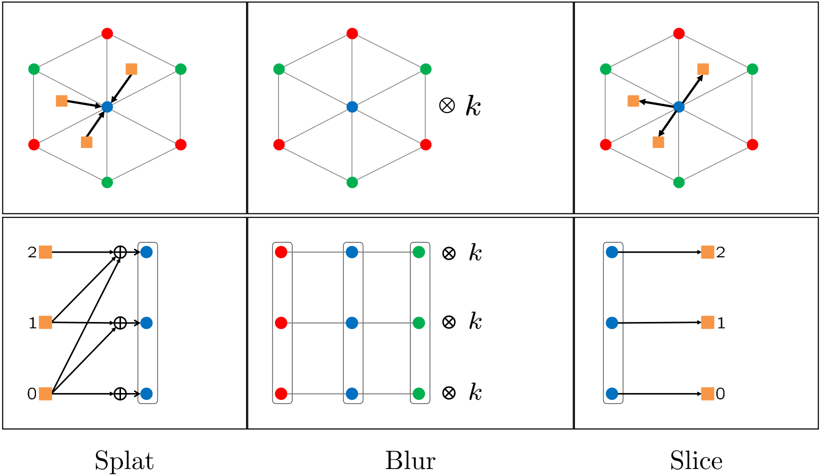
$$\sum_k h_k \circ f_k(\mathbf{x}^{t+1}) \leq \sum_k h_k \circ f_k(\mathbf{x}^t) \ .$$

Permutohedral Lattice



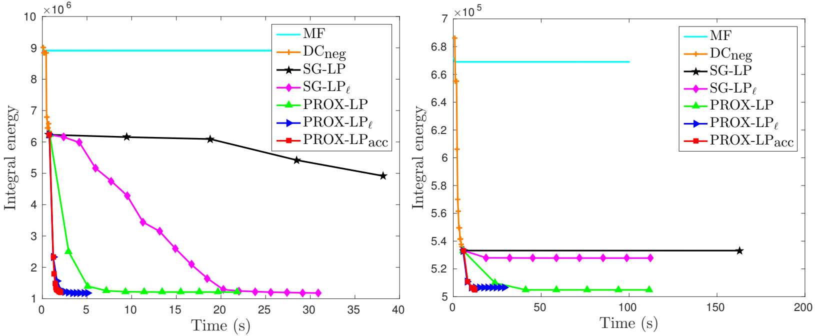
A 2-dimensional hyperplane tessellated by the permutohedral lattice.

Modified Filtering Algorithm



Top row: *Original filtering method.* **Bottom row:** *Our modified filtering method.* $H = 3$.

Segmentation Results



*Assignment energy as a function of time for an image in (**left**) MSRC and (**right**) Pascal*