$L_1 \&$ 

# Efficient algorithms for max-norm and lexicographically optimized labelings

#### Krzysztof Chris Ciesielski

#### Department of Mathematics, West Virginia University and MIPG, Department of Radiology, University of Pennsylvania

Joint work with Filip Malmberg and Robin Strand

#### University of Saõ Paulo, Brazil, May 27, 2019

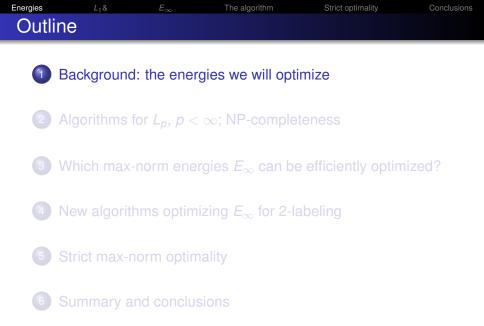
イロト イポト イヨト イヨト



Background: the energies we will optimize

- 2 Algorithms for  $L_p$ ,  $p < \infty$ ; NP-completeness
- 3 Which max-norm energies  $E_{\infty}$  can be efficiently optimized?
- 4 New algorithms optimizing  $E_{\infty}$  for 2-labeling
- 5 Strict max-norm optimality
- 6 Summary and conclusions

ヘロト 人間 ト ヘヨト ヘヨト



くロト (過) (目) (日)

## Energies $L_1$ $E_{\infty}$ The algorithm Strict optimality Optimization in image processing

- Many fundamental problems in image processing and computer vision, such as image filtering, segmentation, registration, and stereo vision, can naturally be formulated as optimization problems.
- Often, these optimization problems can be described as *labeling* problems, in which we wish to assign to each image element (pixel) an element from some finite set of labels.
- We identify each image with a vertex weighted graph *G* = (*V*, *E*, *f*), with vertices *V* being image voxels, edges *E* being pairs {*s*, *t*} of adjacent voxels, and *f*(*s*) image intensity at *s*. Its labeling is a map *l*: *V* → {0,...,*m*-1}, with *m* ≥ 2.

◆□ > ◆□ > ◆豆 > ◆豆 > →

## Energies $L_1$ $E_{\infty}$ The algorithmStrict optimalityOptimization in image processing

- Many fundamental problems in image processing and computer vision, such as image filtering, segmentation, registration, and stereo vision, can naturally be formulated as optimization problems.
- Often, these optimization problems can be described as *labeling* problems, in which we wish to assign to each image element (pixel) an element from some finite set of labels.
- We identify each image with a vertex weighted graph G = (V, E, f), with vertices V being image voxels, edges E being pairs {s, t} of adjacent voxels, and f(s) image intensity at s. Its labeling is a map ℓ: V → {0,...,m-1}, with m ≥ 2.

ヘロン 人間 とくほ とくほ とう

## Energies $L_1$ $E_{\infty}$ The algorithmStrict optimalityOptimization in image processing

- Many fundamental problems in image processing and computer vision, such as image filtering, segmentation, registration, and stereo vision, can naturally be formulated as optimization problems.
- Often, these optimization problems can be described as *labeling* problems, in which we wish to assign to each image element (pixel) an element from some finite set of labels.
- We identify each image with a vertex weighted graph *G* = (*V*, *E*, *f*), with vertices *V* being image voxels, edges *E* being pairs {*s*, *t*} of adjacent voxels, and *f*(*s*) image intensity at *s*. Its labeling is a map *l*: *V* → {0,...,*m*-1}, with *m* ≥ 2.

ヘロン ヘアン ヘビン ヘビン

## Energies $L_1$ $E_{\infty}$ The algorithmStrict optimalityOptimization in image processing

- Many fundamental problems in image processing and computer vision, such as image filtering, segmentation, registration, and stereo vision, can naturally be formulated as optimization problems.
- Often, these optimization problems can be described as *labeling* problems, in which we wish to assign to each image element (pixel) an element from some finite set of labels.
- We identify each image with a vertex weighted graph *G* = (*V*, *E*, *f*), with vertices *V* being image voxels, edges *E* being pairs {*s*, *t*} of adjacent voxels, and *f*(*s*) image intensity at *s*. Its labeling is a map *l*: *V* → {0,...,*m*-1}, with *m* ≥ 2.

ヘロト 人間 ト ヘヨト ヘヨト

Energies

With any image *n*-labeling  $\ell$  we associate local cost map  $\phi_{\ell} \colon V \cup \mathcal{E} \to [0, \infty]$  consisting of

• unary terms  $\phi_{\ell}(s) = \phi_{s}(\ell(s))$ , depending on  $s \in V$ , its label  $\ell(s)$ , and image intensity;

The algorithm

Conclusions

- pairwise terms φ<sub>ℓ</sub>(s, t) = φ<sub>s,t</sub>(ℓ(s), ℓ(t)), depending on {s, t} ∈ ε and their labeling. They reflect desirability of smoothness/regularity of labeling.
   All φ<sub>s,t</sub>(0,0), φ<sub>s,t</sub>(0,1), φ<sub>s,t</sub>(1,0), φ<sub>s,t</sub>(1,1) can be distinct
- $L_1$  (graph cut) energy is defined as

$$E_1(\ell) := \|\phi_\ell\|_1 = \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t)),$$

often represented as (with  $x_i$  denoting label of vertex i)

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{i \in \mathcal{E}} \phi_{ij}(x_i, x_j).$$

Energies

With any image *n*-labeling  $\ell$  we associate local cost map  $\phi_{\ell} \colon V \cup \mathcal{E} \to [0, \infty]$  consisting of

• unary terms  $\phi_{\ell}(s) = \phi_{s}(\ell(s))$ , depending on  $s \in V$ , its label  $\ell(s)$ , and image intensity;

The algorithm

Strict optimality

Conclusions

- pairwise terms φ<sub>ℓ</sub>(s, t) = φ<sub>s,t</sub>(ℓ(s), ℓ(t)), depending on {s, t} ∈ ε and their labeling. They reflect desirability of smoothness/regularity of labeling.
   All φ<sub>s,t</sub>(0,0), φ<sub>s,t</sub>(0,1), φ<sub>s,t</sub>(1,0), φ<sub>s,t</sub>(1,1) can be distinct
- $L_1$  (graph cut) energy is defined as

$$E_1(\ell) := \|\phi_\ell\|_1 = \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t)),$$

often represented as (with  $x_i$  denoting label of vertex i)

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{i \in \mathcal{E}} \phi_{ij}(x_i, x_j).$$

Energies

With any image *n*-labeling  $\ell$  we associate local cost map  $\phi_{\ell} \colon V \cup \mathcal{E} \to [0, \infty]$  consisting of

• unary terms  $\phi_{\ell}(s) = \phi_{s}(\ell(s))$ , depending on  $s \in V$ , its label  $\ell(s)$ , and image intensity;

The algorithm

Strict optimality

Conclusions

pairwise terms φ<sub>ℓ</sub>(s, t) = φ<sub>s,t</sub>(ℓ(s), ℓ(t)), depending on {s, t} ∈ E and their labeling. They reflect desirability of smoothness/regularity of labeling.
 All φ<sub>s,t</sub>(0,0), φ<sub>s,t</sub>(0,1), φ<sub>s,t</sub>(1,0), φ<sub>s,t</sub>(1,1) can be distinct

 $L_1$  (graph cut) energy is defined as

 $E_1(\ell) := \|\phi_\ell\|_1 = \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t)),$ 

often represented as (with  $x_i$  denoting label of vertex i)

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{i \in \mathcal{E}} \phi_{ij}(x_i, x_j).$$

> < 臣 > < 臣 >

Energies

With any image *n*-labeling  $\ell$  we associate local cost map  $\phi_{\ell} \colon V \cup \mathcal{E} \to [0, \infty]$  consisting of

• unary terms  $\phi_{\ell}(s) = \phi_{s}(\ell(s))$ , depending on  $s \in V$ , its label  $\ell(s)$ , and image intensity;

The algorithm

Strict optimality

Conclusions

pairwise terms φ<sub>ℓ</sub>(s, t) = φ<sub>s,t</sub>(ℓ(s), ℓ(t)), depending on {s, t} ∈ E and their labeling. They reflect desirability of smoothness/regularity of labeling.
 All φ<sub>s,t</sub>(0,0), φ<sub>s,t</sub>(0,1), φ<sub>s,t</sub>(1,0), φ<sub>s,t</sub>(1,1) can be distinct!

#### $L_1$ (graph cut) energy is defined as

$$E_1(\ell) := \|\phi_\ell\|_1 = \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t)),$$

often represented as (with  $x_i$  denoting label of vertex i)

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

Energies

With any image *n*-labeling  $\ell$  we associate local cost map  $\phi_{\ell} \colon V \cup \mathcal{E} \to [0, \infty]$  consisting of

• unary terms  $\phi_{\ell}(s) = \phi_{s}(\ell(s))$ , depending on  $s \in V$ , its label  $\ell(s)$ , and image intensity;

The algorithm

Strict optimality

Conclusions

pairwise terms φ<sub>ℓ</sub>(s, t) = φ<sub>s,t</sub>(ℓ(s), ℓ(t)), depending on {s, t} ∈ E and their labeling. They reflect desirability of smoothness/regularity of labeling.
 All φ<sub>s,t</sub>(0,0), φ<sub>s,t</sub>(0,1), φ<sub>s,t</sub>(1,0), φ<sub>s,t</sub>(1,1) can be distinct!

L1 (graph cut) energy is defined as

 $E_1(\ell) := \|\phi_\ell\|_1 = \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t)),$ 

often represented as (with  $x_i$  denoting label of vertex i)

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

Energies

With any image *n*-labeling  $\ell$  we associate local cost map  $\phi_{\ell} \colon V \cup \mathcal{E} \to [0, \infty]$  consisting of

• unary terms  $\phi_{\ell}(s) = \phi_{s}(\ell(s))$ , depending on  $s \in V$ , its label  $\ell(s)$ , and image intensity;

The algorithm

pairwise terms φ<sub>ℓ</sub>(s, t) = φ<sub>s,t</sub>(ℓ(s), ℓ(t)), depending on {s, t} ∈ E and their labeling. They reflect desirability of smoothness/regularity of labeling.
 All φ<sub>s,t</sub>(0,0), φ<sub>s,t</sub>(0,1), φ<sub>s,t</sub>(1,0), φ<sub>s,t</sub>(1,1) can be distinct!

L1 (graph cut) energy is defined as

$$E_1(\ell) := \|\phi_\ell\|_1 = \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t)),$$

often represented as (with  $x_i$  denoting label of vertex i)

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

Strict optimality

Energies  $L_1$   $E_\infty$  The algorithm  $L_p$  energies: the cases of  $p \in (1,\infty]$ 

For  $p \in [1, \infty)$ :

$$E_{p}(\ell) := \|\phi_{\ell}\|_{p} = \left(\sum_{s \in V} (\phi_{s}(\ell(s)))^{p} + \sum_{\{s,t\} \in \mathcal{E}} (\phi_{st}(\ell(s),\ell(t)))^{p}\right)^{1/p}$$

For  $p = \infty$  (of main interest here)

$$\mathsf{E}_{\infty}(\ell) := \|\phi_{\ell}\|_{\infty} = \max\left\{\max_{s\in V} \phi_s(\ell(s)), \max_{\{s,t\}\in\mathcal{E}} \phi_{st}(\ell(s),\ell(t))
ight\}$$

Standard analysis fact:  $E_p(\ell) \nearrow_{p \to \infty} E_{\infty}(\ell)$ .

Energies  $L_1$   $E_\infty$  The algorithm  $L_p$  energies: the cases of  $p \in (1,\infty]$ 

For  $p \in [1, \infty)$ :

$$E_{\rho}(\ell) := \|\phi_{\ell}\|_{\rho} = \left(\sum_{\boldsymbol{s}\in V} (\phi_{\boldsymbol{s}}(\ell(\boldsymbol{s})))^{\rho} + \sum_{\{\boldsymbol{s},t\}\in\mathcal{E}} (\phi_{\boldsymbol{s}t}(\ell(\boldsymbol{s}),\ell(t)))^{\rho}\right)^{1/\rho}$$

For  $p = \infty$  (of main interest here)

$$\mathsf{E}_{\infty}(\ell) := \|\phi_{\ell}\|_{\infty} = \max\left\{\max_{s \in V} \phi_{s}(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$$

Standard analysis fact:  $E_p(\ell) \nearrow_{p \to \infty} E_{\infty}(\ell)$ .

Energies  $L_1$   $E_\infty$  The algorithm  $L_p$  energies: the cases of  $p \in (1,\infty]$ 

For  $p \in [1, \infty)$ :

$$E_{\rho}(\ell) := \|\phi_{\ell}\|_{\rho} = \left(\sum_{\boldsymbol{s}\in V} (\phi_{\boldsymbol{s}}(\ell(\boldsymbol{s})))^{\rho} + \sum_{\{\boldsymbol{s},t\}\in\mathcal{E}} (\phi_{\boldsymbol{s}t}(\ell(\boldsymbol{s}),\ell(t)))^{\rho}\right)^{1/\rho}$$

For  $p = \infty$  (of main interest here)

$$\mathsf{E}_{\infty}(\ell) := \|\phi_{\ell}\|_{\infty} = \max\left\{\max_{s \in V} \phi_{s}(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$$

Standard analysis fact:  $E_p(\ell) \nearrow_{p \to \infty} E_{\infty}(\ell)$ .

Strict optimality

- The value *p* can be seen as a parameter controlling the balance between minimizing the overall cost  $E_p(\ell)$  versus minimizing the magnitude of the individual terms  $\phi_s(\ell(s))$  and  $\phi_{st}(\ell(s), \ell(t))$ .
- For *p* = 1, the optimal labeling may contain (few) arbitrarily large individual terms as long as the sum of the terms is small.
- As *p* increases, a larger penalty is assigned to solutions containing large individual terms. This forces local errors to be distributed more evenly across the image domain.

ヘロト ヘ戸ト ヘヨト ヘヨト

- The value *p* can be seen as a parameter controlling the balance between minimizing the overall cost  $E_p(\ell)$  versus minimizing the magnitude of the individual terms  $\phi_s(\ell(s))$  and  $\phi_{st}(\ell(s), \ell(t))$ .
- For *p* = 1, the optimal labeling may contain (few) arbitrarily large individual terms as long as the sum of the terms is small.
- As *p* increases, a larger penalty is assigned to solutions containing large individual terms. This forces local errors to be distributed more evenly across the image domain.

ヘロト ヘ戸ト ヘヨト ヘヨト

- The value *p* can be seen as a parameter controlling the balance between minimizing the overall cost  $E_p(\ell)$  versus minimizing the magnitude of the individual terms  $\phi_s(\ell(s))$  and  $\phi_{st}(\ell(s), \ell(t))$ .
- For *p* = 1, the optimal labeling may contain (few) arbitrarily large individual terms as long as the sum of the terms is small.
- As *p* increases, a larger penalty is assigned to solutions containing large individual terms. This forces local errors to be distributed more evenly across the image domain.



- 2 Algorithms for  $L_p$ ,  $p < \infty$ ; NP-completeness
- ${}_{\textcircled{3}}$  Which max-norm energies  ${\it E}_{\infty}$  can be efficiently optimized?
- [4] New algorithms optimizing  $E_\infty$  for 2-labeling
- 5 Strict max-norm optimality
- 6 Summary and conclusions

ヘロト ヘ戸ト ヘヨト ヘヨト

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds);
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

Min-cut/max-flow (efficiency between  $O(n^2 \ln n)$  and  $O(n^3)$ ) algorithm returns optimized labeling **for 2-labeling**.

Here and below  $n := |V \cup \mathcal{E}|$ .

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds);
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

Min-cut/max-flow (efficiency between  $O(n^2 \ln n)$  and  $O(n^3)$ ) algorithm returns optimized labeling **for 2-labeling**.

Here and below  $n := |V \cup \mathcal{E}|$ .

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds);
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

Min-cut/max-flow (efficiency between  $O(n^2 \ln n)$  and  $O(n^3)$ ) algorithm returns optimized labeling **for 2-labeling**.

Here and below  $n := |V \cup \mathcal{E}|$ .

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds);
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

Min-cut/max-flow (efficiency between  $O(n^2 \ln n)$  and  $O(n^3)$ ) algorithm returns optimized labeling **for 2-labeling**.

Here and below  $n := |V \cup \mathcal{E}|$ .

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds);
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

Min-cut/max-flow (efficiency between  $O(n^2 \ln n)$  and  $O(n^3)$ ) algorithm returns optimized labeling **for 2-labeling**.

Here and below  $n := |V \cup \mathcal{E}|$ .

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds);
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

Min-cut/max-flow (efficiency between  $O(n^2 \ln n)$  and  $O(n^3)$ ) algorithm returns optimized labeling **for 2-labeling**.

Here and below  $n := |V \cup \mathcal{E}|$ .

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds);
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

Min-cut/max-flow (efficiency between  $O(n^2 \ln n)$  and  $O(n^3)$ ) algorithm returns optimized labeling **for 2-labeling**.

Here and below  $n := |V \cup \mathcal{E}|$ .

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

 $E_1$  (for 2-labeling) is submodular provided, for every  $\{s, t\} \in \mathcal{E}$ ,

 $\phi_{st}(0,0) + \phi_{st}(1,1) \le \phi_{st}(0,1) + \phi_{st}(1,0).$ 

#### Theorem (Kolmogorov & Zabih 2004)

- If *E*<sub>1</sub> is submodular, then min-cut/max-flow algorithm returns optimized labeling.
- If  $E_1$  is **NOT** submodular, then minimizing  $E_1$  is NP-hard.

ヘロト ヘ戸ト ヘヨト ヘヨト

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

 $E_1$  (for 2-labeling) is submodular provided, for every  $\{s, t\} \in \mathcal{E}$ ,

 $\phi_{st}(0,0) + \phi_{st}(1,1) \le \phi_{st}(0,1) + \phi_{st}(1,0).$ 

#### Theorem (Kolmogorov & Zabih 2004)

• If *E*<sub>1</sub> is submodular, then min-cut/max-flow algorithm returns optimized labeling.

• If  $E_1$  is **NOT** submodular, then minimizing  $E_1$  is NP-hard.

くロト (過) (目) (日)

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

 $E_1$  (for 2-labeling) is submodular provided, for every  $\{s, t\} \in \mathcal{E}$ ,

 $\phi_{st}(0,0) + \phi_{st}(1,1) \le \phi_{st}(0,1) + \phi_{st}(1,0).$ 

#### Theorem (Kolmogorov & Zabih 2004)

 If E<sub>1</sub> is submodular, then min-cut/max-flow algorithm returns optimized labeling.

• If  $E_1$  is **NOT** submodular, then minimizing  $E_1$  is NP-hard.

くロト (過) (目) (日)

$$E_1(\ell) := \sum_{s \in V} \phi_s(\ell(s)) + \sum_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))$$

 $E_1$  (for 2-labeling) is submodular provided, for every  $\{s, t\} \in \mathcal{E}$ ,

 $\phi_{st}(0,0) + \phi_{st}(1,1) \le \phi_{st}(0,1) + \phi_{st}(1,0).$ 

#### Theorem (Kolmogorov & Zabih 2004)

- If E<sub>1</sub> is submodular, then min-cut/max-flow algorithm returns optimized labeling.
- If  $E_1$  is **NOT** submodular, then minimizing  $E_1$  is NP-hard.

イロト イ理ト イヨト イヨト

$$(E_{\rho}(\ell))^{\rho} := \sum_{s \in V} (\phi_s(\ell(s)))^{\rho} + \sum_{\{s,t\} \in \mathcal{E}} (\phi_{st}(\ell(s),\ell(t)))^{\rho}$$

 $E_p$  is *p*-submodular provided, for every  $\{s, t\} \in \mathcal{E}$ ,

 $\phi_{st}(0,0)^{p} + \phi_{st}(1,1)^{p} \le \phi_{st}(0,1)^{p} + \phi_{st}(1,0)^{p}.$ 

#### Corollary (Obvious, Malmberg & Strand, IWCIA 2018)

- If E<sub>p</sub> is p-submodular, then min-cut/max-flow algorithm returns optimized labeling.
- If  $E_p$  is **NOT** *p*-submodular, then minimizing  $E_p$  is NP-hard.

ヘロト ヘ戸ト ヘヨト ヘヨト

Strict optimality

$$(E_{\rho}(\ell))^{\rho} := \sum_{s \in V} (\phi_s(\ell(s)))^{\rho} + \sum_{\{s,t\} \in \mathcal{E}} (\phi_{st}(\ell(s),\ell(t)))^{\rho}$$

 $E_p$  is *p*-submodular provided, for every  $\{s, t\} \in \mathcal{E}$ ,

 $\phi_{st}(0,0)^{p} + \phi_{st}(1,1)^{p} \le \phi_{st}(0,1)^{p} + \phi_{st}(1,0)^{p}.$ 

#### Corollary (Obvious, Malmberg & Strand, IWCIA 2018)

- If E<sub>p</sub> is p-submodular, then min-cut/max-flow algorithm returns optimized labeling.
- If  $E_p$  is **NOT** *p*-submodular, then minimizing  $E_p$  is NP-hard.

ヘロト 人間 ト ヘヨト ヘヨト

Strict optimality

$$(E_{\rho}(\ell))^{\rho} := \sum_{s \in V} (\phi_s(\ell(s)))^{\rho} + \sum_{\{s,t\} \in \mathcal{E}} (\phi_{st}(\ell(s),\ell(t)))^{\rho}$$

 $E_p$  is *p*-submodular provided, for every  $\{s, t\} \in \mathcal{E}$ ,

 $\phi_{st}(0,0)^{p} + \phi_{st}(1,1)^{p} \le \phi_{st}(0,1)^{p} + \phi_{st}(1,0)^{p}.$ 

#### Corollary (Obvious, Malmberg & Strand, IWCIA 2018)

 If E<sub>p</sub> is p-submodular, then min-cut/max-flow algorithm returns optimized labeling.

• If  $E_p$  is **NOT** *p*-submodular, then minimizing  $E_p$  is NP-hard.

ヘロト ヘアト ヘビト ヘビト

Strict optimality

$$(E_{\rho}(\ell))^{\rho} := \sum_{s \in V} (\phi_s(\ell(s)))^{\rho} + \sum_{\{s,t\} \in \mathcal{E}} (\phi_{st}(\ell(s),\ell(t)))^{\rho}$$

 $E_p$  is *p*-submodular provided, for every  $\{s, t\} \in \mathcal{E}$ ,

 $\phi_{st}(0,0)^{\rho} + \phi_{st}(1,1)^{\rho} \le \phi_{st}(0,1)^{\rho} + \phi_{st}(1,0)^{\rho}.$ 

#### Corollary (Obvious, Malmberg & Strand, IWCIA 2018)

- If E<sub>p</sub> is p-submodular, then min-cut/max-flow algorithm returns optimized labeling.
- If  $E_p$  is **NOT** *p*-submodular, then minimizing  $E_p$  is NP-hard.

イロト イ押ト イヨト イヨトー

Strict optimality

Energies  $L_1$   $E_\infty$  The algorithm Strict optimality Conclusions  $E_
ho(\ell)$  with 1  $\leq 
ho < \infty$  VS  $E_\infty(\ell)$ 

 $\phi_{st}(0,0)^p + \phi_{st}(1,1)^p \le \phi_{st}(0,1)^p + \phi_{st}(1,0)^p.$ 

*p*-submodular for every  $p < \infty$  implies  $\infty$ -submodularity:

 $\max\{\phi_{st}(0,0),\phi_{st}(1,1)\} \le \max\{\phi_{st}(1,0),\phi_{st}(0,1)\}.$ 

Theorem (Malmberg & Strand, IWCIA 2018)

1- and  $\infty$ -submodularity imply p-submodularity for all p. In such case min-cut/max-flow algorithm optimizes  $E_p$  for every  $p < \infty$ .

Actually,  $\phi$  is  $\infty$ -submodular iff there is an N so that  $\phi$  is *p*-submodular for all  $p \in (N, \infty)$ . Energies  $L_1$   $E_\infty$  The algorithm Strict optimality Conclusions  $E_
ho(\ell)$  with 1  $\leq 
ho < \infty$  VS  $E_\infty(\ell)$ 

$$\phi_{st}(0,0)^{\rho} + \phi_{st}(1,1)^{\rho} \le \phi_{st}(0,1)^{\rho} + \phi_{st}(1,0)^{\rho}.$$

*p*-submodular for every  $p < \infty$  implies  $\infty$ -submodularity:

 $\max\{\phi_{st}(0,0),\phi_{st}(1,1)\} \le \max\{\phi_{st}(1,0),\phi_{st}(0,1)\}.$ 

Theorem (Malmberg & Strand, IWCIA 2018)

1- and  $\infty$ -submodularity imply p-submodularity for all p. In such case min-cut/max-flow algorithm optimizes  $E_p$  for every  $p < \infty$ .

Energies  $L_1$   $E_\infty$  The algorithm Strict optimality Conclusions  $E_
ho(\ell)$  with 1  $\leq 
ho < \infty$  VS  $E_\infty(\ell)$ 

$$\phi_{st}(0,0)^{\rho} + \phi_{st}(1,1)^{\rho} \le \phi_{st}(0,1)^{\rho} + \phi_{st}(1,0)^{\rho}.$$

*p*-submodular for every  $p < \infty$  implies  $\infty$ -submodularity:

 $\max\{\phi_{st}(0,0),\phi_{st}(1,1)\} \le \max\{\phi_{st}(1,0),\phi_{st}(0,1)\}.$ 

Theorem (Malmberg & Strand, IWCIA 2018)

1- and  $\infty$ -submodularity imply p-submodularity for all p. In such case min-cut/max-flow algorithm optimizes  $E_p$  for every  $p < \infty$ .

Energies  $L_1$   $E_\infty$  The algorithm Strict optimality Conclusions  $E_
ho(\ell)$  with  $1 \le 
ho < \infty$  VS  $E_\infty(\ell)$ 

$$\phi_{st}(0,0)^{\rho} + \phi_{st}(1,1)^{\rho} \le \phi_{st}(0,1)^{\rho} + \phi_{st}(1,0)^{\rho}.$$

*p*-submodular for every  $p < \infty$  implies  $\infty$ -submodularity:

 $\max\{\phi_{st}(0,0),\phi_{st}(1,1)\} \le \max\{\phi_{st}(1,0),\phi_{st}(0,1)\}.$ 

Theorem (Malmberg & Strand, IWCIA 2018)

1- and  $\infty$ -submodularity imply p-submodularity for all p. In such case min-cut/max-flow algorithm optimizes  $E_p$  for every  $p < \infty$ .

Energies  $L_1$   $E_\infty$  The algorithm Strict optimality Conclusions  $E_
ho(\ell)$  with  $1 \le 
ho < \infty$  VS  $E_\infty(\ell)$ 

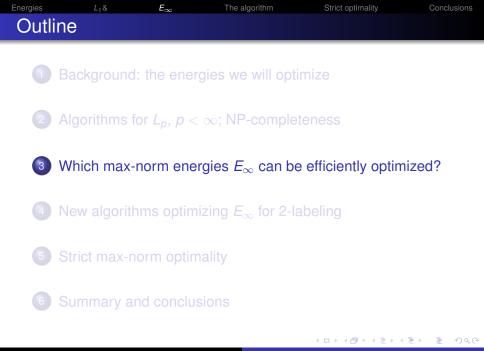
$$\phi_{st}(0,0)^{\rho} + \phi_{st}(1,1)^{\rho} \le \phi_{st}(0,1)^{\rho} + \phi_{st}(1,0)^{\rho}.$$

*p*-submodular for every  $p < \infty$  implies  $\infty$ -submodularity:

 $\max\{\phi_{st}(0,0),\phi_{st}(1,1)\} \le \max\{\phi_{st}(1,0),\phi_{st}(0,1)\}.$ 

Theorem (Malmberg & Strand, IWCIA 2018)

1- and  $\infty$ -submodularity imply p-submodularity for all p. In such case min-cut/max-flow algorithm optimizes  $E_p$  for every  $p < \infty$ .



 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

We get FC segmentations (as minimization, not maximization)

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds, when  $= \infty$ );
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

**Dijkstra algorithm** (efficiency between O(n) and  $O(n \ln n)$ ) returns optimized labeling for *m*-labeling **for arbitrary large** *m*! Better than for  $E_1(\ell)$  (i.e., GC) segmentations.

**Q.** For what other  $E_{\infty}$ s are there efficient optimizing algorithms?

◆□▶ ◆□▶ ★ □▶ ★ □▶ → □ → の Q ()

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

We get FC segmentations (as minimization, not maximization)

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds, when  $= \infty$ );
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t)$ ;
- Cost of cut: φ<sub>st</sub>(ℓ(s), ℓ(t)) > 0 (depending of f(s), f(t)) when ℓ(s) ≠ ℓ(t).

**Dijkstra algorithm** (efficiency between O(n) and  $O(n \ln n)$ ) returns optimized labeling for *m*-labeling **for arbitrary large** *m*! Better than for  $E_1(\ell)$  (i.e., GC) segmentations.

**Q.** For what other  $E_{\infty}$ s are there efficient optimizing algorithms?

◆□▶ ◆□▶ ◆三▶ ◆三▶ ● ○ ○ ○

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

We get FC segmentations (as minimization, not maximization)

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds, when  $= \infty$ );
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t)$ ;
- Cost of cut: φ<sub>st</sub>(ℓ(s), ℓ(t)) > 0 (depending of f(s), f(t)) when ℓ(s) ≠ ℓ(t).

**Dijkstra algorithm** (efficiency between O(n) and  $O(n \ln n)$ ) returns optimized labeling for *m*-labeling **for arbitrary large** *m*! Better than for  $E_1(\ell)$  (i.e., GC) segmentations.

**Q.** For what other  $E_{\infty}$ s are there efficient optimizing algorithms?

◆□▶ ◆□▶ ★ □▶ ★ □▶ → □ → の Q ()

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

We get FC segmentations (as minimization, not maximization)

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds, when  $= \infty$ );
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

**Dijkstra algorithm** (efficiency between O(n) and  $O(n \ln n)$ ) returns optimized labeling for *m*-labeling **for arbitrary large** *m*! Better than for  $E_1(\ell)$  (i.e., GC) segmentations.

**Q.** For what other  $E_{\infty}$ s are there efficient optimizing algorithms?

◆□▶ ◆□▶ ★ □▶ ★ □▶ → □ → の Q ()

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

We get FC segmentations (as minimization, not maximization)

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds, when  $= \infty$ );
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

**Dijkstra algorithm** (efficiency between O(n) and  $O(n \ln n)$ ) returns optimized labeling for *m*-labeling **for arbitrary large** *m*! Better than for  $E_1(\ell)$  (i.e., GC) segmentations.

**Q.** For what other  $E_{\infty}$ s are there efficient optimizing algorithms?

◆□ ▶ ◆□ ▶ ◆ □ ▶ ◆ □ ▶ ● □ ● ● ● ●

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

We get FC segmentations (as minimization, not maximization)

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds, when  $= \infty$ );
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

**Dijkstra algorithm** (efficiency between O(n) and  $O(n \ln n)$ ) returns optimized labeling for *m*-labeling **for arbitrary large** *m*! Better than for  $E_1(\ell)$  (i.e., GC) segmentations.

**Q.** For what other  $E_{\infty}$ s are there efficient optimizing algorithms?

◆□ ▶ ◆□ ▶ ◆ □ ▶ ◆ □ ▶ ● □ ● ● ● ●

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

We get FC segmentations (as minimization, not maximization)

- $\phi_s(\ell(s)) = 0$  in all cases (except seeds, when  $= \infty$ );
- $\phi_{st}(\ell(s), \ell(t)) = 0$  when  $\ell(s) = \ell(t);$
- Cost of cut:  $\phi_{st}(\ell(s), \ell(t)) > 0$  (depending of f(s), f(t)) when  $\ell(s) \neq \ell(t)$ .

**Dijkstra algorithm** (efficiency between O(n) and  $O(n \ln n)$ ) returns optimized labeling for *m*-labeling **for arbitrary large** *m*! Better than for  $E_1(\ell)$  (i.e., GC) segmentations.

**Q.** For what other  $E_{\infty}$ s are there efficient optimizing algorithms?

ヘロト ヘアト ヘビト ヘビト

#### YES! $\infty$ -sub algotithm

Theorem (Malmberg, Ciesielski, Strand, DGCI 2019)

There is an algorithm, quasi-linear with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $\infty$ -submodular energy  $E_{\infty}$ .

The algorithm, efficiency between O(n) and  $O(n \ln n)$ , is NOT Dijkstra-like! This is all that is in the DGCI 2019 paper.

Natural questions, towards post DGCI 2019 work:

Q1: Is  $\infty$ -submodularity assumption essential in the thm?

# YES! $\infty$ -sub algotithm

Theorem (Malmberg, Ciesielski, Strand, DGCI 2019)

There is an algorithm, quasi-linear with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $\infty$ -submodular energy  $E_{\infty}$ .

The algorithm, efficiency between O(n) and  $O(n \ln n)$ , is NOT Dijkstra-like! This is all that is in the DGCI 2019 paper.

Natural questions, towards post DGCI 2019 work:

Q1: Is  $\infty$ -submodularity assumption essential in the thm?

# YES! $\infty$ -sub algotithm

#### Theorem (Malmberg, Ciesielski, Strand, DGCI 2019)

There is an algorithm, quasi-linear with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $\infty$ -submodular energy  $E_{\infty}$ .

The algorithm, efficiency between O(n) and  $O(n \ln n)$ , is NOT Dijkstra-like! This is all that is in the DGCI 2019 paper.

Natural questions, towards post DGCI 2019 work:

Q1: Is  $\infty$ -submodularity assumption essential in the thm?



#### YES! $\infty$ -sub algotithm

Theorem (Malmberg, Ciesielski, Strand, DGCI 2019)

There is an algorithm, quasi-linear with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $\infty$ -submodular energy  $E_{\infty}$ .

The algorithm, efficiency between O(n) and  $O(n \ln n)$ , is NOT Dijkstra-like! This is all that is in the DGCI 2019 paper.

Natural questions, towards post DGCI 2019 work:

Q1: Is  $\infty$ -submodularity assumption essential in the thm?

# YES! $\infty$ -sub algotithm

Theorem (Malmberg, Ciesielski, Strand, DGCI 2019)

There is an algorithm, quasi-linear with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $\infty$ -submodular energy  $E_{\infty}$ .

The algorithm, efficiency between O(n) and  $O(n \ln n)$ , is NOT Dijkstra-like!

This is all that is in the DGCI 2019 paper.

Natural questions, towards post DGCI 2019 work:

Q1: Is  $\infty$ -submodularity assumption essential in the thm?

# YES! $\infty$ -sub algotithm

Theorem (Malmberg, Ciesielski, Strand, DGCI 2019)

There is an algorithm, quasi-linear with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $\infty$ -submodular energy  $E_{\infty}$ .

The algorithm, efficiency between O(n) and  $O(n \ln n)$ , is NOT Dijkstra-like! This is all that is in the DGCI 2019 paper.

Natural questions, towards post DGCI 2019 work:

Q1: Is  $\infty$ -submodularity assumption essential in the thm?

# YES! $\infty$ -sub algotithm

Theorem (Malmberg, Ciesielski, Strand, DGCI 2019)

There is an algorithm, quasi-linear with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $\infty$ -submodular energy  $E_{\infty}$ .

The algorithm, efficiency between O(n) and  $O(n \ln n)$ , is NOT Dijkstra-like! This is all that is in the DGCI 2019 paper.

Natural questions, towards post DGCI 2019 work:

Q1: Is  $\infty$ -submodularity assumption essential in the thm?

# YES! $\infty$ -sub algotithm

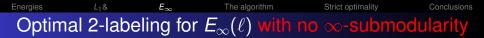
Theorem (Malmberg, Ciesielski, Strand, DGCI 2019)

There is an algorithm, quasi-linear with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $\infty$ -submodular energy  $E_{\infty}$ .

The algorithm, efficiency between O(n) and  $O(n \ln n)$ , is NOT Dijkstra-like! This is all that is in the DGCI 2019 paper.

Natural questions, towards post DGCI 2019 work:

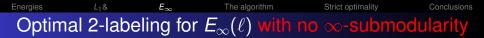
Q1: Is  $\infty$ -submodularity assumption essential in the thm?



Theorem (Malmberg, Ciesielski, Strand; 2019 ???) There is an algorithm,  $O(n^2)$  with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $E_{\infty}(\ell)$ :  $\max \{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\}.$ 

More about the algorithm latter.

イロト イポト イヨト イヨト



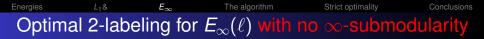
# Theorem (Malmberg, Ciesielski, Strand; 2019 ???)

There is an algorithm,  $O(n^2)$  with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $E_{\infty}(\ell)$ :

 $\max \left\{ \max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t)) \right\}.$ 

More about the algorithm latter.

・ 同 ト ・ ヨ ト ・ ヨ ト



Theorem (Malmberg, Ciesielski, Strand; 2019 ???) There is an algorithm,  $O(n^2)$  with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $E_{\infty}(\ell)$ :

 $\max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

More about the algorithm latter.

・ 同 ト ・ ヨ ト ・ ヨ ト …



Theorem (Malmberg, Ciesielski, Strand; 2019 ???) There is an algorithm,  $O(n^2)$  with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $E_{\infty}(\ell)$ :

 $\max \{ \max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t)) \}.$ 

More about the algorithm latter.

イロト イ押ト イヨト イヨトー



Theorem (Malmberg, Ciesielski, Strand; 2019 ???) There is an algorithm,  $O(n^2)$  with respect to  $n = |V \cup \mathcal{E}|$ , returning minimal 2-labeling for any  $E_{\infty}(\ell)$ :  $\max \{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\}.$ 

More about the algorithm latter.



Theorem (Malmberg, Ciesielski, Strand; 2019 ??? ) Optimization problem of the general form of  $E_{\infty}$  energy for more than 2 labels is NP-hard.

Remaining version of Q2:

Q: Under what conditions there exists an efficient (polynomial-time) algorithm for optimization of  $E_{\infty}$  energy for 3 or more labels?

Can be done in FC/Dijkstra setting. Not (NP-hard) in general.

・ロト ・ 理 ト ・ ヨ ト ・



Theorem (Malmberg, Ciesielski, Strand; 2019 ???)

Optimization problem of the general form of  $E_{\infty}$  energy for more than 2 labels is NP-hard.

Remaining version of Q2:

Q: Under what conditions there exists an efficient (polynomial-time) algorithm for optimization of  $E_{\infty}$  energy for 3 or more labels?

Can be done in FC/Dijkstra setting. Not (NP-hard) in general.

・ロ・ ・ 同・ ・ ヨ・ ・ ヨ・



Theorem (Malmberg, Ciesielski, Strand; 2019 ???)

Optimization problem of the general form of  $E_\infty$  energy for more than 2 labels is NP-hard.

Remaining version of Q2:

Q: Under what conditions there exists an efficient (polynomial-time) algorithm for optimization of  $E_{\infty}$  energy for 3 or more labels?

Can be done in FC/Dijkstra setting. Not (NP-hard) in general.

・ロト ・ 理 ト ・ ヨ ト ・



Theorem (Malmberg, Ciesielski, Strand; 2019 ???)

Optimization problem of the general form of  $E_\infty$  energy for more than 2 labels is NP-hard.

Remaining version of Q2:

Q: Under what conditions there exists an efficient (polynomial-time) algorithm for optimization of  $E_{\infty}$  energy for 3 or more labels?

Can be done in FC/Dijkstra setting. Not (NP-hard) in general.

・ロト ・ 理 ト ・ ヨ ト ・



Theorem (Malmberg, Ciesielski, Strand; 2019 ???)

Optimization problem of the general form of  $E_\infty$  energy for more than 2 labels is NP-hard.

Remaining version of Q2:

Q: Under what conditions there exists an efficient (polynomial-time) algorithm for optimization of  $E_{\infty}$  energy for 3 or more labels?

Can be done in FC/Dijkstra setting. Not (NP-hard) in general.

ヘロト ヘアト ヘビト ヘビト



Theorem (Malmberg, Ciesielski, Strand; 2019 ???)

Optimization problem of the general form of  $E_\infty$  energy for more than 2 labels is NP-hard.

Remaining version of Q2:

Q: Under what conditions there exists an efficient (polynomial-time) algorithm for optimization of  $E_{\infty}$  energy for 3 or more labels?

Can be done in FC/Dijkstra setting. Not (NP-hard) in general.

ヘロン 人間 とくほ とくほ とう



Theorem (Malmberg, Ciesielski, Strand; 2019 ???)

Optimization problem of the general form of  $E_\infty$  energy for more than 2 labels is NP-hard.

Remaining version of Q2:

Q: Under what conditions there exists an efficient (polynomial-time) algorithm for optimization of  $E_{\infty}$  energy for 3 or more labels?

Can be done in FC/Dijkstra setting. Not (NP-hard) in general.

ヘロト 人間 ト ヘヨト ヘヨト

Energies  $L_{1^{\&}}$   $E_{\infty}$  The algorithm Strict optimality **Optimal**  $\geq$  3-labeling of  $E_{\infty}(\ell)$  is NP-hard: proof

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

For a graph  $\mathcal{G} = (V, \mathcal{E})$  put:

- $\phi_s(\ell(s)) = 0$  in all cases;
- $\phi_{st}(\ell(s), \ell(t)) = 1$  when  $\ell(s) = \ell(t)$ ;
- $\phi_{st}(\ell(s), \ell(t) = 0 \text{ when } \ell(s) \neq \ell(t).$

Then, the minimal  $E_{\infty}(\ell)$  is 0 if, and only if,  $\ell$  is a coloring of  $\mathcal{G}$ .

But graph *m*-coloring problem for any  $m \ge 3$  is NP-complete! It is not for m = 2.

ヘロト ヘアト ヘビト ヘビト

Conclusions

Energies  $L_1^{\&}$   $E_{\infty}$  The algorithm Strict optimality **Optimal**  $\geq$  3-labeling of  $E_{\infty}(\ell)$  is NP-hard: proof

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

For a graph  $\mathcal{G} = (V, \mathcal{E})$  put:

•  $\phi_s(\ell(s)) = 0$  in all cases;

- $\phi_{st}(\ell(s), \ell(t)) = 1$  when  $\ell(s) = \ell(t);$
- $\phi_{st}(\ell(s), \ell(t) = 0 \text{ when } \ell(s) \neq \ell(t).$

Then, the minimal  $E_{\infty}(\ell)$  is 0 if, and only if,  $\ell$  is a coloring of  $\mathcal{G}$ .

But graph *m*-coloring problem for any  $m \ge 3$  is NP-complete! It is not for m = 2.

イロト イポト イヨト イヨト

Conclusions

= 990

Energies  $L_1^{\&}$   $E_{\infty}$  The algorithm Strict optimality **Optimal**  $\geq$  3-labeling of  $E_{\infty}(\ell)$  is NP-hard: proof

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

For a graph  $\mathcal{G} = (V, \mathcal{E})$  put:

- $\phi_s(\ell(s)) = 0$  in all cases;
- $\phi_{st}(\ell(s), \ell(t)) = 1$  when  $\ell(s) = \ell(t)$ ;
- $\phi_{st}(\ell(s), \ell(t) = 0$  when  $\ell(s) \neq \ell(t)$ .

Then, the minimal  $E_{\infty}(\ell)$  is 0 if, and only if,  $\ell$  is a coloring of  $\mathcal{G}$ .

But graph *m*-coloring problem for any  $m \ge 3$  is NP-complete! It is not for m = 2.

イロト イポト イヨト イヨト

Conclusions

= 990

Energies  $L_1$   $E_{\infty}$  The algorithm Strict optimality **Optimal**  $\geq$  3-labeling of  $E_{\infty}(\ell)$  is NP-hard: proof

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

For a graph  $\mathcal{G} = (V, \mathcal{E})$  put:

- $\phi_s(\ell(s)) = 0$  in all cases;
- $\phi_{st}(\ell(s), \ell(t)) = 1$  when  $\ell(s) = \ell(t)$ ;
- $\phi_{st}(\ell(s), \ell(t) = 0 \text{ when } \ell(s) \neq \ell(t).$

Then, the minimal  $E_{\infty}(\ell)$  is 0 if, and only if,  $\ell$  is a coloring of  $\mathcal{G}$ .

But graph *m*-coloring problem for any  $m \ge 3$  is NP-complete! It is not for m = 2.

イロト イポト イヨト イヨト

Conclusions

= 990

Energies  $L_{1^{\&}}$   $E_{\infty}$  The algorithm Strict optimality **Optimal**  $\geq$  3-labeling of  $E_{\infty}(\ell)$  is NP-hard: proof

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

For a graph  $\mathcal{G} = (V, \mathcal{E})$  put:

- $\phi_s(\ell(s)) = 0$  in all cases;
- $\phi_{st}(\ell(s), \ell(t)) = 1$  when  $\ell(s) = \ell(t)$ ;
- $\phi_{st}(\ell(s), \ell(t) = 0 \text{ when } \ell(s) \neq \ell(t).$

Then, the minimal  $E_{\infty}(\ell)$  is 0 if, and only if,  $\ell$  is a coloring of  $\mathcal{G}$ .

But graph *m*-coloring problem for any  $m \ge 3$  is NP-complete! It is not for m = 2.

ヘロト 人間 とくほ とくほ とう

Conclusions

= 990

Energies  $L_{1^{\&}}$   $E_{\infty}$  The algorithm Strict optimality **Optimal**  $\geq$  3-labeling of  $E_{\infty}(\ell)$  is NP-hard: proof

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

For a graph  $\mathcal{G} = (V, \mathcal{E})$  put:

- $\phi_s(\ell(s)) = 0$  in all cases;
- $\phi_{st}(\ell(s), \ell(t)) = 1$  when  $\ell(s) = \ell(t)$ ;
- $\phi_{st}(\ell(s), \ell(t) = 0 \text{ when } \ell(s) \neq \ell(t).$

Then, the minimal  $E_{\infty}(\ell)$  is 0 if, and only if,  $\ell$  is a coloring of  $\mathcal{G}$ .

But graph *m*-coloring problem for any  $m \ge 3$  is NP-complete! It is not for m = 2.

ヘロト 人間 とくほ とくほ とう

Conclusions

= 990

Energies  $L_{1^{\&}}$   $E_{\infty}$  The algorithm Strict optimality **Optimal**  $\geq$  3-labeling of  $E_{\infty}(\ell)$  is NP-hard: proof

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

For a graph  $\mathcal{G} = (V, \mathcal{E})$  put:

- $\phi_s(\ell(s)) = 0$  in all cases;
- $\phi_{st}(\ell(s), \ell(t)) = 1$  when  $\ell(s) = \ell(t)$ ;
- $\phi_{st}(\ell(s), \ell(t) = 0 \text{ when } \ell(s) \neq \ell(t).$

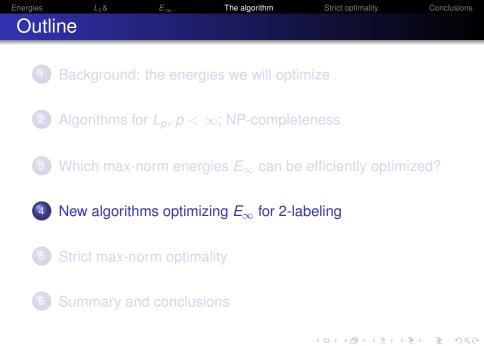
Then, the minimal  $E_{\infty}(\ell)$  is 0 if, and only if,  $\ell$  is a coloring of  $\mathcal{G}$ .

But graph *m*-coloring problem for any  $m \ge 3$  is NP-complete! It is not for m = 2.

ヘロン 人間 とくほ とくほ とう

Conclusions

= 990



 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

Atoms  $\mathcal{A}(\ell)$  of  $\ell$ : input for  $\phi$ .. and  $\phi$ . (to calculate  $E_{\infty}(\ell)$ ), i.e.,  $\mathcal{A}(\ell) := \{\{(s, \ell(s))\} : s \in V\} \cup \{\{(s, \ell(s)), (t, \ell(t))\} : \{s, t\} \in \mathcal{E}\}$ 

Atoms  $\mathcal{A}$  of  $E_{\infty}$ : all such possible atoms, i.e.,

unary: two  $\{(s, 0)\}$  and  $\{(s, 1)\}$  for each  $v \in V$ binary: four  $\{(s, i), (t, j)\}$   $(i, j \in \{0, 1\})$  for each  $\{s, t\} \in \mathcal{E}$ .

• Cost of a unary atom  $\{(s, i)\}$ :  $\phi_s(i)$ 

• Cost of a binary atom  $\{(s, i), (t, j)\}$ :  $\phi_{st}(i, j)$ 

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

Atoms  $\mathcal{A}(\ell)$  of  $\ell$ : input for  $\phi$ .. and  $\phi$ . (to calculate  $E_{\infty}(\ell)$ ), i.e.,  $\mathcal{A}(\ell) := \{\{(s, \ell(s))\} : s \in V\} \cup \{\{(s, \ell(s)), (t, \ell(t))\} : \{s, t\} \in \mathcal{E}\}$ 

Atoms  $\mathcal{A}$  of  $E_{\infty}$ : all such possible atoms, i.e.,

unary: two  $\{(s, 0)\}$  and  $\{(s, 1)\}$  for each  $v \in V$ binary: four  $\{(s, i), (t, j)\}$   $(i, j \in \{0, 1\})$  for each  $\{s, t\} \in \mathcal{E}$ .

• Cost of a unary atom  $\{(s, i)\}$ :  $\phi_s(i)$ 

• Cost of a binary atom  $\{(s, i), (t, j)\}$ :  $\phi_{st}(i, j)$ 

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

Atoms  $\mathcal{A}(\ell)$  of  $\ell$ : input for  $\phi$ .. and  $\phi$ . (to calculate  $E_{\infty}(\ell)$ ), i.e.,  $\mathcal{A}(\ell) := \{\{(s, \ell(s))\} : s \in V\} \cup \{\{(s, \ell(s)), (t, \ell(t))\} : \{s, t\} \in \mathcal{E}\}$ 

Atoms  $\mathcal{A}$  of  $E_{\infty}$ : all such possible atoms, i.e.,

unary: two  $\{(s, 0)\}$  and  $\{(s, 1)\}$  for each  $v \in V$ binary: four  $\{(s, i), (t, j)\}$   $(i, j \in \{0, 1\})$  for each  $\{s, t\} \in \mathcal{E}$ .

• Cost of a unary atom  $\{(s, i)\}$ :  $\phi_s(i)$ 

• Cost of a binary atom  $\{(s, i), (t, j)\}$ :  $\phi_{st}(i, j)$ 

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

Atoms  $\mathcal{A}(\ell)$  of  $\ell$ : input for  $\phi$ .. and  $\phi$ . (to calculate  $E_{\infty}(\ell)$ ), i.e.,  $\mathcal{A}(\ell) := \{\{(s, \ell(s))\} : s \in V\} \cup \{\{(s, \ell(s)), (t, \ell(t))\} : \{s, t\} \in \mathcal{E}\}$ 

#### Atoms $\mathcal{A}$ of $E_{\infty}$ : all such possible atoms, i.e.,

unary: two  $\{(s, 0)\}$  and  $\{(s, 1)\}$  for each  $v \in V$ binary: four  $\{(s, i), (t, j)\}$   $(i, j \in \{0, 1\})$  for each  $\{s, t\} \in \mathcal{E}$ .

• Cost of a unary atom  $\{(s, i)\}$ :  $\phi_s(i)$ 

• Cost of a binary atom  $\{(s, i), (t, j)\}$ :  $\phi_{st}(i, j)$ 

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

Atoms  $\mathcal{A}(\ell)$  of  $\ell$ : input for  $\phi$ .. and  $\phi$ . (to calculate  $E_{\infty}(\ell)$ ), i.e.,  $\mathcal{A}(\ell) := \{\{(s, \ell(s))\} : s \in V\} \cup \{\{(s, \ell(s)), (t, \ell(t))\} : \{s, t\} \in \mathcal{E}\}$ 

Atoms  $\mathcal{A}$  of  $E_{\infty}$ : all such possible atoms, i.e.,

unary: two  $\{(s, 0)\}$  and  $\{(s, 1)\}$  for each  $v \in V$ binary: four  $\{(s, i), (t, j)\}$   $(i, j \in \{0, 1\})$  for each  $\{s, t\} \in \mathcal{E}$ .

• Cost of a unary atom  $\{(s, i)\}$ :  $\phi_s(i)$ 

• Cost of a binary atom  $\{(s, i), (t, j)\}$ :  $\phi_{st}(i, j)$ 

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

Atoms  $\mathcal{A}(\ell)$  of  $\ell$ : input for  $\phi$ .. and  $\phi$ . (to calculate  $E_{\infty}(\ell)$ ), i.e.,  $\mathcal{A}(\ell) := \{\{(s, \ell(s))\} : s \in V\} \cup \{\{(s, \ell(s)), (t, \ell(t))\} : \{s, t\} \in \mathcal{E}\}$ 

Atoms  $\mathcal{A}$  of  $E_{\infty}$ : all such possible atoms, i.e.,

unary: two {(*s*, 0)} and {(*s*, 1)} for each  $v \in V$ binary: four {(*s*, *i*), (*t*, *j*)} (*i*, *j*  $\in$  {0, 1}) for each {*s*, *t*}  $\in \mathcal{E}$ .

• Cost of a unary atom  $\{(s, i)\}$ :  $\phi_s(i)$ 

• Cost of a binary atom  $\{(s, i), (t, j)\}$ :  $\phi_{st}(i, j)$ 

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

Atoms  $\mathcal{A}(\ell)$  of  $\ell$ : input for  $\phi$ .. and  $\phi$ . (to calculate  $E_{\infty}(\ell)$ ), i.e.,  $\mathcal{A}(\ell) := \{\{(s, \ell(s))\} : s \in V\} \cup \{\{(s, \ell(s)), (t, \ell(t))\} : \{s, t\} \in \mathcal{E}\}$ 

Atoms  $\mathcal{A}$  of  $E_{\infty}$ : all such possible atoms, i.e.,

unary: two {(*s*, 0)} and {(*s*, 1)} for each  $v \in V$ binary: four {(*s*, *i*), (*t*, *j*)} (*i*, *j*  $\in$  {0, 1}) for each {*s*, *t*}  $\in \mathcal{E}$ .

• Cost of a unary atom  $\{(s, i)\}$ :  $\phi_s(i)$ 

• Cost of a binary atom  $\{(s, i), (t, j)\}$ :  $\phi_{st}(i, j)$ 

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

Atoms  $\mathcal{A}(\ell)$  of  $\ell$ : input for  $\phi$ .. and  $\phi$ . (to calculate  $E_{\infty}(\ell)$ ), i.e.,  $\mathcal{A}(\ell) := \{\{(s, \ell(s))\} : s \in V\} \cup \{\{(s, \ell(s)), (t, \ell(t))\} : \{s, t\} \in \mathcal{E}\}$ 

Atoms  $\mathcal{A}$  of  $E_{\infty}$ : all such possible atoms, i.e.,

unary: two {(*s*, 0)} and {(*s*, 1)} for each  $v \in V$ binary: four {(*s*, *i*), (*t*, *j*)} (*i*, *j*  $\in$  {0, 1}) for each {*s*, *t*}  $\in \mathcal{E}$ .

- Cost of a unary atom  $\{(s, i)\}$ :  $\phi_s(i)$
- Cost of a binary atom  $\{(s, i), (t, j)\}$ :  $\phi_{st}(i, j)$

 $E_{\infty}(\ell) := \max\left\{\max_{s \in V} \phi_s(\ell(s)), \max_{\{s,t\} \in \mathcal{E}} \phi_{st}(\ell(s), \ell(t))\right\}$ 

Atoms  $\mathcal{A}(\ell)$  of  $\ell$ : input for  $\phi$ .. and  $\phi$ . (to calculate  $E_{\infty}(\ell)$ ), i.e.,  $\mathcal{A}(\ell) := \{\{(s, \ell(s))\} : s \in V\} \cup \{\{(s, \ell(s)), (t, \ell(t))\} : \{s, t\} \in \mathcal{E}\}$ 

Atoms  $\mathcal{A}$  of  $E_{\infty}$ : all such possible atoms, i.e.,

unary: two {(*s*, 0)} and {(*s*, 1)} for each  $v \in V$ binary: four {(*s*, *i*), (*t*, *j*)} (*i*, *j*  $\in$  {0, 1}) for each {*s*, *t*}  $\in \mathcal{E}$ .

- Cost of a unary atom  $\{(s, i)\}$ :  $\phi_s(i)$
- Cost of a binary atom  $\{(s, i), (t, j)\}$ :  $\phi_{st}(i, j)$

Energies  $L_1$   $E_\infty$  The algorithm Strict optimality Conclusions  $\infty$ -sub algorithm

List all atoms in a list S in a decreasing cost so that if atoms A<sub>0</sub> and A<sub>1</sub> have the same cost and A<sub>1</sub> = {(s, i), (t, i)}, then A<sub>1</sub> proceeds A<sub>0</sub>.

- While S is non-empty do
  - Remove the first atom A from S
  - If A is the last atom for its vertex/edge, insert it to list L
  - Consecutively remove from S all atoms that are locally inconsistent with current S ∪ L
- 3 Return labeling  $\ell = \bigcup L$

The locally inconsistency loop is natural.

The trick is to show that the algorithm works property for all  $\infty$ -submodular energies.

ヘロト ヘワト ヘビト ヘビト

Energies  $L_1$   $E_\infty$  The algorithm Strict optimality Conclusions  $\infty$ -sub algorithm

### List all atoms in a list S in a decreasing cost

- so that if atoms  $A_0$  and  $A_1$  have the same cost and  $A_1 = \{(s, i), (t, i)\}$ , then  $A_1$  proceeds  $A_0$ .
- While S is non-empty do
  - Remove the first atom A from S
  - If A is the last atom for its vertex/edge, insert it to list L
  - Consecutively remove from S all atoms that are locally inconsistent with current S ∪ L
- 3 Return labeling  $\ell = \bigcup L$

The locally inconsistency loop is natural.

The trick is to show that the algorithm works property for all  $\infty$ -submodular energies.

ヘロト ヘワト ヘビト ヘビト



- List all atoms in a list S in a decreasing cost so that if atoms A<sub>0</sub> and A<sub>1</sub> have the same cost and A<sub>1</sub> = {(s, i), (t, i)}, then A<sub>1</sub> proceeds A<sub>0</sub>.
- While *S* is non-empty do
  - Remove the first atom A from S
  - If A is the last atom for its vertex/edge, insert it to list L
  - Consecutively remove from S all atoms that are locally inconsistent with current S ∪ L
- 3 Return labeling  $\ell = \bigcup L$

The trick is to show that the algorithm works property for all  $\infty$ -submodular energies.



- List all atoms in a list S in a decreasing cost so that if atoms A<sub>0</sub> and A<sub>1</sub> have the same cost and A<sub>1</sub> = {(s, i), (t, i)}, then A<sub>1</sub> proceeds A<sub>0</sub>.
- While S is non-empty do
  - Remove the first atom A from S
  - If A is the last atom for its vertex/edge, insert it to list L
  - Consecutively remove from S all atoms that are locally inconsistent with current S ∪ L
- 3 Return labeling  $\ell = \bigcup L$

The trick is to show that the algorithm works property for all  $\infty$ -submodular energies.



- List all atoms in a list S in a decreasing cost so that if atoms A<sub>0</sub> and A<sub>1</sub> have the same cost and A<sub>1</sub> = {(s, i), (t, i)}, then A<sub>1</sub> proceeds A<sub>0</sub>.
- While S is non-empty do
  - Remove the first atom A from S
  - If A is the last atom for its vertex/edge, insert it to list L
  - Consecutively remove from S all atoms that are locally inconsistent with current S ∪ L
- 3 Return labeling  $\ell = \bigcup L$

The trick is to show that the algorithm works property for all  $\infty$ -submodular energies.



- List all atoms in a list S in a decreasing cost so that if atoms A<sub>0</sub> and A<sub>1</sub> have the same cost and A<sub>1</sub> = {(s, i), (t, i)}, then A<sub>1</sub> proceeds A<sub>0</sub>.
- While S is non-empty do
  - Remove the first atom A from S
  - If A is the last atom for its vertex/edge, insert it to list L
  - Consecutively remove from S all atoms that are locally inconsistent with current S ∪ L

3 Return labeling  $\ell = \bigcup L$ 

The locally inconsistency loop is natural.

The trick is to show that the algorithm works property for all  $\infty$ -submodular energies.



- List all atoms in a list S in a decreasing cost so that if atoms A<sub>0</sub> and A<sub>1</sub> have the same cost and A<sub>1</sub> = {(s, i), (t, i)}, then A<sub>1</sub> proceeds A<sub>0</sub>.
- While S is non-empty do
  - Remove the first atom A from S
  - If A is the last atom for its vertex/edge, insert it to list L
  - Consecutively remove from S all atoms that are locally inconsistent with current S ∪ L
- 3 Return labeling  $\ell = \bigcup L$

The trick is to show that the algorithm works property for all  $\infty$ -submodular energies.

ヘロト ヘ戸ト ヘヨト ヘヨト



- List all atoms in a list S in a decreasing cost so that if atoms A<sub>0</sub> and A<sub>1</sub> have the same cost and A<sub>1</sub> = {(s, i), (t, i)}, then A<sub>1</sub> proceeds A<sub>0</sub>.
- While S is non-empty do
  - Remove the first atom A from S
  - If A is the last atom for its vertex/edge, insert it to list L
  - Consecutively remove from S all atoms that are locally inconsistent with current S ∪ L
- 3 Return labeling  $\ell = \bigcup L$

The trick is to show that the algorithm works property for all  $\infty$ -submodular energies.

ヘロト ヘ戸ト ヘヨト ヘヨト



- List all atoms in a list S in a decreasing cost so that if atoms A<sub>0</sub> and A<sub>1</sub> have the same cost and A<sub>1</sub> = {(s, i), (t, i)}, then A<sub>1</sub> proceeds A<sub>0</sub>.
- While S is non-empty do
  - Remove the first atom A from S
  - If A is the last atom for its vertex/edge, insert it to list L
  - Consecutively remove from S all atoms that are locally inconsistent with current S ∪ L
- 3 Return labeling  $\ell = \bigcup L$

The trick is to show that the algorithm works property for all  $\infty\mathchar`-submodular energies.$ 

くロト (過) (目) (日)

For atoms  $A = \{(s, i)\}$  and  $A' = \{(s, i), (t, j)\}$  define formulas

The algorithm

$$\psi_{\mathcal{A}}(\boldsymbol{s}) := "\boldsymbol{s} 
eq i" = egin{cases} 
eg \boldsymbol{s} & ext{if } i = 1, \\ \boldsymbol{s} & ext{if } i = 0 \end{cases}$$

 $\psi_{A}(s,t) := "(s \neq i) \lor (t \neq j)" = \psi_{\{(s,i)\}}(s) \lor \psi_{\{(t,j)\}}(t).$ For a set  $\mathcal{A}' = \{A_1, A_2, \dots, A_k\}$  of atoms the formula  $\psi_{\mathcal{A}'} := \psi_{A_1} \land \dots \land \psi_{A_k}.$  is in 2-conjunctive normal form.

Theorem

Energies

A set  $A_1 \subseteq A$  of atoms is consistent if, and only if, the 2-satisfiability problem for a formula  $\psi_{A_1^c}$  has a positive solution.

So, consistency of  $A_1 \subseteq A$  can be decided by in linear time. (Aspvall et al. algorithm.)

Strict optimality

For atoms  $A = \{(s, i)\}$  and  $A' = \{(s, i), (t, j)\}$  define formulas

The algorithm

$$\psi_{\mathcal{A}}(\boldsymbol{s}) := \text{``} \boldsymbol{s} 
eq i extsf{``} = egin{cases} 
eg \boldsymbol{s} & extsf{if} \ i = 1, \ \boldsymbol{s} & extsf{if} \ i = 0 \end{cases}$$

$$\psi_{\mathcal{A}}(s,t) := "(s \neq i) \lor (t \neq j)" = \psi_{\{(s,i)\}}(s) \lor \psi_{\{(t,j)\}}(t).$$

For a set  $\mathcal{A}' = \{A_1, A_2, \dots, A_k\}$  of atoms the formula  $\psi_{\mathcal{A}'} := \psi_{\mathcal{A}_1} \land \dots \land \psi_{\mathcal{A}_k}$  is in *2-conjunctive normal form* 

Theorem

Energies

A set  $A_1 \subseteq A$  of atoms is consistent if, and only if, the 2-satisfiability problem for a formula  $\psi_{A_1^c}$  has a positive solution.

So, consistency of  $A_1 \subseteq A$  can be decided by in linear time. (Aspvall et al. algorithm.)

Strict optimality

For atoms  $A = \{(s, i)\}$  and  $A' = \{(s, i), (t, j)\}$  define formulas

The algorithm

$$\psi_{\mathcal{A}}(\boldsymbol{s}) := \text{``} \boldsymbol{s} 
eq i extsf{``} = egin{cases} 
eg \boldsymbol{s} & extsf{if} \ i = 1, \ \boldsymbol{s} & extsf{if} \ i = 0 \end{cases}$$

$$\psi_{\mathcal{A}}(\boldsymbol{s},t) := "(\boldsymbol{s} \neq i) \lor (t \neq j)" = \psi_{\{(\boldsymbol{s},i)\}}(\boldsymbol{s}) \lor \psi_{\{(t,j)\}}(t).$$

For a set  $\mathcal{A}' = \{A_1, A_2, \dots, A_k\}$  of atoms the formula  $\psi_{\mathcal{A}'} := \psi_{A_1} \wedge \dots \wedge \psi_{A_k}$  is in 2-conjunctive normal form.

Theorem

Energies

A set  $A_1 \subseteq A$  of atoms is consistent if, and only if, the 2-satisfiability problem for a formula  $\psi_{A_1^c}$  has a positive solution.

So, consistency of  $A_1 \subseteq A$  can be decided by in linear time. (Aspvall et al. algorithm.)

Strict optimality

For atoms  $A = \{(s, i)\}$  and  $A' = \{(s, i), (t, j)\}$  define formulas

The algorithm

$$\psi_{\mathcal{A}}(\boldsymbol{s}) := \text{``} \boldsymbol{s} 
eq i extsf{``} = egin{cases} 
eg \boldsymbol{s} & extsf{if} \ i = 1, \ \boldsymbol{s} & extsf{if} \ i = 0 \end{cases}$$

$$\psi_{\mathcal{A}}(\boldsymbol{s},t) := "(\boldsymbol{s} \neq i) \lor (t \neq j)" = \psi_{\{(\boldsymbol{s},i)\}}(\boldsymbol{s}) \lor \psi_{\{(t,j)\}}(t).$$

For a set  $\mathcal{A}' = \{A_1, A_2, \dots, A_k\}$  of atoms the formula  $\psi_{\mathcal{A}'} := \psi_{A_1} \wedge \dots \wedge \psi_{A_k}$  is in 2-conjunctive normal form.

#### Theorem

Energies

A set  $A_1 \subseteq A$  of atoms is consistent if, and only if, the 2-satisfiability problem for a formula  $\psi_{A_1^c}$  has a positive solution.

So, consistency of  $A_1 \subseteq A$  can be decided by in linear time. (Aspvall et al. algorithm.)

Strict optimality

For atoms  $A = \{(s, i)\}$  and  $A' = \{(s, i), (t, j)\}$  define formulas

The algorithm

$$\psi_{\mathcal{A}}(\boldsymbol{s}) := \text{``} \boldsymbol{s} 
eq i extsf{``} = egin{cases} 
eg \boldsymbol{s} & extsf{if} \ i = 1, \ \boldsymbol{s} & extsf{if} \ i = 0 \end{cases}$$

$$\psi_{\mathcal{A}}(\boldsymbol{s},t) := "(\boldsymbol{s} \neq i) \lor (t \neq j)" = \psi_{\{(\boldsymbol{s},i)\}}(\boldsymbol{s}) \lor \psi_{\{(t,j)\}}(t).$$

For a set  $\mathcal{A}' = \{A_1, A_2, \dots, A_k\}$  of atoms the formula  $\psi_{\mathcal{A}'} := \psi_{A_1} \wedge \dots \wedge \psi_{A_k}$  is in 2-conjunctive normal form.

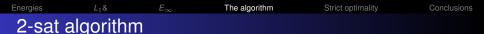
#### Theorem

Energies

A set  $A_1 \subseteq A$  of atoms is consistent if, and only if, the 2-satisfiability problem for a formula  $\psi_{A_1^c}$  has a positive solution.

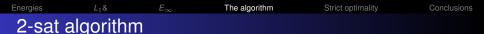
So, consistency of  $A_1 \subseteq A$  can be decided by in linear time. (Aspvall et al. algorithm.)

Strict optimality



- List all atoms in a list S in a decreasing cost
- While S is non-empty do
  - Remove the first atom A from S
  - If  $S \cup L$  is not consistent, insert A to L
- 3 Return labeling  $\ell = \bigcup L$

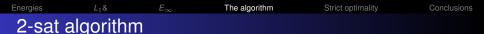
The  $S \cup L$  is not consistent clause is decided by Aspvall et al. algorithm.



### List all atoms in a list S in a decreasing cost

- While S is non-empty do
  - Remove the first atom A from S
  - If  $S \cup L$  is not consistent, insert A to L
- 3 Return labeling  $\ell = \bigcup L$

The  $S \cup L$  is not consistent clause is decided by Aspvall et al. algorithm.

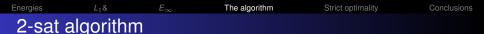


List all atoms in a list S in a decreasing cost
While S is non-empty do

Remove the first atom A from S
If S ∪ L is not consistent, insert A to L

Return labeling l = U L

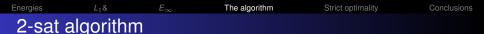
The  $S \cup L$  is not consistent clause is decided by Aspvall et al. algorithm.



## • List all atoms in a list S in a decreasing cost

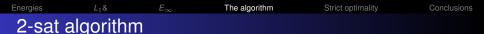
- While S is non-empty do
  - Remove the first atom A from S
  - If  $S \cup L$  is not consistent, insert A to L
- 3 Return labeling  $\ell = \bigcup L$

The  $S \cup L$  is not consistent clause is decided by Aspvall et al. algorithm.



- List all atoms in a list S in a decreasing cost
- While S is non-empty do
  - Remove the first atom A from S
  - If  $S \cup L$  is not consistent, insert A to L
- 3 Return labeling  $\ell = \bigcup L$

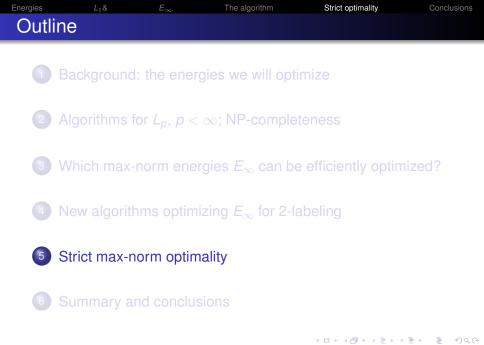
The  $S \cup L$  is not consistent clause is decided by Aspvall et al. algorithm.



- List all atoms in a list S in a decreasing cost
- While S is non-empty do
  - Remove the first atom A from S
  - If  $S \cup L$  is not consistent, insert A to L
- 3 Return labeling  $\ell = \bigcup L$

The  $S \cup L$  is not consistent clause is decided by Aspvall et al. algorithm.

・ 同 ト ・ ヨ ト ・ ヨ ト



# Energies $L_1$ & $E_{\infty}$ The algorithmStrict optimalityStrict optimality via lexicographical order

Max-norm identifies  $\ell_1$  and  $\ell_2$  when  $E_{\infty}(\ell_1) = E_{\infty}(\ell_2)$ .

Lexicographical order  $\leq$  is a sharper distinguishing tool.

For labeling  $\ell$ , let  $\vec{\ell} = \langle \ell_1, \dots, \ell_n \rangle = \langle \Phi(A_1), \dots, \Phi(A_n) \rangle$ non-increasing for an enumeration  $\mathcal{A}(\ell) = \{A_1, \dots, A_n\}$ .

 $\ell \prec \ell'$  iff  $\ell_i < \ell'_i$ , where  $i := \min\{k \colon \ell_k < \ell'_k\}$ .

 $\ell$  is strictly optimal when it is maximal w.r.t.  $\leq$ .

Strictly optimal implies max-norm optimal, but not converse.

#### Q: Can we efficiently find also strict optimizers?

くロト (過) (目) (日)

# Energies $L_1$ & $E_{\infty}$ The algorithmStrict optimalityStrict optimality via lexicographical order

Max-norm identifies  $\ell_1$  and  $\ell_2$  when  $E_{\infty}(\ell_1) = E_{\infty}(\ell_2)$ .

#### Lexicographical order $\leq$ is a sharper distinguishing tool.

For labeling  $\ell$ , let  $\vec{\ell} = \langle \ell_1, \dots, \ell_n \rangle = \langle \Phi(A_1), \dots, \Phi(A_n) \rangle$ non-increasing for an enumeration  $\mathcal{A}(\ell) = \{A_1, \dots, A_n\}$ .

 $\ell \prec \ell'$  iff  $\ell_i < \ell'_i$ , where  $i := \min\{k \colon \ell_k < \ell'_k\}$ .

 $\ell$  is strictly optimal when it is maximal w.r.t.  $\leq$ .

Strictly optimal implies max-norm optimal, but not converse.

#### Q: Can we efficiently find also strict optimizers?

ヘロト 人間 ト ヘヨト ヘヨト

Max-norm identifies  $\ell_1$  and  $\ell_2$  when  $E_{\infty}(\ell_1) = E_{\infty}(\ell_2)$ .

Lexicographical order  $\leq$  is a sharper distinguishing tool.

For labeling  $\ell$ , let  $\vec{\ell} = \langle \ell_1, \dots, \ell_n \rangle = \langle \Phi(A_1), \dots, \Phi(A_n) \rangle$ non-increasing for an enumeration  $\mathcal{A}(\ell) = \{A_1, \dots, A_n\}$ .

 $\ell \prec \ell'$  iff  $\ell_i < \ell'_i$ , where  $i := \min\{k \colon \ell_k < \ell'_k\}$ .

 $\ell$  is strictly optimal when it is maximal w.r.t.  $\leq$ .

Strictly optimal implies max-norm optimal, but not converse.

Q: Can we efficiently find also strict optimizers?

ヘロト 人間 ト ヘヨト ヘヨト

Max-norm identifies  $\ell_1$  and  $\ell_2$  when  $E_{\infty}(\ell_1) = E_{\infty}(\ell_2)$ .

Lexicographical order  $\leq$  is a sharper distinguishing tool.

For labeling  $\ell$ , let  $\vec{\ell} = \langle \ell_1, \dots, \ell_n \rangle = \langle \Phi(A_1), \dots, \Phi(A_n) \rangle$ non-increasing for an enumeration  $\mathcal{A}(\ell) = \{A_1, \dots, A_n\}$ .

 $\ell \prec \ell'$  iff  $\ell_i < \ell'_i$ , where  $i := \min\{k \colon \ell_k < \ell'_k\}$ .

 $\ell$  is strictly optimal when it is maximal w.r.t.  $\leq$ .

Strictly optimal implies max-norm optimal, but not converse.

Q: Can we efficiently find also strict optimizers?

ヘロト ヘアト ヘビト ヘビト

Max-norm identifies  $\ell_1$  and  $\ell_2$  when  $E_{\infty}(\ell_1) = E_{\infty}(\ell_2)$ .

Lexicographical order  $\leq$  is a sharper distinguishing tool.

For labeling  $\ell$ , let  $\vec{\ell} = \langle \ell_1, \dots, \ell_n \rangle = \langle \Phi(A_1), \dots, \Phi(A_n) \rangle$ non-increasing for an enumeration  $\mathcal{A}(\ell) = \{A_1, \dots, A_n\}$ .

 $\ell \prec \ell'$  iff  $\ell_i < \ell'_i$ , where  $i := \min\{k \colon \ell_k < \ell'_k\}$ .

 $\ell$  is strictly optimal when it is maximal w.r.t.  $\leq$ .

Strictly optimal implies max-norm optimal, but not converse.

Q: Can we efficiently find also strict optimizers?

・ロト ・ 理 ト ・ ヨ ト ・

Max-norm identifies  $\ell_1$  and  $\ell_2$  when  $E_{\infty}(\ell_1) = E_{\infty}(\ell_2)$ .

Lexicographical order  $\leq$  is a sharper distinguishing tool.

For labeling  $\ell$ , let  $\vec{\ell} = \langle \ell_1, \dots, \ell_n \rangle = \langle \Phi(A_1), \dots, \Phi(A_n) \rangle$ non-increasing for an enumeration  $\mathcal{A}(\ell) = \{A_1, \dots, A_n\}$ .

 $\ell \prec \ell'$  iff  $\ell_i < \ell'_i$ , where  $i := \min\{k \colon \ell_k < \ell'_k\}$ .

 $\ell$  is strictly optimal when it is maximal w.r.t.  $\preceq.$ 

Strictly optimal implies max-norm optimal, but not converse.

Q: Can we efficiently find also strict optimizers?

・ロト ・ 理 ト ・ ヨ ト ・

Max-norm identifies  $\ell_1$  and  $\ell_2$  when  $E_{\infty}(\ell_1) = E_{\infty}(\ell_2)$ .

Lexicographical order  $\leq$  is a sharper distinguishing tool.

For labeling  $\ell$ , let  $\vec{\ell} = \langle \ell_1, \dots, \ell_n \rangle = \langle \Phi(A_1), \dots, \Phi(A_n) \rangle$ non-increasing for an enumeration  $\mathcal{A}(\ell) = \{A_1, \dots, A_n\}$ .

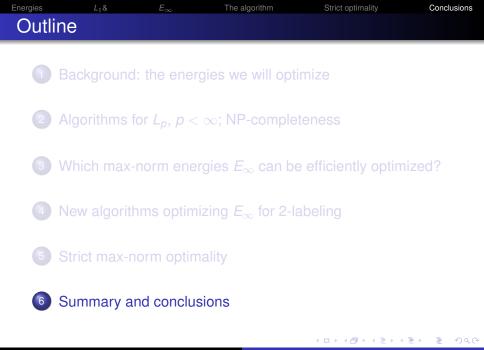
 $\ell \prec \ell'$  iff  $\ell_i < \ell'_i$ , where  $i := \min\{k \colon \ell_k < \ell'_k\}$ .

 $\ell$  is strictly optimal when it is maximal w.r.t.  $\leq$ .

Strictly optimal implies max-norm optimal, but not converse.

#### Q: Can we efficiently find also strict optimizers?

・ロ・ ・ 同・ ・ ヨ・



L1&

The alg

Strict optimalit

Conclusions

### Summary (including new results)

	2 labels	$\geq$ 3 labels
general case strict optimization	NP-hard problem	NP-hard problem
$\infty$ -submodular strict optimization		NP-hard problem
	<b>2-sat algorithm</b> $O(n^2)$	NP-hard problem
		NP-hard problem
		NP-hard problem

< 同.

프 🖌 🛪 프 🕨

 $L_1 \&$ 

The algorit

Strict optimalit

Conclusions

### Summary (including new results)

	2 labels	$\geq$ 3 labels
general case strict optimization	NP-hard problem	NP-hard problem
$\infty$ -submodular strict optimization	$\max - \text{flow/min-cut} \\ O(n^2 \ln n) \le \cdot \le O(n^3)$	NP-hard problem
unique weights strict optimization	<b>2-sat algorithm</b> $O(n^2)$	NP-hard problem
		NP-hard problem
		NP-hard problem

< 同.

프 🖌 🛪 프 🕨

 $L_1 \&$ 

The algorit

Strict optimalit

Conclusions

### Summary (including new results)

	2 labels	$\geq$ 3 labels
general case strict optimization	NP-hard problem	NP-hard problem
$\infty$ -submodular strict optimization	$\begin{array}{l} \text{max-flow/min-cut} \\ O(n^2 \ln n) \leq \cdot \leq O(n^3) \end{array}$	NP-hard problem
unique weights strict optimization	<b>2-sat algorithm</b> $O(n^2)$	NP-hard problem
general case	2-sat algorithm; $O(n^2)$	NP-hard problem
$\infty$ -submodular		NP-hard problem

< 同.

프 🖌 🛪 프 🕨

 $L_1 \&$ 

The algorit

Strict optimalit

Conclusions

### Summary (including new results)

	2 labels	$\geq$ 3 labels
general case strict optimization	NP-hard problem	NP-hard problem
$\infty$ -submodular strict optimization	$\begin{array}{l} \max \text{-flow/min-cut} \\ O(n^2 \ln n) \leq \cdot \leq O(n^3) \end{array}$	NP-hard problem
unique weights strict optimization	<b>2-sat algorithm</b> $O(n^2)$	NP-hard problem
general case	2-sat algorithm; $O(n^2)$	NP-hard problem
$\infty$ -submodular	$\infty$ -sub algorithm $O(n) \leq \cdot \leq O(n \ln n)$	NP-hard problem

< 同.

프 🖌 🛪 프 🕨

 $L_1 \&$ 

The algorit

Strict optimalit

Conclusions

### Summary (including new results)

	2 labels	$\geq$ 3 labels
general case strict optimization	NP-hard problem	NP-hard problem
$\infty$ -submodular strict optimization	$\begin{array}{l} \text{max-flow/min-cut} \\ O(n^2 \ln n) \leq \cdot \leq O(n^3) \end{array}$	NP-hard problem
unique weights strict optimization	<b>2-sat algorithm</b> $O(n^2)$	NP-hard problem
general case	2-sat algorithm; $O(n^2)$	NP-hard problem
$\infty$ -submodular	$\infty$ -sub algorithm $O(n) \le \cdot \le O(n \ln n)$	NP-hard problem
$\phi_{\boldsymbol{s}}(i) = \phi_{\boldsymbol{s}t}(i,i) = \boldsymbol{0};$	Dijkstra algorithm	Dijkstra algorithm
$\phi_{st}(i,j) = \phi_{st}(j,i) \ge 0$		

< 同.

프 🖌 🛪 프 🕨

 $L_1 \&$ 

The algori

Strict optimalit

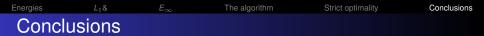
Conclusions

### Summary (including new results)

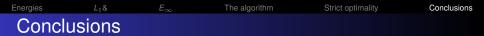
	2 labels	$\geq$ 3 labels
general case strict optimization	NP-hard problem	NP-hard problem
$\infty$ -submodular strict optimization	$\max - \text{flow/min-cut} \\ O(n^2 \ln n) \le \cdot \le O(n^3)$	NP-hard problem
unique weights strict optimization	<b>2-sat algorithm</b> $O(n^2)$	NP-hard problem
general case	2-sat algorithm; $O(n^2)$	NP-hard problem
$\infty$ -submodular	$\infty$ -sub algorithm $O(n) \le \cdot \le O(n \ln n)$	NP-hard problem
$\phi_{s}(i) = \phi_{st}(i,i) = 0;$	Dijkstra algorithm	Dijkstra algorithm
$\phi_{st}(i,j) = \phi_{st}(j,i) \ge 0$	$O(n) \leq \cdot \leq O(n \ln n)$	$O(n) \leq \cdot \leq O(n \ln n)$

< 同.

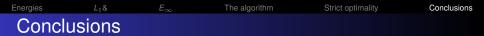
프 🖌 🛪 프 🕨



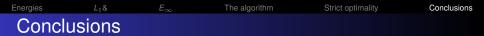
- Optimization problems, specifically pixel labeling problems, are frequently occurring in image processing applications.
- We are specifically interested in problems where the objective function is given by the max-norm of the local errors.
- For many such problems, globally optimal solutions can be found very efficiently, in quasi linear or quadratic time.
- Some max-norm for  $\geq$  3-labeling are NP-hard.



- Optimization problems, specifically pixel labeling problems, are frequently occurring in image processing applications.
- We are specifically interested in problems where the objective function is given by the max-norm of the local errors.
- For many such problems, globally optimal solutions can be found very efficiently, in quasi linear or quadratic time.
- Some max-norm for  $\geq$  3-labeling are NP-hard.



- Optimization problems, specifically pixel labeling problems, are frequently occurring in image processing applications.
- We are specifically interested in problems where the objective function is given by the max-norm of the local errors.
- For many such problems, globally optimal solutions can be found very efficiently, in quasi linear or quadratic time.
- Some max-norm for  $\geq$  3-labeling are NP-hard.



- Optimization problems, specifically pixel labeling problems, are frequently occurring in image processing applications.
- We are specifically interested in problems where the objective function is given by the max-norm of the local errors.
- For many such problems, globally optimal solutions can be found very efficiently, in quasi linear or quadratic time.
- Some max-norm for  $\geq$  3-labeling are NP-hard.

ies	L <sub>1</sub> &
-----	------------------

### Thank you for your attention!

K. Chris Ciesielski Optimization of Max-Norm Objective Functions 20 of 20

< 🗇 🕨

★ Ξ → ★ Ξ →

ъ