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 Campinas, Brazil, May 30, 2019

Outline

1. Background: the energies we will optimize
2. Algorithms for L_p, $p < \infty$; NP-completeness
3. Which max-norm energies E_∞ can be efficiently optimized?
4. New algorithms optimizing E_∞ for 2-labeling
5. Strict max-norm optimality
6. Summary and conclusions

Outline

1. Background: the energies we will optimize
2. Algorithms for L_p, $p < \infty$; NP-completeness
3. Which max-norm energies E_∞ can be efficiently optimized?
4. New algorithms optimizing E_∞ for 2-labeling
5. Strict max-norm optimality
6. Summary and conclusions

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 $\mathcal{G} = (V, \mathcal{E}, f)$, with vertices V being image voxels, edges $\mathcal{E}$ being pairs $\{s, t\}$ of adjacent voxels, and $f(s)$ image intensity at s. Its labeling is a map $\ell \colon V \to \{0, \ldots, m-1\}$, with $m \geq 2$.

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 $\mathcal{G} = (V, \mathcal{E}, f)$, with vertices V being image voxels, edges $\mathcal{E}$ being pairs $\{s, t\}$ of adjacent voxels, and $f(s)$ image intensity at s. Its labeling is a map $\ell \colon V \to \{0, \ldots, m-1\}$, with $m \geq 2$.

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 $\mathcal{G} = (V, \mathcal{E}, f)$, with vertices V being image voxels, edges $\mathcal{E}$ being pairs $\{s, t\}$ of adjacent voxels, and $f(s)$ image intensity at s. Its labeling is a map $\ell \colon V \to \{0, \ldots, m-1\}$, with $m \geq 2$.

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 $\mathcal{G} = (V, \mathcal{E}, f)$, with vertices V being image voxels, edges $\mathcal{E}$ being pairs $\{s, t\}$ of adjacent voxels, and $f(s)$ image intensity at s. Its labeling is a map $\ell \colon V \to \{0, \ldots, m-1\}$, with $m \geq 2$.

L_p energies: the case of L_1

With any image n-labeling ℓ 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;
- pairwise terms $\phi_\ell(s,t) = \phi_{s,t}(\ell(s), \ell(t))$, depending on $\{s,t\} \in \mathcal{E}$ and their labeling. They reflect desirability of smoothness/regularity of labeling.
  All $\phi_{s,t}(0,0)$, $\phi_{s,t}(0,1)$, $\phi_{s,t}(1,0)$, $\phi_{s,t}(1,1)$ 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)

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

L_p energies: the case of L_1

With any image n-labeling ℓ 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;
- pairwise terms $\phi_\ell(s,t) = \phi_{s,t}(\ell(s), \ell(t))$, depending on $\{s,t\} \in \mathcal{E}$ and their labeling. They reflect desirability of smoothness/regularity of labeling.
  All $\phi_{s,t}(0,0)$, $\phi_{s,t}(0,1)$, $\phi_{s,t}(1,0)$, $\phi_{s,t}(1,1)$ 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)

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

L_p energies: the case of L_1

With any image n-labeling ℓ 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;
- pairwise terms $\phi_\ell(s,t) = \phi_{s,t}(\ell(s), \ell(t))$, depending on $\{s,t\} \in \mathcal{E}$ and their labeling. They reflect desirability of smoothness/regularity of labeling.
  All $\phi_{s,t}(0,0)$, $\phi_{s,t}(0,1)$, $\phi_{s,t}(1,0)$, $\phi_{s,t}(1,1)$ 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)

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

L_p energies: the case of L_1

With any image n-labeling ℓ 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;
- pairwise terms $\phi_\ell(s,t) = \phi_{s,t}(\ell(s), \ell(t))$, depending on $\{s,t\} \in \mathcal{E}$ and their labeling. They reflect desirability of smoothness/regularity of labeling.
  All $\phi_{s,t}(0,0)$, $\phi_{s,t}(0,1)$, $\phi_{s,t}(1,0)$, $\phi_{s,t}(1,1)$ 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)

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

L_p energies: the case of L_1

With any image n-labeling ℓ 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;
- pairwise terms $\phi_\ell(s,t) = \phi_{s,t}(\ell(s), \ell(t))$, depending on $\{s,t\} \in \mathcal{E}$ and their labeling. They reflect desirability of smoothness/regularity of labeling.
  All $\phi_{s,t}(0,0)$, $\phi_{s,t}(0,1)$, $\phi_{s,t}(1,0)$, $\phi_{s,t}(1,1)$ 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)

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

L_p energies: the case of L_1

With any image n-labeling ℓ 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;
- pairwise terms $\phi_\ell(s,t) = \phi_{s,t}(\ell(s), \ell(t))$, depending on $\{s,t\} \in \mathcal{E}$ and their labeling. They reflect desirability of smoothness/regularity of labeling.
  All $\phi_{s,t}(0,0)$, $\phi_{s,t}(0,1)$, $\phi_{s,t}(1,0)$, $\phi_{s,t}(1,1)$ 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)

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

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)

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

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

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)

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

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

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)

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

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

What is the effect of p?

- 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.

What is the effect of p?

- 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.

What is the effect of p?

- 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.

Outline

$p = 1$: Graph Cut segmentation via min-cut/max-flow

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

- $\phi_s(\ell(s)) = 0$ 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 $n := |V \cup \mathcal{E}|$.

Optimization is NP-hard **for $\geq$ 3-labeling**.

$p = 1$: Graph Cut segmentation via min-cut/max-flow

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

- $\phi_s(\ell(s)) = 0$ 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 $n := |V \cup \mathcal{E}|$.

Optimization is NP-hard **for $\geq$ 3-labeling.**

$p = 1$: Graph Cut segmentation via min-cut/max-flow

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

- $\phi_s(\ell(s)) = 0$ 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 $n := |V \cup \mathcal{E}|$.

Optimization is NP-hard **for $\geq$ 3-labeling**.

$p = 1$: Graph Cut segmentation via min-cut/max-flow

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

- $\phi_s(\ell(s)) = 0$ 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 $n := |V \cup \mathcal{E}|$.

Optimization is NP-hard **for $\geq$ 3-labeling**.

$p = 1$: Graph Cut segmentation via min-cut/max-flow

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

- $\phi_s(\ell(s)) = 0$ 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 $n := |V \cup \mathcal{E}|$.

Optimization is NP-hard **for $\geq$ 3-labeling.**

$p = 1$: Graph Cut segmentation via min-cut/max-flow

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

- $\phi_s(\ell(s)) = 0$ 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 $n := |V \cup \mathcal{E}|$.

Optimization is NP-hard **for** $\geq$ **3-labeling.**

$p = 1$: Graph Cut segmentation via min-cut/max-flow

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

- $\phi_s(\ell(s)) = 0$ 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 $n := |V \cup \mathcal{E}|$.

Optimization is NP-hard **for** $\geq$ **3-labeling.**

2-labeling for general $E_1(\ell)$-optimization

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

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

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

Theorem (Kolmogorov & Zabih 2004)

- *If E_1 is submodular, then min-cut/max-flow algorithm returns optimized labeling.*
- *If E_1 is* **NOT** *submodular, then minimizing E_1 is NP-hard.*

2-labeling for general $E_1(\ell)$-optimization

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

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_1 is submodular, then min-cut/max-flow algorithm returns optimized labeling.*
- *If E_1 is **NOT** submodular, then minimizing E_1 is NP-hard.*

2-labeling for general $E_1(\ell)$-optimization

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

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

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

Theorem (Kolmogorov & Zabih 2004)

- *If E_1 is submodular, then min-cut/max-flow algorithm returns optimized labeling.*
- *If E_1 is* **NOT** *submodular, then minimizing E_1 is NP-hard.*

2-labeling for general $E_1(\ell)$-optimization

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

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_1 is submodular, then min-cut/max-flow algorithm returns optimized labeling.*
- *If E_1 is* **NOT** *submodular, then minimizing E_1 is NP-hard.*

$E_p(\ell)$ with $1 \leq p < \infty$ is as $E_1(\ell)$

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

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

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

Corollary (Obvious, Malmberg & Strand, IWCIA 2018)

- *If E_p 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.*

$E_p(\ell)$ with $1 \leq p < \infty$ is as $E_1(\ell)$

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

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

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

Corollary (Obvious, Malmberg & Strand, IWCIA 2018)

- *If E_p 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.*

$E_p(\ell)$ with $1 \le p < \infty$ is as $E_1(\ell)$

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

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_p 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.*

$E_p(\ell)$ with $1 \le p < \infty$ is as $E_1(\ell)$

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

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_p 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.*

$E_p(\ell)$ with $1 \leq p < \infty$ vs $E_\infty(\ell)$

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

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

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

Theorem (Malmberg & Strand, IWCIA 2018)

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

Actually, ϕ is ∞-submodular iff there is an N so that ϕ is p-submodular for all $p \in (N, \infty)$.

$E_p(\ell)$ with $1 \leq p < \infty$ vs $E_\infty(\ell)$

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

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

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

Theorem (Malmberg & Strand, IWCIA 2018)

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

Actually, ϕ is ∞-submodular iff there is an N so that ϕ is p-submodular for all $p \in (N, \infty)$.

$E_p(\ell)$ with $1 \le p < \infty$ 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 ∞-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 ∞-submodularity imply p-submodularity for all p. In such case min-cut/max-flow algorithm optimizes E_p for every $p < \infty$.

Actually, ϕ is ∞-submodular iff there is an N so that ϕ is p-submodular for all $p \in (N, \infty)$.

$E_p(\ell)$ with $1 \leq p < \infty$ vs $E_\infty(\ell)$

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

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

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

Theorem (Malmberg & Strand, IWCIA 2018)

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

Actually, ϕ is ∞-submodular iff there is an N so that ϕ is p-submodular for all $p \in (N, \infty)$.

$E_p(\ell)$ with $1 \le p < \infty$ 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 ∞-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 ∞-submodularity imply p-submodularity for all p. In such case min-cut/max-flow algorithm optimizes E_p for every $p < \infty$.

Actually, ϕ is ∞-submodular iff there is an N so that ϕ is p-submodular for all $p \in (N, \infty)$.

Outline

1. Background: the energies we will optimize
2. Algorithms for L_p, $p < \infty$; NP-completeness
3. **Which max-norm energies E_∞ can be efficiently optimized?**
4. New algorithms optimizing E_∞ for 2-labeling
5. Strict max-norm optimality
6. Summary and conclusions

FC segmentations are E_∞ optimized segmentations

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

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_∞s are there efficient optimizing algorithms?

FC segmentations are E_∞ optimized segmentations

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

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_∞s are there efficient optimizing algorithms?

FC segmentations are E_∞ optimized segmentations

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

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_∞s are there efficient optimizing algorithms?

FC segmentations are E_∞ optimized segmentations

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

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_∞s are there efficient optimizing algorithms?

FC segmentations are E_∞ optimized segmentations

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

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_∞s are there efficient optimizing algorithms?

FC segmentations are E_∞ optimized segmentations

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

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_∞s are there efficient optimizing algorithms?

FC segmentations are E_∞ optimized segmentations

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

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_∞s are there efficient optimizing algorithms?

Efficient algorithm for 2-labeling ∞-submodular E_∞?

YES! ∞-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 ∞-submodular energy E_∞.

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 ∞-submodularity assumption essential in the thm?

Q2: Is there efficient algorithm for $\geq$ 3-labelings?

Efficient algorithm for 2-labeling ∞-submodular E_∞?

YES! ∞-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 ∞-submodular energy E_∞.

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 ∞-submodularity assumption essential in the thm?

Q2: Is there efficient algorithm for $\geq$ 3-labelings?

Efficient algorithm for 2-labeling ∞-submodular E_∞?

YES! ∞-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 ∞-submodular energy E_∞.

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 ∞-submodularity assumption essential in the thm?

Q2: Is there efficient algorithm for $\geq$ 3-labelings?

Efficient algorithm for 2-labeling ∞-submodular E_∞?

YES! ∞-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 ∞-submodular energy E_∞.

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 ∞-submodularity assumption essential in the thm?

Q2: Is there efficient algorithm for $\geq$ 3-labelings?

Efficient algorithm for 2-labeling ∞-submodular E_∞?

YES! ∞-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 ∞-submodular energy E_∞.

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 ∞-submodularity assumption essential in the thm?

Q2: Is there efficient algorithm for $\geq$ 3-labelings?

Efficient algorithm for 2-labeling ∞-submodular E_∞?

YES! ∞-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 ∞-submodular energy E_∞.

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 ∞-submodularity assumption essential in the thm?

Q2: Is there efficient algorithm for $\geq$ 3-labelings?

Efficient algorithm for 2-labeling ∞-submodular E_∞?

YES! ∞-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 ∞-submodular energy E_∞.

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 ∞-submodularity assumption essential in the thm?

Q2: Is there efficient algorithm for $\geq$ 3-labelings?

Efficient algorithm for 2-labeling ∞-submodular E_∞?

YES! ∞-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 ∞-submodular energy E_∞.

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 ∞-submodularity assumption essential in the thm?

Q2: Is there efficient algorithm for $\geq$ 3-labelings?

Optimal 2-labeling for $E_\infty(\ell)$ with no ∞-submodularity

Full answer to Q1: 2-sat algorithm

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.

Optimal 2-labeling for $E_\infty(\ell)$ with no ∞-submodularity

Full answer to Q1: 2-sat algorithm

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.

Optimal 2-labeling for $E_\infty(\ell)$ with no ∞-submodularity

Full answer to Q1: 2-sat algorithm

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.

Optimal 2-labeling for $E_\infty(\ell)$ with no ∞-submodularity

Full answer to Q1: 2-sat algorithm

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.

Optimal 2-labeling for $E_\infty(\ell)$ with no ∞-submodularity

Full answer to Q1: 2-sat algorithm

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.

Q2: What about optimal $\geq$ 3-labeling for $E_\infty(\ell)$?

Partial answer to Q2:

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

Optimization problem of the general form of E_∞ 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_∞ energy for 3 or more labels?

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

Q2: What about optimal $\geq$ 3-labeling for $E_\infty(\ell)$?

Partial answer to Q2:

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

Optimization problem of the general form of E_∞ 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_∞ energy for 3 or more labels?

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

Q2: What about optimal $\geq$ 3-labeling for $E_\infty(\ell)$?

Partial answer to Q2:

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

Optimization problem of the general form of E_∞ 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_∞ energy for 3 or more labels?

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

Q2: What about optimal $\geq$ 3-labeling for $E_\infty(\ell)$?

Partial answer to Q2:

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

Optimization problem of the general form of E_∞ 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_∞ energy for 3 or more labels?

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

Q2: What about optimal $\geq$ 3-labeling for $E_\infty(\ell)$?

Partial answer to Q2:

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

Optimization problem of the general form of E_∞ 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_∞ energy for 3 or more labels?

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

Q2: What about optimal $\geq$ 3-labeling for $E_\infty(\ell)$?

Partial answer to Q2:

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

Optimization problem of the general form of E_∞ 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_∞ energy for 3 or more labels?

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

Q2: What about optimal $\geq$ 3-labeling for $E_\infty(\ell)$?

Partial answer to Q2:

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

Optimization problem of the general form of E_∞ 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_∞ energy for 3 or more labels?

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

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, ℓ is a coloring of $\mathcal{G}$.

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

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, ℓ is a coloring of $\mathcal{G}$.

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

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, ℓ is a coloring of $\mathcal{G}$.

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

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, ℓ is a coloring of $\mathcal{G}$.

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

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, ℓ is a coloring of $\mathcal{G}$.

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

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, ℓ is a coloring of $\mathcal{G}$.

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

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, ℓ is a coloring of $\mathcal{G}$.

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

Outline

1. Background: the energies we will optimize
2. Algorithms for L_p, $p < \infty$; NP-completeness
3. Which max-norm energies E_∞ can be efficiently optimized?
4. New algorithms optimizing E_∞ for 2-labeling
5. Strict max-norm optimality
6. Summary and conclusions

Atoms of E_∞ and their cost

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

Atoms $\mathcal{A}(\ell)$ of ℓ: input for $\phi_{\cdot\cdot}$ and $\phi_{\cdot}$ (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_∞: 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 $\{(s,i),(t,j)\}$: $\phi_{st}(i,j)$

Set $\mathcal{A}'\subset\mathcal{A}$ of atoms is consistent when $\mathcal{A}(\ell)\subset\mathcal{A}'$ for some ℓ.

Atoms of E_∞ and their cost

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

Atoms $\mathcal{A}(\ell)$ of ℓ: input for $\phi_{\cdot\cdot}$ and $\phi_\cdot$ (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_∞: 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 $\{(s,i),(t,j)\}$: $\phi_{st}(i,j)$

Set $\mathcal{A}'\subset\mathcal{A}$ of atoms is consistent when $\mathcal{A}(\ell)\subset\mathcal{A}'$ for some ℓ.

Atoms of E_∞ and their cost

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

Atoms $\mathcal{A}(\ell)$ of ℓ: 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_∞: 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 $\{(s,i),(t,j)\}$: $\phi_{st}(i,j)$

Set $\mathcal{A}'\subset\mathcal{A}$ of atoms is consistent when $\mathcal{A}(\ell)\subset\mathcal{A}'$ for some ℓ.

Atoms of E_∞ and their cost

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

Atoms $\mathcal{A}(\ell)$ of ℓ: input for $\phi_{..}$ and $\phi_.$ (to calculate $E_\infty(\ell)$), i.e.,

$$\mathcal{A}(\ell) := \{\{(s,\ell(s))\} \colon s\in V\} \cup \{\{(s,\ell(s)),(t,\ell(t))\} \colon \{s,t\}\in\mathcal{E}\}$$

Atoms $\mathcal{A}$ of E_∞: 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 $\{(s,i),(t,j)\}$: $\phi_{st}(i,j)$

Set $\mathcal{A}' \subset \mathcal{A}$ of atoms is consistent when $\mathcal{A}(\ell) \subset \mathcal{A}'$ for some ℓ.

Atoms of E_∞ and their cost

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

Atoms $\mathcal{A}(\ell)$ of ℓ: input for $\phi_{..}$ and $\phi_.$ (to calculate $E_\infty(\ell)$), i.e.,

$$\mathcal{A}(\ell) := \left\{\{(s,\ell(s))\}\colon s\in V\right\}\cup\left\{\{(s,\ell(s)),(t,\ell(t))\}\colon \{s,t\}\in\mathcal{E}\right\}$$

Atoms $\mathcal{A}$ of E_∞: 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 $\{(s,i),(t,j)\}$: $\phi_{st}(i,j)$

Set $\mathcal{A}'\subset\mathcal{A}$ of atoms is consistent when $\mathcal{A}(\ell)\subset\mathcal{A}'$ for some ℓ.

Atoms of E_∞ and their cost

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

Atoms $\mathcal{A}(\ell)$ of ℓ: input for $\phi_{\cdot\cdot}$ and $\phi_\cdot$ (to calculate $E_\infty(\ell)$), i.e.,

$$\mathcal{A}(\ell) := \{\{(s,\ell(s))\}\colon s\in V\} \cup \{\{(s,\ell(s)),(t,\ell(t))\}\colon \{s,t\}\in\mathcal{E}\}$$

Atoms $\mathcal{A}$ of E_∞: 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 $\{(s,i),(t,j)\}$: $\phi_{st}(i,j)$

Set $\mathcal{A}'\subset\mathcal{A}$ of atoms is consistent when $\mathcal{A}(\ell)\subset\mathcal{A}'$ for some ℓ.

Atoms of E_∞ and their cost

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

Atoms $\mathcal{A}(\ell)$ of ℓ: input for $\phi_{\cdot\cdot}$ and $\phi_\cdot$ (to calculate $E_\infty(\ell)$), i.e.,

$$\mathcal{A}(\ell) := \left\{\{(s,\ell(s))\}\colon s\in V\right\} \cup \left\{\{(s,\ell(s)),(t,\ell(t))\}\colon \{s,t\}\in\mathcal{E}\right\}$$

Atoms $\mathcal{A}$ of E_∞: 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 $\{(s,i),(t,j)\}$: $\phi_{st}(i,j)$

Set $\mathcal{A}'\subset\mathcal{A}$ of atoms is consistent when $\mathcal{A}(\ell)\subset\mathcal{A}'$ for some ℓ.

Atoms of E_∞ and their cost

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

Atoms $\mathcal{A}(\ell)$ of ℓ: 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_∞: 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 $\{(s,i),(t,j)\}$: $\phi_{st}(i,j)$

Set $\mathcal{A}' \subset \mathcal{A}$ of atoms is consistent when $\mathcal{A}(\ell) \subset \mathcal{A}'$ for some ℓ.

Atoms of E_∞ and their cost

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

Atoms $\mathcal{A}(\ell)$ of ℓ: 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_∞: 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 $\{(s,i),(t,j)\}$: $\phi_{st}(i,j)$

Set $\mathcal{A}' \subset \mathcal{A}$ of atoms is consistent when $\mathcal{A}(\ell) \subset \mathcal{A}'$ for some ℓ.

∞-sub algorithm

1. 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.
2. 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 $S \cup L$
3. Return labeling $\ell = \bigcup L$

The locally inconsistency loop is natural.

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

∞-sub algorithm

1. 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.
2. 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 $S \cup L$
3. Return labeling $\ell = \bigcup L$

The locally inconsistency loop is natural.

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

∞-sub algorithm

1. 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.
2. 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 $S \cup L$
3. Return labeling $\ell = \bigcup L$

The locally inconsistency loop is natural.

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

∞-sub algorithm

1. 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.
2. 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 $S \cup L$
3. Return labeling $\ell = \bigcup L$

The locally inconsistency loop is natural.

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

∞-sub algorithm

1. 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.
2. 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 $S \cup L$
3. Return labeling $\ell = \bigcup L$

The locally inconsistency loop is natural.

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

∞-sub algorithm

1. 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.
2. 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 $S \cup L$
3. Return labeling $\ell = \bigcup L$

The locally inconsistency loop is natural.

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

∞-sub algorithm

1. 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.
2. 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 $S \cup L$
3. Return labeling $\ell = \bigcup L$

The locally inconsistency loop is natural.

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

∞-sub algorithm

1. 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.
2. 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 $S \cup L$
3. Return labeling $\ell = \bigcup L$

The locally inconsistency loop is natural.

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

∞-sub algorithm

1. 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.
2. 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 $S \cup L$
3. Return labeling $\ell = \bigcup L$

The locally inconsistency loop is natural.

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

Towards 2-sat algorithm: 2-satisfiability

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

$$\psi_A(s) := \text{“}s \neq i\text{”} = \begin{cases} \neg s & \text{if } i = 1, \\ s & \text{if } i = 0 \end{cases}$$

$$\psi_A(s, t) := \text{“}(s \neq i) \vee (t \neq j)\text{”} = \psi_{\{(s,i)\}}(s) \vee \psi_{\{(t,j)\}}(t).$$

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

Theorem

A set $\mathcal{A}_1 \subseteq \mathcal{A}$ of atoms is consistent if, and only if, the 2-satisfiability problem for a formula $\psi_{\mathcal{A}_1^c}$ has a positive solution.

So, consistency of $\mathcal{A}_1 \subseteq \mathcal{A}$ can be decided by in linear time. (Aspvall et al. algorithm.)

Towards 2-sat algorithm: 2-satisfiability

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

$$\psi_A(s) := \text{“}s \neq i\text{”} = \begin{cases} \neg s & \text{if } i = 1, \\ s & \text{if } i = 0 \end{cases}$$

$$\psi_A(s,t) := \text{“}(s \neq i) \vee (t \neq j)\text{”} = \psi_{\{(s,i)\}}(s) \vee \psi_{\{(t,j)\}}(t).$$

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

Theorem

A set $\mathcal{A}_1 \subseteq \mathcal{A}$ of atoms is consistent if, and only if, the 2-satisfiability problem for a formula $\psi_{\mathcal{A}_1^c}$ has a positive solution.

So, consistency of $\mathcal{A}_1 \subseteq \mathcal{A}$ can be decided by in linear time. (Aspvall et al. algorithm.)

Towards 2-sat algorithm: 2-satisfiability

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

$$\psi_A(s) := \text{“}s \neq i\text{”} = \begin{cases} \neg s & \text{if } i = 1, \\ s & \text{if } i = 0 \end{cases}$$

$$\psi_A(s,t) := \text{“}(s \neq i) \vee (t \neq j)\text{”} = \psi_{\{(s,i)\}}(s) \vee \psi_{\{(t,j)\}}(t).$$

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

Theorem

A set $\mathcal{A}_1 \subseteq \mathcal{A}$ of atoms is consistent if, and only if, the 2-satisfiability problem for a formula $\psi_{\mathcal{A}_1^c}$ has a positive solution.

So, consistency of $\mathcal{A}_1 \subseteq \mathcal{A}$ can be decided by in linear time. (Aspvall et al. algorithm.)

Towards 2-sat algorithm: 2-satisfiability

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

$$\psi_A(s) := \text{“}s \neq i\text{”} = \begin{cases} \neg s & \text{if } i = 1, \\ s & \text{if } i = 0 \end{cases}$$

$$\psi_A(s,t) := \text{“}(s \neq i) \vee (t \neq j)\text{”} = \psi_{\{(s,i)\}}(s) \vee \psi_{\{(t,j)\}}(t).$$

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

Theorem

A set $\mathcal{A}_1 \subseteq \mathcal{A}$ of atoms is consistent if, and only if, the 2-satisfiability problem for a formula $\psi_{\mathcal{A}_1^c}$ has a positive solution.

So, consistency of $\mathcal{A}_1 \subseteq \mathcal{A}$ can be decided by in linear time. (Aspvall et al. algorithm.)

Towards 2-sat algorithm: 2-satisfiability

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

$$\psi_A(s) := \text{“}s \neq i\text{”} = \begin{cases} \neg s & \text{if } i = 1, \\ s & \text{if } i = 0 \end{cases}$$

$$\psi_A(s, t) := \text{“}(s \neq i) \vee (t \neq j)\text{”} = \psi_{\{(s,i)\}}(s) \vee \psi_{\{(t,j)\}}(t).$$

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

Theorem

A set $\mathcal{A}_1 \subseteq \mathcal{A}$ of atoms is consistent if, and only if, the 2-satisfiability problem for a formula $\psi_{\mathcal{A}_1^c}$ has a positive solution.

So, consistency of $\mathcal{A}_1 \subseteq \mathcal{A}$ can be decided by in linear time. (Aspvall et al. algorithm.)

2-sat algorithm

1. List all atoms in a list S in a decreasing cost
2. 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.

2-sat algorithm

1. List all atoms in a list S in a decreasing cost
2. 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.

2-sat algorithm

1. List all atoms in a list S in a decreasing cost
2. 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.

2-sat algorithm

1. List all atoms in a list S in a decreasing cost
2. 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.

2-sat algorithm

1. List all atoms in a list S in a decreasing cost
2. 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.

2-sat algorithm

1. List all atoms in a list S in a decreasing cost
2. 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.

Outline

1. Background: the energies we will optimize
2. Algorithms for L_p, $p < \infty$; NP-completeness
3. Which max-norm energies E_∞ can be efficiently optimized?
4. New algorithms optimizing E_∞ for 2-labeling
5. Strict max-norm optimality
6. Summary and conclusions

Strict optimality via lexicographical order

Max-norm identifies ℓ_1 and ℓ_2 when $E_\infty(\ell_1) = E_\infty(\ell_2)$.

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

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

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

ℓ 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?

Strict optimality via lexicographical order

Max-norm identifies ℓ_1 and ℓ_2 when $E_\infty(\ell_1) = E_\infty(\ell_2)$.

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

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

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

ℓ 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?

Strict optimality via lexicographical order

Max-norm identifies ℓ_1 and ℓ_2 when $E_\infty(\ell_1) = E_\infty(\ell_2)$.

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

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

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

ℓ 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?

Strict optimality via lexicographical order

Max-norm identifies ℓ_1 and ℓ_2 when $E_\infty(\ell_1) = E_\infty(\ell_2)$.

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

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

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

ℓ 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?

Strict optimality via lexicographical order

Max-norm identifies ℓ_1 and ℓ_2 when $E_\infty(\ell_1) = E_\infty(\ell_2)$.

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

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

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

ℓ 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?

Strict optimality via lexicographical order

Max-norm identifies ℓ_1 and ℓ_2 when $E_\infty(\ell_1) = E_\infty(\ell_2)$.

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

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

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

ℓ 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?

Strict optimality via lexicographical order

Max-norm identifies ℓ_1 and ℓ_2 when $E_\infty(\ell_1) = E_\infty(\ell_2)$.

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

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

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

ℓ 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?

Outline

Summary (including new results)

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

Summary (including new results)

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

Summary (including new results)

	2 labels	≥ 3 labels
general case strict optimization	**NP-hard problem**	NP-hard problem
∞-submodular strict optimization	max-flow/min-cut $O(n^2 \ln n) \leq \cdot \leq O(n^3)$	NP-hard problem
unique weights strict optimization	**2-sat algorithm** $O(n^2)$	NP-hard problem
general case	2-sat algorithm; $O(n^2)$	NP-hard problem
∞-submodular	∞-sub algorithm $O(n) \leq \cdot \leq O(n \ln n)$	NP-hard problem
$\phi_s(i) = \phi_{st}(i,i) = 0$; $\phi_{st}(i,j) = \phi_{st}(j,i) \geq 0$	Dijkstra algorithm $O(n) \leq \cdot \leq O(n \ln n)$	Dijkstra algorithm $O(n) \leq \cdot \leq O(n \ln n)$

Summary (including new results)

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

Summary (including new results)

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

Summary (including new results)

	2 labels	≥ 3 labels
general case strict optimization	**NP-hard problem**	NP-hard problem
∞-submodular strict optimization	max-flow/min-cut $O(n^2 \ln n) \leq \cdot \leq O(n^3)$	NP-hard problem
unique weights strict optimization	**2-sat algorithm** $O(n^2)$	NP-hard problem
general case	2-sat algorithm; $O(n^2)$	NP-hard problem
∞-submodular	∞-sub algorithm $O(n) \leq \cdot \leq O(n \ln n)$	NP-hard problem
$\phi_s(i) = \phi_{st}(i,i) = 0;$ $\phi_{st}(i,j) = \phi_{st}(j,i) \geq 0$	Dijkstra algorithm $O(n) \leq \cdot \leq O(n \ln n)$	Dijkstra algorithm $O(n) \leq \cdot \leq O(n \ln n)$

Conclusions

- 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.

Conclusions

- 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.

Conclusions

- 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.

Conclusions

- 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.

Thank you for your attention!