Memory Efficient Max Flow for Multi-label Submodular MRFs

Thalaiyasingam Ajanthan Richard Hartley Mathieu Salzmann

The Australian National University

Data61, CSIRO

Data61, May 2016

Introduction

Minimize

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

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

Multi-label submodular

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

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

Current method

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

Introduction

Minimize

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

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

Multi-label submodular

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

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

Current method

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

Introduction

Minimize

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

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

Multi-label submodular

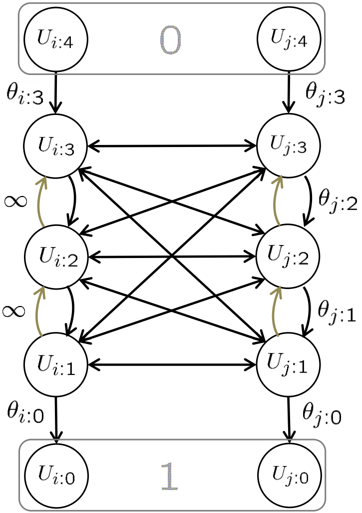
$$\theta_{ij}(\lambda', \mu) + \theta_{ij}(\lambda, \mu') - \theta_{ij}(\lambda, \mu) - \theta_{ij}(\lambda', \mu') \geq 0 \ ,$$

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

Current method

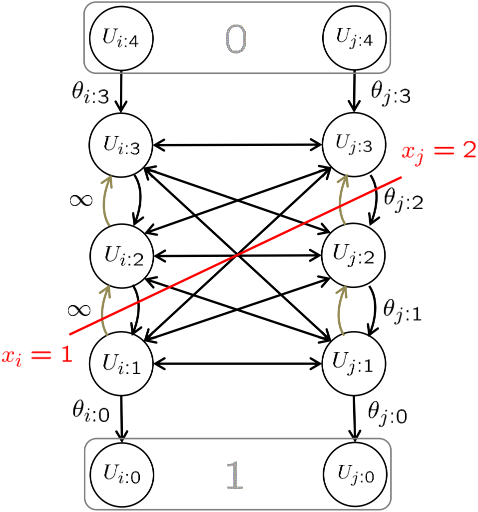
- Ishikawa algorithm [Ishikawa-2003, Schlesinger-2006]

The Ishikawa algorithm



The Ishikawa graph

The Ishikawa algorithm



The Ishikawa graph

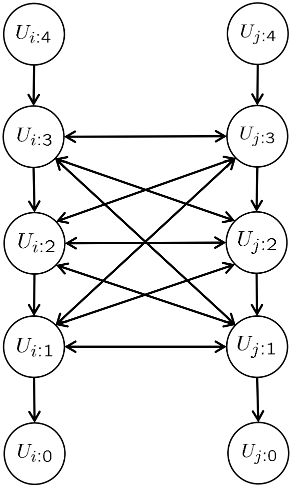
The Ishikawa algorithm

Drawback

- Stores $2\,\ell^2$ edges for each pair of neighbours.

Idea

- Stores $2\,\ell$ values for each pair of neighbours.



The Ishikawa algorithm

Drawback

- Stores $2\,\ell^2$ edges for each pair of neighbours.

E.g. $|\mathcal{V}| = 10^6$, $\ell = 256$
Edges $\approx 2 \times 10^6 \times 2 \times 256^2$
Memory $\approx$ 1000 GB

$\Downarrow$

Idea

- Stores $2\,\ell$ values for each pair of neighbours.

Memory $\approx$ 4 GB

The Ishikawa algorithm

Drawback

- Stores $2\,\ell^2$ edges for each pair of neighbours.

Idea

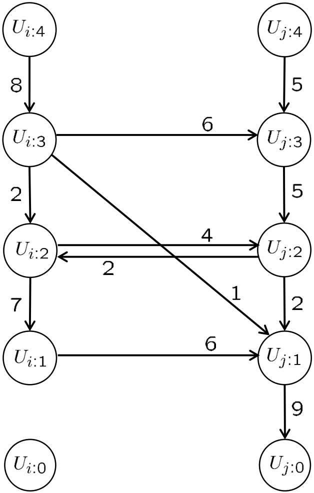
- Stores $2\,\ell$ values for each pair of neighbours.

E.g. $|\mathcal{V}| = 10^6$, $\ell = 256$
Edges $\approx 2 \times 10^6 \times 2 \times 256^2$
Memory $\approx$ 1000 GB

$\Downarrow$

Memory $\approx$ 4 GB

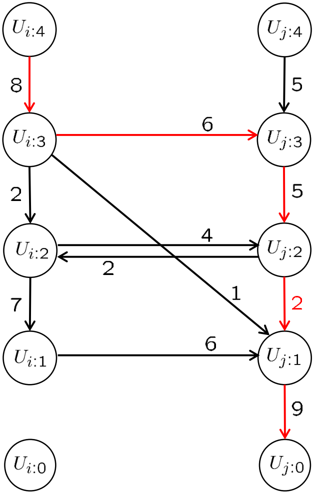
Max flow on the Ishikawa graph



Initial Ishikawa graph

Flow = 0

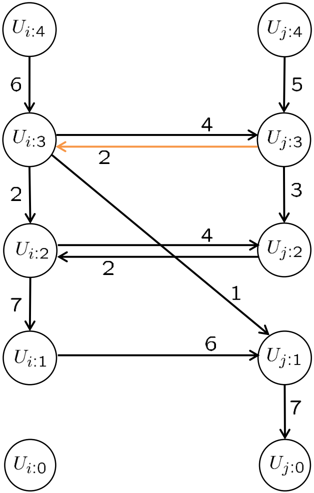
Max flow on the Ishikawa graph



Max-flow in progress

Flow $= 0$

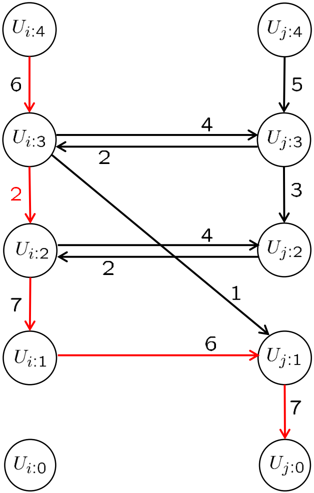
Max flow on the Ishikawa graph



Max-flow in progress

Flow = 2

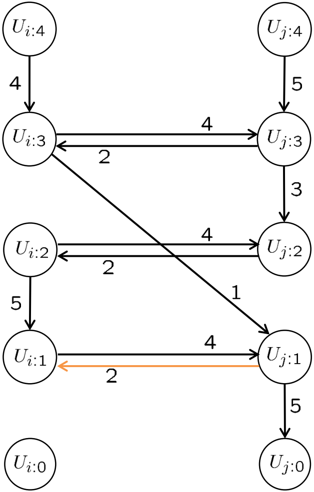
Max flow on the Ishikawa graph



Max-flow in progress

Flow = 2

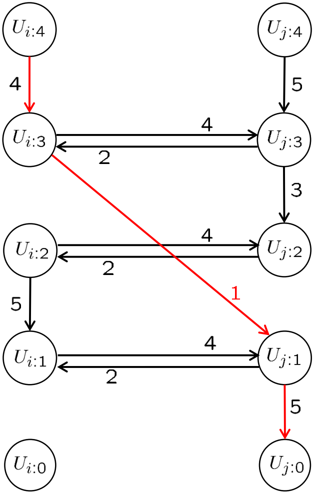
Max flow on the Ishikawa graph



Max-flow in progress

Flow $= 4$

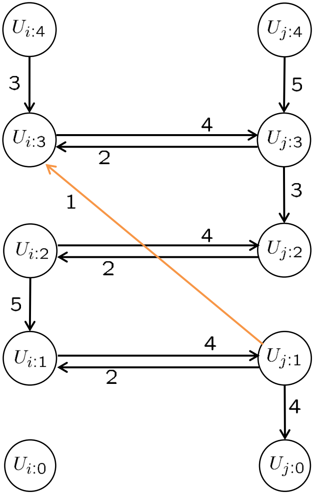
Max flow on the Ishikawa graph



Max-flow in progress

Flow $= 4$

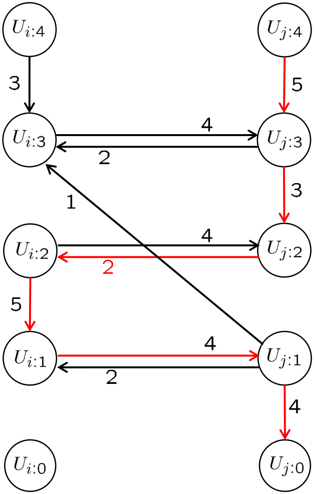
Max flow on the Ishikawa graph



Max-flow in progress

Flow $= 5$

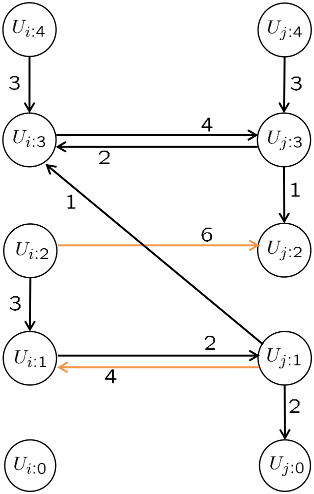
Max flow on the Ishikawa graph



Max-flow in progress

Flow $= 5$

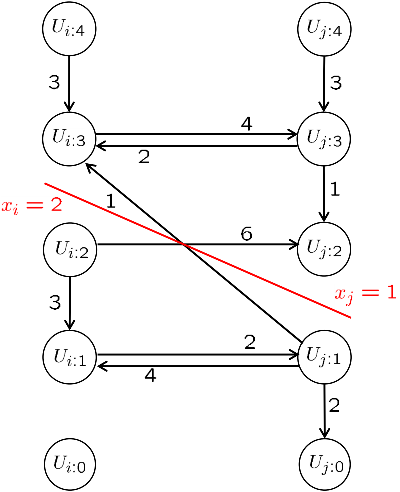
Max flow on the Ishikawa graph



Max-flow in progress

Flow $= 7$

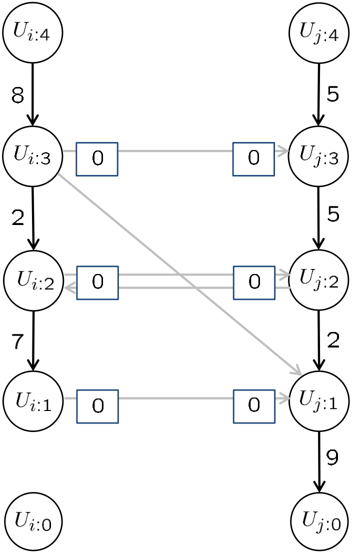
Max flow on the Ishikawa graph



Min-cut

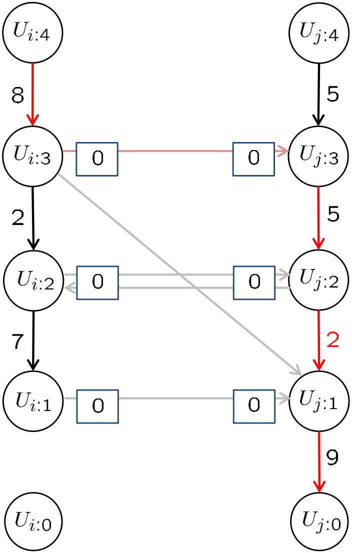
Flow $= 7$

Memory efficient flow encoding



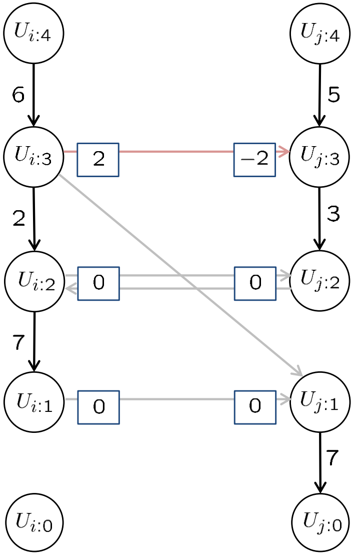
Initial exit-flows

Memory efficient flow encoding



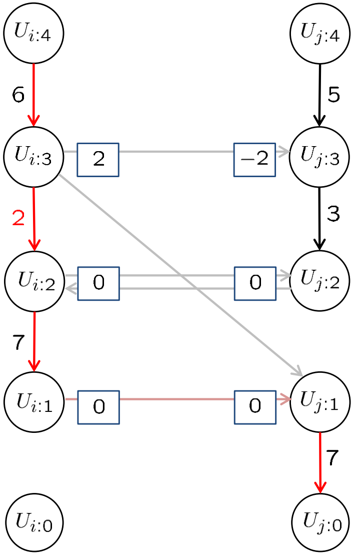
Update exit-flows

Memory efficient flow encoding



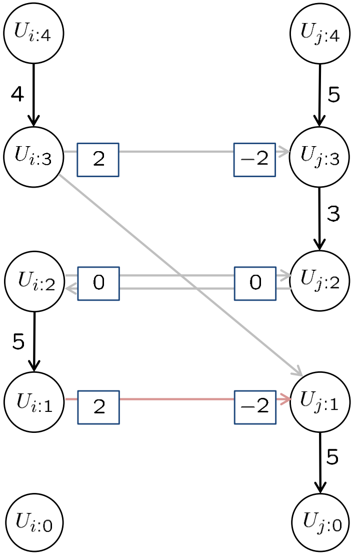
Update exit-flows

Memory efficient flow encoding



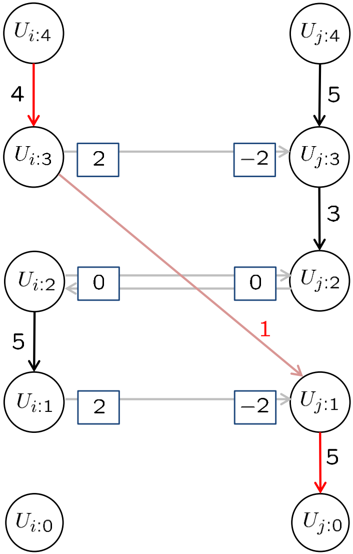
Update exit-flows

Memory efficient flow encoding



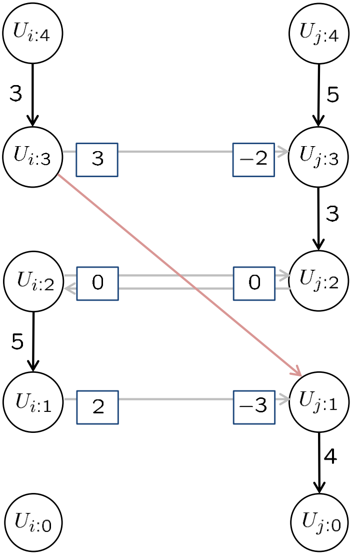
Update exit-flows

Memory efficient flow encoding



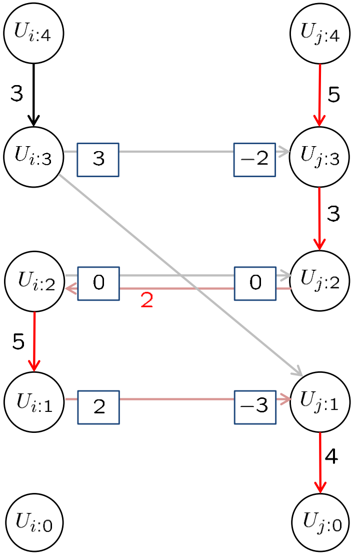
Update exit-flows

Memory efficient flow encoding



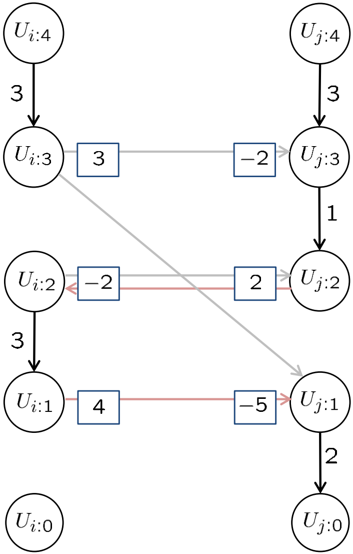
Update exit-flows

Memory efficient flow encoding



Update exit-flows

Memory efficient flow encoding

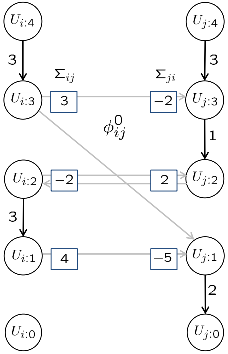


Update exit-flows

Memory efficient max flow

Algorithm

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



Memory efficient max flow

Algorithm

Require: ϕ^0 ▷ Initial Ishikawa capacities

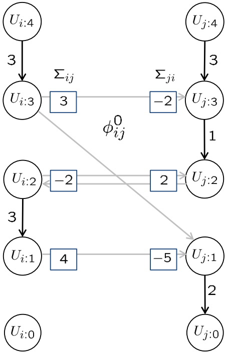
$\Sigma \leftarrow 0$ ▷ Initialize exit-flows

repeat

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

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

until no augmenting paths possible



Space complexity: $\mathcal{O}(|\mathcal{E}|\ell)$

Memory efficient max flow

Algorithm

Require: ϕ^0 ▷ Initial Ishikawa capacities

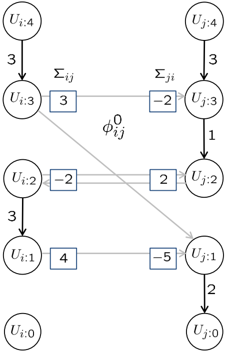
$\Sigma \leftarrow 0$ ▷ Initialize exit-flows

repeat

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

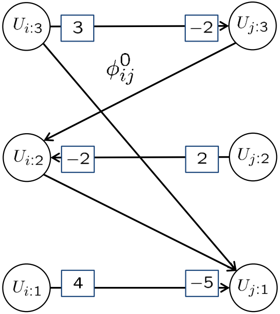
$\Sigma \leftarrow \text{augment}(P, \phi^0, \Sigma)$

until no augmenting paths possible

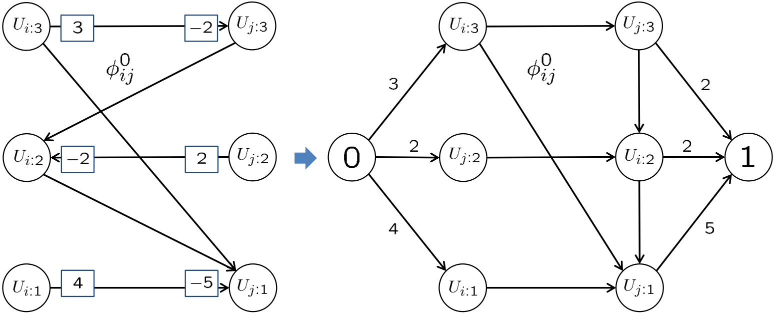


Space complexity: $\mathcal{O}(|\mathcal{E}|\ell)$

Flow reconstruction / Computing residual edges

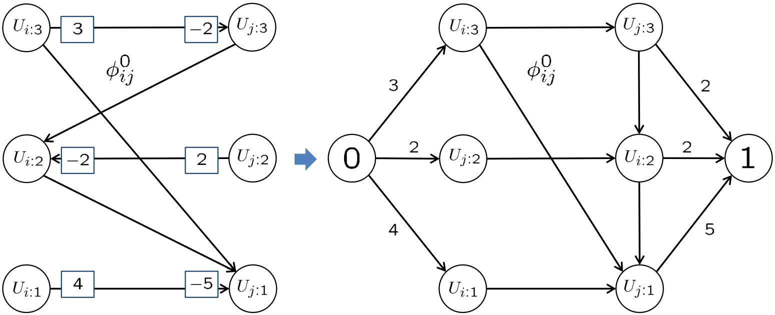


Flow reconstruction / Computing residual edges



Flow reconstruction as a small max-flow problem

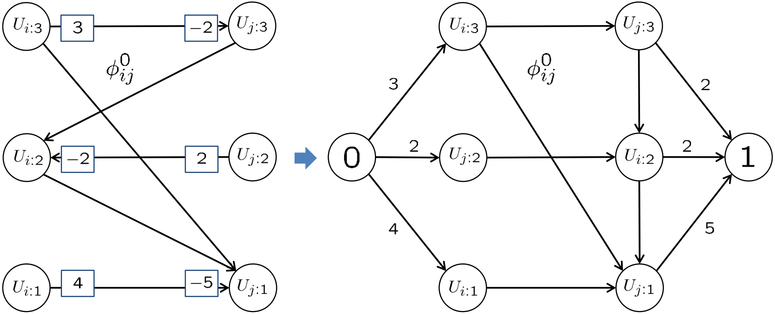
Flow reconstruction / Computing residual edges



Flow reconstruction as a small max-flow problem

All flow-reconstructions are equivalent.

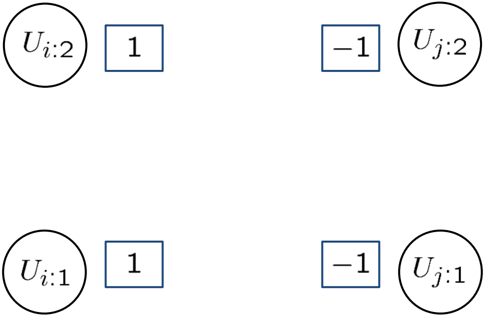
Flow reconstruction / Computing residual edges



Flow reconstruction as a small max-flow problem

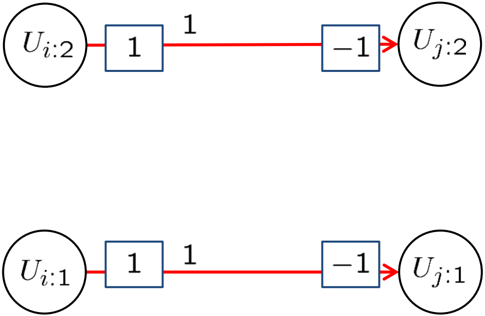
Time complexity: $\mathcal{O}(\ell^3)$

Flow equivalence - an example



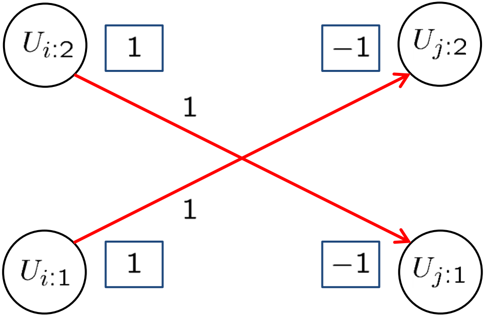
Exit-flows

Flow equivalence - an example



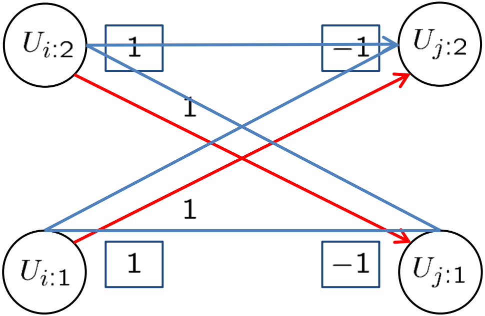
A reconstructed flow

Flow equivalence - an example



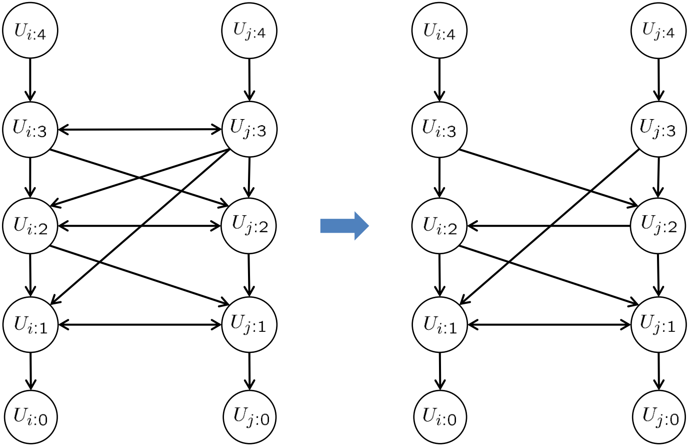
Another reconstructed flow

Flow equivalence - an example



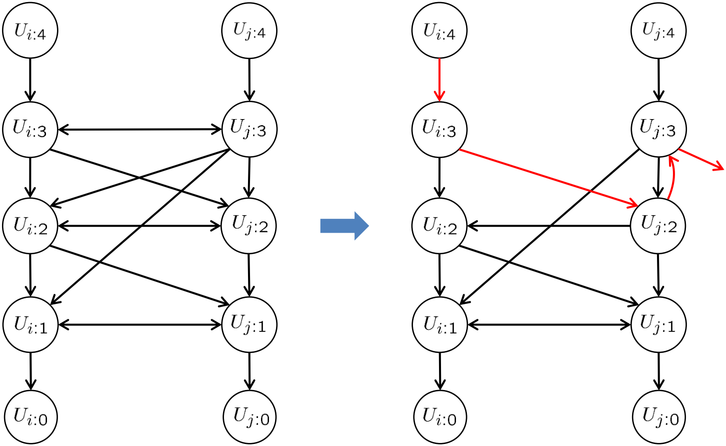
Both reconstructions are equivalent

Finding an augmenting path



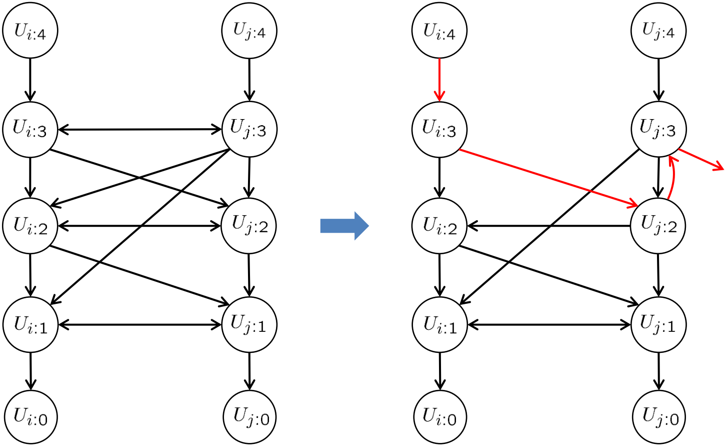
Find augmenting paths on a subgraph

Finding an augmenting path



Find augmenting paths on a subgraph

Finding an augmenting path



Find augmenting paths on a subgraph

Overall time complexity: $\mathcal{O}(|\mathcal{V}||\mathcal{E}|\ell^6)$

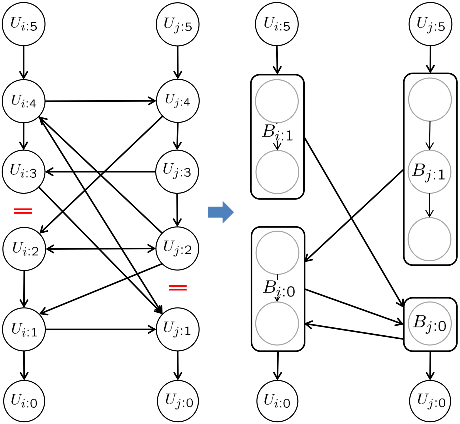
Efficiently finding an augmenting path

Simplified graph

- Sparse graph.
- Fewer augmenting paths.

Search-tree-recycling

- Good empirical performance.



Simplified graph representation

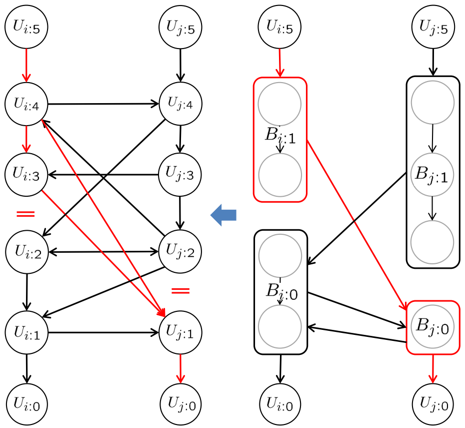
Efficiently finding an augmenting path

Simplified graph

- Sparse graph.
- Fewer augmenting paths.

Search-tree-recycling

- Good empirical performance.



Simplified graph representation

Efficiently finding an augmenting path

Simplified graph

- Sparse graph.
- Fewer augmenting paths.

Search-tree-recycling

- Good empirical performance.

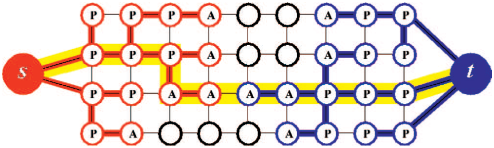


Image courtesy of [Boykov-2004]

Results

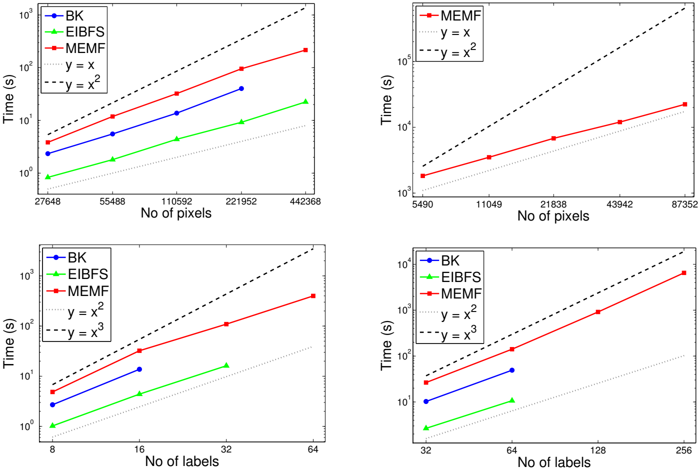
Problem	Memory [MB]			Time [s]		
	BK	EIBFS	**MEMF**	BK	EIBFS	**MEMF**
Tsukuba	3195	2495	211	14	4	30
Venus	7626	5907	396	35	9	60
Sawtooth	7566	5860	393	31	8	35
Map	6454	4946	219	57	9	36
Cones	*72303	*55063	1200	-	-	371
Teddy	*72303	*55063	1200	-	-	2118
KITTI	*88413	*67316	2215	-	-	19008
Penguin	*173893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

Comparison with other max-flow implementations

BK Boykov-2004

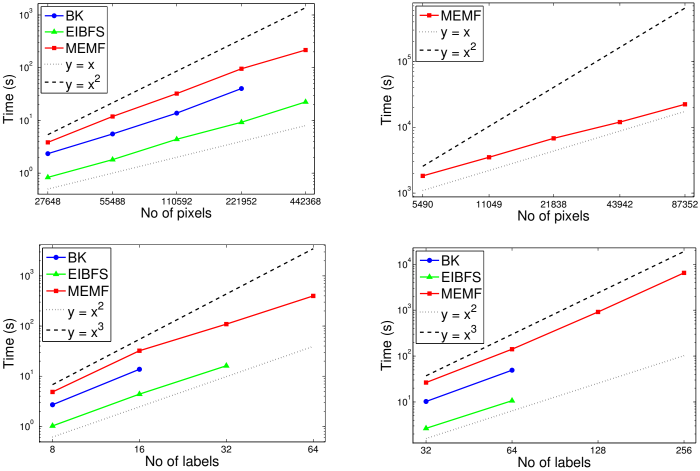
EIBFS Goldberg-2015

Empirical time complexity



Tsukuba *Penguin*

Empirical time complexity



Tsukuba *Penguin*

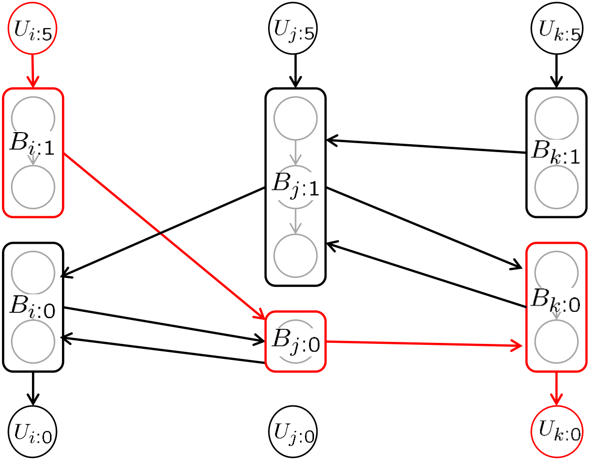
Empirical time complexity: $\mathcal{O}(|\mathcal{V}|\ell^3)$

Conclusion

- We have introduced a memory efficient alternative to the max-flow algorithm that can optimally minimize multi-label submodular MRF energies.

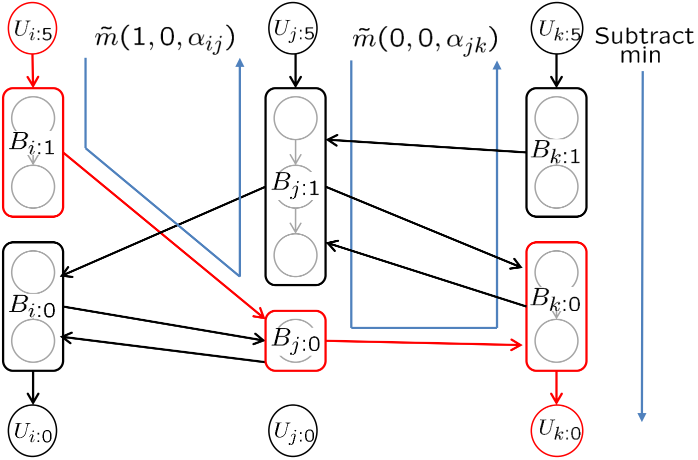
Thank you!

Augmentation



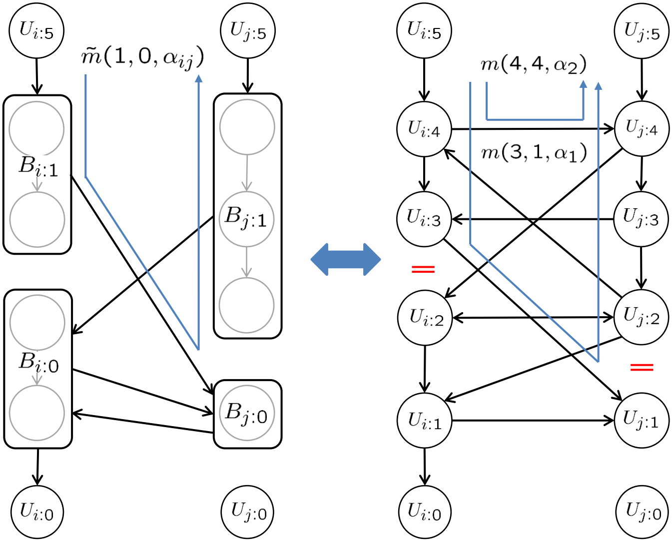
Augmentation

Augmentation



Augmentation

Augmentation



Augmentation