Memory Efficient Max Flow for Multi-label Submodular MRFs

Australian National University DATA 61 CSIRO

Thalaiyasingam Ajanthan[1,2] Richard Hartley[1,2] Mathieu Salzmann[3]

[1]Australian National University [2]Data61, CSIRO [3]CVLab, EPFL

CVPR 2016

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Introduction

Problem: Minimize a multi-label MRF with pairwise interactions

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

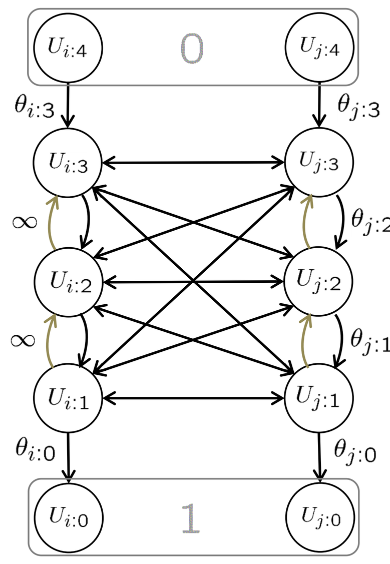
where $x_i \in \{0, 1, \cdots, \ell-1\}$.

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 $\mu < \mu'$ [4].

Current method: Ishikawa algorithm [3].



The Ishikawa graph

Memory complexity: $O(|\mathcal{E}|\ell^2)$

E.g. $|\mathcal{V}| = 10^6$, $\ell = 256$
$|\mathcal{E}| \approx 2 \times 10^6$ (4-connected)
Ishikawa edges $\approx 2\times10^6\times2\times256^2$
Memory $\approx$ 1000 GB

Contribution: An algorithm with memory complexity $O(|\mathcal{E}|\ell)$.

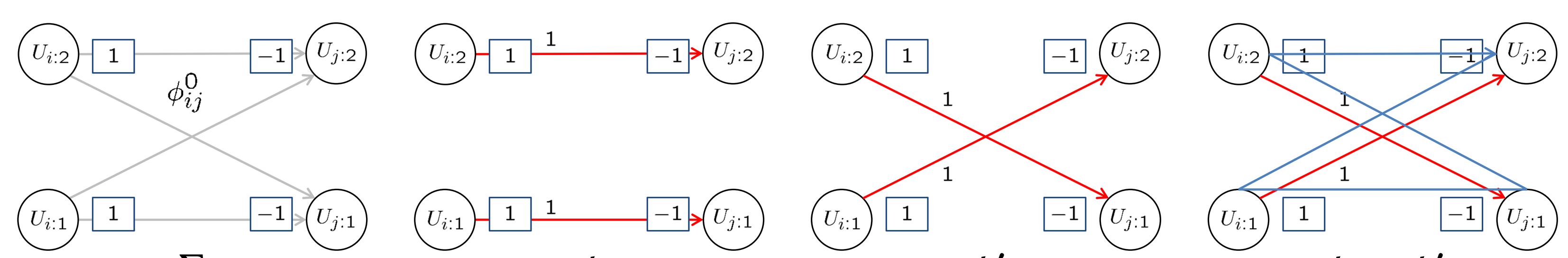
Memory Efficient Flow Encoding

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

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

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

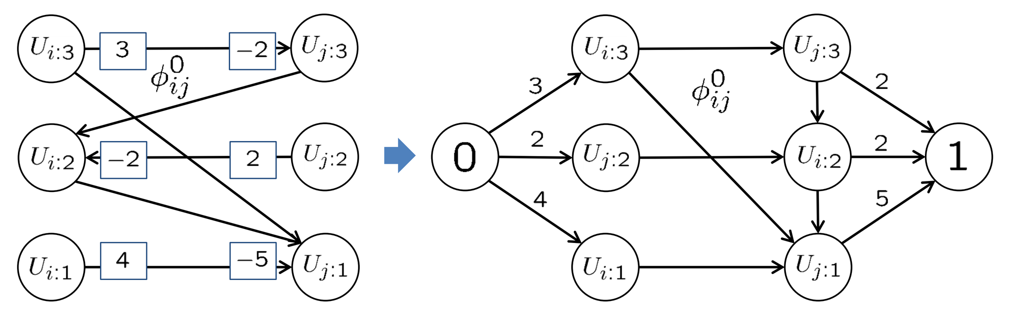
The residual graph can be rapidly computed from the exit-flows.



Σ ψ ψ' $\psi \equiv \psi'$

Rapidly computing the residual graph:

Idea: Formulate a small max-flow problem.



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

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

Efficiently Finding an Augmenting Path

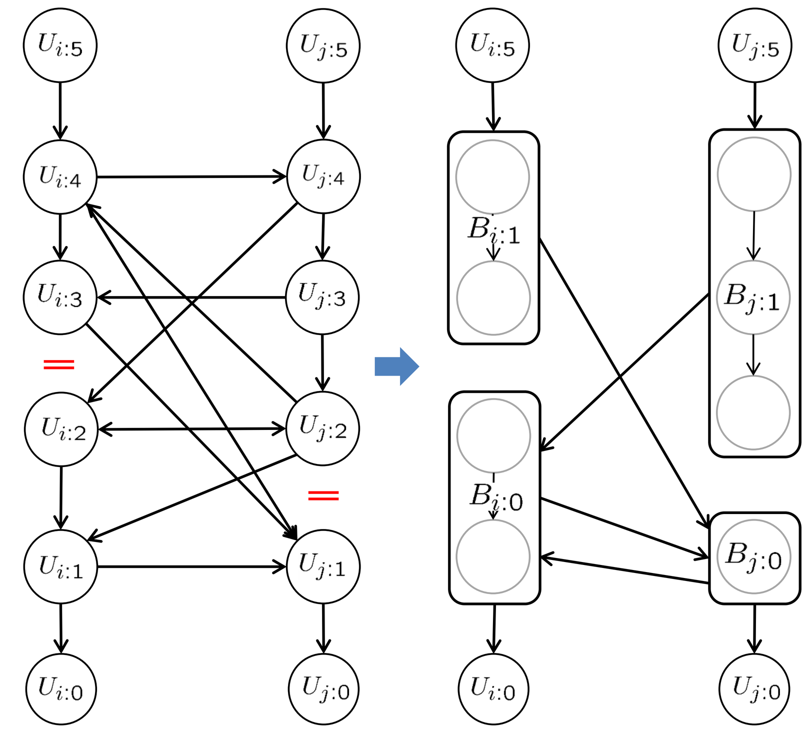
Idea: Search for augmenting paths in a simplified graph.

Simplified graph:

- Unweighted sparse graph.
- Fewer augmenting paths.

Search-tree-recycling:

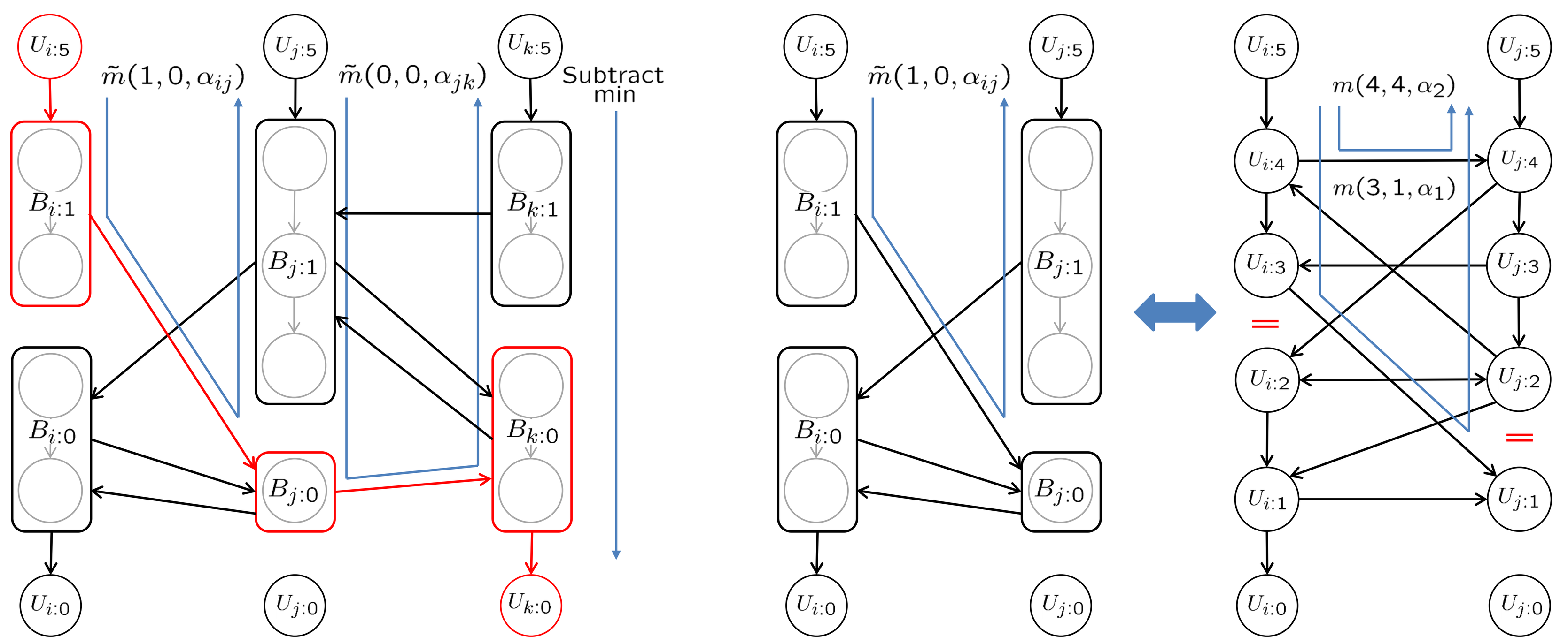
- Good empirical performance.



Augmentation

Idea: Pass flow around loops.

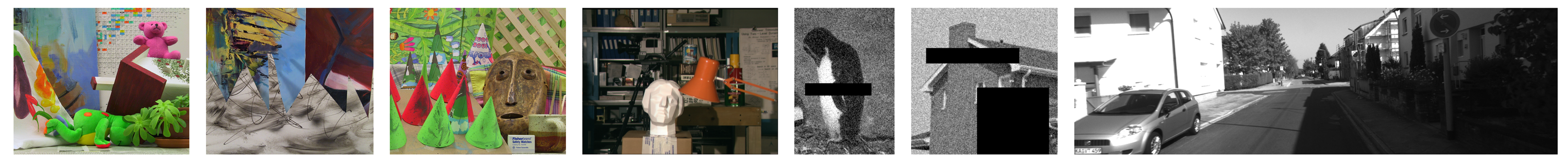
- Push the maximum permissible flow through each flow-loop.
- Applying flow-loops translates to updating the exit-flows.



Experiments

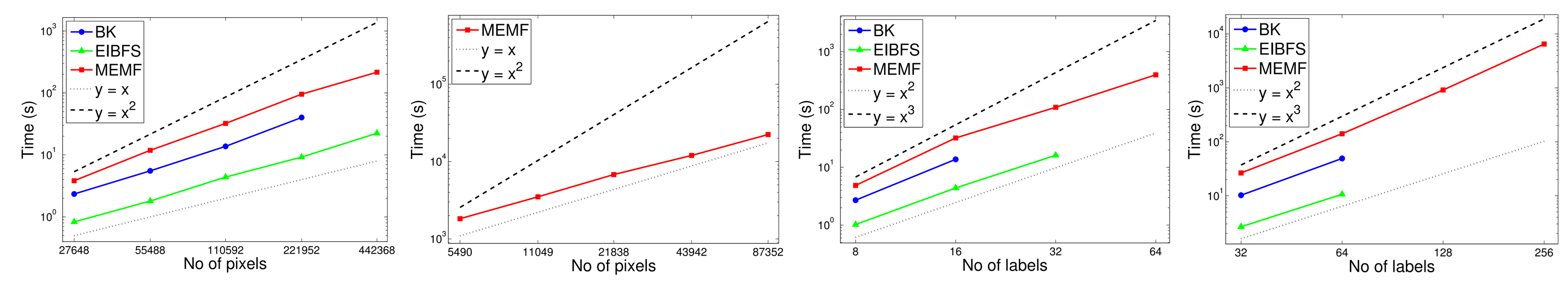
Dataset:

- Middlebury stereo and inpainting instances.
- KITTI stereo instance.



Problem		Memory [MB]			Time [s]		
Name	Labels	BK [1]	EIBFS [2]	**MEMF**	BK	EIBFS	**MEMF**
Tsukuba	16	3195	2495	211	14	4	30
Venus	20	7626	5907	396	35	9	60
Sawtooth	20	7566	5860	393	31	8	35
Map	30	6454	4946	219	57	9	36
Cones	60	*72303	*55063	1200	-	-	371
Teddy	60	*72303	*55063	1200	-	-	2118
KITTI	40	*88413	*67316	2215	-	-	19008
Penguin	256	*173893	*130728	663	-	-	6835
House	256	*521853	*392315	1986	-	-	9290

Comparison with other max-flow implementations



Tsukuba *Penguin* *Tsukuba* *Penguin*

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

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

References

[1] Y Boykov and V Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 2004.

[2] A V Goldberg, S Hed, H Kaplan, P Kohli, R E Tarjan, and R F Werneck. Faster and more dynamic maximum flow by incremental breadth-first search. In *Algorithms–ESA*. 2015.

[3] H Ishikawa. Exact optimization for markov random fields with convex priors. *PAMI*, 2003.

[4] D Schlesinger and B Flach. *Transforming an arbitrary minsum problem into a binary one*. TU, Fak. Informatik, 2006.