Memory Efficient Max Flow for Multi-label Submodular MRFs - Supplementary Material

Thalaiyasingam Ajanthan, Richard Hartley
Australian National University & NICTA
Canberra, Australia

Mathieu Salzmann
CVLAB, EPFL
Lausanne, Switzerland

1. Proof of Theorem 3.2

Theorem. *Given the set of Ishikawa capacities ϕ, there is an augmenting path in the simplified graph if and only if there exists an augmenting path in the Ishikawa graph.*

Proof. First, we will prove that, if there is an augmenting path in the simplified graph, then there exists an augmenting path in the Ishikawa graph. It is clear that an augmenting path in the simplified graph contains an edge from node 0 to a block and then a sequence of edges $B_{i:\gamma} \rightarrow B_{j:\delta}$ and finally an edge from a block to node 1. Note that an edge from node 0 to a block $B_{i:\gamma}$ corresponds to a positive edge $e_{i:\ell-1}$ in the Ishikawa graph; similarly an edge from a block $B_{j:\delta}$ to node 1 corresponds to a positive edge $e_{j:0}$. Now, consider an edge $B_{i:\gamma} \rightarrow B_{j:\delta}$ in the augmenting path. Corresponding to this, there exists a positive edge $e_{ij:\lambda\mu}$ such that $U_{i:\lambda} \in B_{i:\gamma'}$ for some $\gamma' \geq \gamma$ and $U_{j:\mu} \in B_{j:\delta}$ in the Ishikawa graph. Also along the column i, there are upward infinite capacity edges, and nodes corresponding to a block are also connected with positive bidirectional edges. Hence, there exists an augmenting path in the Ishikawa graph, corresponding to the augmenting path in the simplified graph.

Now, we will prove the converse. Consider an augmenting path in the Ishikawa graph. The path may contain a sequence of positive edges $e_{i:\lambda}$, $e_{ij:\lambda\mu}$ and infinite capacity edges $e_{ii:\lambda\lambda+1}$. Note that, by construction, the $e_{i:\lambda}$ edges either will be in the same block $B_{i:\gamma}$ in the simplified graph, or will be between a block and node 0 or node 1. Furthermore, the infinite capacity edges either will be in the same block, or there will be an edge $B_{i:\gamma} \rightarrow B_{j:\delta}$ in the simplified graph to represent them. Finally, if $e_{ij:\lambda\mu}$ is a positive edge, then, by construction of the simplified graph, there exists an edge $B_{i:\gamma} \rightarrow B_{j:\delta'}$ where $U_{i:\lambda} \in B_{i:\gamma}$ and $U_{j:\mu} \in B_{j:\delta}$ with $\delta' \leq \delta$. Hence, if there is an augmenting path in the Ishikawa graph, then there exists an augmenting path in the simplified graph. □

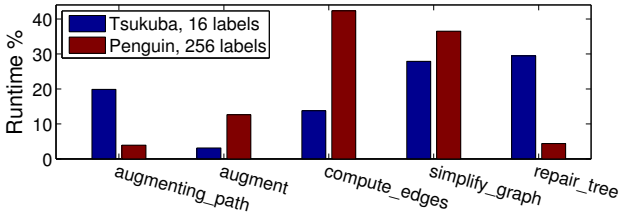

Figure 1: *Percentage of time taken by each subroutine. Note that, in Penguin, due to the large number of labels, the percentages of time spent on compute_edges and simplify_graph are high.*

2. Additional experiments

2.1. Runtime analysis for each subroutine

In Fig. 1, we report the percentage of time taken by each subroutine of our algorithm for the Tsukuba and Penguin instances. Note that the individual time complexities of the subroutines *compute_edges* and *simplify_graph* are $\mathcal{O}(\ell^3)$ and $\mathcal{O}(\ell^2)$, respectively. Therefore, they become dominant when the number of labels is large, and hence the corresponding percentages of time are high, particularly for Penguin.

2.2. Robust regularizer

Since robust regularizers are highly effective in computer vision, we tested our algorithm with the Huber loss function [1] as regularizer. The results are summarized in Table 1. In this experiment, the Huber value was set to 4 for Tsukuba, Venus and Sawtooth, 6 for Map, 20 for Cones and Teddy, 10 for KITTI, and 25 for Penguin and House. Note that, even in this case, our algorithm lets us solve much larger problems than the BK algorithm and EIBFS, and is an order of magnitude faster than state-of-the-art message-passing algorithms.

2.3. Parallelization

We parallelized our algorithm based on the dual-decomposition technique of [4] and evaluated it on the same

Problem	Memory [MB]				Time [s]			
	BK	EIBFS	TRWS	MEMF	BK	EIBFS	TRWS	MEMF
Tsukuba	1715	1385	287	**211**	8	**3**	198	28
Venus	3375	2719	638	**396**	17	**5**	211	57
Sawtooth	3348	2698	633	**393**	15	**4**	467	34
Map	2680	2116	494	**219**	22	**5**	>2953	36
Cones	*42155	*32167	5025	**1200**	-	-	1118	**363**
Teddy	*42155	*32167	5025	**1200**	-	-	6879	**2064**
KITTI	*42161	*32627	6416	**2215**	-	-	>30165	**18923**
Penguin	*33487	*25423	**215**	663	-	-	>50000	**6277**
House	*100494	*76295	**643**	1986	-	-	>50000	**8568**

Table 1: *Memory consumption and runtime comparison with state-of-the-art baselines. A "*" indicates a memory estimate, and ">" indicates that the algorithm did not converge to the optimum within the specified time. Note that our algorithm has a much lower memory consumption than the max-flow-based methods and is an order of magnitude faster than message-passing algorithms. Compared to EIBFS, our algorithm is 7 – 11 times slower, but requires 7 – 10 times less memory, which makes it applicable to more realistic problems. In all stereo problems, TRWS cached the pairwise potentials in an array for faster retrieval, but in the case of inpainting, it was not possible due to excessive memory requirement.*

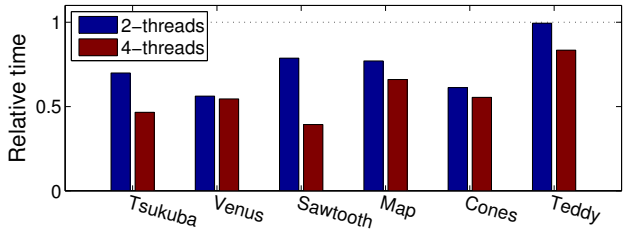

Figure 2: *Our algorithm can be accelerated using the parallel max-flow technique. The relative times ranged from 0.56 to 0.99 with 2-threads and from 0.39 to 0.83 with 4-threads. In Teddy, in the case of 2-threads, the multi-threaded algorithm performs almost the same as the single-threaded algorithm, which, we expect, is due to a bad image splitting strategy.*

six stereo instances from the Middlebury dataset [2, 3] as before. The relative times t_m/t_s, where t_m stands for the multi-thread time and t_s for the single-thread one, are shown in Fig. 2 for two and four threads. In this experiment, for all problems, the image grid was split vertically into two and four equally-sized blocks, respectively. Note that this spliting strategy is fairly arbitrary, and may affect the performance of the multi-threaded algorithm. In fact finding better splits may itself be a possible future direction.

References

[1] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. 1

[2] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002. 2

[3] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195. IEEE, 2003. 2

[4] P. Strandmark and F. Kahl. Parallel and distributed graph cuts by dual decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2085–2092. IEEE, 2010. 1