Minimal Component-Hypertrees

Alexandre Morimitsu, Wonder A. L. Alves, Dennis J. Silva, Charles F. Gobber, Ronaldo F. Hashimoto
alexandre.morimitsu@usp.br, ronaldo@usp.br

1. Introduction

Component trees [1]:

- Represent the hierarchy of connected components (CCs) of the level sets of a grayscale image;
- Efficiently represented by max-trees and widely used;

Component-hypertrees [2]:

- Extension of component trees using increasing connectivities;
- Efficient ways of updating hypertrees from previous connectivities have been presented [3];
- Goal: an efficient way of representing hypertrees.

2. Proposed Method

Suppose a grayscale image f and a sequence $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ of increasing sets of neighboring pixels are given. Then, the proposed algorithm follows the template below:

1. Initialize **parent** (array representing a max-tree);
2. Initialize the hypertree (empty at the beginning);
3. For $1 \leq i \leq n$:
   (a) For (p, q) neighbors in $\mathcal{A}_i$:
      i. Update **parent** by connecting p and q [4]. Track changes by marking nodes and arcs;
   (b) Update the hypertree by allocating marked nodes and arcs.

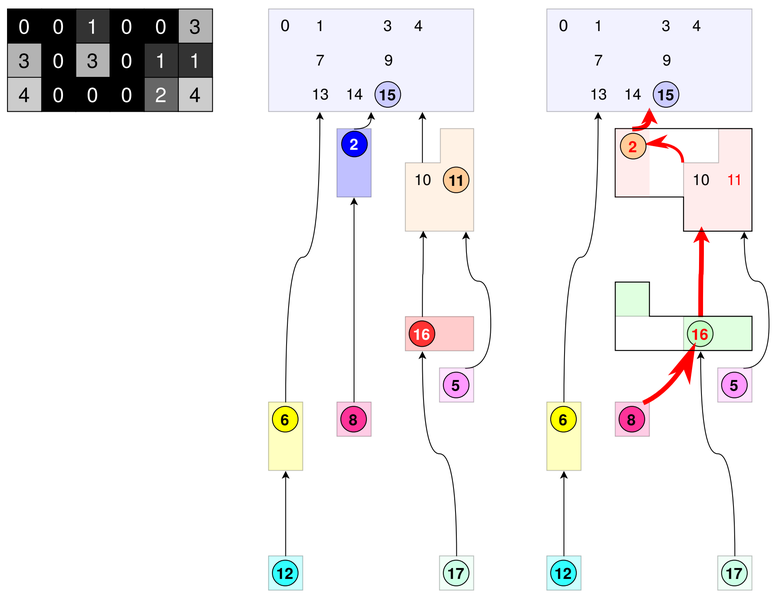

Figure 1: Left: a grayscale image; middle: its max-tree using 4-connectivity; right: updating the max-tree by connecting pixels $(8, 16)$. Nodes and arcs in red are marked.

3. Minimal Component-Hypertree

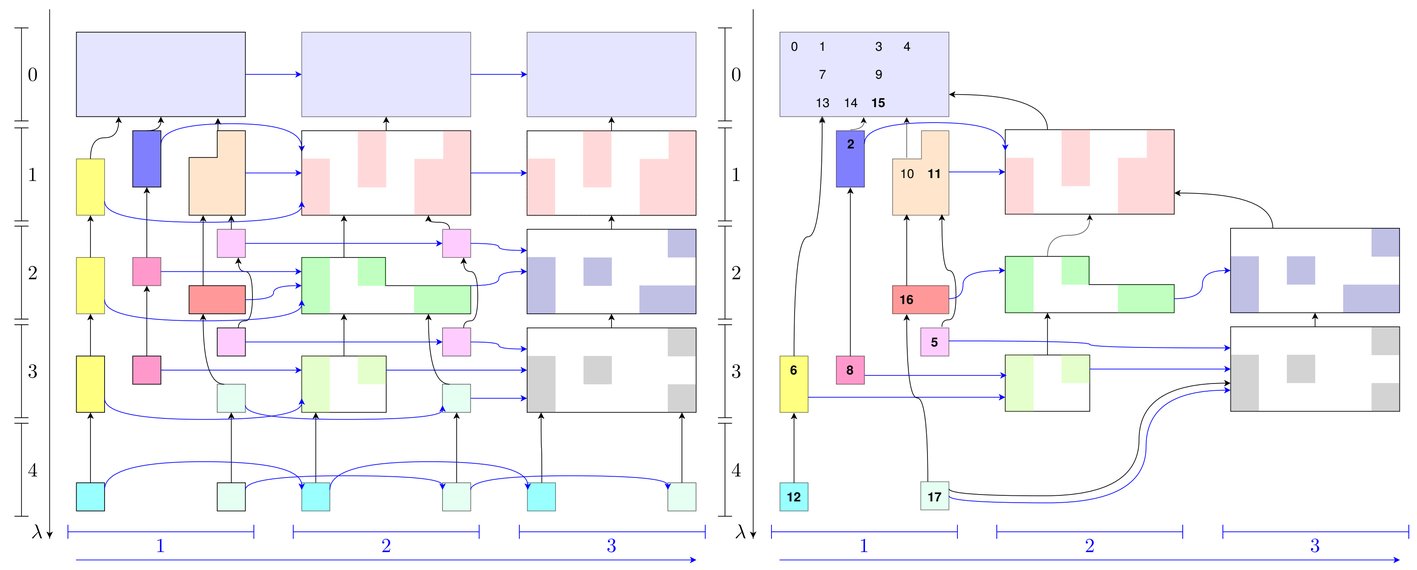

Figure 2: Left: complete hypertree; right: minimal component-hypertree. Numbers inside nodes represent stored pixels.

Minimal component-hypertree is the smallest graph satisfying:

1. No repeated nodes;
2. All inclusion relations;
3. Pixels stored only once;
4. Nodes do not depend on nodes with higher connectivity index to be reconstructed.

4. Results

Time consumption:

- Updating the max-tree is the most time consuming step;
- Only 3% to 6% of time used to allocate structures.

Number of nodes and arcs compared to other representations:

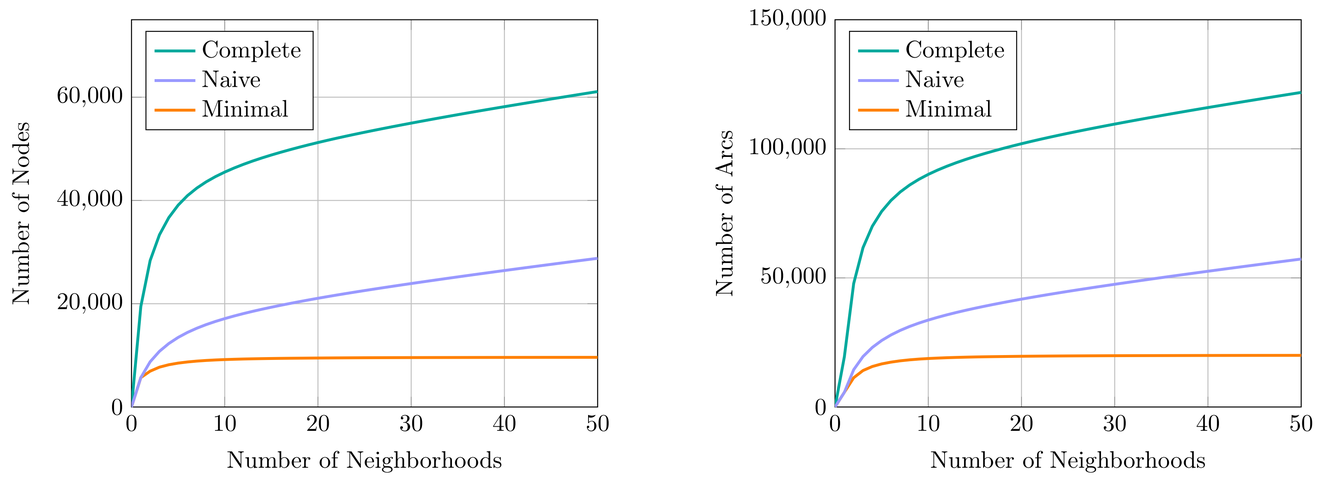

Figure 3: Comparison of number of nodes and arcs for 3 different representations of component-hypertrees. "Naive" refers to n max-trees built independently. Data was obtained from the Born Digital Dataset from the ICDAR [5].

5. Conclusion

- An efficient way of computing and storing hypertree was presented;
- Computation of attributes in this structure will be presented in a later date.

6. Acknowledgements

- This study was financed in part by the CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Finance Code 001);
- FAPESP - Fundação de Amparo a Pesquisa do Estado de São Paulo (Proc. 2018/15652-7);
- CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico (Proc. 428720/2018-8).

7. References

[1] Philippe Salembier, Albert Oliveras, and Luis Garrido. Antiextensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing*, 7(4):555–570, 1998.

[2] Nicolas Passat and Benoît Naegel. Component-hypertrees for image segmentation. In *ISMM*, volume 6671, pages 284–295. Springer, 2011.

[3] Alexandre Morimitsu, Wonder A. L. Alves, and Ronaldo F. Hashimoto. Incremental and efficient computation of families of component trees. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 681–692. Springer, 2015.

[4] Michael H. F. Wilkinson, Hui Gao, Wim H. Hesselink, Jan-Eppo Jonker, and Arnold Meijster. Concurrent computation of attribute filters on shared memory parallel machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1800–1813, 2008.

[5] Dimosthenis Karatzas, Sergi R. Mestre, Joan Mas, Farshad Nourbakhsh, and Partha P. Roy. ICDAR 2011 robust reading competition-challenge 1: reading text in born-digital images (web and email). In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1485–1490. IEEE, 2011.