Handling and computing hierarchies with graphs

Jean Cousty

ICIP tutorial on
Optimizations on Hierarchies of Partitions
2014

ESIEE ENGINEERING

UNIVERSITÉ PARIS-EST

Introduction

- Many kind of hierarchies of partitions are used in image processing
  - Quad-tree like [Bouman 94]
  - Watershed based [Beucher 94, Najman 96]
  - Binary partition tree [Salembier 00]
  - Energy based [Guigues 06]
  - Learning techniques based [Arbelaez 08]
  - etc.
- How do we efficiently compute and manage them?

Introduction

- Graphs are adapted for computerized procedures since they are:
  - adapted to discrete settings
  - generic structures
  - a well studied algorithmic field
- We present generic tools that can be used for computing and handling many of these hierarchies with graphs

Outline of the lecture

Graph settings

- We consider an *(undirected) graph* $G = (V, E)$, where V is a finite set and E is composed of unordered pairs of distinct elements in V, *i.e.*, E is a subset of $\{\{x, y\} \subseteq V \mid x \neq y\}$
- Each element of V is called a *vertex or a point (of G)*
- Each element of E is called an *edge* (of G)
- Subsequently, X denotes any subgraph of G; its vertex and edge sets are denoted by $V(X)$ and $E(X)$
- We consider the usual notions of a *connected graph* and of a *connected component of a graph*

Connected hierarchies

- A *partition of V is connected (for G)* if any of its regions is connected for G
- A *hierarchy on V is connected (for G)* if any of its partitions is connected
- We only consider connected hierarchies for G

Remark

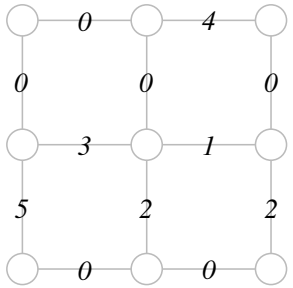
- *Any hierarchy on V is connected when G is complete*
- *Thus, any hierarchy can be treated with the proposed tools*

Quasi-flat zones hierarchy

- Significantly used as a scale space since the late 70's [Nagao 79, Meyer 99]
- Also know as the α-tree [Soille 08]
- Also used for Fuzzy Connectedness [Udupa 96]
- In fact, it is fundamental tool for handling any hierarchy

Quasi-flat zones hierarchy

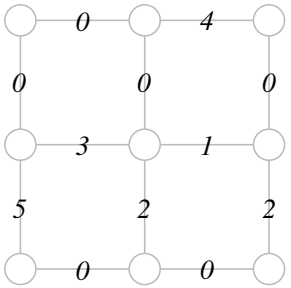
- Consider an *edge weighted graph* (G, w)
- w is a map from E to $\mathbb{R}^+$
- $w(u)$ is the *weight* of the edge u
- For image analysis,
  - G can be any pixel adjacency graph (*e.g.*, 4-adjacency graph)
  - w can be a gradient of intensity (*e.g.*, $w(\{x, y\} = |I(x) - I(y)|$, where I is an image)



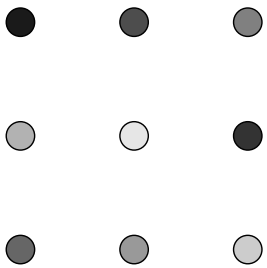
(G, w)

Quasi-flat zones hierarchy

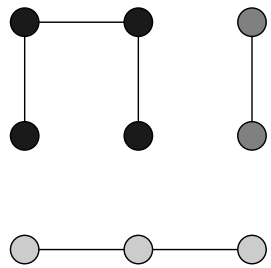
- For any subgraph X of G
- the *λ-level set of X* is the set of all edges of X of weight less than λ
- the *λ-level graph of X* is the subgraph of X whose edge set is the λ-level set of X and whose vertex set is the one of X



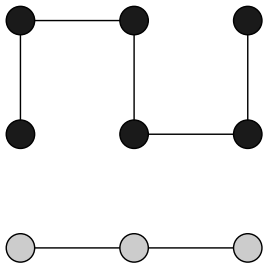
(G, w)



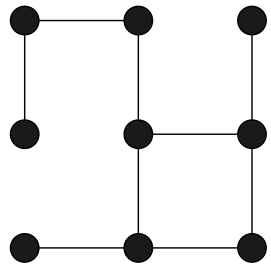
0-level graph



1-level graph



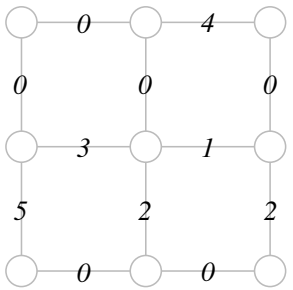
2-level graph

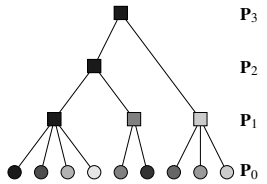


3-level graph

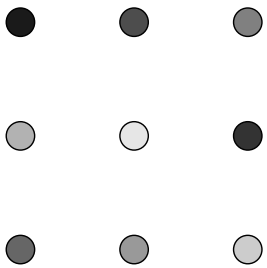
Quasi-flat zones hierarchy

- The connected components partition of the λ-level graph of X is called the *λ-level partition of X*
- The sequence $\mathcal{Q}_X(w)$ of all λ-level partitions of X is a hierarchy called the *quasi-flat zones hierarchy of X*

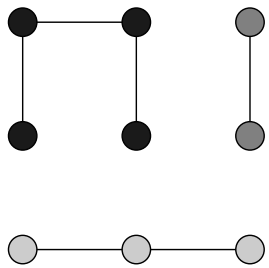




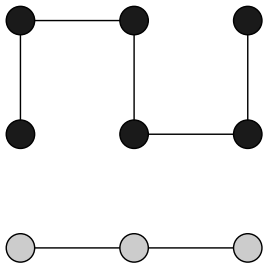
(G, w) and $\mathcal{Q}_G(w)$



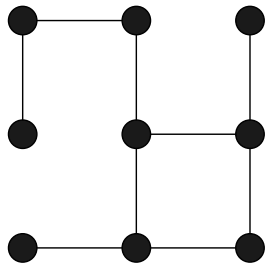
0-level graph



1-level graph



2-level graph

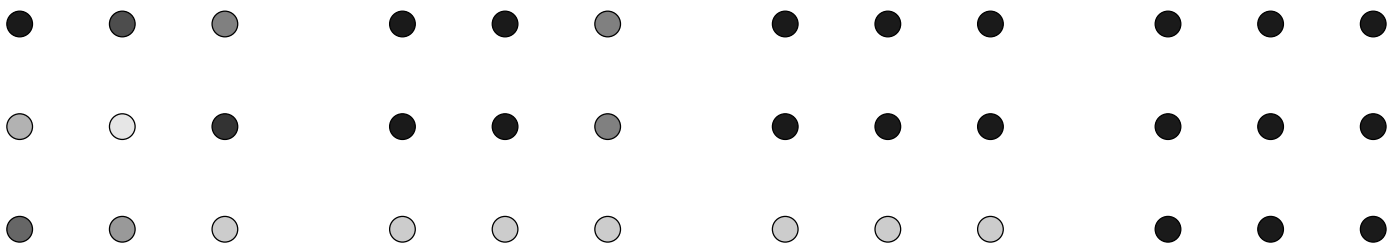


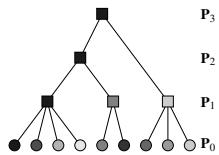
3-level graph

Saliency map: the other way round

- The *cut of a partition* $\mathbf{P}$ *of* V, denoted by $\phi(\mathbf{P})$, is the set of edges of G made of two vertices in different classes of $\mathbf{P}$:

$$\phi(\mathbf{P}) = \{\{x, y\} \in E(G) \mid \mathbf{P}_x \neq \mathbf{P}_y\}.$$



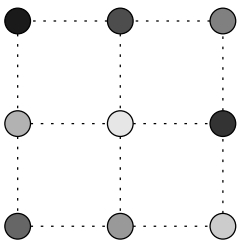


$\mathcal{H}$

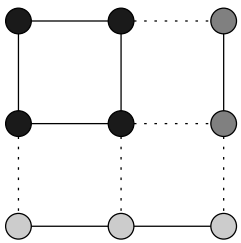
Saliency map: the other way round

- The *cut of a partition* $\mathbf{P}$ *of* V, denoted by $\phi(\mathbf{P})$, is the set of edges of G made of two vertices in different classes of $\mathbf{P}$:

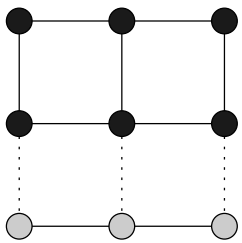
$$\phi(\mathbf{P}) = \{\{x, y\} \in E(G) \mid \mathbf{P}_x \neq \mathbf{P}_y\}.$$



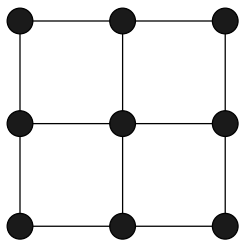
$\phi(\mathbf{P}_0)$



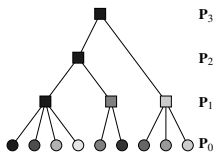
$\phi(\mathbf{P}_1)$



$\phi(\mathbf{P}_2)$



$\phi(\mathbf{P}_3)$



$\mathcal{H}$

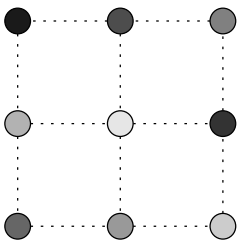
Saliency map: the other way round

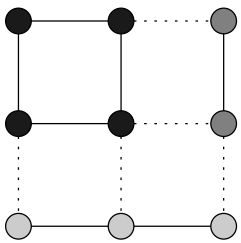
- The *cut of a partition* $\mathbf{P}$ *of* V, denoted by $\phi(\mathbf{P})$, is the set of edges of G made of two vertices in different classes of $\mathbf{P}$:

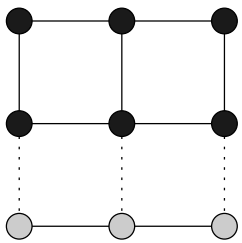
$$\phi(\mathbf{P}) = \{\{x, y\} \in E(G) \mid \mathbf{P}_x \neq \mathbf{P}_y\}.$$

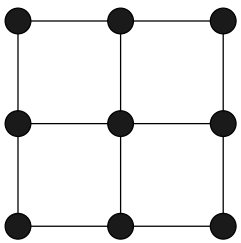
- The *saliency map of a hierarchy* $\mathcal{H} = (\mathbf{P}_0, \ldots, \mathbf{P}_\ell)$ *on* V is the map $\Phi(\mathcal{H})$ such that the weight of u for $\Phi(\mathcal{H})$ is the maximum value λ such that u belongs to the cut of $\mathbf{P}_\lambda$:

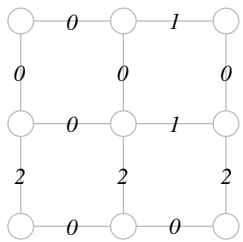
$$\Phi(\mathcal{H})(u) = \max\{\lambda \in \{0, \ldots, \ell\} \mid u \in \phi(\mathbf{P}_\lambda)\}.$$


$\phi(\mathbf{P}_0)$


$\phi(\mathbf{P}_1)$


$\phi(\mathbf{P}_2)$


$\phi(\mathbf{P}_3)$

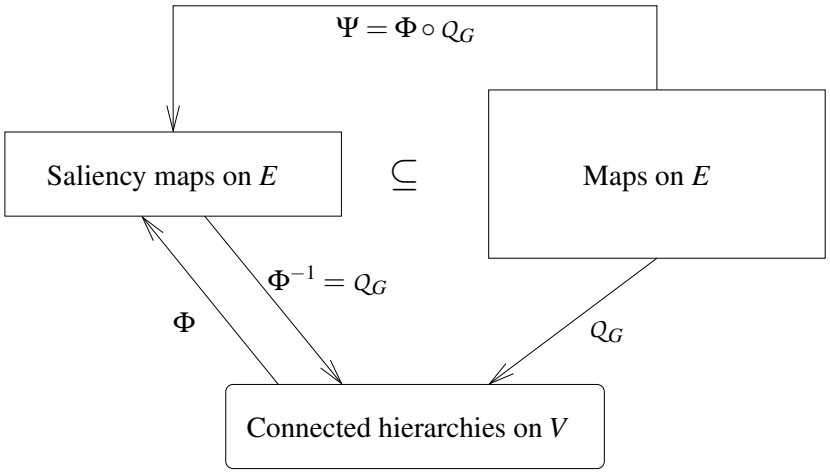


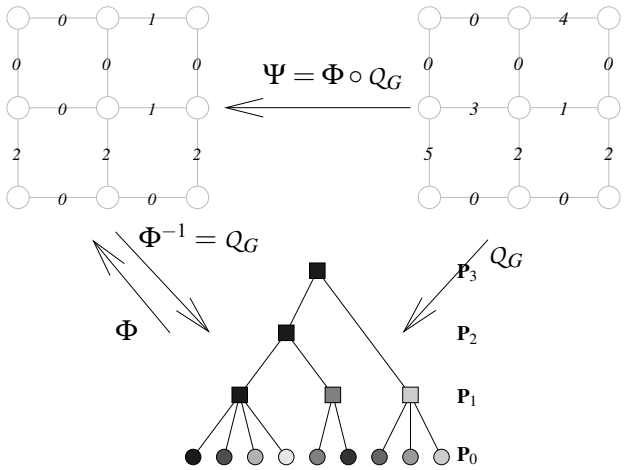
$\Phi(\mathcal{H})$

Hierarchy: equivalent representations

Theorem

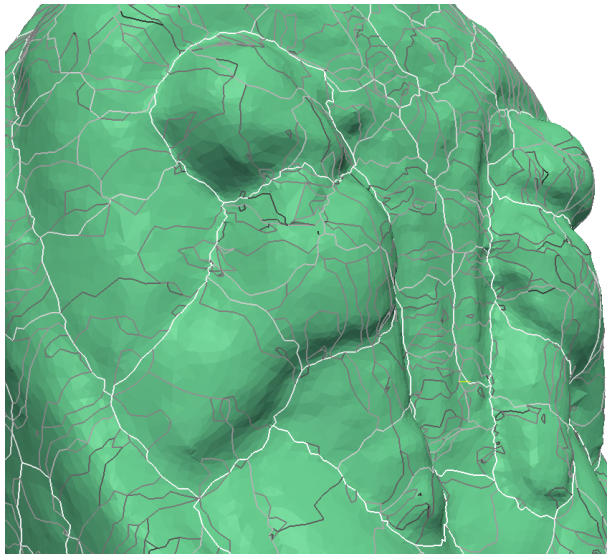
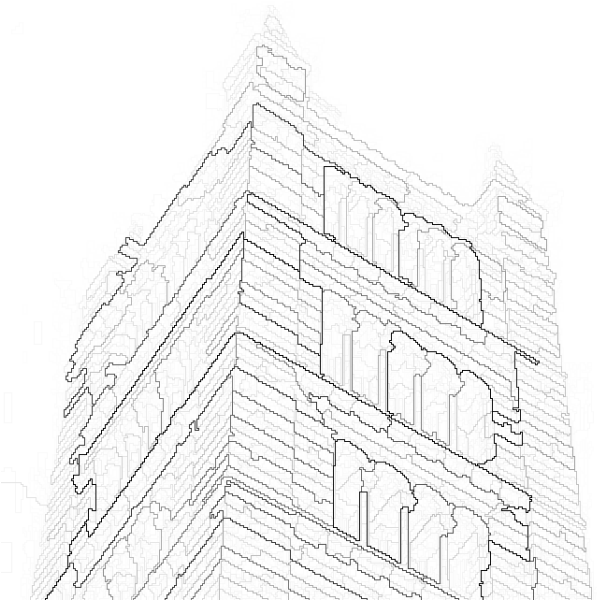
The map Φ is a one-to-one correspondence between the connected hierarchies on V (of depth $|E|$) and the saliency maps (of range $\{0, \ldots, |E|\}$). The inverse Φ^{-1} of Φ associates to any saliency map w its quasi-flat zones hierarchy: $\Phi^{-1}(w) = \mathcal{Q}_G(w)$.





Saliency map: practical interests

- Provide a compact visualization of a hierarchy
  - when the graph has some particular property (*e.g.*, planarity)
- Handle a hierarchy differently: a function rather than a tree
  - [Arbelaez 08] computes saliency maps of gPb contour detector


Problem: can we get a more compact representation?

Question

(P) Given an edge-weighted graph $((V, E), w)$, find a minimal set $E' \subseteq E$ of edges such that the quasi-flat zones hierarchies of (V, E) and of (V, E') are the same?

Solution to (P) : minimum spanning tree (MST)

Theorem

A subgraph X of G is an MST of (G, w) if and only if the two following statements hold true:

1. *the quasi-flat zones hierarchies of X and of G are the same; and*
2. *the graph X is minimal for proposition 1, i.e., for any subgraph Y of X, if the quasi-flat zones hierarchy of Y for w is the one of G for w, then we have $Y = X$.*

Consequences

- An MST is enough to handle a connected hierarchy
- Quasi-flat zones hierarchy can be computed with MST algorithms
  - Kruskal proposed a quasi-linear time algorithm, provided a sorting of the edges

Kruskal algorithm for MST (High-Level View)

1. create a forest $\mathcal{F}$ (a set of trees), where each vertex in the graph is a separate tree
2. create a set S containing all the edges in the graph
3. while S is nonempty and $\mathcal{F}$ is not yet a single tree
   4. remove an edge with minimum weight from S
   5. if that edge connects two different trees, then add it to the forest, combining two trees into a single tree
   6. otherwise discard that edge.

- At the termination of the algorithm, the forest has only one component and forms a minimum spanning tree of the graph
- The **if** test at line 5 is the time-complexity bottleneck. It can be tested thanks to *disjoint sets* management algorithms

The *disjoint set* problem

The disjoint set problem consists in maintaining a collection $\mathcal{Q}$ of disjoint sets under the operation of union.
Each set Q in $\mathcal{Q}$ is represented by a unique element of Q, called the *canonical element*.

- `MakeSet`(q_1)
- `FindCanonical`(q_1)
- `Union`(q_1, q_2)

Kruskal algorithm for MST (Implementation)

Data: An edge-weighted graph $((V, E), w)$.
Result: A minimum spanning tree MST
Result: A collection $\mathcal{Q}$

```
// Collection Q is initialized to ∅
1 e := 0;
2 for all x_i ∈ V do MakeSet(i);
3 for all edges {x, y} by (strict) increasing weight w({x, y}) do
4 |   c_x := Q.FindCanonical(x); c_y := Q.FindCanonical(y);
5 |   if c_x ≠ c_y then
6 |   |   Q.Union(c_x, c_y);
7 |   |   MST[e] := {x, y}; e := e + 1;
```

Main question in Kruskal implementation

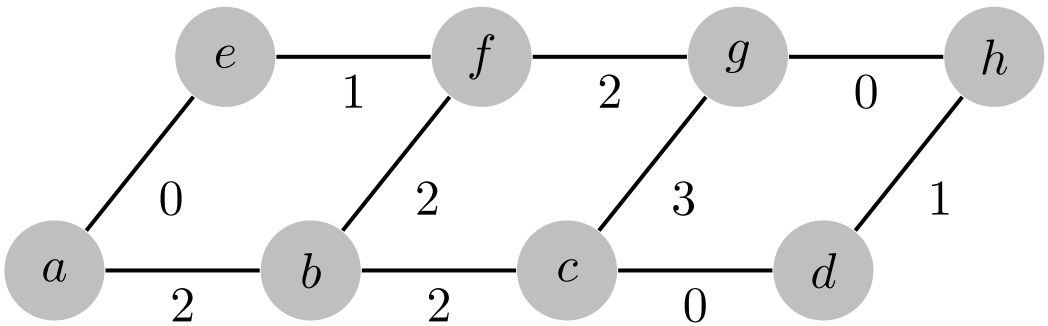
Question

- How to represent and implement the collection $\mathcal{Q}$?

Answer

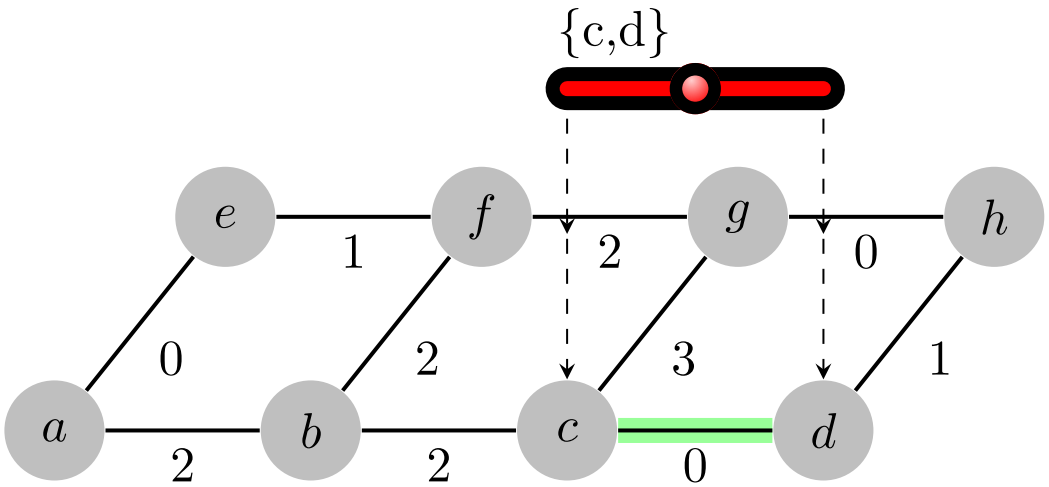
- A good representation for $\mathcal{Q}$ is as a set of trees

Binary Partition Tree by altitude ordering



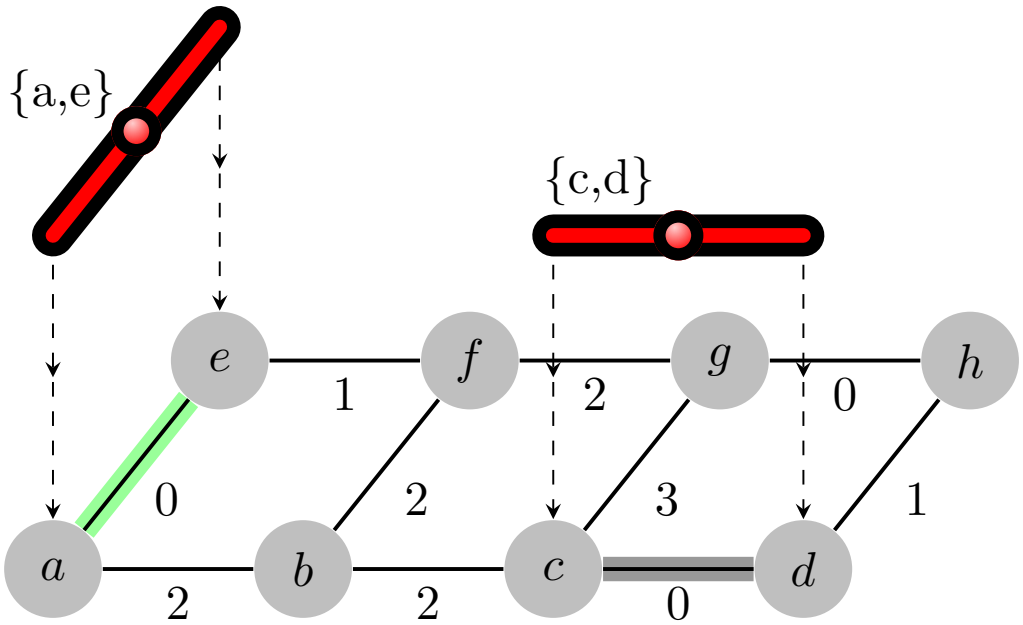
Edge-weighted graph

Binary Partition Tree by altitude ordering



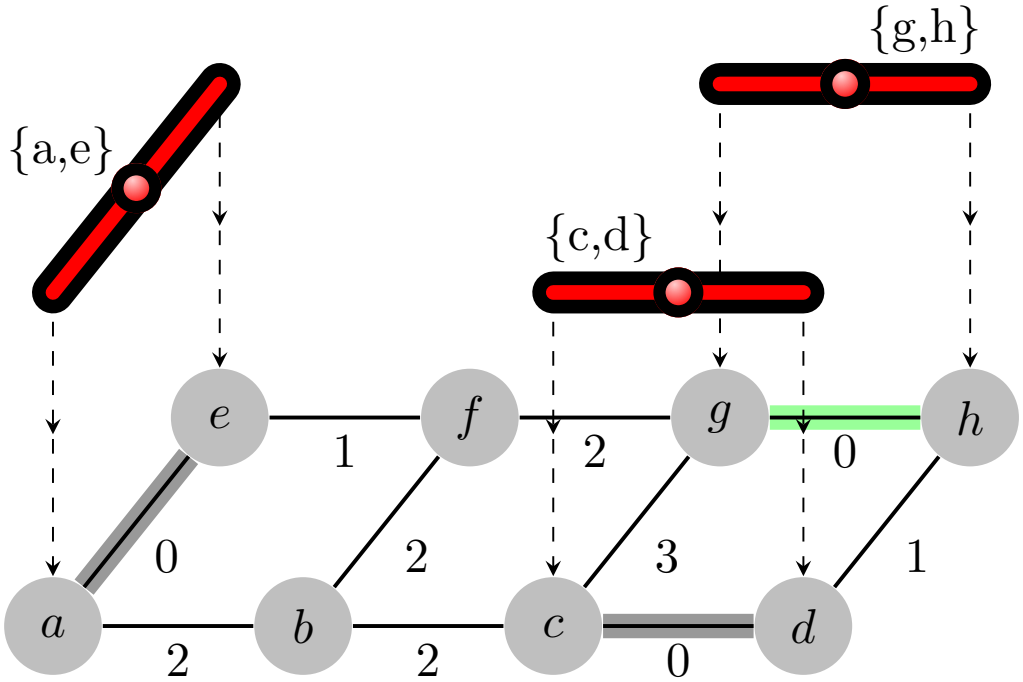
First edge-node

Binary Partition Tree by altitude ordering



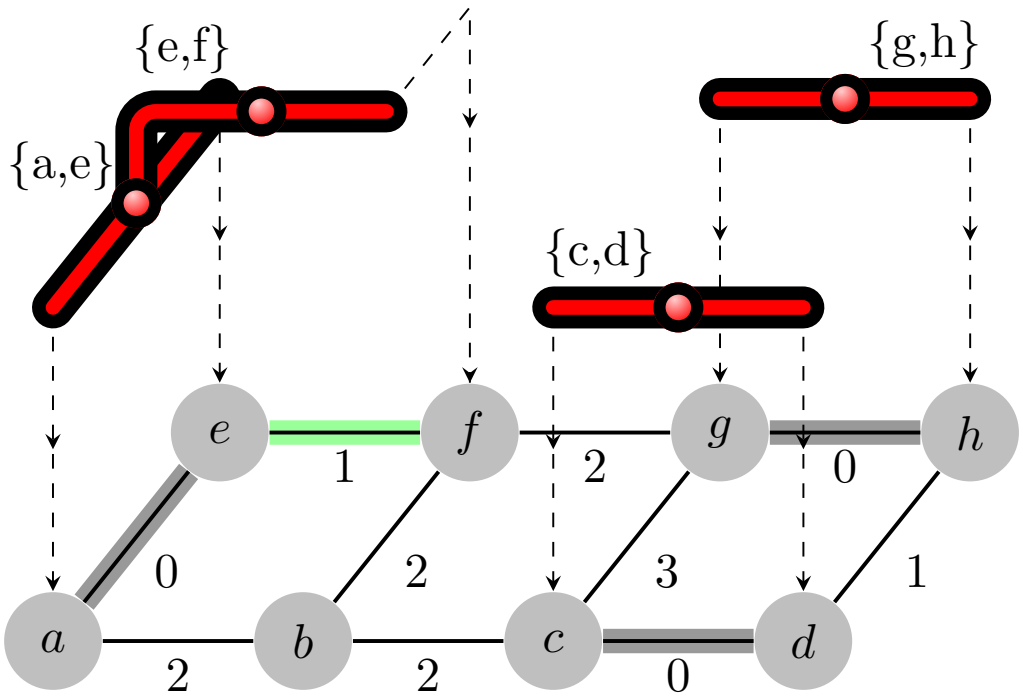
Second edge-node

Binary Partition Tree by altitude ordering



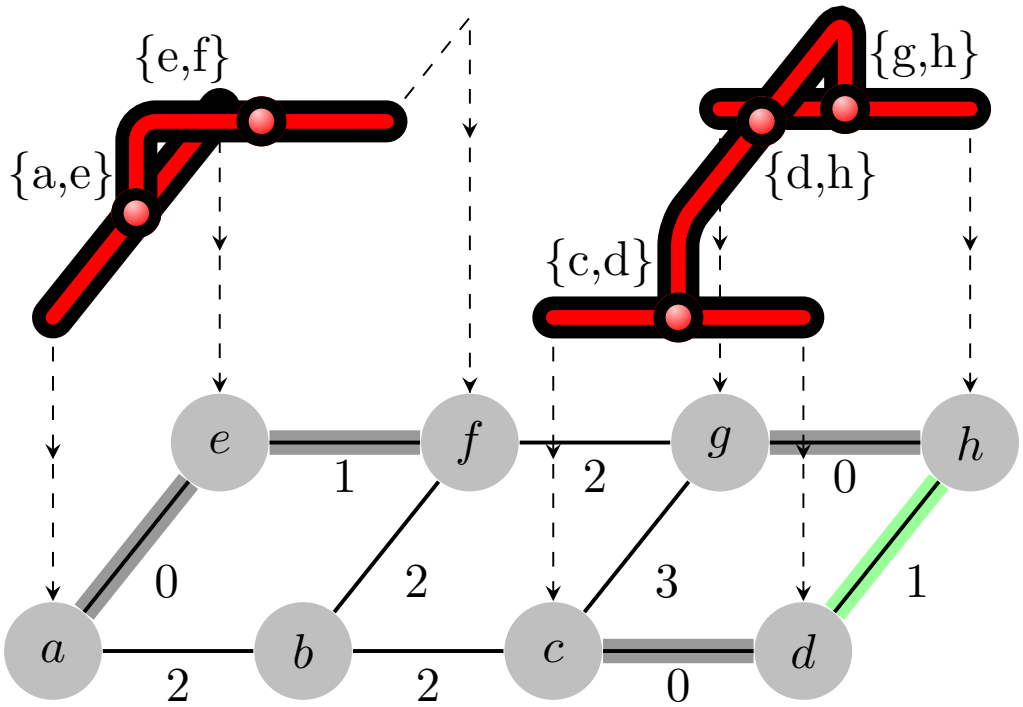
Third edge-node

Binary Partition Tree by altitude ordering



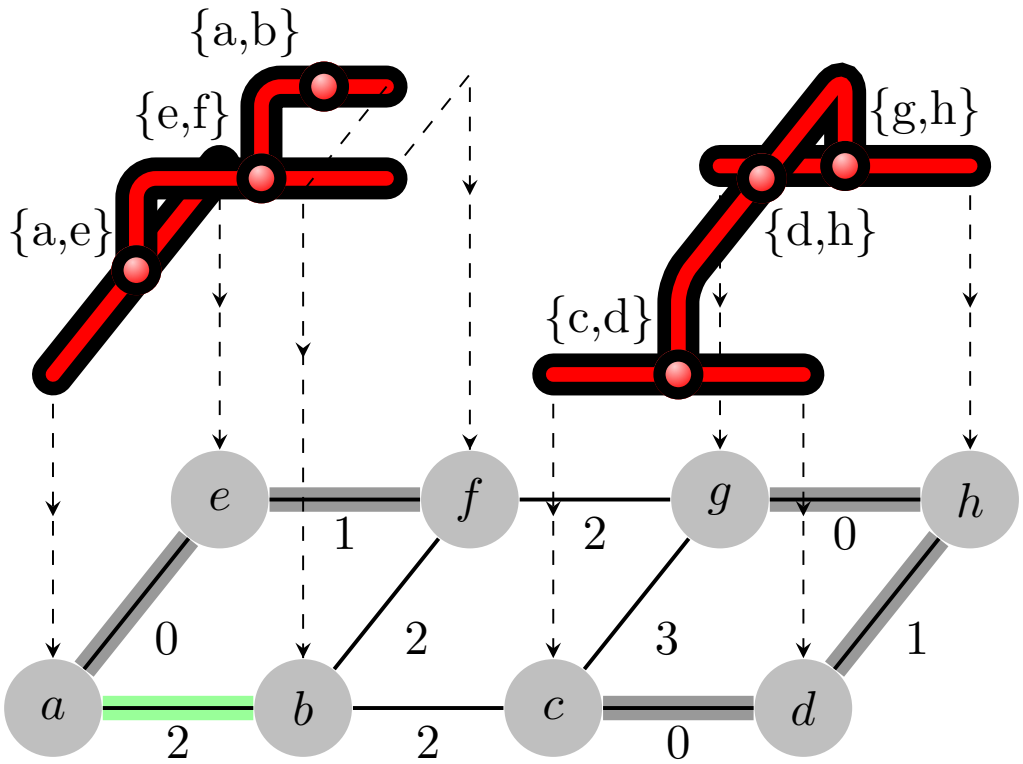
Fourth edge-node

Binary Partition Tree by altitude ordering



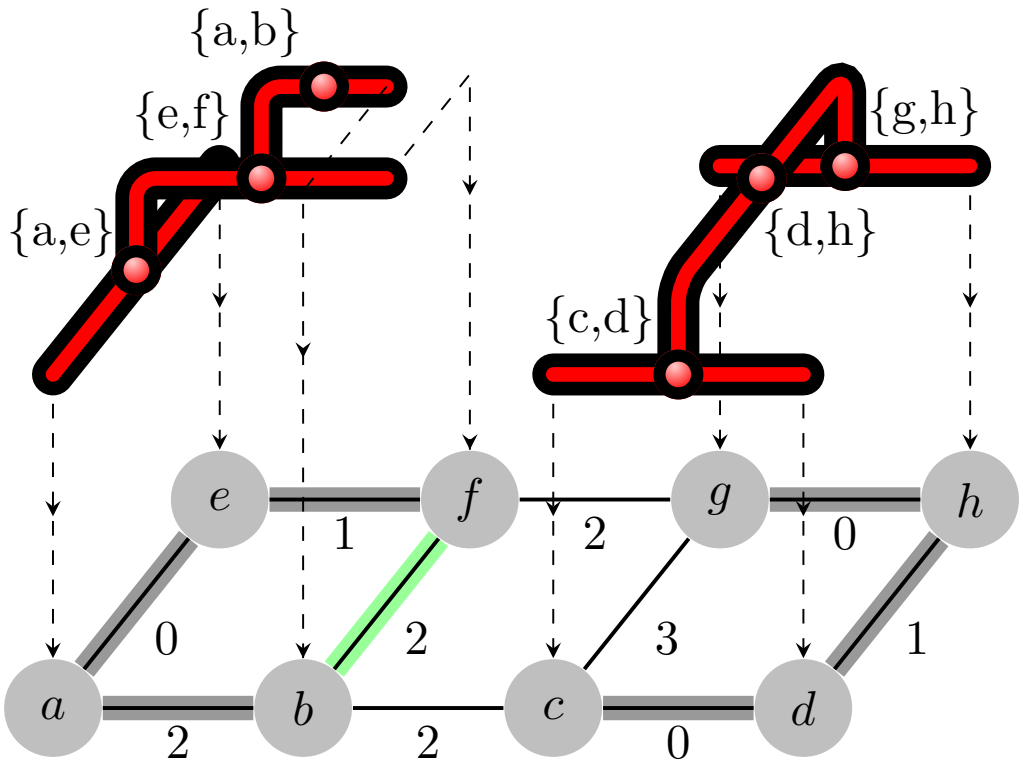
Fifth edge-node

Binary Partition Tree by altitude ordering



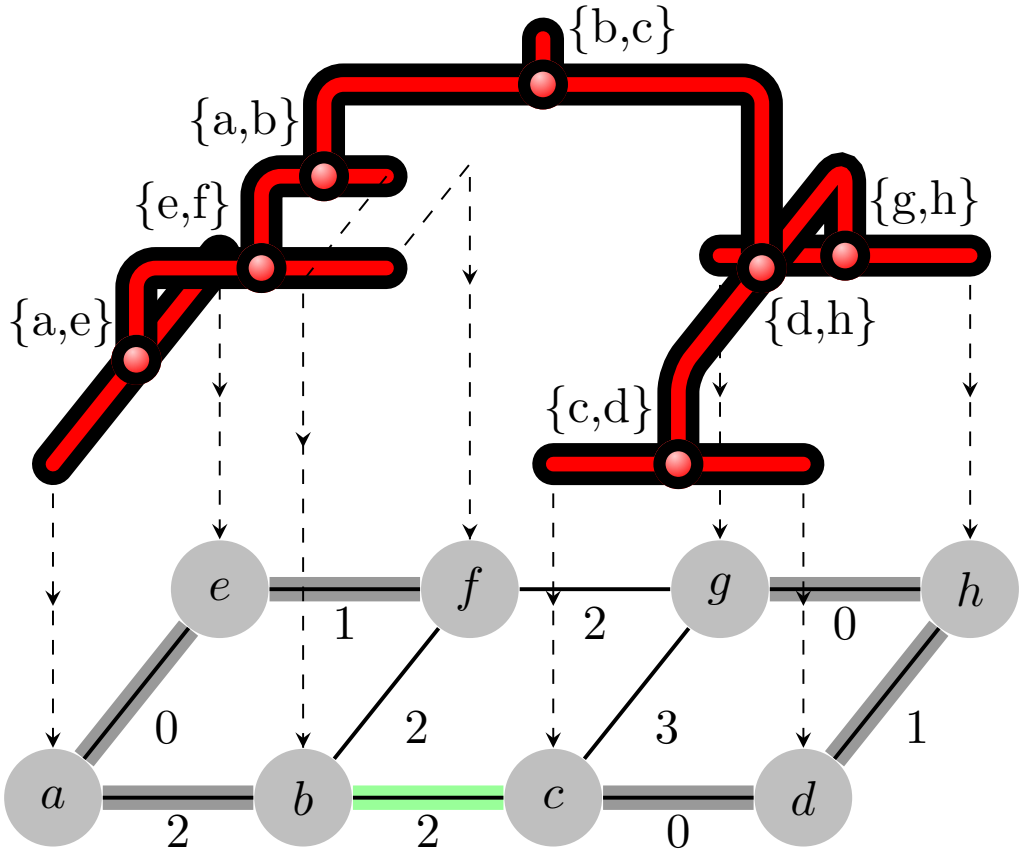
Sixth edge-node

Binary Partition Tree by altitude ordering



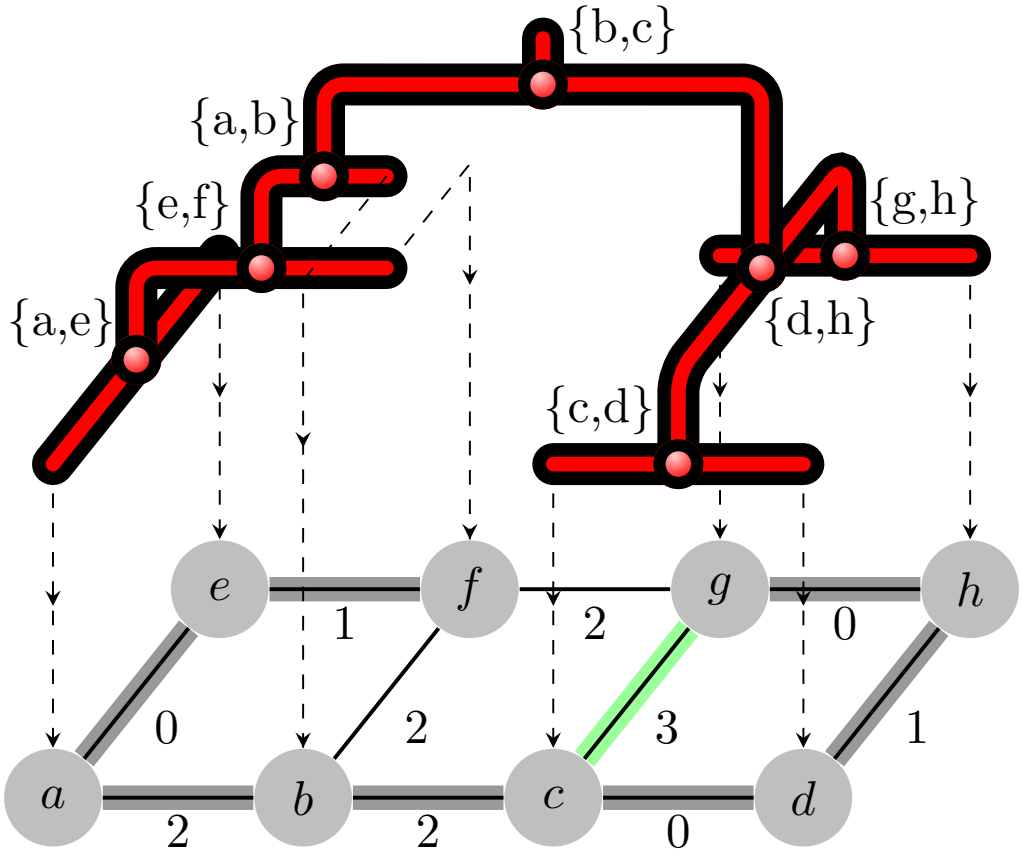
No new node

Binary Partition Tree by altitude ordering



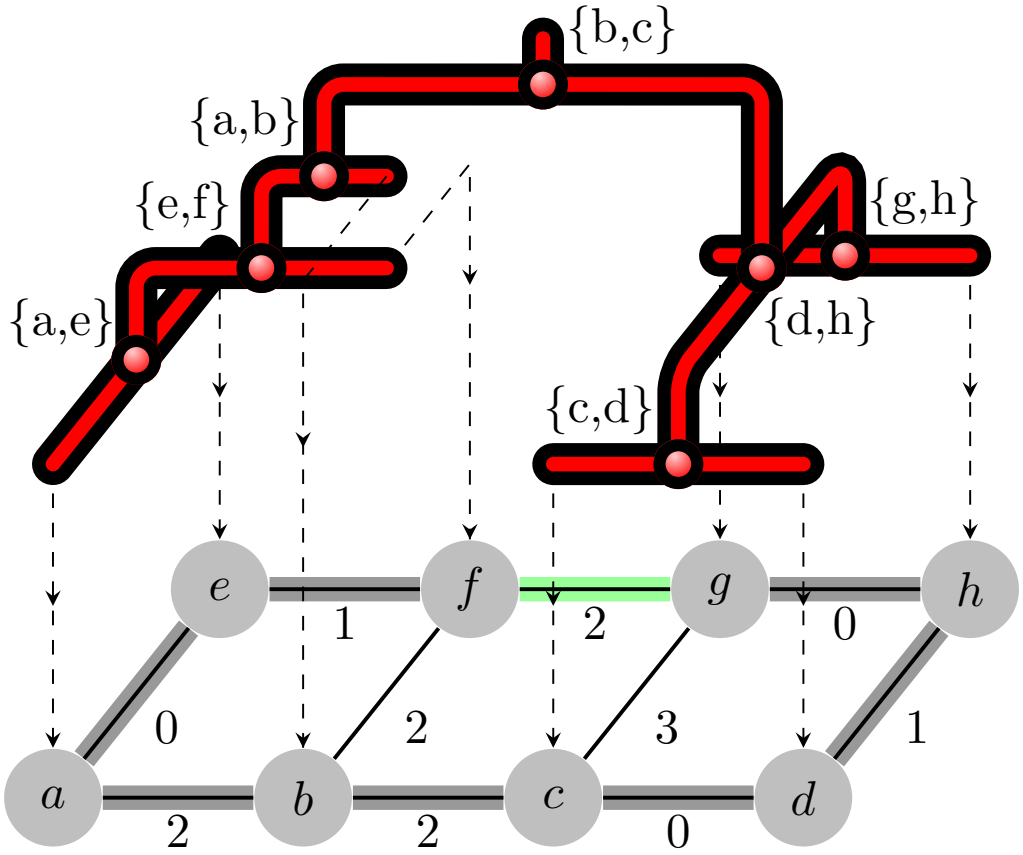
Seventh edge-node

Binary Partition Tree by altitude ordering



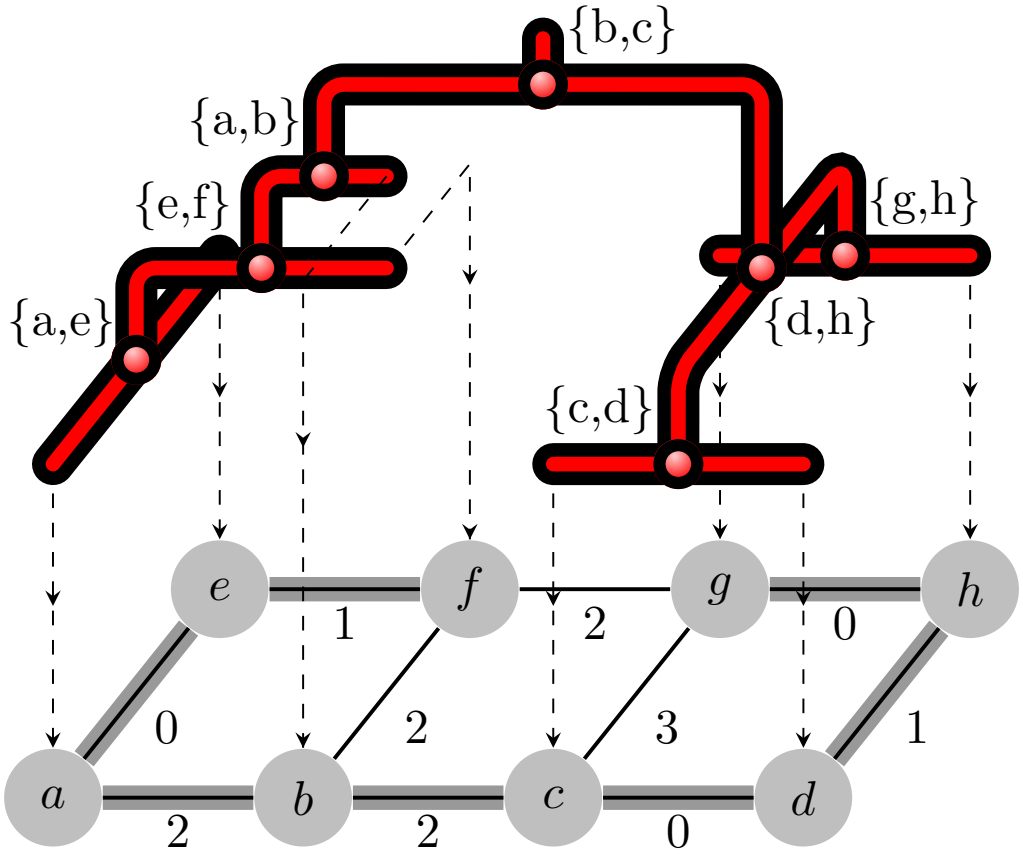
No new node

Binary Partition Tree by altitude ordering



No new node

Binary Partition Tree by altitude ordering



Final Q_{BT}

Q_{BT} Union-Find

Procedure Q_{BT}.MakeSet(q)

1 Q_{BT}.parent[q] := -1; Q_{BT}.size += 1;

Function Q_{BT}.FindCanonical(q)

1 **while** Q_{BT}.*parent[q]*≥ 0 **do** q :=Q_{BT}.parent[q];
2 **return** q;

Function Q_{BT}.Union(c_x, c_y)

1 Q_{BT}.parent[c_x]:=Q_{BT}.size; Q_{BT}.parent[c_y]:=Q_{BT}.size;
2 Q_{BT}.MakeSet(Q_{BT}.size);
3 **return** Q_{BT}.*size-1*;

Q_{BT} Union-Find

Interest

- The produced tree is useful
- It is a Binary Partition Tree (BPT) by altitude
  - BPT are often used in hierarchical segmentation [Salembier 00]

Drawback

The algorithm is slow: $O(|V|^2)$

Tarjan Union-Find

Interest

- Quasi-linear complexity

Drawback

- The produced tree is not useful for our purpose

Tarjan Union-Find

Procedure Q_T.MakeSet(q)

```
1 Q_T.parent[Q_T.size] := −1;  Q_T.Rnk[Q_T.size] := 0; Q_T.size += 1;
```

Function Q_T.FindCanonical(q)

```
1 r := q;
2 while Q_T.parent[r] ≥ 0 do r :=Q_T.parent[r];
3 while Q_T.parent[q] ≥ 0 do
4   tmp := q; q :=Q_T.parent[q]; Q_T.parent[tmp] := r
```

Function Q_T.Union(c_x, c_y)

```
1 if (Q_T.Rnk[c_x] >Q_T.Rnk[c_y]) then swap(c_x, c_y);
2 if (Q_T.Rnk[c_x] == Q_T.Rnk[c_y]) then Q_T.Rnk[c_y] += 1;
3 Q_T.parent[c_x] := c_y;
4 return c_y;
```

Q_{EBT}: Efficient Q_{BT} Union-Find

Interest

- Combination of both Q_{BT} and Q_T.
- Quasi-linear time complexity with respect to $|E| + |V|$
- One of the produced trees, Q_{BT}, is useful.

Q_{EBT}: Efficient Q_{BT} Union-Find

Procedure Q_{EBT}.MakeSet(q)

1 Q_{EBT}.Root[q]:=q; Q_{BT}.MakeSet(q); Q_T.MakeSet(q);

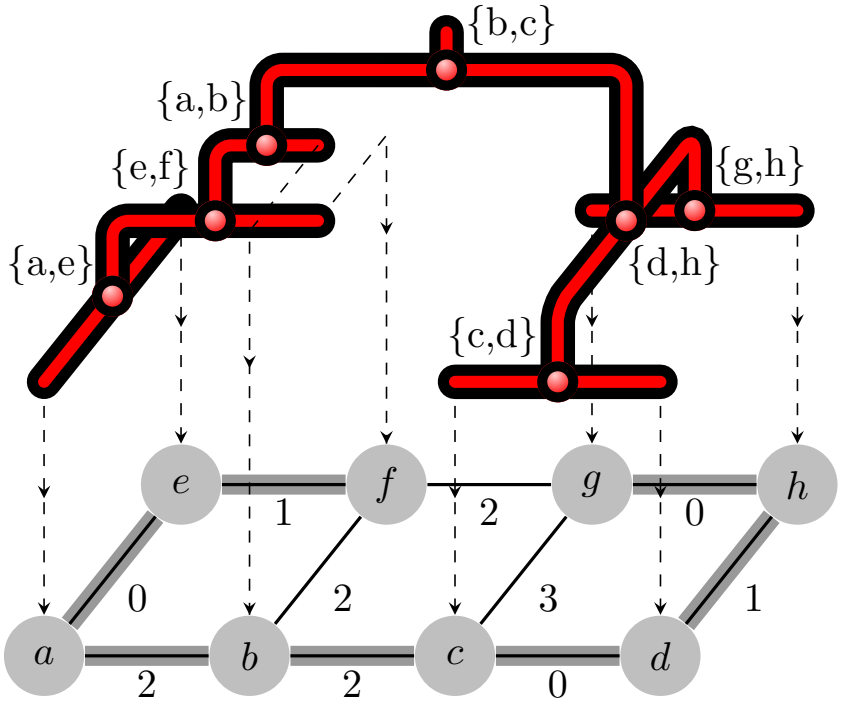
Function Q_{EBT}.Union(c_x, c_y)

1 t_u:=Q_{EBT}.Root[c_x]; t_v := Q_{EBT}.Root[c_y];
2 Q_{BT}.parent[t_u] := Q_{BT}.parent[t_v] := Q_{BT}.size;
3 Q_{BT}.children[Q_{BT}.size].add($\{t_u\}$);
4 Q_{BT}.children[Q_{BT}.size].add($\{t_v\}$);
5 c:=Q_T.Union(c_x,c_y); `// Union in` Q_T `(with compression)`
6 Q_{EBT}.Root[c] := Q_{BT}.size; `// Update the root of` Q_{EBT}
7 Q_{BT}.MakeSet(Q_{BT}.size);
8 **return** Q_{BT}.*size-1*;

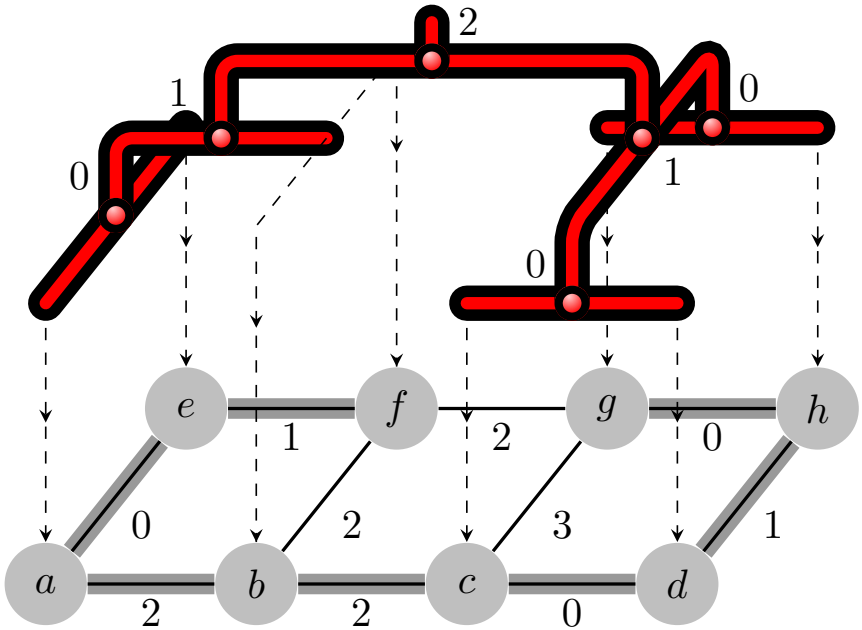
Function Q_{EBT}.FindCanonical(q)

1 **return** Q_T.*FindCanonical(q)*;

Q_{CT}: Quasi-flat zones hierarchy



Q_{BT}



Q_{CT}

Some helper functions

Function getEdge(n)

Data: a (non-leaf) node n of Q_{BT}
Result: the edge e of the MST corresponding to the n^{th} node

```
return n − |V|;
```

Function weightNode(n)

Data: a (non-leaf) node of the tree
Result: the weight of the MST edge associated with the n^{th} node of Q_{BT}

```
return w(MST[getEdge(n)]);
```

Quasi-flat zones hierarchy

Procedure CanonizeQ_{BT}

Data: Q_{BT}
Result: Q_{CT}, a canonized version of Q_{BT}

```
1 for all nodes n of Q_BT do Q_CT.parent[n]:=Q_BT.parent[n]; Q_CT.size+=1;
2 for each non-leaf and non-root node n of Q_BT by decreasing order do
3 |   p := Q_CT.parent[n];
4 |   if (weightNode(p) == weightNode(n)) then
5 |   |   for all c ∈ Q_BT.children[n] do  Q_CT.parent[c]:=p ;
6 |   |   Q_CT.parent[n]:=n; // Delete node n of Q_CT
  // If needed, build the list of children:
7 for all nodes n of Q_CT do
8 |   p:=Q_CT.parent[n]; if p ≥ 0 and p≠ n then Q_CT.children[p].add(n);
```

Quasi-flat zones hierarchy

Complexity analysis

- Q_{CT} is the quasi-flat zone hierarchy of (G, w)
- The time-complexity of CanonizeQ_{BT} is linear with respect to $|V|$
- Thus, the overall time-complexity for computing the quasi-flat zones hierarchy of (G, w) is quasi-linear with respect to $|E| + |V|$, provided a sorting of the edges of G

Q_{BT} or Q_{CT}?

- It is possible to merge the Q_{BT} and Q_{CT} computation steps in order to obtain Q_{CT} in one step
- Q_{BT} contains more information than Q_{CT}

Quasi-linear time saliency map algorithm

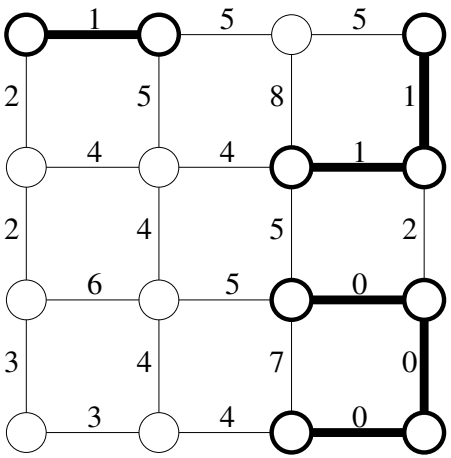
1. Given a weighted graph (G, w)
2. Compute the quasi-flat zones hierarchy Q_{CT} of (G, w)
3. For each edge $\{x, y\}$ of G
    4. Set $\Phi(\{x, y\})$ to the weight of the lowest common ancestor of $\{x\}$ and $\{y\}$ in the tree Q_{CT}

Property

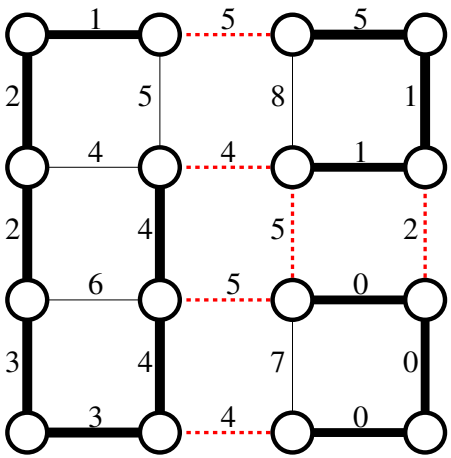
- Φ *is the* ultrametric opening / ultrametric contour map of w *(i.e., the saliency map of the quasi-flat zones hierarchy of* w*)*
- *Step 4 can be done in constant time [Bender 08] provided a linear time preprocessing*
- *The overall time complexity is quasi-linear with respect to* $|V| + |E|$

Watershed cuts/minimum spanning forests

- Partitions defined thanks to the *drop of water* principle [Cousty 09]
- Or equivalently thanks to
  - minimum spanning forests relative to the regional minima of w
- A *minimum spanning forest (MSF) rooted in a subgraph X* of G is a minimum weight spanning subgraph Y such that each connected component of Y includes exactly one connected component of X



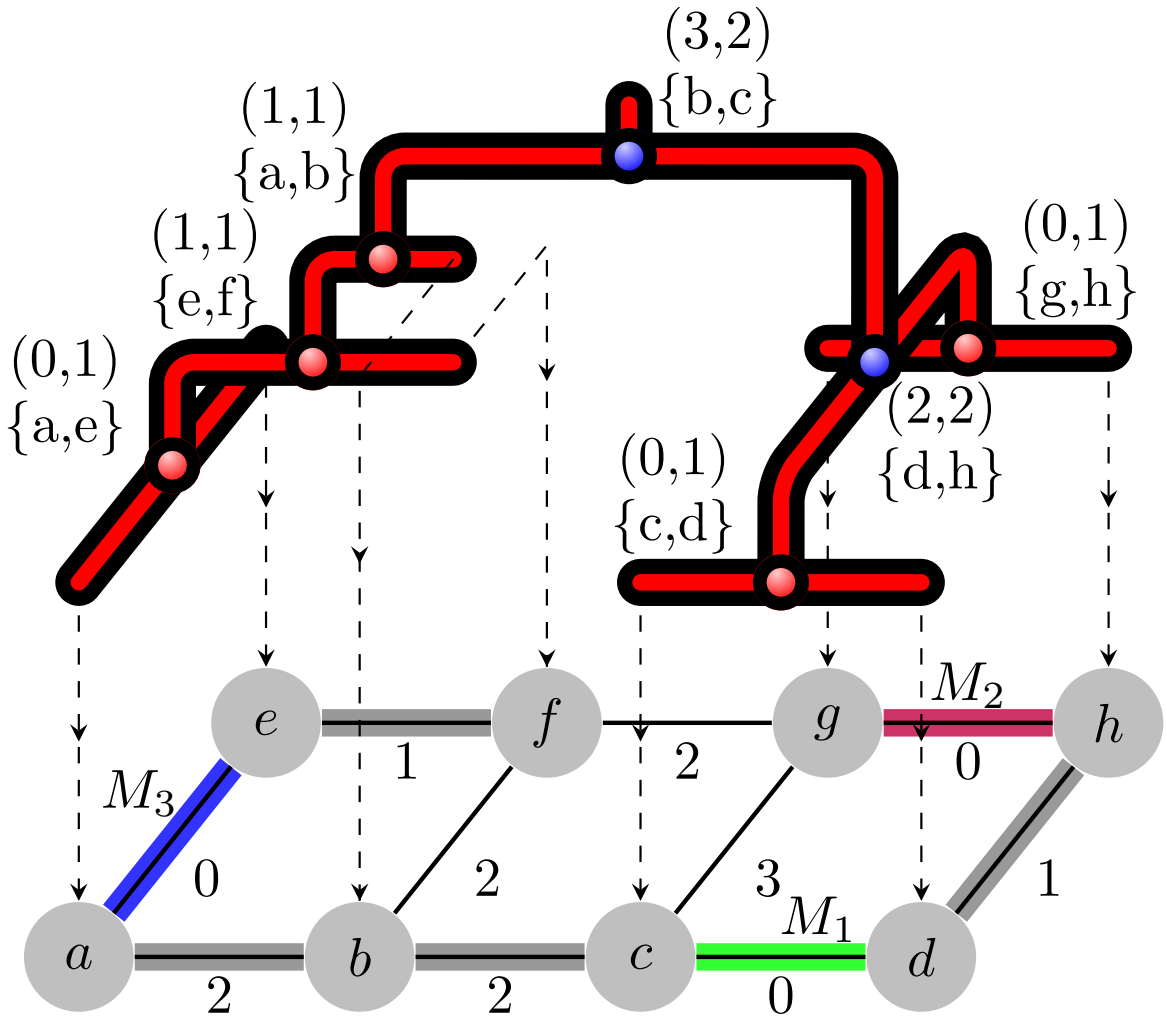
Minima in bold



MSF in bold &

Watershed cut in red

Watershed cuts from Q_{BT}: intuition



Watershed cuts from Q_{BT}

Function watershed

Data: Q_{BT}

Result: A binary array *ws* indicating which MST edges are watershed

```
1  for all leaf-nodes n of Q_BT do minima[n]:=0;
2  for each non-leaf node n of Q_BT by increasing order do
3      flag := TRUE; nb := 0;
4      for all c ∈ Q_BT.children[n] do
5          m := minima[c]; nb := nb + m;
6          if (m == 0) then flag := FALSE;
7      ws[getEdge(n)] := flag;
8      if (nb ≠ 0) then minima[n] := nb;
9      else
10         if (n is the root of Q_BT) then minima[n] := 1;
11         else
12             p := Q_BT.parent[n];
13             if (weightNode[n]<weightNode[p]) then minima[n] := 1;
14             else minima[n]:=0 ;
```

Watershed cuts from Q_{BT}

Property

- *The connected component partition induced by the set of edges* $\{u \in E \mid ws[u] = TRUE\}$ *flaged true by the algorithm is a watershed cut of* w
- *The time complexity of watershed procedure is linear with respect to* $|V|$

- A first hierarchical watershed is obtained by
  - setting to 0 the edges flaged true by the algorithm and keeping the weight of all other edges
  - considering the quasi-flat zones hierarchy of this new weight map
- Other interesting hierarchical watersheds can be designed

Hierarchical watersheds/MSFs

- Let $\prec = \langle M_1, \ldots, M_\ell \rangle$ be the sequence of regional minima of w ranked by importance according to some given criterion
- A *hierarchical watershed for* $\prec$ is a hierarchy $\langle \mathbf{P}_0, \ldots, \mathbf{P}_\ell \rangle$ of partitions such that, for any $i \in [0, \ell]$:
  - $\mathbf{P}_i$ is the connected component partition of an MSF rooted in the minima ranked after i

Remark

- *Every partition of a hierarchical watershed is an optimal cut of the original graph*
- *The ordering $\prec$ can be obtained from morphological flooding of w (e.g., dynamics, area, volume, range, energy based filters) which amounts of taking non horizontal cuts in the quasi-flat zones hierarchy of w [COUSTY 2014]*

Algorithmic scheme: MST re-weighting

1. Consider the hierarchy of quasi-flat zones through
   - Q_{BT} and its associated MST
2. Compute some attributes on the regions directly from Q_{BT} and/or from external data (*e.g.*, the original image)
3. Rank the minima of w based on these attributes (extinction values)
4. Re-weight the edges of MST with a map w' such that
   - The hierarchical watershed is the quasi-flat zones hierarchy of (MST, w')

MST re-weighting

Procedure AttributeBasedHierarchy

Data: Q_{BT}, $\prec = \langle M_1, \ldots, M_\ell \rangle$

Result: a re-weighting w' of MST whose quasi-flat zone hierarchy is the attribute-based hierarchy

1 **for** *any* i *in* $\{1, \ldots, \ell\}$ **do**
2 Extract one vertex n of M_i;
3 $attribute[n] := i$;
4 **for** *any non-leaf node* n *of* Q_{BT} *by increasing order* **do**
5 $a_1 := attribute[children[n].left]$;
6 $a_2 := attribute[children[n].right]$;
7 $attribute[n] := \max(a_1, a_2)$;
8 $w'[getEdge(n)] := \min(a_1, a_2)$;

Algorithmic scheme: MST re-weighting

1. Consider the hierarchy of quasi-flat zones through
   - Q_{BT} and its associated MST
2. Compute some attributes on the regions directly from Q_{BT} and/or from external data (*e.g.*, the original image)
3. Rank the minima of w based on the attributes (extinction value)
4. Re-weight the edges of MST with a map w' such that
   - The hierarchical watershed is the quasi-flat zones hierarchy of (MST, w')

Property

- *The time-complexity of AttributeBasedHierarchy (step 4) is linear with respect to* $|V|$
- *When step 2 can be done in linear-time, the overall time-complexity is quasi-linear with respect to* $|V| + |E|$

Other transformations of hierarchy

- The exact same algorithmic scheme can be used for:
  - [Soille 08]: constrained connectivity
  - [Guimaraes 12]: hierarchy based on observation scale [Felzenszwalb 04]
- Saliency map algorithm can be used for
  - [Arbelaez 08]: GPB ultrametric contour map hierarchy
- The scheme has to be adapted for efficient implementations of
  - [Guigues 06, Kiran 14] Hierarchies by scale increasing energies

On-line demonstration/source codes

- On-line tool for marker-based segmentation in hierarchies: `http://www.esiee.fr/~perretb/ISeg/`
- Source codes in C of some of the presented algorithms: `http://www.esiee.fr/~info/sm/`

Conclusion

- Versatile and efficient framework for computing hierarchies
- Open issue : how to evaluate hierarchy of partitions?
  - [Arbelaez 11] or [Pont-Tuset 13]
- The topic is hot:
  - [Arbelaez 11] has already 686 citations on google scholar
  - The PhD defense of Kiran on 10/31/2014
  - Your presence today !

References # 1: handling hierarchies in graphs

- Najman, *On the Equivalence Between Hierarchical Segmentations and Ultrametric Watersheds*, JMIV 2011
- Cousty & Najman, *Incremental algorithm for hierarchical minimum spanning forests and saliency of watershed cuts*, ISMM 2011
- Cousty, Najman & Perret *Constructive links between some morphological hierarchies on edge-weighted graphs*, ISMM 2013
- Najman, Cousty & Perret *Playing with Kruskal: algorithms for morphological trees in edge-weighted graphs*, ISMM 2013

- The proofs of the two theorems of this lecture are going to be given in a forthcoming article

References # 2: Pioneering work

- Nagao, Matsuyama & Ikeda *Region extraction and shape analysis in aerial photographs* , CGIP 1979
- Bouman & Shapiro *A multiscale random field model for Bayesian image segmentation*, TIP 1994
- Beucher, *Watershed, hierarchical segmentation and waterfall algorithm*, ISMM 1994
- Najman & Schmitt *Geodesic saliency of watershed contours and hierarchical segmentation*, PAMI 1996
- Udupa & Samarasekera *Fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation*, GMIP 1996
- Meyer & Maragos *Playing with Kruskal: algorithms for morphological trees in edge-weighted graphs*, Scale-Space 1999
- Salembier & Garrido *Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval* , TIP 2000

References # 3: more recent work

- Felzenszwalb & Huttenlocher *Efficient Graph-Based Image Segmentation*, IJCV 2004
- Guigues & Cocquerez *Scale-sets image analysis*, IJCV 2006
- Soille *Constrained connectivity for hierarchical image partitioning and simplification*, PAMI 2008
- Arbelaez & Cohen, *Constrained image segmentation from hierarchical boundaries*, CVPR 2008
- Cousty, Bertrand, Najman & Couprie, *Watershed cuts: minimum spanning forests, and the drop of water principle*, PAMI 2009

References # 4: more recent work

- Arbelaez, Maire, Fowlkes & Malik, *Contour Detection and Hierarchical Image Segmentation*, PAMI 2011
- Guimaraes, Cousty, Kenmochi & Najman, *A Hierarchical Image Segmentation Algorithm Based on an Observation Scale*, SSSPR 2012
- Pont-Tuset & Marques, *Measures and Meta-Measures for the Supervised Evaluation of Image Segmentation*, CVPR 2013
- Cousty & Najman, *Morphological Floodings And Optimal Cuts In Hierarchies*, ICIP 2014
- Kiran, *Energetic-Lattice based optimization*, PhD Thesis, to be defended at ESIEE Paris on 10/31/2014