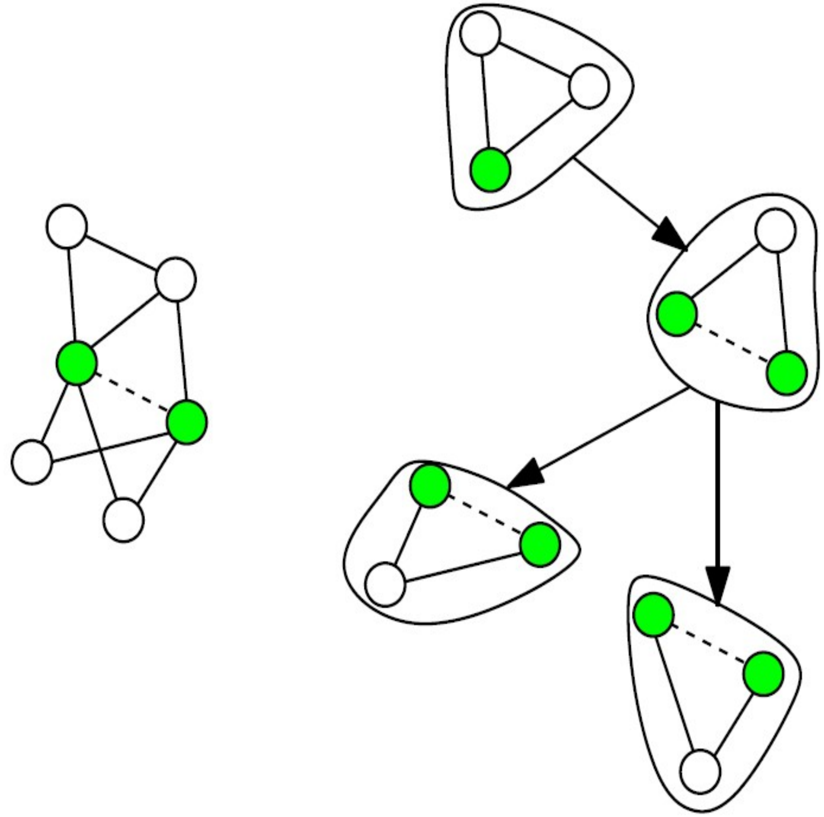
Minimum Cycle Bases in Partial 2-Trees



Carola Doerr G. Ramakrishna Jens M. Schmidt

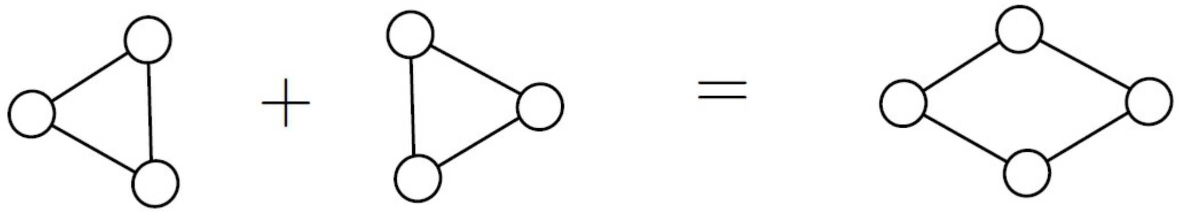
Minimum Cycle Basis (MCB)

- G = graph with non-negative weights

Minimum Cycle Basis (MCB)

- G = graph with non-negative weights
- cycle basis = minimum-cardinality set of cycles that can generate every cycle in G

$$C_1 + C_2 := E(C_1) \Delta E(C_2)$$

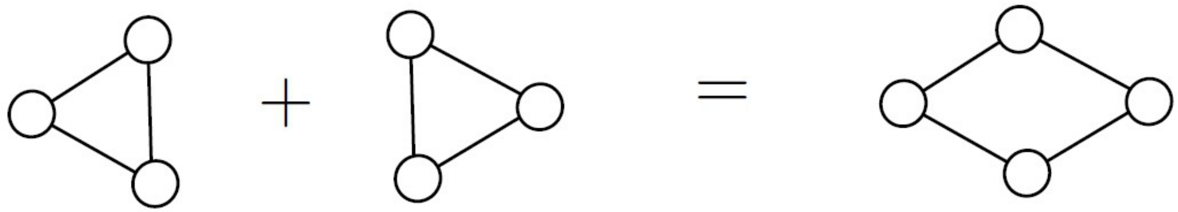


Minimum Cycle Basis (MCB)

- G = graph with non-negative weights
- cycle basis = minimum-cardinality set of cycles that can generate every cycle in G

m-n+1

$$C_1 + C_2 := E(C_1) \Delta E(C_2)$$

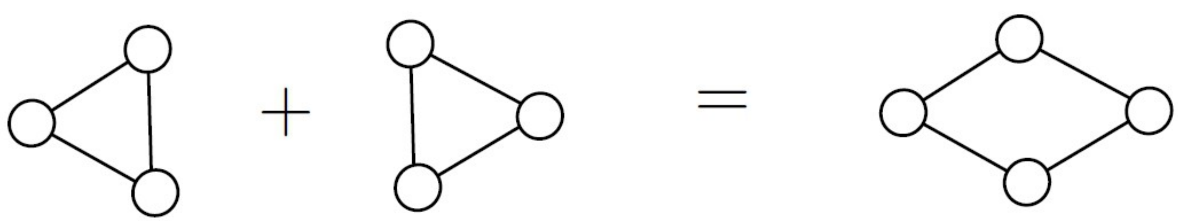


Minimum Cycle Basis (MCB)

- G = graph with non-negative weights
- cycle basis = minimum-cardinality set of cycles that can generate every cycle in G

m-n+1

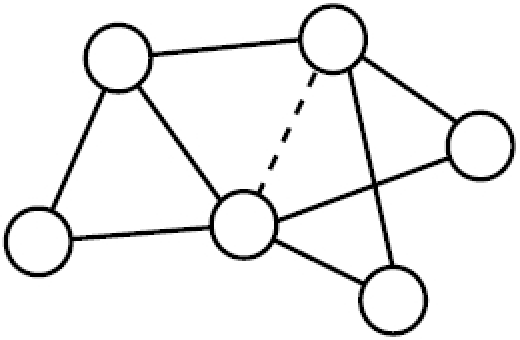
$$C_1 + C_2 := E(C_1) \Delta E(C_2)$$



- minimum cycle basis = minimize sum of edge-weights

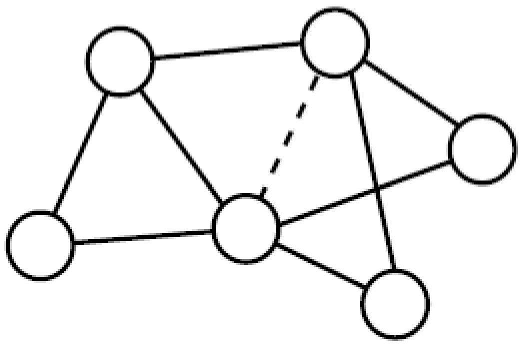
Partial 2-Trees

- 2-trees:
  - Start with a triangle
  - Add iteratively new vertex that is adjacent to a K_2



Partial 2-Trees

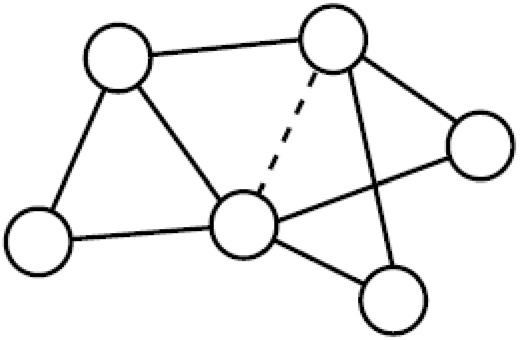
- 2-trees:
  - Start with a triangle
  - Add iteratively new vertex that is adjacent to a K_2



- partial 2-trees = subgraphs of 2-trees

Partial 2-Trees

- 2-trees:
  - Start with a triangle
  - Add iteratively new vertex that is adjacent to a K_2



- partial 2-trees = subgraphs of 2-trees
- outerplanar graphs ⊂ partial 2-trees

Recent Work

- For general graphs:
  - Deterministic: $O(m^2 n / \log n)$ [Amaldi-Iuliano-Rizzi '10]
  - Monte-Carlo: $O(m^\omega)$ [“-Jurkiewicz-Mehlhorn '09]

Recent Work

	[LL 2010]	[Our Result]	[BSW 2010]
	Outerplanar	Partial 2-trees	Planar
Preprocessing time	$O(n)$	$O(n)$	$O(n \log^5 n)$
Space required	$O(n)$	$O(n)$	$O(n \log n)$
Reporting time	$size(MCB)$	$size(MCB)$	$size(MCB)$

Recent Work

	[LL 2010] Outerplanar	[Our Result] Partial 2-trees	[BSW 2010] Planar
Preprocessing time	$O(n)$	$O(n)$	$O(n \log^5 n)$
Space required	$O(n)$	$O(n)$	$O(n \log n)$
Reporting time	$size(MCB)$	$size(MCB)$	$size(MCB)$

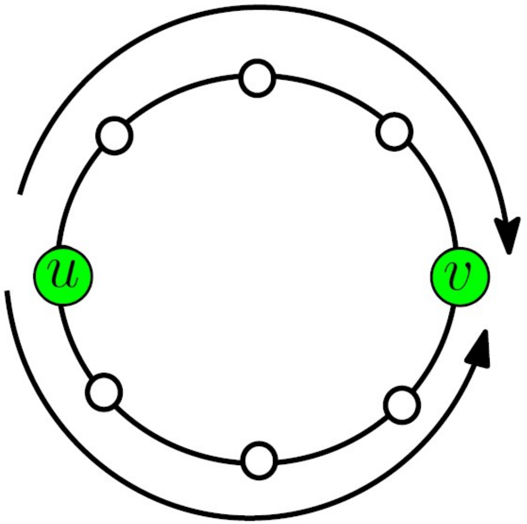
Can be big: $\Omega(n^2)$

Short Cycles

- Assumption:
  Unique shortest paths, otherwise [Hartvigsen-Mardon '94]

Short Cycles

- Assumption:
  Unique shortest paths, otherwise [Hartvigsen-Mardon '94]
- sp(u,v) = shortest path between u and v
- Cycle C is short if for all u,v in C, sp(u,v) ⊆ C

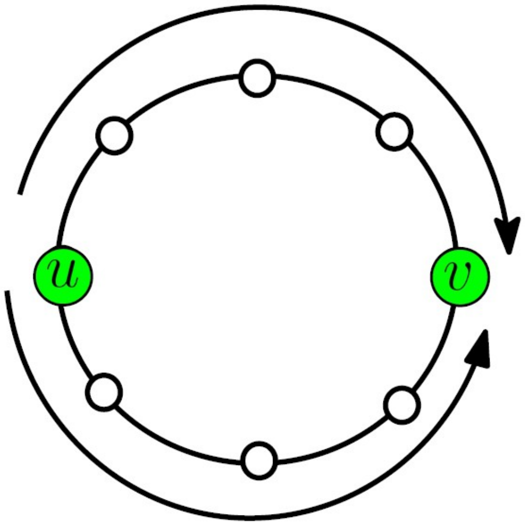


Short Cycles

- Assumption:
  Unique shortest paths, otherwise [Hartvigsen-Mardon '94]

- sp(u,v) = shortest path between u and v

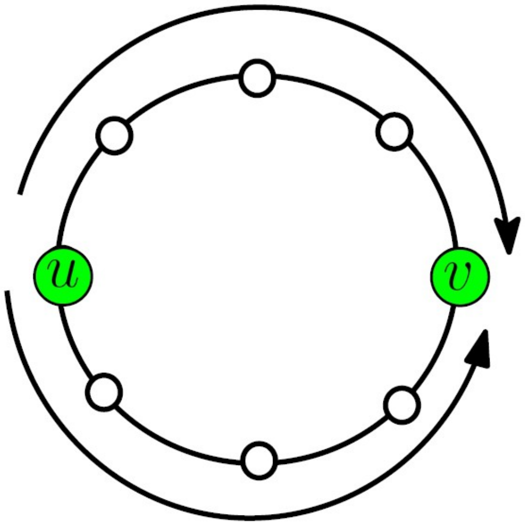
- Cycle C is short if for all u,v in C, sp(u,v) ⊆ C



- SC = set of all short cycles

Short Cycles

- Assumption:
  Unique shortest paths, otherwise [Hartvigsen-Mardon '94]
- sp(u,v) = shortest path between u and v
- Cycle C is short if for all u,v in C, sp(u,v) $\subseteq$ C

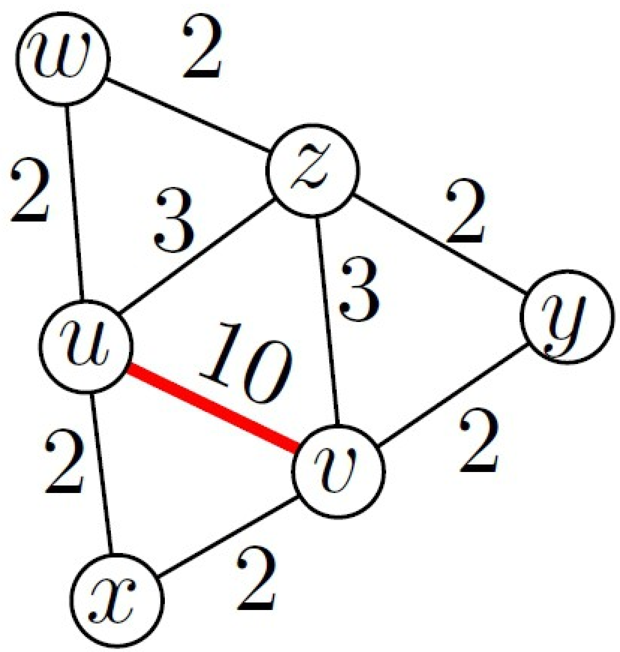


- SC = set of all short cycles

Thm [N.+Ramakrishna '12]: MCB = SC for partial 2-trees

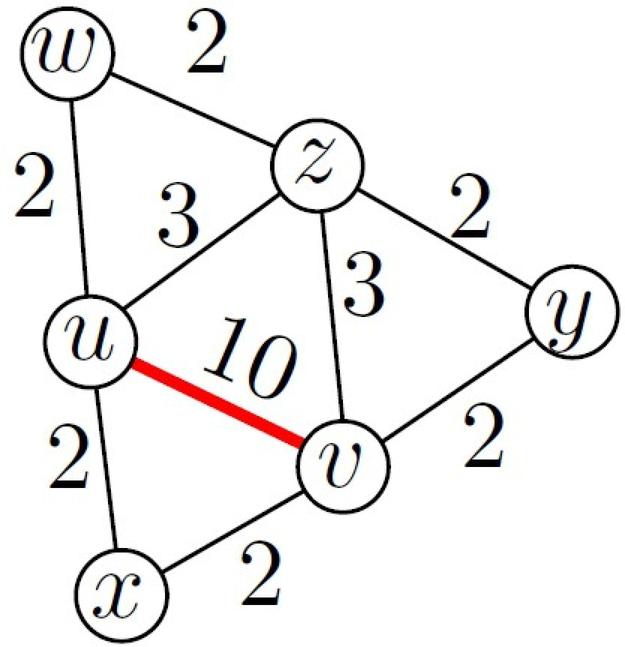
Tight Edges

- Edge (u,v) tight if sp(u,v)=(u,v), otherwise loose
- Every loose edge appears in exactly one SC



Tight Edges

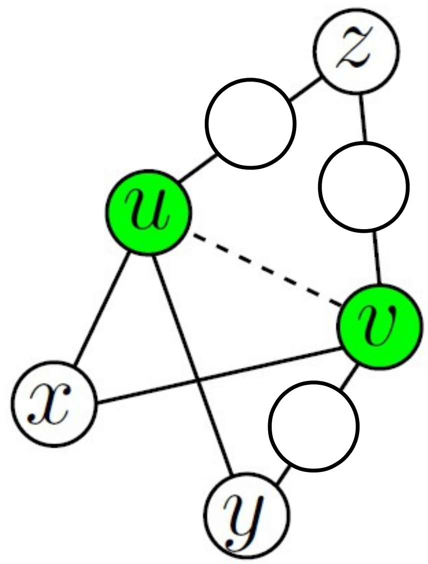
- Edge (u,v) tight if sp(u,v)=(u,v), otherwise loose
- Every loose edge appears in exactly one SC



- Precompute these cycles => G contains only tight edges

Decomposition

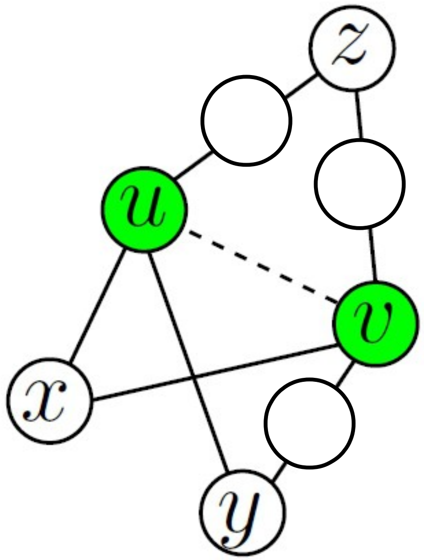
- If partial 2-tree has no $K_{2,3}$-subdivision => outerplanar!



- Decompose G along $K_{2,3}$-subdivisions into outerplanar graphs $G_1,...,G_r$

Decomposition

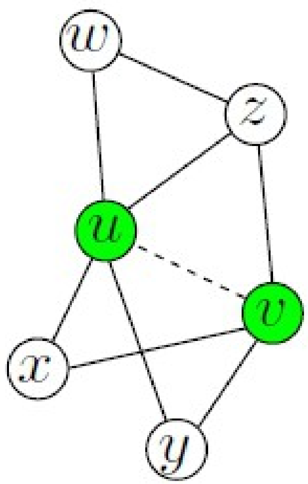
- If partial 2-tree has no $K_{2,3}$-subdivision => outerplanar!

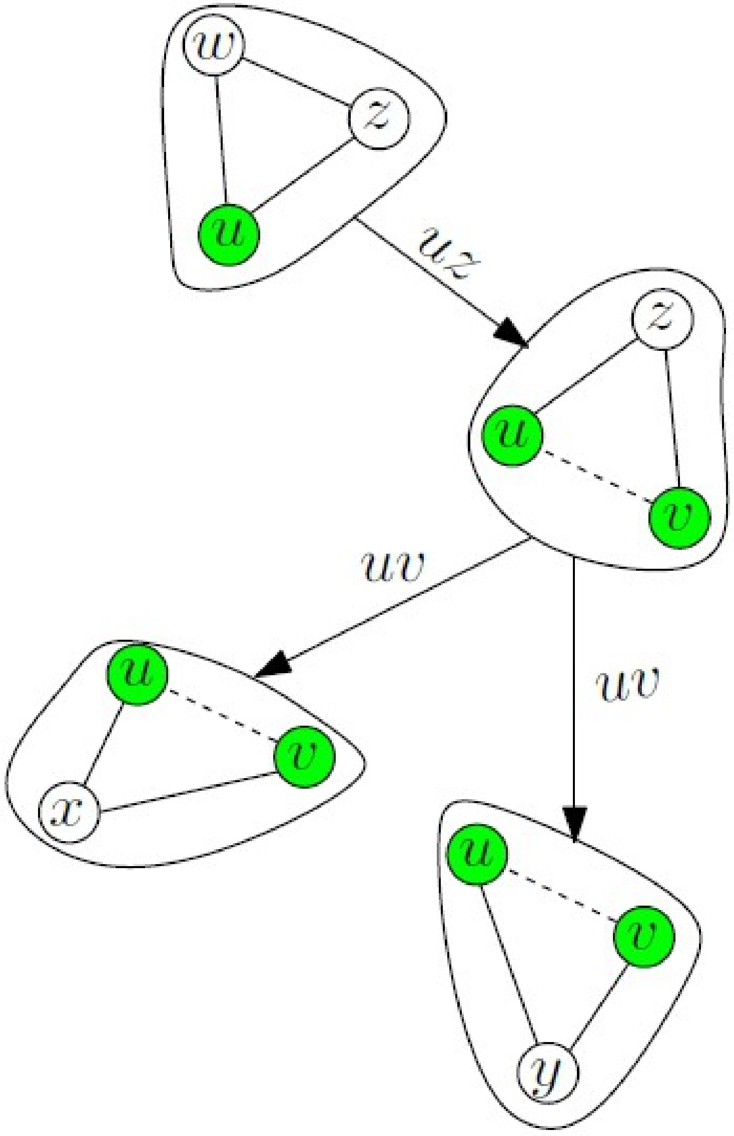


- Decompose G along $K_{2,3}$-subdivisions into outerplanar graphs $G_1,...,G_r$

Thm [N.+Ramakrishna '12]: $SC(G) = SC(G_1) \cup \ldots \cup SC(G_r)$

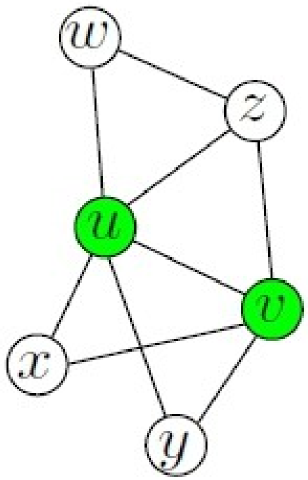
Decomposition

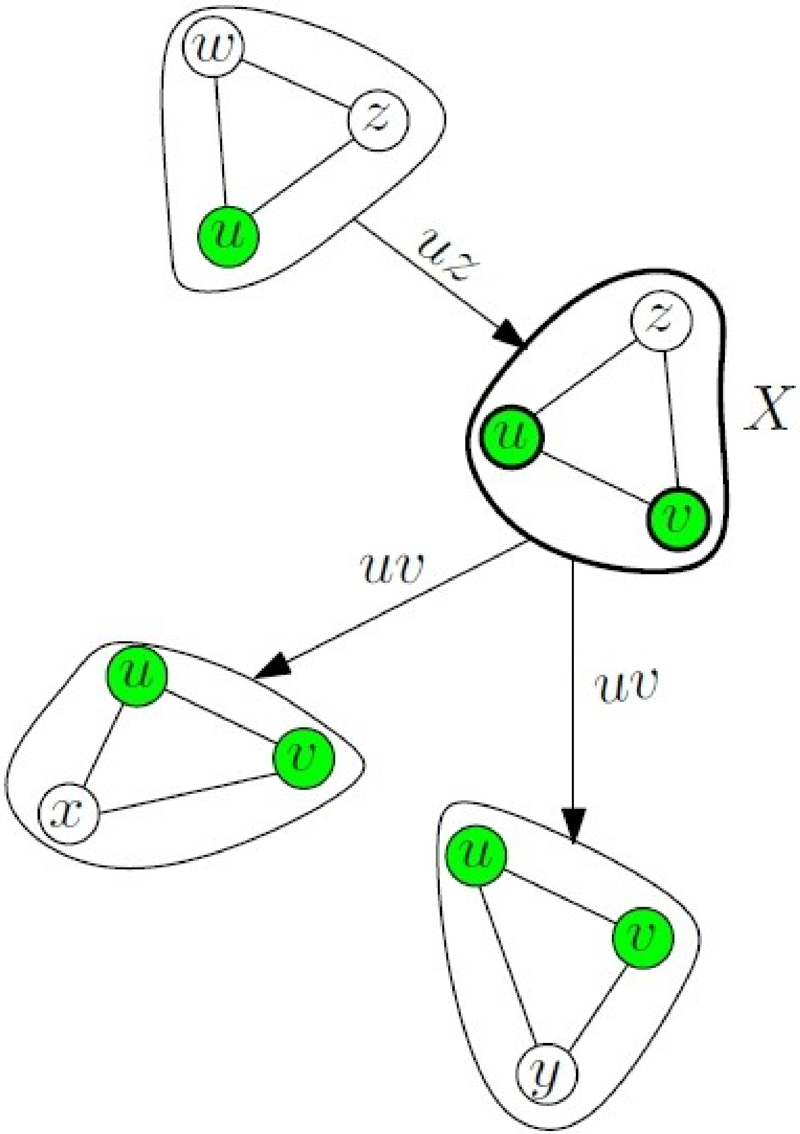




Decomposition

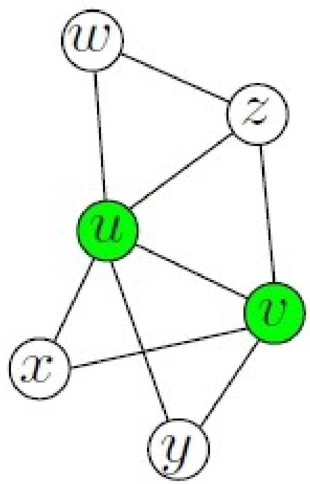
If $(u,v) \in G$:

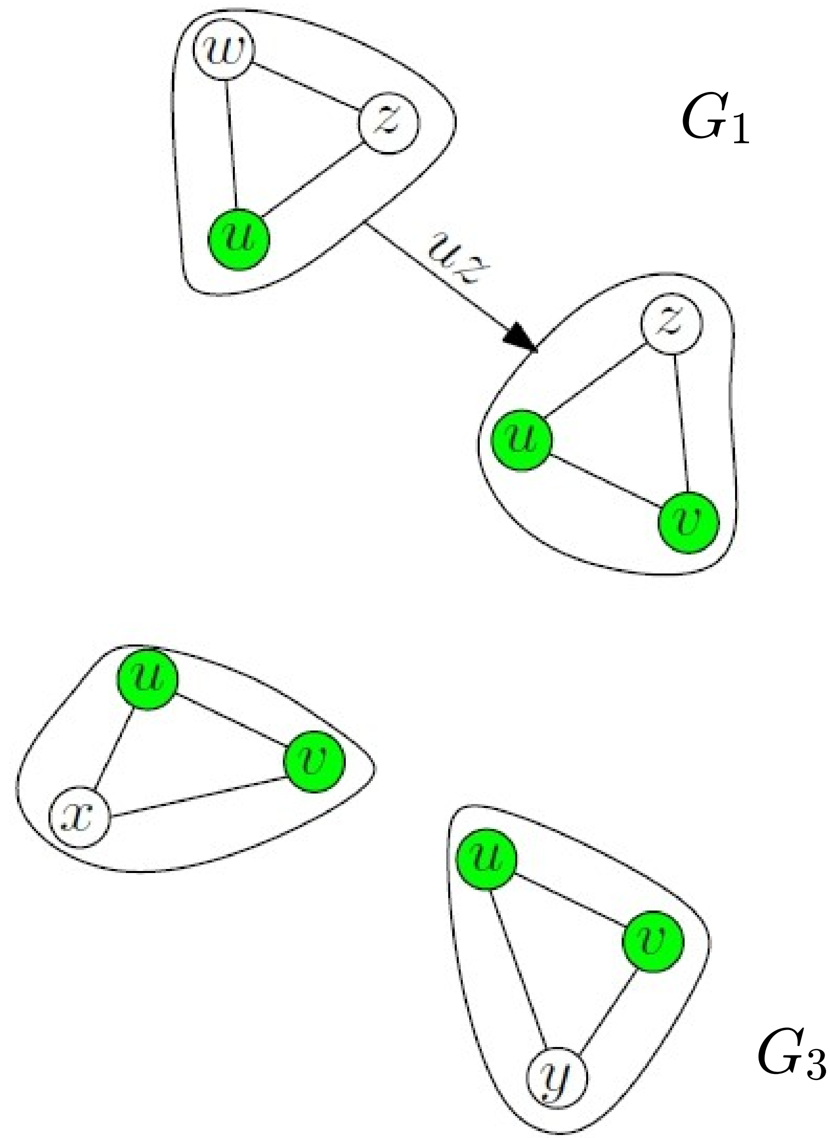




Decomposition

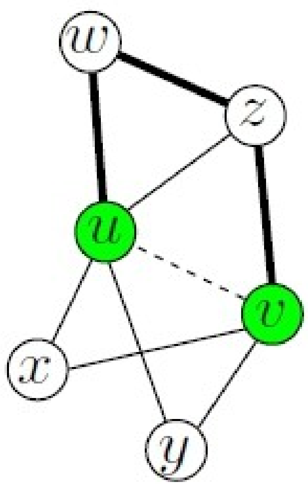
If $(u,v) \in G$:

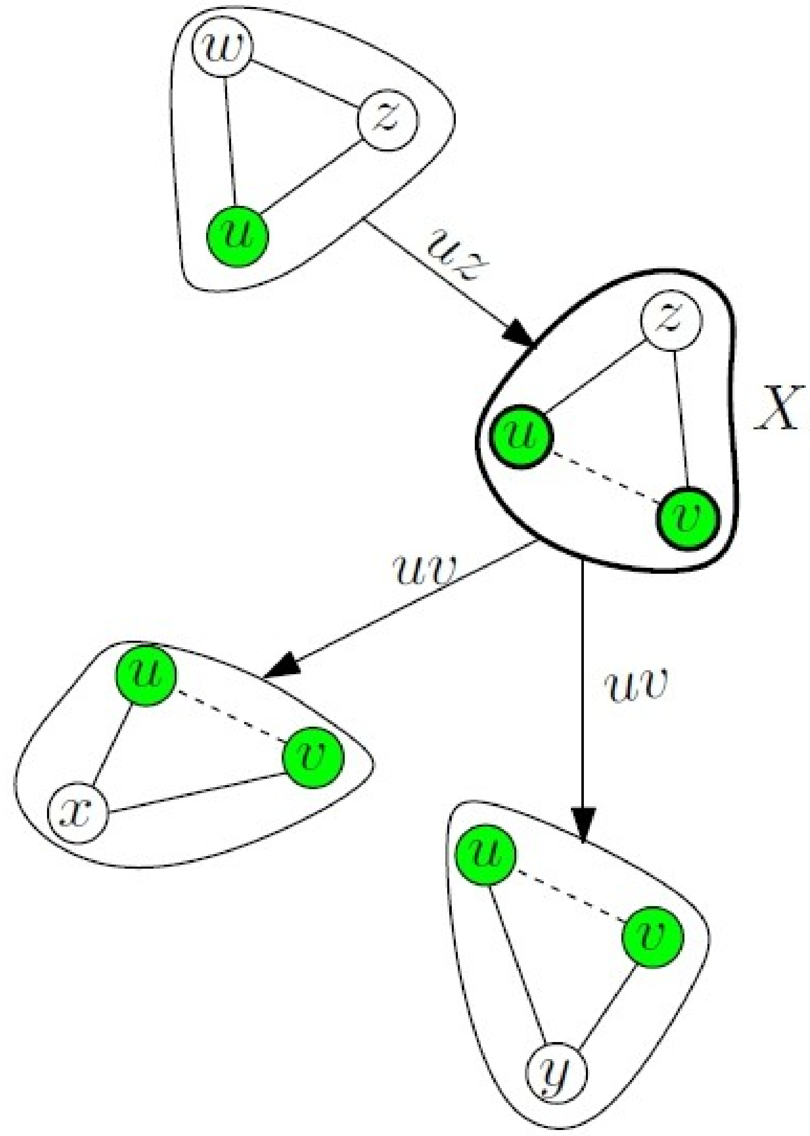




Decomposition

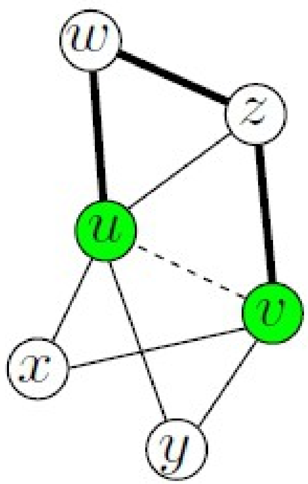
If $(u,v) \notin$ G:

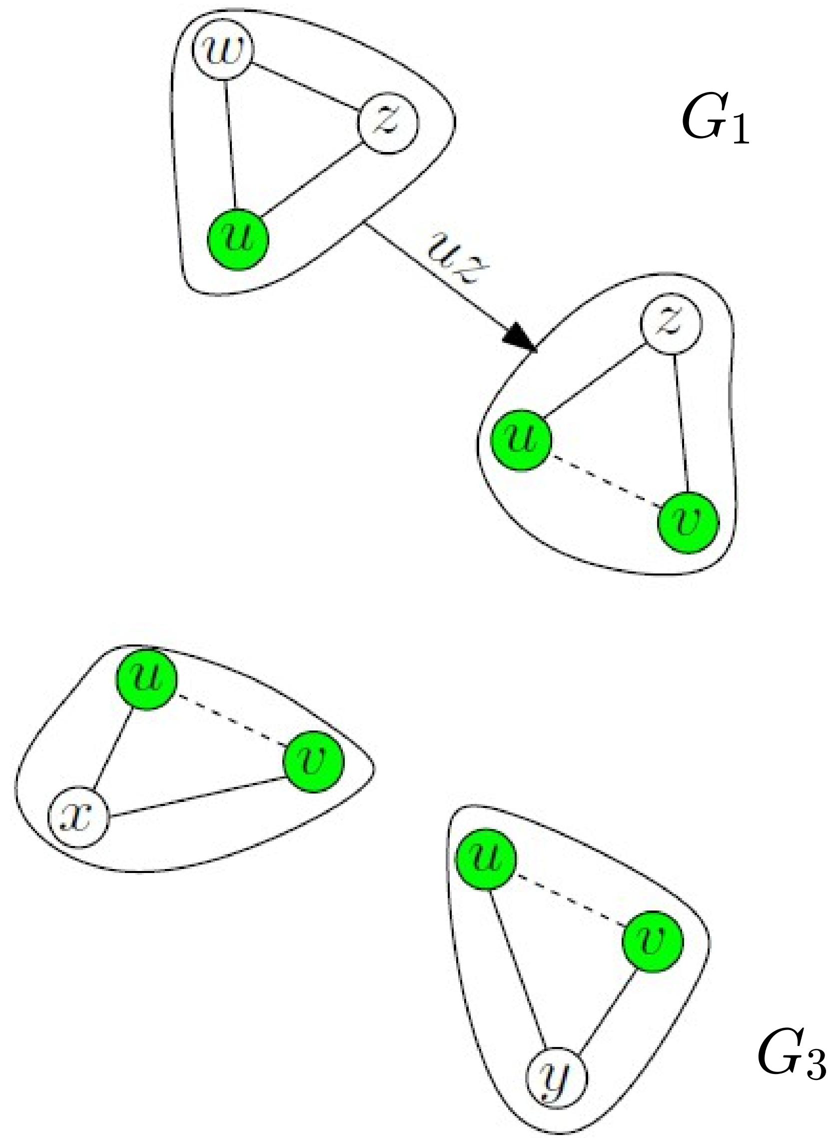




Decomposition

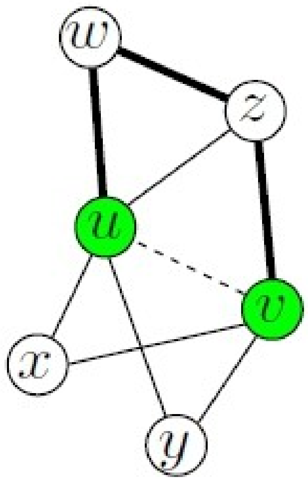
- If $(u,v) \notin$ G:

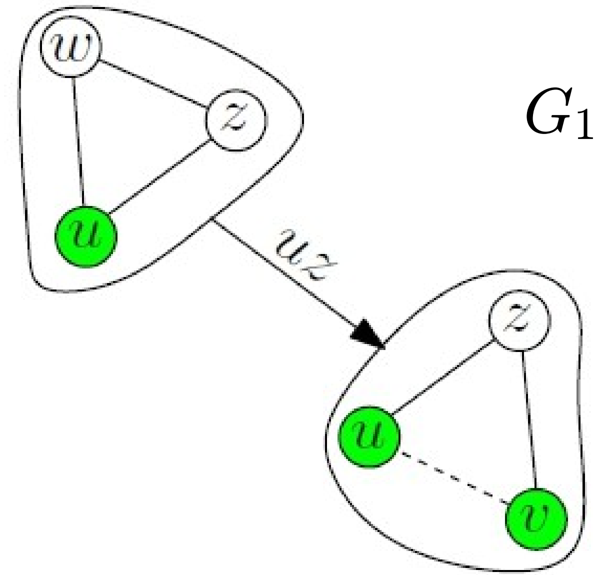


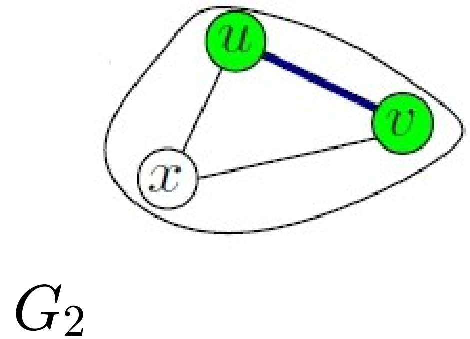


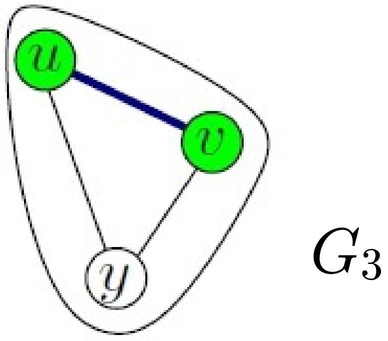
Decomposition

If $(u,v) \notin \mathrm{G}$:





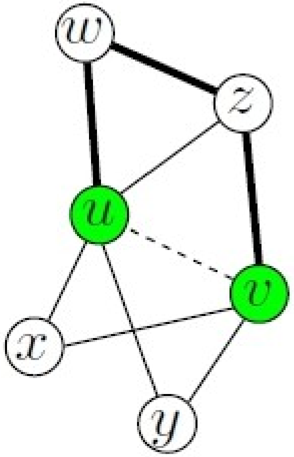




blue edges represent sp(u,v)

Algorithm

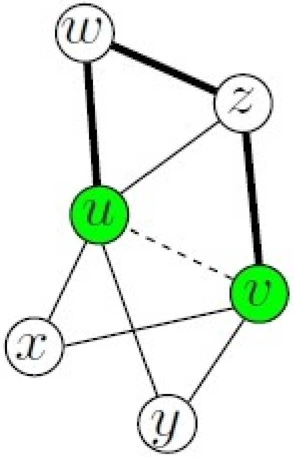
- Find all u,v such that G-u-v has at least 3 components



Algorithm

- Find all u,v such that G-u-v has at least 3 components

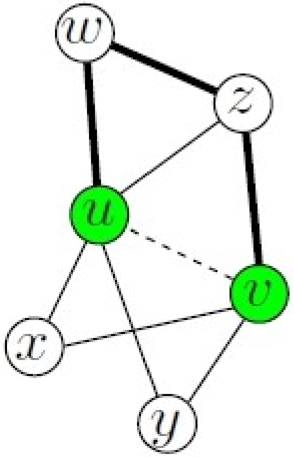
Suitable tree decomposition
(u,v in common bag)



Algorithm

- Find all u,v such that G-u-v has at least 3 components

  Suitable tree decomposition
  (u,v in common bag)

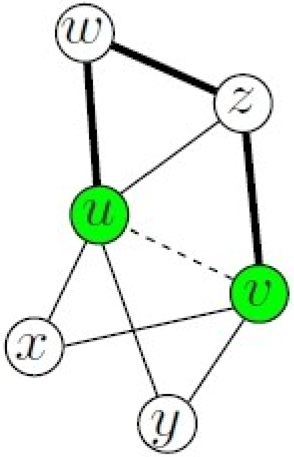


- Find G_i that contains sp(u,v) and decompose

Algorithm

- Find all u,v such that G-u-v has at least 3 components

  Suitable tree decomposition
  (u,v in common bag)



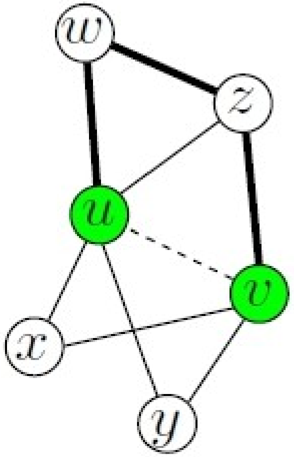
- Find G_i that contains sp(u,v) and decompose

  [Chaudhuri-Zaroliagis '00]
  Data structure for fixed tree-width graphs supporting
  - intermediate vertex queries in O(1)
  - bag location queries in O(1)
  - distance queries in O(1)

Algorithm

- Find all u,v such that G-u-v has at least 3 components

    Suitable tree decomposition
    (u,v in common bag)



- Find G_i that contains sp(u,v) and decompose

    [Chaudhuri-Zaroliagis '00]
    Data structure for fixed tree-width graphs supporting
    - intermediate vertex queries in O(1)
    - bag location queries in O(1)
    - distance queries in O(1)

- Compute MCB from $SC(G_i)$ and the SC from loose edges

Thank you!