30^{th} TBI Winterseminar Bled:

Heuristic for Cograph Editing

Adrian Fritz

February 18, 2015

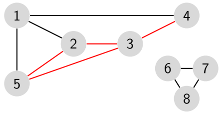
Based on the master thesis supervised by Dr. Marc Hellmuth

Cographs

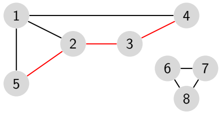
- Recursive definition (omitted)

Cographs

- Recursive definition (omitted)
- A graph is a cograph if and only if it does not contain an induced P_4 [1]



Cograph

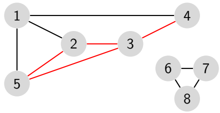


No cograph, marked induced P_4

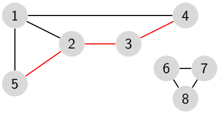
[1] Lerchs, Burlingham, Stewart: "Complement reducible graphs" (Discrete Applied Mathematics, 1981)

Cographs

- Recursive definition (omitted)
- A graph is a cograph if and only if it does not contain an induced P_4 [1]



Cograph



No cograph, marked induced P_4

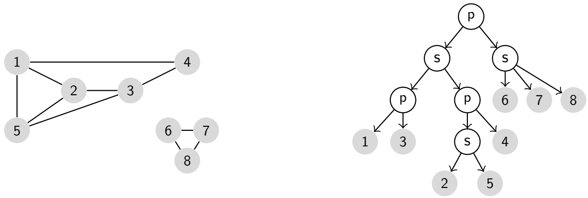
- Can uniquely be represented as a cotree

[1] Lerchs, Burlingham, Stewart: "Complement reducible graphs" (Discrete Applied Mathematics, 1981)

Cotree

- Leaves are vertices from the original graph, internal nodes are labelled s(eries) or p(arallel)

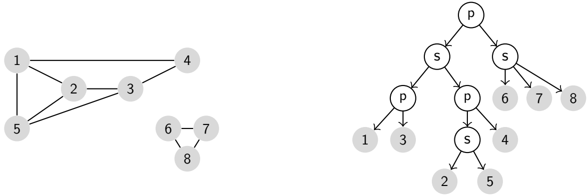
A Cograph and its corresponding Cotree



Cotree

- Leaves are vertices from the original graph, internal nodes are labelled s(eries) or p(arallel)
- Two nodes have an edge if and only if their LCA is a series node

A Cograph and its corresponding Cotree



Plan for Cograph-editing heuristic

Plan for Cograph-editing heuristic

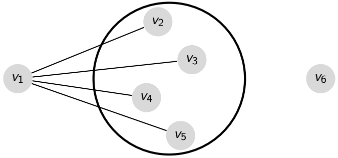
1. Perform Modular Decomposition
2. Edit prime nodes in the Modular Decomposition tree
3. Cograph-Editing: Return cograph G^* edited from graph G with as few as possible edge operations

Modular Decomposition:

- Generalization of cotree concept to all graphs
- Internal nodes of the MD tree are so-called modules

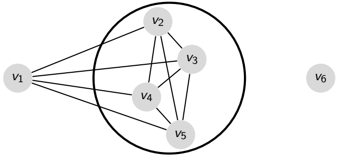
Modular Decomposition:

- Generalization of cotree concept to all graphs
- Internal nodes of the MD tree are so-called modules
- Two vertices are in the same module if they have the same *outer* neighborhood
- Module type depends on the *inner* neighborhood
    1. Inner neighborhood is a stable set: Parallel



Modular Decomposition:

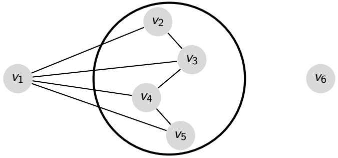
- Generalization of cotree concept to all graphs
- Internal nodes of the MD tree are so-called modules
- Two vertices are in the same module if they have the same *outer* neighborhood
- Module type depends on the *inner* neighborhood
  1. Inner neighborhood is a stable set: Parallel
  2. Inner neighborhood is a clique: Series



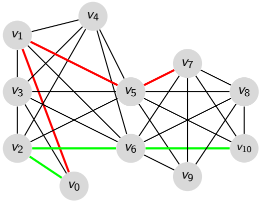
"Series" module

Modular Decomposition:

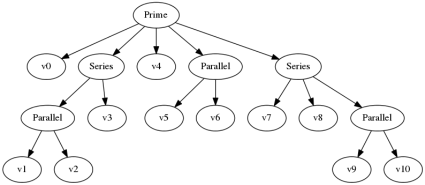
- Generalization of cotree concept to all graphs
- Internal nodes of the MD tree are so-called modules
- Two vertices are in the same module if they have the same *outer* neighborhood
- Module type depends on the *inner* neighborhood
  1. Inner neighborhood is a stable set: Parallel
  2. Inner neighborhood is a clique: Series
  3. "Everything else": Prime



Modular Decomposition



Graph G with two marked P_4's



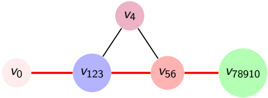
Corresponding Modular Decomposition Tree

- Why Modular Decomposition?

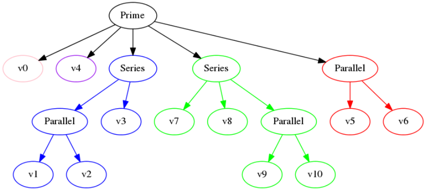
  All induced P_4's are entirely contained in prime modules[2]
  Hence: Edge operations have only to be performed on children of prime nodes

[2] McConnell, Spinrad: "Modular decomposition and transitive orientation" (Discrete Mathematics, 1999)

Modular Decomposition



Quotient graph G_p of prime node

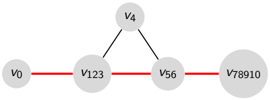


Corresponding Modular Decomposition Tree

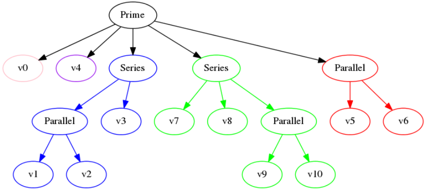
- Why Modular Decomposition?

  All induced P_4's are entirely contained in prime modules[2]
  Hence: Edge operations have only to be performed on children of prime nodes

[2] McConnell, Spinrad: "Modular decomposition and transitive orientation" (Discrete Mathematics, 1999)

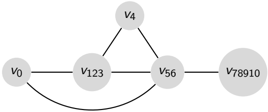


Quotient graph G_p of prime node

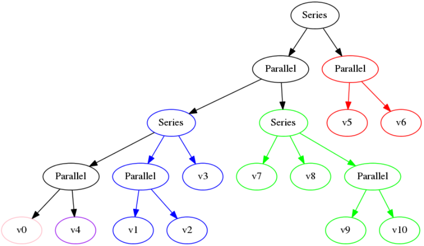


Corresponding Modular Decomposition Tree

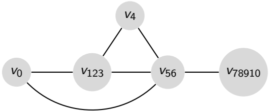
- Goal: Eliminate P_4's



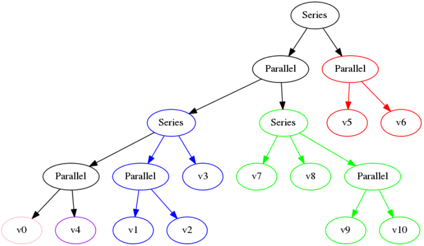
Quotient graph G_p



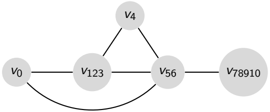
- Goal: Eliminate P_4's
- Done by adding or deleting edges



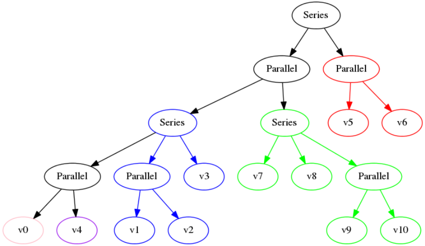
Quotient graph G_p



- Goal: Eliminate P_4's
- Done by adding or deleting edges
- Which edges do we edit?



Quotient graph G_p



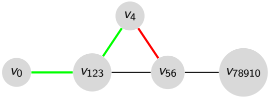
- Goal: Eliminate P_4's
- Done by adding or deleting edges
- Which edges do we edit?
- Reformulation of the problem

Edit MD tree to cotree

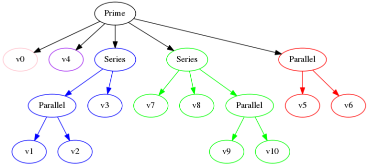
- Get rid of “unresolved” prime modules:

Edit MD tree to cotree

- Get rid of “unresolved” prime modules:
- Merge two children of prime module with similar neighborhood



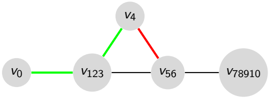
Quotient graph G_p



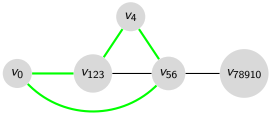
Corresponding Modular Decomposition Tree

Edit MD tree to cotree

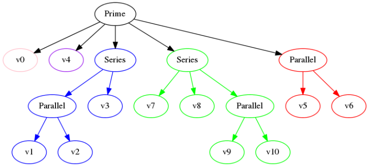
- Get rid of "unresolved" prime modules:
- Merge two children of prime module with similar neighborhood



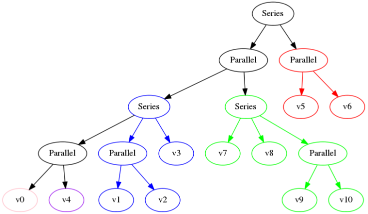
Quotient graph G_p



Quotient graph G_p



Corresponding Modular Decomposition Tree



Corresponding Modular Decomposition Tree

- Termination

- Correctness

- Termination
  1. All prime modules have at least 4 children because they contain induced P_4
  2. By resolving, the number of children is reduced by at least 1
  3. For a prime module with n children, we need to merge at most n-3 times
  4. Start with “lowest” one, propagate upwards
- Correctness

- Termination
  1. All prime modules have at least 4 children because they contain induced P_4
  2. By resolving, the number of children is reduced by at least 1
  3. For a prime module with n children, we need to merge at most n-3 times
  4. Start with "lowest" one, propagate upwards
- Correctness
  1. If no prime modules are left - graph is cograph
  2. Theorem: Merging can always find an optimal solution (proof: work in progress)

- Termination
  1. All prime modules have at least 4 children because they contain induced P_4
  2. By resolving, the number of children is reduced by at least 1
  3. For a prime module with n children, we need to merge at most n-3 times
  4. Start with "lowest" one, propagate upwards
- Correctness
  1. If no prime modules are left - graph is cograph
  2. Theorem: Merging can always find an optimal solution (proof: ~~work in progress~~ done!)

- Termination
  1. All prime modules have at least 4 children because they contain induced P_4
  2. By resolving, the number of children is reduced by at least 1
  3. For a prime module with n children, we need to merge at most n-3 times
  4. Start with "lowest" one, propagate upwards
- Correctness
  1. If no prime modules are left - graph is cograph
  2. Theorem: Merging can always find an optimal solution (proof: ~~work in progress~~ done!)
- Which modules do we merge?

Module Merging

- Greedy: Choose the two modules that are cheapest to merge
- $C_1 = |N[x] \triangle N[y]|$ for x, y in the two modules to merge

Module Merging

- Greedy: Choose the two modules that are cheapest to merge
- $C_1 = |N[x] \triangle N[y]|$ for x, y in the two modules to merge
- Greedy: Choose the two modules eliminating most P_4's
- $C_2 = \#P_4$'s in graph $- \#P_4$'s in resolved graph

Module Merging

- Greedy: Choose the two modules that are cheapest to merge
- $C_1 = |N[x] \triangle N[y]|$ for x, y in the two modules to merge
- Greedy: Choose the two modules eliminating most P_4's
- $C_2 = \#P_4$'s in graph $- \#P_4$'s in resolved graph
- Combination of both: $C = \frac{C_1}{C_2}$

Testing

- Compared results of heuristics against ILP

Testset	Avg. error	max error	max %	% perfect
Co20_10	0.45	3	150	68
Co20_20	0.8	6	40	63
Co20_50	0.62	5	42.85	61
Co50_10	1.8	10	41.18	40
Co50_20	4.3	6	33.33	18
Co50_50	?	?	?	?
Co100_10	?	?	?	?
Co100_20	?	?	?	?
Co100_50	?	?	?	?

Outlook & To-Do

1. Complete some proofs

Outlook & To-Do

1. Complete some proofs
2. Increase performance and efficiency

Outlook & To-Do

1. Complete some proofs
2. Increase performance and efficiency
3. Testing with more data sets
   - Worst-cases and possible solutions
   - Biological data (Artificial Life Framework)

Thank you

Testing

- Biological data are "noisy" cographs
- Create cograph, randomly add (or remove) some edges

Testset	$\|V\|$	$\|E\|$ (*avg.*)	% changed	size
Co20_10	20	147.5	10	100
Co20_20	20	150.9	20	100
Co20_50	20	162.9	50	100
Co50_10	50	1007.9	10	100
Co50_20	50	1030.2	20	100
Co50_50	50	1080.0	50	100
Co100_10	100	4132.3	10	100
Co100_20	100	4175.6	20	100
Co100_50	100	4397.5	50	100