On the DCJ Median Problem

Mingfu Shao   Bernard M.E. Moret

Ecole Polytechnique Fédérale de Lausanne (EPFL),
Lausanne, Switzerland

June 12, 2014

Evolution of Genomes

- Point mutations
  - Base pair substitutions
  - Base pair insertions and deletions

Evolution of Genomes

- Point mutations
  - Base pair substitutions
  - Base pair insertions and deletions

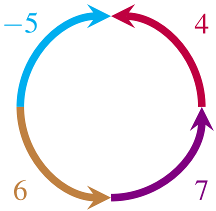
- Large-scale evolutionary events
  - Content-modifying events
    - Segmental duplications
    - Insertions and deletions
  - Rearrangements
    - Inversions
    - Translocations
    - Chromosomal fissions and fusions

Model of Genomes

- **Genome:** a set of chromosomes
- **Chromosome:** a linear/circular list of genes (signed integers)

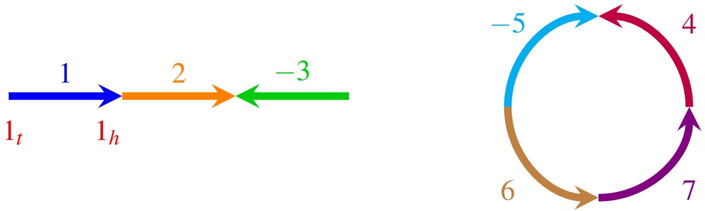
Model of Genomes

- **Genome:** a set of chromosomes
- **Chromosome:** a linear/circular list of genes (signed integers)



Model of Genomes

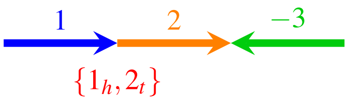
- **Genome:** a set of chromosomes
- **Chromosome:** a linear/circular list of genes (signed integers)

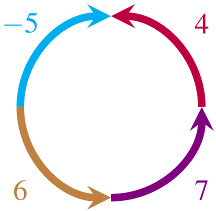


- **Extremities:** two ends (**head** and **tail**) of a gene

Model of Genomes

- **Genome:** a set of chromosomes
- **Chromosome:** a linear/circular list of genes (signed integers)

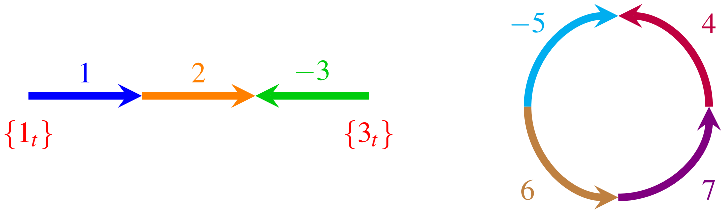




- **Extremities:** two ends (**head** and **tail**) of a gene
- **Adjacency:** two consecutive extremities

Model of Genomes

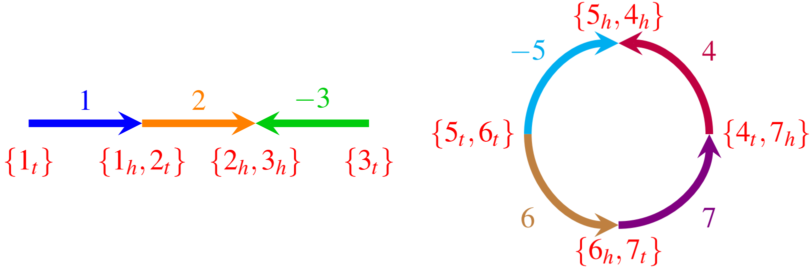
- **Genome:** a set of chromosomes
- **Chromosome:** a linear/circular list of genes (signed integers)



- **Extremities:** two ends (**head** and **tail**) of a gene
- **Adjacency:** two consecutive extremities
- **Telomeres:** two ends of a linear chromosome

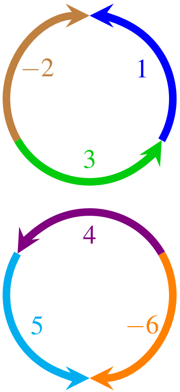
Model of Genomes

- **Genome:** a set of chromosomes
- **Chromosome:** a linear/circular list of genes (signed integers)

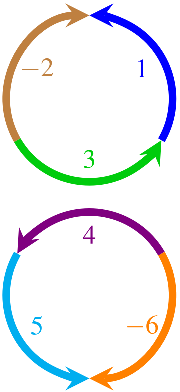


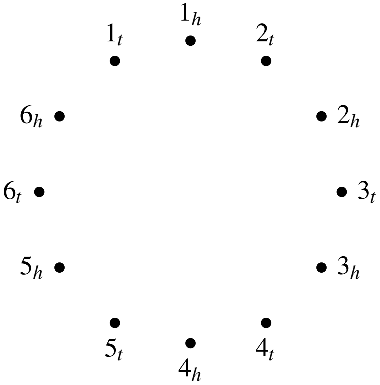
- **Extremities:** two ends (**head** and **tail**) of a gene
- **Adjacency:** two consecutive extremities
- **Telomeres:** two ends of a linear chromosome

Graph Representation

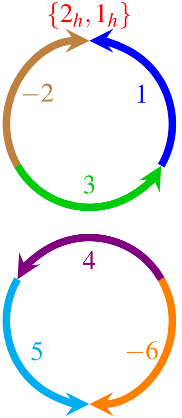


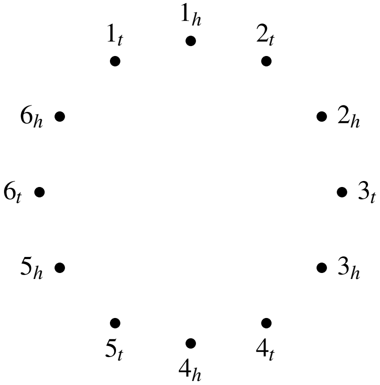
Graph Representation



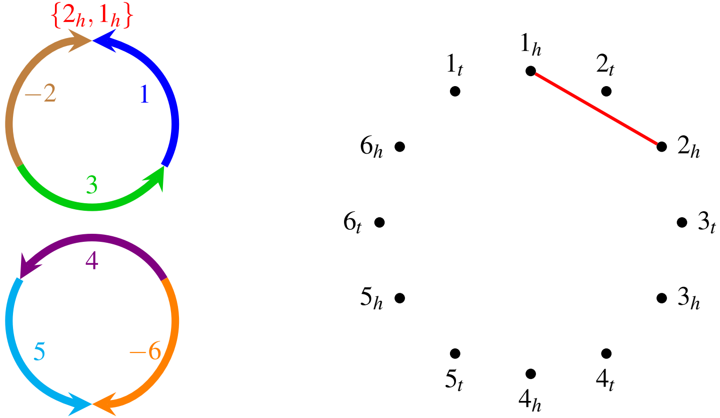


Graph Representation

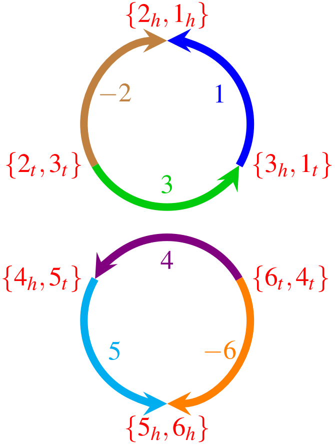


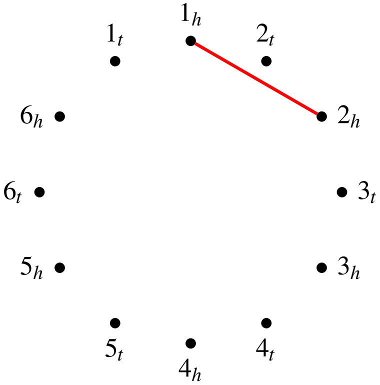


Graph Representation

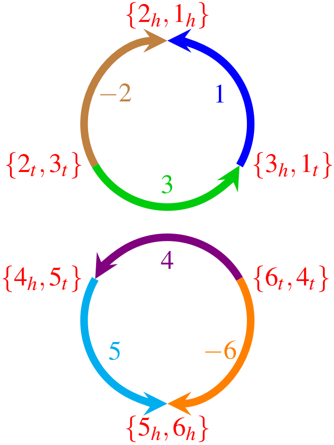


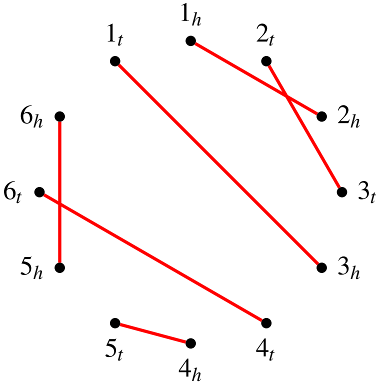
Graph Representation



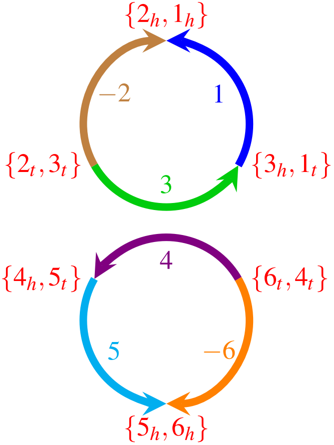


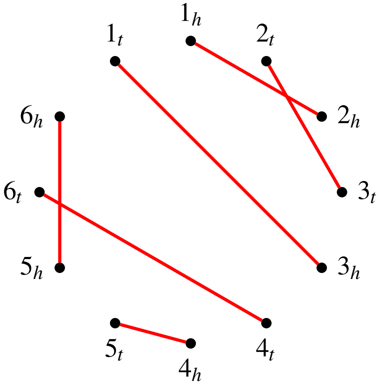
Graph Representation





Graph Representation





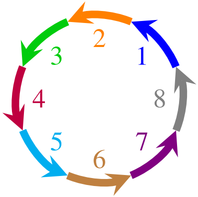
- There is an one-to-one correspondence between all perfect matchings and all genomes on the same set of genes.

Double-Cut-and-Join (DCJ) Operation

- A commonly used model for genome rearrangement
  Input: $\{p,q\}+\{s,t\}$
  Output: $\{p,s\}+\{q,t\}$ or $\{p,t\}+\{q,s\}$

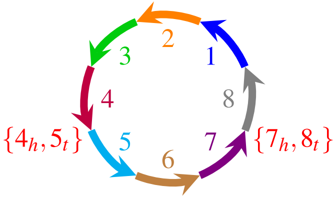
Double-Cut-and-Join (DCJ) Operation

- A commonly used model for genome rearrangement
  Input: $\{p,q\}+\{s,t\}$
  Output: $\{p,s\}+\{q,t\}$ or $\{p,t\}+\{q,s\}$



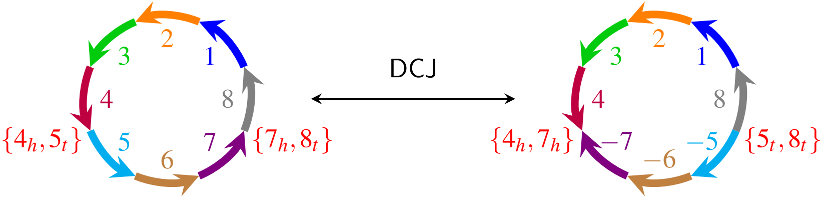
Double-Cut-and-Join (DCJ) Operation

- A commonly used model for genome rearrangement
  Input: $\{p,q\}+\{s,t\}$
  Output: $\{p,s\}+\{q,t\}$ or $\{p,t\}+\{q,s\}$



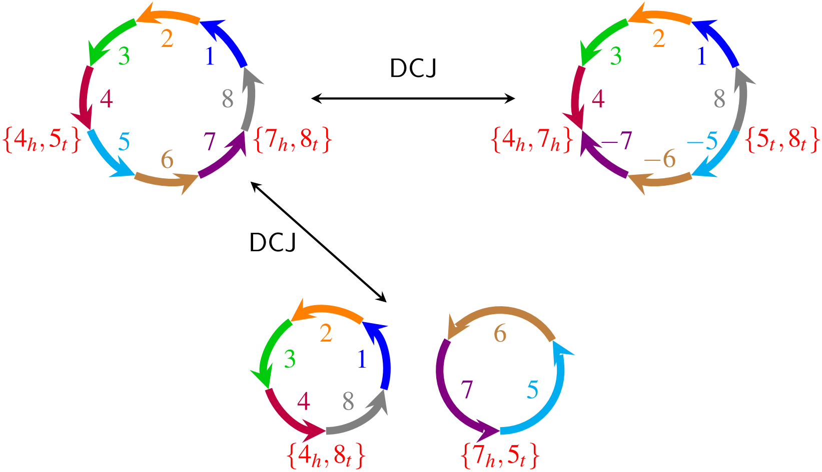
Double-Cut-and-Join (DCJ) Operation

- A commonly used model for genome rearrangement
  Input: $\{p,q\}+\{s,t\}$
  Output: $\{p,s\}+\{q,t\}$ or $\{p,t\}+\{q,s\}$



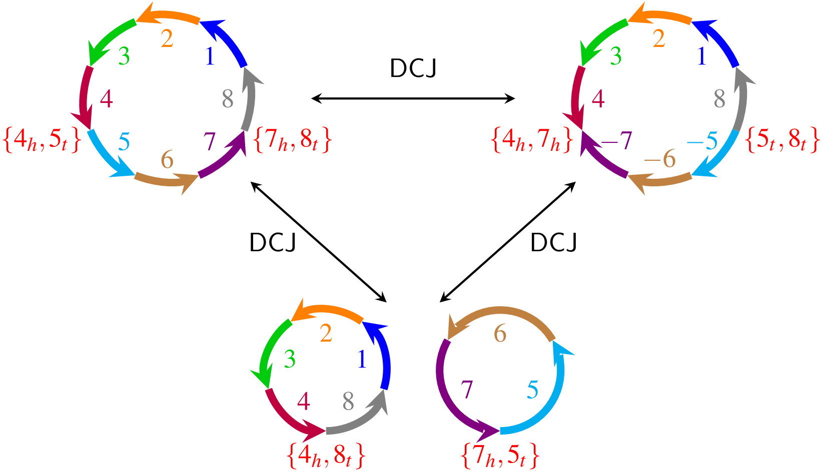
Double-Cut-and-Join (DCJ) Operation

- A commonly used model for genome rearrangement
  Input: $\{p,q\}+\{s,t\}$
  Output: $\{p,s\}+\{q,t\}$ or $\{p,t\}+\{q,s\}$



Double-Cut-and-Join (DCJ) Operation

- A commonly used model for genome rearrangement
  Input: $\{p,q\}+\{s,t\}$
  Output: $\{p,s\}+\{q,t\}$ or $\{p,t\}+\{q,s\}$

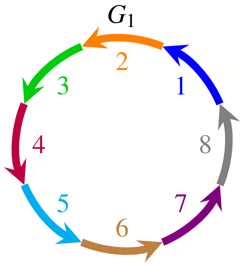


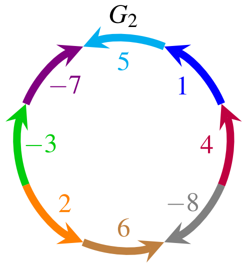
DCJ Distance Problem

- The minimum number of DCJ operations to transform G_1 into G_2

DCJ Distance Problem

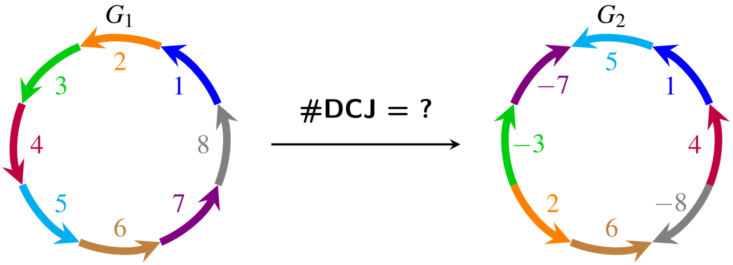
- The minimum number of DCJ operations to transform G_1 into G_2





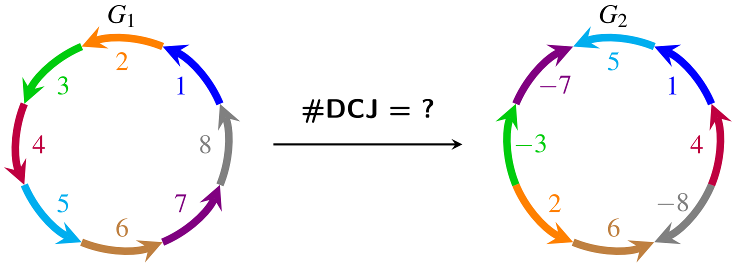
DCJ Distance Problem

- The minimum number of DCJ operations to transform G_1 into G_2



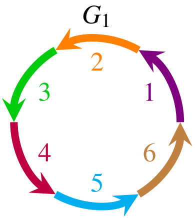
DCJ Distance Problem

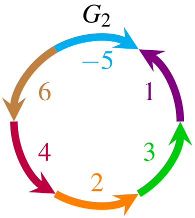
- The minimum number of DCJ operations to transform G_1 into G_2



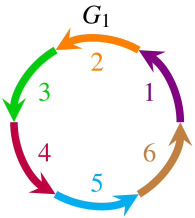
- The DCJ distance can be computed in **linear time** (*Yancopoulos et al., 2005, Bergeron et al., 2006*).

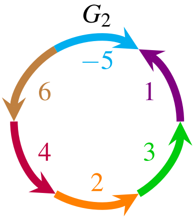
DCJ Distance Formula

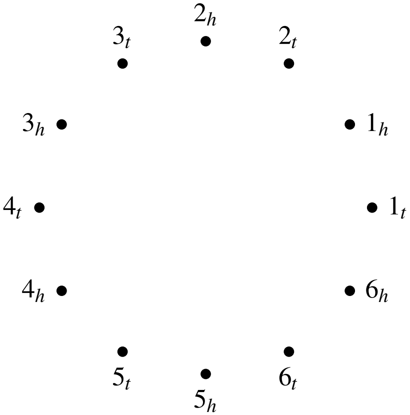




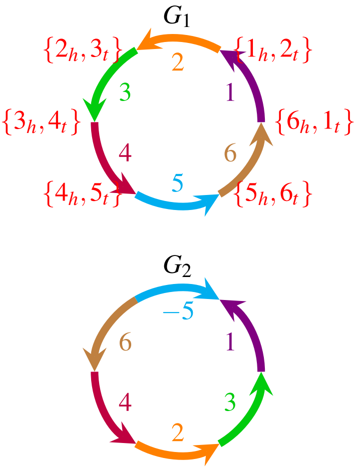
DCJ Distance Formula

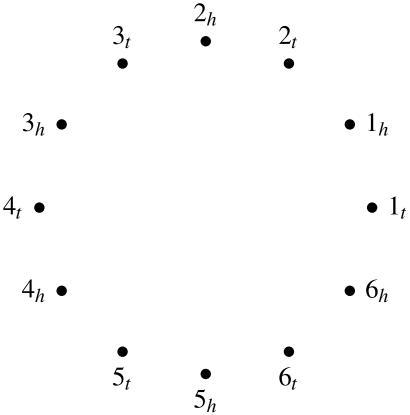




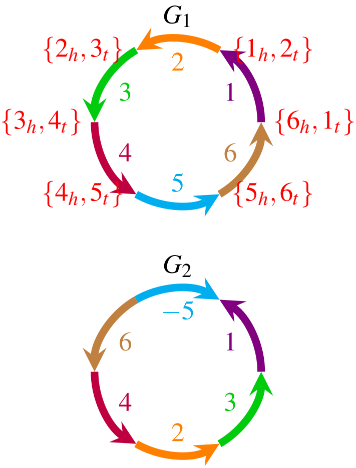


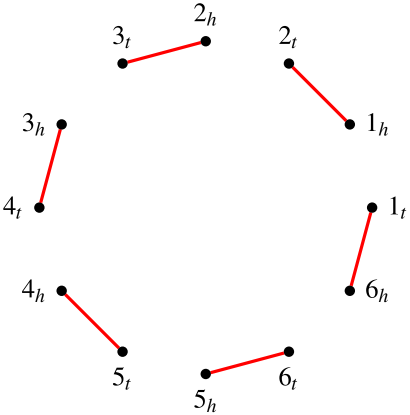
DCJ Distance Formula



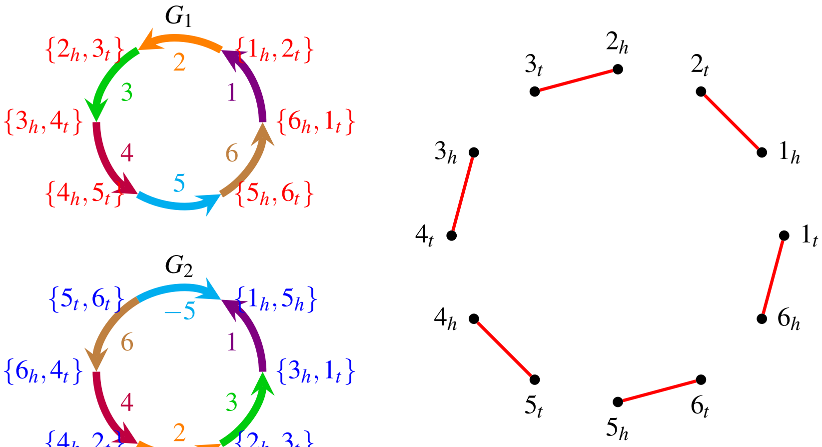


DCJ Distance Formula

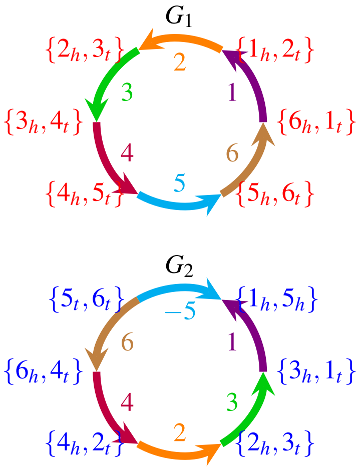


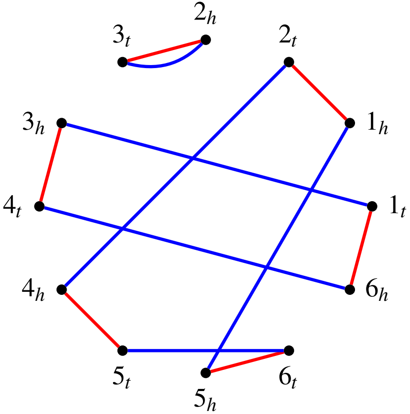


DCJ Distance Formula

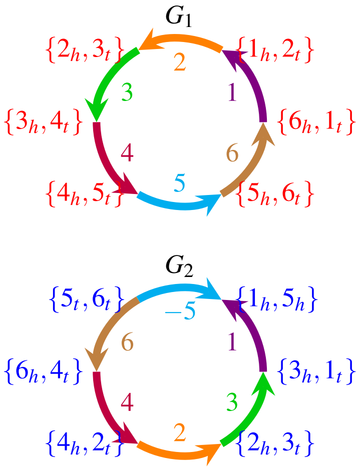


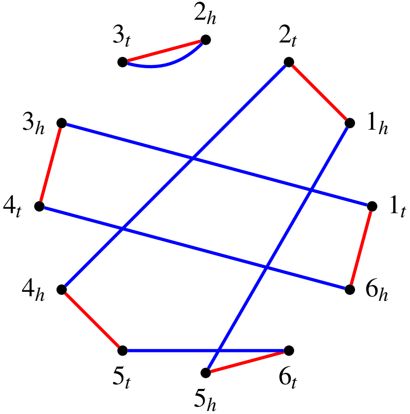
DCJ Distance Formula





DCJ Distance Formula





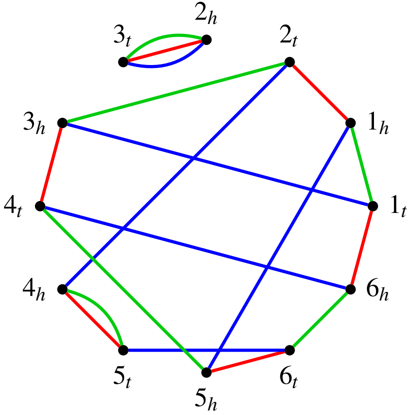
$$d(G_1, G_2) = |V|/2 - c(G_1, G_2)$$

DCJ Median Problem

- Given G_1, G_2 and G_3, compute G to minimize $\sum_{k=1}^{3} d(G, G_k)$

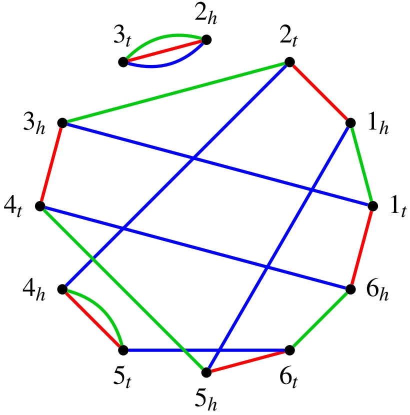
DCJ Median Problem

- Given G_1, G_2 and G_3, compute G to minimize $\sum_{k=1}^{3} d(G, G_k)$



DCJ Median Problem

- Given G_1, G_2 and G_3, compute G to minimize $\sum_{k=1}^{3} d(G, G_k)$



$$\max_{G} \sum_{k=1}^{3} c(G, G_k)$$

DCJ Median Problem: Current Progress

- The median problem has been widely studied and applied on multi-genome comparison and phylogeny construction (*Moret et al., 2002, Eriksen, 2007, Adam et al., 2008*).

DCJ Median Problem: Current Progress

- The median problem has been widely studied and applied on multi-genome comparison and phylogeny construction (*Moret et al., 2002, Eriksen, 2007, Adam et al., 2008*).

- DCJ median problem is NP-hard (*Caprara, 2003, Tannier et al., 2008*).

DCJ Median Problem: Current Progress

- The median problem has been widely studied and applied on multi-genome comparison and phylogeny construction (*Moret et al., 2002, Eriksen, 2007, Adam et al., 2008*).
- DCJ median problem is NP-hard (*Caprara, 2003, Tannier et al., 2008*).
- Previous algorithms:
  - Heuristics: (*Arndt et al. 2008, Rajan et al., 2010*)
  - Exact algorithms: (*Zhang et al., 2009, Xu et al., 2008, 2009*)

DCJ Median Problem: Current Progress

- The median problem has been widely studied and applied on multi-genome comparison and phylogeny construction (*Moret et al., 2002, Eriksen, 2007, Adam et al., 2008*).

- DCJ median problem is NP-hard (*Caprara, 2003, Tannier et al., 2008*).

- Previous algorithms:
  - Heuristics: (*Arndt et al. 2008, Rajan et al., 2010*)
  - Exact algorithms: (*Zhang et al., 2009, Xu et al., 2008, 2009*)

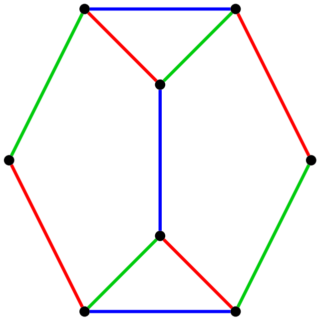
- **Decomposition approach (***Xu et al., 2008, 2009***)**
  - **Adequate subgraph:** optimal substructure
  - **ASMedian:** fast solver using adequate subgraphs

Adequate Subgraph

- We say (G_1, G_2, G_3) is **adequate** if there exists a **certificate** matching G such that $\sum_{k=1}^{3} c(G, G_k) \geq 3 \cdot |V|/4$.

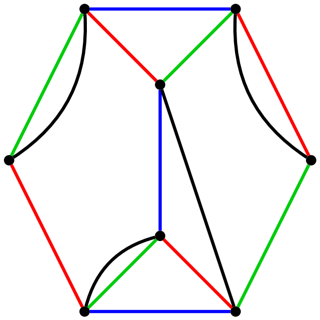
Adequate Subgraph

- We say (G_1, G_2, G_3) is **adequate** if there exists a **certificate** matching G such that $\sum_{k=1}^{3} c(G, G_k) \geq 3 \cdot |V|/4$.



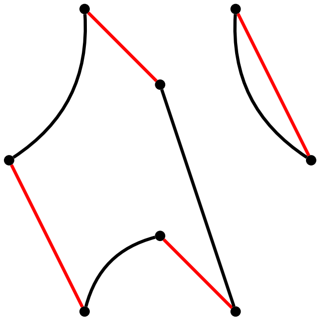
Adequate Subgraph

- We say (G_1, G_2, G_3) is **adequate** if there exists a **certificate** matching G such that $\sum_{k=1}^{3} c(G, G_k) \geq 3 \cdot |V|/4$.



Adequate Subgraph

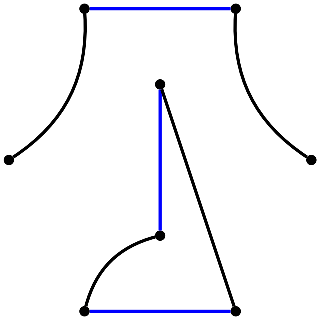
- We say (G_1, G_2, G_3) is **adequate** if there exists a **certificate** matching G such that $\sum_{k=1}^{3} c(G, G_k) \geq 3 \cdot |V|/4$.



$c(G, G_1) = 2$

Adequate Subgraph

- We say (G_1, G_2, G_3) is **adequate** if there exists a **certificate** matching G such that $\sum_{k=1}^{3} c(G, G_k) \geq 3 \cdot |V|/4$.

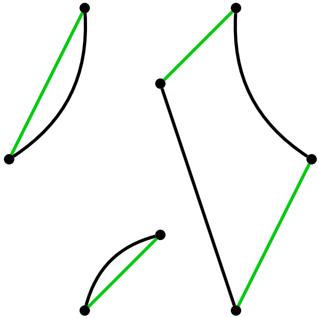


$c(G, G_1) = 2$

$c(G, G_2) = 1$

Adequate Subgraph

- We say (G_1, G_2, G_3) is **adequate** if there exists a **certificate** matching G such that $\sum_{k=1}^{3} c(G, G_k) \geq 3 \cdot |V|/4$.



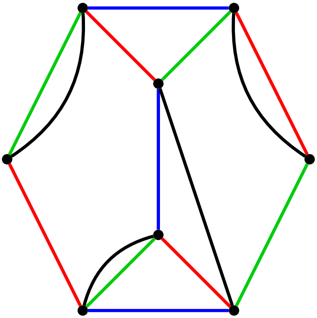
$c(G, G_1) = 2$

$c(G, G_2) = 1$

$c(G, G_3) = 3$

Adequate Subgraph

- We say (G_1, G_2, G_3) is **adequate** if there exists a **certificate** matching G such that $\sum_{k=1}^{3} c(G, G_k) \geq 3 \cdot |V|/4$.



$c(G, G_1) = 2$

$c(G, G_2) = 1$

$c(G, G_3) = 3$

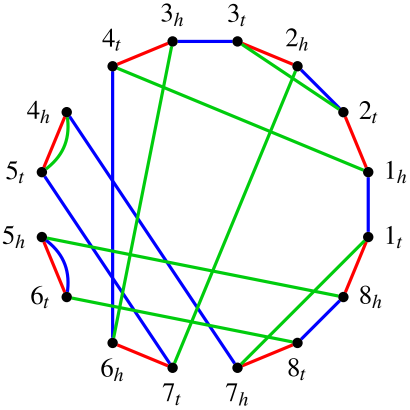
$\sum_{k=1}^{3} c(G, G_k) = 6 \geq 3|V|/4 = 6$

Decomposition Theorem (*Xu et al., 2008*)

- Let $X' = (G'_1, G'_2, G'_3)$ is a subgraph of $X = (G_1, G_2, G_3)$. If X' is adequate, then for any optimal matching G'_0 of X', there exists an optimal matching G_0 of X such that $G'_0 \subset G_0$.

Decomposition Theorem (*Xu et al., 2008*)

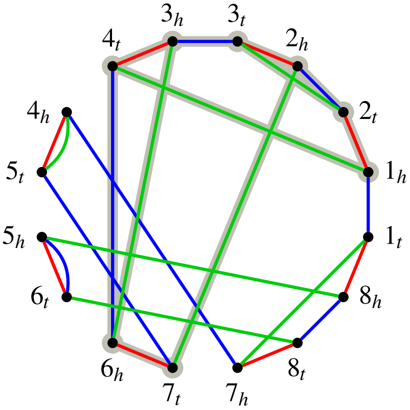
- Let $X' = (G'_1, G'_2, G'_3)$ is a subgraph of $X = (G_1, G_2, G_3)$. If X' is adequate, then for any optimal matching G'_0 of X', there exists an optimal matching G_0 of X such that $G'_0 \subset G_0$.



$X = (G_1, G_2, G_3)$

Decomposition Theorem (*Xu et al., 2008*)

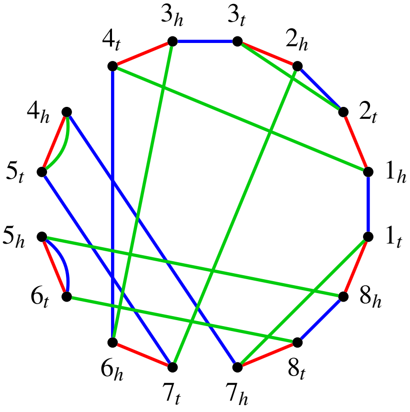
- Let $X' = (G'_1, G'_2, G'_3)$ is a subgraph of $X = (G_1, G_2, G_3)$. If X' is adequate, then for any optimal matching G'_0 of X', there exists an optimal matching G_0 of X such that $G'_0 \subset G_0$.



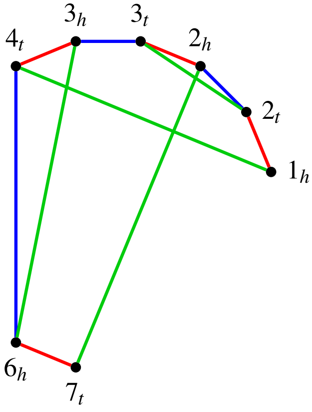
$X = (G_1, G_2, G_3)$

Decomposition Theorem (*Xu et al., 2008*)

- Let $X' = (G'_1, G'_2, G'_3)$ is a subgraph of $X = (G_1, G_2, G_3)$. If X' is adequate, then for any optimal matching G'_0 of X', there exists an optimal matching G_0 of X such that $G'_0 \subset G_0$.



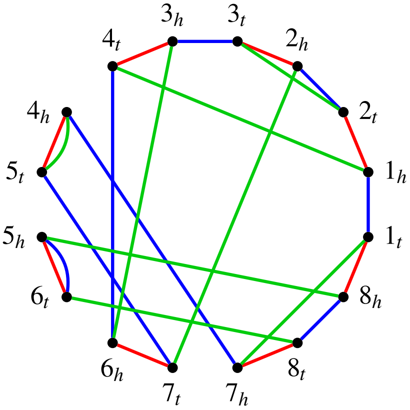
$X = (G_1, G_2, G_3)$



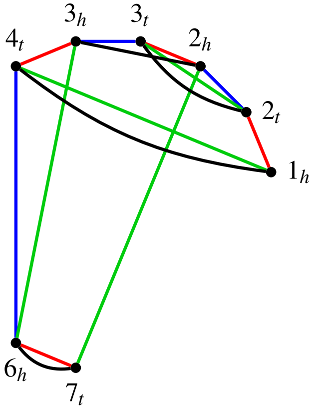
$X' = (G'_1, G'_2, G'_3)$

Decomposition Theorem (*Xu et al., 2008*)

- Let $X' = (G'_1, G'_2, G'_3)$ is a subgraph of $X = (G_1, G_2, G_3)$. If X' is adequate, then for any optimal matching G'_0 of X', there exists an optimal matching G_0 of X such that $G'_0 \subset G_0$.



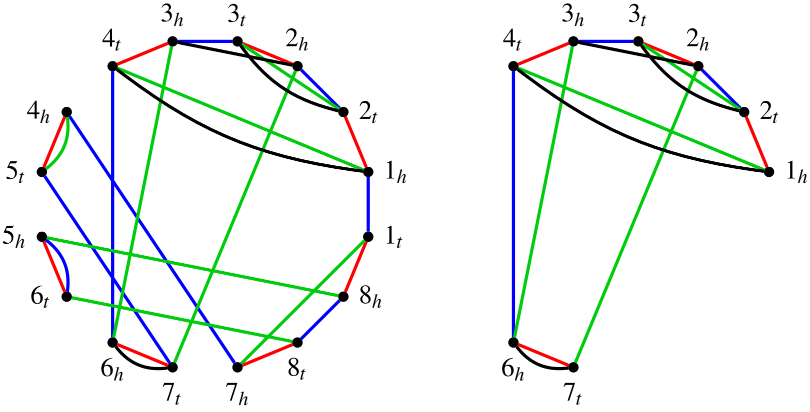
$X = (G_1, G_2, G_3)$



$X' = (G'_1, G'_2, G'_3)$

Decomposition Theorem (*Xu et al., 2008*)

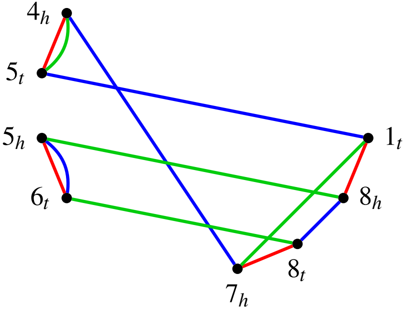
- Let $X' = (G'_1, G'_2, G'_3)$ is a subgraph of $X = (G_1, G_2, G_3)$. If X' is adequate, then for any optimal matching G'_0 of X', there exists an optimal matching G_0 of X such that $G'_0 \subset G_0$.



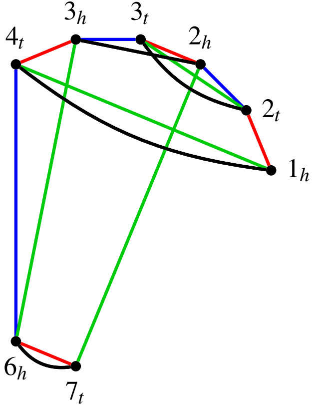
$X = (G_1, G_2, G_3)$ $X' = (G'_1, G'_2, G'_3)$

Decomposition Theorem (*Xu et al., 2008*)

- Let $X' = (G'_1, G'_2, G'_3)$ is a subgraph of $X = (G_1, G_2, G_3)$. If X' is adequate, then for any optimal matching G'_0 of X', there exists an optimal matching G_0 of X such that $G'_0 \subset G_0$.



$X = (G_1, G_2, G_3)$



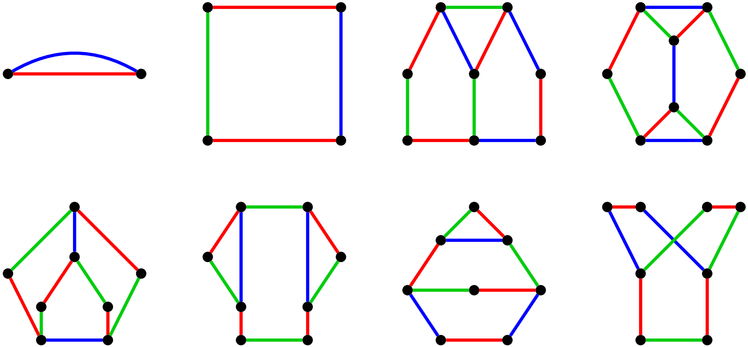
$X' = (G'_1, G'_2, G'_3)$

Finding Adequate Subgraphs

- Xu *et al.* (*2008*) enumerate adequate subgraphs with $|V| \leq 8$.

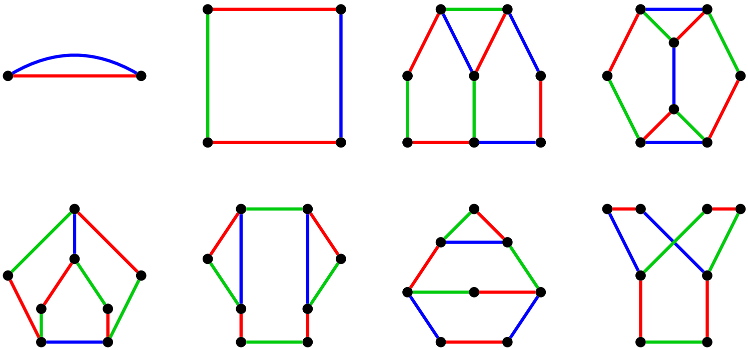
Finding Adequate Subgraphs

- Xu *et al.* (*2008*) enumerate adequate subgraphs with $|V| \leq 8$.



Finding Adequate Subgraphs

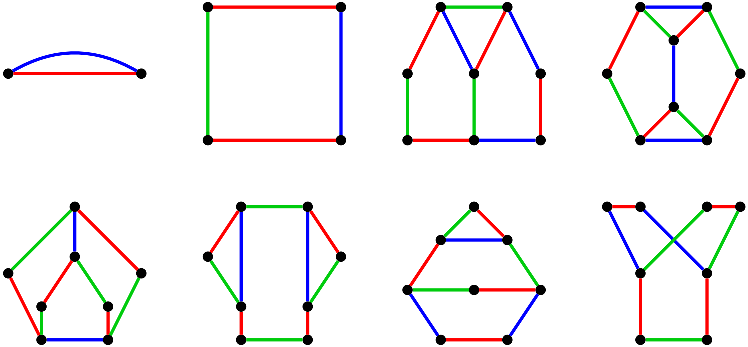
- Xu *et al.* (*2008*) enumerate adequate subgraphs with $|V| \leq 8$.



- Subgraph isomorphism problem is NP-complete.

Finding Adequate Subgraphs

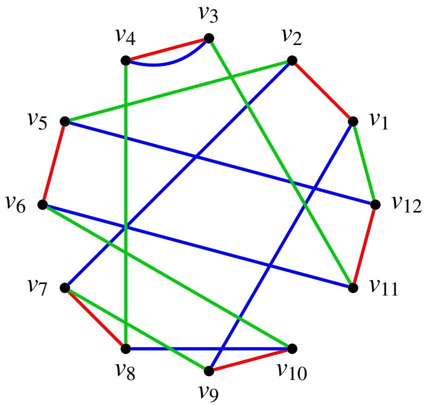
- Xu *et al.* (*2008*) enumerate adequate subgraphs with $|V| \leq 8$.



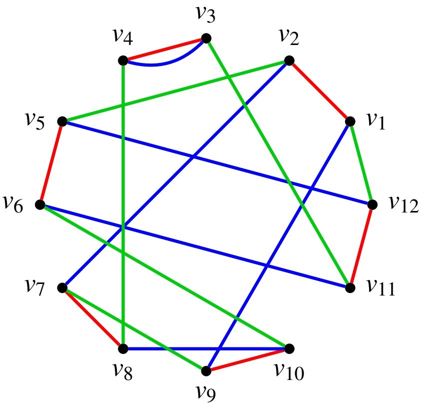
- Subgraph isomorphism problem is NP-complete.
- There are infinite many adequate subgraphs.

Our Algorithm to Find Adequate Subgraphs

Our Algorithm to Find Adequate Subgraphs

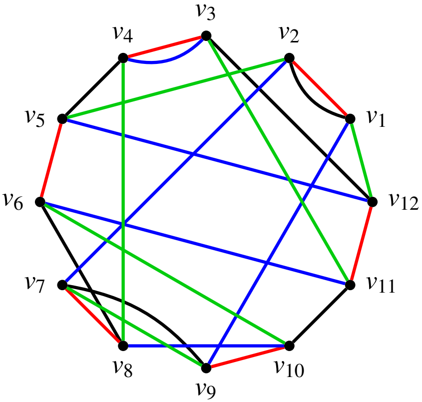


Our Algorithm to Find Adequate Subgraphs



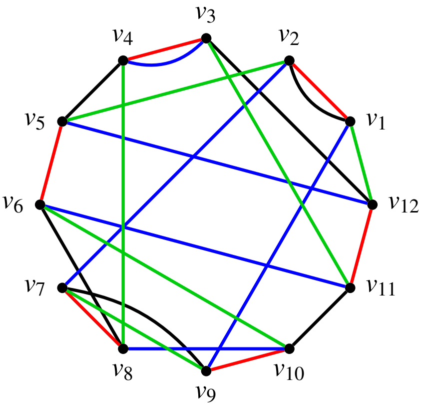
- Assume that a perfect matching G is given.

Our Algorithm to Find Adequate Subgraphs



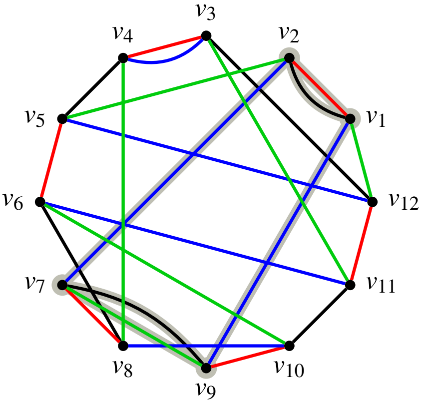
- Assume that a perfect matching G is given.

Our Algorithm to Find Adequate Subgraphs



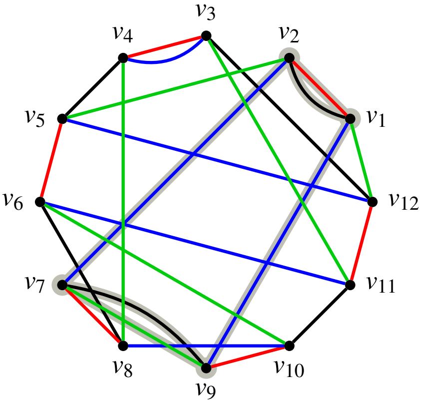
- Assume that a perfect matching G is given.
- **Problem:** Decide whether there exists adequate subgraphs using $G' \subset G$ as certificate.

Our Algorithm to Find Adequate Subgraphs



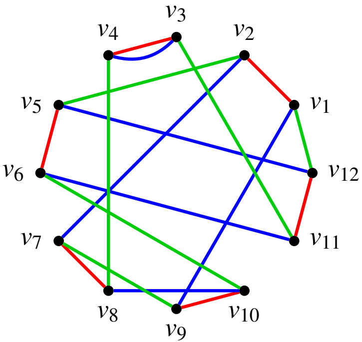
- Assume that a perfect matching G is given.
- **Problem:** Decide whether there exists adequate subgraphs using $G' \subset G$ as certificate.

Our Algorithm to Find Adequate Subgraphs

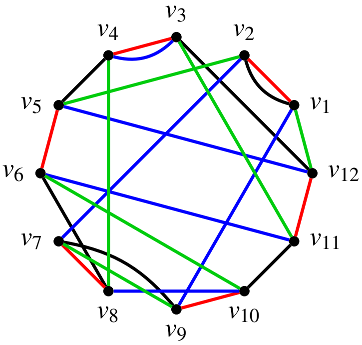


- Assume that a perfect matching G is given.
- **Problem:** Decide whether there exists adequate subgraphs using $G' \subset G$ as certificate.
- This problem can be formulated as a **network flow** problem.

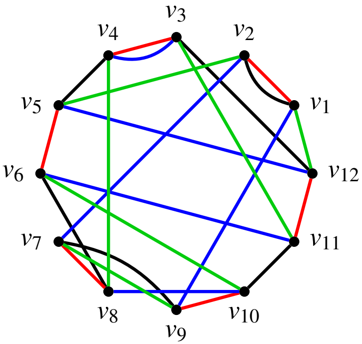
Network Construction



Network Construction

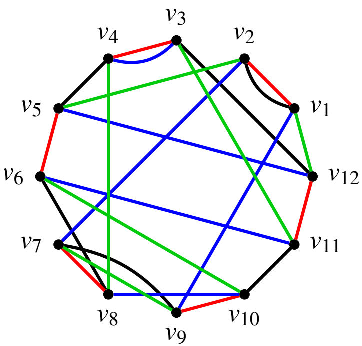


Network Construction



- Add one vertex for each extremity.

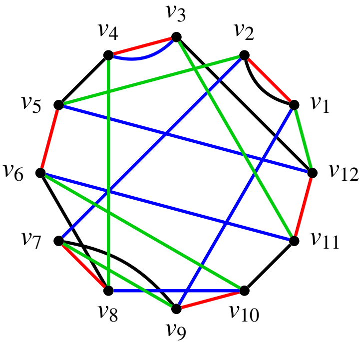
Network Construction



① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫

- Add one vertex for each extremity.

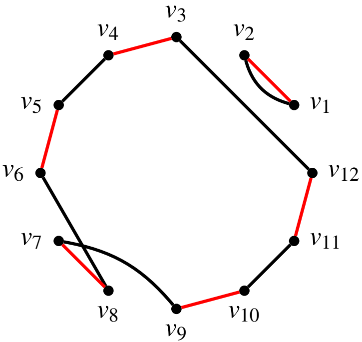
Network Construction



① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫

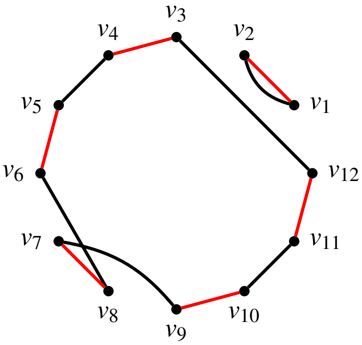
- Add one vertex for each extremity.
- Add one vertex for each cycle, and connect it to the corresponding extremities with infinite capacity.

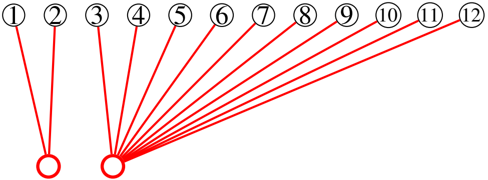
Network Construction



- Add one vertex for each extremity.
- Add one vertex for each cycle, and connect it to the corresponding extremities with infinite capacity.

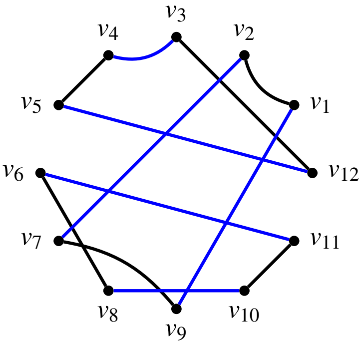
Network Construction

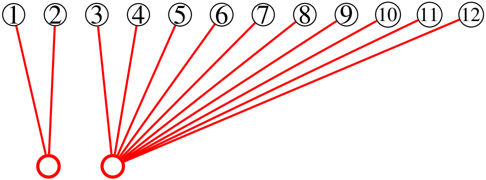




- Add one vertex for each extremity.
- Add one vertex for each cycle, and connect it to the corresponding extremities with infinite capacity.

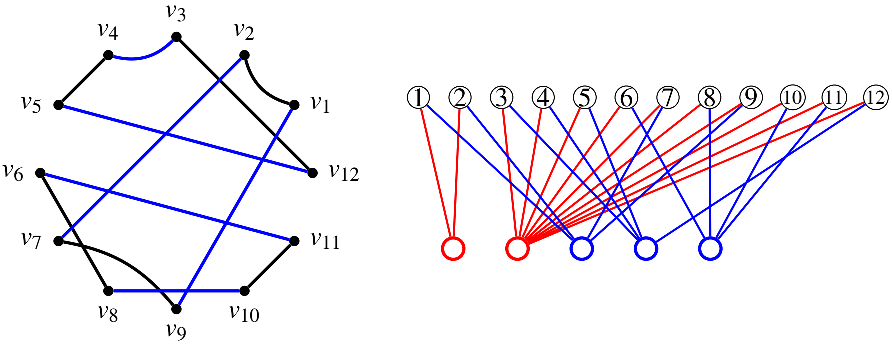
Network Construction





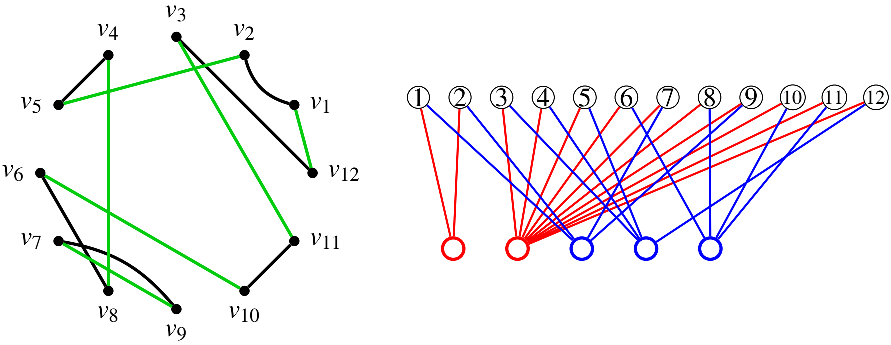
- Add one vertex for each extremity.
- Add one vertex for each cycle, and connect it to the corresponding extremities with infinite capacity.

Network Construction



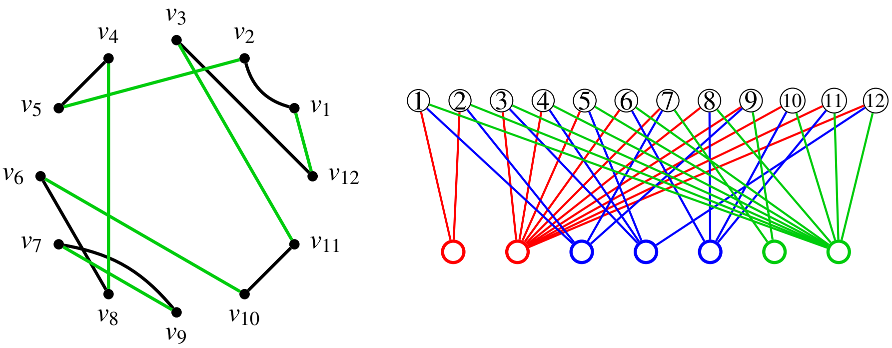
- Add one vertex for each extremity.
- Add one vertex for each cycle, and connect it to the corresponding extremities with infinite capacity.

Network Construction



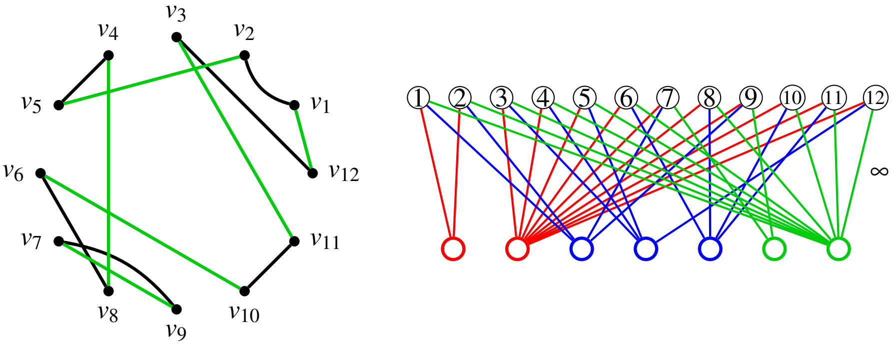
- Add one vertex for each extremity.
- Add one vertex for each cycle, and connect it to the corresponding extremities with infinite capacity.

Network Construction



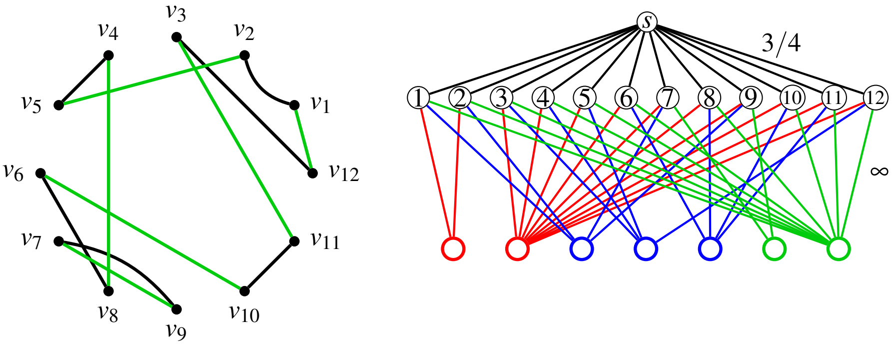
- Add one vertex for each extremity.
- Add one vertex for each cycle, and connect it to the corresponding extremities with infinite capacity.

Network Construction



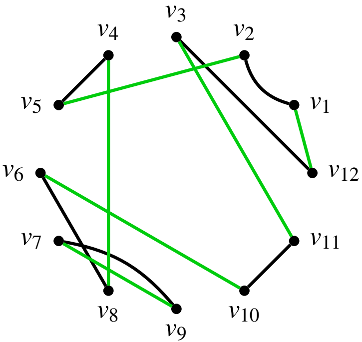
- Add one vertex for each extremity.
- Add one vertex for each cycle, and connect it to the corresponding extremities with infinite capacity.

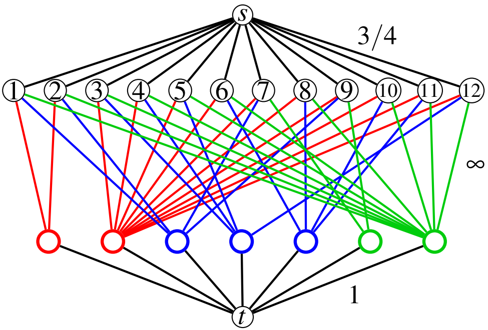
Network Construction



- Add one vertex for each extremity.
- Add one vertex for each cycle, and connect it to the corresponding extremities with infinite capacity.
- Add one source s, and connect it to all extremities with capacity of $3/4$.

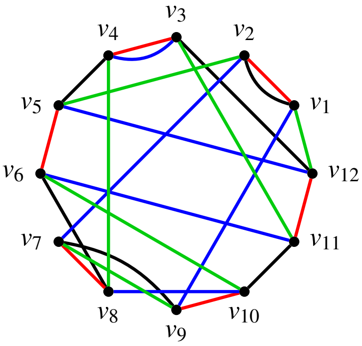
Network Construction

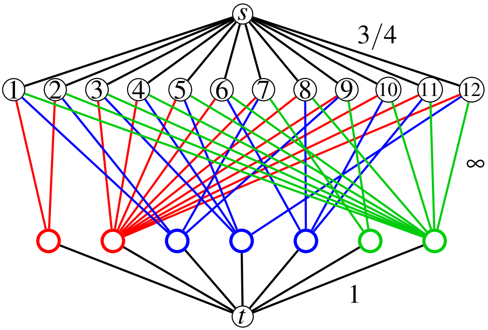




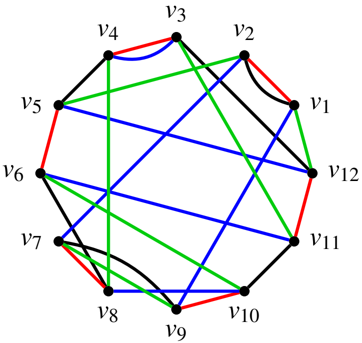
- Add one vertex for each extremity.
- Add one vertex for each cycle, and connect it to the corresponding extremities with infinite capacity.
- Add one source s, and connect it to all extremities with capacity of $3/4$.
- Add one sink t, and connect it to all cycles with unit capacity.

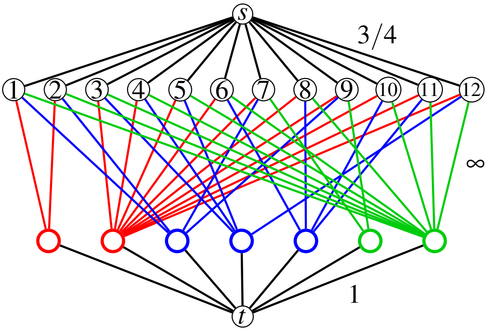
Decide and Compute the Adequate Subgraphs





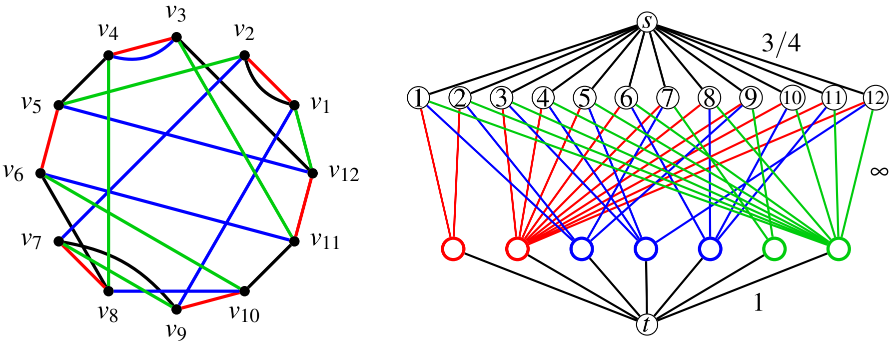
Decide and Compute the Adequate Subgraphs





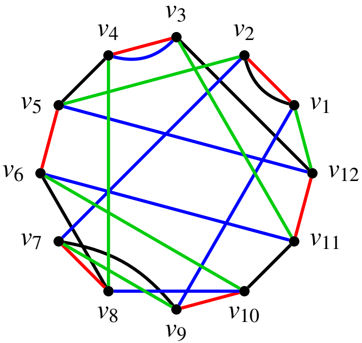
- Compute max s-t flow f, and the residual graph N_f w.r.t. f.

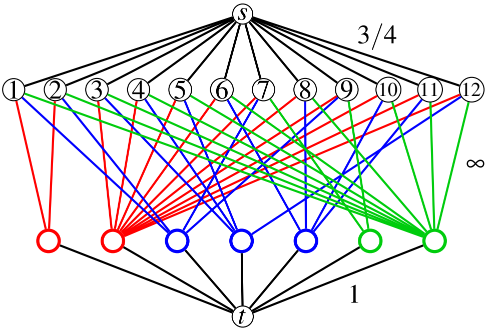
Decide and Compute the Adequate Subgraphs



- Compute max s-t flow f, and the residual graph N_f w.r.t. f.
- Let T be the set of vertices that can reach t in N_f.

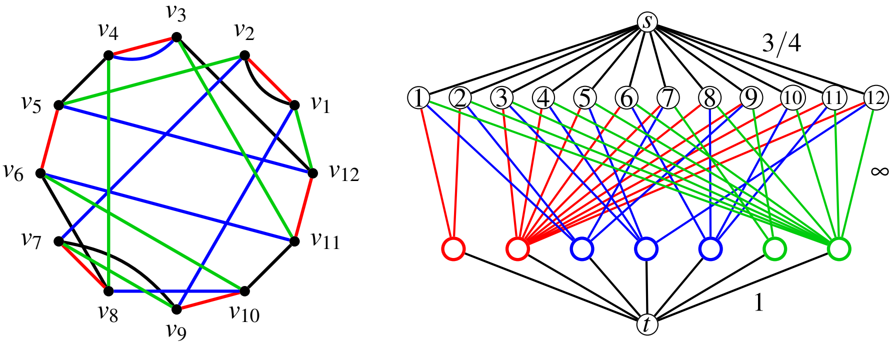
Decide and Compute the Adequate Subgraphs





- Compute max s-t flow f, and the residual graph N_f w.r.t. f.
- Let T be the set of vertices that can reach t in N_f.
- **Theorem:** There exists adequate subgraphs with $G' \subset G$ as certificate if and only if $\{t\} \subsetneq T$.

Decide and Compute the Adequate Subgraphs



- Compute max s-t flow f, and the residual graph N_f w.r.t. f.
- Let T be the set of vertices that can reach t in N_f.
- **Theorem:** There exists adequate subgraphs with $G' \subset G$ as certificate if and only if $\{t\} \subsetneq T$.
- If such an adequate subgraph exists, we can also retrieve one from N_f.

Bounds of the Median Score

- **Median:** $G_0 := \arg\max_G \sum_{k=1}^{3} d(G, G_k)$

Bounds of the Median Score

- **Median:** $G_0 := \arg\max_G \sum_{k=1}^{3} d(G, G_k)$
- **Median distance:** $d_m := \sum_{k=1}^{3} d(G_0, G_k)$

Bounds of the Median Score

- **Median:** $G_0 := \arg\max_G \sum_{k=1}^{3} d(G, G_k)$
- **Median distance:** $d_m := \sum_{k=1}^{3} d(G_0, G_k)$
- **Triangle distance:** $d_t := d(G_1, G_2) + d(G_1, G_3) + d(G_2, G_3)$

Bounds of the Median Score

- **Median:** $G_0 := \arg\max_G \sum_{k=1}^{3} d(G, G_k)$
- **Median distance:** $d_m := \sum_{k=1}^{3} d(G_0, G_k)$
- **Triangle distance:** $d_t := d(G_1, G_2) + d(G_1, G_3) + d(G_2, G_3)$
- **Lower bound:**

$$\left.\begin{array}{l} d(G_1,G_2) \leq d(G_0,G_1) + d(G_0,G_2) \\ d(G_1,G_3) \leq d(G_0,G_1) + d(G_0,G_3) \\ d(G_2,G_3) \leq d(G_0,G_2) + d(G_0,G_3) \end{array}\right\} \quad \Longrightarrow \quad d_m \geq d_t/2$$

Bounds of the Median Score

- **Median:** $G_0 := \arg\max_G \sum_{k=1}^{3} d(G, G_k)$
- **Median distance:** $d_m := \sum_{k=1}^{3} d(G_0, G_k)$
- **Triangle distance:** $d_t := d(G_1, G_2) + d(G_1, G_3) + d(G_2, G_3)$
- **Lower bound:**

$$\left.\begin{array}{l} d(G_1,G_2) \leq d(G_0,G_1) + d(G_0,G_2) \\ d(G_1,G_3) \leq d(G_0,G_1) + d(G_0,G_3) \\ d(G_2,G_3) \leq d(G_0,G_2) + d(G_0,G_3) \end{array}\right\} \quad \Longrightarrow \quad d_m \geq d_t/2$$

- **Upper bound:**

$$\left.\begin{array}{l} d_m \leq d(G_1,G_2) + d(G_1,G_3) \\ d_m \leq d(G_2,G_1) + d(G_2,G_3) \\ d_m \leq d(G_3,G_1) + d(G_3,G_2) \end{array}\right\} \quad \Longrightarrow \quad d_m \leq 2 \cdot d_t/3$$

Tightness of the Bounds

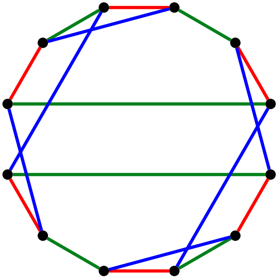
- The lower bound is tight.

Tightness of the Bounds

- The lower bound is tight.
- We prove that upper bound is also tight.

Tightness of the Bounds

- The lower bound is tight.
- We prove that upper bound is also tight.



This graph is adequate, thus we can build an example of arbitray size.

Deciding the Equality to the Bounds

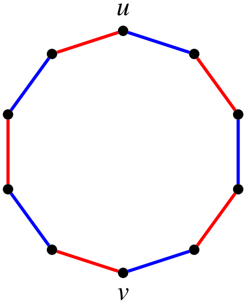
- Given (G_1, G_2, G_3), decide whether $d_m = d_t/2$ or $d_m = 2 \cdot d_t/3$.

Deciding the Equality to the Bounds

- Given (G_1, G_2, G_3), decide whether $d_m = d_t/2$ or $d_m = 2 \cdot d_t/3$.
- We say (u, v) is **strong** w.r.t. (G_1, G_2) if it can increase $c(G_1, G_2)$ by 2.

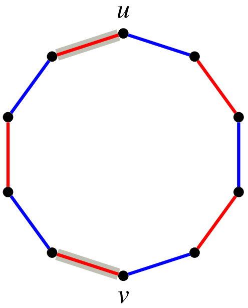
Deciding the Equality to the Bounds

- Given (G_1, G_2, G_3), decide whether $d_m = d_t/2$ or $d_m = 2 \cdot d_t/3$.
- We say (u, v) is **strong** w.r.t. (G_1, G_2) if it can increase $c(G_1, G_2)$ by 2.



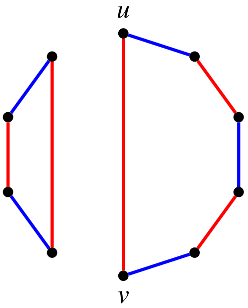
Deciding the Equality to the Bounds

- Given (G_1, G_2, G_3), decide whether $d_m = d_t/2$ or $d_m = 2 \cdot d_t/3$.
- We say (u, v) is **strong** w.r.t. (G_1, G_2) if it can increase $c(G_1, G_2)$ by 2.



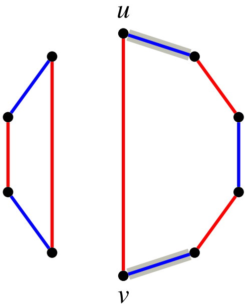
Deciding the Equality to the Bounds

- Given (G_1, G_2, G_3), decide whether $d_m = d_t/2$ or $d_m = 2 \cdot d_t/3$.
- We say (u, v) is **strong** w.r.t. (G_1, G_2) if it can increase $c(G_1, G_2)$ by 2.



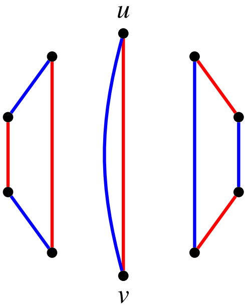
Deciding the Equality to the Bounds

- Given (G_1, G_2, G_3), decide whether $d_m = d_t/2$ or $d_m = 2 \cdot d_t/3$.
- We say (u, v) is **strong** w.r.t. (G_1, G_2) if it can increase $c(G_1, G_2)$ by 2.



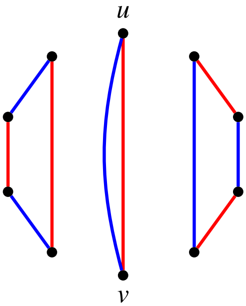
Deciding the Equality to the Bounds

- Given (G_1, G_2, G_3), decide whether $d_m = d_t/2$ or $d_m = 2 \cdot d_t/3$.
- We say (u, v) is **strong** w.r.t. (G_1, G_2) if it can increase $c(G_1, G_2)$ by 2.



Deciding the Equality to the Bounds

- Given (G_1, G_2, G_3), decide whether $d_m = d_t/2$ or $d_m = 2 \cdot d_t/3$.
- We say (u, v) is **strong** w.r.t. (G_1, G_2) if it can increase $c(G_1, G_2)$ by 2.



- We say (u, v) is **strong** w.r.t. (G_1, G_2, G_3) if it is strong w.r.t. (G_1, G_2), (G_1, G_3) and (G_2, G_3).

Conditions for the Equality to the Bounds

- **Theorem:** We have $d_m = d_t/2$ if and only if there exists $|V|/2$ independent strong pairs w.r.t. (G_1, G_2, G_3).

Conditions for the Equality to the Bounds

- **Theorem:** We have $d_m = d_t/2$ if and only if there exists $|V|/2$ independent strong pairs w.r.t. (G_1, G_2, G_3).

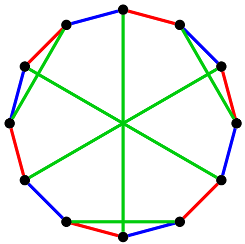
- **Theorem:** We have $d_m = 2 \cdot d_t/3$ only if there is no strong pair w.r.t. (G_1, G_2, G_3).

Conditions for the Equality to the Bounds

- **Theorem:** We have $d_m = d_t/2$ if and only if there exists $|V|/2$ independent strong pairs w.r.t. (G_1, G_2, G_3).
- **Theorem:** We have $d_m = 2 \cdot d_t/3$ only if there is no strong pair w.r.t. (G_1, G_2, G_3).
- The above condition is not sufficient.

Conditions for the Equality to the Bounds

- **Theorem:** We have $d_m = d_t/2$ if and only if there exists $|V|/2$ independent strong pairs w.r.t. (G_1, G_2, G_3).
- **Theorem:** We have $d_m = 2 \cdot d_t/3$ only if there is no strong pair w.r.t. (G_1, G_2, G_3).
- The above condition is not sufficient.



Conclusion

- We proposed a new algorithm to find adequate subgraphs using a network flow formulation.

Conclusion

- We proposed a new algorithm to find adequate subgraphs using a network flow formulation.
- We showed that the upper bound of the median distance is also tight.

Conclusion

- We proposed a new algorithm to find adequate subgraphs using a network flow formulation.
- We showed that the upper bound of the median distance is also tight.
- We gave some conditions to decide whether the median distance can reach its lower or upper bounds.

Thank You!