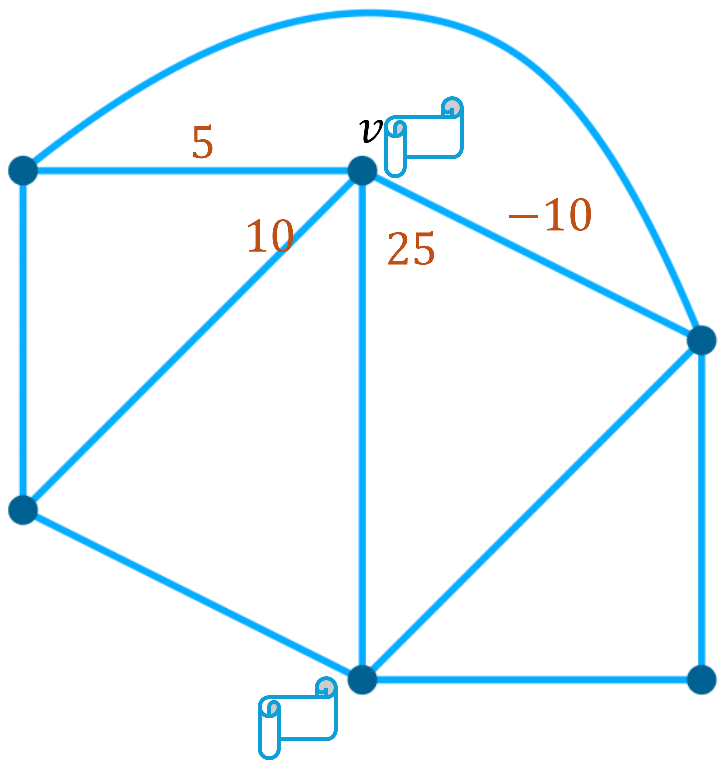
Distributed Maximum Flow in Planar Graphs

Yaseen Abd-Elhaleem[1],
Michal Dory[1], Merav Parter[2] and Oren Weimann[1]

[1]University of Haifa
[2]Weizmann Institute of Science

CONGEST

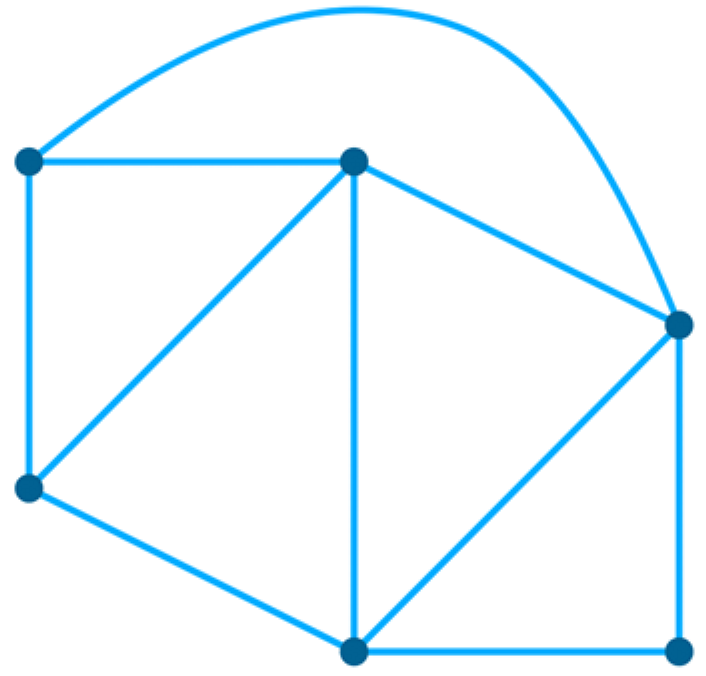
- Vertices = computational units
- Unique $\Theta(\log n)$-bit ID for vertex
- Synchronous rounds
- Message size = $\Theta(\log n)$-bits
- Complexity = #rounds
- Graph is planar
- $D \coloneqq G$'s hop-diameter
- I/O is local



Motivation

- In general graphs, for various problems (e.g., Max-Flow, MST, Min-Cut):
  - $\tilde{O}(D + \sqrt{n})$ upper bound
  - $\tilde{\Omega}(D + \sqrt{n})$ existential lower bound

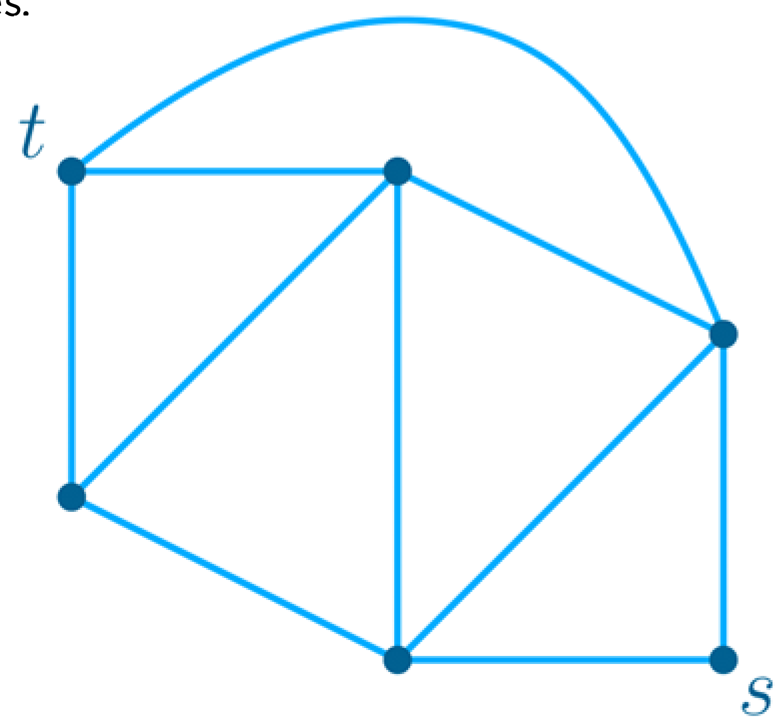
- Planar graphs:
  - Circumvent lower bound topology
  - Rich structure



Maximum st-flow

Find max-flow that can be pushed from s to t in graph with edge capacities.

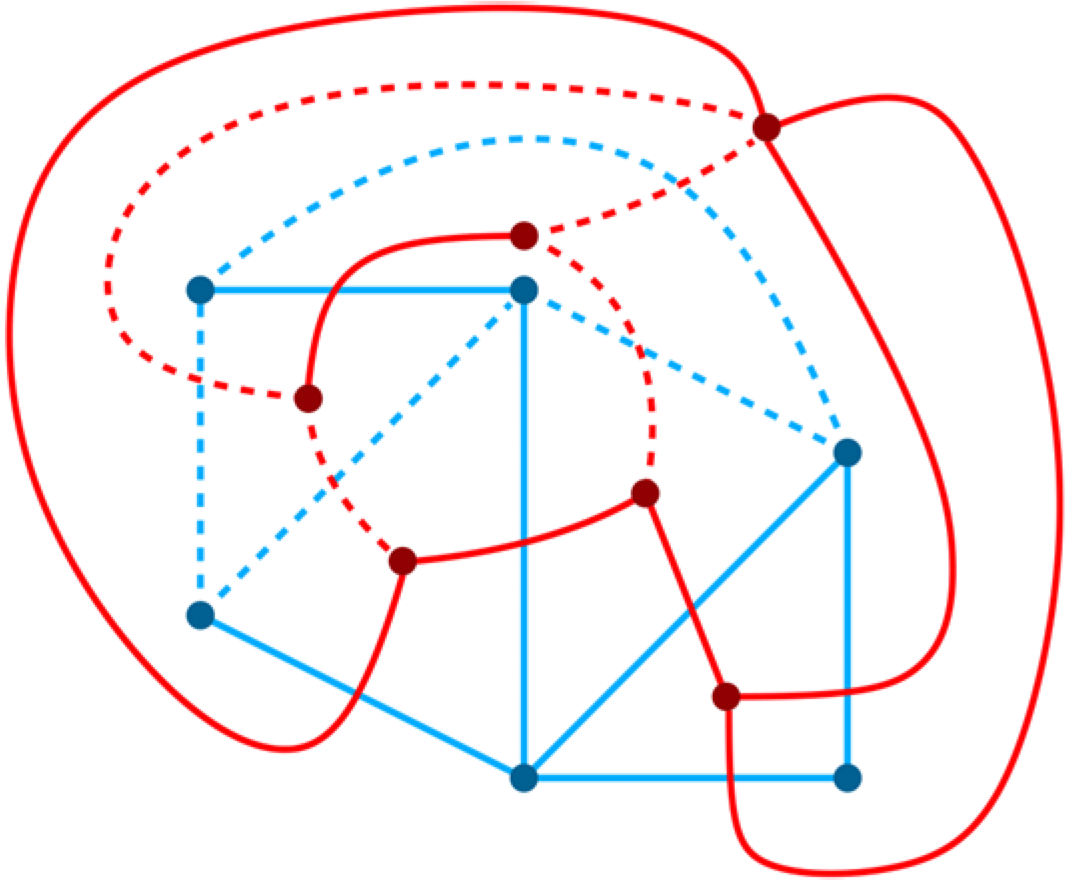
This work	Planar	Directed	Exact	$\tilde{O}(D^2)$
	Planar	Undirected, s, t on same face	$1 - \epsilon$ approximation	$D \cdot n^{o(1)}$
[GKK+'15]	General	Undirected	$1 + o(1)$ approximation	$(D + \sqrt{n})n^{o(1)}$
[Vos'23]	Planar	Directed	Exact	$D \cdot n^{1/2 + o(1)}$



General approach: *duality*!

- The dual graph G^*.
  - Faces of G are nodes in G^*
  - Adjacent faces of G are connected in G^*

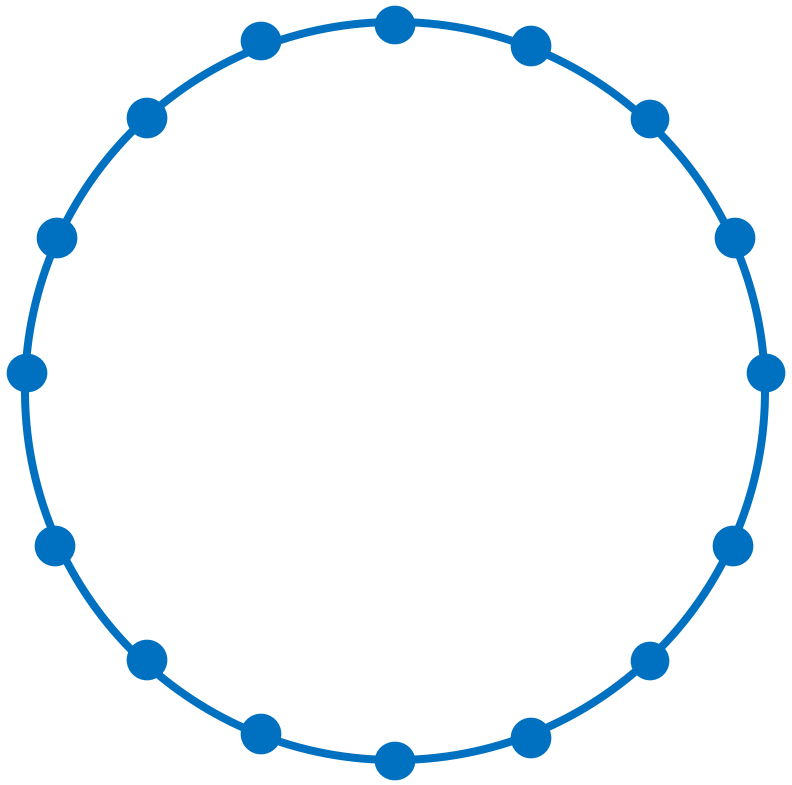
- Properties.
  - Cuts in G^* are cycles in G
  - Distances in G^* imply flow in G



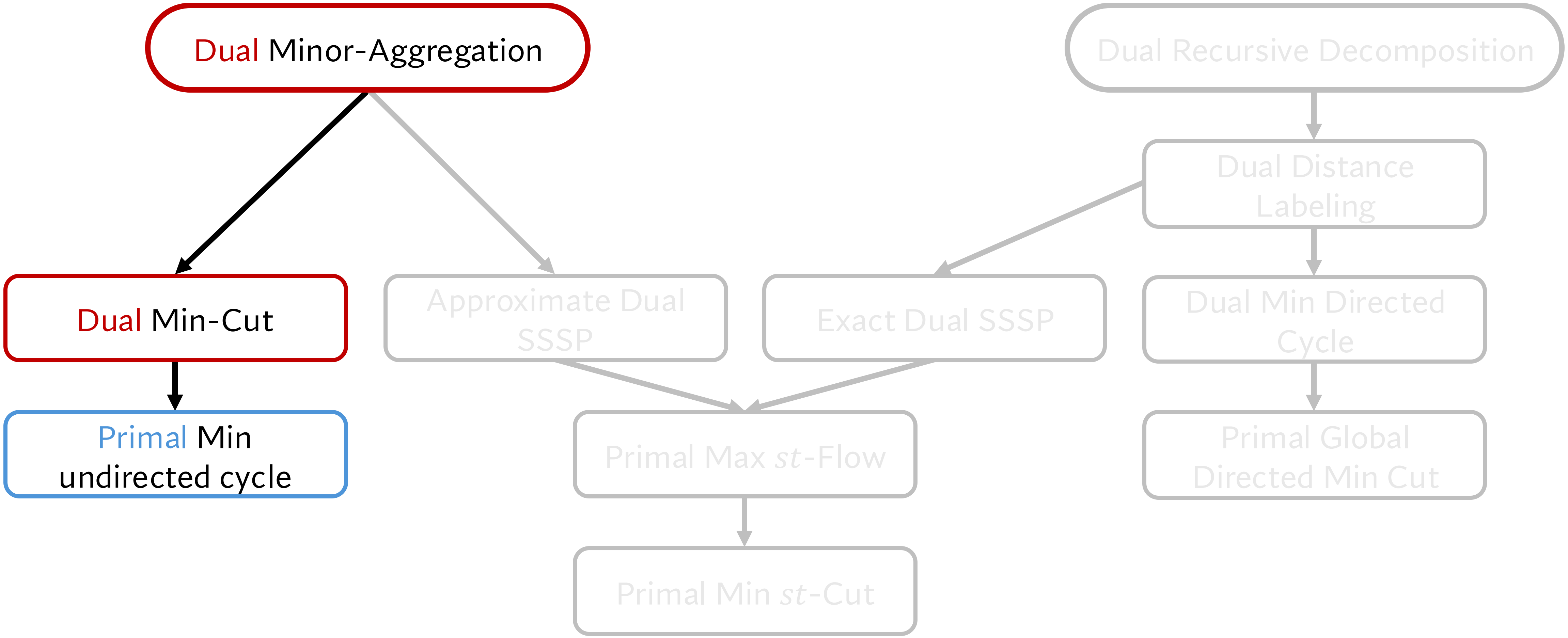
Challenges

- G^* is not the communication network.
  - Simulate a face?

- The hop-diameter of G^* might be much larger than that of G

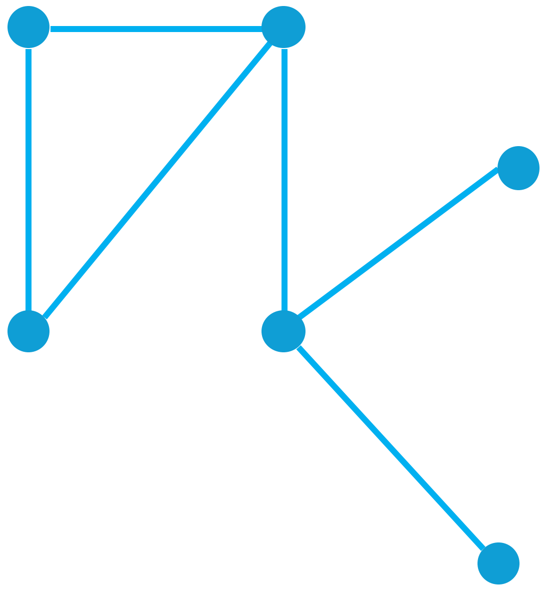


Warm-up: Minimum Weight Cycle



Minor-Aggregation Model

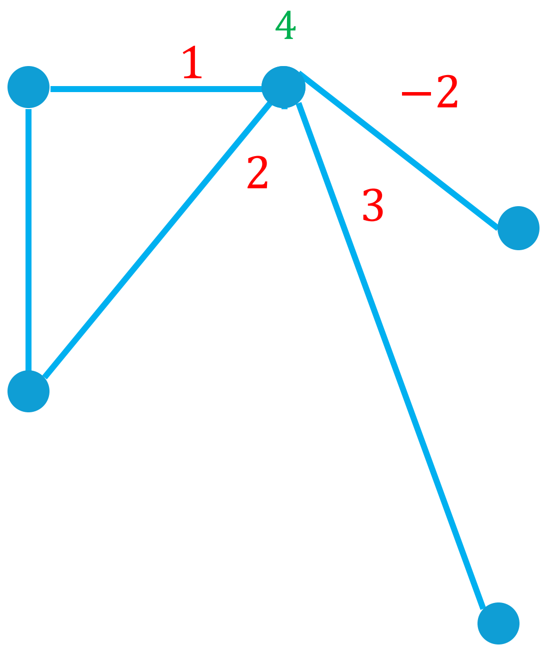
- Round:
  - contract some edges
  - compute aggregate operators over vertices' leaving edges
- Aggregate operator:
  - Input: two ℓ-bit strings, a, b
  - Output: a single ℓ-bit string $a \oplus b$
  - E.g. $\min$, $+$, $\arg\max$.



Minor-Aggregation Model

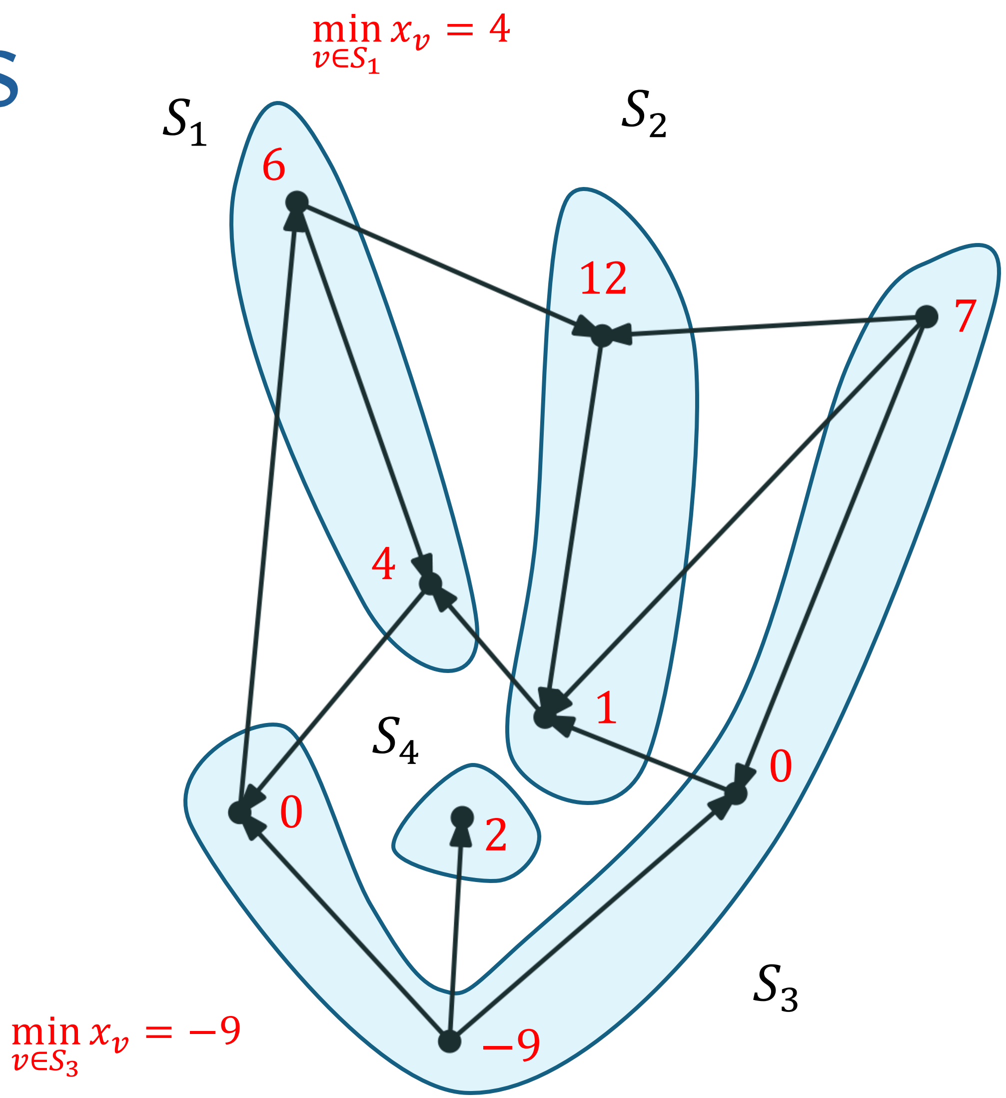
- Round:
  - contract some edges
  - compute aggregate operators over vertices' leaving edges

- Aggregate operator:
  - Input: two ℓ-bit strings, a, b
  - Output: a single ℓ-bit string $a \oplus b$
  - E.g. min, $+$, arg max.



Primal Minor-Aggregations

- Part-wise aggregation.
  - Input: disjoint connected subgraphs $S_1, S_2, \ldots$; and inputs x_v for $v \in G$
  - Output: vertices in S_i know $\bigoplus_{v \in S_i} x_v$ for all i
  - $\tilde{O}(D)$ rounds via low-congestion shortcuts [GH'16] in planar graphs
- Minor-aggregation round = $\tilde{O}(1)$ part-wise aggregations [ZGY+'22] = $\tilde{O}(D)$ rounds

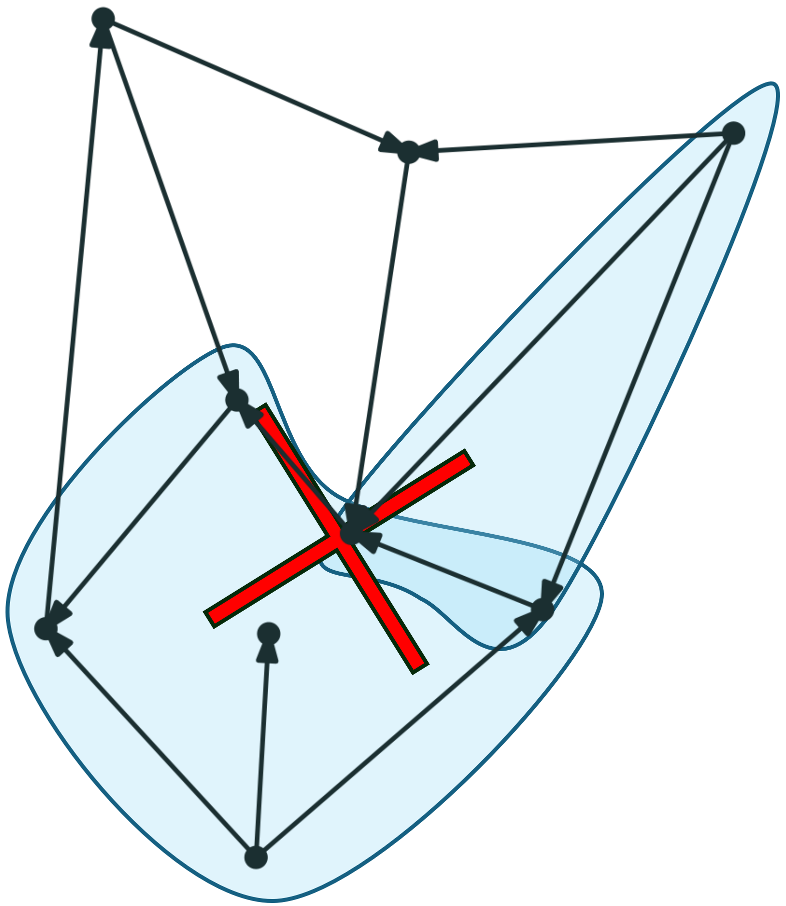


Dual Minor-Aggregations

- Aggregation on faces?
  - Set each S_i to be a face?
  - Faces are not disjoint

  $\Rightarrow$Challenge

- Solution: related graph $\hat{G}$ [GP'17]
  - faces of G = disjoint subgraphs of $\hat{G}$
  - $\hat{G}$ has small diameter
  - $\hat{G}$ is planar

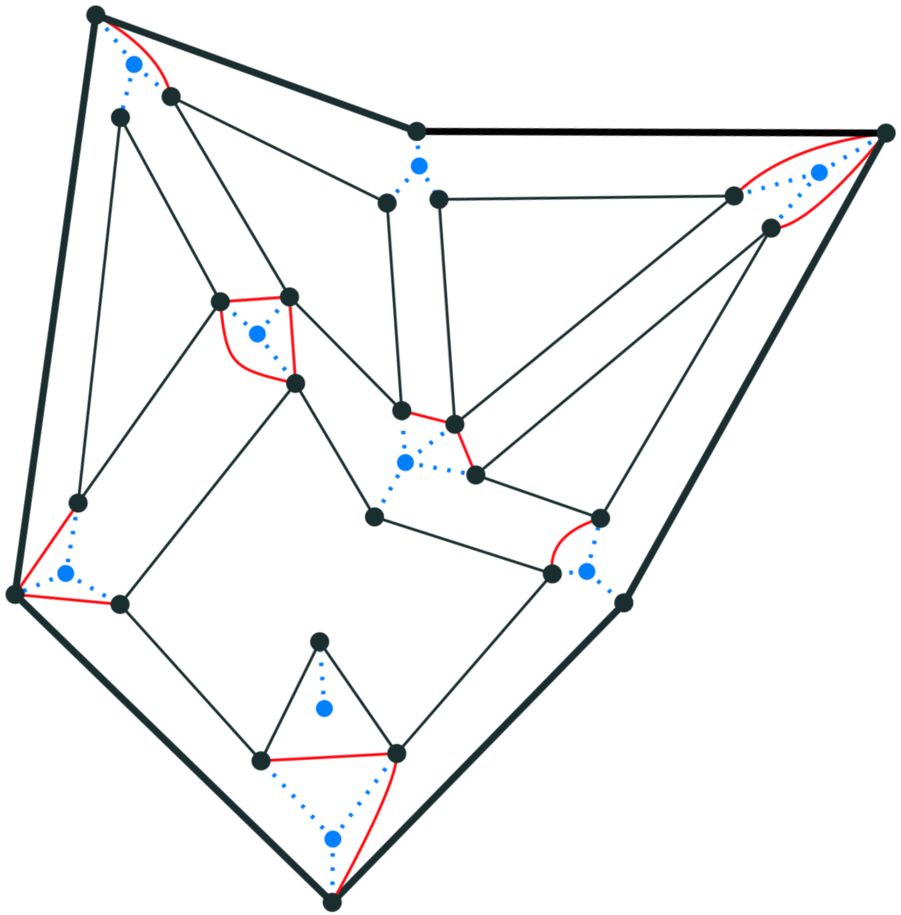


Dual Minor-Aggregations

- Aggregation on faces?
  - Set each S_i to be a face?
  - Faces are not disjoint

  $\Rightarrow$Challenge

- Solution: related graph $\hat{G}$ [GP'17]
  - faces of G = disjoint subgraphs of $\hat{G}$
  - $\hat{G}$ has small diameter
  - $\hat{G}$ is planar



Dual Minor-Aggregations

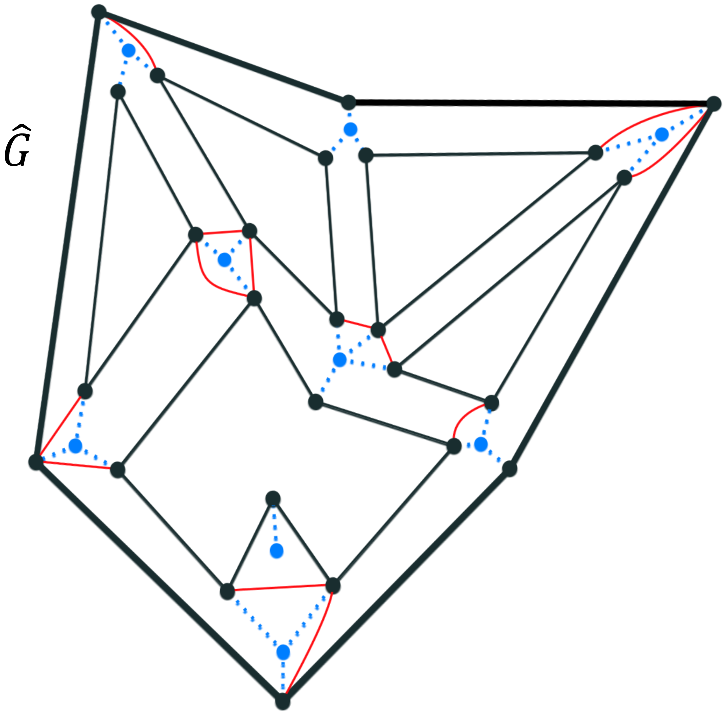
- [GP'17] implement aggregations on the dual for special cases with $\hat{G}$ in $\tilde{O}(D)$ rounds.
- We generalize $\hat{G}$ to allow aggregations over edges of the dual.

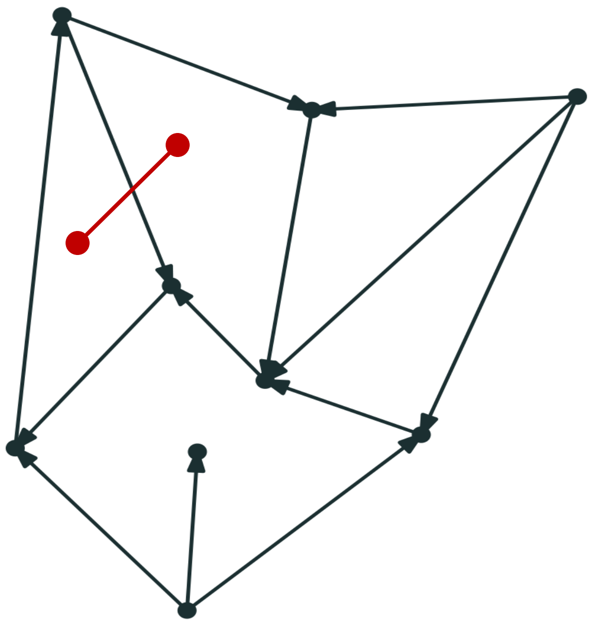
Can simulate $\tilde{O}(1)$-round minor aggregation algorithms on G^* in $\tilde{O}(D)$ rounds.

$\tilde{O}(1)$ round minor-aggregation min-cut algorithm [GZ'22]

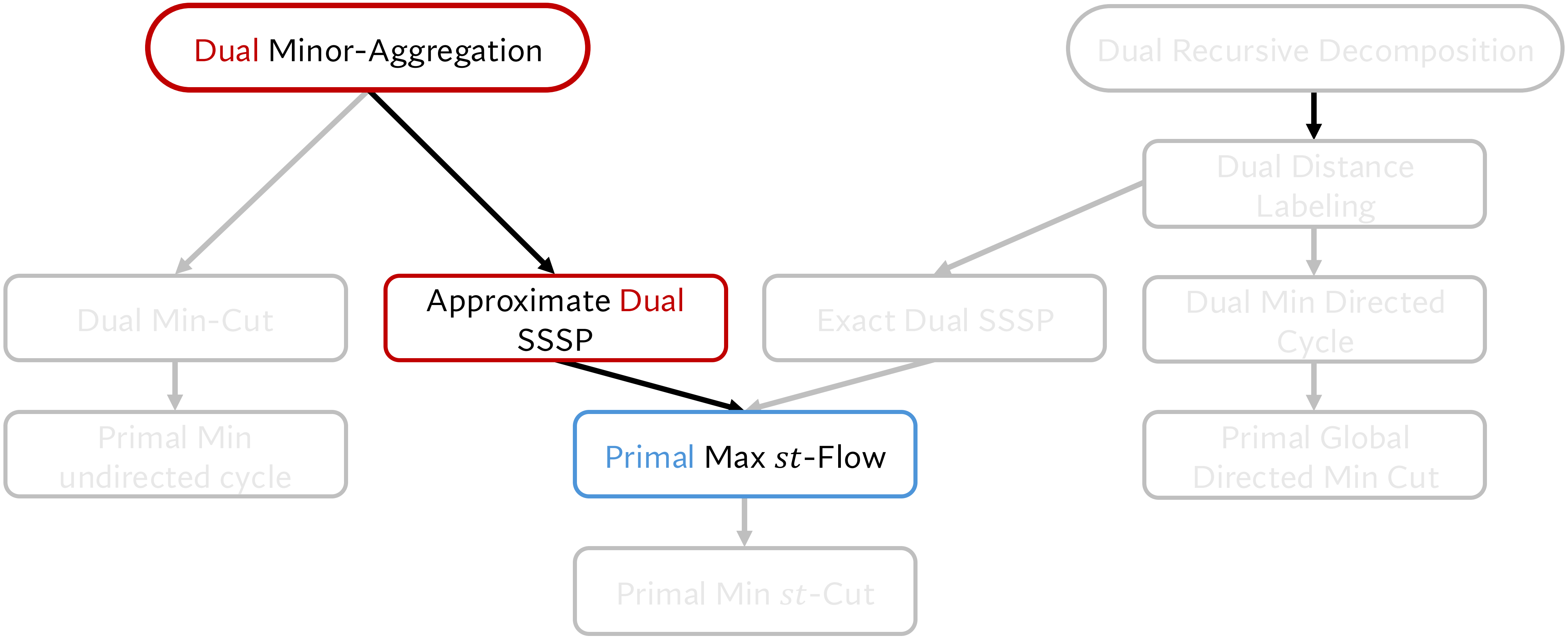
Minimum weight cycle of G is found in $\tilde{O}(D)$ rounds.

$\hat{G}$

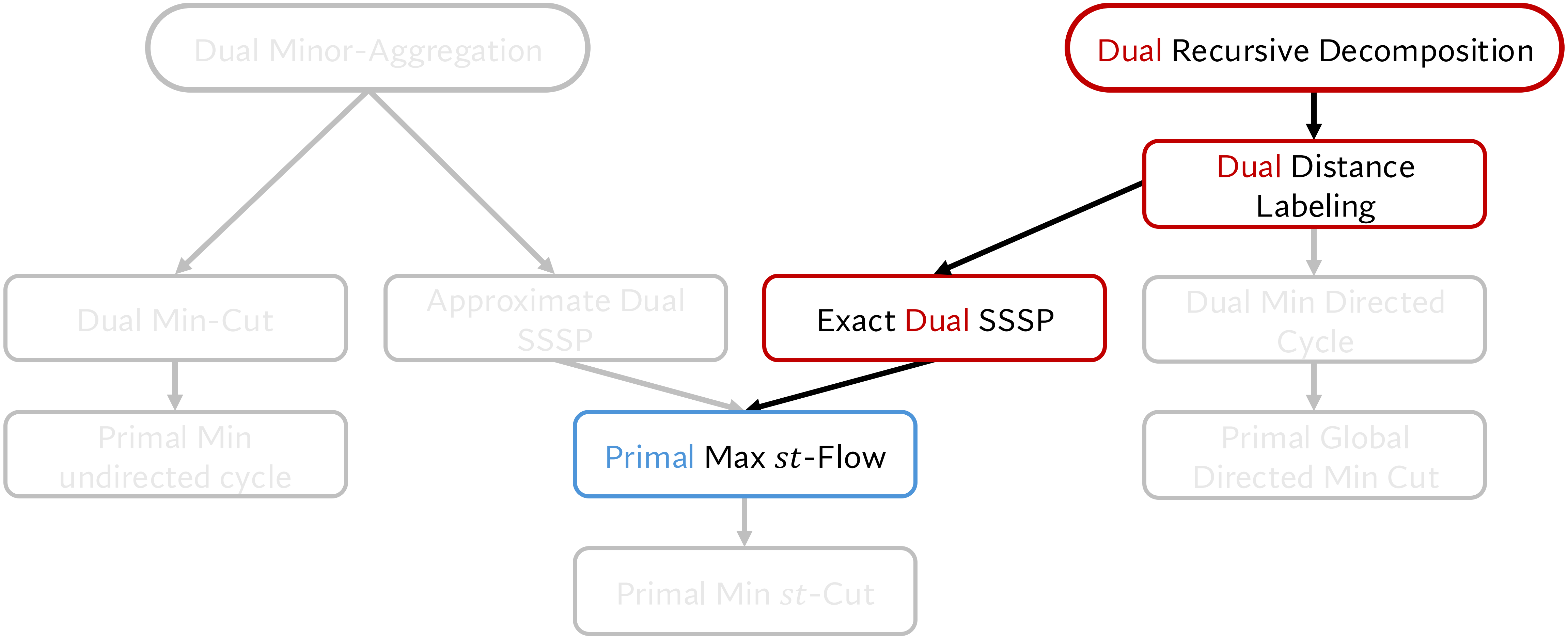




Approximate Maximum st-Flow



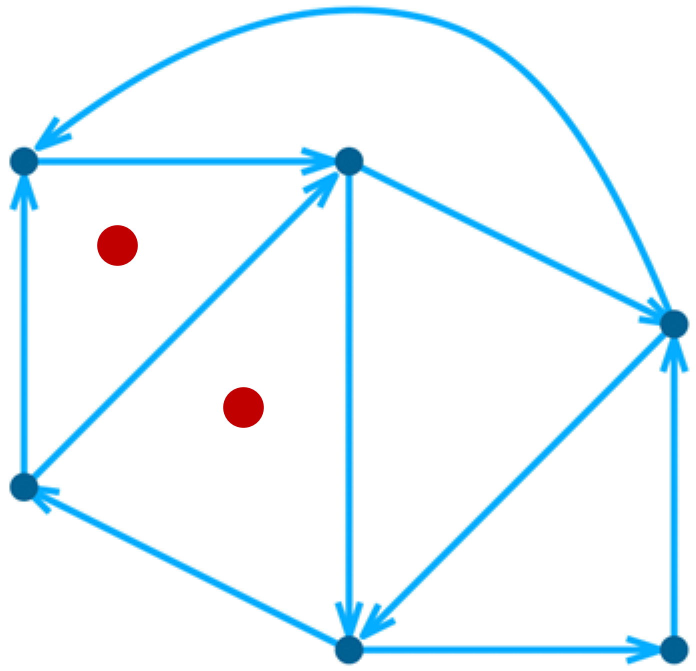
Exact Maximum *st*-Flow



Maximum st-Flow via Dual SSSP

- [Ven'83] (non-trivial):
  - Maximum st-flow in G = $\log \lambda$ SSSPs in G^*
  - Positive and negative edge-lengths
  - Arbitrary source

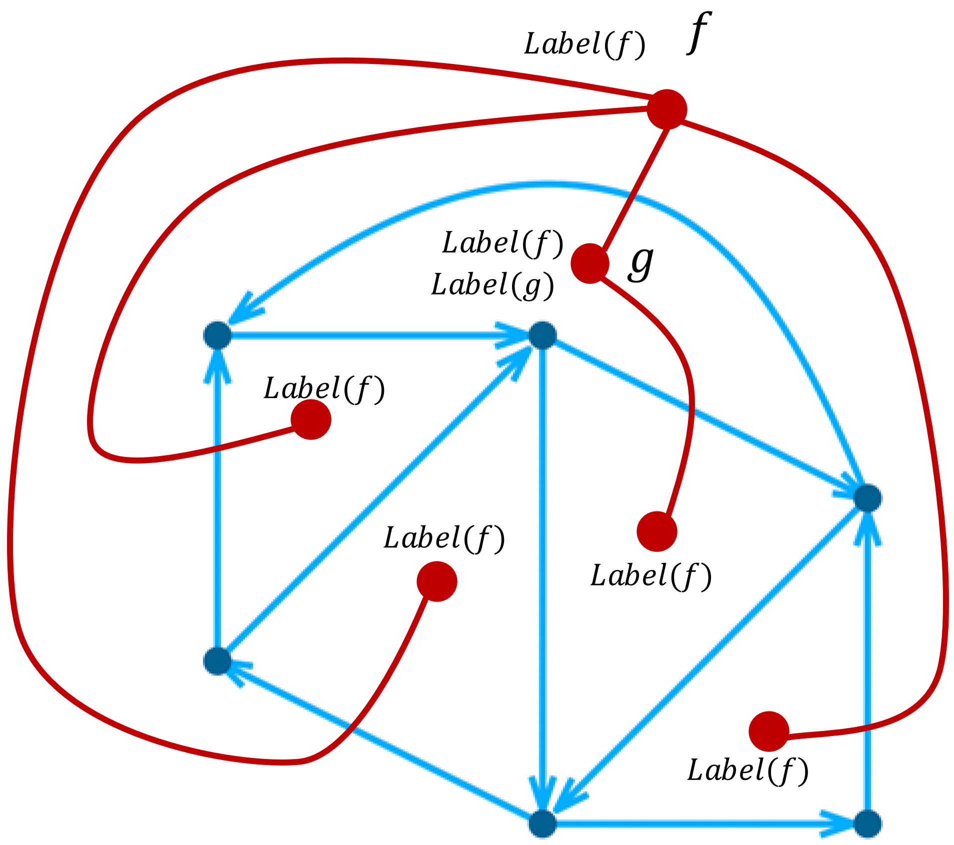
SSSP in G^* within r rounds = max st-flow in G within $\tilde{O}(r)$ rounds



Goal: Dual SSSP

- How?: Distance Labeling
  - Faces assigned ℓ-bits labels

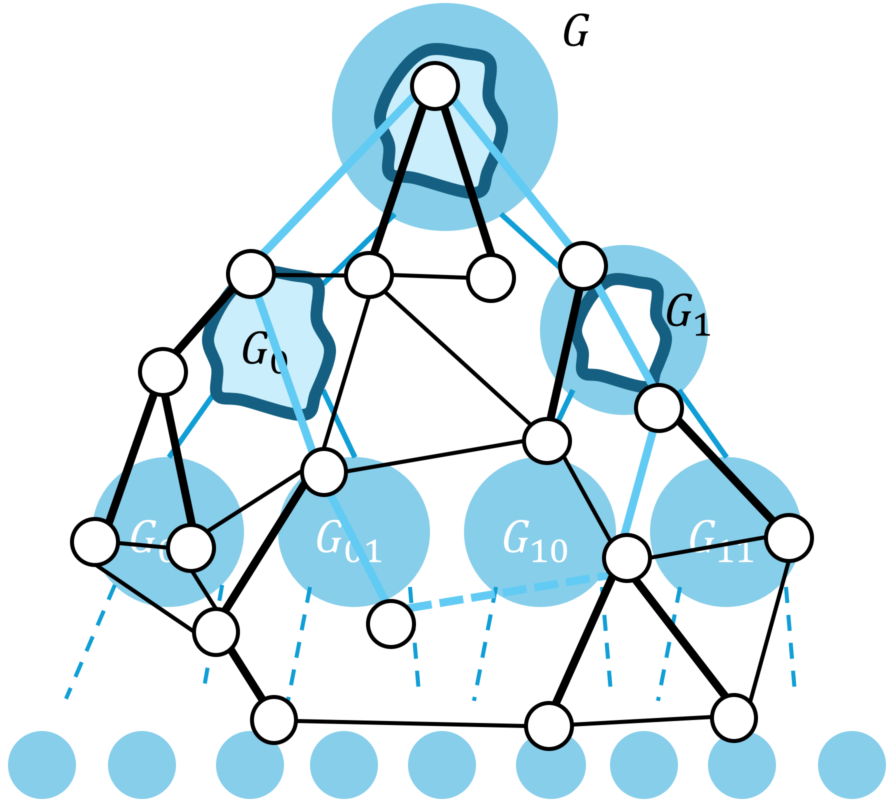
$$\left.\begin{array}{l} Label(g) \\ Label(f) \end{array}\right\} d(f,g)$$



Primal Recursive Separator Decomposition

- Planar Cycle Separators [LT'79, Mil'86]:
  - S = cycle of $O(D)$ vertices
  - $G \setminus S$ is disconnected
  - Components' sizes ≤ ½ $|G|$
  - S = two paths of a spanning tree + edge e
  - Possibly e $\notin E(G)$

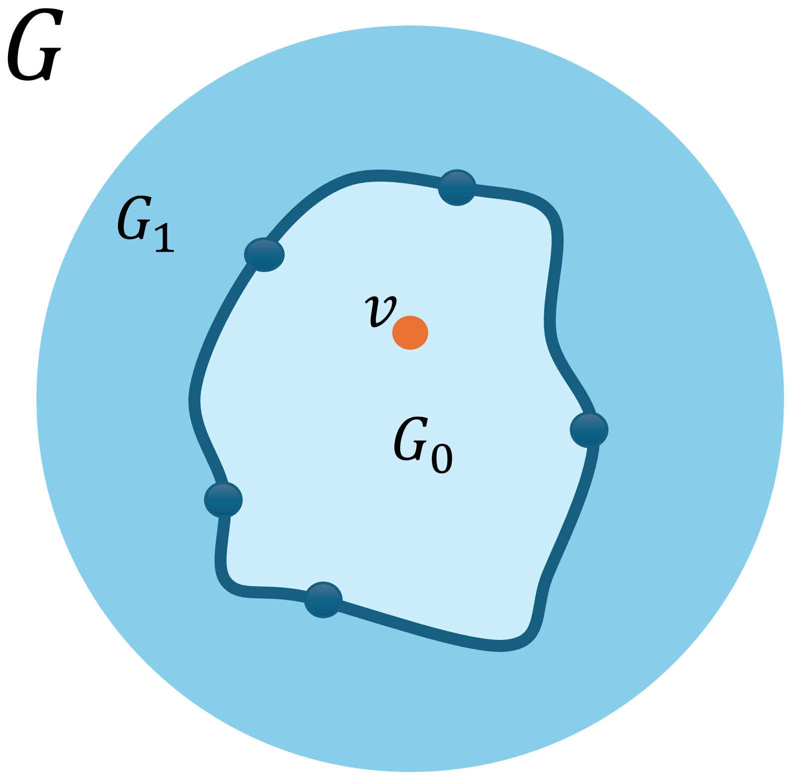
- Apply recursively..



Primal Distance Labeling

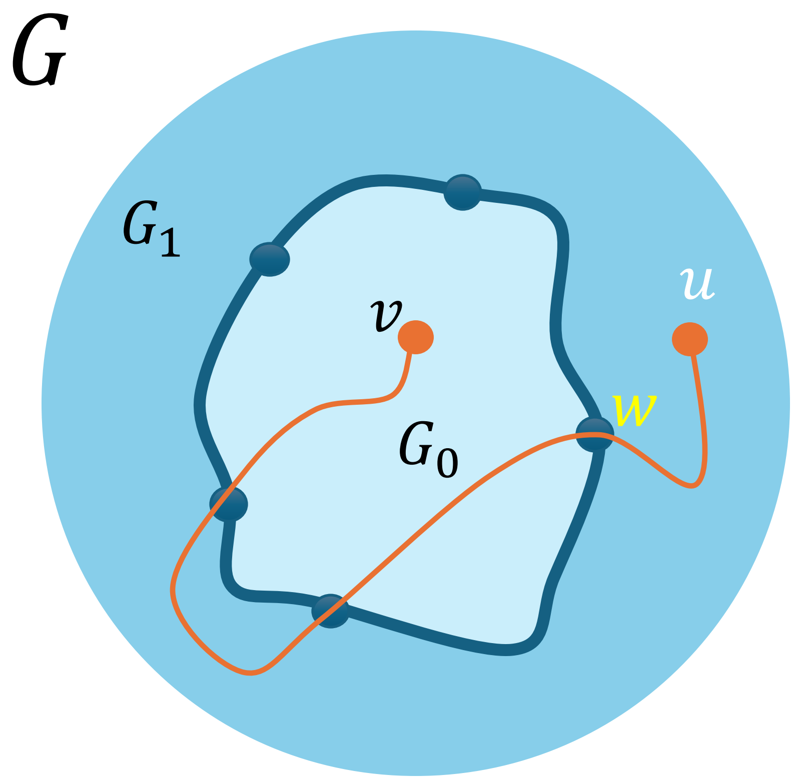
- Labels' definition [GPP+'04]:
  - $Label_G(v)$ = dist(v, s), dist(s, v) for all $s \in S$
  - If $v \notin S$, append to the label $Label_{G_i}(v)$

- Label size = $\tilde{O}(D)$



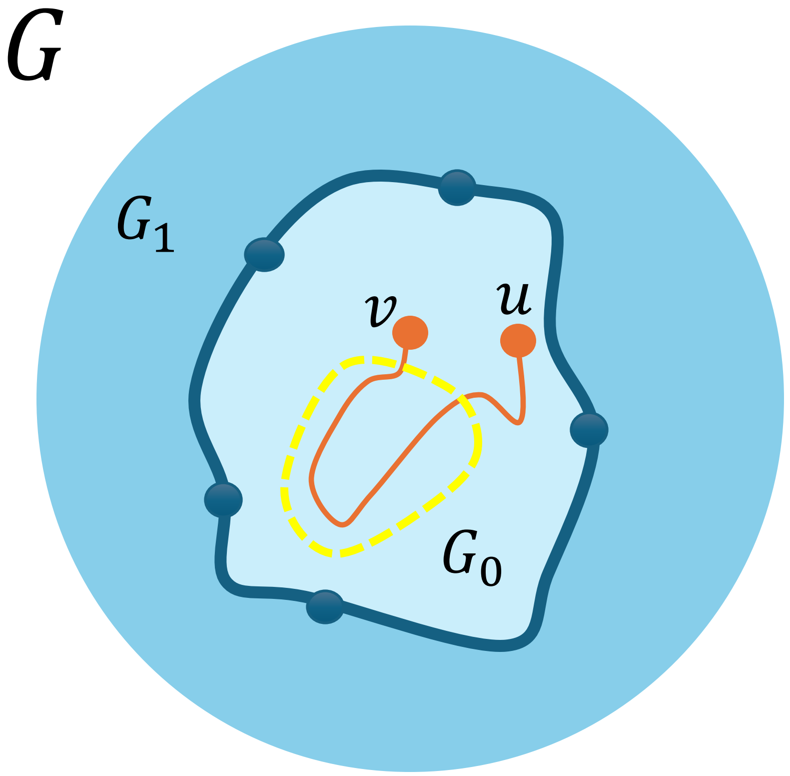
Primal Distance Labeling

- Correctness. Since S is a cycle, there are two cases:
  I. Path crosses S



Primal Distance Labeling

- Correctness. Since S is a cycle, there are two cases:
    I. Path crosses S
    II. Path does not cross S

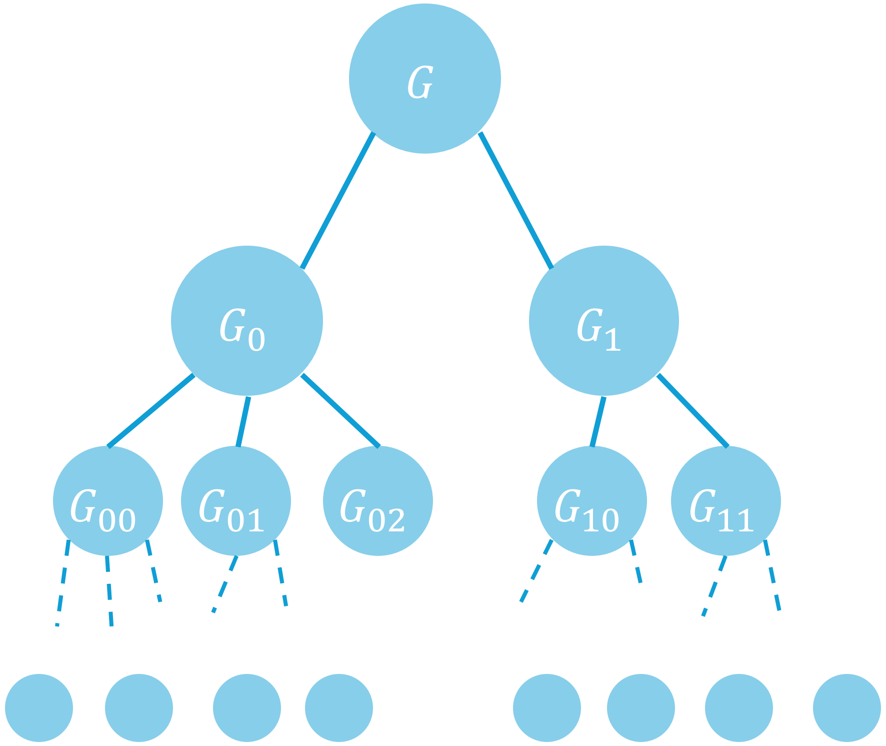


Distributed Primal Labeling Algorithm

- Bounded Diameter Decomposition (BDD) [GP'17, LP'19]:
  - Computed in $\tilde{O}(D)$ rounds
  - Subgraphs are of diameter $\tilde{O}(D)$
  - Same level subgraphs are nearly disjoint

  $\Rightarrow$ Parallel broadcast of ℓ-bits in all subgraphs of same level in $\tilde{O}(D + \ell)$ rounds

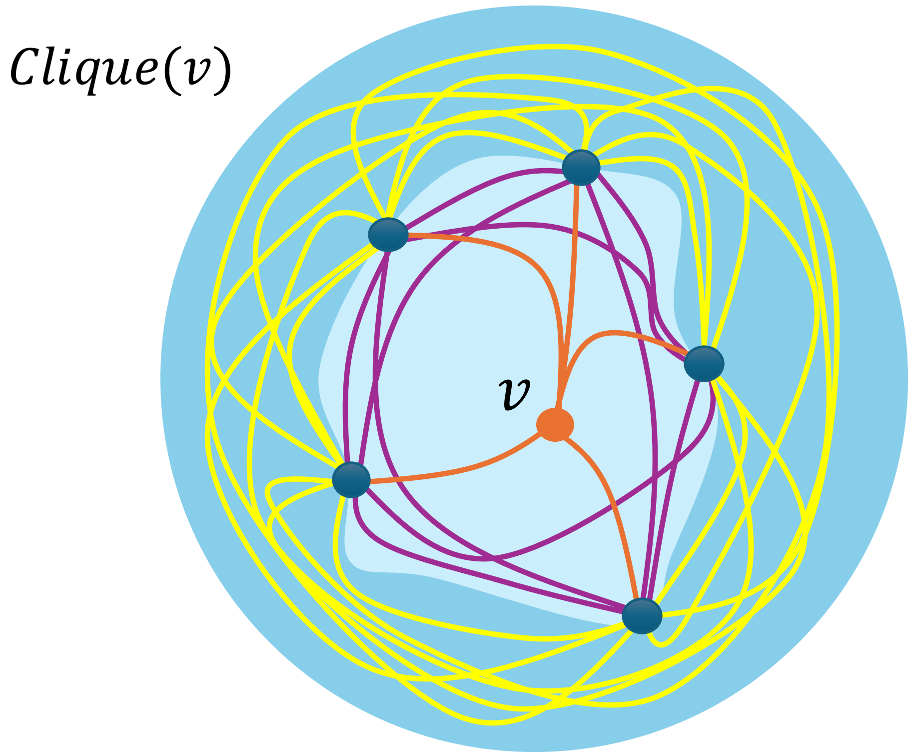
- Application: distance labeling in G



Distributed Primal Labeling Algorithm

- Compute the labels bottom-up on BDD:
  - Collect leaf subgraphs (locally compute labels)
  - Non-leaf subgraphs (recursively):
  1. Broadcast labels of separator - $\tilde{O}(D^2)$ bits
  2. Locally construct $Clique(v)$
  3. Locally compute distances between v and separator vertices in $Clique(v)$

- $\tilde{O}(D^2)$ rounds



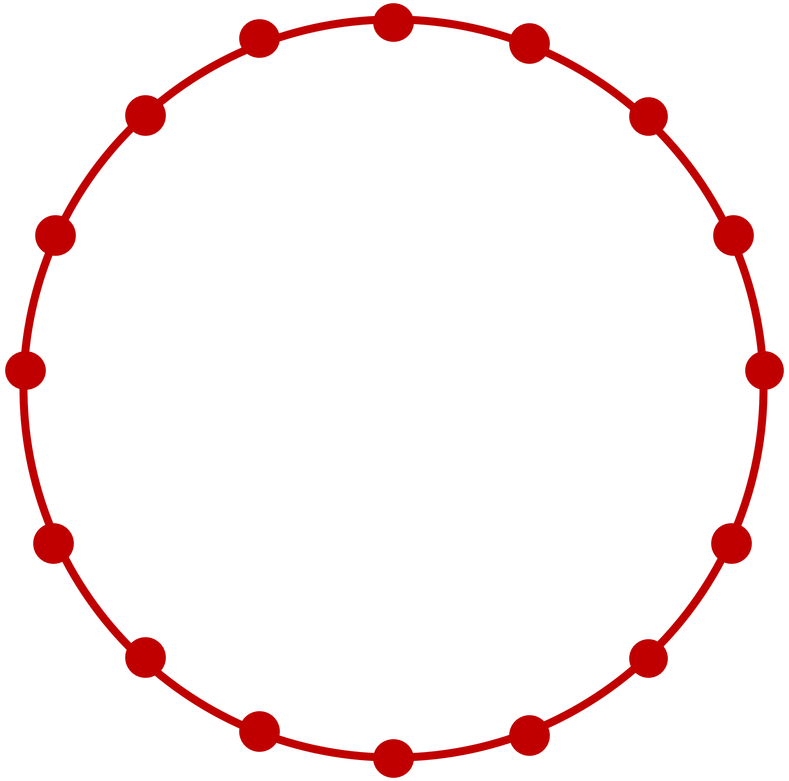
Distributed algorithm for Dual Labeling

Idea: compute a BDD for the dual and compute labeling (??)

Try 1: Naïve approach

- Run same algorithm on G^*?
  - Not the communication network.
  - The diameter of G^* might be $\Omega(n)$ while $D = O(1)$.

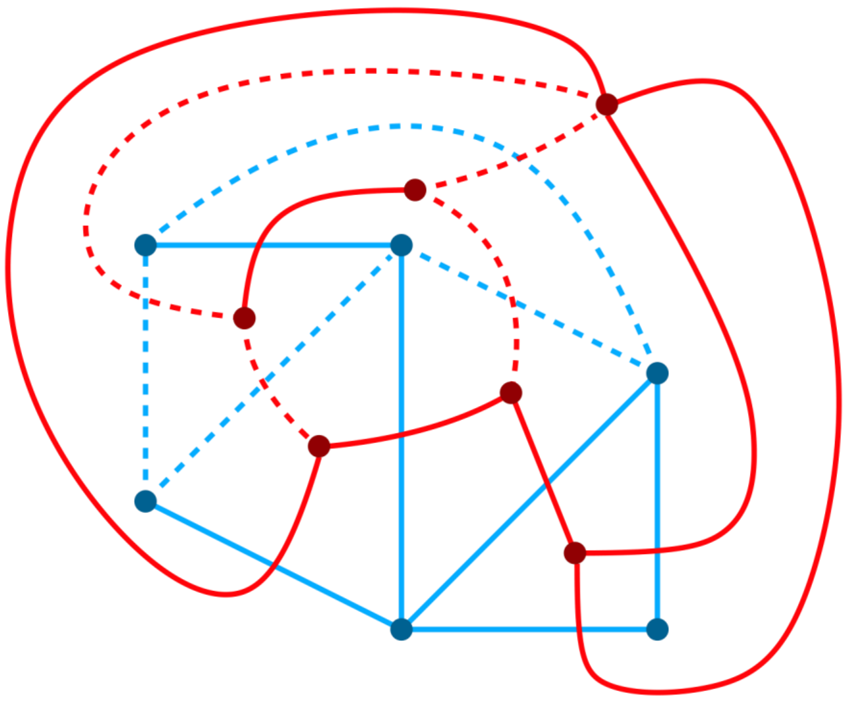
  => Does not work :/



Try 2: Recurse on G solve on G^*

- Key observation: hierarchical decomposition of G is also a decomposition of G^*
  - **Assume** separator S_G is cycle in G
  - Cycle-cut duality => S_G is edge separator of G^*
  - Apply recursively

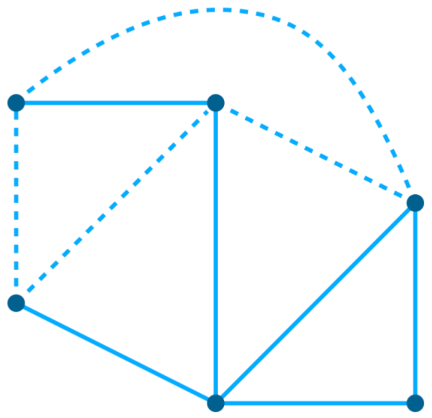
  => Has a chance but has difficulties



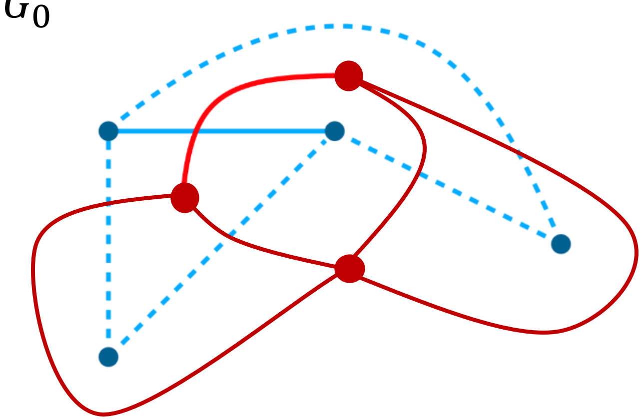
Try 2: Recurse on G solve on G^*

- Dual subgraphs in decomposition ≠ standard dual of primal

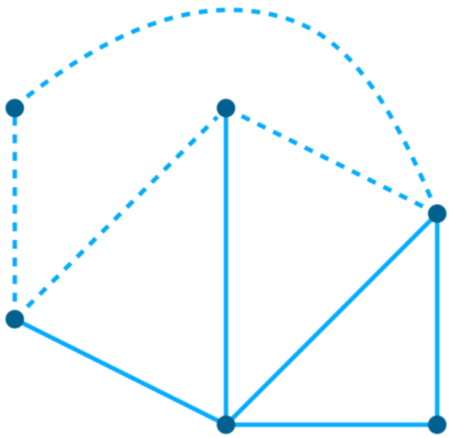
G



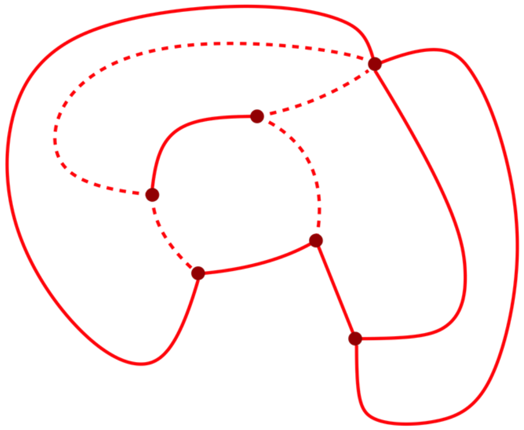
G_0



G_1

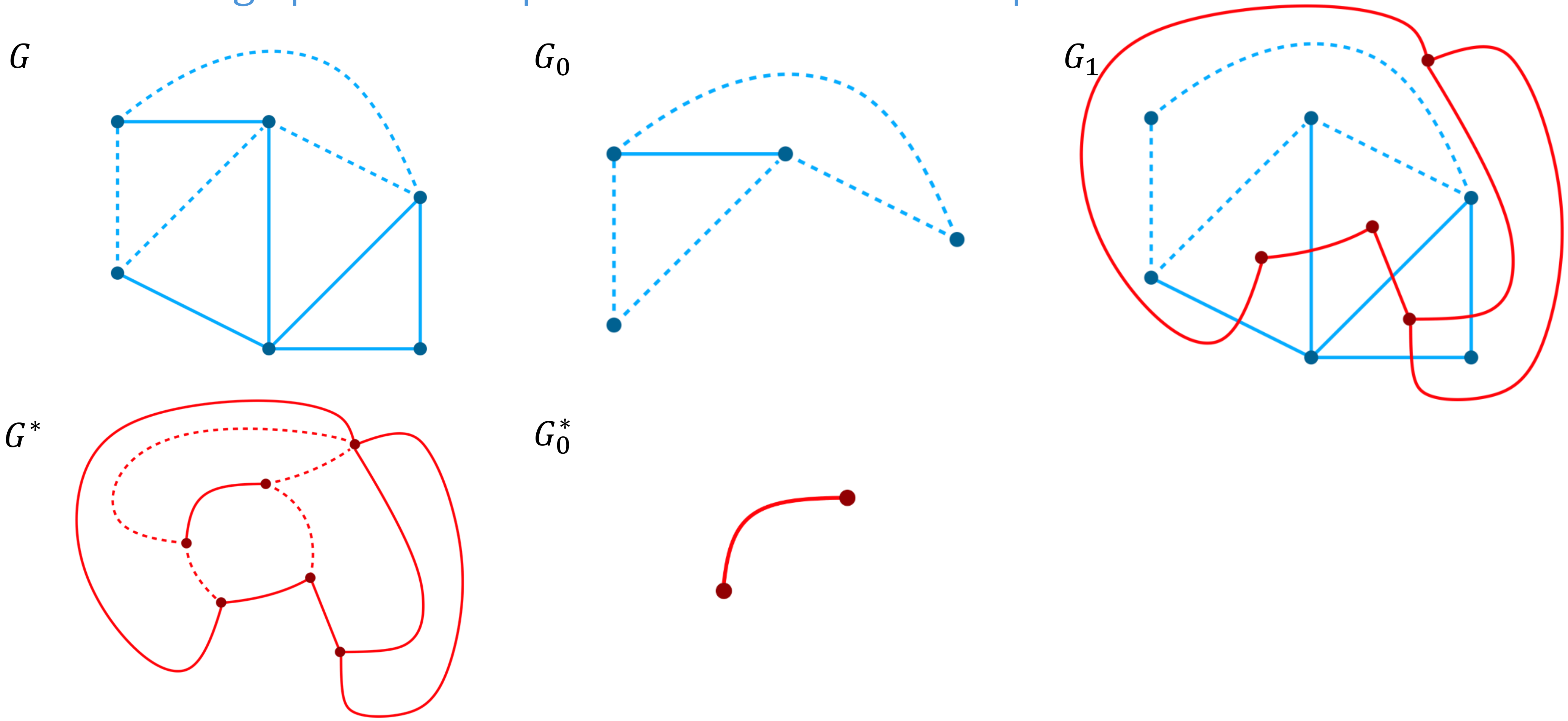


G^*



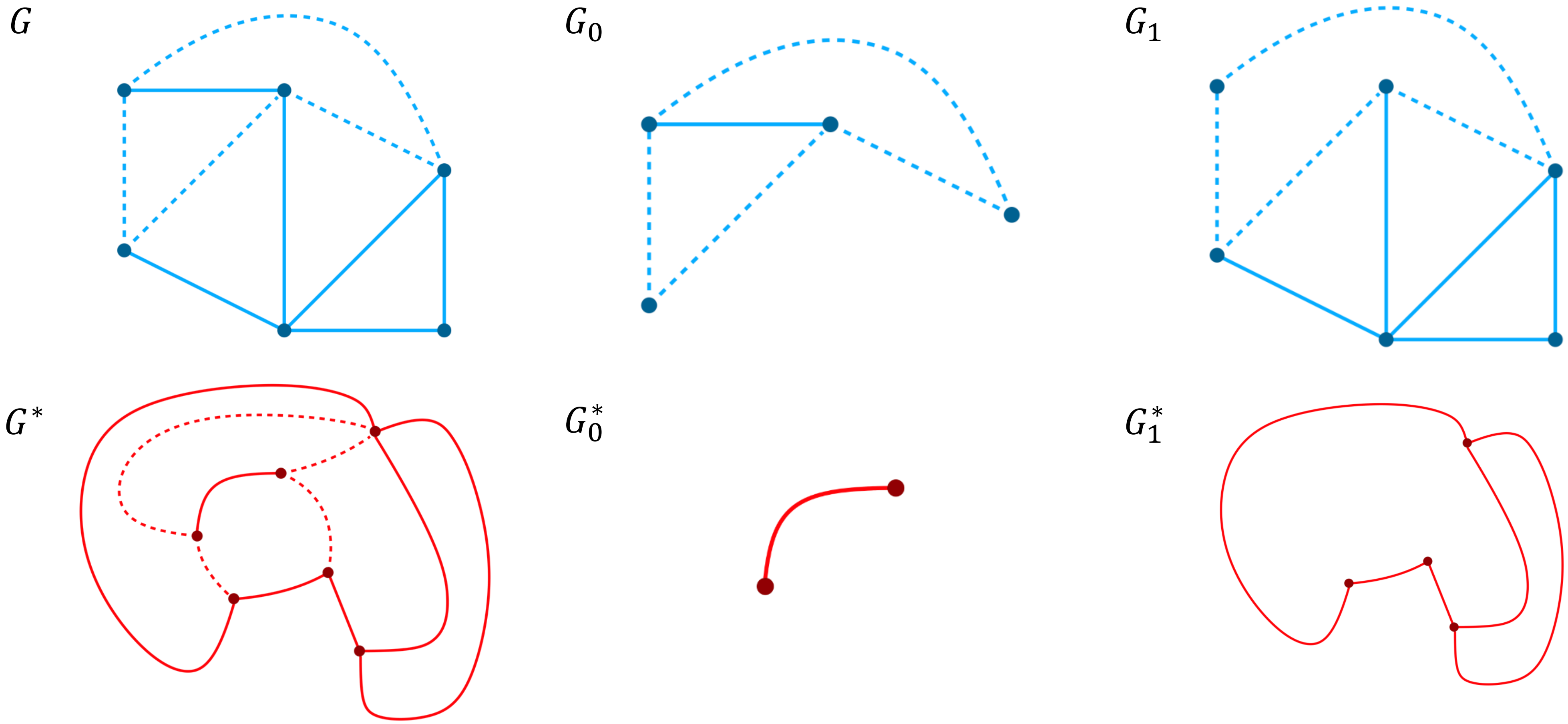
Try 2: Recurse on G solve on G^*

- Dual subgraphs in decomposition ≠ standard dual of primal



Try 2: Recurse on G solve on G^*

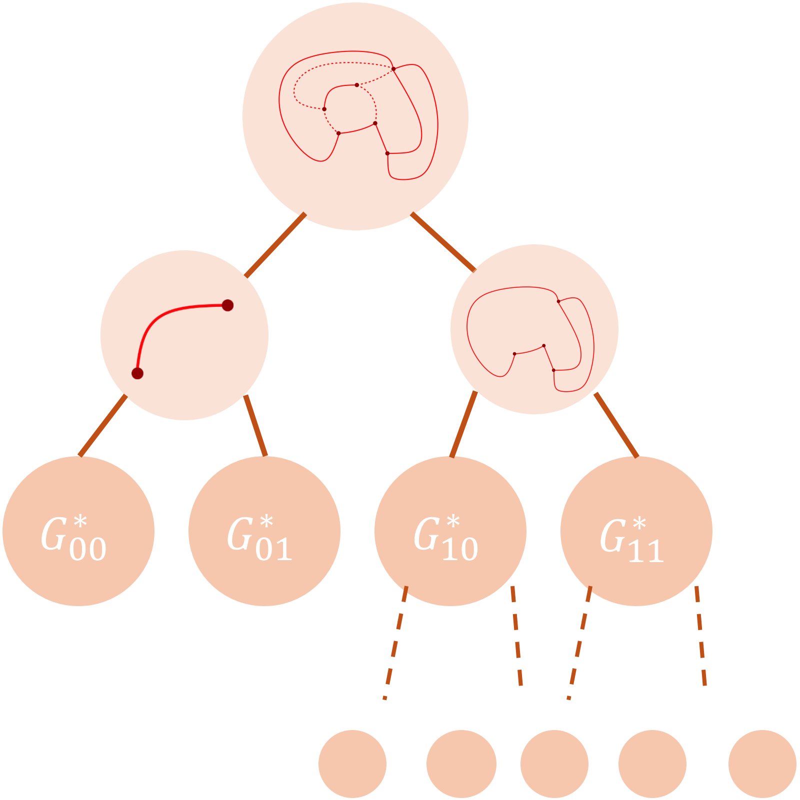
- Dual subgraphs in decomposition ≠ standard dual of primal



Dual Recursive Separator Decomposition

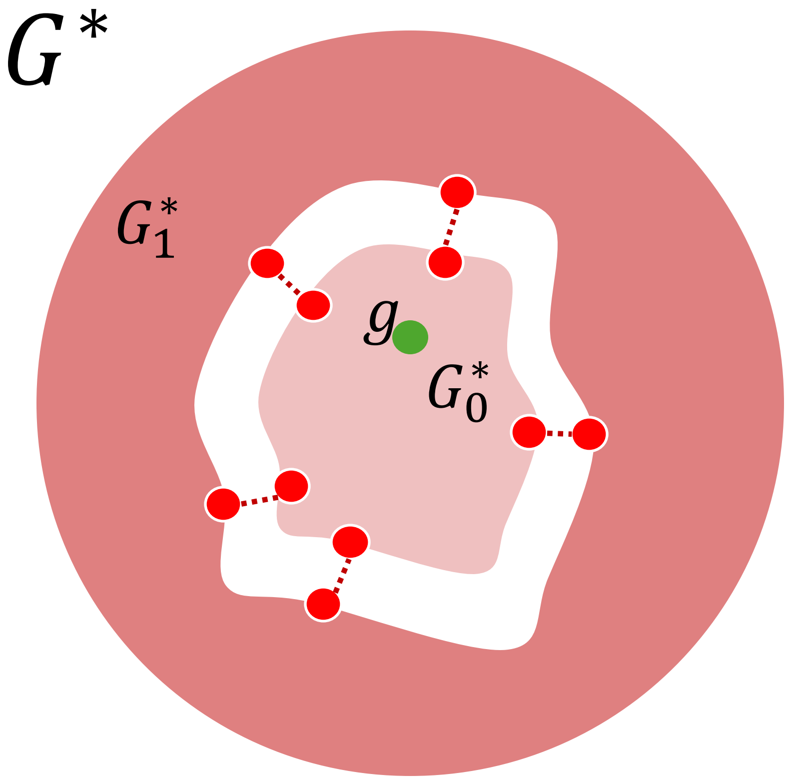
- Dual separator:
  - F = edges of S in G^*
  - $G^* \setminus F$ is disconnected
  - Components are constant factor smaller

- Apply recursively..



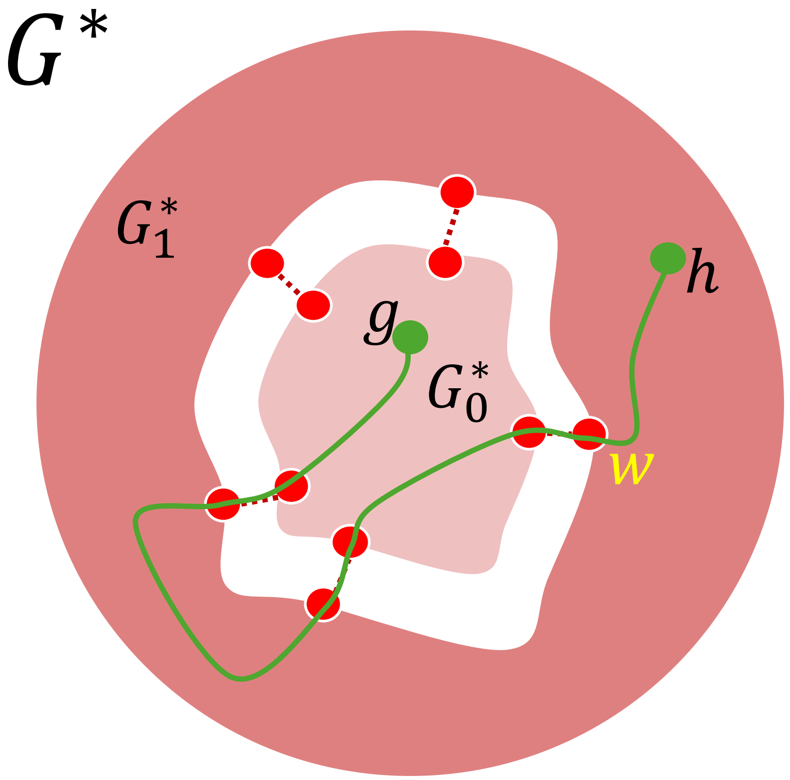
Dual Distance Labeling

- Labels' definition:
  - $Label_{G^*}(g)$ = dist(g, f), dist(f, g) for all $f \in F$
  - If $g \notin F$, append to the label $Label_{G_i}(g)$
- Label size = $\tilde{O}(D)$



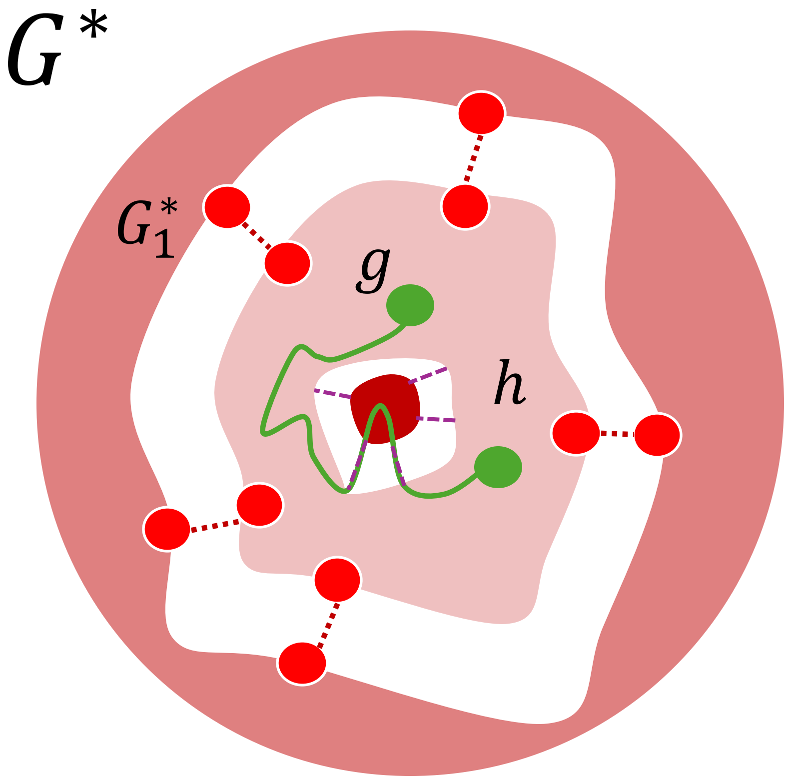
Dual Distance Labeling

- Correctness. Since F is a cut, there are two cases:
    I. Path crosses F.



Dual Distance Labeling

- Correctness. Since F is a cut, there are two cases:
  I. Path crosses F.
  II. Path does not cross F.

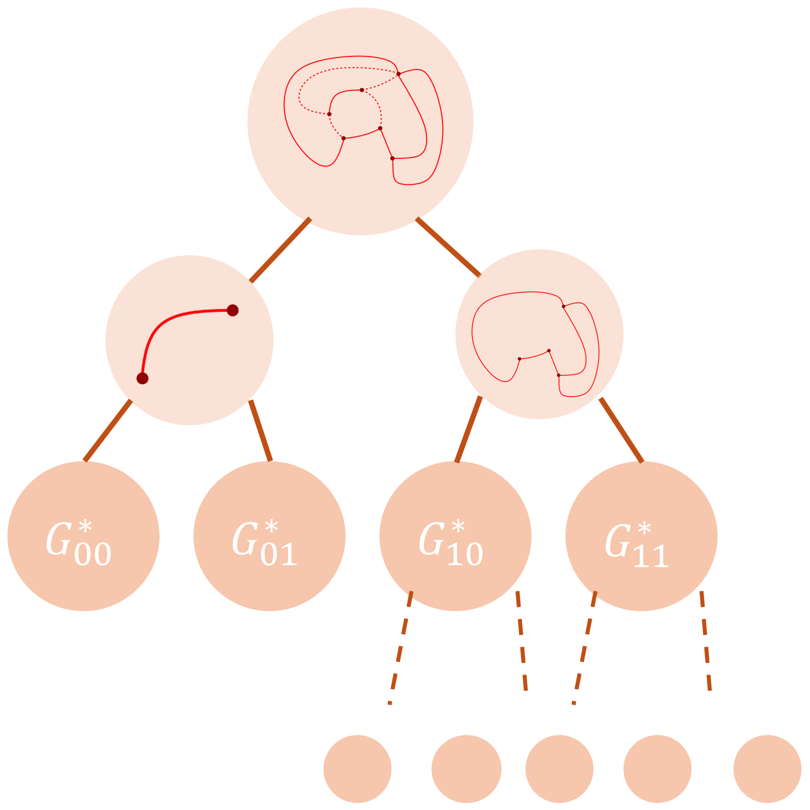


Distributed Dual Labeling Algorithm

- Dual properties for BDD.
  - Dual separator F of size $\tilde{O}(D)$
  - Can be learned in $\tilde{O}(D)$-rounds
  - Takes care of complications

- Application: distance labeling in G^*

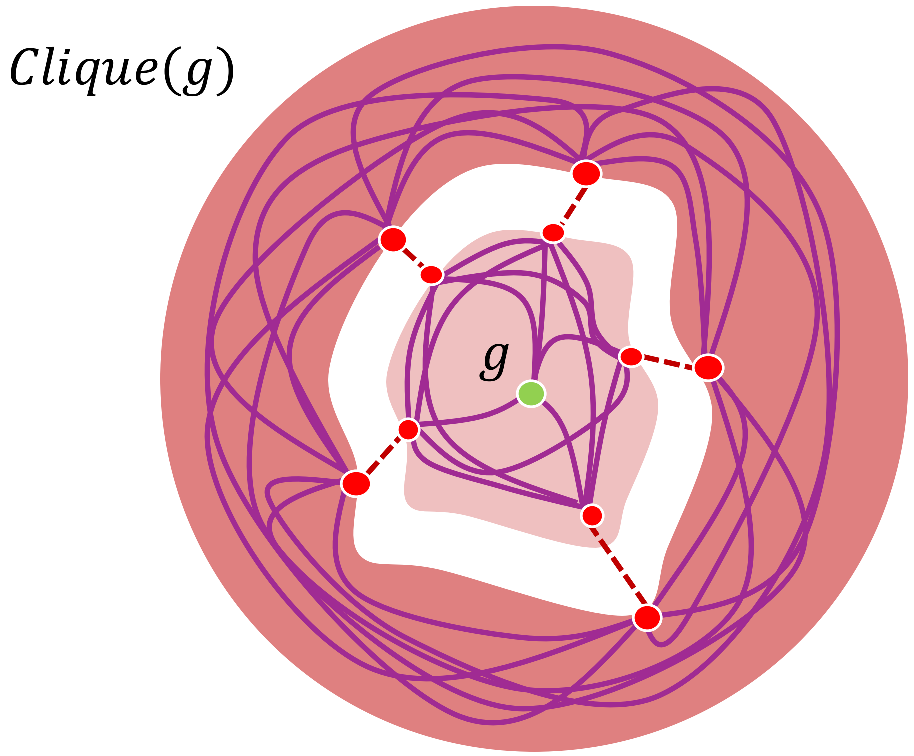
  $\Rightarrow$Max st-Flow in G



Distributed Dual Labeling Algorithm

- Compute the labels bottom-up on Extended BDD:
  - Collect leaf subgraphs (locally compute labels)
  - Non-leaf subgraphs (recursively):
  1. Broadcast labels+ **edges** of separator - $\tilde{O}(D^2)$ bits
  2. Locally construct $Clique(g)$
  3. Locally compute distances between g and separator vertices in $Clique(g)$

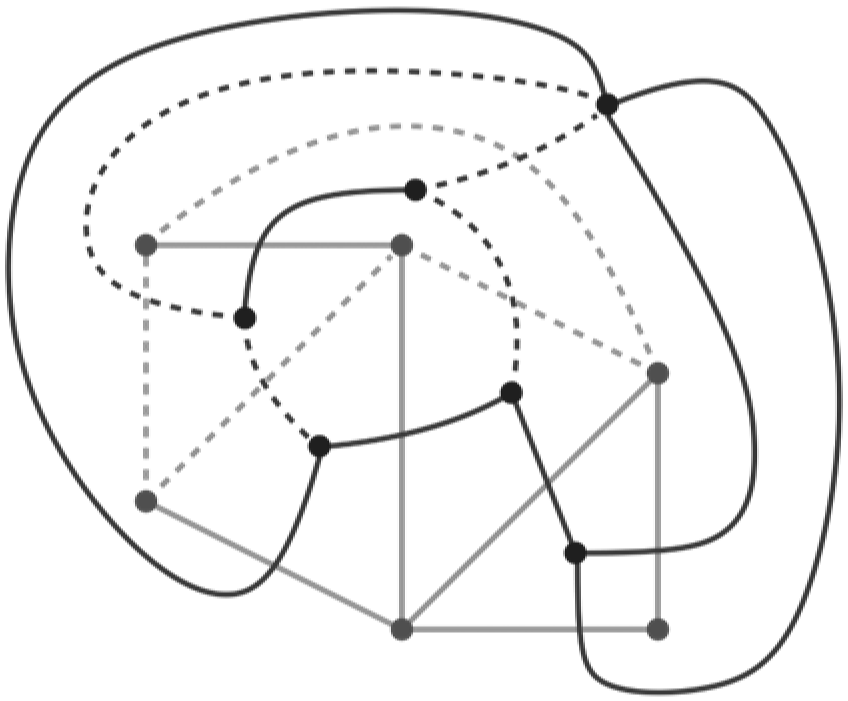
- $\tilde{O}(D^2)$ rounds = primal complexity!



Try 2: Recurse on G solve on G^*

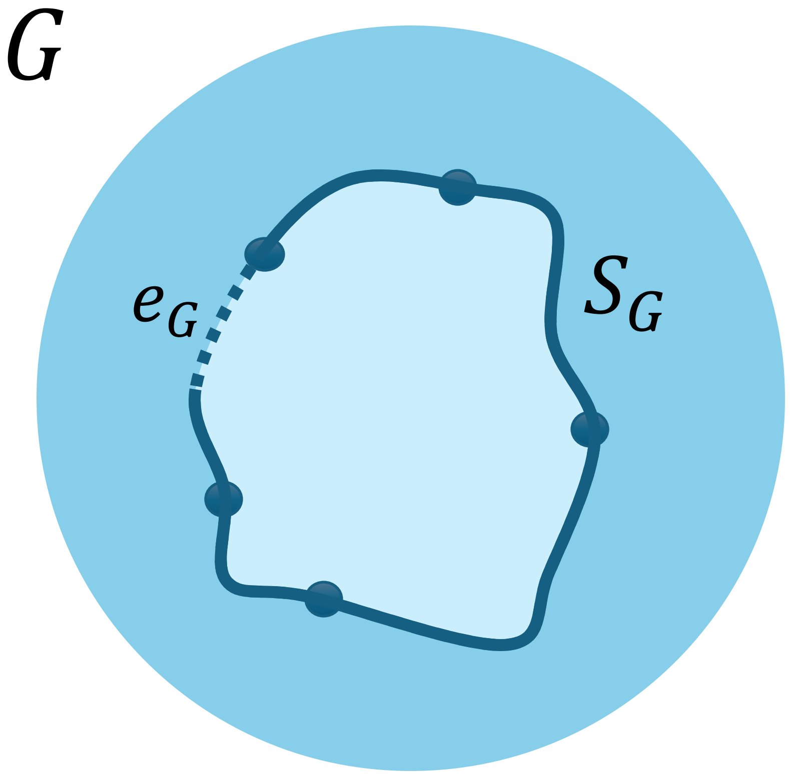
- Key observation: hierarchical decomposition of G is also a decomposition of G^*
  - Assume separator S_G is cycle in G
  - Cycle-cut duality => S_G is edge separator of G^*
  - Apply recursively

  => Has a chance but has difficulties



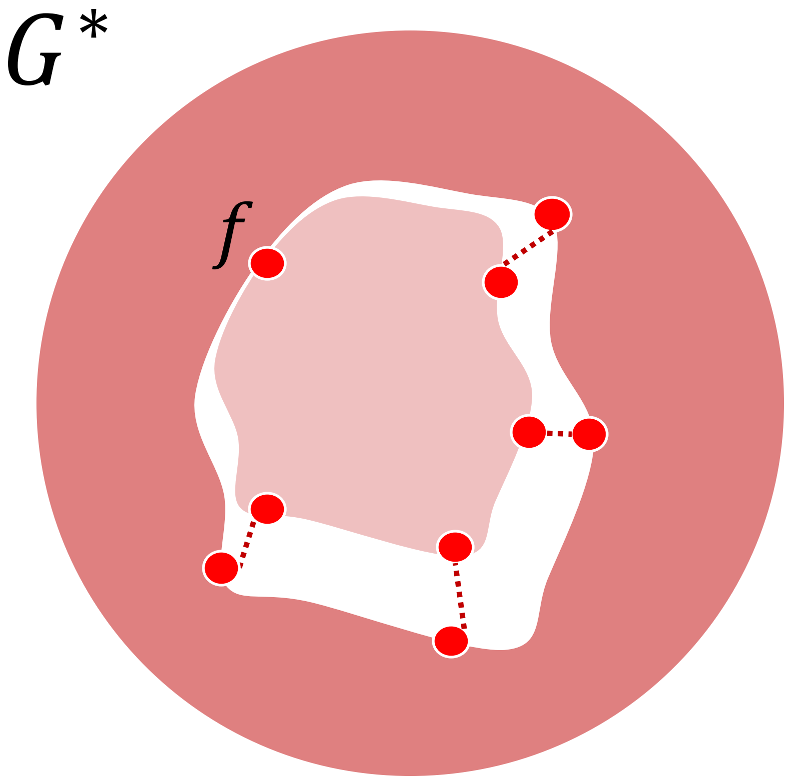
Difficulties

- Separator S_G = two BFS paths plus ***virtual*** edge e that may not be in $\mathrm{E}(G)$



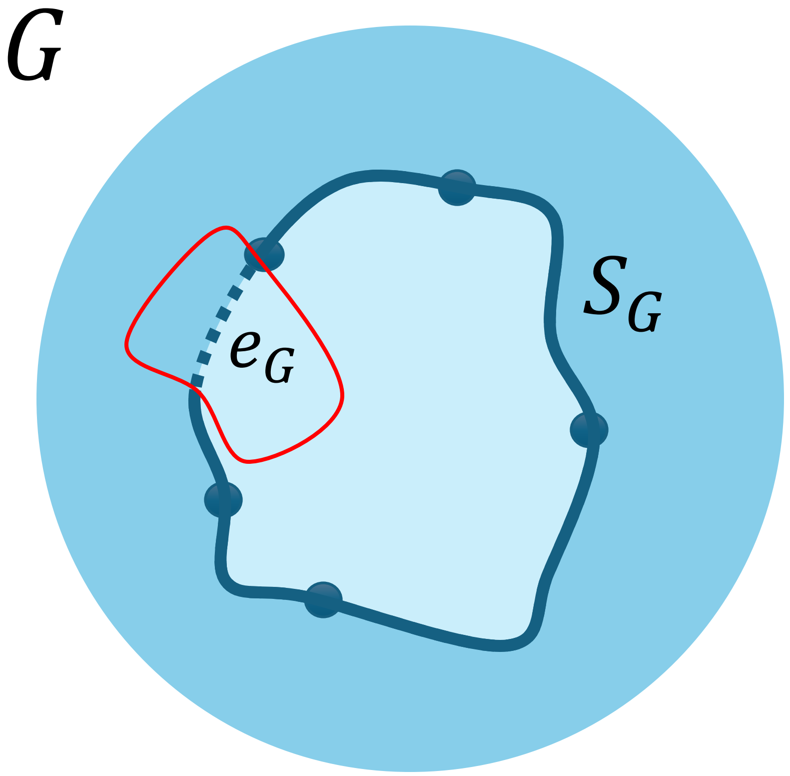
Difficulties

- Separator S_G = two BFS paths plus ***virtual*** edge e that may not be in $\mathrm{E}(G)$
  - S_G is not a cut in G^*.



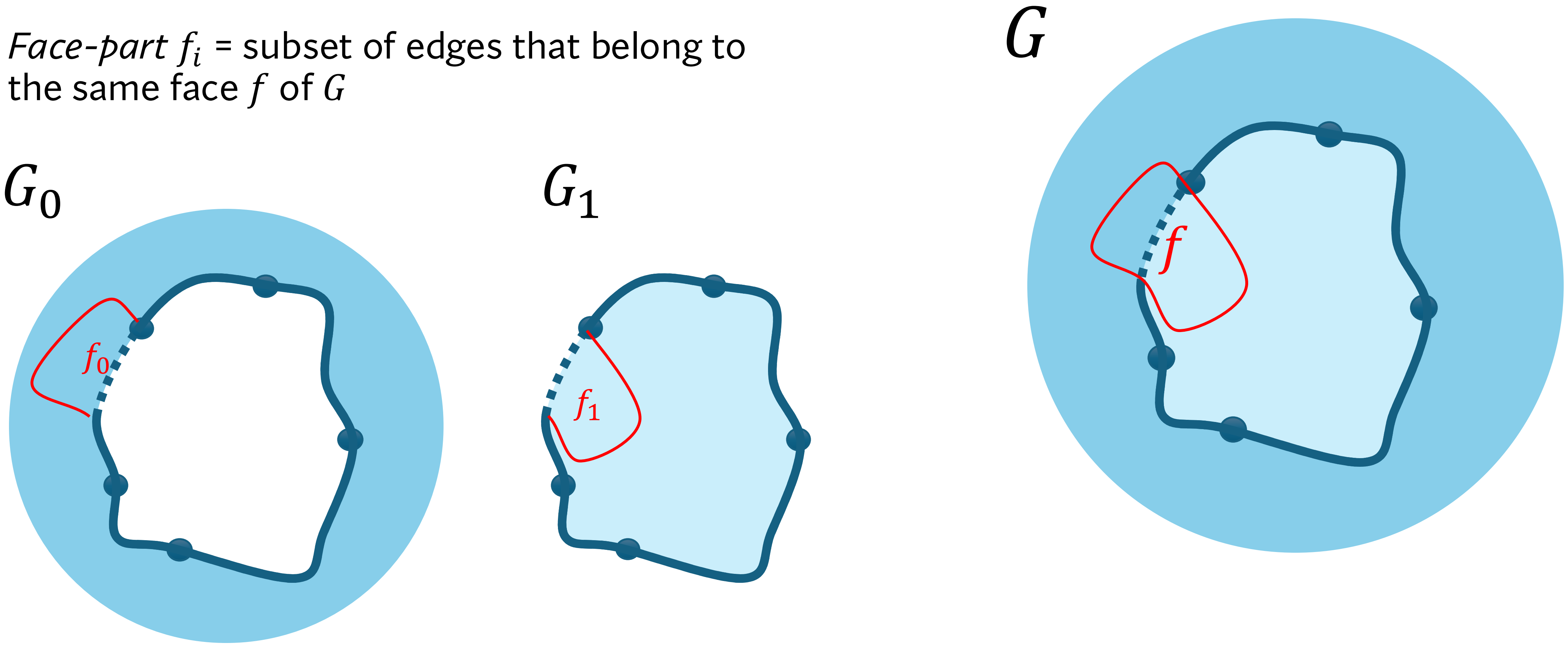
Difficulties

- Separator S_G = two BFS paths plus ***virtual*** edge e that may not be in $\mathrm{E}(G)$
  - S_G is not a cut in G^*
  - Adding e_G splits a face (dual node)
  - e_G cannot be communicated on



New Dual: work with *face-parts*, not faces

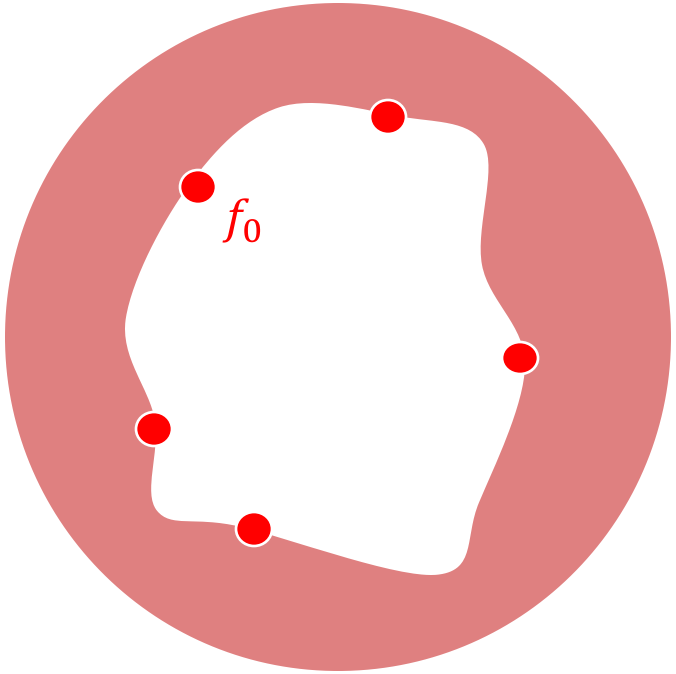
- *Face-part* f_i = subset of edges that belong to the same face f of G



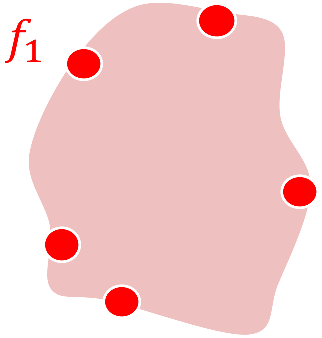
New Dual: work with *face-parts*, not faces

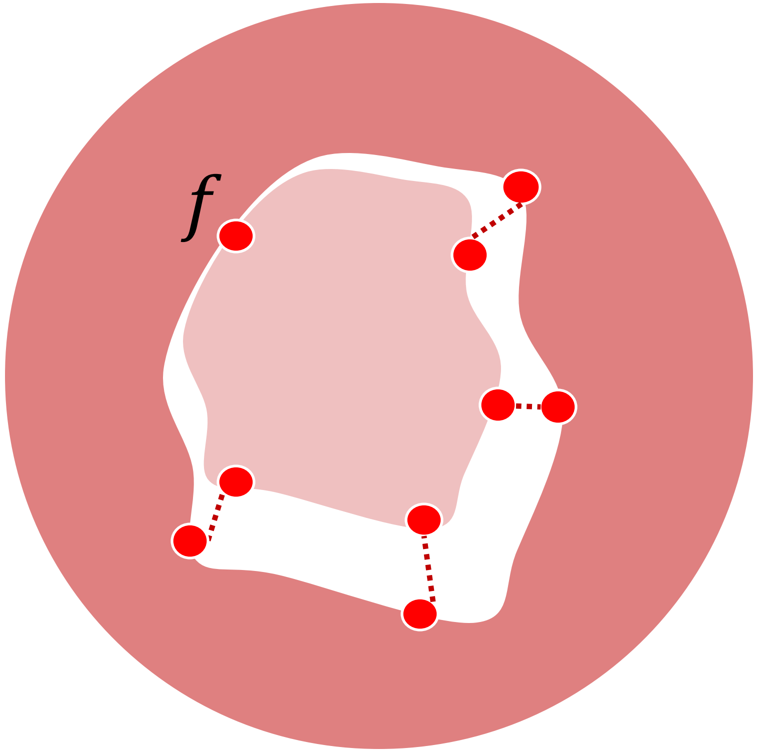
- New dual = node for each face-part as well as faces of G
- This is the standard dual G^* when considering G

G_0^*



G_1^*

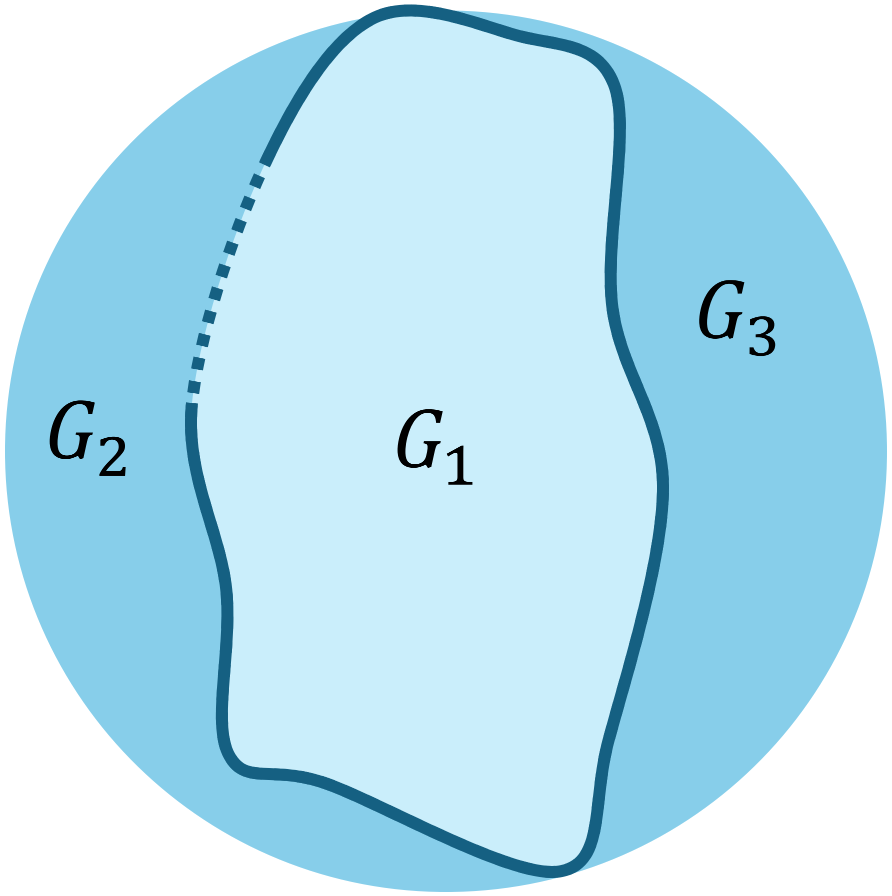




New Structural Properties for BDD

1) Any subgraph of the BDD contains at most $O(\log n)$ face-parts.

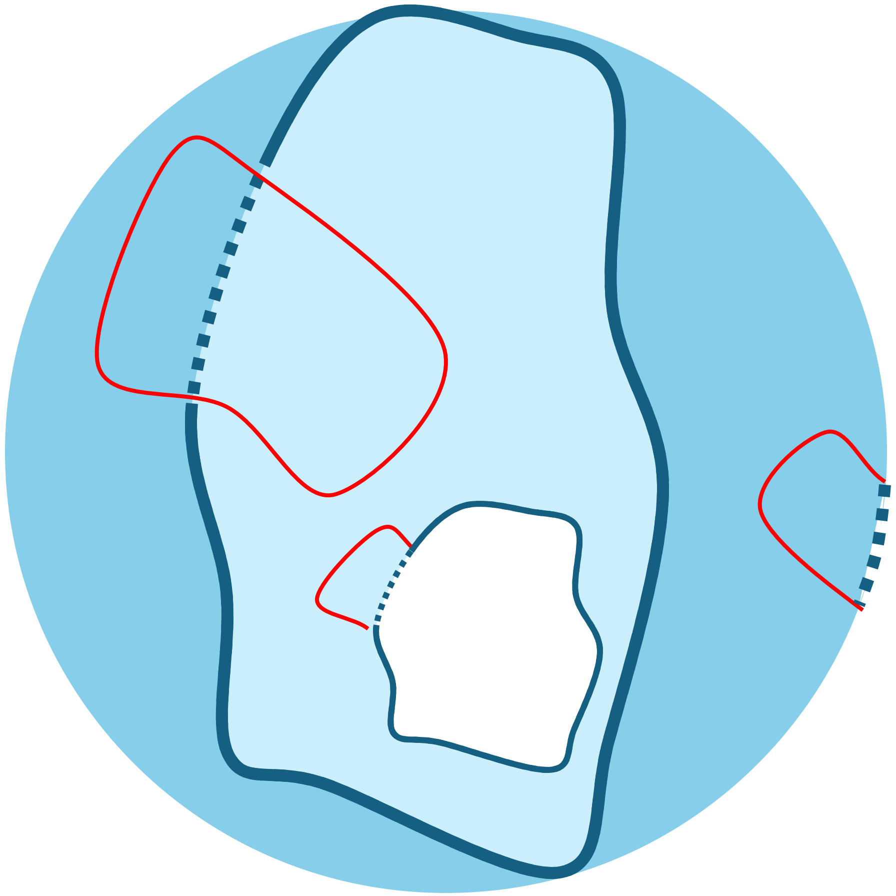
- Non-trivial: subgraph in BDD might be broken into $\widetilde{\Theta}(D)$ subgraphs



New Structural Properties for BDD

1) Any subgraph of the BDD contains at most $O(\log n)$ face-parts.

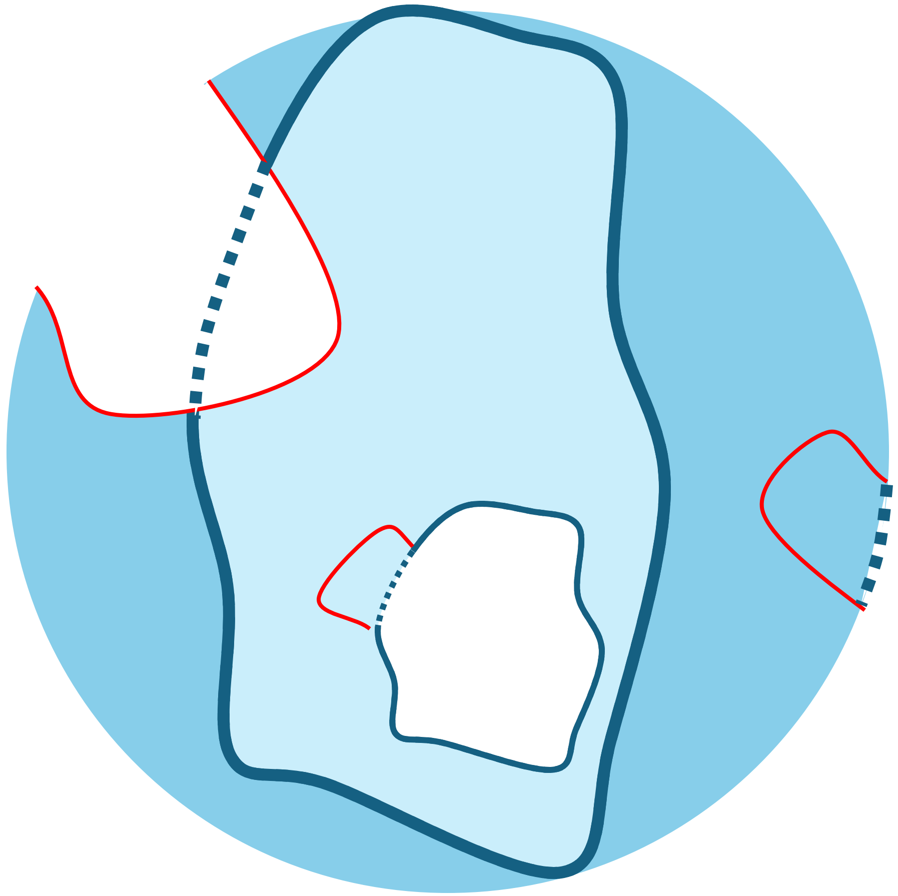
- Non-trivial: subgraph in BDD might be broken into $\widetilde{\Theta}(D)$ subgraphs



New Structural Properties for BDD

1) Any subgraph of the BDD contains at most $O(\log n)$ face-parts.

- Non-trivial: subgraph in BDD might be broken into $\widetilde{\Theta}(D)$ subgraphs
- The face that contains e might be already partitioned



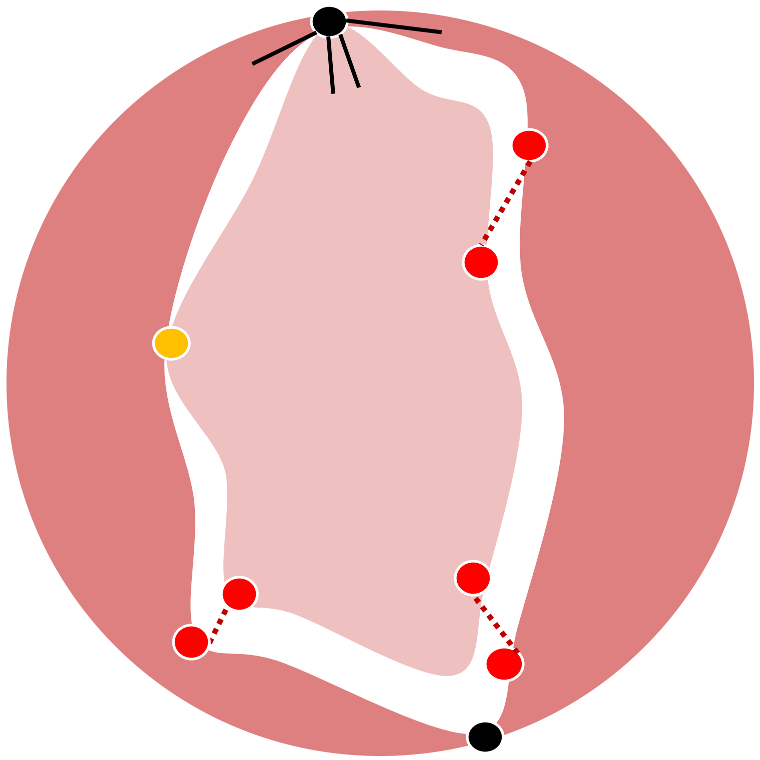
New Dual Separator

2) The set F_X that contains parts and endpoints of S_X is a separator of X of size $\tilde{O}(D)$.

Separator must cut paths between child subgraphs:

(1) Paths can use S_X edges

(2) Paths can use face-parts or the face that contains e



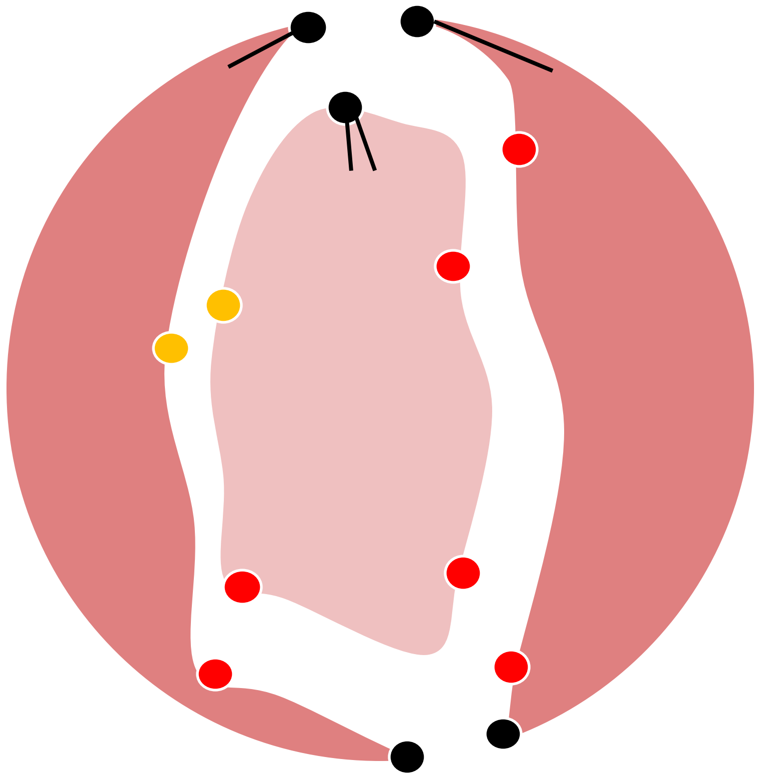
New Dual Separator

2) The set F_X that contains parts and endpoints of S_X is a separator of X of size $\tilde{O}(D)$.

Separator must cut paths between child subgraphs:

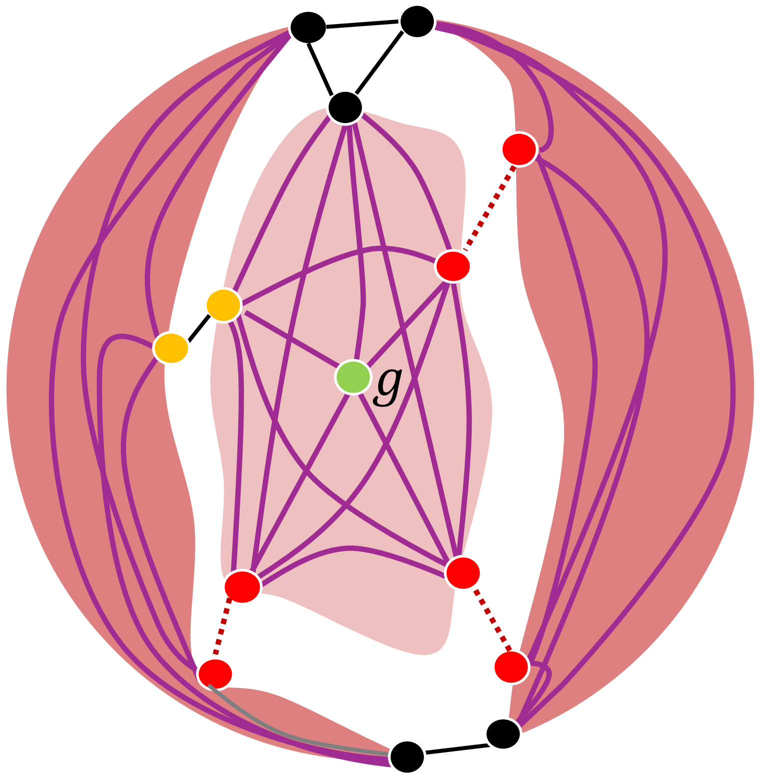
(1) Paths can use S_X edges

(2) Paths can use face-parts or the face that contains e



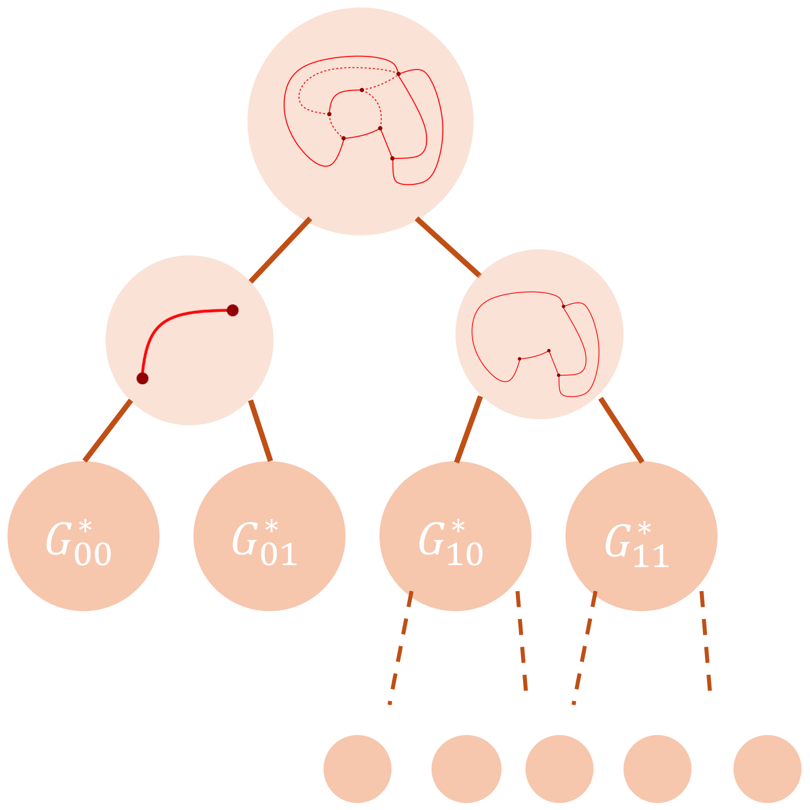
New Dual $Clique(g)$

- Vertices: g and node-parts f' of $f \in F_X$ in subgraphs of X^*
- Edges:
  - S_X edges in X^*
  - Clique edges between each two node parts f_1, f_2 of $f \in F_X$
  - Clique edges between f, h in same child subgraph

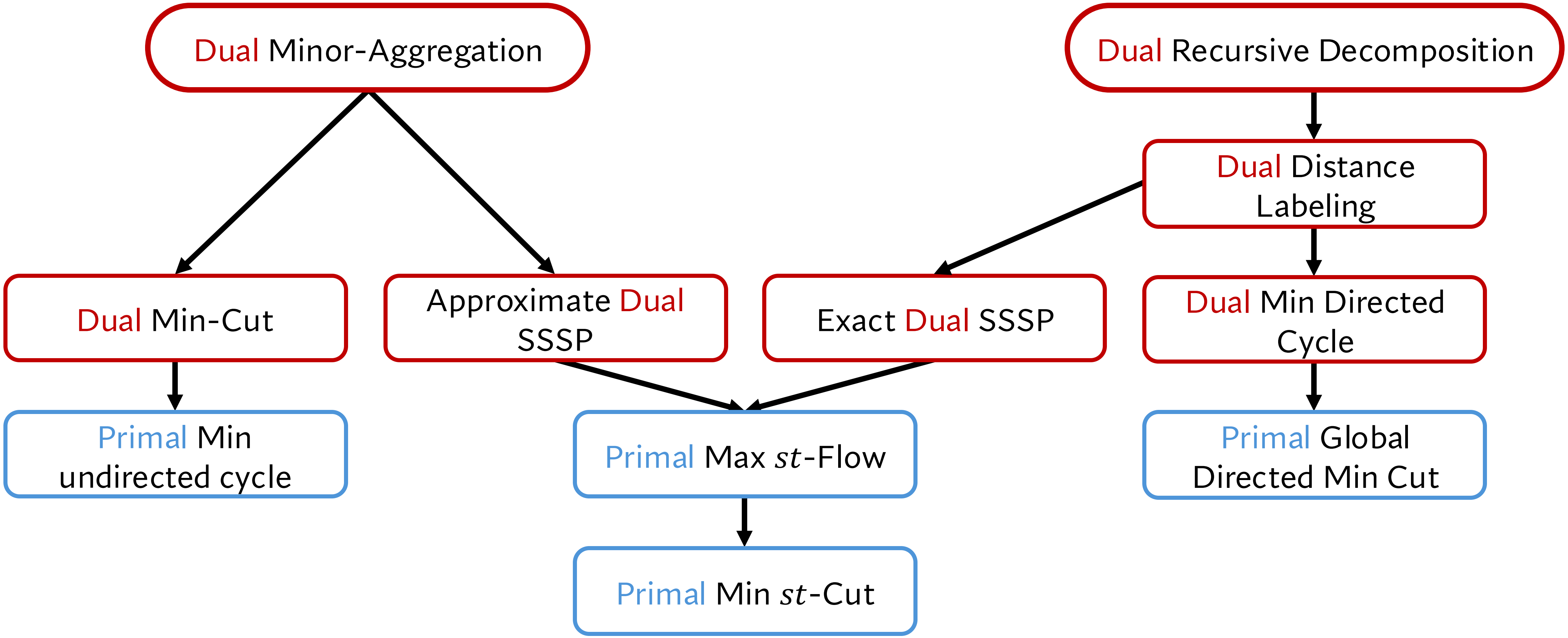


Correct Dual Labeling Algorithm

- Learn new decomposition (faces and face-parts)
- Then apply similar algorithm as in simplified case



Summary



Dual Minor-Aggregation
Dual Recursive Decomposition
Dual Distance Labeling
Dual Min-Cut
Approximate Dual SSSP
Exact Dual SSSP
Dual Min Directed Cycle
Primal Min undirected cycle
Primal Max st-Flow
Primal Global Directed Min Cut
Primal Min st-Cut

Open problems

- Max st-Flow in $\tilde{o}(D^2)$ rounds?
- Directed SSSP in G in $\tilde{o}(D^2)$ rounds?
- Exact Undirected SSSP in $\tilde{O}(D)$ rounds?
- Extend other planar centralized techniques to CONGEST?
- Extension to bounded genus graphs?

Thanks!
Questions?