

# 8.1 Multiway Cut Problem and Minimum-Cut-Based Algorithm

Approximating Multi-Terminal Cuts  
via Disjoint Isolating Regions

# Multiway Cut and the Breakdown of Planarity

- Elias Dahlhaus et al., 1992

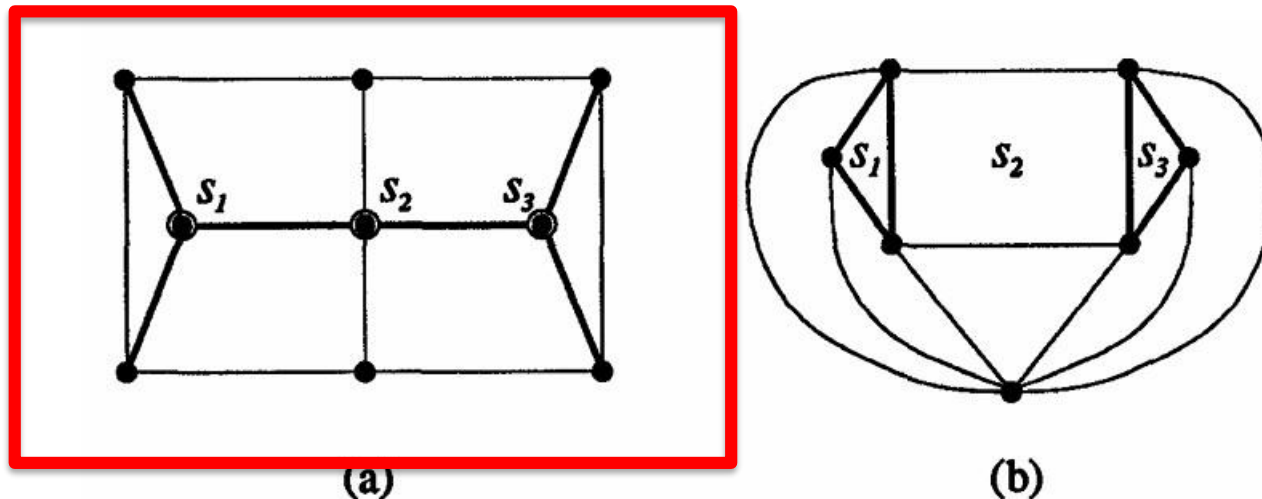
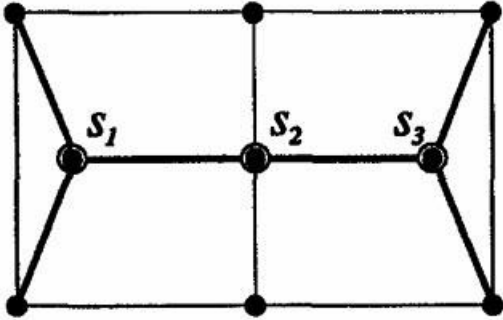


FIGURE 1. A planar 3-way cut (a) and its dual (b).

“Before introducing the algorithm, let’s clearly define what the Multiway Cut Problem is.”

# Multi-way Cut Problem



In this graph, every edge can be **cut**, and removing an edge changes the **connectivity** of the graph.

Each edge has a **weight (or cost)** representing the expense of cutting it.

Our goal is to remove a set of edges so that the three terminals

$s_1, s_2, s_3$ , are **no longer connected** to each other — that is, there is **no path** between any pair of terminals.

# Multi-way Cut Problem

- Given an undirected graph  $G = (V, E)$  with nonnegative edge costs

$$c_e \geq 0 \quad \forall e \in E,$$

- and a set of  $k$  designated terminals

$$S = \{s_1, s_2, \dots, s_k\} \subseteq V.$$

# Multi-way Cut Problem

Goal :

- Find a subset of edges  $F \subseteq E$  such that, after removing  $F$  from  $G$ ,
- every pair of distinct terminals  $s_i, s_j \in S$  lies in **different connected components** of  $G(V, E \setminus F)$ , and the total cost of the removed edges is minimized.

- Minimize

$$c(F) = \sum_{e \in F} c_e$$

- subject to  $s_i$  and  $s_j$  are disconnected in

$$G(V, E \setminus F), \forall i \neq j.$$

# Key Concept

- **Input:** A weighted undirected graph  $G(V, E)$  and a set of terminals  $S = \{s_1, s_2, s_3\}$
- **Output:** A minimum-cost edge set  $E' \subseteq E$  such that, after removing  $E'$ , all terminals become disconnected from one another.
- **Difficulty:** When the number of terminals  $k \geq 3$ , this problem becomes **NP-Hard**.
- Therefore, we typically rely on **approximation algorithms**, such as the *Isolating Cut Heuristic*, to obtain solutions that are *near-optimal* rather than exactly optimal.

# Example Intuition

- - Cutting an edge changes the graph's connectivity.
- - Objective: separate all terminals with minimum total cost.
- - Common applications: network security, clustering, distributed systems isolation.

# Motivation

- - In a simple min s-t cut, we can find the exact minimum using max-flow.
- - When number of terminals  $k > 2 \rightarrow$  Multiway Cut Problem.
- - Dahlhaus et al. (1992):
  - NP-hard for general graphs.
  - Polynomial-time solvable only when planar and  $k$  fixed.



# Intuitive Example — Distributed Computing

- Each vertex = an object or process.
- Each edge = communication between objects.
- $c_e$  = communication cost.
- Terminals  $s_i$  must be placed on machine  $i$ .
- Objective: minimize inter-machine communication.

# Isolating Cuts

- For each terminal  $s_i$ , define its region  $C_i$  as the set of vertices connected to  $s_i$  after removing  $F$ .
- $F_i = \delta(C_i)$
- Each  $F_i$  is an isolating cut separating  $s_i$  from the other terminals  $\{s_1, \dots, s_k\}$ .
- A single edge  $e$  may appear in multiple  $F_i$ 's if it connects two regions  $C_i, C_j$ .

# Algorithm Idea

- For each  $i \in \{1, \dots, k\}$  :
    1. Add a virtual sink  $t$ .
    2. Connect all other terminals  $s_j, j \neq i$  to  $t$  with infinite-cost edges.
    3. Compute the minimum  $s_i$ - $t$  cut — this gives the smallest  $F_i$ .
- Output  $F = \bigcup_{i=1}^k F_i$  as the final multiway cut.

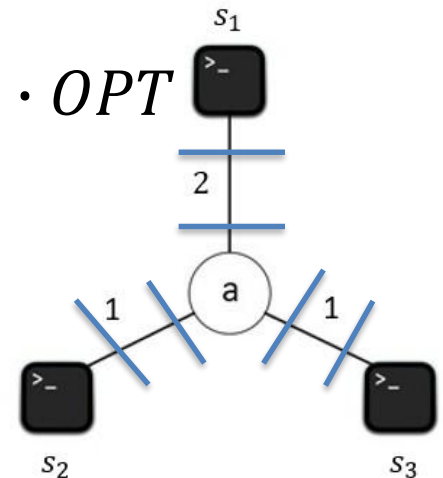
- Then:

$$c(F) \leq 2 \left( 1 - \frac{1}{k} \right) \cdot OPT$$

# Theorem 8.1 — 2-Approximation

- Let  $F^*$  be the optimal multiway cut. For each  $s_i$ , let  $F_i^*$  be its isolating cut in  $F^*$ .
- Because  $F_i^*$  is a minimum isolating cut for  $s_i$ , we know that  $c(F) \leq c(F^*)$ . Hence the cost of the solution of the algorithm is at most  $\sum_{i=1}^k c(F_i^*) \leq \sum_{i=1}^k c(F_i^*)$ .
- Observed : each edge can belong to at most two  $F_i^*$ 's:

$$\sum_{i=1}^k c(F) \leq \sum_{i=1}^k c(F_i^*) \leq 2 \cdot c(F^*) = 2 \cdot OPT$$



# Improved Version — $(2 - 2/k)$ - Approximation

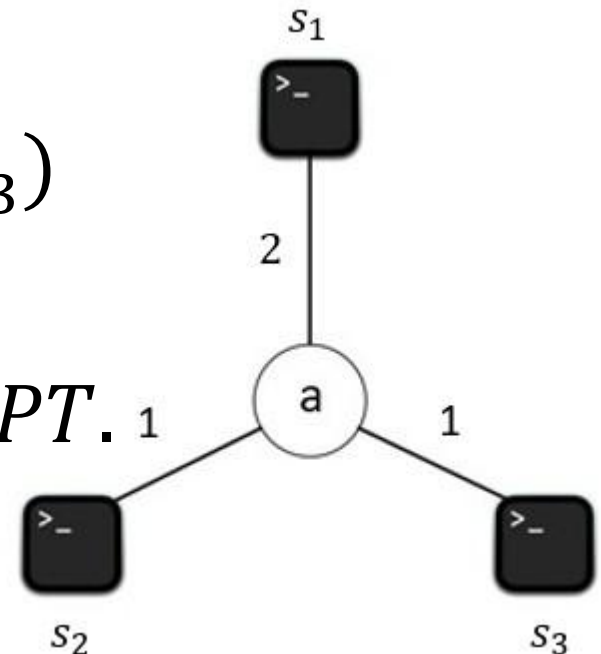
- If we discard the most expensive among the  $k$  isolating cuts and keep only the cheapest  $(k - 1)$  cuts:  $F = \bigcup_{i=1}^k F_i$
- Then:

$$c(F) \leq 2\left(1 - \frac{1}{k}\right) \cdot OPT$$

Main text begins

# Planar Case (Normal Situation)

- Algorithm:
- $F_1 = \{(s_1, a)\}$  or  $\{(a, s_2), (a, s_3)\}$  cost=2
- $F_2 = \{(a, s_2)\}$  cost=1
- $F_3 = \{(a, s_3)\}$  cost=1
- That is ,  $c(F_1) + c(F_2) + c(F_3)$   
 $= 2 + 1 + 1 = 4.$
- Union=4, take  $k - 1 = 2 \rightarrow OPT.$





Well done, But The Multiway Cut problem is NP-hard.





# “The Complexity of Multiterminal Cuts” (Dahlhaus et al., 1992)

- established a complete complexity classification of the Multiway Cut problem:
- For **two terminals** ( $k = 2$ ), it reduces to the *minimum  $s$ - $t$  cut*, solvable in polynomial time via max-flow.
- For **three or more terminals** ( $k \geq 3$ ), the problem becomes **NP-hard** in general graphs.
- However, it is **polynomial-time solvable** when the graph is **planar** and the number of terminals  $k$  is fixed.

# “The Complexity of Multiterminal Cuts” (Dahlhaus et al., 1992)

No.	Result Type	Summary
(1)	Exact algorithm for planar graphs ( $k = 3$ )	<b>In planar graphs</b> , the three-terminal Multiway Cut can be solved exactly in $O(n^3 \log n)$ time, using specialized flow-network and dual-graph techniques.
(2)	Exact algorithm for planar graphs (fixed $k$ )	For any fixed number of terminals $k$ , the Multiway Cut in planar graphs remains polynomial-time solvable, <b>but the runtime grows exponentially with <math>k</math></b> (e.g., $O(n^{O(k)})$ ).
(3)	NP-hardness results	When $k$ is part of the input (not fixed), <u>the problem remains NP-hard even in planar graphs with unit edge weights.</u> In general (non-planar) graphs, it is NP-hard even for $k = 3$ .
(4)	Approximation algorithm	The first polynomial-time approximation algorithm achieves a ratio of $2 - 2/k$ . Later studies proved this bound is essentially tight unless $P = NP$ .

# **“The Complexity of Multiterminal Cuts” (Dahlhaus et al., 1992)**

- “The results clarify the boundary between the tractable and intractable cases of the Multiway Cut problem, and give a simple, near-optimal approximation algorithm.”

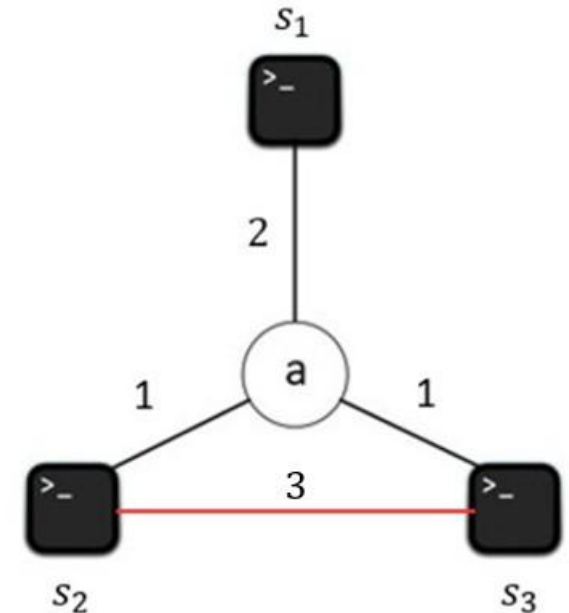
# “The Complexity of Multiterminal Cuts” (Dahlhaus et al., 1992)

- The study clearly distinguishes the boundary between tractable and intractable cases of the Multiway Cut problem:  
planar graphs with fixed  $k$  are solvable in polynomial time, while the problem becomes NP-hard for non-planar graphs or unbounded  $k$ . Moreover, they provide a simple polynomial-time algorithm achieving a near-optimal approximation ratio of  $2 - \frac{2}{k}$ .

# Breaks topological separability between regions.

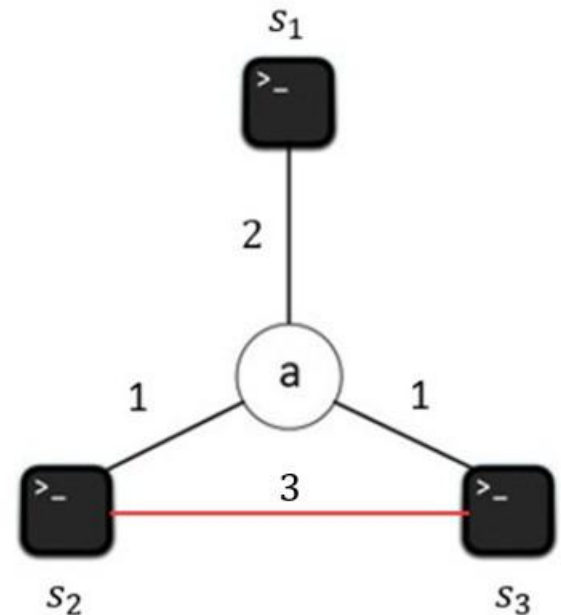
- Add edge  $(s_2, s_3) = 3$
- Now terminals  $s_2$  and  $s_3$  are directly connected → **breaks topological separability between regions.**

# “destroying the topology” refers to breaking the **disjointness** (i.e., the separability of terminal regions), rather than making the graph **geometrically non-planar**.



# Breaks topological separability between regions.

- The direct link between terminals destroys the disjoint structure of isolating cuts — the regions now overlap topologically even though the graph is still planar.



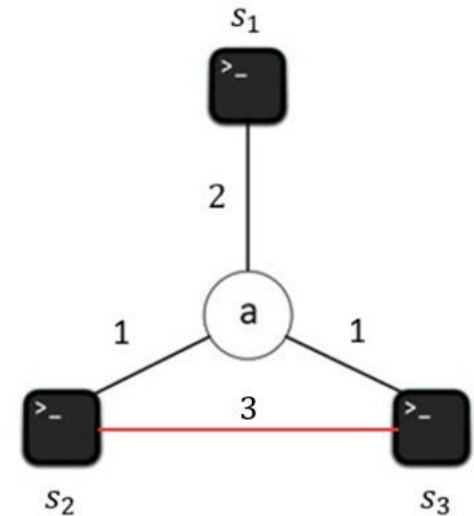
# Start by looking at the **isolating cuts** for the three terminals:

- Isolating cuts:
  - $s_1$ :  $\{(s_1, a)\} \rightarrow \text{cost} = 2$  (if select this way)
  - $s_2$ :  $\{(a, s_2)\} \rightarrow \text{cost} = 1$
  - $s_3$ :  $\{(a, s_3)\} \rightarrow \text{cost} = 1$
- Without  $(s_2, s_3)$ : total = 4  
Add  $(s_2, s_3) = 3 \rightarrow$  overlaps occur  
 $F' = \{(s_1, a), (a, s_2), (a, s_3), (s_2, s_3)\} \rightarrow \text{cost} = 7$

(The total cost = 7 here does **not** come from the theoretical step of “dropping the largest cut.”)

$$OPT = \{(a, s_2), (a, s_3), (s_2, s_3)\} \rightarrow \text{cost} = 5$$

Overlapping isolating regions  $\Rightarrow$  higher cost





# Why It Fails

- Direct edge between terminals **breaks topological separability.**  
(Terminals are no longer isolated by disjoint regions.)
- Isolating regions overlap  $\rightarrow$  edges are double-counted.  
(Cuts share common edges, increasing total cost.)
- Each edge can appear in  $\leq 2$  isolating cuts  $\rightarrow$  total  $\leq 2 \times \text{OPT}$ .  
(By double counting argument, total cost  $\leq 2 \times \text{OPT}$ .)
- Removing the most expensive cut  $\rightarrow$  approximation ratio  $= 2 - \frac{k}{2}$ .

# Isolating Cut Heuristic $\rightarrow 2 - 2/k$

## Approximation

Possible isolating cuts for  $s_1$ :

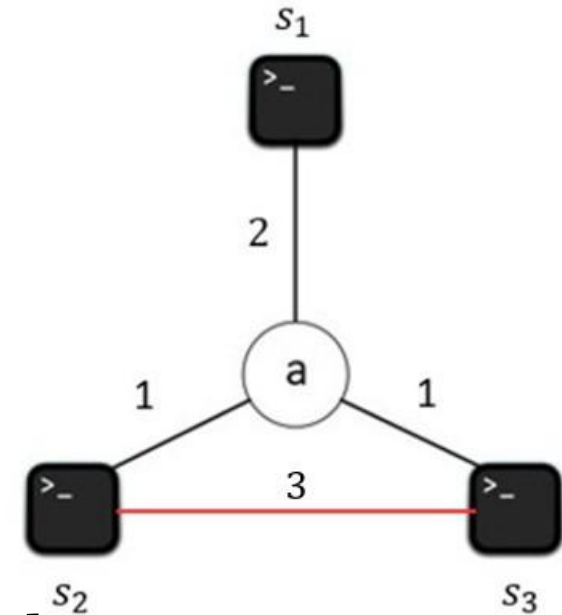
$$F_1 = \{(s_1, a)\} \rightarrow \text{cost} = 2$$

$$F_1' = \{(a, s_2), (a, s_3)\} \rightarrow \text{cost} = 2$$

(alternative choice)

Although both have the same cost, they affect the overall ratio differently:

- $F_1$  isolates one region  $\rightarrow$  minimal overlap.
- $F_1'$  connects two terminals  $\rightarrow$  larger overlap in union.



# Isolating Cut Heuristic $\rightarrow 2 - 2/k$ Approximation

Other isolating cuts:

$$F_2 = \{(a, s_2), (s_2, s_3)\} \rightarrow \text{cost} = 4$$

$$F_3 = \{(a, s_3), (s_2, s_3)\} \rightarrow \text{cost} = 4$$

Sum of isolating cuts =  $2 + 4 + 4 = 10$

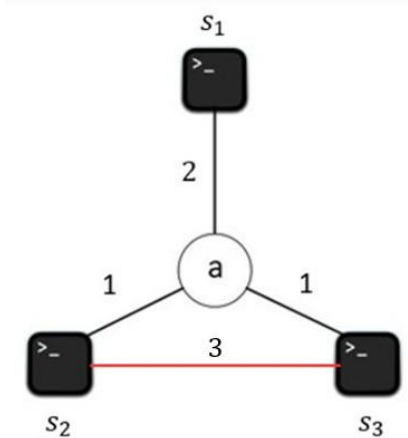
Union of all edges =  $\{(s_1, a), (a, s_2), (a, s_3), (s_2, s_3)\}$

$\rightarrow \text{cost} = 7$

True  $OPT = \{(a, s_2), (a, s_3), (s_2, s_3)\} \rightarrow \text{cost} = 5$  (OPT)

$\Rightarrow$  Equal cost  $\neq$  equal ratio impact.

ALG depends on overlap structure, not just edge weights.



# 2-Approximation Algorithm

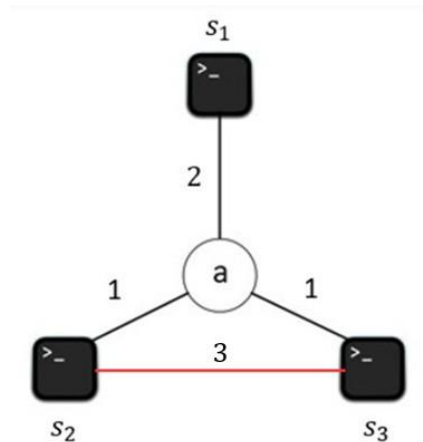
# Isolating Cut Heuristic $\rightarrow 2 - 2/k$ Approximation

Possible isolating cuts for  $s_1$ :

$$F_1 = \{(s_1, a)\} \rightarrow \text{cost} = 2$$

$$F_1' = \{(a, s_2), (a, s_3)\} \rightarrow \text{cost} = 2$$

(alternative choice)



Other isolating cuts:

$$F_2 = \{(a, s_2), (s_2, s_3)\} \rightarrow \text{cost} = 4$$

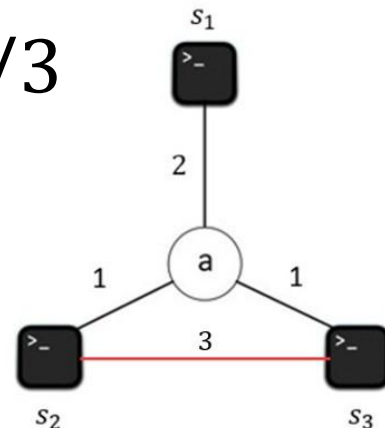
$$F_3 = \{(a, s_3), (s_2, s_3)\} \rightarrow \text{cost} = 4$$

# Isolating Cut Heuristic $\rightarrow 2 - 2/k$

## Approximation

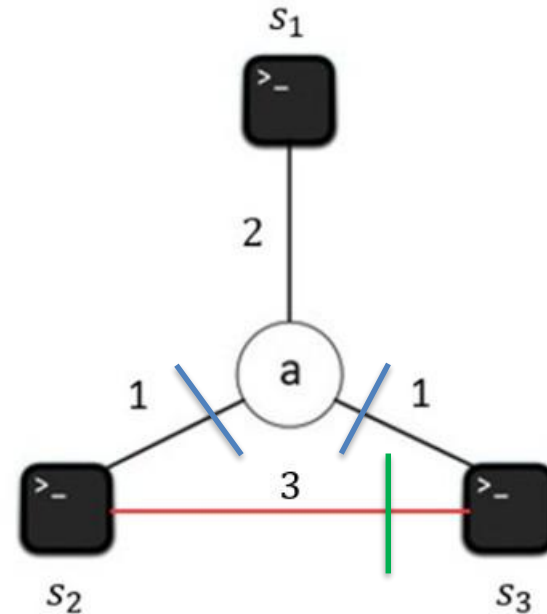
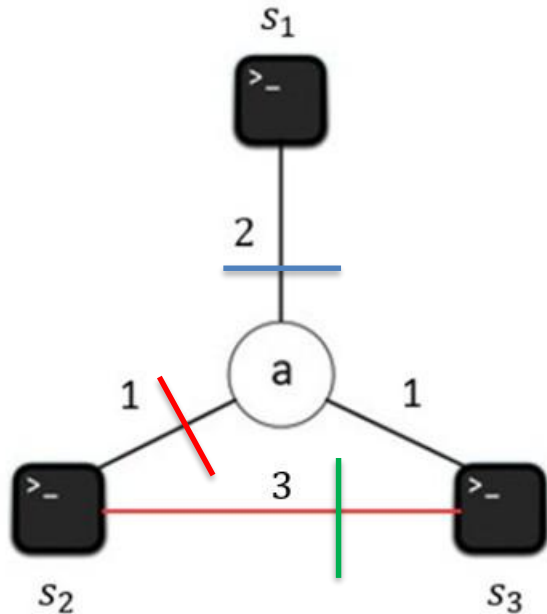
- Case A (choose  $F_1$  and  $F_2$ ; i.e.,  $s_1$  takes  $F_1$ )  
Union =  $\{(s_1, a), (a, s_2), (s_2, s_3)\} \rightarrow ALG = 6$   
With  $OPT = 5$ ,  
ratio =  $6/5 = 1.20 \leq (2 - 2/3) = 4/3$
- Case B (choose  $F_1'$  and  $F_2$ ; i.e.,  $s_1$  takes  $F_1'$ )  
Union =  $\{(a, s_2), (a, s_3), (s_2, s_3)\} \rightarrow ALG = 5$   
With  $OPT = 5$ , ratio =  $5/5 = 1.00 \leq 4/3$

# 2-2/k -Approximation Algorithm



# Compare Case A and Case B

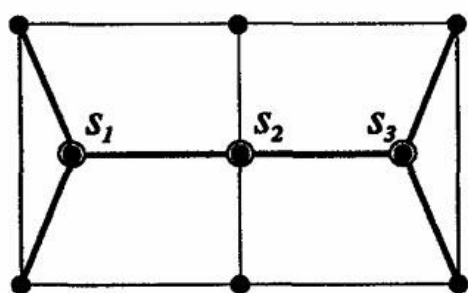
Case A (choose  $F_1$  and  $F_2$ ; i.e.,  $s_1$  takes  $F_1$ )



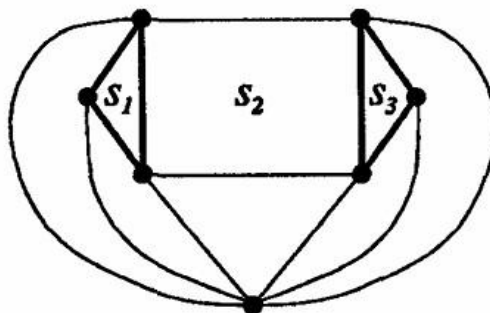
Case B (choose  $F_1'$  and  $F_2$ ; i.e.,  $s_1$  takes  $F_1'$ )

# Topological Meaning of Multiway Cuts — Separation, not Geometry

- Dahlhaus et al. (1992) used many planar and dual constructions, but their use of topology was not about geometry—it was about separability.
- In other words, they cared about whether there exists a set of edges that can separate each terminal into different connected components, such that those separating cuts are disjoint in topology.
- This “topological separability” defines when a graph is tractable:
  - When the cuts are separable (planar, disjoint regions) → solvable in polynomial time.
  - When cuts overlap (non-separable regions) → NP-hard.
- So, topology here means the structure of separability, not whether lines cross in a geometric drawing.

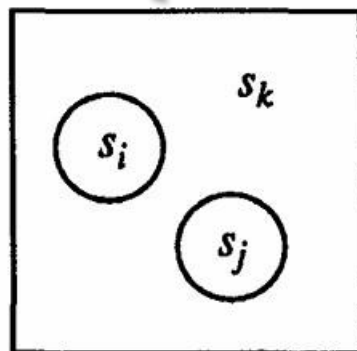


(a)

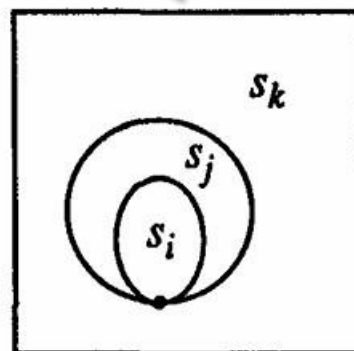


(b)

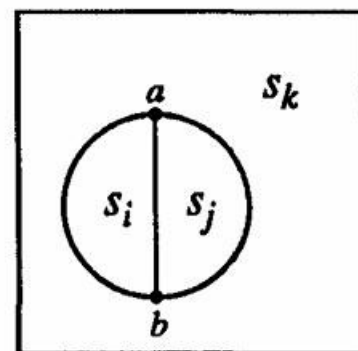
FIGURE 1. A planar 3-way cut (a) and its dual (b).



(a)



(b)



(c)

FIGURE 2. Types of 3-way cuts: Type I (a) and (b), Type II (c).

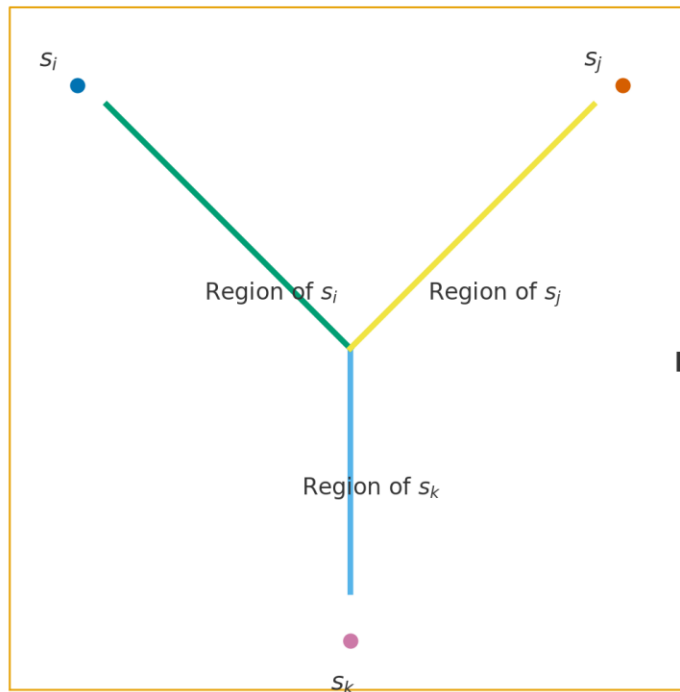


# Classification of 3-Way Cut Topologies

Type	Dual Graph ( $C^D$ ) Shape	Geometric Intuition	Meaning in Primal Graph
Type I	Two non-overlapping cycles (possibly tangent at one point)	Like two separate loops dividing the plane into three disjoint regions	Each cycle corresponds to a terminal's isolating cut; regions are topologically disjoint
Type II	Three boundary paths connecting the same pair of vertices (a–b)	Like a 'three-petal flower' or Y-shaped structure	Terminal regions are pairwise adjacent, sharing boundaries → non-disjoint separability

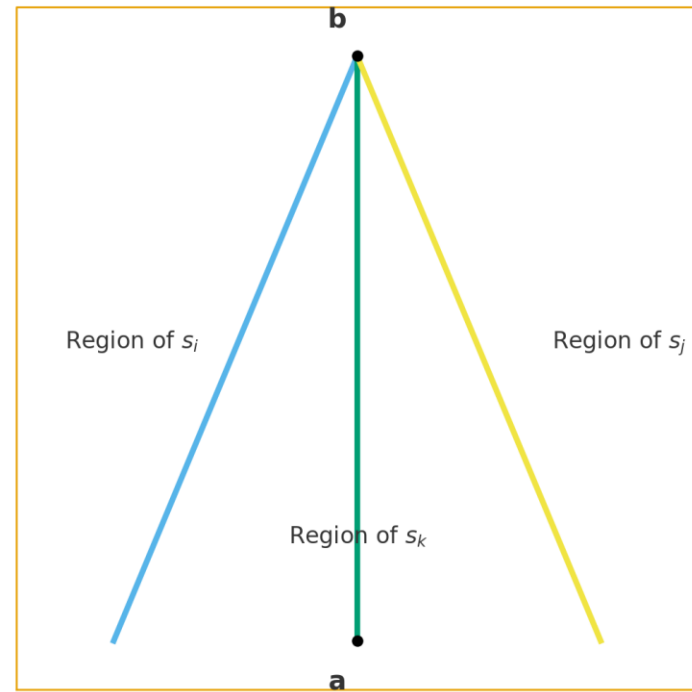
# Dahlhaus et al. (1992) *Type II 3-way cut*

Primal Graph  $G$  (Type II Y-shape cut)



Dual  $\leftrightarrow$

Dual Graph  $G^D$  (Three a-b paths, Type II)



# Our example belong to Type II

# Levels of Topological Breakdown in Multiway Cuts

情況	拓撲型態	幾何外觀	被破壞的性質	結果
直接加入 ( $s_2, s_3$ )	Edge cycle (Type II)	形成三角形	Edge disjointness 破 壞	強破壞 → NP- hard
共用節點 a 但 無 ( $s_2, s_3$ )	Node overlap (Type I)	區域在 a 接觸	Region disjointness 破 壞	弱破壞 → 仍 NP-hard
完全分離	Disjoint regions (Type I)	各區域互不接 觸	Topological DAG 保留	可 polytime 解

# Key Insight

- When terminals are directly connected, the graph loses its original **DAG-like hierarchical separability**.  
As a result, **isolating cuts begin to overlap**, leading to **duplicate edge counting** and increased total cost.  
Among all tractable cases, **planarity** is the last structural property that preserves **topological separability**.

# Summary

- **Planar graphs** possess a DAG-like topological separability, so the solution produced by the **Isolating Cut Heuristic** is close to the optimal ( $\approx$  OPT).
- In **non-planar graphs**, direct connections between terminals cause overlaps between cuts, resulting in higher costs and approximation errors.
- Thus, the loss of **planarity and separability** marks the **structural boundary between polynomial-time solvable and NP-hard cases**.

- **CS 598CSC: Approximation Algorithms**

*Instructor:* Chandra Chekuri

*Scribe:* Charles Blatti

*Lecture date:* February 11, 2009

- **Title:**

**“Multiway Cut and k-Cut Problems”**

- # 備註：這個版本是用**Chekuri, C.**的版本，以下的演算法比較是用該版本的數學表達，而非**David P. Williamson and David B. Shmoys.**(8.1章節教材的)

# 1.1 Isolating Cut Heuristic

## Input:

- An undirected weighted graph  $G = (V, E)$
- Edge weights  $w : E \rightarrow \mathbb{R}^+$
- A set of terminals  $S = \{s_1, s_2, \dots, s_k\} \subseteq V$

## Goal:

Find a set of edges  $A \subseteq E$  whose removal separates all terminals, minimizing the total weight  $w(A)$ .

## Algorithm:

1. **For each terminal  $s_i \in S$ :**
  - (a) Connect all other terminals  $S \setminus \{s_i\}$  to a new vertex  $t$  using edges of **infinite weight**.
  - (b) Compute the **minimum  $s_i - t$  cut** in this modified graph.
  - (c) Let  $E_i$  denote the set of edges in that minimum isolating cut.
2. **Sort the isolating cuts  $E_1, E_2, \dots, E_k$**   
by their total weight  $w(E_i)$ ,  
so that  $w(E_1) \leq w(E_2) \leq \dots \leq w(E_k)$ .
3. **Select the  $(k - 1)$  lightest cuts:**

$$A = E_1 \cup E_2 \cup \dots \cup E_{k-1}.$$

## 4. Output:

The edge set  $A$  as the **multiway cut**.

# 1.2 Greedy Splitting Algorithm

Input:

- An undirected weighted graph  $G = (V, E)$
- Edge weights  $w : E \rightarrow \mathbb{R}^+$
- A set of terminals  $S = \{s_1, s_2, \dots, s_k\} \subseteq V$

Goal:

Find a minimum-weight set of edges  $A \subseteq E$

whose removal separates all terminals into distinct connected components.

Algorithm:

1. Initialization:

Start with the entire graph as one partition:

$$P_0 = \{V\}.$$

2. Iteratively split the graph:

For each iteration  $i = 1, 2, \dots, k - 1$ :

1. Find the **cheapest cut** in the current graph that divides one of the existing components into two smaller components, such that after the split, **each component contains at least one terminal**.
2. Add the edges of this cut to the solution set.
3. Update the partition:

$$P_i = P_{i-1} \text{ with one component split into two.}$$

3. **Stop** when there are exactly  $k$  components, each containing one distinct terminal.

4. Output:

The union of all edges removed during the process as the **multiway cut**.



Aspect / 比較面向	1.1 Isolating Cut Heuristic (ICH)	1.2 Greedy Splitting Algorithm (GSA)
Algorithm Type / 類型	Parallel independent heuristic ( 平行獨立啟發式 )	Iterative greedy algorithm ( 逐步貪婪演算法 )
Basic Idea / 核心概念	For each terminal, find the minimum isolating cut independently, then take the union of the $k-1$ lightest cuts.	Start from the full graph and repeatedly find the cheapest cut to split one component, until there are $k$ components.
Execution Order / 執行順序	All cuts are computed <b>independently and simultaneously</b> .	Cuts are found <b>one by one</b> ; each new cut depends on the previous partition.
Graph Update / 是否更新圖結構	❌ No — Each isolating cut is computed on the original graph.	✅ Yes — Each iteration refines the current partition of the graph.
Greedy Nature / 是否為貪婪法	Weakly greedy (選最小的 $k-1$ cuts, but not iterative).	Strictly greedy (每步都根據當前狀態選最便宜的 cut).
Computation Structure / 計算架構	Parallelizable — all isolating cuts can be solved concurrently using max-flow.	Sequential — each iteration's result affects the next decision.
Mathematical Analysis / 分析重點	Uses the property $2w(E^*) = \sum_i w(\delta(V_i))$ ; discards one heaviest cut to get $2 - 2/k$ .	Uses induction on partition refinement; each split adds at most $w(\delta(V_h))$ , summing to $2 - 2/k$ .
Approximation Ratio / 近似比	$2 - \frac{2}{k}$	$2 - \frac{2}{k}$
Advantages / 優點	Simple, parallel, easy to implement; intuitive flow-based structure.	True greedy behavior; theoretical proof generalizes to $k$ -Cut and Gomory-Hu Tree methods.
Disadvantages / 缺點	Lacks dynamic adjustment; may double-count overlapping cuts.	Requires multiple min-cut computations; sequentially dependent.
Interpretation / 直觀比喻	"Cut everyone apart first, then keep the cheapest $k-1$ ."	"Split one region at a time, always choosing the cheapest next split."

# 備註(我自己要深入進去看)

- Min-Cut / Max-Flow 定理

# Min-Cut / Max-Flow 定理

## ✿ 一、故事直覺：水流與水壩的比喻

想像你有一張網路圖 ( Graph ) :

nginx

複製程式碼

s → → → t

- 每條邊代表一條「管線」或「水道」。
- 每條邊有容量 (capacity)  $c(u, v)$  ,  
表示最多能通過的水量。
- 你的目標：從 source s 把水盡可能多地送到 sink t。

🟡 在這張圖中有兩種問題：

類別	問題名稱	問什麼？
流量問題	Max-Flow	在不違反容量限制的前提下，最多能流多少水？
切割問題	Min-Cut	若要完全阻止水流，最少要「切掉」多少管線？

結果：

這兩個問題雖然看起來相反，  
但它們的最佳解數值永遠相等。  
這就是 最大流最小割定理 ( Max-Flow Min-Cut Theorem ) 。

# Min-Cut / Max-Flow 定理

## ✚ 二、形式定義 (數學版)

- ◆ 給定一張有向圖  $G = (V, E)$  .

每條邊有容量  $c(u, v) \geq 0$  .

指定 source  $s$  與 sink  $t$  .

- ◆ 目標：找到最大流量  $f$

$$\max \sum_{(s,v) \in E} f(s,v)$$

subject to

$$\begin{cases} f(u,v) \leq c(u,v) & \text{容量限制} \\ f(u,v) = -f(v,u) & \text{反向流} \\ \sum_{(u,v)} f(u,v) = 0, & \forall u \in V \setminus \{s, t\} \text{ (流守恆)} \end{cases}$$

# Min-Cut / Max-Flow 定理

## ◆ 最小割問題 ( Min-Cut )

選一個節點子集  $S \subseteq V$ ，使  $s \in S, t \notin S$ 。

割集 (cut) 定義為所有從  $S$  指向  $V-S$  的邊：

$$C(S) = \{(u, v) \in E \mid u \in S, v \notin S\}$$

其「容量」是

$$\text{cap}(S) = \sum_{(u,v) \in C(S)} c(u, v)$$

目標：

$$\min_S \text{cap}(S)$$

# Min-Cut / Max-Flow 定理

## ✚ 三、定理內容

**Max-Flow Min-Cut Theorem (Ford & Fulkerson, 1956):**

在任意網路  $G$  中，最大可行流的流量

等於最小  $s$ - $t$  cut 的總容量。

$$\max_f |f| = \min_S \text{cap}(S)$$

這代表：

- 找到最大流量  $\implies$  自動知道最小割容量。
- 找到最小割  $\implies$  自動知道最大流量。

它們是 線性規劃的強對偶關係。

# Min-Cut / Max-Flow 定理

## ✦ 四、幾何直覺 (平面圖的對偶)

在平面圖中：

原圖

對偶圖

一條被切掉的邊

對偶圖上一段邊界曲線

最小割

對偶圖上最短路徑

最大流

對偶圖上最短距離 (電位差)

所以：「在原圖找最小 cut」=「在對偶圖找最短路徑」。

這是 Multiway Cut 延伸時「planarity 與 duality」概念的基礎。

# Min-Cut / Max-Flow 定理

## ✿ 五、演算法流程簡述

1. 初始化流量  $f=0$ 。
2. 尋找可增廣路徑 (augmenting path) :  
從  $s \rightarrow t$  找出沿途還有剩餘容量的路徑。
3. 沿路徑增加流量。
4. 重複步驟 2-3，直到找不到增廣路徑。
5. 結果：
  - 流量值 = 最小割容量。
  - $S$  集合 = 從  $s$  可達節點；  
     $T$  集合 = 其他節點。
  - $E(S, T)$  即最小割邊集。

這就是 Edmonds-Karp 或 Dinic 演算法的核心思想。



# Min-Cut / Max-Flow 定理

- **Max-Flow Min-Cut** 定理說明：  
「阻止水流的最小代價」 = 「能流出的最大水量」。
- 幾何上是「最短路徑 vs. 最小割」的對偶；  
演算法上是「增廣流 vs. 切斷路」的對偶；  
理論上是「Primal-Dual 線性規劃」的強對偶關係。

(By ChatGPT)

# Reference

- **Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis.**  
*The Complexity of Multiway Cuts.*  
*SIAM Journal on Computing*, Vol. **23**, No. **4**, pp. **864–894**, 1994.  
(Preliminary version appeared in *Proc. 24th ACM Symposium on Theory of Computing (STOC)*, 1992.)
- **David P. Williamson and David B. Shmoys.**  
*The Design of Approximation Algorithms.*  
Cambridge University Press, 2011.  
(Chapter 8.1: Multiway Cut Problem and Minimum-Cut-Based Algorithm.)
- Chekuri, C. (2009, February 11). *Lecture 7: Multiway Cut and k-Cut Problems.*  
CS 598CSC: Approximation Algorithms. University of Illinois at Urbana-Champaign.