University of Michigan–Ann Arbor

Department of Electrical Engineering and Computer Science

EECS 498 004 **Advanced Graph Algorithms**, Fall 2021

Lecture 16: Approximating Expansion via The Cut-Matching Game

October 29, 2021

Instructor: Thatchaphol Saranurak

Scribe: Chaitanya Nalam

Finding exact conductance (expansion) of a graph is NP-hard. In this course we have seen two methods to approximate the conductance.

- Cheeger's Inequality where we find a cut S where $\Phi_G(S) \leq \phi$ or reports $\Phi(G) \geq \Omega(\phi^2)$ in $O(\frac{m}{\phi^2})$. Approximation ratio is $\frac{1}{\phi}$ and can be very bad when ϕ is small.
- Leighten-Rao relaxation finds a cut S where $\Phi_G(S) \leq \phi$ or reports $\Phi(G) \geq \Omega(\phi/\log n)$. Although approximation ratio is good i.e. is $O(\log n)$. It is slow and takes the time to solve LP.

Today we are going to see an algorithm using the cut-matching game described in previous lecture, that can give best of both worlds i.e. good approximation and fast running time. In particular

- Approximation ratio: $O(\log^2 n)$ (#rounds in the cut-matching game)
- Time: $O(\log^2 n)$ max flow calls. (Best known running time for max flow is $\tilde{O}(m + n^{1.5})$ time).

We will see two applications based on cut-matching game which are:

1. Approximating expansion/conductance
2. Finding balanced sparse cuts

The cut-matching game framework is very flexible and can be used to solve many variants of the above problems. However, in this notes we focus on undirected, unweighted graph with n vertices and m edges.

1. Approximating Expansion

Recall that the expansion of a graph is denoted by $\Psi(G)$ and defined as

$$\Psi(G) = \min_{S \subseteq V} \frac{|E(S, V \setminus S)|}{\min\{|S|, |V \setminus S|\}}$$

where $E(S, V \setminus S)$ is the edges set crossing the cut S. We will discuss later how we can change the algorithm to make it work for approximation conductance or $d-$expansion.

To approximate expansion to $O(\log^2 n)$ factor given a graph $G = (V, E)$ and parameter ϕ, either

- Find a cut S where $\Psi_G(S) \leq \phi$ (or)
- Report that $\Psi(G) \geq \Omega(\phi/\log^2 n)$.

In the first case if we return a sparse cut S then we are done, however in second case what does it mean by reporting $\Psi(G) \geq \Omega(\phi/\log^2 n)$.

What is the certificate or witness for saying that it has $\Omega(\phi/\log^2 n)$ expansion?

Answer: Embedding an expander of expansion $\Omega(1)$ in the graph G with at most $O(\log^2 n/\phi)$ congestion proves that expansion of the graph is high as well.

1.1. Lemma. *If* $X \preccurlyeq^{\text{flow}} O(\frac{\log^2 n}{\phi}) \cdot G$ *and* $\Psi(X) \geq \Omega(1)$, *then* $\Psi(G) \geq \Omega(\phi/\log^2 n)$.

Proof. For any $S \subseteq V$, we have

$$\begin{aligned} |E_G(S, V \setminus S)| &\geq \Omega(\frac{\phi}{\log^2 n}) \cdot |E_X(S, V \setminus S)| && \text{as } X \preccurlyeq^{\text{cut}} O(\frac{\log^2 n}{\phi}) \cdot G \\ &\geq \Omega(\frac{\phi}{\log^2 n}) \cdot \min\{|S|, |V \setminus S|\} && \text{as } \Psi(X) \geq \Omega(1) \end{aligned}$$

□

How can we embed an expander? This is where cut-matching game comes to our rescue. According to cut-matching game expander can be thought of as union of carefully selected matchings. If we embed every matching with a lesser congestion i.e. $O(1/\phi)$ then we can embed the whole expander which is union of $O(\log^2 n)$ matchings with at most $O(\log^2 n/\phi)$ congestion as required.

So from now on our goal is to either find a sparse cut S (first case) or find a matching embedding with $1/\phi$ congestion (towards the second case).

2. A Key Subroutine: Sparse Cut (or) Matching Embedding

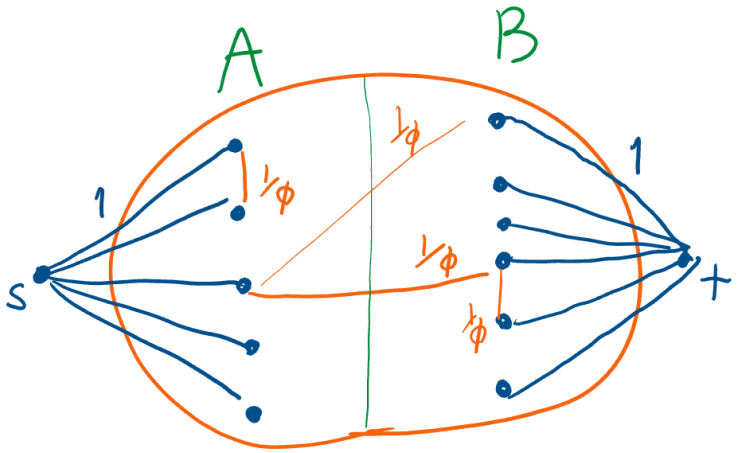
Formally we need to solve the following problem which will be a key subroutine to be used in cut-matching game.

Input: A graph $G = (V, E)$, a cut (A, B) where $|A| \leq |B|$, and a parameter ϕ

Output: Either

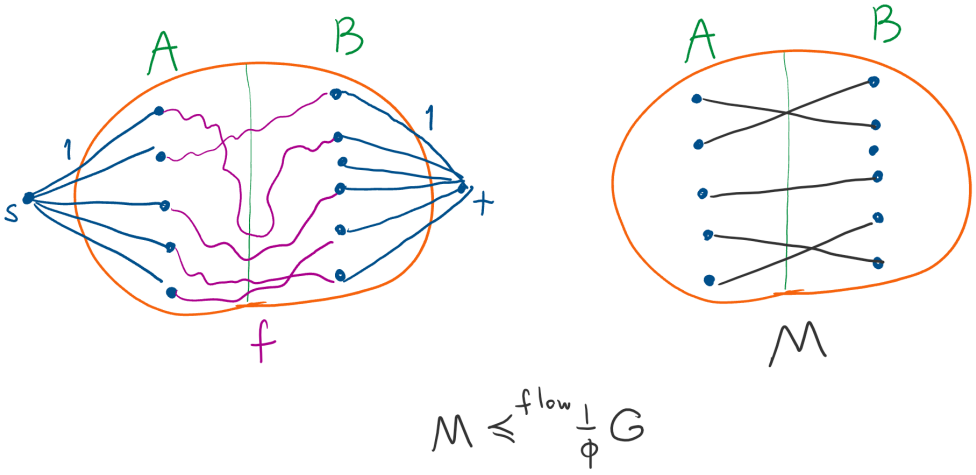
- A matching M between A and B of size $|A|$ and $M \preccurlyeq^{\text{flow}} \frac{1}{\phi} \cdot G$ (i.e. M is embedded into G with congestion at most $1/\phi$)
- A cut S where $\Psi_G(S) < \phi$.

In the graph G we are given a cut (A, B) with $|A| \leq |B|$. Consider the following (s, t)-flow problem where all the vertices in A are joined to a super source s and all the vertices in B are joined to a super sink t with capacities 1 and scale the capacities of edges in the graph G by $1/\phi$. Since it is unweighted graph initially new capacities would be equal to $1/\phi$. Let the resulting graph be $G_{s,t}$.



Solving the max flow min cut problem in $G_{s,t}$ will give us two cases. Either the max flow would be $|A|$ or less than it. It cannot be greater because the total out flow from s can have at most value equal to $|A|$.

Case 1: Max flow value is $|A|$. Let f be the corresponding $(s,t)-$ max flow. As f is an integral flow we can perform flow-path decomposition which can be done in $O(m)$ time using link-cut trees. There are $|A|$ paths exactly from s to t which have to go through a vertex in A and B as shown in figure below. For each of such path add the endpoints $a \in A, b \in B$ of the path (path excluding s,t) as an edge to set M.



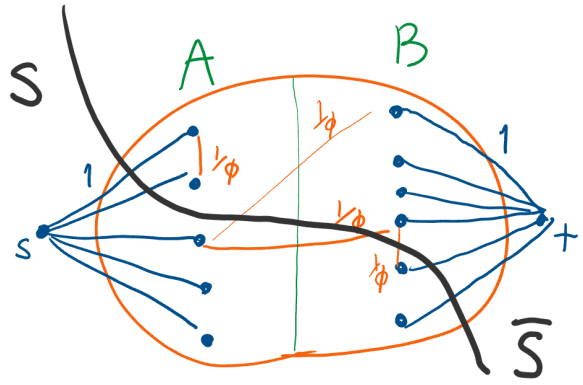
Observe that M is a matching because any vertex in A cannot send flow more than 1 receiving from s and any vertex in B cannot send more than 1 flow to t. Hence at most 1 path can be incident on any vertex as end points. Matching is of size $|A|$ and since this is a feasible flow returned by max flow algorithm which means congestion in original graph G is at most $1/\phi$.

$$M \preccurlyeq^{\text{flow}} \frac{1}{\phi} \cdot G$$

Case 2: Max flow is $< |A|$. So we have a cut whose size is $< |A|$ in graph $G_{s,t}$. Let that cut be S', where $s \in S'$ and $\overline{S'} = V(G_{s,t}) \backslash S'$.

$$c(E_{G_{s,t}}(S', \overline{S'})) < |A|$$

Consider the set $S = S' \setminus \{s\}$.



2.1. Claim. $\Psi_G(S) < \phi$.

Proof. From the picture above, observe that the flow from some vertices in $A \cap S$ is trapped in S and could not flow to other side to reach t even though the capacities are scaled by $1/\phi$. Hence this is a sparse cut. Formalized as follows.

$$|A \setminus S| + |B \cap S| + \frac{1}{\phi}|E_G(S, \overline{S})| = c(E_{G_{s,t}}(S, \overline{S})) < |A|$$
$$\frac{1}{\phi}|E_G(S, \overline{S})| < |A \cap S| - |B \cap S| \leq |S|$$

For another direction, we can also write

$$|B \setminus \overline{S}| + |A \cap \overline{S}| + \frac{1}{\phi}|E_G(S, \overline{S})| = c(E_{G_{s,t}}(S, V \setminus S)) < |A| \leq |B|$$
$$\frac{1}{\phi}|E_G(S, \overline{S})| < |B \cap \overline{S}| - |A \cap \overline{S}| \leq |\overline{S}|$$

So

$$|E_G(S, \overline{S})| < \phi \min\{|S|, |\overline{S}|\}.$$

□

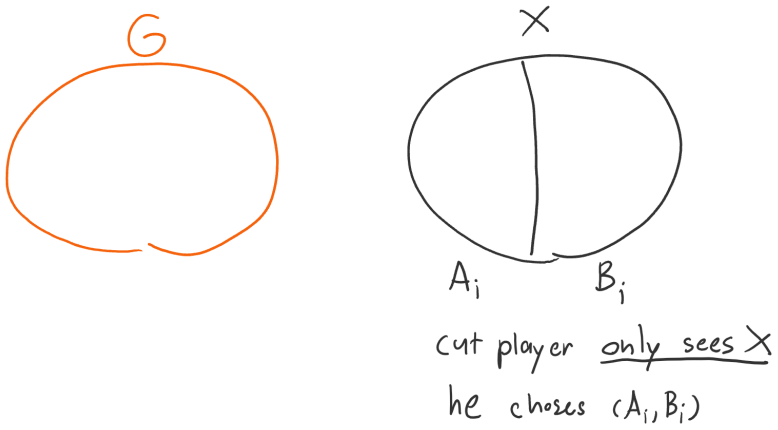
Now that we have embedded a matching we are one step closer to embedding an expander as it is union of matchings. Using this sub-routine and cut-matching game we formalize embedding an expander in the next section.

3. Combining the Two Ingredients

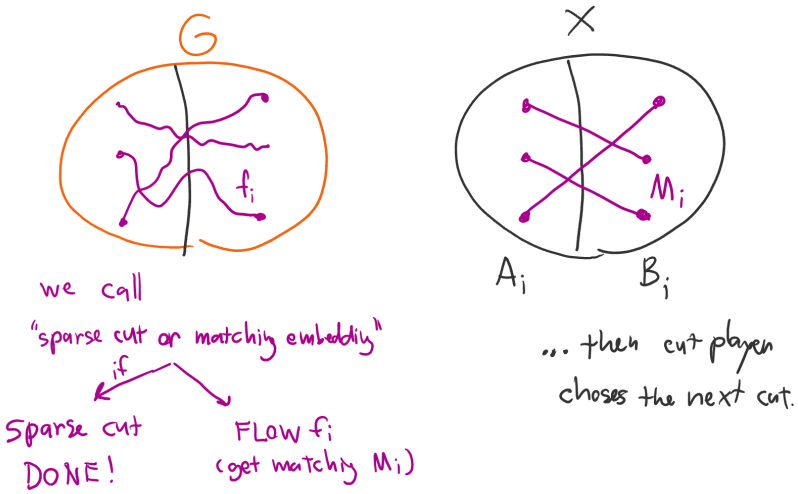
Recall that we have two players in the cut-matching game. Cut player chooses a cut (balanced cut containing sparse cut) from the graph constructed so far and matching player matches all the vertices on the smaller side by adding a matching to the graph.

For approximating expansion we either need to find a sparse cut or embed and expander. We combine our sub-routine "Sparse Cut or Matching Embedding" and cut-matching game as follows.

Cut player starts with an empty graph X and finds a cut A_i, B_i where i represents round in cut-matching game.



Given a cut A_i, B_i matching player calls the sub-routine "Sparse Cut or Matching Embedding". If a sparse cut is found then we are done by just returning that sparse cut in G, if not we get a flow from which matching player adds a matching to X between vertices in A_i, B_i. Observe that as per requirements of cut-matching game whenever we get a matching it perfectly matches every vertex in A_i.



Now again the cut player chooses a new cut A_{i+1}, B_{i+1} and game repeats.

There is a different way of thinking this process. One wants to find a sparse cut out of all possible subsets. However we want to be faster in that we should be finding sparse cut if it exists in $\tilde{O}(m)$ time instead of 2^n time. So we start searching for it by applying max flow min cut in an orientation (joining some vertices to s and other to t) and see if we can get a sparse cut. If we do not get one then we have to find different orientation of the graph where applying max flow

min cut will give us a chance to hit the sparsest cut. However we do not know which direction to proceed i.e. what vertices to combine to s and t.

This exact hint is given by X which keep tracks of all the directions you have explored till now. How? By storing all the matchings (directions) you have explored until now and suggesting you a new cut (which is sparse).

Why such a cut is useful hint to you? Because in X that "direction" characterized by the cut is having less edges across it. That means it is not explored enough and might have higher chance of containing the sparse cut.

Summarizing the formal algorithm:

Algorithm 1: $SparseCutOrExpanderEmbed(G, \phi)$

Initialize: $G = (V, E)$ be the input graph
$X_0 = (V, \{\})$ empty graph on V and it will be expander at the end
Let $R = O(\log^2 n)$ be the number of rounds in the cut-matching game
Result: Sparse cut (or) Expander embedding
for $i = 1, \cdots, R$ **do**
  Cut player reads X_{i-1} and choose a cut (A_i, B_i)
  Call Max flow on the constructed graph $G_{s,t}$ from (A_i, B_i)
  if *Flow value* $= |A_i|$ **then**
    Construct a matching from the end points of the flow's path decomposition;
    $M_i \preccurlyeq^{\text{flow}} \frac{1}{\phi} \cdot G$ and $X_i \leftarrow X_{i-1} \cup M_i$
  else
    return Sparse cut $S = S' \setminus \{s\}$ and we proved $\Psi_G(S) < \phi$
  end
end
$\Psi(X_R) = \Omega(1)$ by the cut-matching guarantee
$X_R \preccurlyeq^{\text{flow}} \frac{R}{\phi} \cdot G$ by the union of flows found so far
return X_R along with flow embedding as witness for $\Psi(G) \geq \frac{\phi}{R}$ as proved earlier.

Runtime analysis:

- Total running time of the cut-player:
  - $\tilde{O}(n)$ randomized (from previous class by KRV[1]).
  - $n^{1+o(1)}$ deterministic (by this paper[2]).
- Total running time of the matching player („Sparse Cut or Matching Embedding"):
  - $R = O(\log^2 n)$ max flow calls.

So in total runtime of $\tilde{O}(1)$ max flow calls we can either find a sparse cut S where $\Psi_G(S) \leq \phi$ or report that $\Psi(G) \geq \Omega(\phi/\log^2 n)$.

[1] https://dl.acm.org/doi/10.1145/1538902.1538903
[2] https://arxiv.org/pdf/1910.08025.pdf

4. Balanced Sparse Cut

Lets look at another problem of finding a balanced sparse cut which is useful has many applications. In the above sub-routine we found a sparse cut however there is no guarantee on how balanced it is. Sparse cut we obtain in previous sub-routine might have only few vertices. Formally the problem is as follows.

Problem:

Given a graph $G = (V, E)$ and parameters ϕ, β, we will either

- Find a cut S where $\Psi_G(S) \leq \phi$ and $\beta \leq |S| \leq n/2$, or
- Report that, for any cut S where $2000\beta R \leq |S| \leq n/2$, we have $\Psi_G(S) \geq \Omega(\phi/R)$.
  - Any cut that is much more balanced than β cannot be much sparse than ϕ.

4.1. Question. *How is such a guarantee useful?*

We can use it and do binary search to find

- "most balanced ϕ-sparse cut" or
- "sparsest β-balanced cut"

However, because of the nature of our guarantee we have to allow some gap of approximation in both expansion and balance.

As we have seen for previous problem, what does it mean to say that no cut which is more balanced is not much worse. In this case we need to return something that witnesses this guarantee. So we define our algorithmic goal as follows and we also prove why such an algorithmic goal would imply the required guarantee.

Our Algorithmic Goal: Given a graph $G = (V, E)$ and parameters ϕ, β, we will either

- Find a cut S where $\Psi_G(S) \leq \phi$ and $|S| \geq \beta$, or
- Return an "almost-expander" X' and a set F of "fake edges"
  - $\Psi(X) \geq 1/1000$ where $X = X' \cup F$
  - $X' \preccurlyeq^{\text{flow}}_{\frac{R}{\phi}} \cdot G$
  - $|F| \leq \beta R$

Note that we embed X' an "almost expander" into G instead of X which is an expander. X' is "almost expander" because it is close to becoming an expander by the addition of set of edges F which is of size at most βR. We call F "fake" because we do not embed edges in F into G.

Below we prove the lemma why such a witness would imply the required guarantee.

4.2. Lemma. *Given the above X' and F, every cut S where $2000\beta R \leq |S| \leq n/2$ is such that $\Psi_G(S) \geq \Omega(\phi/R)$.*

Proof. For any $S \subseteq V$ of large enough size i.e. $|S| \geq 2000\beta R$, we have

$$|E_X(S, V \setminus S)| \geq |S|/1000 \geq 2\beta R$$

First inequality follows as X is an $\Omega(1)-$expander. As we will prove later that $|F| \leq \beta R$ we have that

$$|F| \leq |E_X(S, V \setminus S)|/2.$$

Since, F is at most half the size of cut $(S, V \setminus S)$ in X. Cut size of $(S, V \setminus S)$ in X' is at least half the size of the cut in X as X and X' just differ by the edges in F. (Note that this is only true when the cut is more balanced i.e. $|S| \geq 2000\beta R$)

$$|E_{X'}(S', V \setminus S')| \geq |E_X(S, V \setminus S)| - |F| \geq |E_X(S, V \setminus S)|/2.$$

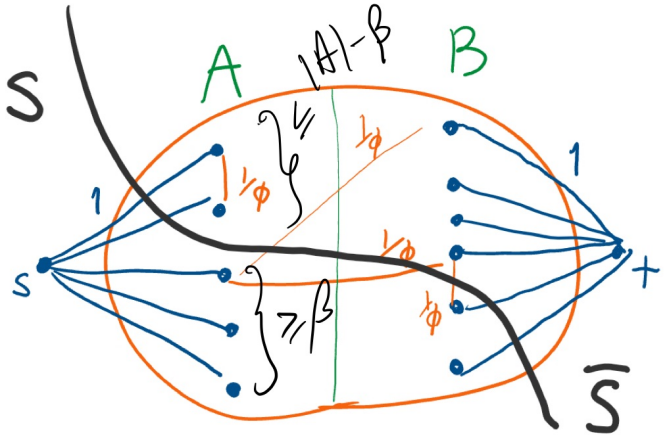
$$\begin{aligned}
|E_G(S, V \setminus S)| &\geq \frac{\phi}{R} \cdot |E_{X'}(S, V \setminus S)| && \text{as } X' \preccurlyeq^{\text{cut}} \frac{R}{\phi} \cdot G \\
&\geq \frac{\phi}{R} \cdot \frac{|E_X(S, V \setminus S)|}{2} && \text{as shown above for } |S| \geq 2000\beta R \\
&\geq \Omega(\frac{\phi}{\log^2 n}) \cdot \min\{|S|, |V \setminus S|\} && \text{as } \Psi(X) \geq \Omega(1) \\
\Psi_G(S) &\geq \Omega(\frac{\phi}{\log^2 n})
\end{aligned}$$

□

We solve the "Balanced Sparse Cut or Matching Embedding with Fake edges" similar to above problem. Instead of dividing into two cases based on value of flow at $|A|$ our new threshold would be $|A| - \beta$.

Why such a threshold would help?

Because when the value of flow is $\leq |A| - \beta$ there are at least β vertices that belong to A which are trapped in the cut $S = S' \setminus \{s\}$ as shown in diagram below. So the cut S is $\beta-$balanced.



If the flow is $> |A| - \beta$ then the matching M' would be of size at least $|A| - \beta$ so we get a flow embedding $M' \preccurlyeq^{\text{flow}} \frac{1}{\phi} G$. However because we want the matching to be of size $|A|$ i.e. whole of A should be matched for the cut-matching game to proceed. We now introduce some "fake" edges F that fill the gap and they can be at most β i.e. $|F| \leq \beta$ as we have got at least $|A| - \beta$ flow. But these are the edges created by us and not end points of flow path decomposition these are not embedded in G. Note that cut-matching game does not impose any constraints on the matching given by the matching player except that it is $A-$perfect. Since $|B| \geq |A|$ there exist β many vacant vertices on B side that we can match to unmatched vertices on A to form F whose union with M' gives $M = M' \cup F$.

Note that union of matchings M form an expander X after R rounds. But what is embedded in G is X' which is the union of edges in M' obtained in each of R rounds. Note that set of fake edges F are bounded by βR as we only get at most β in each round.

Given above explanation it is an easy exercise to show formally this works.

4.3. Exercise. Consider the graph $G_{s,t}$ with super source s and super sink t as defined before. Let F be the (s,t)-max flow value in $G_{s,t}$.

- If $|F| \geq |A| - \beta$, then we can find M' and F above.
- If $|F| < |A| - \beta$, then we can find S above.

4.4. Exercise. Show how to combine "Balanced Sparse Cut or Matching Embedding with Fake Edge" subroutine with the cut-matching game to solve the goal.

5. Extending to Capacitated graphs

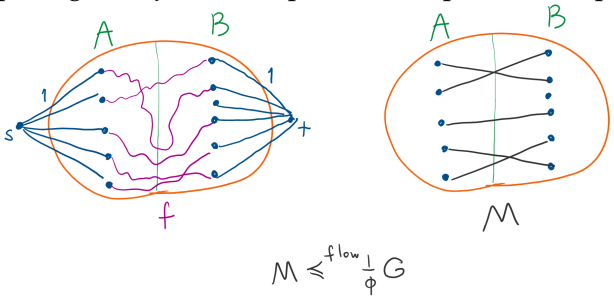
Till now we have assumed that our graph is unweighted however we will show in this section that cut-matching framework is good for extending to other variants. There are some problems though which we will see as we proceed and discuss solutions for the same.

Note that the definition of the expansion is different.

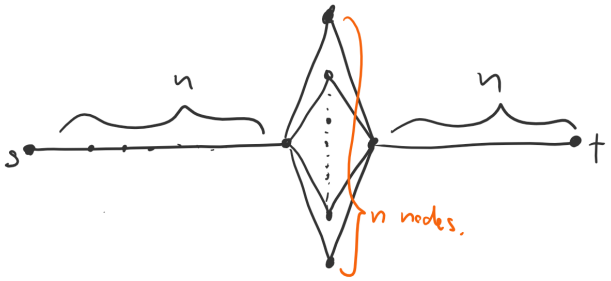
$$\Psi(G) = \min_{(S,V\setminus S)} \frac{c(E(S, V \setminus S))}{\min\{|S|, |V \setminus S|\}}.$$

Problem 1:

Flow-path decomposition in "Sparse Cut or Matching Embedding" subroutine. Recall that, after computing flow f, we compute a flow-path decomposition f.



In capacicated graph, explicit flow-path decomposition might take $\Omega(mn)$ time. In fact, just the total length of paths (ignore computation time) might be $\Omega(mn)$. For example, look at max flow in this graph



In general, there can be $\Omega(m)$ paths, each of length $\Omega(n)$. This is called the flow-decomposition barrier. It cannot be more than m paths because each and every time we extract a path we are maxing out at least one edge that belongs to that path making the residual capacity to 0 and this can be done at most m times.

However, using link-cut trees data structure again, we can still extract **the endpoints of these paths** in $O(m \log n)$ time, without listing the paths themselves.

From this, we get a **fractional matching** M where

- M has at most m non-zero weighted edges. (Before, it was an integral matching with at most n edges).
- M still has matching value $|A|$, i.e. $\sum_{e \in E(M)} w(e) = |A|$.

Problem 2:

Fractional (not integral) matching in the cut-matching game.

In round i of the cut-matching game, given a cut (A_i, B_i), we get a fractional matching M_i of size $|A_i|$ instead of an integral matching. Although we are not formally proving it here Cut-matching game would extend seamlessly to this setting. But the total running time of the cut player is $\tilde{O}(m)$ instead of $\tilde{O}(n)$ as fractional matching consists of at most $O(m)$ edges in each round because of which cut player have to find a cut (A_i, B_i) from a graph of size $\tilde{O}(m)$.

6. $d-$expansion

Cut matching game can be extended for approximating the general version of expansion i.e. $d-$expansion $\Phi_d(G)$. We need to adjust the two main ingredients of the cut matching game which are:

- Sparse Cut or $d-$Matching Embedding
- The Cut Matching Game for constructing $d-$expander

Sparse Cut or $d-$Matching Embedding:

- **Input**: a graph $G = (V, E)$, a cut (A, B) where $\boldsymbol{d}(A) \leq \boldsymbol{d}(B)$, and a parameter ϕ

- **Output:** return either
  - A "bipartite $\boldsymbol{d}$-matching" M between A and B of size $\boldsymbol{d}(A)$ such that $M \preccurlyeq^{\text{flow}} \frac{1}{\phi} \cdot G$. That is,
    - for all $a \in A$, $\deg_M(a) = \boldsymbol{d}(a)$
    - for all $b \in B$, $\deg_M(b) \leq \boldsymbol{d}(b)$
  - A cut S where $\Phi_{G,\boldsymbol{d}}(S) < \phi$.

Note that above sub routine can be solved by max flow min cut by joining the vertices to either s or t with capacities of their respective d values. We have to flow-path decomposition to get the endpoints of the $d-$Matching too. We restrict ourselves to integer $d-$expansions so that we can get an integral matching possibly with multi edges.

A Cut Matching Game (degree profile d version)

Given a target degree profile $\boldsymbol{d} : [n] \to \mathbb{Z}_{\geq 1}$, we can formally show how to construct an expander $G = (V, E)$ with n vertices and where each node v has degree $\boldsymbol{d}(v) \leq \deg_G(v) \leq \boldsymbol{d}(v) \cdot O(\log^2 n)$ using the cut-matching game framework, where in each round the matching player returns a "bipartite $\boldsymbol{d}$-matching".

The number of rounds should be $O(\log^2 n)$. The running time per round of the cut-player should be poly(n) or even $\tilde{O}(|E(G)|)$. That is, they should be independent from $\boldsymbol{d}(V)$.

7. Conclusion: Flexibility of The Cut-Matching Framework

- The framework of the cut-matching game for approximating expansion is very very flexible.
- We saw that it is useful for approximating
  - Expansion, conductance, and $\boldsymbol{d}$-expansion. (But not H-expansion for general H)
  - Balanced version for all of the above.
  - But these are kinds of edge expansion in undirected graphs (captured by theory we saw in the first half of the course).
- **Important:** But the cut-matching framework also works for
  - Vertex expansion
  - Directed graph version of all the above
  - Hypergraph version of all the above
- The total running time of each version $\approx$ polylog(n) calls to max flow on that graphs.

  For example,
  - vertex expansion: vertex capacitated max flow
  - many kinds of expansion in hypergraph: max flow on hypergraphs (which is even a special case of vertex capacitated max flow)

– many kinds of expansion in directed graphs using max flow on directed graphs

7.1. *Remark.* All variants of "Sparse Cut or Matching Embedding" can be solved (with slightly worse guarantee) using approximate max flow too.

- We showed that approximate sparsest cuts and balanced cuts reduce to polylog(n) max flow calls.
- Actually, they can be reduced to polylog(n) approximate max flow calls.
  - There are fast approximate max flow algorithms in **undirected** graphs
    - $\tilde{O}(m)$ time: edge capacitated graphs[3]
    - $m^{1+o(1)}$ time: vertex capacitated graphs, and hypergraphs[4]
  - So the total running time is almost always almost-linear in undirected graphs.

8. Reflection: How did we avoid solving multi-commodity flow?

Let Graph G be with degree profile $\boldsymbol{d}$ and K be the $\boldsymbol{d}$-product graph with same degree profile. Graph G can have conductance at least ϕ iff G can allow a flow embedding of $\phi \cdot K$ graph with at most $O(\log n)$ congestion i.e.

$$\Phi(G) \geq \phi \iff \phi K \preccurlyeq^{\text{flow}} O(\log n) G$$

So the largest value of ϕ for which we can embed $\phi \cdot K$ in G is the conductance of the graph G up to $\log n$ factors. Essentially approximating conductance of a graph is same as solving the multi-commodity flow problem by embedding K the $d-$product graph which have $|E(K)| = O(n^2)$ demands.

However there is another idea to reduce the number of demands that we need to embed. From flow view of expander we also know that for any $\tilde{\Omega}(1)$-expander X with degree profile $\boldsymbol{d}$ we have $X \approx^{\text{cut}}_{\text{polylog}(n)} K$ and so up to a polylog(n) factor,

$$\Phi(G) \geq \phi \iff \phi X \preccurlyeq^{\text{flow}} G$$

which need only $\tilde{O}(n)$ demands to be embedded.

So how did we approximate the conductance of the graph G without solving the maximum congestion flow problem?

Problem: As long as we fix an expander X_0, checking if $\phi X_0 \preccurlyeq^{\text{flow}} G$ is a multi-commodity flow problem with $\Omega(n)$ demands.

Idea: The best thing that *single-commodity flow subroutine* can do is embed a large matching by just joining the matching vertices by creating a super sink and super source which just increase the number of vertices by 2. But we cannot embed arbitrary matchings as they may not help us finally embed an expander X.

Solution: The cut-matching game is developed for this situation. It "forces" the union of arbitrary matchings to become an expander X by cleverly choosing the cuts in each round. Note that

[3]https://arxiv.org/pdf/1304.2077.pdf
[4]https://arxiv.org/abs/2101.07149

X is not fixed and not chosen by us, but it is an expander and that is good enough. Moreover this takes just $O(\log^2(n))$ *single-commodity flow* calls but embedding a matching each time.

This is the real motivation behind the development of the cut-matching game.