Online Matching with Recourse: Random Arrivals

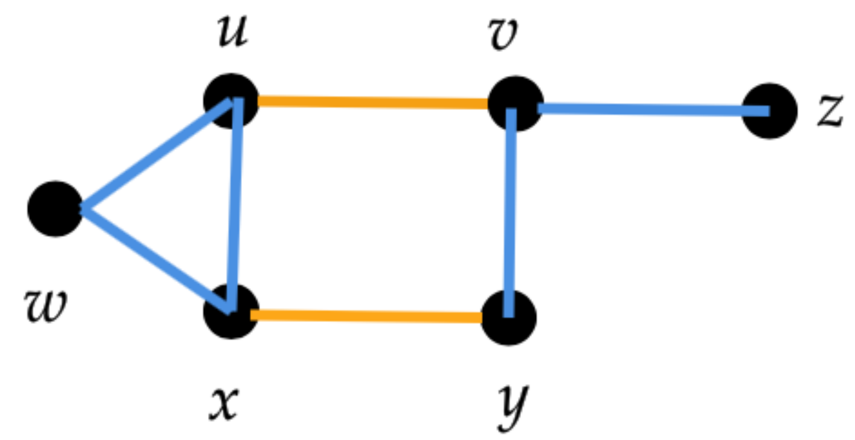
Matching

Matching

- Given a graph G, with vertex set V, and edge set E, a matching is a set of pairwise non-adjacent edges.

Matching

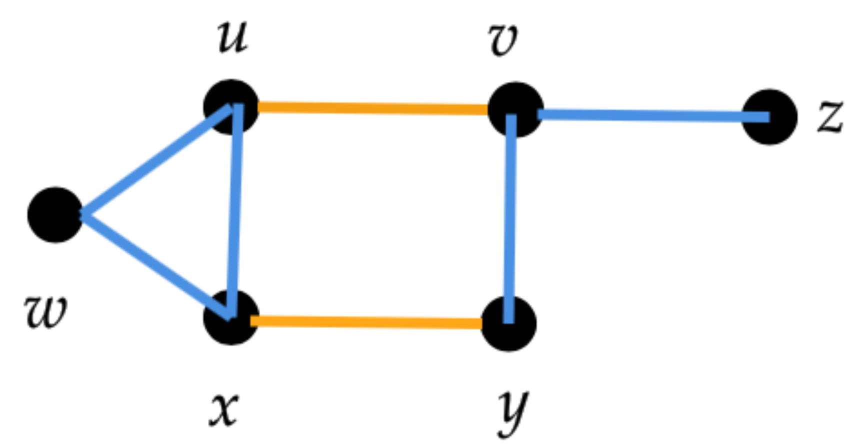
- **Given a graph G, with vertex set V, and edge set E, a matching is a set of pairwise non-adjacent edges.**



Matching

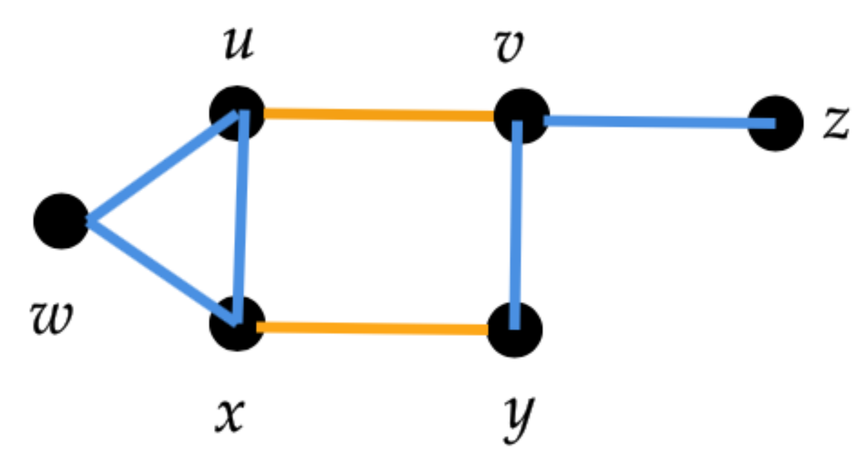
- Given a graph G, with vertex set V, and edge set E, a matching is a set of pairwise non-adjacent edges.

- Maximum matching is a matching that contains the largest possible number of edges.

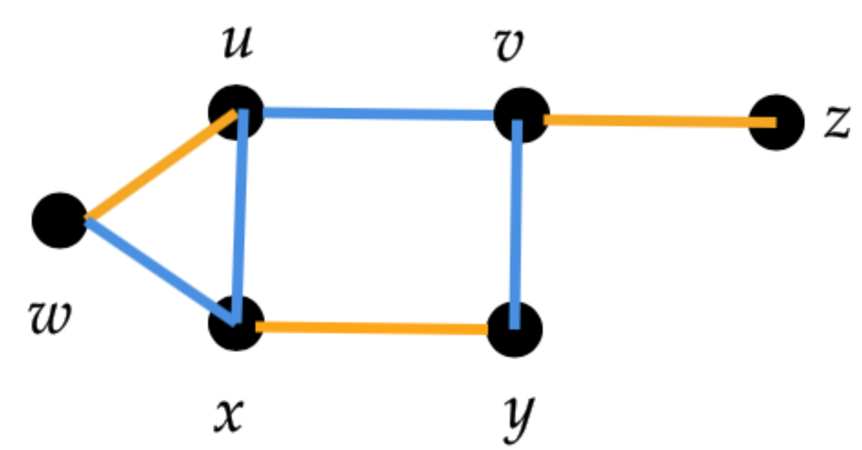


Matching

- Given a graph G, with vertex set V, and edge set E, a matching is a set of pairwise non-adjacent edges.



- Maximum matching is a matching that contains the largest possible number of edges.



Maximum matching

Applications

Applications

- **Ad-Allocation:** Ad slots are allocated through contracts. There are demand and supply constraints that directly lead to the question of finding an optimal matching between the slots and advertisers.

Applications

- **Ad-Allocation:** Ad slots are allocated through contracts. There are demand and supply constraints that directly lead to the question of finding an optimal matching between the slots and advertisers.

- **Job Scheduling:** We have a set of servers with different capabilities available to process jobs from persistent sources - jobs that need to be processed over long periods of time.

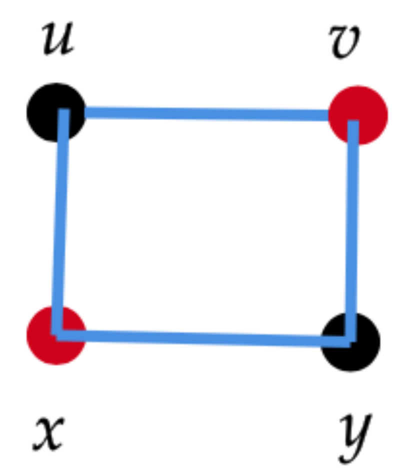
Bipartite Graphs

Bipartite Graphs

- A graph with vertex set V and edge set E, is called a bipartite graph if V can be partitioned into sets V_1 and V_2 such that all the edges are between vertices in V_1 and V_2 .

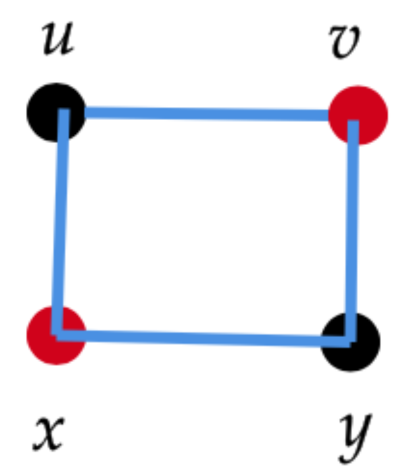
Bipartite Graphs

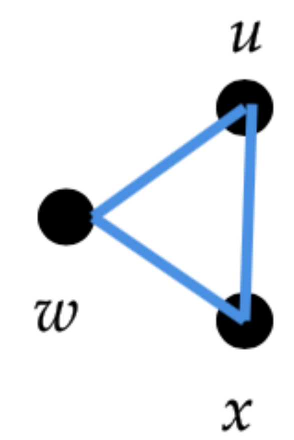
- A graph with vertex set V and edge set E, is called a bipartite graph if V can be partitioned into sets V_1 and V_2 such that all the edges are between vertices in V_1 and V_2.



Bipartite Graphs

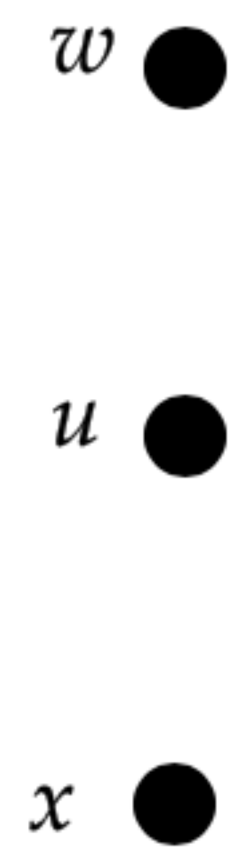
- A graph with vertex set V and edge set E, is called a bipartite graph if V can be partitioned into sets V_1 and V_2 such that all the edges are between vertices in V_1 and V_2 .





Online Model

- **Typically, a bipartite graph G=(U,V,E). The set U is known to the algorithm. Vertices in V arrive one at a time, and reveal edges incident on them.**
- **The goal is to match (or forego) a vertex as soon as it arrives.**
- **The decisions made are irrevocable.**

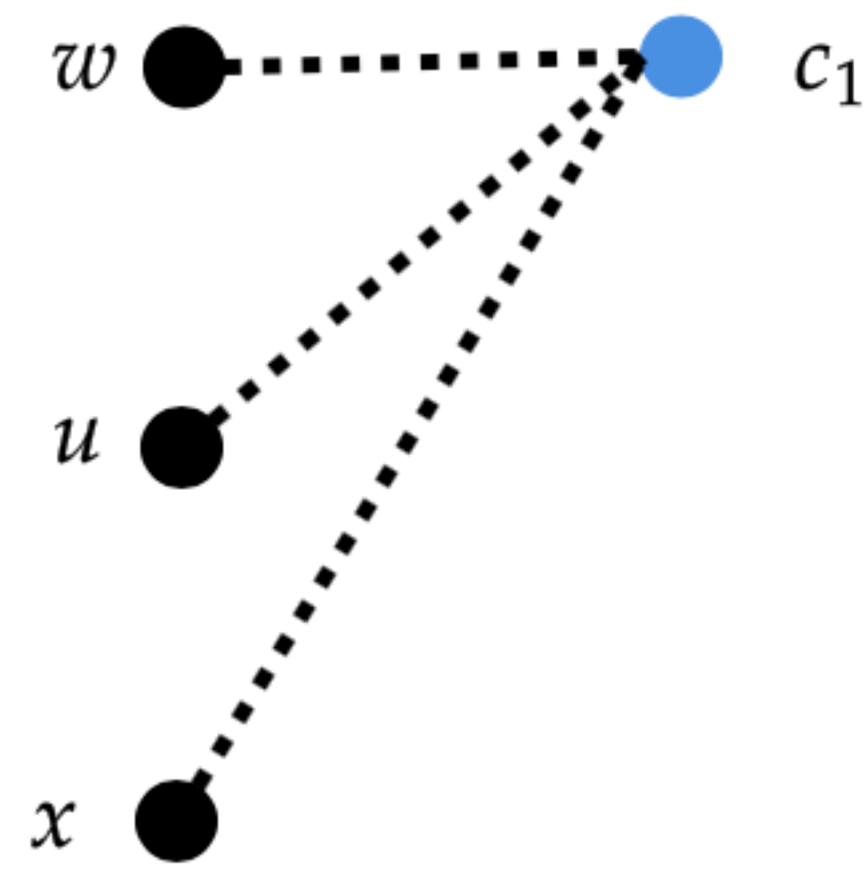


Online Model

- Typically, a bipartite graph G=(U,V,E). The set U is known to the algorithm. Vertices in V arrive one at a time, and reveal edges incident on them.

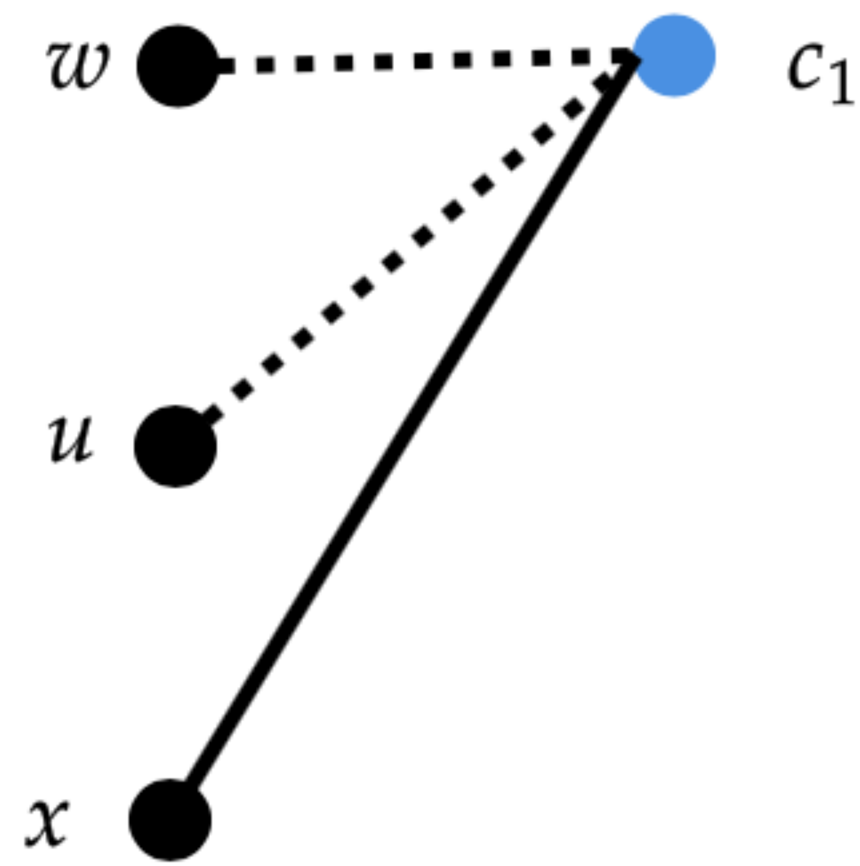
- The goal is to match (or forego) a vertex as soon as it arrives.

- The decisions made are irrevocable.



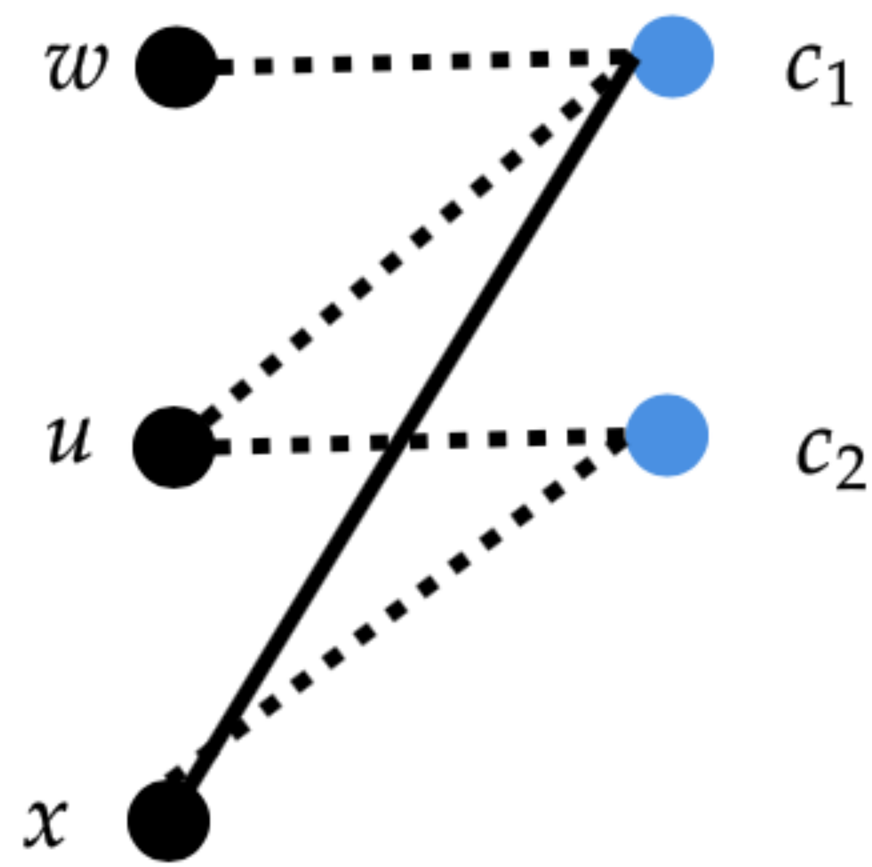
Online Model

- Typically, a bipartite graph G=(U,V,E). The set U is known to the algorithm. Vertices in V arrive one at a time, and reveal edges incident on them.
- The goal is to match (or forego) a vertex as soon as it arrives.
- The decisions made are irrevocable.



Online Model

- **Typically, a bipartite graph G=(U,V,E). The set U is known to the algorithm. Vertices in V arrive one at a time, and reveal edges incident on them.**
- **The goal is to match (or forego) a vertex as soon as it arrives.**
- **The decisions made are irrevocable.**

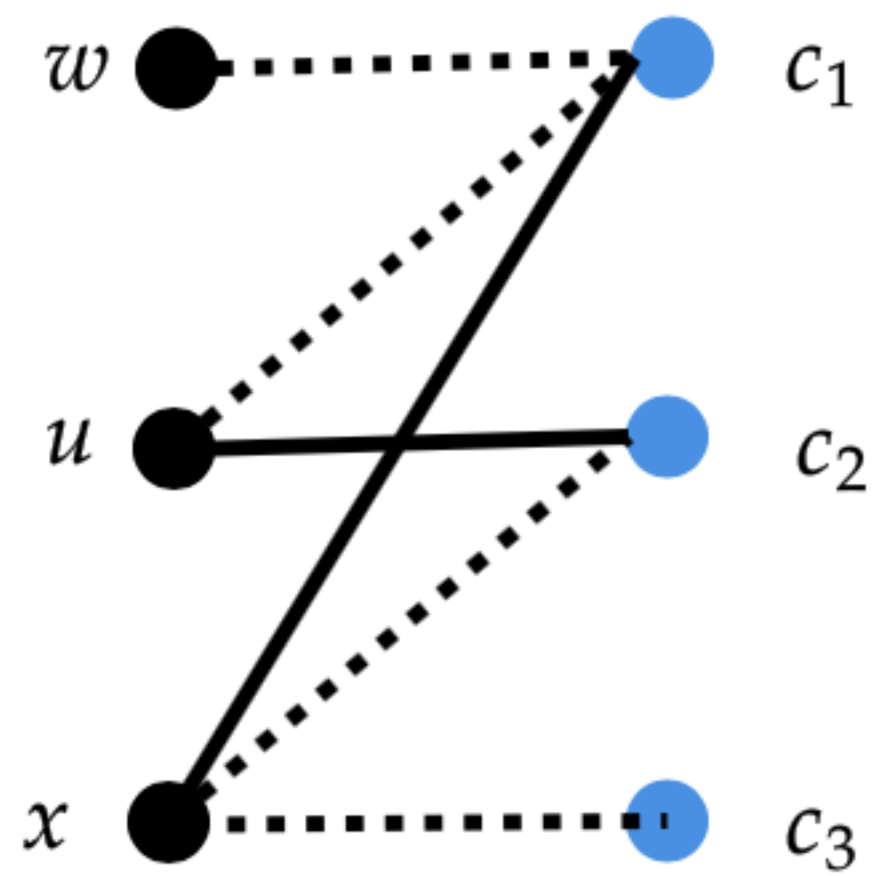


Online Model

- Typically, a bipartite graph G=(U,V,E). The set U is known to the algorithm. Vertices in V arrive one at a time, and reveal edges incident on them.

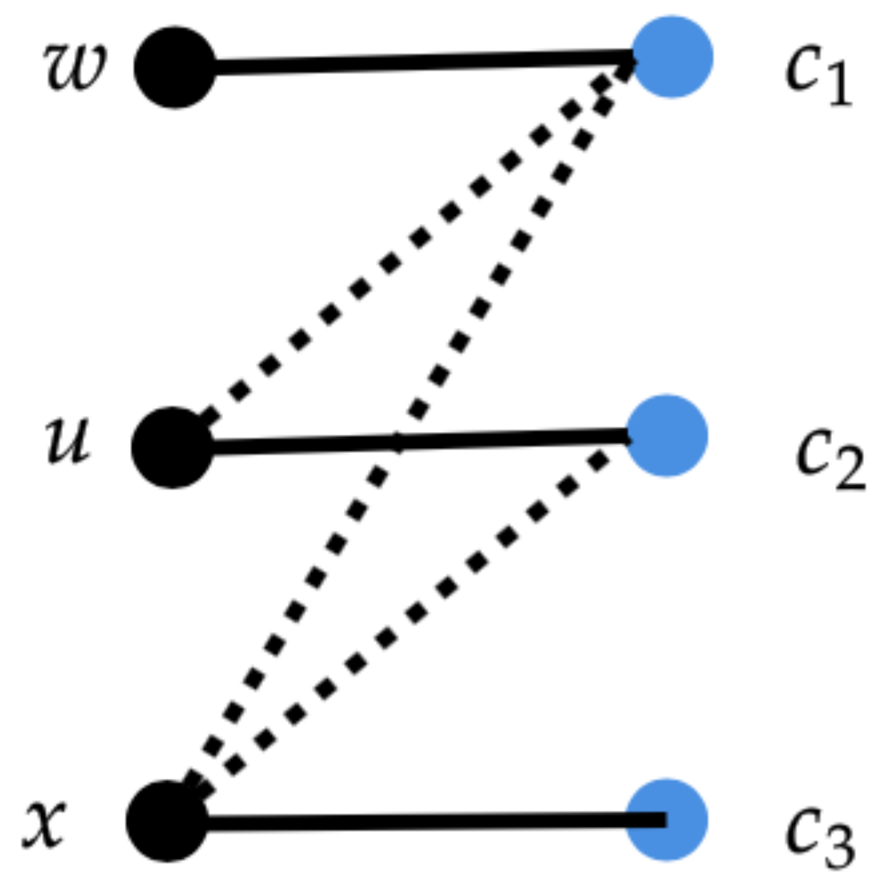
- The goal is to match (or forego) a vertex as soon as it arrives.

- The decisions made are irrevocable.



Online Model

- No **"take-backs"** implies that an exact solution cannot be guaranteed. The best we can do is a $\left(1 - \frac{1}{e}\right)$ - approximation (due to **Karp-Vazirani-Vazirani.**)



Revoking decisions

Revoking decisions

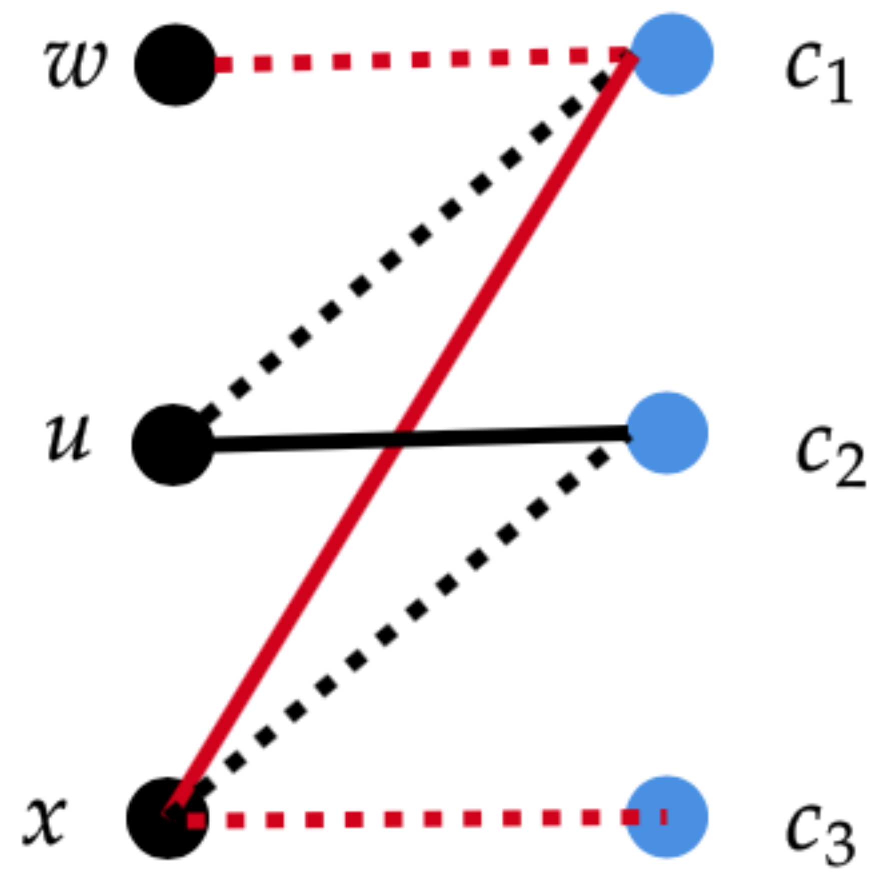
- Revoking decisions can increase the size of the matching.

Revoking decisions

- Revoking decisions can increase the size of the matching.
- If a larger matching is possible, then there is a way to change your decisions so that you get a larger matching.

Revoking decisions

- Revoking decisions can increase the size of the matching.
- If a larger matching is possible, then there is a way to change your decisions so that you get a larger matching.



Revoking decisions

- Revoking decisions can increase the size of the matching.
- If a larger matching is possible, then there is a way to change your decisions so that you get a larger matching.

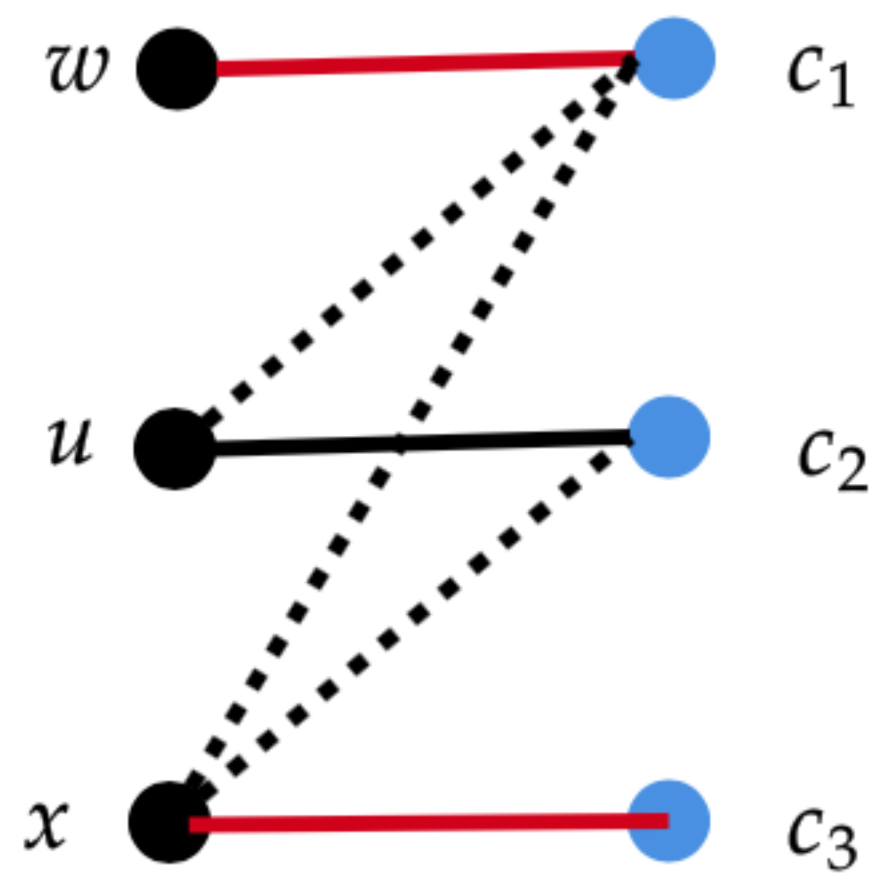
Revoking decisions

- Revoking decisions can increase the size of the matching.
- If a larger matching is possible, then there is a way to change your decisions so that you get a larger matching.
- These changes are made along an "augmenting path".

Revoking decisions

- Revoking decisions can increase the size of the matching.

- If a larger matching is possible, then there is a way to change your decisions so that you get a larger matching.
- These changes are made along an "augmenting path".



Is the condition realistic?

Is the condition realistic?

- **Is the “no take-backs” condition realistic?**

Is the condition realistic?

- Is the “no take-backs” condition realistic?
- In the case of Ad Allocation, once a slot has been assigned to an advertiser, it doesn’t make sense to reassign it.

Is the condition realistic?

- Is the "no take-backs" condition realistic?
- In the case of Ad Allocation, once a slot has been assigned to an advertiser, it doesn't make sense to reassign it.
- However, it makes sense to re-assign clients to another server in situations such as job scheduling.

Is the condition realistic?

- Is the "no take-backs" condition realistic?
- In the case of Ad Allocation, once a slot has been assigned to an advertiser, it doesn't make sense to reassign it.
- However, it makes sense to re-assign clients to another server in situations such as job scheduling.
- On the other hand, re-assigning might be costly, or may cause interruptions. So it makes sense to insist on minimizing changes.

Online Model With Recourse

Online Model With Recourse

- **In this model, the clients arrive one at a time along with their edges, and ask to be matched to a server.**

Online Model With Recourse

- **In this model, the clients arrive one at a time along with their edges, and ask to be matched to a server.**

- **The algorithm is allowed to change the matching over time, and is required to always maintain a maximum matching.**

Online Model With Recourse

- In this model, the clients arrive one at a time along with their edges, and ask to be matched to a server.
- The algorithm is allowed to change the matching over time, and is required to always maintain a maximum matching.
- The goal is to minimize the total number of changes made to the matching, denoted recourse.

Online Model With Recourse

Online Model With Recourse

This model is well-studied. Suppose   is the number of clients in the graph then,

Online Model With Recourse

This model is well-studied. Suppose n is the number of clients in the graph then,

Online Model With Recourse

This model is well-studied. Suppose n is the number of clients in the graph then,

- **The trivial bound is**

Online Model With Recourse

This model is well-studied. Suppose n is the number of clients in the graph then,

- **The trivial bound is $O(n^2)$.**

Online Model With Recourse

This model is well-studied. Suppose n is the number of clients in the graph then,

- **The trivial bound is $O(n^2)$.**

- **The best known result on bipartite graphs is due to Bernstein-Holm-Rotenberg which nearly matches the lower bound of due to Grove-Kao-Krishnan-Vitter.**

Online Model With Recourse

This model is well-studied. Suppose n is the number of clients in the graph then,

- **The trivial bound is $O(n^2)$.**
- **The best known result on bipartite graphs is $O(n \log^2 n)$ due to Bernstein-Holm-Rotenberg which nearly matches the lower bound of due to Grove-Kao-Krishnan-Vitter.**

Online Model With Recourse

This model is well-studied. Suppose n is the number of clients in the graph then,

- **The trivial bound is $O(n^2)$.**
- **The best known result on bipartite graphs is $O(n \log^2 n)$ due to Bernstein-Holm-Rotenberg which nearly matches the lower bound of $\Omega(n \log n)$ due to Grove-Kao-Krishnan-Vitter.**

Online Model With Recourse

This model is well-studied. Suppose n is the number of clients in the graph then,

- The trivial bound is $O(n^2)$.
- The best known result on bipartite graphs is $O(n \log^2 n)$ due to Bernstein-Holm-Rotenberg which nearly matches the lower bound of $\Omega(n \log n)$ due to Grove-Kao-Krishnan-Vitter.
- The best known upper bound on trees is which matches the lower bound (due to Bosek et al.)

Online Model With Recourse

This model is well-studied. Suppose n is the number of clients in the graph then,

- The trivial bound is $O(n^2)$.
- The best known result on bipartite graphs is $O(n \log^2 n)$ due to Bernstein-Holm-Rotenberg which nearly matches the lower bound of $\Omega(n \log n)$ due to Grove-Kao-Krishnan-Vitter.
- The best known upper bound on trees is $O(n \log n)$, which matches the lower bound (due to Bosek et al.)

Edge-Arrival Model

Edge-Arrival Model

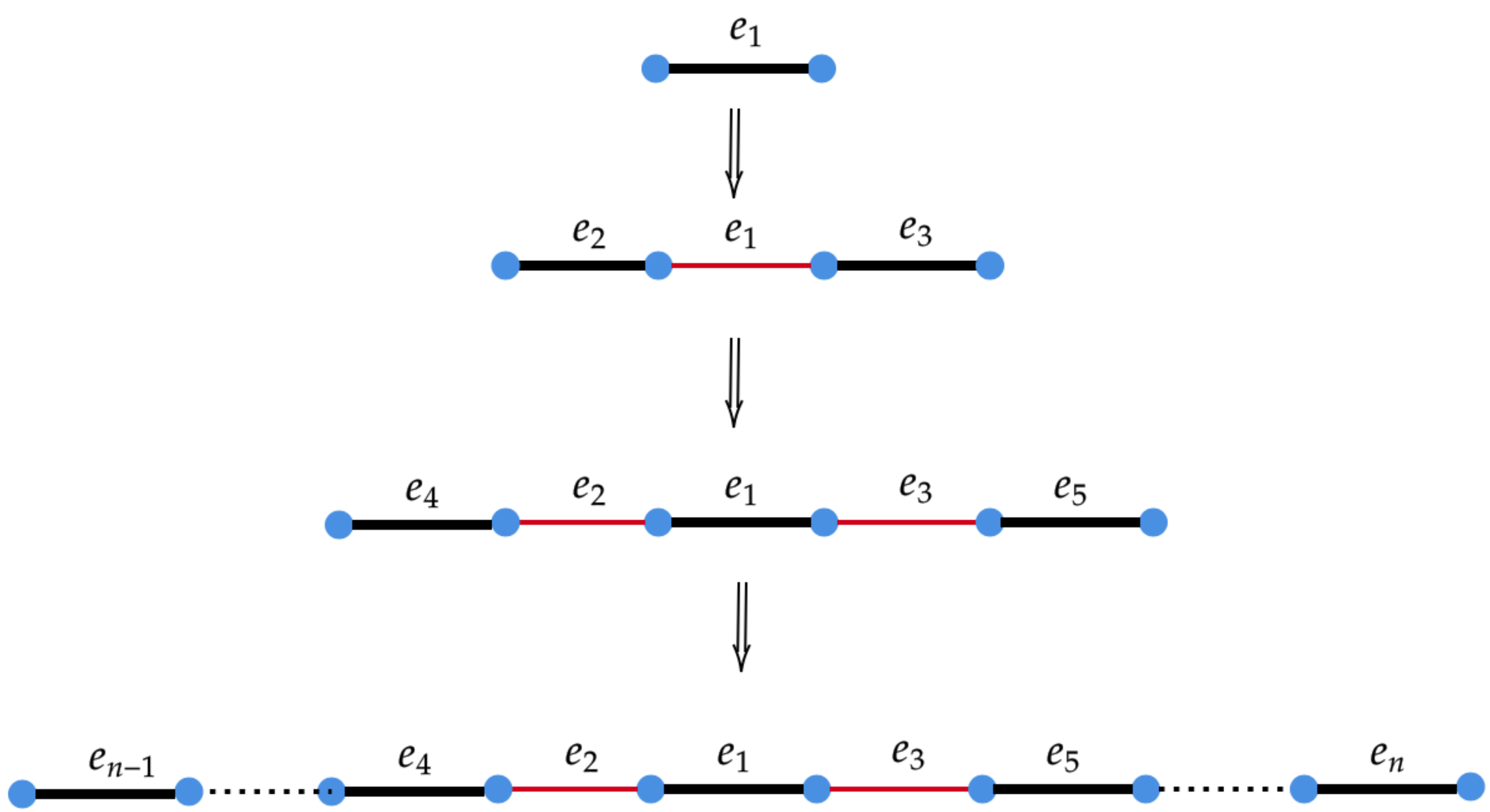
- **We consider a generalization of the "vertex-arrival" model. We allow the graph to be non-bipartite and the edges of the graph are revealed one at a time.**

Edge-Arrival Model

- **We consider a generalization of the “vertex-arrival” model. We allow the graph to be non-bipartite and the edges of the graph are revealed one at a time.**

- **However, in this model, strong-lower bounds are known for even simple graphs with adversarial ordering of the edges. As an example, the path graph.**

Edge Arrival: Adversarial Ordering



Total number of changes $= 3 + 5 + \cdots + n = \Theta(n^2)$

Random Arrival

Random Arrival

- To overcome the lower bound, we consider a natural relaxation of the problem where the adversary can choose the graph, but the edges of the graph arrive in a random order.

Random Arrival

- **To overcome the lower bound, we consider a natural relaxation of the problem where the adversary can choose the graph, but the edges of the graph arrive in a random order.**

- **In practical situations, it is unlikely that we land in a doubly worst-case situation — a worst case graph as well as a worst case ordering of the edges.**

Our Results

Our Results

- We show that for the case of general graphs (non-bipartite) that random arrival doesn't help. We get a lower bound of   on expected recourse.

Our Results

- **We show that for the case of general graphs (non-bipartite) that random arrival doesn't help. We get a lower bound of $\Omega\left(n^2/\log n\right)$ on expected recourse.**

Our Results

- **We show that for the case of general graphs (non-bipartite) that random arrival doesn't help. We get a lower bound of** $\Omega\left(n^2/\log n\right)$ **on expected recourse.**

- **For trees, we get an upper bound of   on expected recourse**

Our Results

- **We show that for the case of general graphs (non-bipartite) that random arrival doesn't help. We get a lower bound of $\Omega\left(n^2/\log n\right)$ on expected recourse.**

- **For trees, we get an upper bound of $O(n\log^2 n)$ on expected recourse**

Our Results

- **We show that for the case of general graphs (non-bipartite) that random arrival doesn't help. We get a lower bound of $\Omega\left(n^2/\log n\right)$ on expected recourse.**

- **For trees, we get an upper bound of $O(n\log^2 n)$ on expected recourse**

- **For paths, we get an upper bound of on expected recourse.**

Our Results

- **We show that for the case of general graphs (non-bipartite) that random arrival doesn't help. We get a lower bound of $\Omega\left(n^2/\log n\right)$ on expected recourse.**

- **For trees, we get an upper bound of $O(n\log^2 n)$ on expected recourse**

- **For paths, we get an upper bound of $O(n\log n)$ on expected recourse.**

Our Results

- We show that for the case of **general graphs** (non-bipartite) that random arrival doesn't help. We get a lower bound of $\Omega\left(n^2/\log n\right)$ on expected recourse.
- For **trees**, we get an upper bound of $O(n \log^2 n)$ on expected recourse
- For **paths**, we get an upper bound of $O(n \log n)$ on expected recourse.
- For **trees**, we also have a lower bound of on expected recourse.

Our Results

- **We show that for the case of general graphs (non-bipartite) that random arrival doesn't help. We get a lower bound of $\Omega\left(n^2/\log n\right)$ on expected recourse.**

- **For trees, we get an upper bound of $O(n\log^2 n)$ on expected recourse**

- **For paths, we get an upper bound of $O(n\log n)$ on expected recourse.**

- **For trees, we also have a lower bound of $\Omega(n\log n)$ on expected recourse.**

Our Results

- We show that for the case of general graphs (non-bipartite) that random arrival doesn't help. We get a lower bound of $\Omega\left(n^2/\log n\right)$ on expected recourse.

- For trees, we get an upper bound of $O(n\log^2 n)$ on expected recourse

- For paths, we get an upper bound of $O(n\log n)$ on expected recourse.

- For trees, we also have a lower bound of $\Omega(n\log n)$ on expected recourse.

Our Results

- **We show that for the case of general graphs (non-bipartite) that random arrival doesn't help. We get a lower bound of $\Omega\left(n^2/\log n\right)$ on expected recourse.**

- **For trees, we get an upper bound of $O(n \log^2 n)$ on expected recourse**

- **For paths, we get an upper bound of $O(n \log n)$ on expected recourse.**

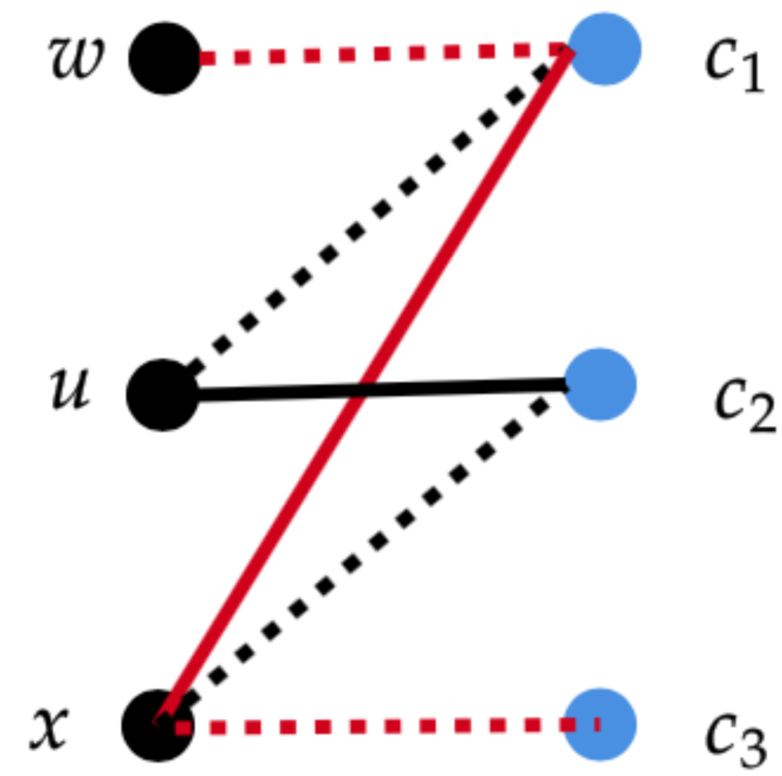
- **For trees, we also have a lower bound of $\Omega(n \log n)$ on expected recourse.**

Augmenting Paths: An augmenting path is a path with alternating matched and unmatched edges that ends in free vertices.

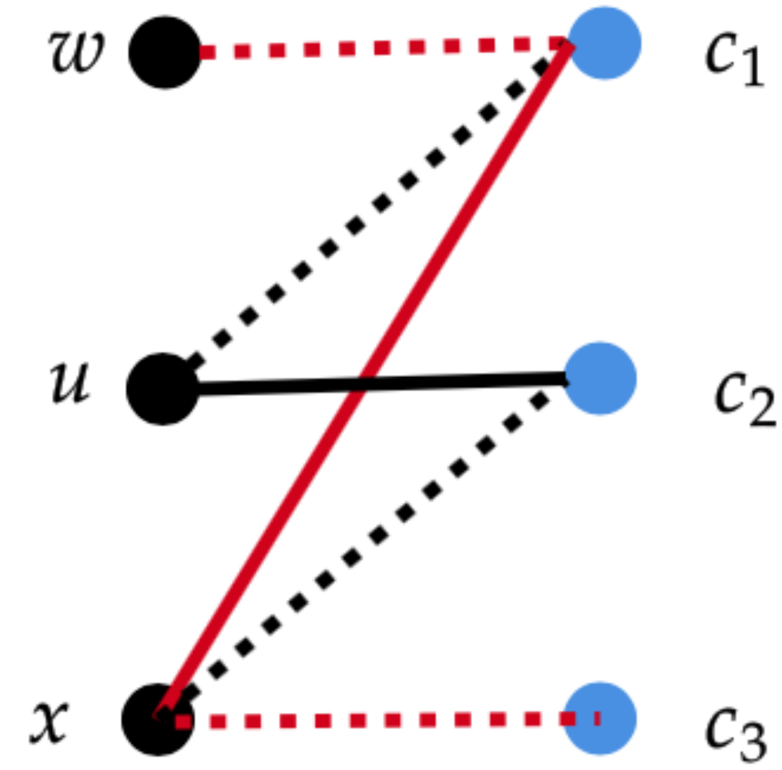
Interchanging matched and unmatched edges along an augmenting path increases the size of a matching.

Augmenting Paths: An augmenting path is a path with alternating matched and unmatched edges that ends in free vertices.

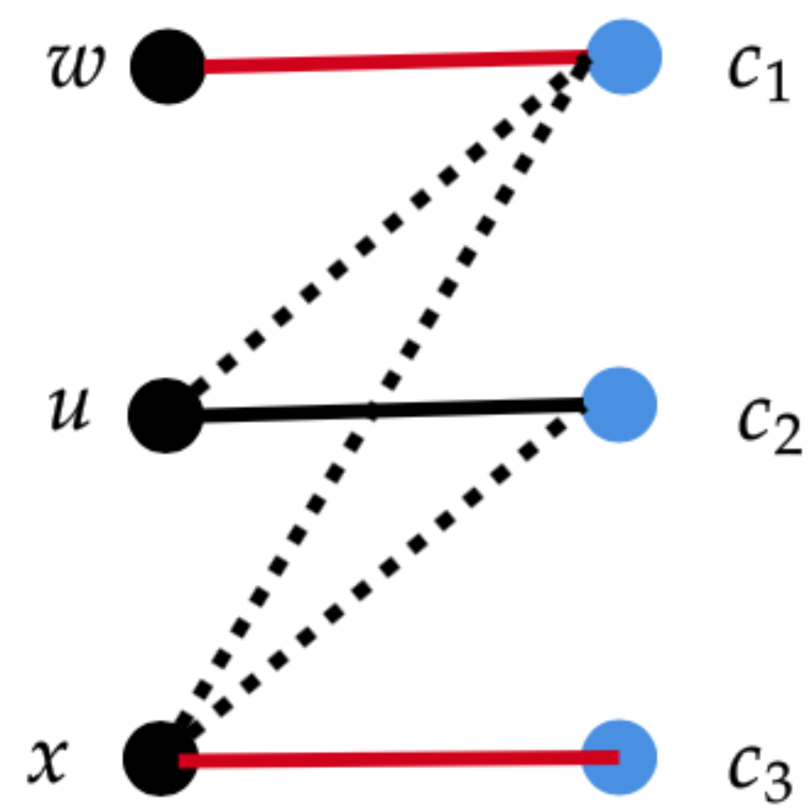
Interchanging matched and unmatched edges along an augmenting path increases the size of a matching.

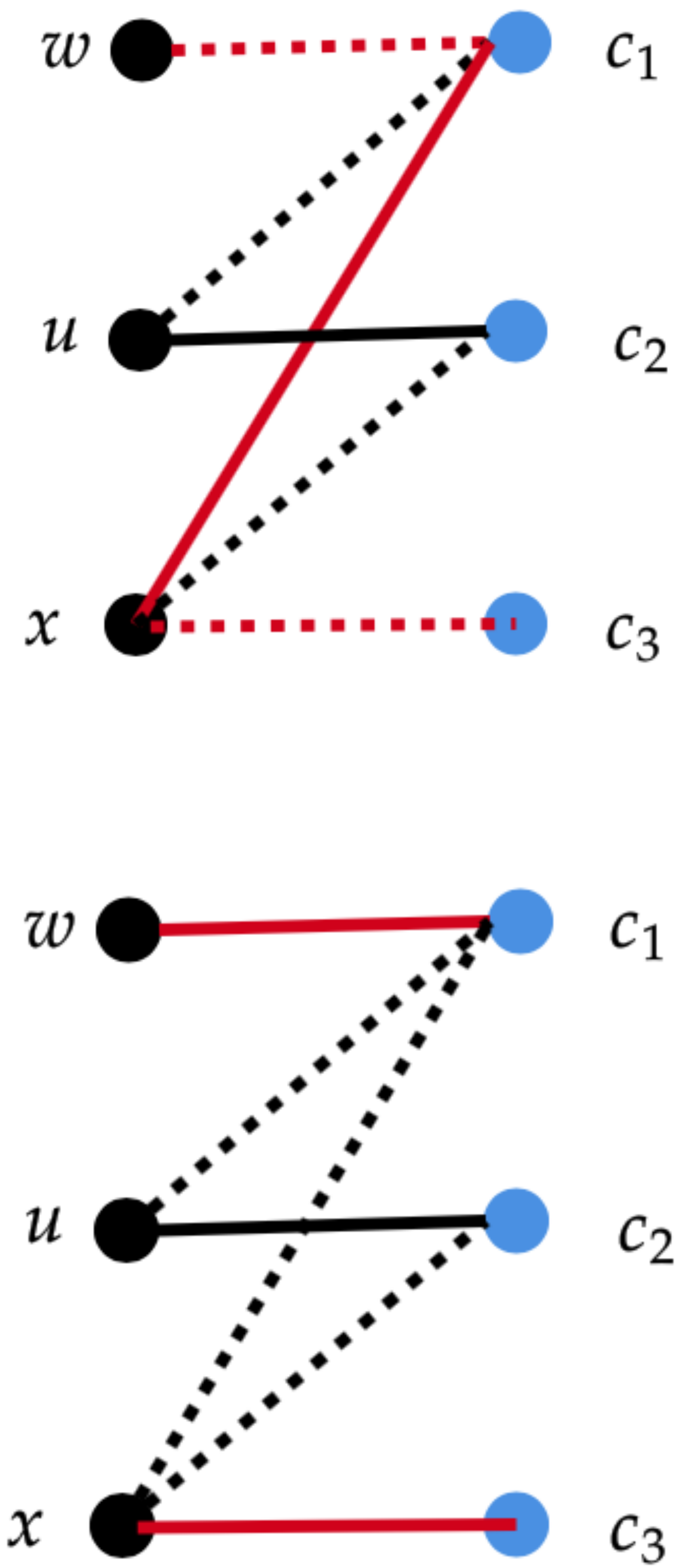


Augmenting Paths: **An augmenting path is a path with alternating matched and unmatched edges that ends in free vertices.**

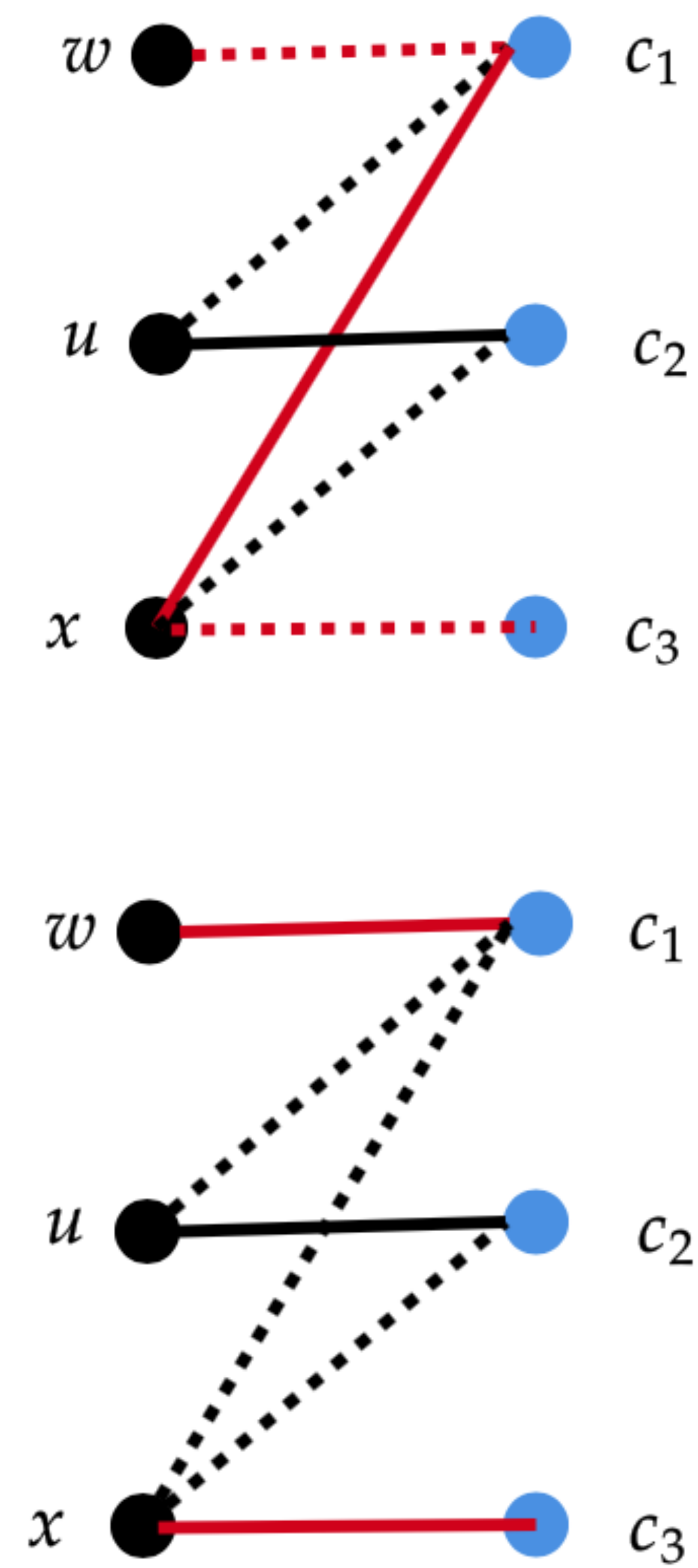


Interchanging matched and unmatched edges along an augmenting path increases the size of a matching.




w
u
x
c_1
c_2
c_3
w
u
x
c_1
c_2
c_3

Remark: Total recourse taken by the algorithm corresponds exactly to the total length of the augmenting paths taken by the algorithm. We will upper and lower bound the length of augmenting paths which will give us a bound on the total recourse as well.

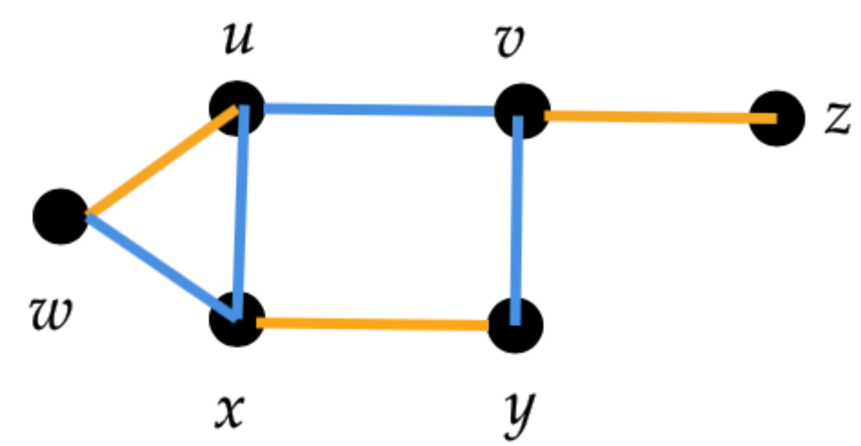


Perfect Matching: In case of graphs with an even number of vertices, it is a matching that matches all vertices in the graph.

Near-Perfect Matching: In case of graphs with an odd number of vertices, it is a matching that matches all vertices but one in the graph

Perfect Matching: In case of graphs with an even number of vertices, it is a matching that matches all vertices in the graph.

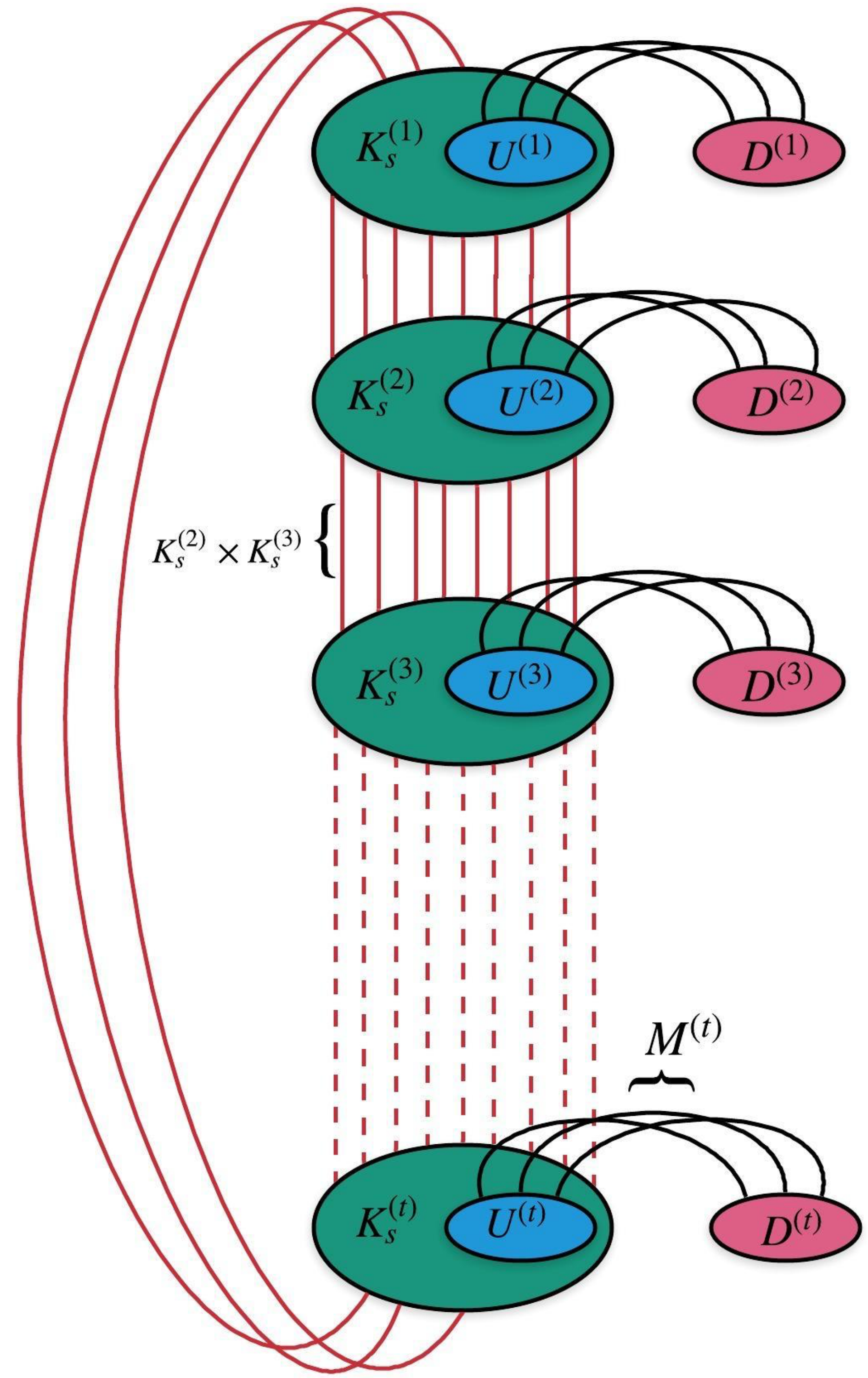
Near-Perfect Matching: In case of graphs with an odd number of vertices, it is a matching that matches all vertices but one in the graph



Maximum matching

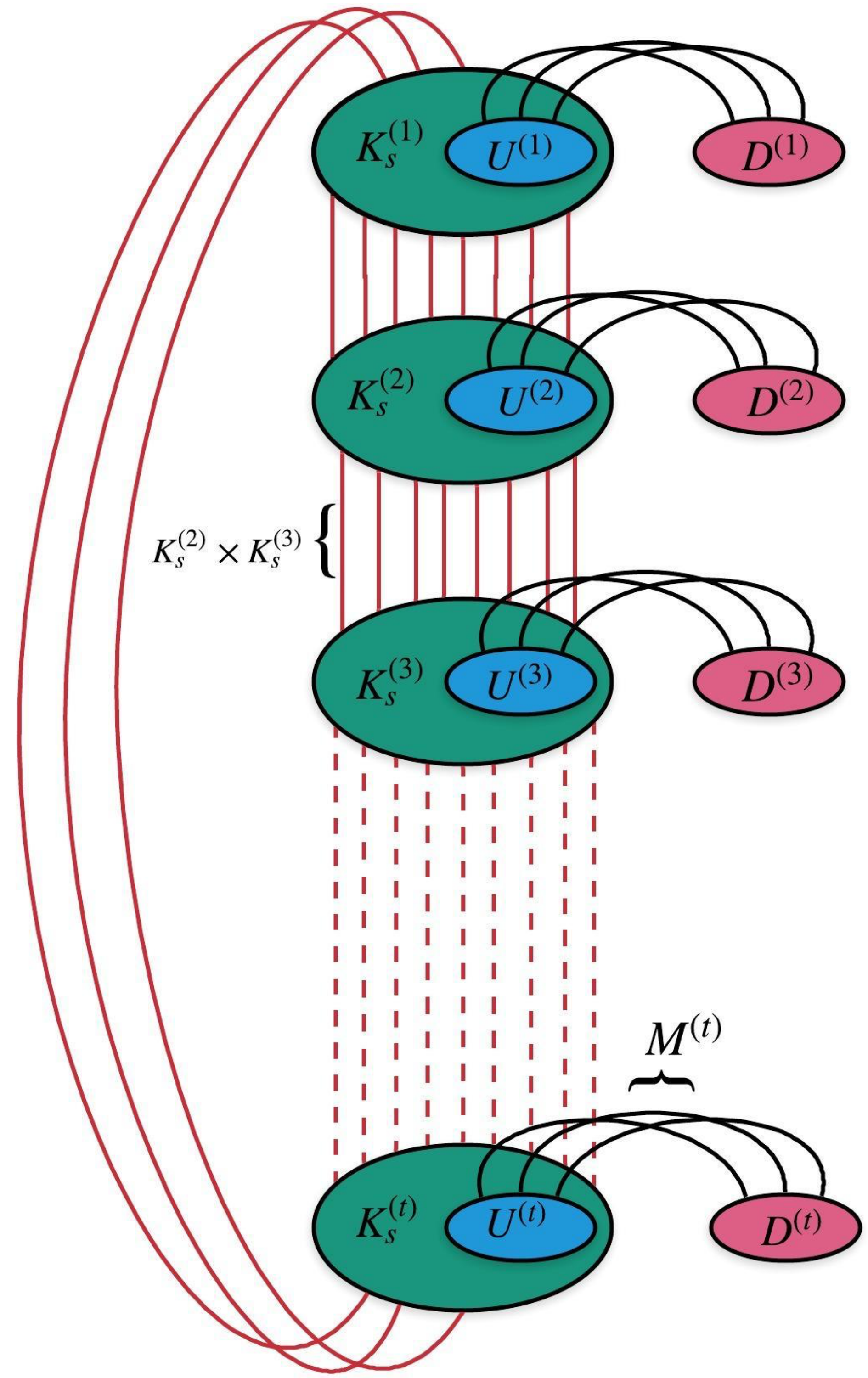
Lower Bound Graph

Lower Bound Graph



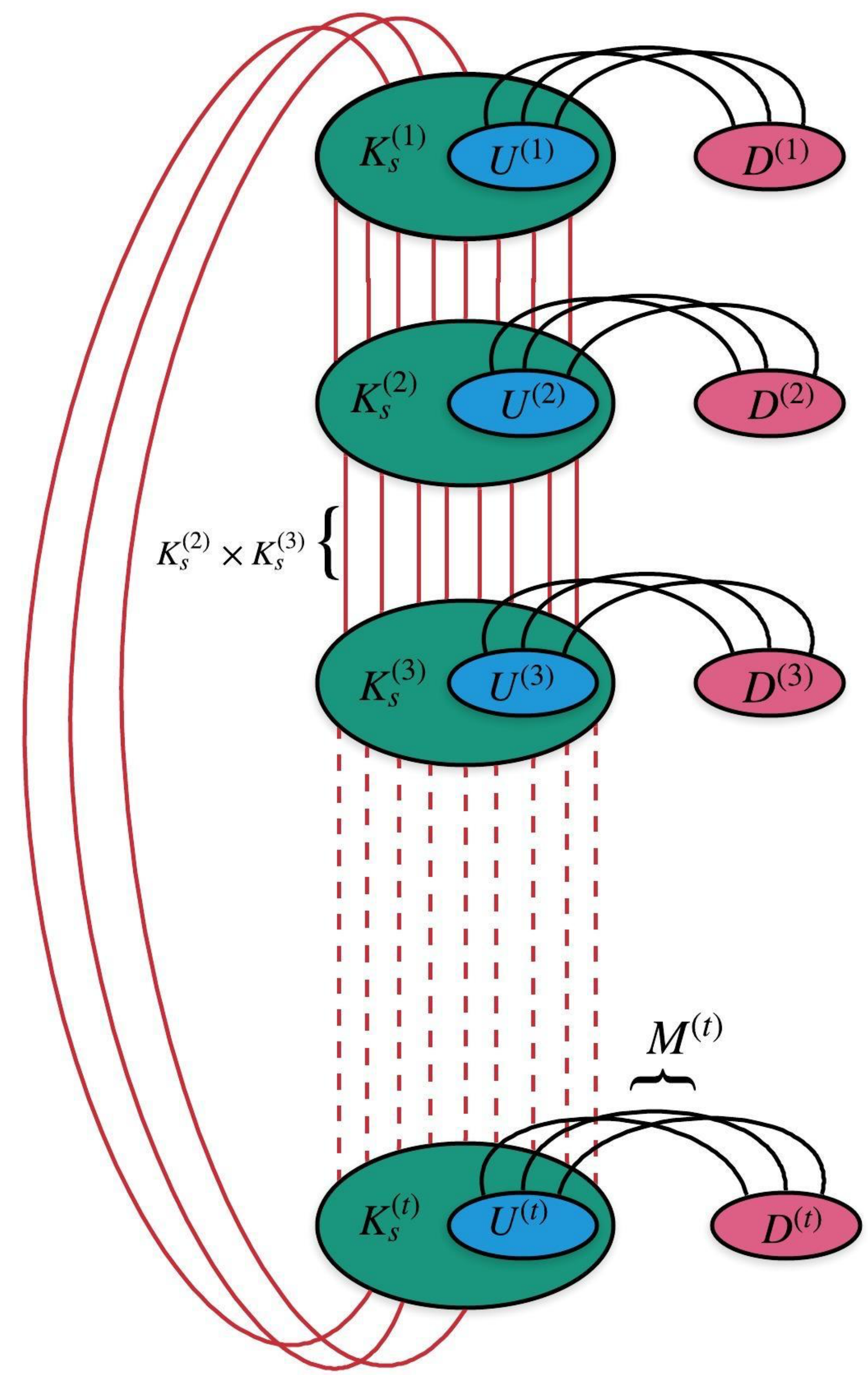
Lower Bound Graph

- We have copies of the complete graph on vertices, where



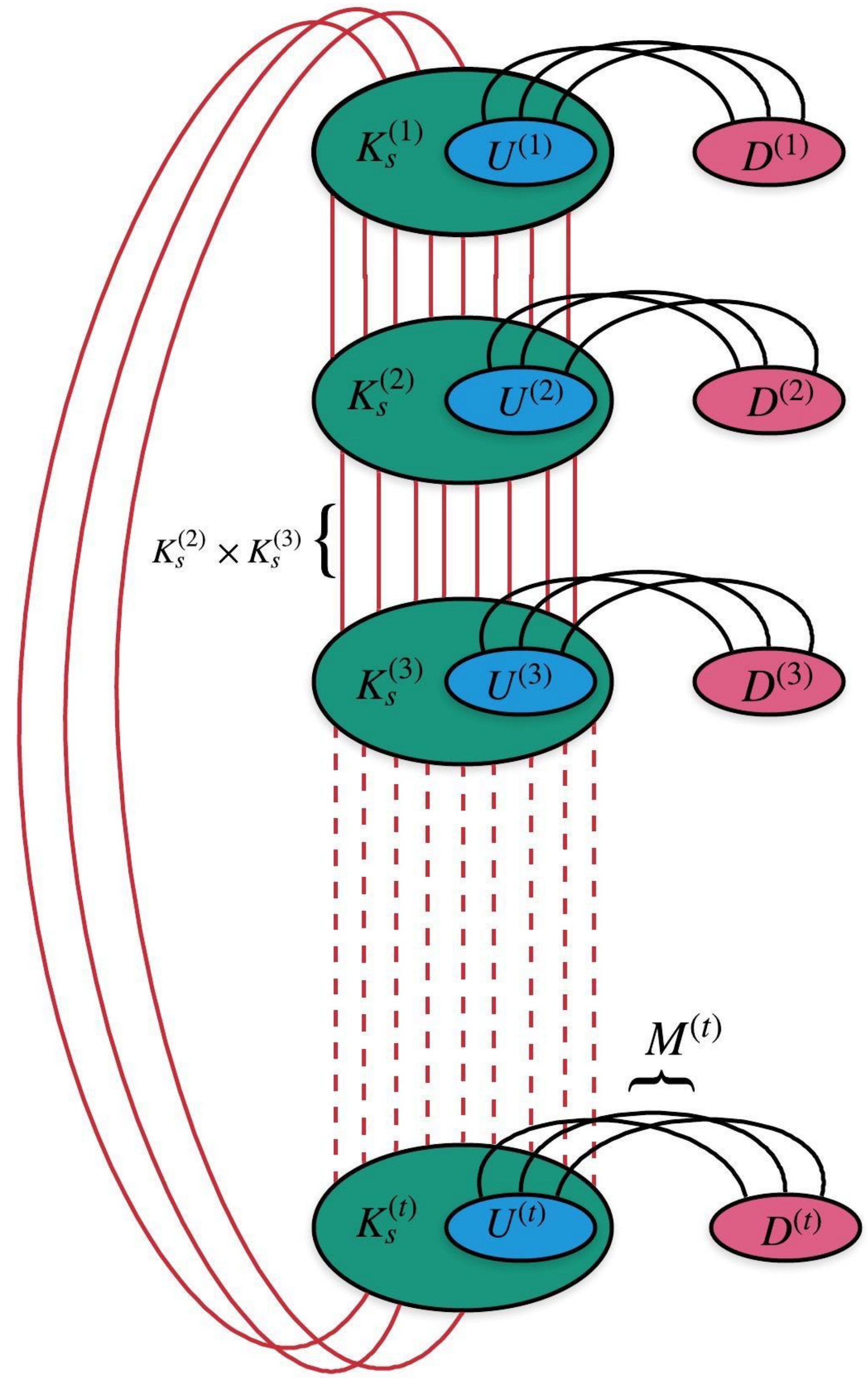
Lower Bound Graph

- **We have** $t = \dfrac{n}{500 \log n}$ **copies of the complete graph on vertices, where**



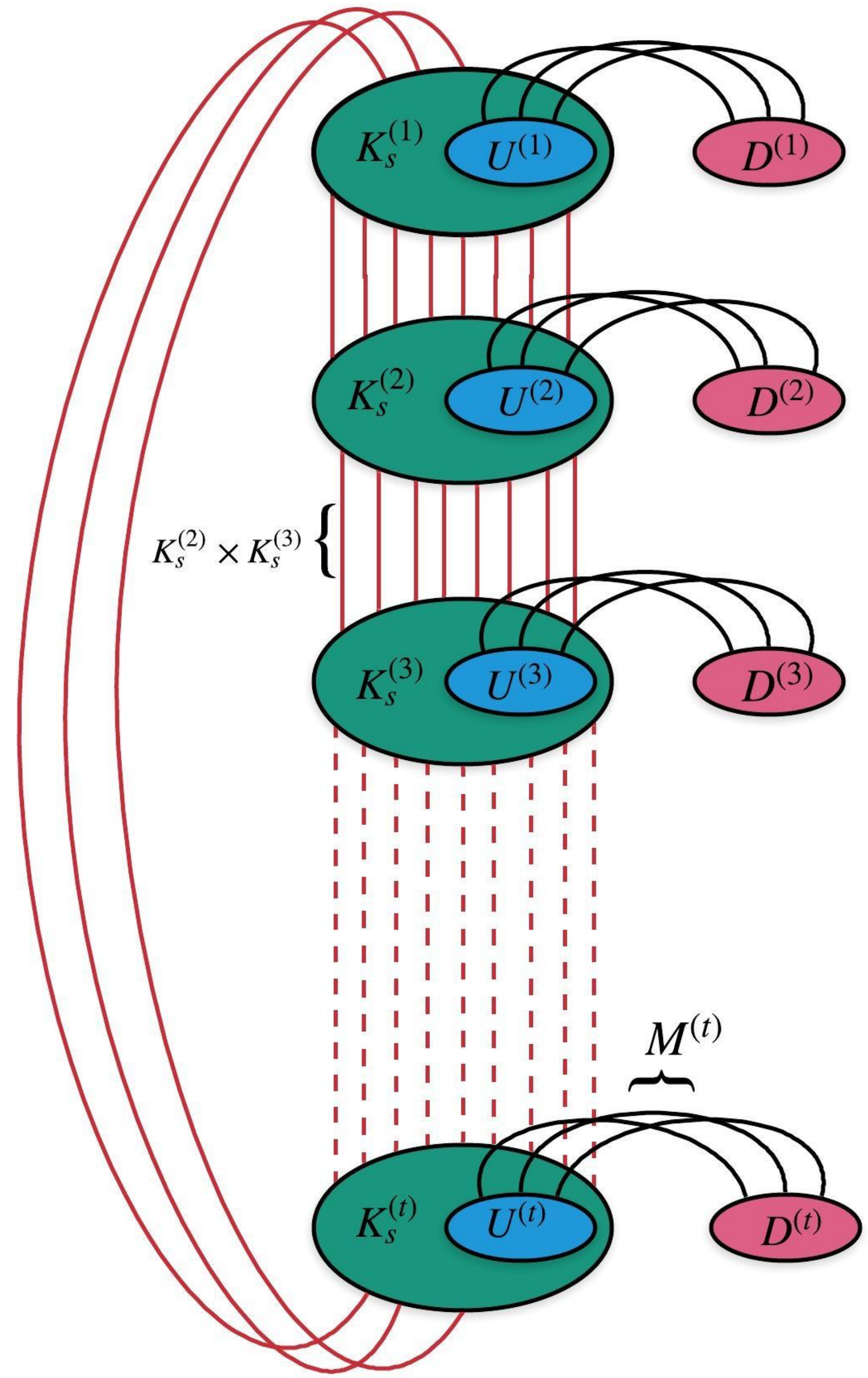
Lower Bound Graph

- **We have $t = \dfrac{n}{500 \log n}$ copies of the K_s, complete graph on vertices, where**



Lower Bound Graph

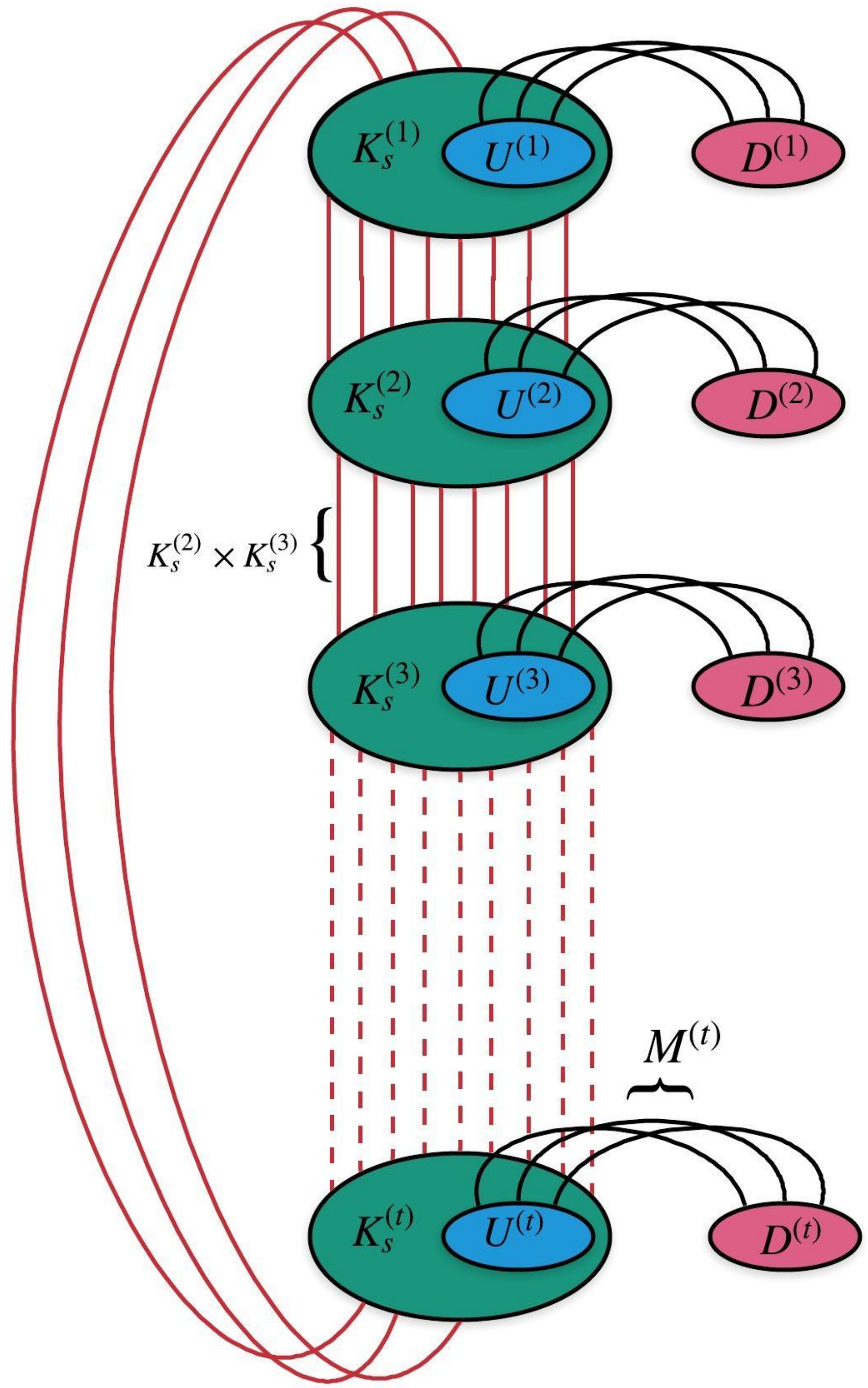
- We have $t = \dfrac{n}{500 \log n}$ copies of the K_s, complete graph on vertices, where $s = 400 \log n$.



Lower Bound Graph

- We have $t = \frac{n}{500 \log n}$ copies of the K_s, complete graph on vertices, where $s = 400 \log n$.

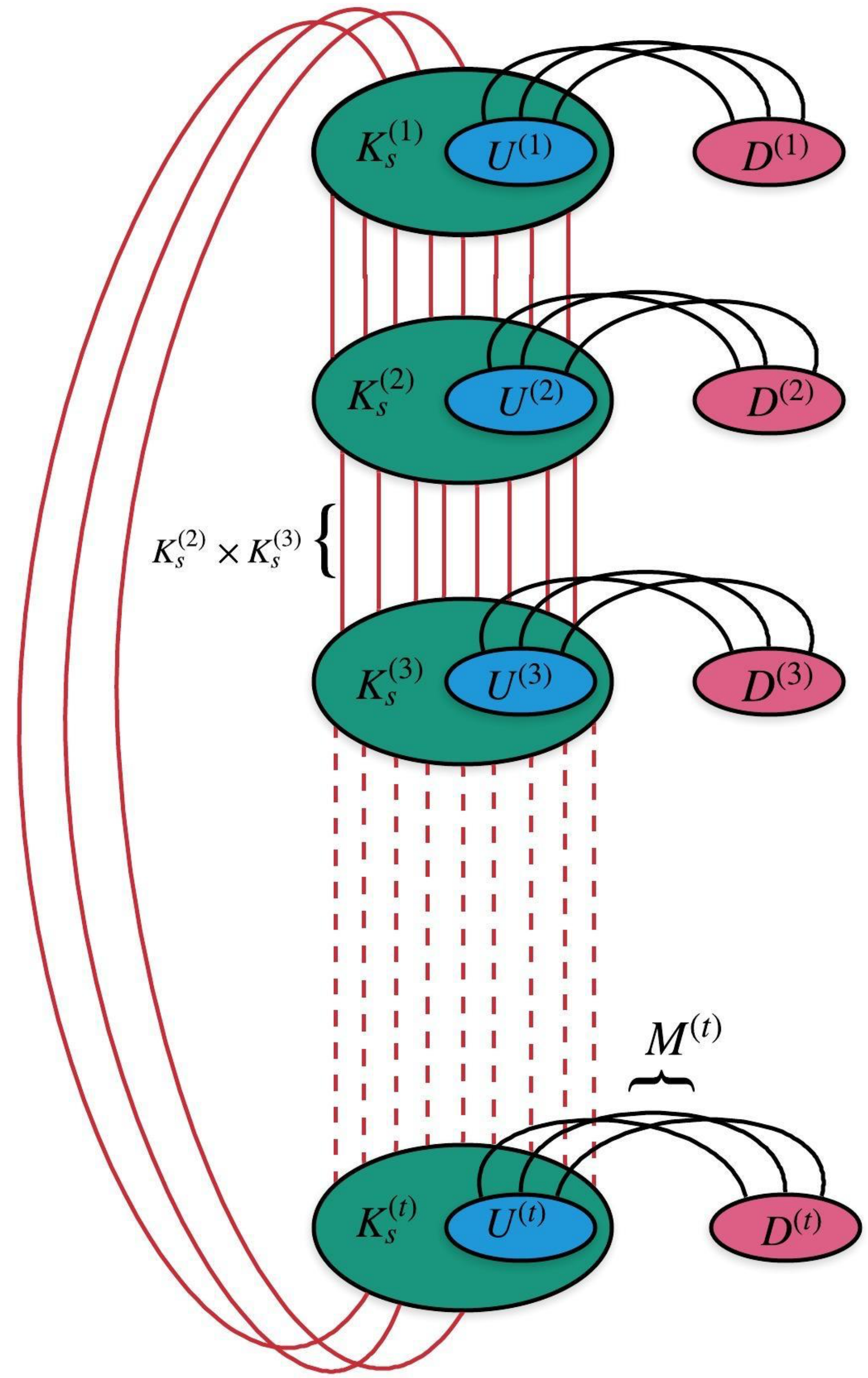
- These ’s are arranged in a ring, with edges between each pair of vertices of consecutive ’s.



Lower Bound Graph

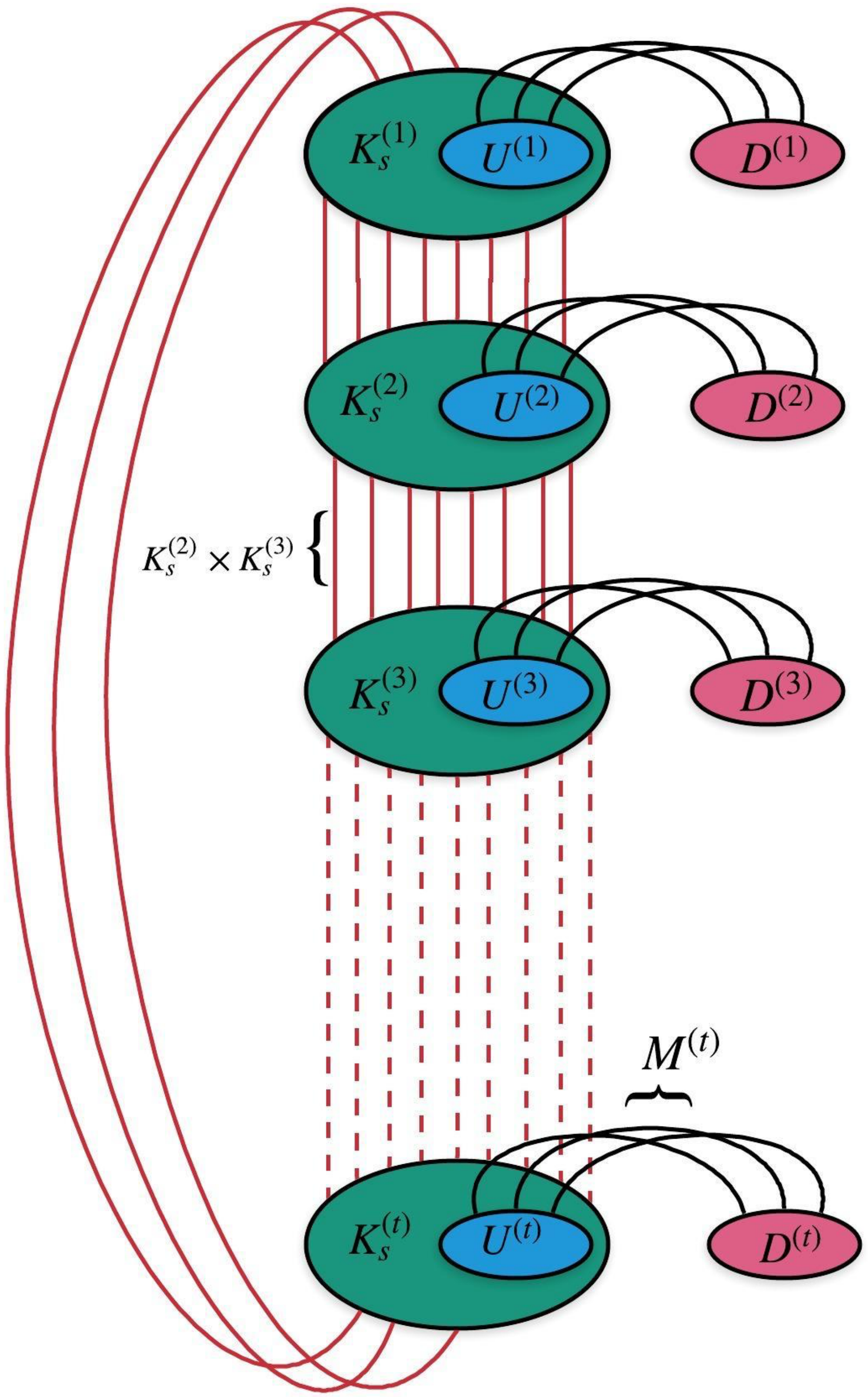
- We have $t = \dfrac{n}{500 \log n}$ copies of the K_s, complete graph on vertices, where $s = 400 \log n$.

- These K_s's are arranged in a ring, with edges between each pair of vertices of consecutive 's.



Lower Bound Graph

- **We have $t = \dfrac{n}{500 \log n}$ copies of the K_s, complete graph on vertices, where $s = 400 \log n$.**
- **These K_s's are arranged in a ring, with edges between each pair of vertices of consecutive K_s's.**

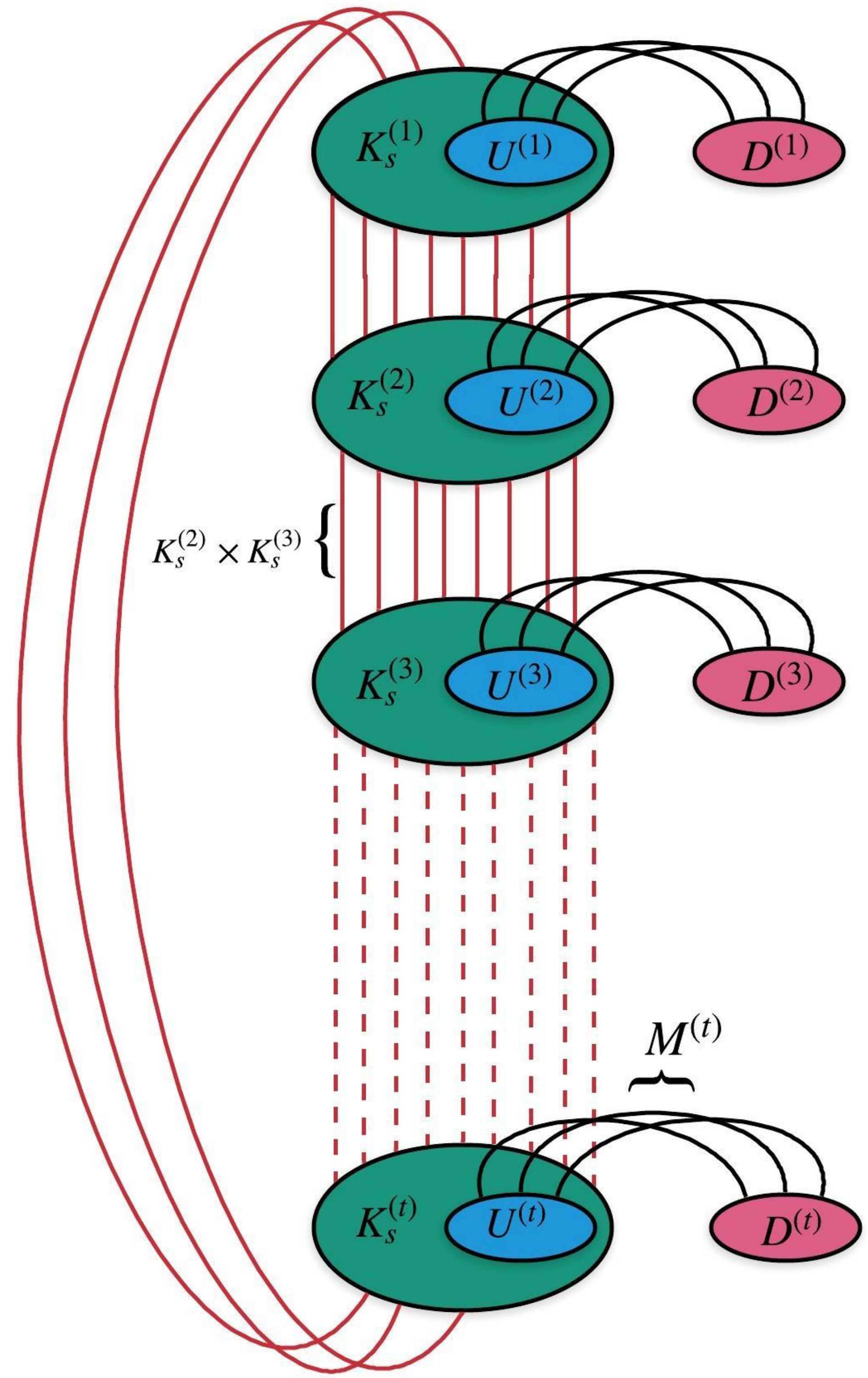


Lower Bound Graph

- We have $t = \dfrac{n}{500 \log n}$ copies of the K_s, complete graph on vertices, where $s = 400 \log n$.

- These K_s's are arranged in a ring, with edges between each pair of vertices of consecutive K_s's.

- In each copy of   we have a set of   vertices, called Each of these   's are matched to   We call these "dangling" edges.

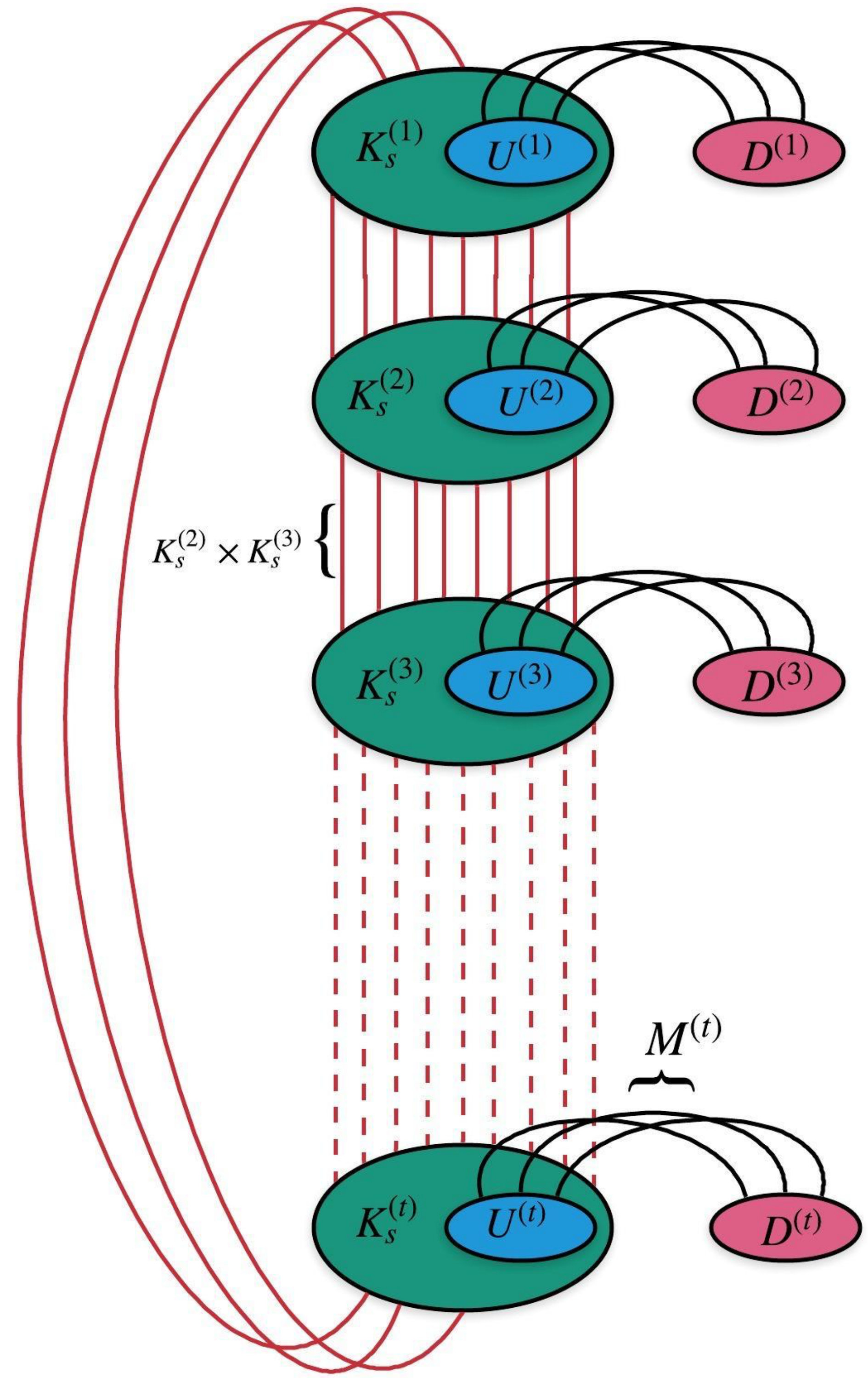


Lower Bound Graph

- **We have $t = \dfrac{n}{500\log n}$ copies of the K_s, complete graph on vertices, where $s = 400\log n$.**

- **These K_s's are arranged in a ring, with edges between each pair of vertices of consecutive K_s's.**

- **In each copy of K_s, we have a set of vertices, called Each of these 's are matched to We call these "dangling" edges.**

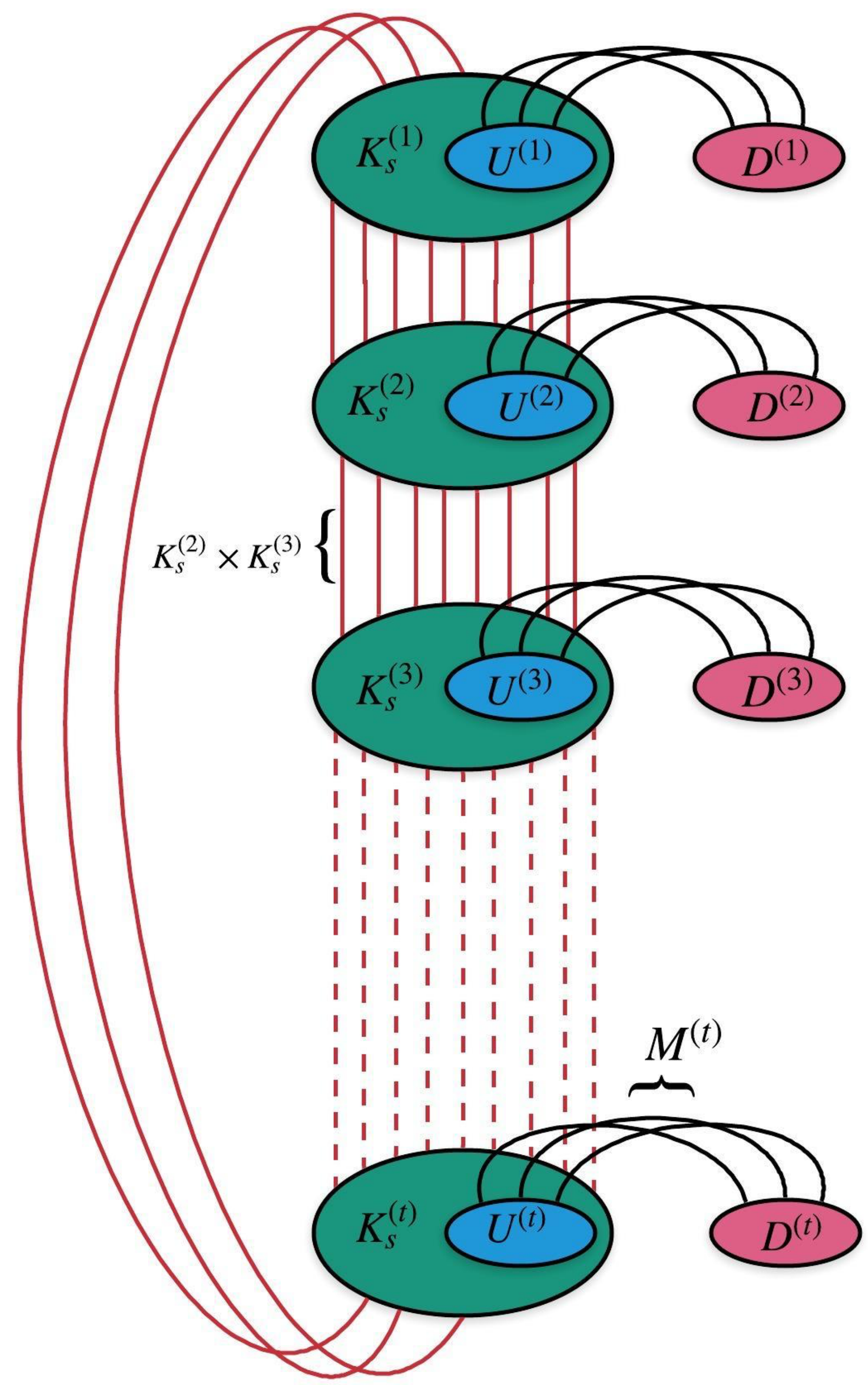


Lower Bound Graph

- We have $t = \dfrac{n}{500 \log n}$ copies of the K_s, complete graph on vertices, where $s = 400 \log n$.

- These K_s’s are arranged in a ring, with edges between each pair of vertices of consecutive K_s’s.

- In each copy of K_s, we have a set of $100 \log n$ vertices, called Each of these ’s are matched to We call these “dangling” edges.

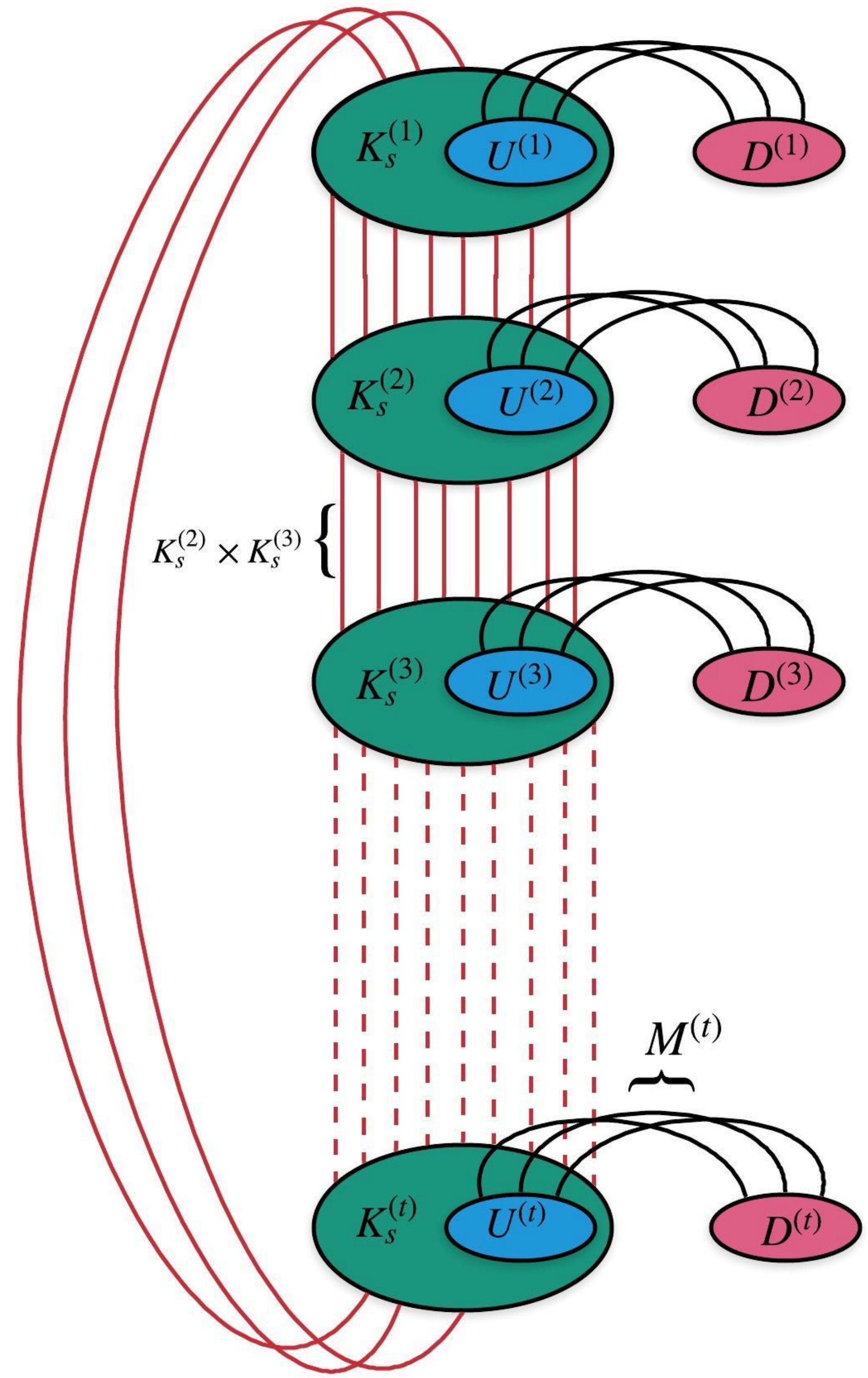


Lower Bound Graph

- We have $t = \dfrac{n}{500 \log n}$ copies of the K_s, complete graph on vertices, where $s = 400 \log n$.

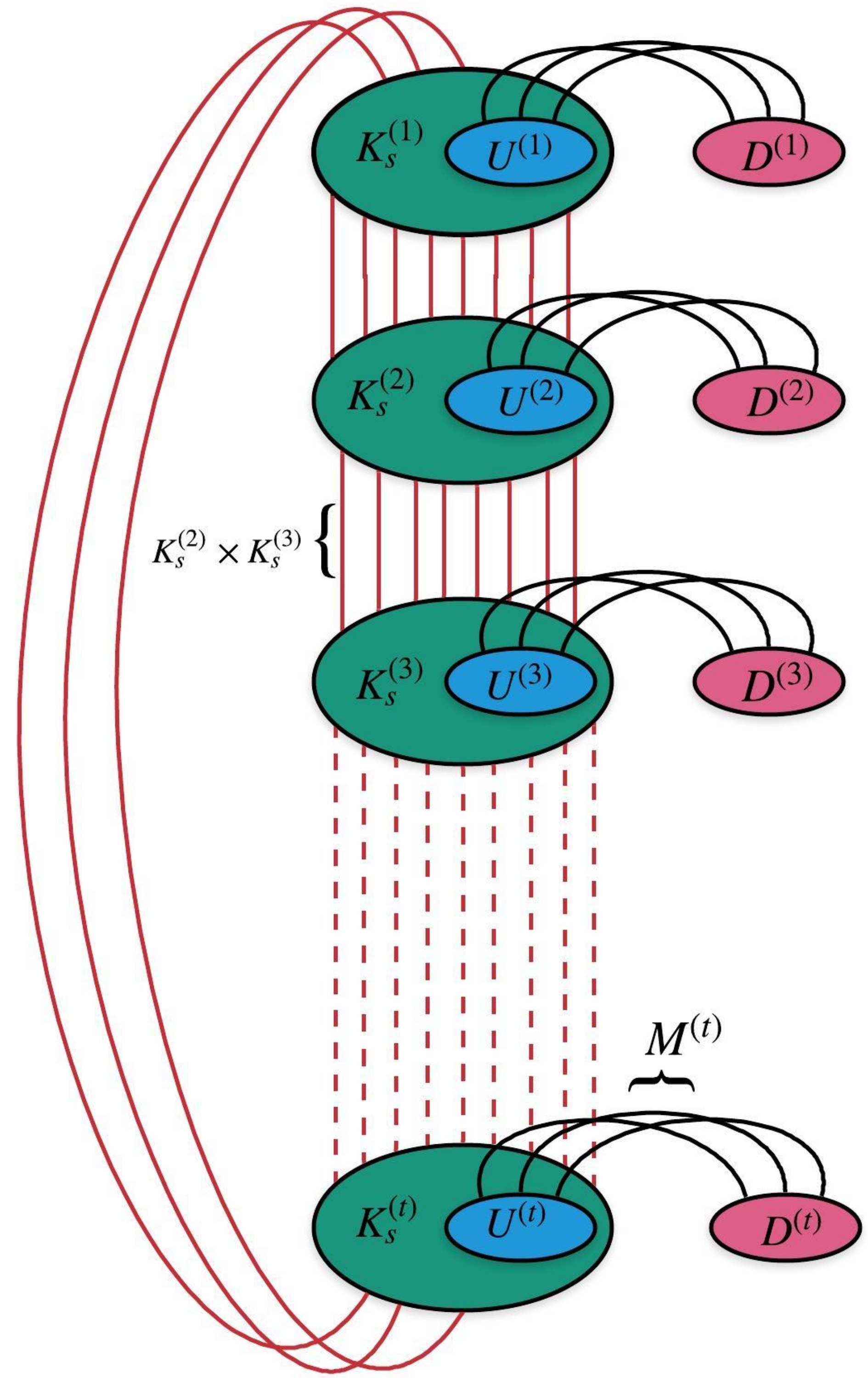
- These K_s's are arranged in a ring, with edges between each pair of vertices of consecutive K_s's.

- In each copy of K_s, we have a set of $100 \log n$ vertices, called $U^{(i)}$. Each of these ’s are matched to We call these “dangling” edges.



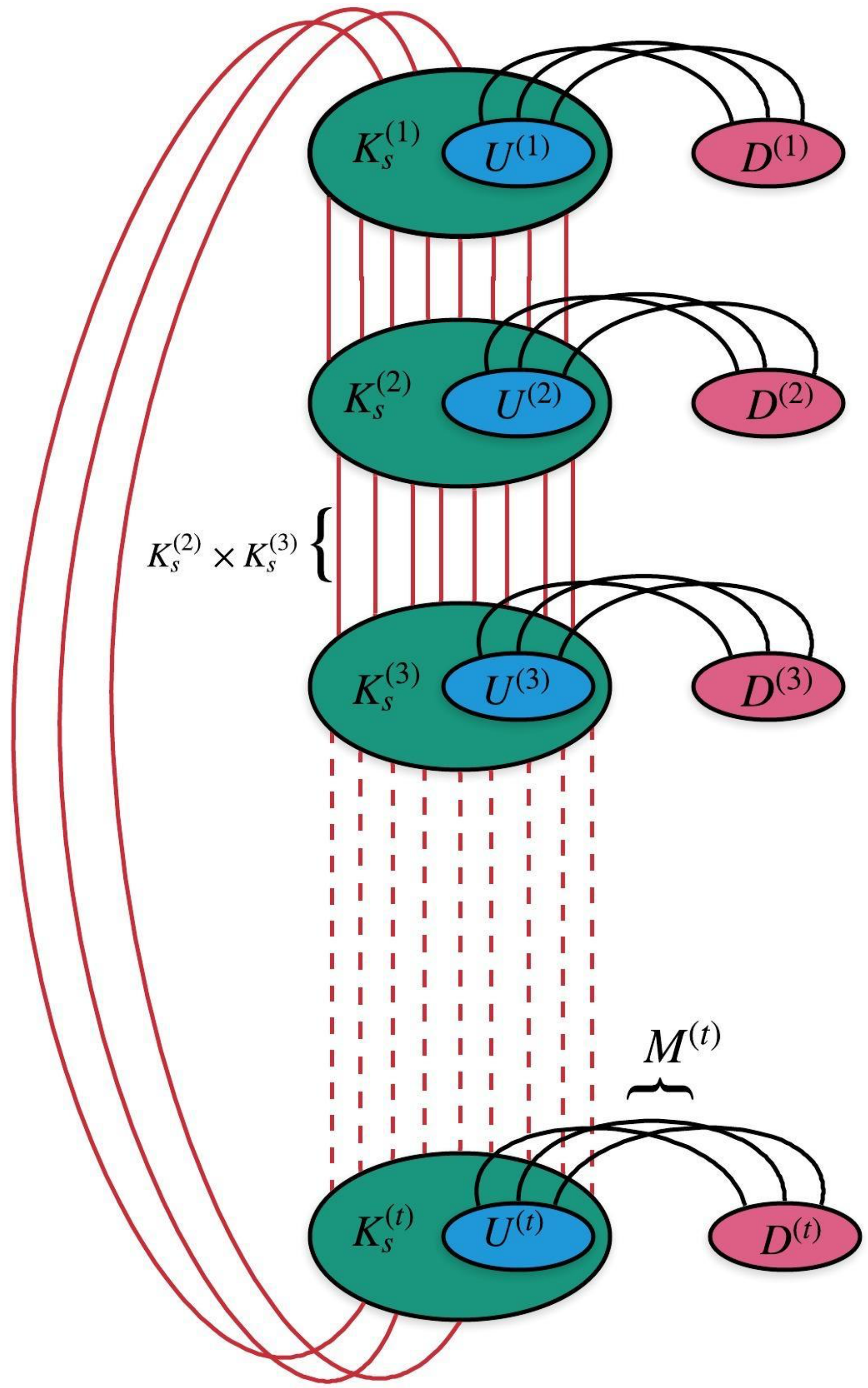
Lower Bound Graph

- We have $t = \dfrac{n}{500 \log n}$ copies of the K_s, complete graph on vertices, where $s = 400 \log n$.
- These K_s's are arranged in a ring, with edges between each pair of vertices of consecutive K_s's.
- In each copy of K_s, we have a set of $100 \log n$ vertices, called $U^{(i)}$. Each of these $U^{(i)}$'s are matched to We call these "dangling" edges.



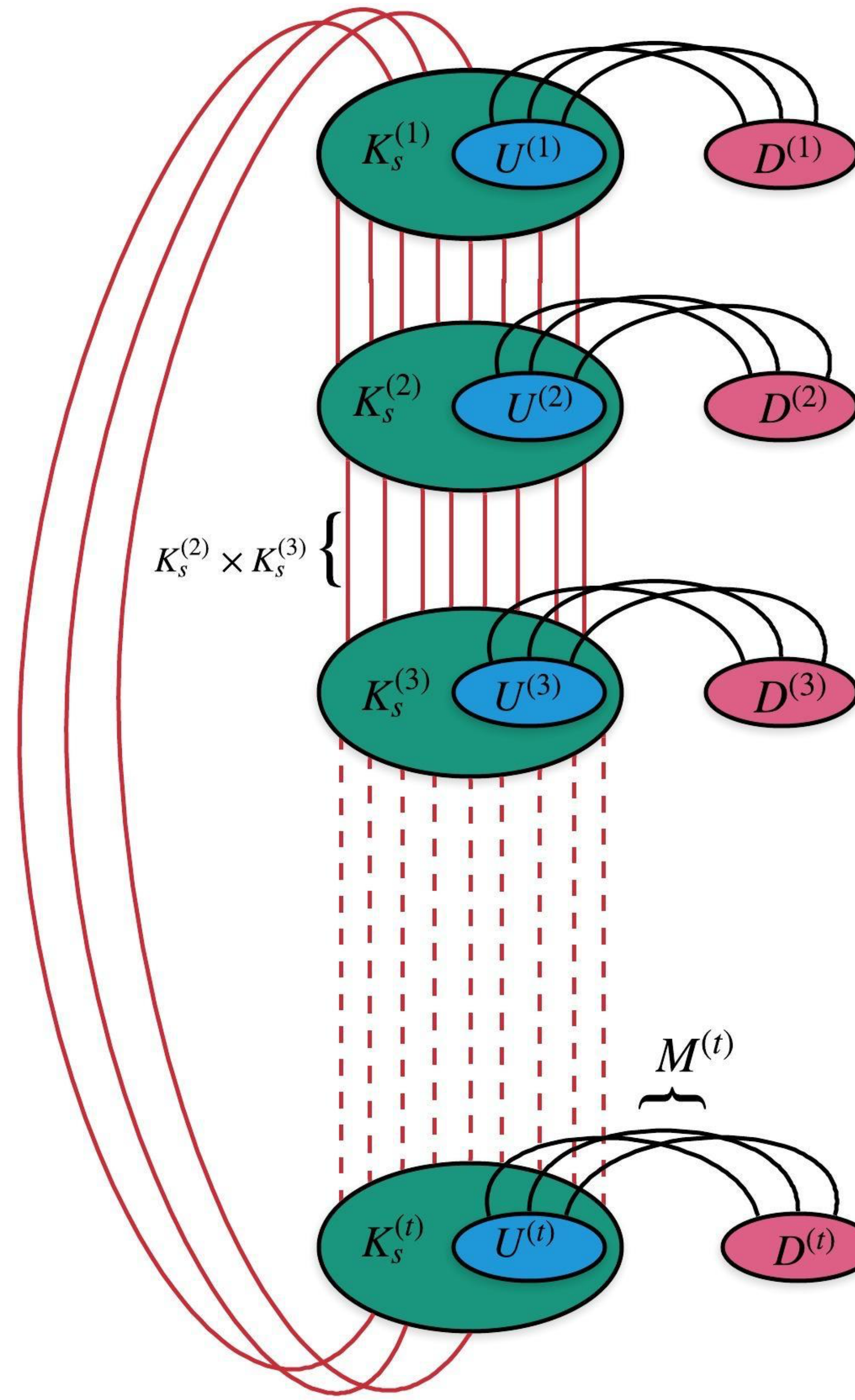
Lower Bound Graph

- **We have $t = \dfrac{n}{500 \log n}$ copies of the K_s, complete graph on vertices, where $s = 400 \log n$.**
- **These K_s's are arranged in a ring, with edges between each pair of vertices of consecutive K_s's.**
- **In each copy of K_s, we have a set of $100 \log n$ vertices, called $U^{(i)}$. Each of these $U^{(i)}$'s are matched to $D^{(i)}$. We call these "dangling" edges.**



The Lower Bound: I

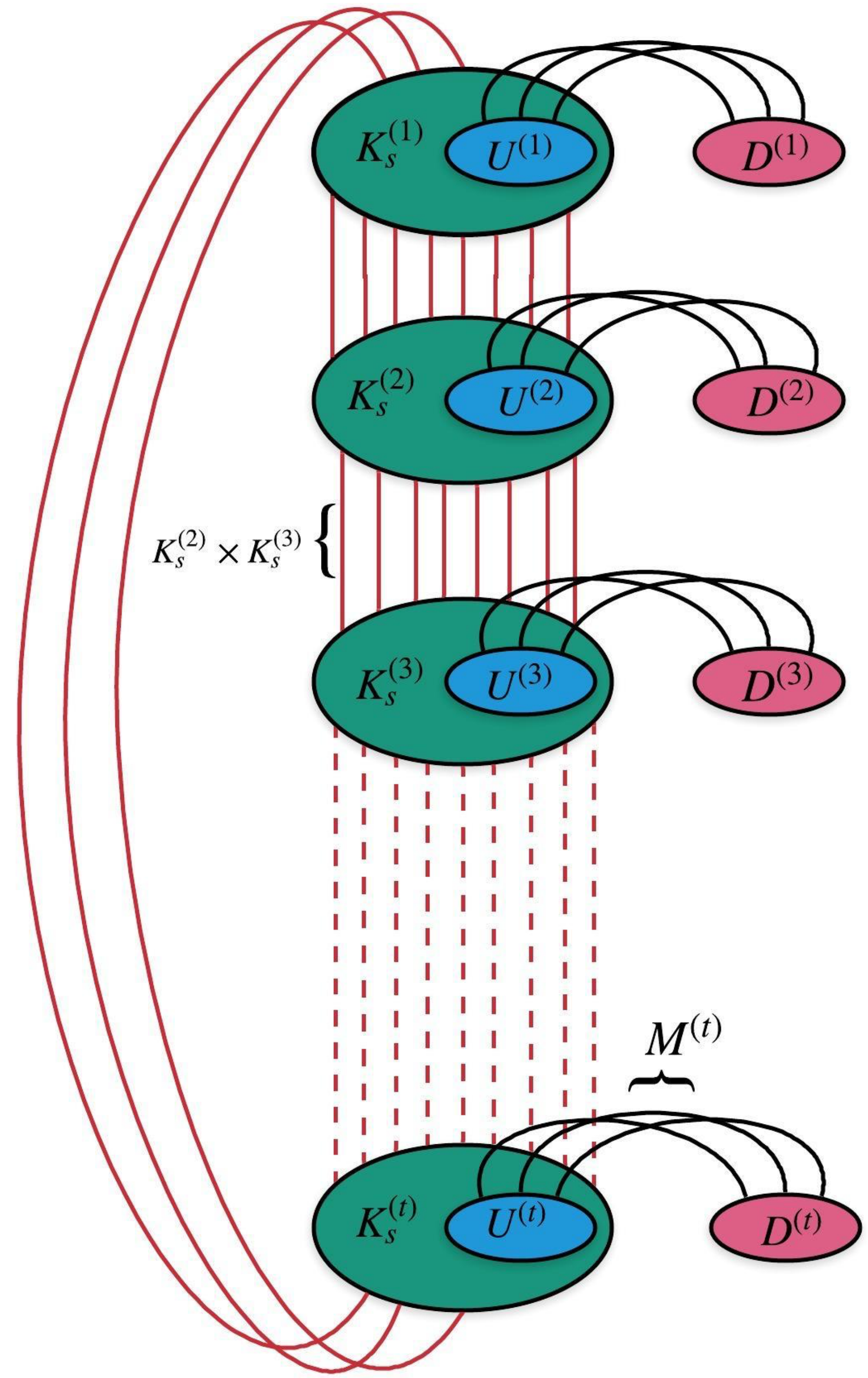
The Lower Bound: I



The Lower Bound: I

- Let be the lower bound graph. Let denote the graph obtained by sampling the edges of independently with probability and ignoring the degree 0 vertices of 's.

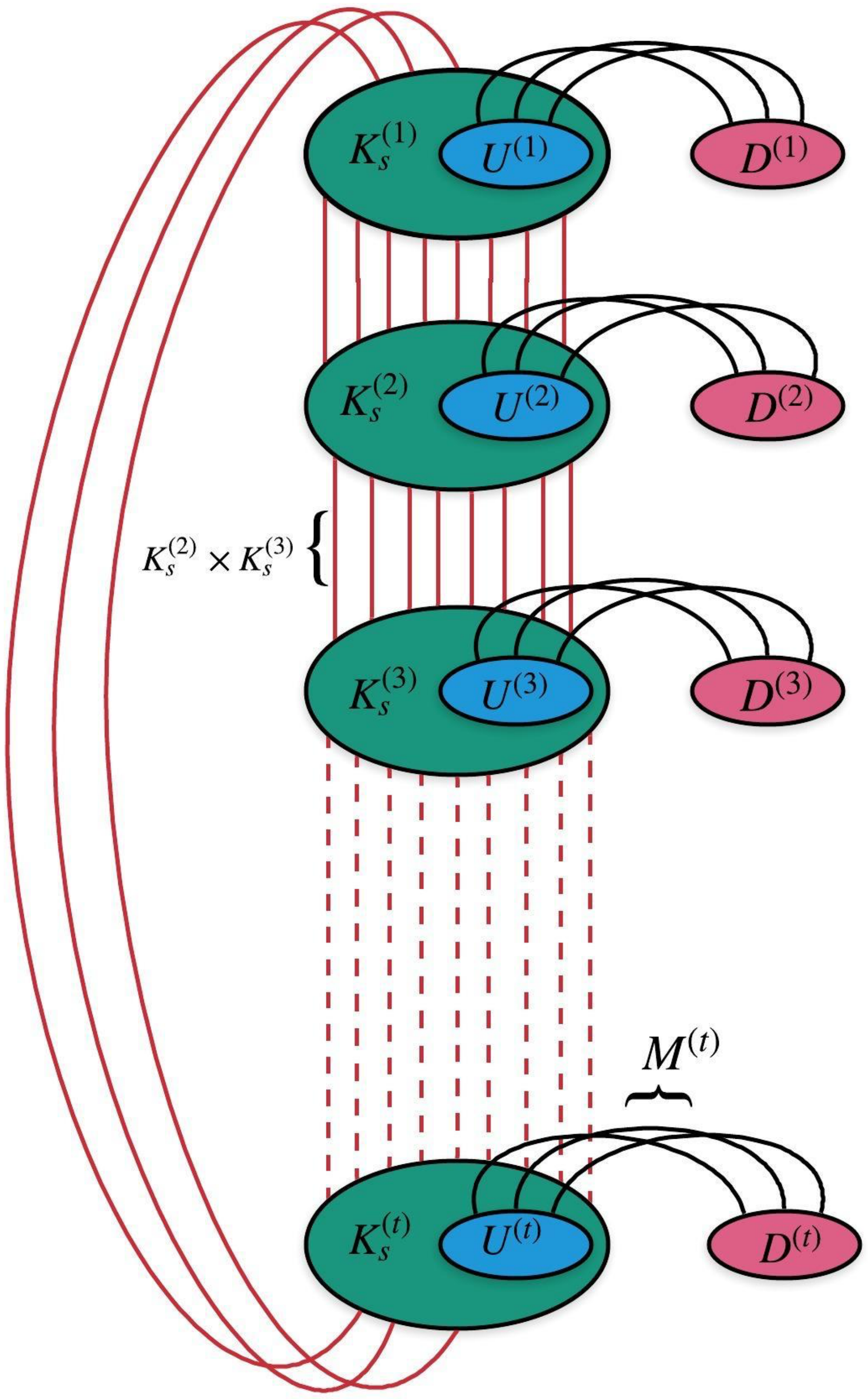
  Theorem 1: For , contains a perfect matching or a near perfect matching with high probability.



The Lower Bound: I

- Let G be the lower bound graph. Let denote the graph obtained by sampling the edges of independently with probability and ignoring the degree 0 vertices of 's.

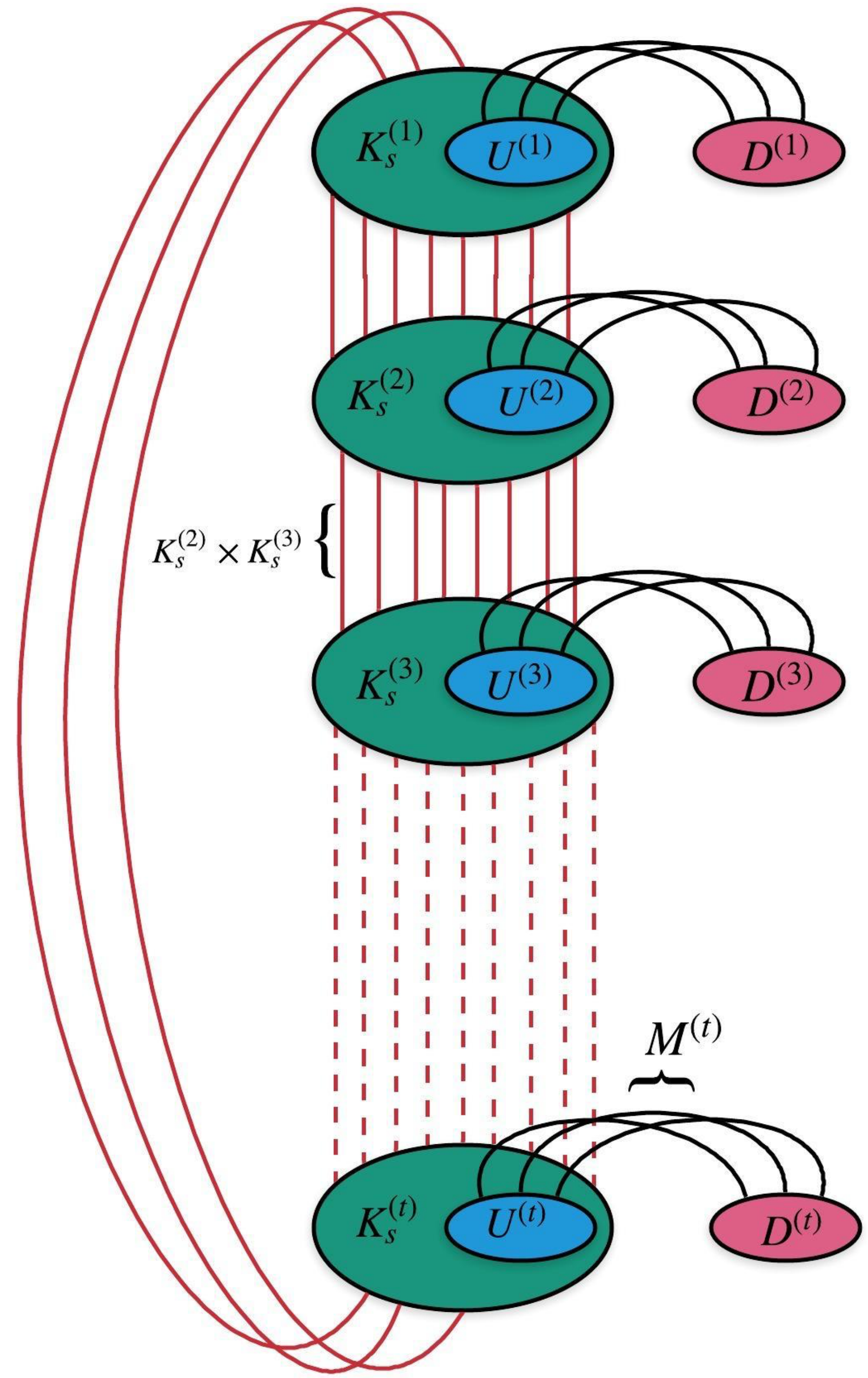
Theorem 1: For , contains a perfect matching or a near perfect matching with high probability.



The Lower Bound: I

- **Let G be the lower bound graph. Let G_p denote the graph obtained by sampling the edges of independently with probability and ignoring the degree 0 vertices of ’s.**

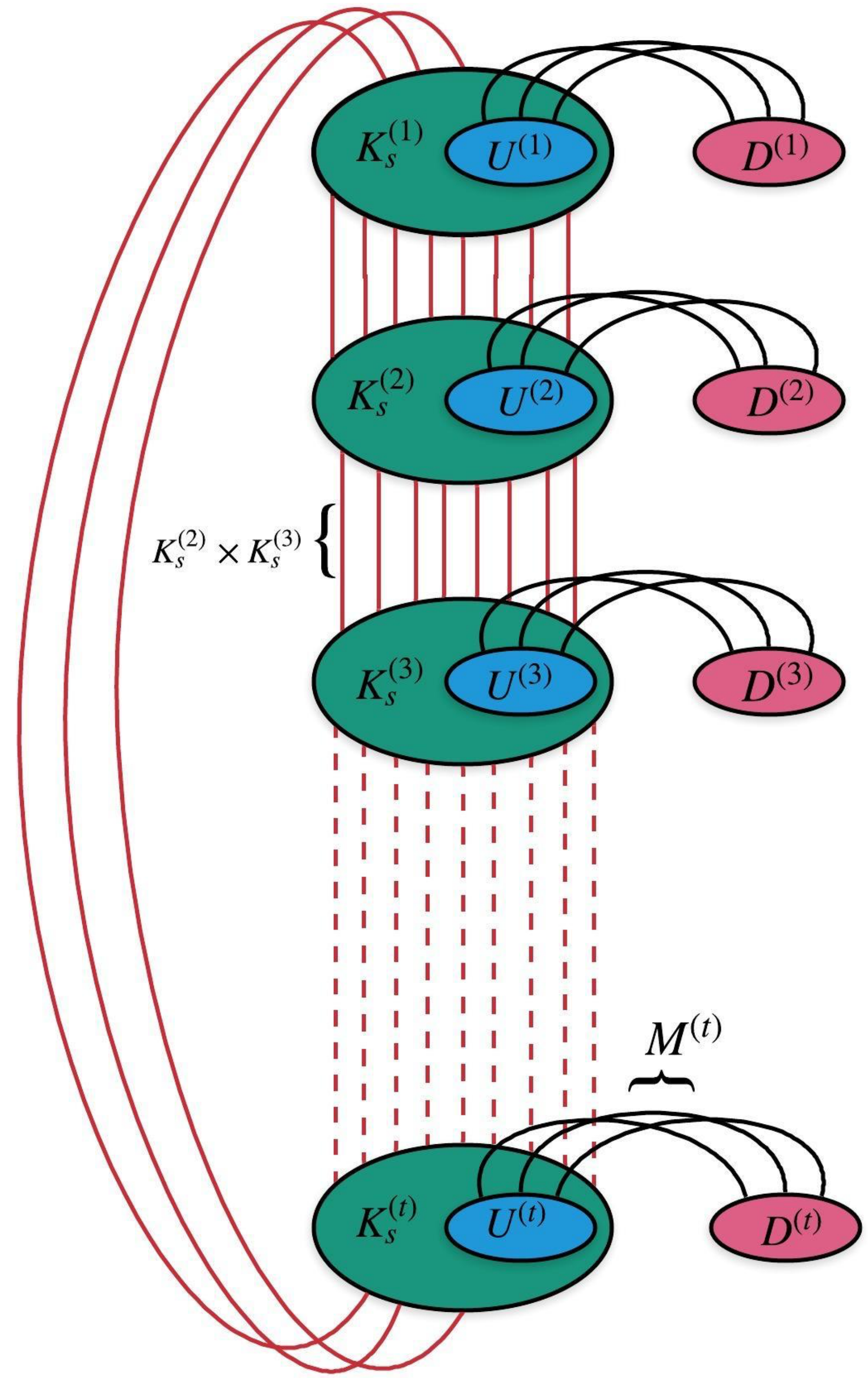
  Theorem 1: **For ,**
  contains a perfect matching or a near perfect matching with high probability.



The Lower Bound: I

- **Let G be the lower bound graph. Let G_p denote the graph obtained by sampling the edges of G independently with probability and ignoring the degree 0 vertices of ’s.**

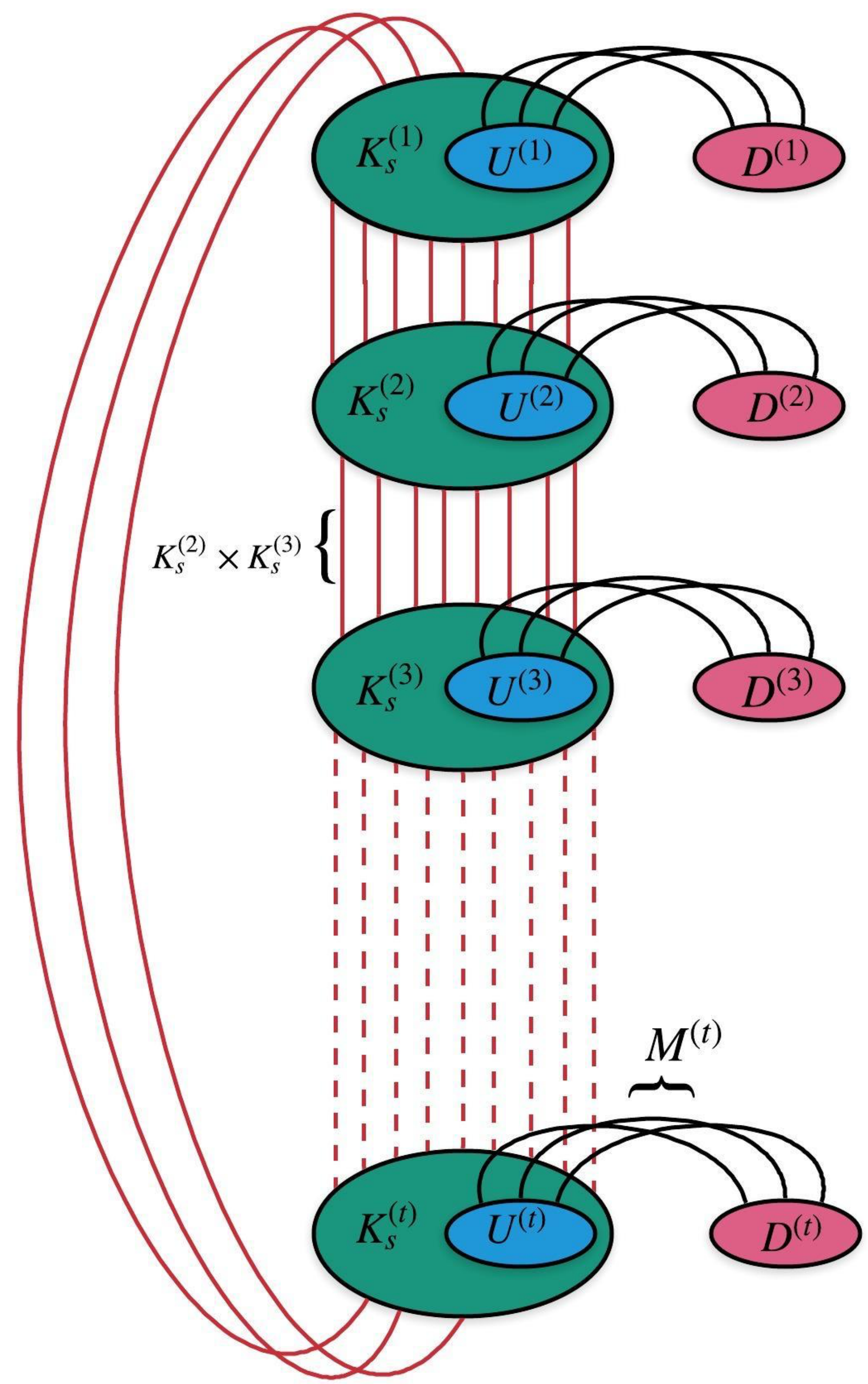
  Theorem 1: **For ,**
  contains a perfect matching or a near perfect matching with high probability.



The Lower Bound: I

- **Let G be the lower bound graph. Let G_p denote the graph obtained by sampling the edges of G independently with probability $p,$ and ignoring the degree 0 vertices of ’s.**

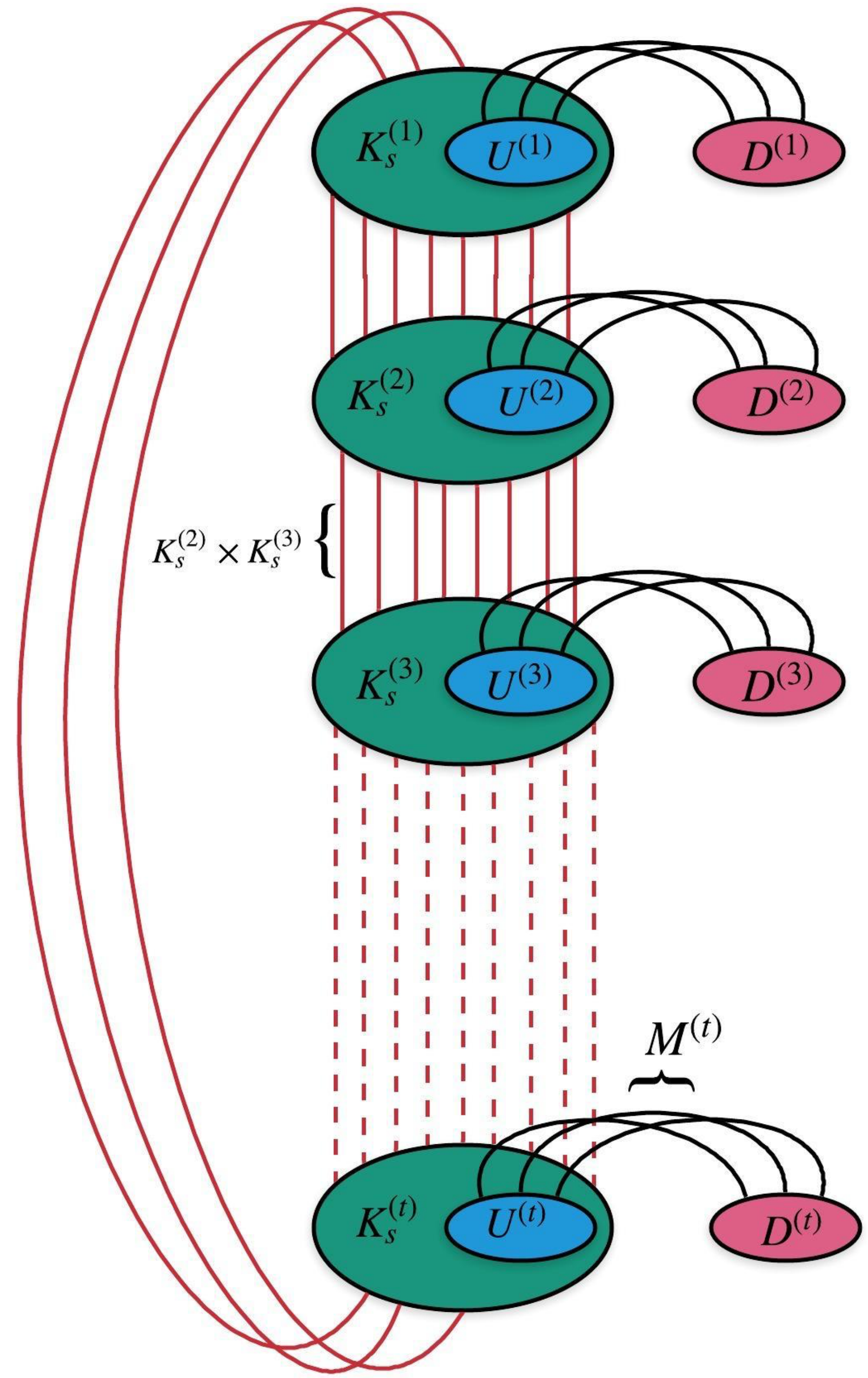
  Theorem 1: **For ,
  contains a perfect matching or a near perfect matching with high probability.**



The Lower Bound: I

- Let G be the lower bound graph. Let G_p denote the graph obtained by sampling the edges of G independently with probability p, and ignoring the degree 0 vertices of $D^{(i)}$'s.

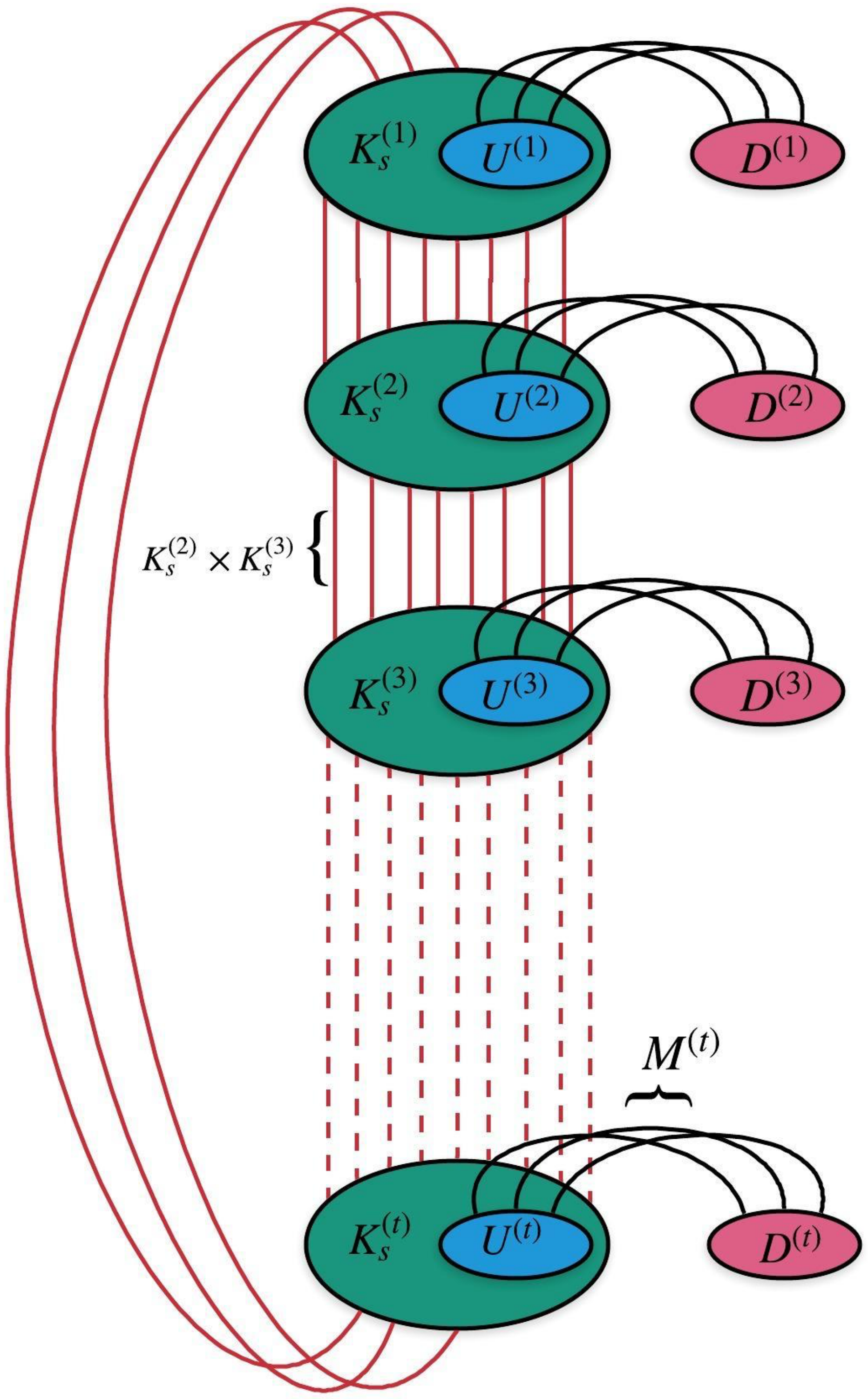
  Theorem 1: For ,
  contains a perfect matching or a near perfect matching with high probability.



The Lower Bound: I

- Let G be the lower bound graph. Let G_p denote the graph obtained by sampling the edges of G independently with probability p, and ignoring the degree 0 vertices of $D^{(i)}$'s.

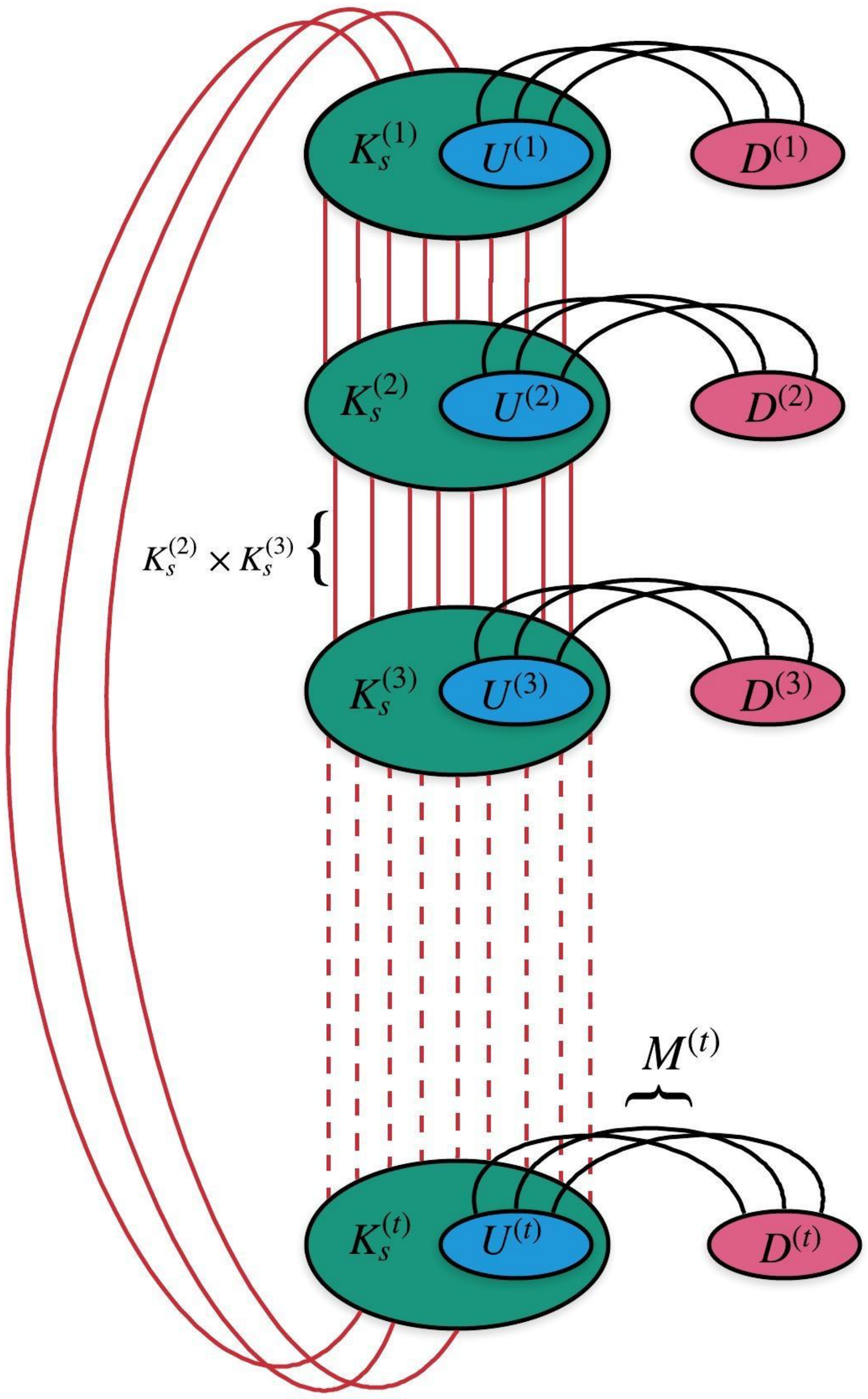
  Theorem 1: For , G_p contains a perfect matching or a near perfect matching with high probability.



The Lower Bound: I

- Let G be the lower bound graph. Let G_p denote the graph obtained by sampling the edges of G independently with probability p, and ignoring the degree 0 vertices of $D^{(i)}$'s.

  Theorem 1: For $p \in [1/2, 3/4]$, G_p contains a perfect matching or a near perfect matching with high probability.



The Lower Bound: II

- **Let m denote the total number of edges in the graph. Let $G^{p \cdot m}$ denote the graph obtained by sampling any $p \cdot m$ edges of G, and ignoring the degree 0 vertices of $D^{(i)}$'s.**

Theorem 1: For $p \in [1/2, 3/4]$, G_p contains a perfect matching or a near perfect matching with high probability.

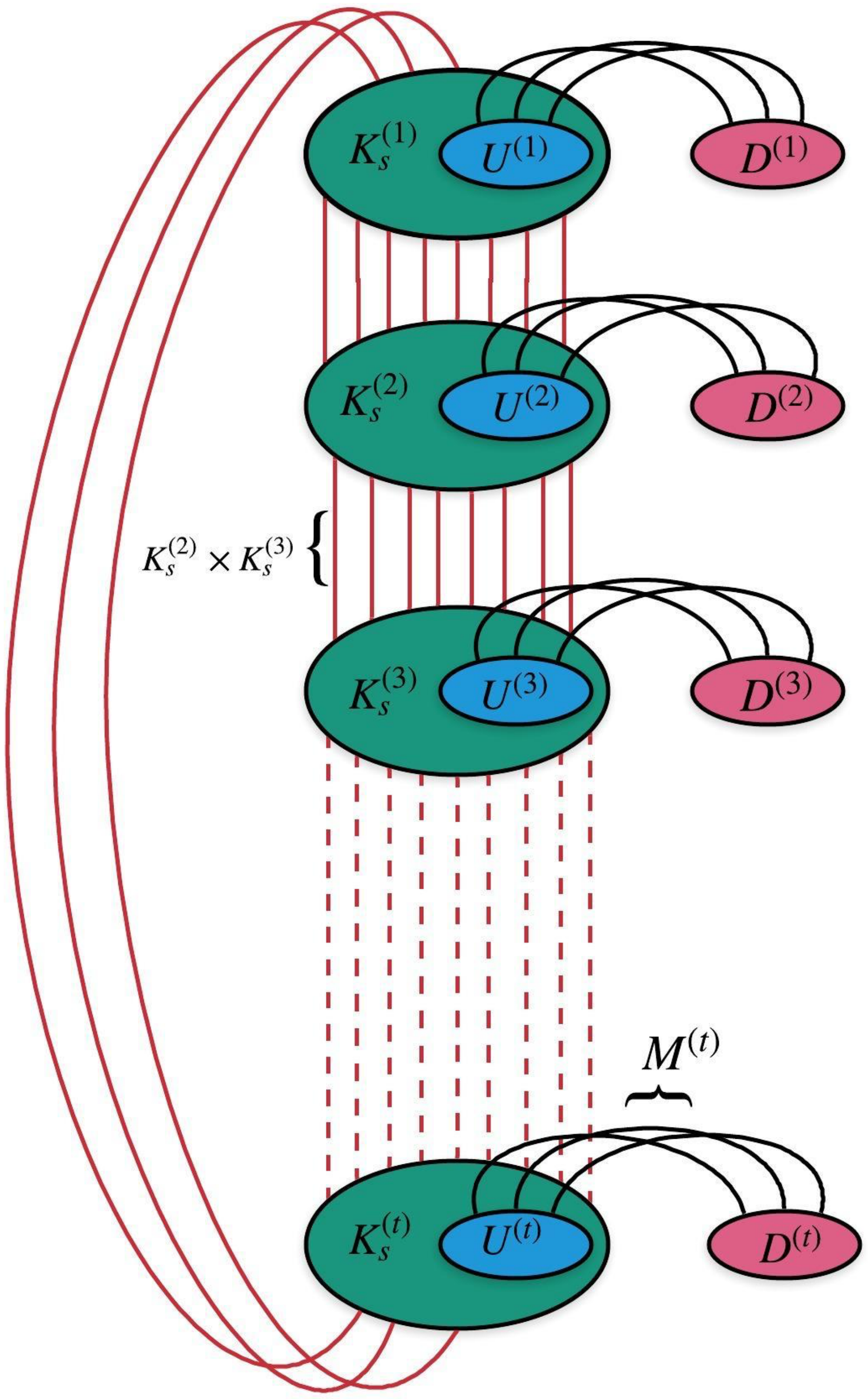
Theorem 2: For $p \in [1/2, 3/4]$, $G^{p \cdot m}$ contains a perfect matching or a near perfect matching with high probability.

The Lower Bound: II

- Let m denote the total number of edges in the graph. Let $G^{p \cdot m}$ denote the graph obtained by sampling any $p \cdot m$ edges of G, and ignoring the degree 0 vertices of $D^{(i)}$'s.

Theorem 1: For $p \in [1/2, 3/4]$, G_p contains a perfect matching or a near perfect matching with high probability.

Theorem 2: For $p \in [1/2, 3/4]$, $G^{p \cdot m}$ contains a perfect matching or a near perfect matching with high probability.



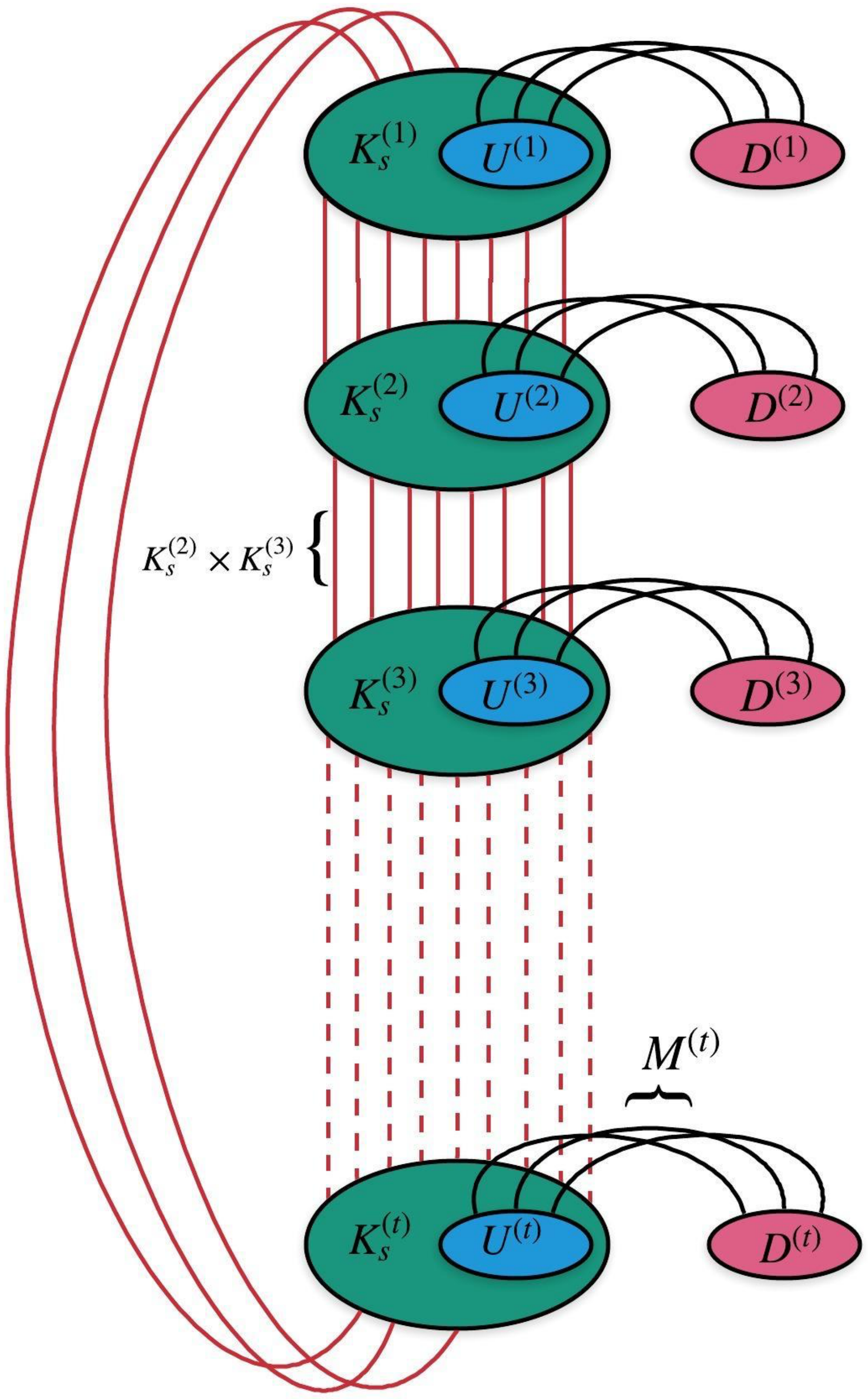
The Lower Bound: II

- **Let m denote the total number of edges in the graph. Let $G^{p \cdot m}$ denote the graph obtained by sampling any $p \cdot m$ edges of G, and ignoring the degree 0 vertices of $D^{(i)}$'s.**

Theorem 1: For $p \in [1/2, 3/4]$, G_p contains a perfect matching or a near perfect matching with high probability.

$$\Downarrow$$

Theorem 2: For $p \in [1/2, 3/4]$, $G^{p \cdot m}$ contains a perfect matching or a near perfect matching with high probability.



The Lower Bound: III

Theorem 2: For $p \in [1/2, 3/4]$, $G^{p \cdot m}$ contains a perfect matching or a near perfect matching with high probability.

Lower bound of $\Omega\left(n^2/\log n\right)$ on the recourse.

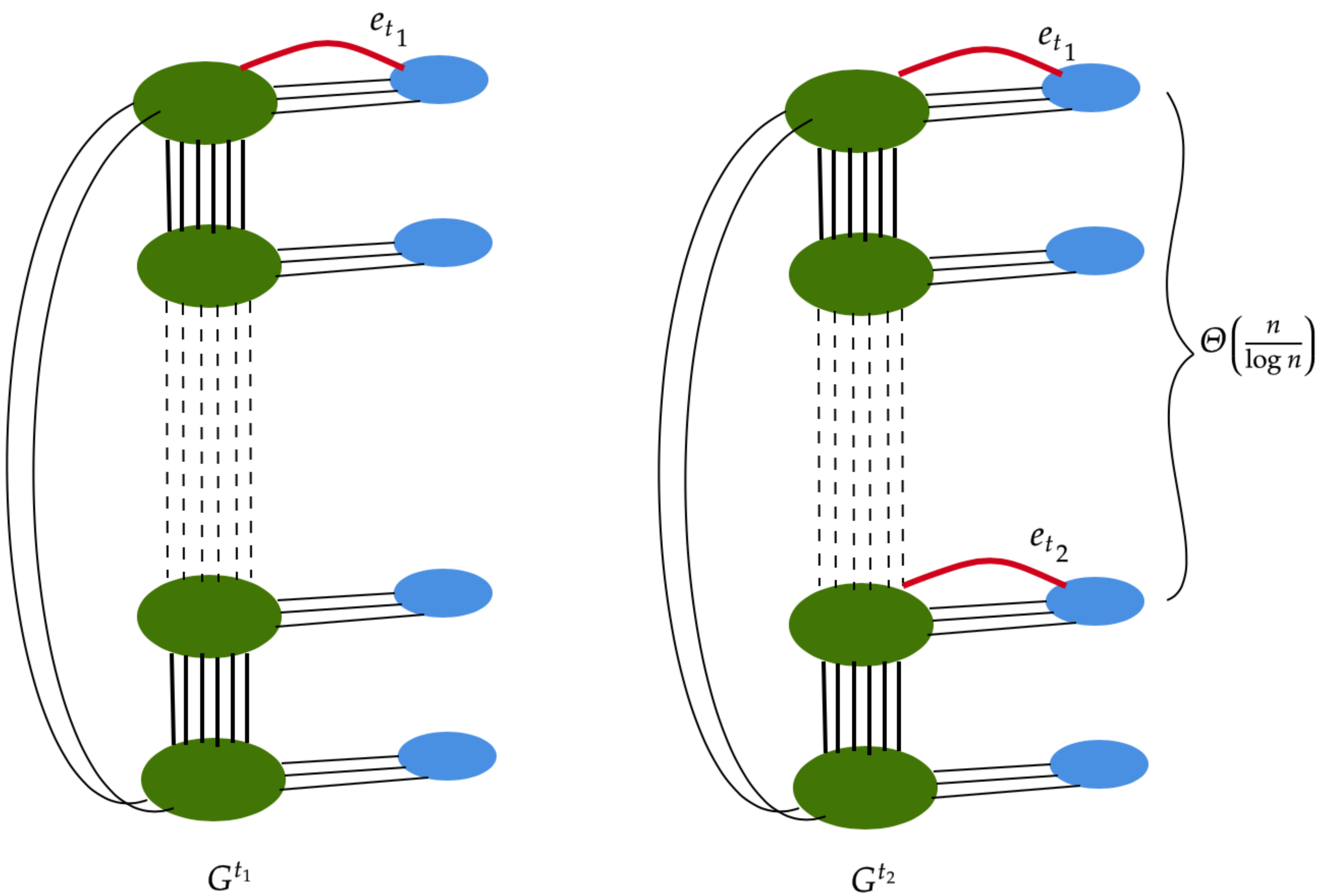
The Lower Bound: III

Theorem 2: **For** $p \in [1/2, 3/4]$**,** $G^{p \cdot m}$ **contains a perfect matching or a near perfect matching with high probability.**

$$\Downarrow$$

Lower bound of $\Omega\left(n^2/\log n\right)$ **on the recourse.**

The Lower Bound: IV



Lower Bound: VI

Lower Bound: VI

- **With high probability, between times and at least "dangling" edges arrive.**

Lower Bound: VI

- **With high probability, between times** $t = \frac{1}{2} \cdot m$ **and at least "dangling" edges arrive.**

Lower Bound: VI

- **With high probability, between times** $t = \frac{1}{2} \cdot m$ **and** $t = \frac{3}{4} \cdot m$, **at least “dangling” edges arrive.**

Lower Bound: VI

- **With high probability, between times** $t = \frac{1}{2} \cdot m$ **and** $t = \frac{3}{4} \cdot m,$ **at least** $\Omega(n)$ **"dangling" edges arrive.**

Lower Bound: VI

- **With high probability, between times $t = \frac{1}{2} \cdot m$ and $t = \frac{3}{4} \cdot m$, at least $\Omega(n)$ "dangling" edges arrive.**

- **Each of these join an augmenting path of expected length**

Lower Bound: VI

- **With high probability, between times** $t = \frac{1}{2} \cdot m$ **and** $t = \frac{3}{4} \cdot m$, **at least** $\Omega(n)$ **"dangling" edges arrive.**

- **Each of these join an augmenting path of expected length** $\Theta\left(n/\log n\right)$.

Lower Bound: VI

- **With high probability, between times $t = \frac{1}{2} \cdot m$ and $t = \frac{3}{4} \cdot m$, at least $\Omega(n)$ "dangling" edges arrive.**
- **Each of these join an augmenting path of expected length $\Theta\left(n/\log n\right)$.**
- **This implies total expected recourse is**

Lower Bound: VI

- **With high probability, between times** $t = \frac{1}{2} \cdot m$ **and** $t = \frac{3}{4} \cdot m$, **at least** $\Omega(n)$ **"dangling" edges arrive.**

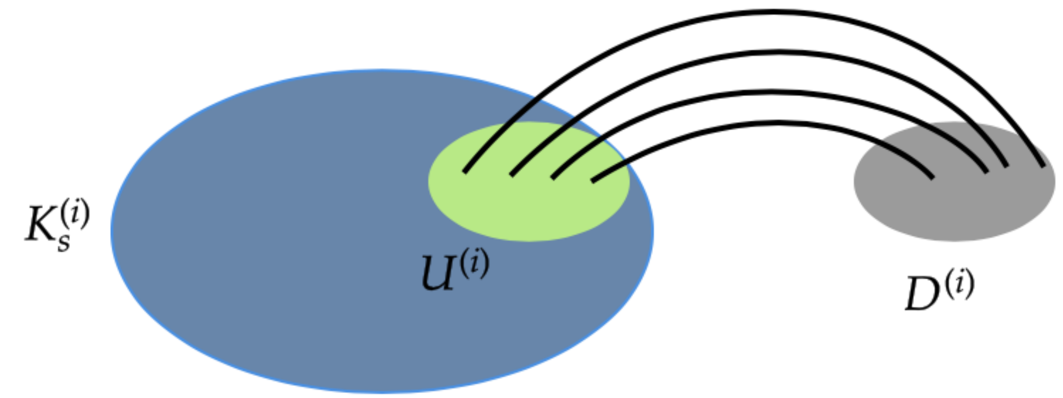
- **Each of these join an augmenting path of expected length** $\Theta\left(n / \log n\right)$.

- **This implies total expected recourse is** $\Omega\left(n^2 / \log n\right)$.

Proof of Theorem 1

Theorem 1: **For $p \in [1/2, 3/4]$, G_p contains a perfect matching or a near perfect matching with high probability.**

Claim 1: **Consider $K_s^{(i)}$ while ignoring the vertices whose "dangling" edges have been included in G_p. Then, $K_s^{(i)}$ contains a perfect or a near perfect matching with high probability.**

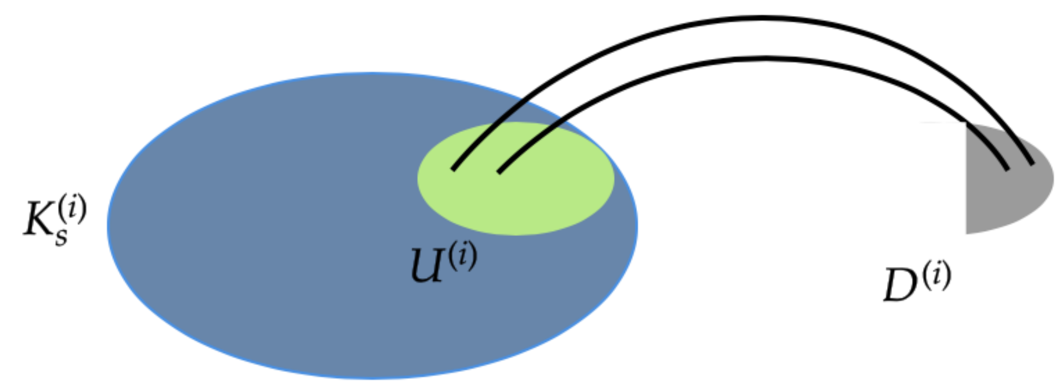


$K_s^{(i)}$ in graph G

Proof of Theorem 1

Theorem 1: For $p \in [1/2, 3/4]$, G_p contains a perfect matching or a near perfect matching with high probability.

Claim 1: Consider $K_s^{(i)}$ while ignoring the vertices whose "dangling" edges have been included in G_p. Then, $K_s^{(i)}$ contains a perfect or a near perfect matching with high probability.

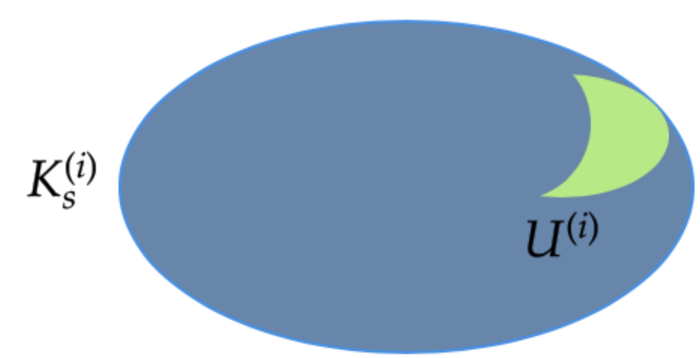


$K_s^{(i)}$ in graph G_p

Proof of Theorem 1

Theorem 1: For $p \in [1/2, 3/4]$, G_p contains a perfect matching or a near perfect matching with high probability.

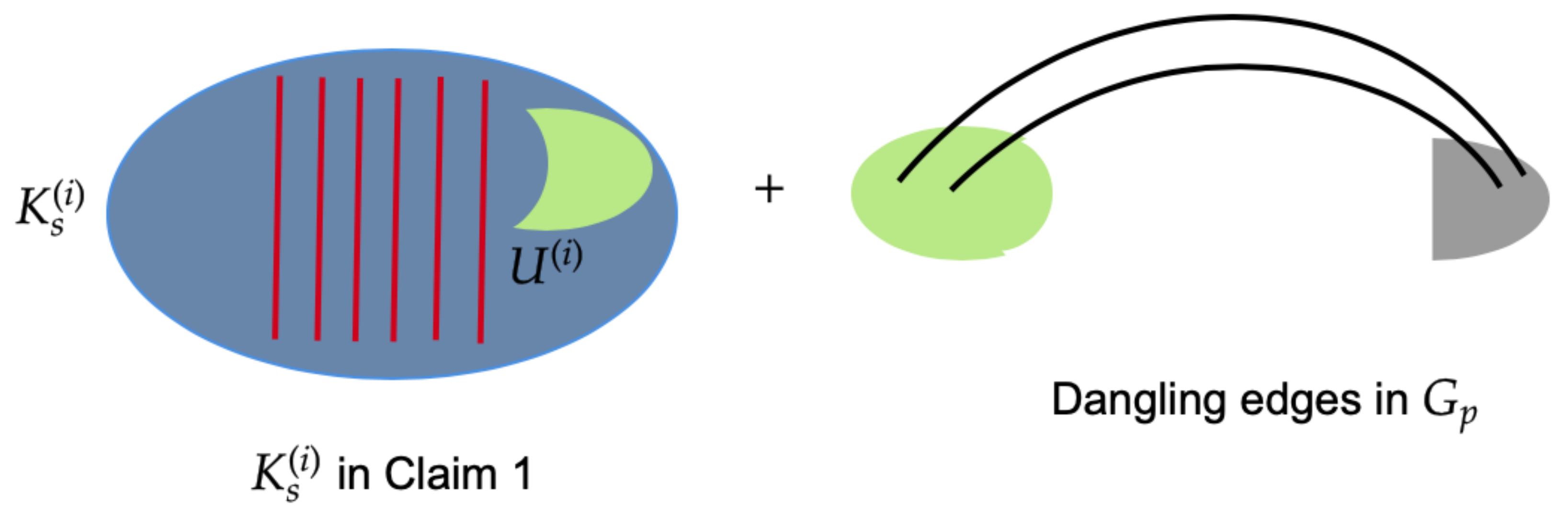
Claim 1: Consider $K_s^{(i)}$ while ignoring the vertices whose "dangling" edges have been included in G_p. Then, $K_s^{(i)}$ contains a perfect or a near perfect matching with high probability.



$K_s^{(i)}$ in Claim 1

Proof of Theorem 1

Suppose each $K_s^{(i)}$ contains a perfect matching, then G_p contains a perfect matching.



Proof of Theorem 1

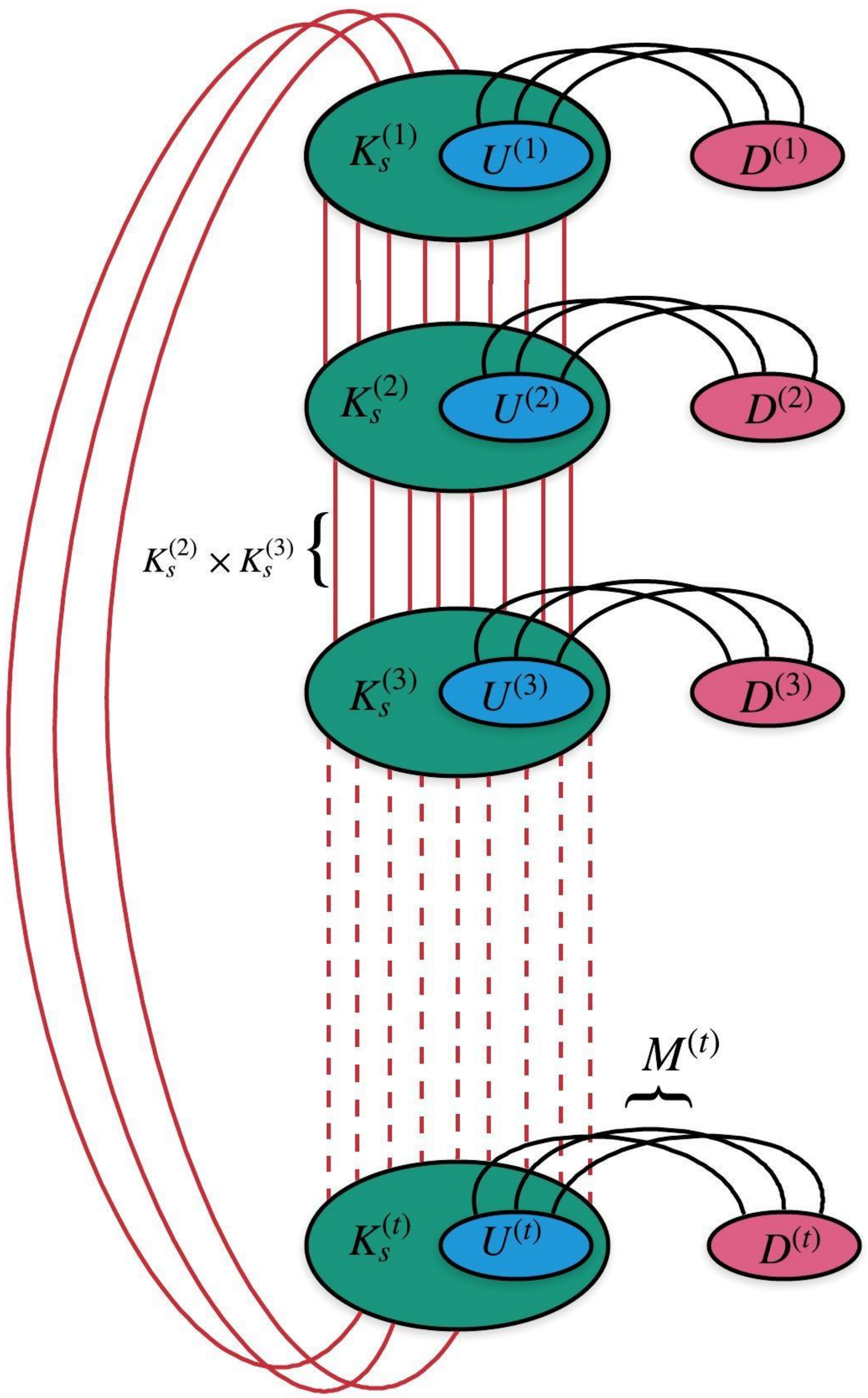
Theorem 1: For $p \in [1/2, 3/4]$, G_p contains a perfect matching or a near perfect matching with high probability.

Claim 2: Fix maximum matching $\mathscr{M}_i$ in $K_s^{(i)}$. Conditioning on Claim 1, let $K_s^{(j)}$ and $K_s^{(k)}$ be two consecutive "deficient" K_s's (that is K_s's that have a near perfect matching). Then, with high probability, there is an augmenting path joining the unmatched vertex in $K_s^{(j)}$ with the unmatched vertex in $K_s^{(k)}$.

Proof of Theorem 1

Theorem 1: For $p \in [1/2, 3/4]$, G_p contains a perfect matching or a near perfect matching with high probability.

Claim 2: Fix maximum matching $\mathscr{M}_i$ in $K_s^{(i)}$. Conditioning on Claim 1, let $K_s^{(j)}$ and $K_s^{(k)}$ be two consecutive "deficient" K_s's (that is K_s's that have a near perfect matching). Then, with high probability, there is an augmenting path joining the unmatched vertex in $K_s^{(j)}$ with the unmatched vertex in $K_s^{(k)}$.

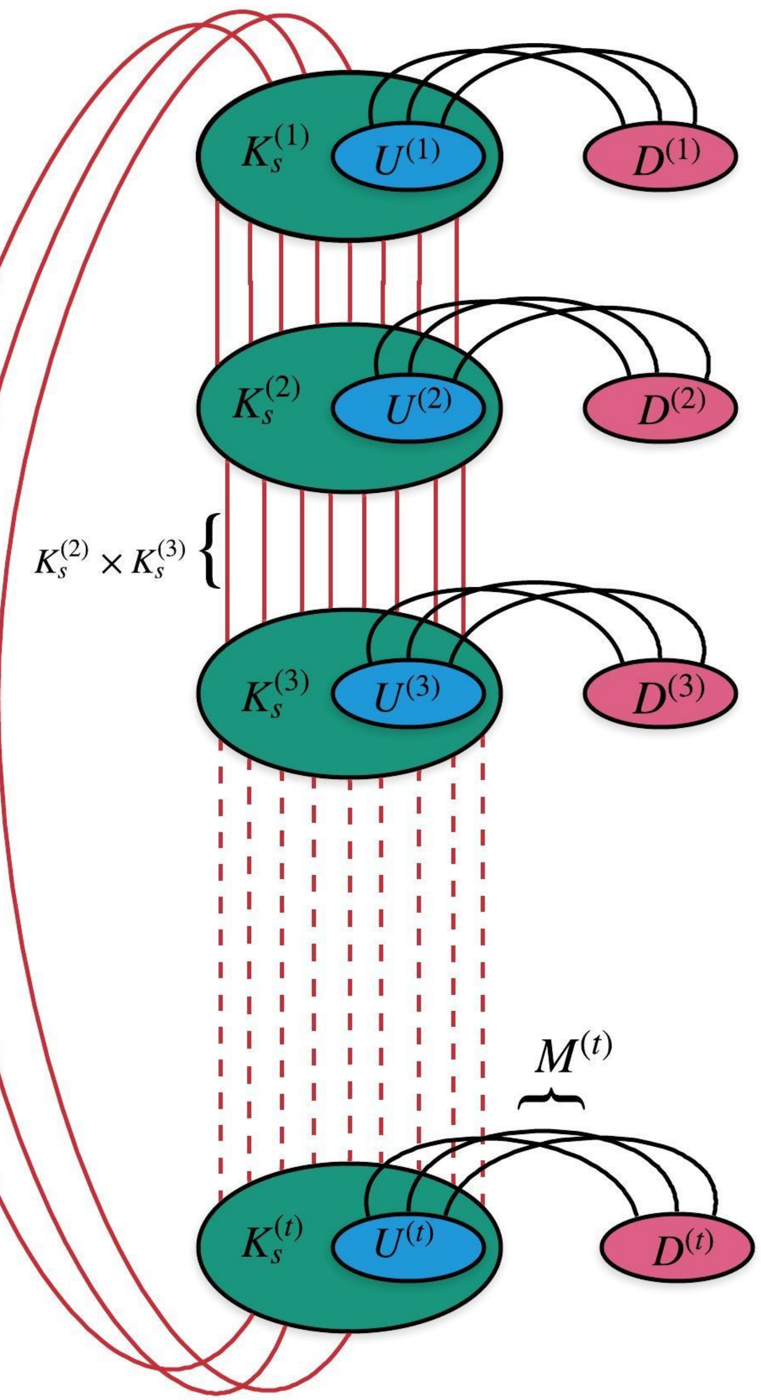


Proof of Theorem 1

- Claim 1 and Claim 2 imply Theorem 1. We essentially show that there is a near perfect or perfect matching in each of the K_s 's with high probability. We pair up the deficient K_s 's, and show that there is an augmenting path between the unmatched vertices with high probability. It follows that there is at most one vertex in all of G_p that is unmatched, which implies that there is a perfect or a near perfect matching.

Proof of Theorem 1

- **Claim 1** and **Claim 2** imply Theorem 1. We essentially show that there is a near perfect or perfect matching in each of the K_s 's with high probability. We pair up the deficient K_s 's, and show that there is an augmenting path between the unmatched vertices with high probability. It follows that there is at most one vertex in all of G_p that is unmatched, which implies that there is a perfect or a near perfect matching.



Proof of Claim 1

- **Claim 1 follows from the following well-known theorem:**

  Theorem: Let $G_{n,p}$ be the graph obtained by adding an edge between every pair of vertices independently and with probability p, then,

Proof of Claim 1

- **Claim 1 follows from the following well-known theorem:**

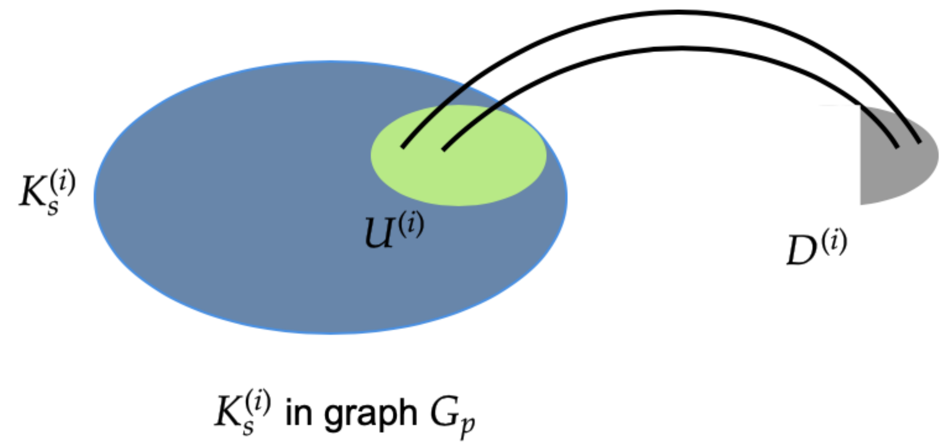
  Theorem: Let $G_{n,p}$ be the graph obtained by adding an edge between every pair of vertices independently and with probability p, then,

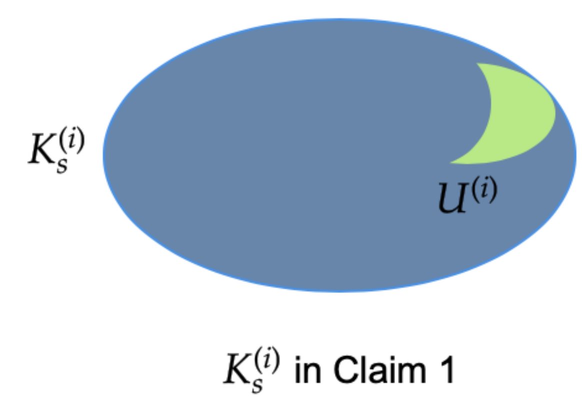
$$\textbf{Pr}(G_{n,p} \textbf{ does not contain a perfect matching}) = O\left(ne^{-np}\right)$$

Proof of Claim 1

- **Claim 1 follows from the following well-known theorem:**

  Theorem: Let $G_{n,p}$ be the graph obtained by adding an edge between every pair of vertices independently and with probability p, then,





$$\Pr(G_{n,p} \text{ does not contain a perfect matching}) = O\left(ne^{-np}\right)$$

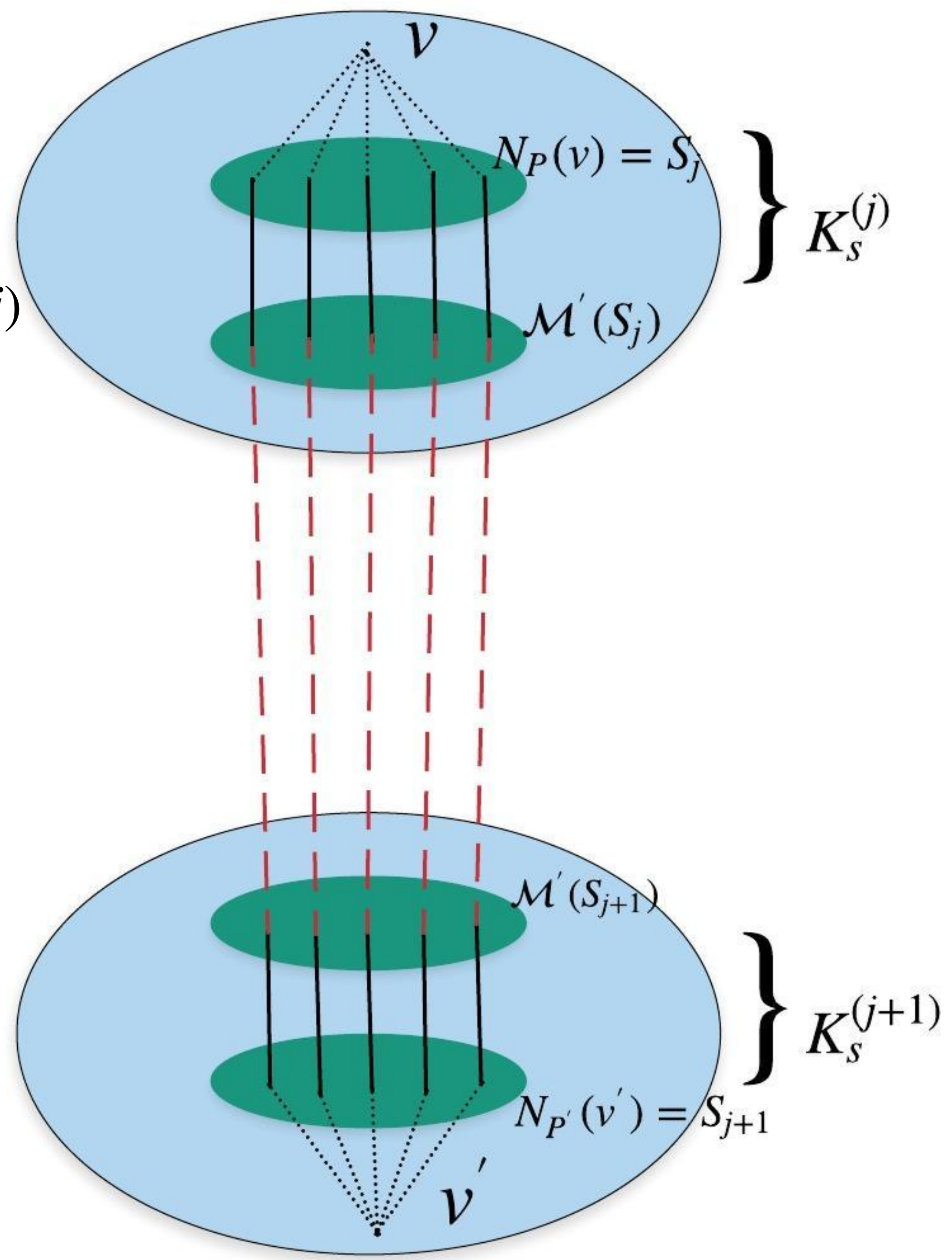
Proof of Claim 2

- Case 1: **When "deficient" K_s's are consecutive.**

Proof idea: **Consider a bipartition $P \cup Q$ of $K_s^{(j)}$ and a bipartition $P' \cup Q'$ of $K_s^{(j+1)}$ according to $\mathscr{M}_j$ and $\mathscr{M}_{j+1}$.**

Note that $|N_P(v)| = \Omega(\log n)$. Further, conditioned on Claim 1, $|\mathscr{M}_j(N_P(v))| = \Omega(\log n)$.

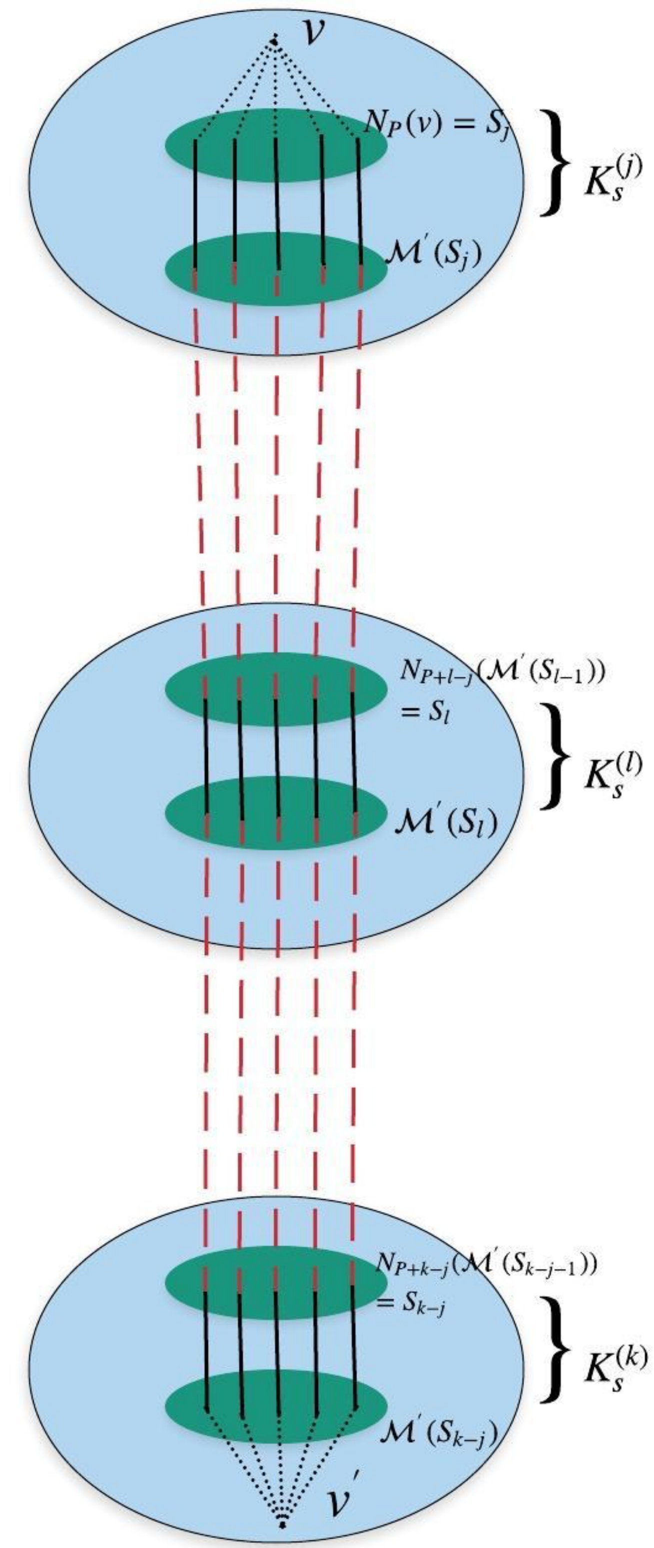
This holds for v' as well. Since $|\mathscr{M}_j(N_P(v))| = \Omega(\log n)$ and $|\mathscr{M}_{j+1}(N'_P(v'))| = \Omega(\log n)$ it follows there must an edge between these sets and therefore an augmenting path between v and v'.



Proof of Claim 2

- **Case 2: When the "deficient" K_s 's are not consecutive.**

Proof idea: Let $(P_l) \cup (Q_l)$ denote the bipartition of $K_s^{(l)}$ for $j \le l \le k$. We can inductively prove that v has alternating paths to a large number of vertices in Q_l for $j \le l \le k$ and therefore, to Q_k. Since the number of such vertices is large, it follows that v' must have an edge to one of them. Therefore, there is an augmenting path from v to v'.



Open Questions

- Can we show a lower bound of for the case of bipartite graphs as well?
- Close the gap between upper and lower bounds for the case of trees?

Open Questions

- Can we show a lower bound of $\Omega\left(n^2/\log n\right)$ for the case of bipartite graphs as well?
- Close the gap between upper and lower bounds for the case of trees?