CS 362, Lecture 25

Jared Saia
University of New Mexico

Today's Outline

- Approximation algorithms for NP-Hard Problems
- Data Migration

Motivation

- Performance of large storage system depends on having data assignment which balances request load
- Optimal layout of data changes with changing workloads
- Want to periodically recompute a new optimal assignment of data to devices
- Must move data to new optimal assignment **as quickly as possible**

Migration Problem

Given:

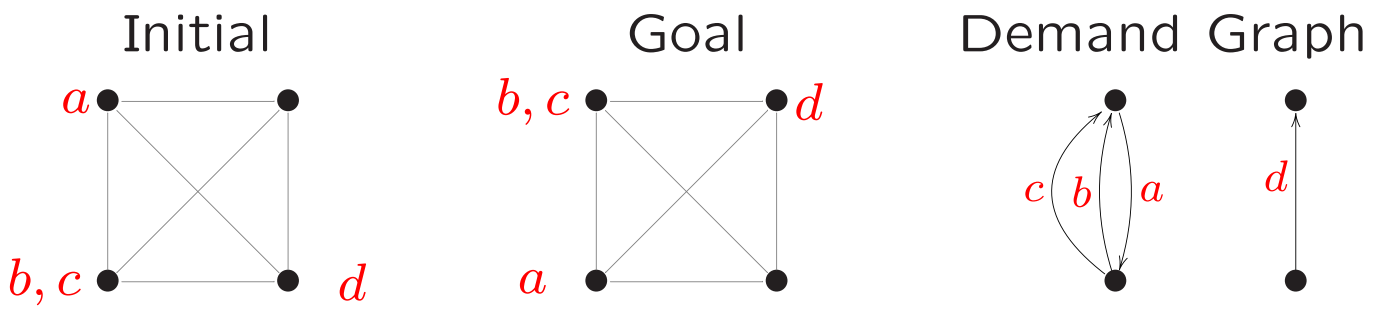
- Initial and final configuration of data objects on devices
- Description of the storage system.

Our Goal: To compute a *migration plan* obeying network constraints and using minimal number of time steps.

Key constraint:
In one time step each storage device can be involved in the transfer of one data object.

Demand Graph

Assume network is a complete graph



Output: Assignment of time steps to each edge in demand graph such that no two edges incident to same node are assigned the same time step

Edge Coloring

Simple Migration Model:

- Assume fully connected network
- In one time step, each storage device can be involved in the transfer of one data object.

Thus the simple migration problem is equivalent to edge coloring.

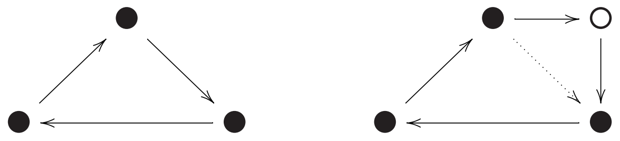
Multigraph Edge Coloring

- Minimum number of colors for edge coloring is χ'
- Maximum degree of graph (Δ) is trivial lower bound on χ'
- Computing χ' is NP-Complete [Hoyer, 1981]
- Best known approximation algorithm for multigraphs is $9/8\chi' + 3/4$ [Hochbaum et al, 1996]

Beyond Edge Coloring

- Indirect migration
  - Allowed to send an object through intermediate or *bypass nodes*
  - This can decrease the number of time steps required
- Migration with memory constraints
  - In any intermediate stage of the plan, all machines must have a number of objects stored on them less than or equal to the memory of that machine.

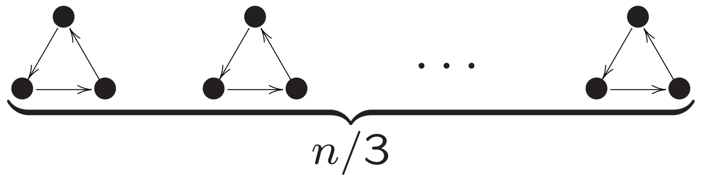
Indirect Migration



What can we hope for?

Given: arbitrary directed multigraph G with max degree Δ and n nodes.

- Clearly need $\geq \Delta$ steps for any migration
- In worst case, need $n/3$ bypass nodes to achieve Δ



Direct migrations

- need $\geq \chi'$ steps
- in worst case, $\chi' = \frac{3}{2}\Delta$

Migration Results

Given: Directed multigraph G with max degree Δ and n nodes.

Direct Migration (edge-coloring):

- Upperbound: $3\lceil \Delta/2 \rceil$ steps
- Lowerbound: $2\lceil \Delta/2 \rceil$ steps

Indirect Migration (edge-coloring):

- Upperbound: $2\lceil \Delta/2 \rceil$ steps with $n/3$ bypass nodes
- Lowerbound: Δ steps

Graph Factoring

- Key Idea: Graph Factoring
- Let G be a k-regular graph where k is an even number.
- We can *factor* G into $k/2$ 2-regular graphs, G_i, $1 \leq i \leq k/2$ such that each edge in G occurs exactly once in some G_i

Regular Graphs

- Key Idea: Can make any graph, G, with max degree Δ into a graph G' which is $\Delta' = 2\lceil \Delta/2 \rceil$-regular graph by adding dummy edges to G
- Any migration for G' gives a migration for G (just ignore the dummy edges)

Making a graph Δ'-regular

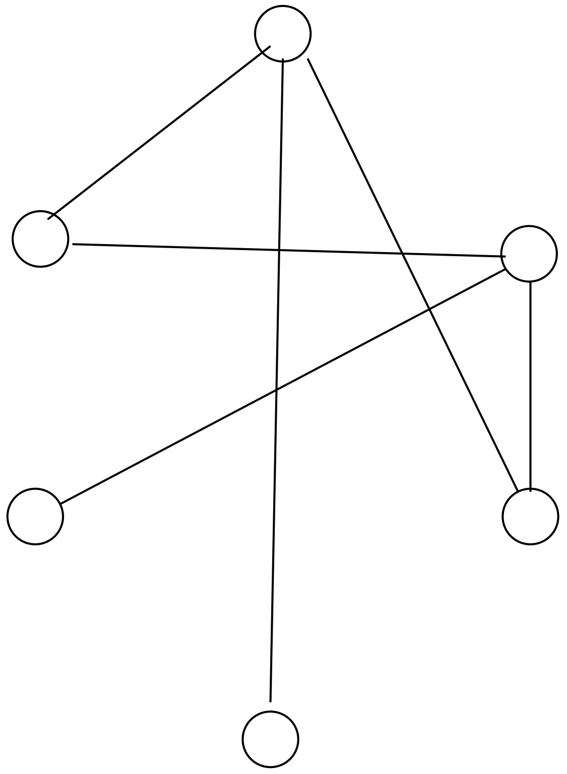
The following algorithm takes a graph with degree Δ and adds edges to it to get a graph of degree $\Delta' = 2\lceil \Delta/2 \rceil$

Algorithm *Regular*

1. While there exists a vertex with degree less than $\Delta' - 1$, add a self loop to that vertex.
2. While there exist two distinct vertices of degree $\Delta' - 1$, add an arbitrarily directed edge between them.

In-Class Excercise

Make the following graph 4 regular as specified in Algorithm *Regular*



Q: Intuitively, why does this algorithm work?

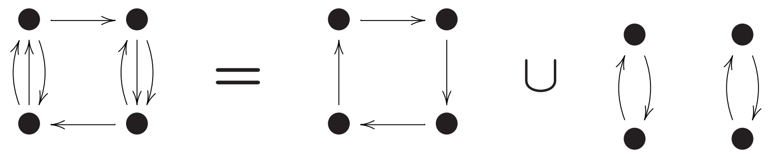
Factoring

- Now that we have a Δ'-regular graph, we can factor it into a bunch of 2-regular graphs
- Once we have the 2-factors of the graph, the migration algorithms are easy
- The challenging part is getting the 2-factors

Direct Migration - Results

$3\lceil \Delta/2 \rceil$ steps maximum

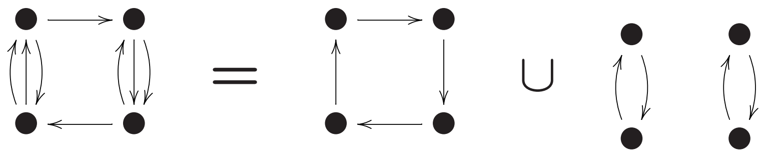
Idea: Factor graph into set of 2-regular graphs



Indirect Migration - Results

$2\lceil \Delta/2 \rceil$ steps with $n/3$ bypass nodes.

Idea: Factor graph into set of 2-regular graphs. Use $n/3$ bypass nodes for each "factor" to satisfy it in 2 steps.



Graph Factoring

Let G be a k-regular graph where k is an even number. We factor G as follows:

1. Construct an Euler-tour of G.
2. Orient the edges according to the direction of the tour. That is, if the tour enters v on edge e_1 and leaves on edge e_2, then e_1 is an in-edge to v and e_2 is an out-edge. Thus we have $d_{\text{in}} = d_{\text{out}} = k$.
3. Set up a bipartite matching problem, B_G, with a representative of each vertex in the graph on both sides. Add in all directed edges going from left to right. Note that each edge is represented in the matching problem exactly once.
4. Find a matching (which is guaranteed to exist by Hall's Theorem). The matched edges induce a 2-factor of the original graph. Remove these edges from B_G and repeat this step until there are no edges left.

Euler Tour

An *Euler Tour* of a graph G is a tour which starts and ends at the same vertex and traverses every edge exactly once.

It is well known that a graph G has an Euler Tour if and only if each vertex of the graph has even degree.

Defn: A Matching

Let G be any graph with edge set E. Then a *matching* on G is a collection of edges E' where $E' \subseteq E$ and no two edges in E' share an endpoint.

Question: If G has n vertices, what are the maximum and minimum possible sizes of a matching?

Defns

A graph G is *bipartite* if it's vertex set V can be partitioned into two sets L, R such that all edges in G have one edge in L and one edge in R.

A matching is *perfect* if for every vertex $l \in L$, there is some edge in the matching which is incident to l.

Definition: Let $G = (L, R)$. For any subset L' of L, define $N(L')$ to be the set of vertices which are neighbors of L.

Question: If $|N(L)| < |L|$, can there be a perfect matching?

Hard Question: When does a bipartite graph have a perfect matching?

Hall's Theorem

Let $G = (L, R)$ be a bipartite graph. Then:

- There is a perfect matching for G

if and only if

- $\forall L' \subseteq L$, $|L'| \leq |N(L')|$

We will say a bipartite graph "has Hall's property" if it fulfills this condition.

Corollary

For a bipartitie $G = (L, R)$, if

- G is k-regular,

Then

- $\forall L' \subseteq L$, $|L'| \leq |N(L')|$.

Proof: Consider any set $L' \subseteq L$, then

- Number of edges between L' and $N(L') = k * |L'|$ (since k edges from each vertex in L)
- Number of edges between L' and $N(L') \leq k * |N(L')|$ (since k edges from each vertex in $N(L')$)
- Implies $|L'| \leq |N(L')|$

Proof of Hall's Theorem

We will now prove Hall's Theorem:

Let $G = (L, R)$ be a bipartite graph. Then

- There is a perfect matching for G

if and only if

- $\forall L' \subseteq L$, $|L'| \leq |N(L')|$

where $|X|$ is the number of vertices in the set X and $N(X)$ is the set of neighbors of the set X.

Proof of Hall's Theorem

Easy direction:

If

- there is a perfect matching for G,

then

- $\forall L' \subseteq L,\ |L'| \leq |N(L')|$

Easy Direction

if

- there is a perfect matching for G,

then

- $\forall L' \subseteq L$, $|L'| \leq |N(L')|$

Proof:

Assume there is a matching and that there is some set L' such that $|L'| > |N(L')|$. We know that every vertex in L' must be matched in the matching, hence we know that $N(L')$ contains at least $|L'|$ neighbors. This is a contradiction since we assumed that $|L'| > |N(L')|$.

Hard Direction

If

- $\forall L' \subseteq L,\ |L'| \leq |N(L')|$

then

- there is a perfect matching for G,

Proof of Hall's

To show: If for some graph $G = (L, R)$, $\forall L' \subseteq L$, $|L'| \leq |N(L')|$ then there is a perfect matching for G.

We will say that a graph is "slack" if it's the case that $\forall L' \subset L$, $|L'| \leq |N(L')| - 1$

Proof:

We will prove this by strong induction on the number of vertices in L.

Base case: If there is a single vertex the statement holds trivially.

Inductive Step

Inductive Step:

Case 1: The graph is slack.

Let l be some vertex in L, we know that l has some neighbor in R, let r be one of those neighbors. Let G' be the graph G with the vertices l and r removed. Since G is slack, G' satisfies Hall's condition (any subset of vertices on the left side of G' has a number of neighbors at least as large as the set). Thus, by the inductive hypothesis, there is a perfect matching for G'. If we add to this matching the edge between l and r, we then have a perfect matching for G.

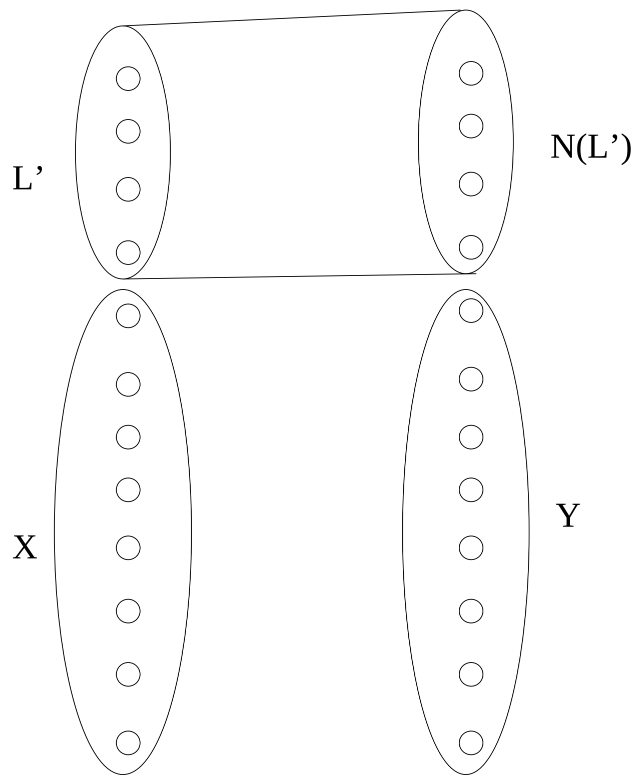
Proof of Hall's

Case 2: The graph is not slack.

In this case, there is some subset, L', of L such that $|L'| = |N(L')|$ (and $L' \neq L$). Since $|L'| < |L|$, and Hall's condition holds for the vertices in this set, we can use the inductive hypothesis to say there is a perfect matching for the vertices of L'.

Now let $X = L - L'$, and let $Y = R - N(L')$. We will now show that the condition for Hall's theorem holds for the graph (X, Y). Consider any $X' \subseteq X$. We will show that the set of neighbors of X' which are in Y is at least as large as the number of vertices in X'.

Proof of Hall's



Inductive Hypothesis

For any bipartite graph $H = (A, B)$ where $|A| < |L|$,

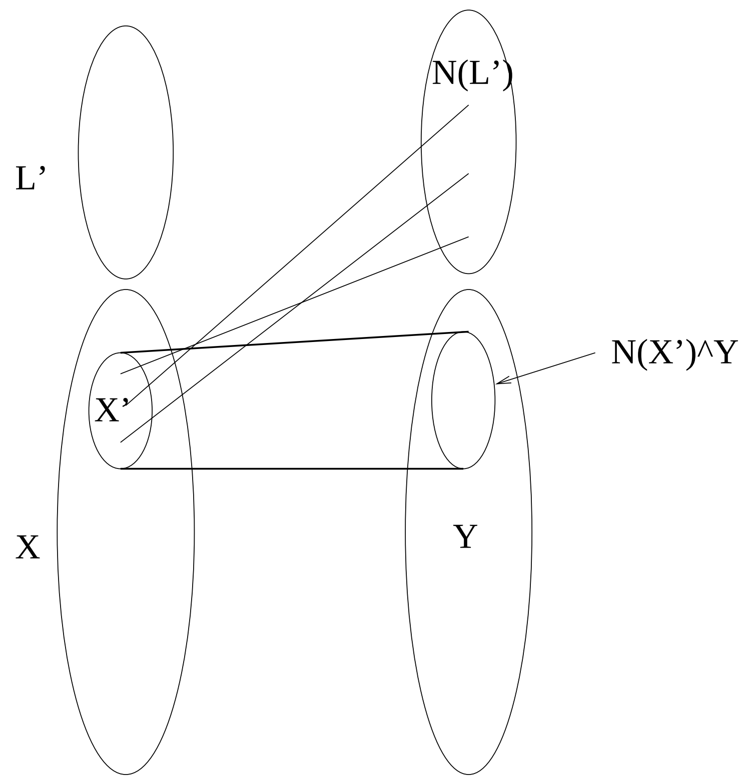
if

- $\forall A' \subseteq A, |A'| \leq |N(A')|$

then

- there is a perfect matching for H.

Proof of Hall's



Proof of Hall's

We know that for any X' which is a subset of X:

$$|X' \cup L'| \leq |N(X' \cup L')|$$

and we know that:

$$N(X' \cup L') = N(L') \cup (N(X') \cap Y)$$

We also know by assumption (since the graph was not slack) that:

$$|L'| = |N(L')|$$

Proof of Hall's

For any $X' \subseteq X$, the following holds:

$$\begin{aligned}
|X'| + |L'| &= |X' \cup L'| && (1)\\
&\leq |N(X' \cup L')| && (2)\\
&= ||N(L') \cup (N(X') \cap Y)| && (3)\\
&= |N(L')| + |(N(X') \cap Y)| && (4)\\
&= |L'| + |(N(X') \cap Y)| && (5)
\end{aligned}$$

Cancelling $|L'|$ from the first and last equation then gives that $|X'| \leq |N(X') \cap Y|$. So Hall's condition holds for the graph (X, Y), which means by the i.h. that there is a perfect matching of X which uses only vertices in Y. Combining this with the perfect matching of L' which uses vertices in $N(L')$ gives a perfect matching of all vertices in L.

Getting a Matching

- By Hall's theorem, we know that a perfect matching exists for our bipartite graph
- Q: How do we actually get this matching
- A: There is a polynomial time algorithm to do this.

Matching Algorithm

For a bipartite graph G with edge set E:

Defn: A *chain* is a sequence of vertices and edges $v_1, e_1, v_2, e_2, ..., e_l, v_l$ such that the edge $e_i = (v_{i-1}, v_i)$.

Let M be some (possibly non-maximum) matching on G.

Defn: A chain $v_1, e_1, v_2, e_2, \ldots, e_l, v_l$ is called *M-alternating* if the edges $e_2, e_4, \ldots$ are in M.

Defn: A chain $v_1, e_1, v_2, e_2, \ldots, e_l, v_l$ is called *M-augmenting* if it is M-alternating and the vertices v_1 and v_l are not incident to any edges in M.

Theorem: A Matching M is maximum for G if and only if the graph contains no M-augmenting chains.

Algorithm Find-Matching

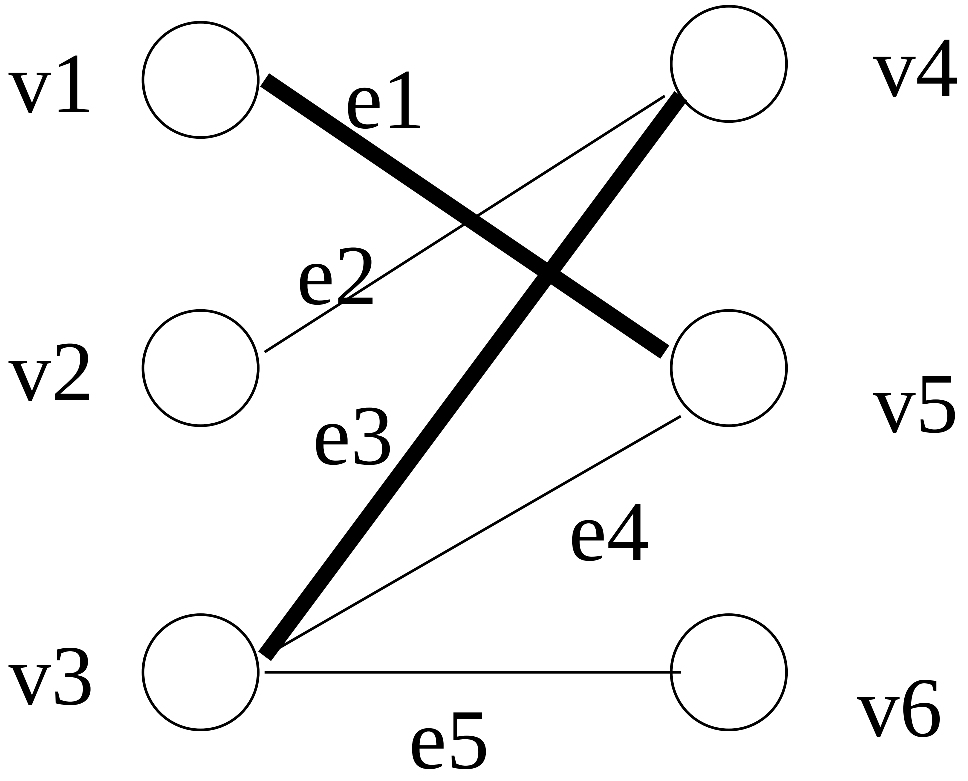
Input: A bipartite graph $G = (L, R)$

Algorithm *Find-Matching*:

1. Let $M = \phi$
2. While there is still an unmatched vertex v in L do
   (a) Search for an M-augmenting chain by doing a breadth first search from v
   (b) If a chain was found, let $A = v_1, e_1, v_2, e_2, v_3, e_3 \ldots, e_{l-1}, v_{l-1}, e_l, v_l$ be that chain. Set the new matching M to be $M \cup \{e_1, e_3, \ldots, e_l\} - \{e_2, e_4, \ldots, e_{l-1}\}$

Exercise

In the graph below, let $M = \{e1, e3\}$
Increase the size of M by finding an M-augmenting chain.
Give the M-augmenting chain and the new larger matching.



Example

-

Conclusion and Open Problems

Results:

- Approximation algorithm for NP-Hard migration problems
- For worst case inputs, results are near optimal!

Many open problems:

- Is chromatic index with space constraints $>$ chromatic index without?
- What is tradeoff between number of bypass nodes available and number of steps required?
- What can we say about migration on incomplete topologies?