CS270: Lecture 1.

1. Overview
2. Administration
3. Dueling Subroutines: Congestion/Tolls.

Algorithms.

Undergraduate.

Algorithms.

Undergraduate.

1. Classical.

Algorithms.

Undergraduate.

1. Classical.

2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.

Algorithms.

Undergraduate.

1. Classical.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions:

Algorithms.

Undergraduate.

1. Classical.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions: effective

Algorithms.

Undergraduate.

1. Classical.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions: effective precise bounds!

Algorithms.

Undergraduate.

1. Classical.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions: effective precise bounds!
4. Techniques:

Algorithms.

Undergraduate.

1. Classical.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions: effective precise bounds!
4. Techniques: Greedy

Algorithms.

Undergraduate.

1. Classical.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions: effective precise bounds!
4. Techniques: Greedy Dyn. Programming

Algorithms.

Undergraduate.

1. Classical.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions: effective precise bounds!
4. Techniques: Greedy Dyn. Programming Linear Programming.

Algorithms.

Undergraduate.

1. Classical.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions: effective precise bounds!
4. Techniques: Greedy Dyn. Programming Linear Programming.
5. Techniques tend to be Combinatorial.

Algorithms.

Undergraduate.
This class.

1. Classical.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions: effective precise bounds!
4. Techniques: Greedy Dyn. Programming Linear Programming.
5. Techniques tend to be Combinatorial.

Algorithms.

Undergraduate.
This class.

1. Classical.
   Flavor of the week?
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions: effective precise bounds!
4. Techniques: Greedy Dyn. Programming Linear Programming.
5. Techniques tend to be Combinatorial.

Algorithms.

Undergraduate.
This class.

1. Classical.
   Modern.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
3. Solutions: effective precise bounds!
4. Techniques: Greedy Dyn. Programming Linear Programming.
5. Techniques tend to be Combinatorial.

Algorithms.

Undergraduate.
This class.

1. Classical.
   Modern.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
   Vaguely stated problems!
3. Solutions: effective precise bounds!
4. Techniques: Greedy Dyn. Programming Linear Programming.
5. Techniques tend to be Combinatorial.

Algorithms.

Undergraduate.
This class.

1. Classical.
   Modern.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
   Address problems; messy or not.
3. Solutions: effective precise bounds!
4. Techniques: Greedy Dyn. Programming Linear Programming.
5. Techniques tend to be Combinatorial.

Algorithms.

Undergraduate.
This class.

1. Classical.
   Modern.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
   Address problems; messy or not.
3. Solutions: effective precise bounds!
   Ineffective ..imprecise!
4. Techniques: Greedy Dyn. Programming Linear Programming.
5. Techniques tend to be Combinatorial.

Algorithms.

Undergraduate.
This class.

1. Classical.
   Modern.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
   Address problems; messy or not.
3. Solutions: effective precise bounds!
   Analysis sometimes based on modelling world.
4. Techniques: Greedy Dyn. Programming Linear Programming.
5. Techniques tend to be Combinatorial.

Algorithms.

Undergraduate.
This class.

1. Classical.
   Modern.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
   Address problems; messy or not.
3. Solutions: effective precise bounds!
   Analysis sometimes based on modelling world.
4. Techniques: Greedy Dyn. Programming Linear Programming.
   Heuristic, in practice.
5. Techniques tend to be Combinatorial.

Algorithms.

Undergraduate.
This class.

1. Classical.
   Modern.
2. Cleanly Stated Problems. Shortest Paths, max-flow, MST.
   Address problems; messy or not.
3. Solutions: effective precise bounds!
   Analysis sometimes based on modelling world.
4. Techniques: Greedy Dyn. Programming Linear Programming.
   Heuristic, in practice.
5. Techniques tend to be Combinatorial.
   Probabilistic, linear algebra methods, continuous.

Example Problem: clustering.

- Points: documents, dna, preferences.
- Graphs: applications to VLSI, parallel processing, image segmentation.

Image example.

Image Segmentation

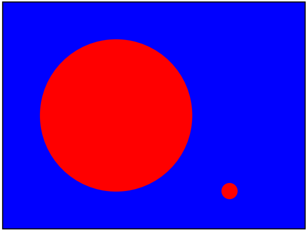
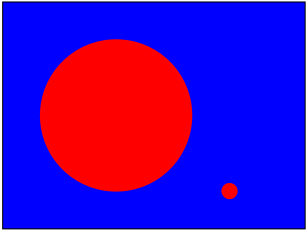


Image Segmentation



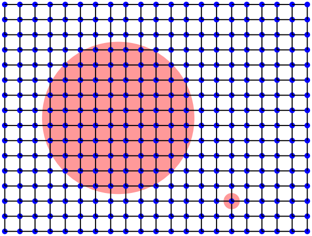
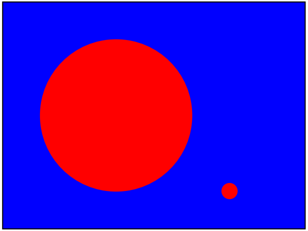
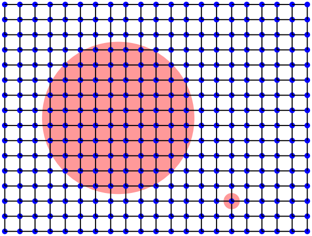


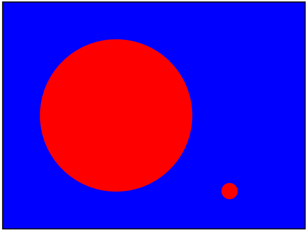
Image Segmentation

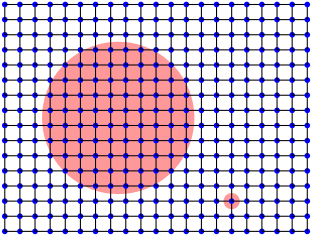




Which region?

Image Segmentation

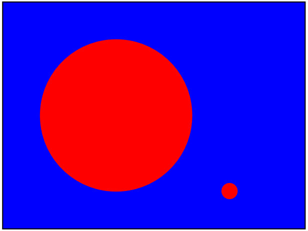


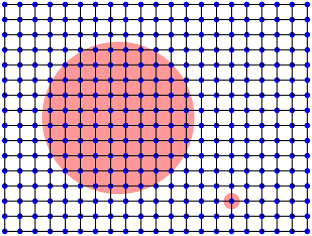


Which region? Normalized Cut: Find S, which minimizes

$$\frac{w(S,\overline{S})}{w(S) \times w(\overline{S})}.$$

Image Segmentation





Which region? Normalized Cut: Find S, which minimizes

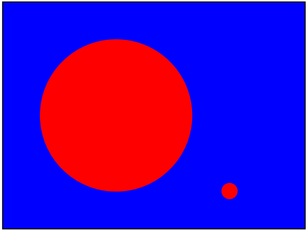
$$\frac{w(S,\overline{S})}{w(S)\times w(\overline{S})}.$$

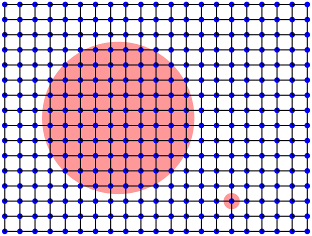
Ratio Cut: minimize

$$\frac{w(S,\overline{S})}{w(S)},$$

$w(S)$ no more than half the weight. (Minimize cost per unit weight that is removed.)

Image Segmentation





Which region? Normalized Cut: Find S, which minimizes

$$\frac{w(S,\overline{S})}{w(S) \times w(\overline{S})}.$$

Ratio Cut: minimize

$$\frac{w(S,\overline{S})}{w(S)},$$

$w(S)$ no more than half the weight. (Minimize cost per unit weight that is removed.)

Either is generally useful!

Example: recommendations.

Sarah Palin likes True Grit (the old one.)

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

What about you?

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

What about you?

Are you Hillary?

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

What about you?

Are you Hillary? Are you Sarah?

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

What about you?

Are you Hillary? Are you Sarah? A bit of both?

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

What about you?

Are you Hillary? Are you Sarah? A bit of both?

High dimensional data: dimension for each movie.

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

What about you?

Are you Hillary? Are you Sarah? A bit of both?

High dimensional data: dimension for each movie.
More than three dimensions!

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

What about you?

Are you Hillary? Are you Sarah? A bit of both?

High dimensional data: dimension for each movie.
More than three dimensions!
Nearest neighbors.

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

What about you?

Are you Hillary? Are you Sarah? A bit of both?

High dimensional data: dimension for each movie.
More than three dimensions!
Nearest neighbors. Principal Components methods.

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

What about you?

Are you Hillary? Are you Sarah? A bit of both?

High dimensional data: dimension for each movie.
More than three dimensions!
Nearest neighbors. Principal Components methods.
Topic Models.

Example: recommendations.

Sarah Palin likes True Grit (the old one.)
Sarah Palin doesn't like The Social Network.
Sarah Palin doesn't like Black Swan.
Sarah Palin likes Sarah Palin on Discovery channel.

Hillary Clinton doesn't like True Grit (the old one.)
Hillary Clinton likes The Social Network.
Hillary Clinton likes Black Swan.

Should you recommend the discovery channel to Hillary?

What about you?

Are you Hillary? Are you Sarah? A bit of both?

High dimensional data: dimension for each movie.
More than three dimensions!
Nearest neighbors. Principal Components methods.
Topic Models.
Reasoning about these methods.

Linear Systems.

Revolution!

Linear Systems.

Revolution!

Linear Systems.

Revolution!

Linear Systems.

Revolution!

Linear Systems.

Revolution!

Linear Systems.

Revolution!

Physical Simulation.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School:

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution,

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Now: $\tilde{O}(m)$.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Now: $\tilde{O}(m)$. Hmmm.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Now: $\tilde{O}(m)$. Hmmm. What's that tilde?

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Now: $\tilde{O}(m)$. Hmmm. What's that tilde?

Techniques:

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Now: $\tilde{O}(m)$. Hmmm. What's that tilde?

Techniques:
Relate graph theory to matrix properties.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Now: $\tilde{O}(m)$. Hmmm. What's that tilde?

Techniques:
Relate graph theory to matrix properties.
Dense matrix (graph) to sparse matrix (graph).

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Now: $\tilde{O}(m)$. Hmmm. What's that tilde?

Techniques:
Relate graph theory to matrix properties.
Dense matrix (graph) to sparse matrix (graph).
Approximating distances by trees.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Now: $\tilde{O}(m)$. Hmmm. What's that tilde?

Techniques:
Relate graph theory to matrix properties.
Dense matrix (graph) to sparse matrix (graph).
Approximating distances by trees.
Electrical networks analysis.

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Now: $\tilde{O}(m)$. Hmmm. What's that tilde?

Techniques:
Relate graph theory to matrix properties.
Dense matrix (graph) to sparse matrix (graph).
Approximating distances by trees.
Electrical networks analysis.

Combinatorial Applications:

Linear Systems.

Revolution!

Physical Simulation. Airflow.

Solve $Ax = b$.

How long?

$n \times n$ matrix A.

Middle School: substitution, adding equations ...
Time: $O(n^3)$.

Now: $\tilde{O}(m)$. Hmmm. What's that tilde?

Techniques:
Relate graph theory to matrix properties.
Dense matrix (graph) to sparse matrix (graph).
Approximating distances by trees.
Electrical networks analysis.

Combinatorial Applications: Better Max Flow!

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.

Graphs: Sparse graph.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.

Graphs: Sparse graph.

Data: average, statistics.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.

Graphs: Sparse graph.

Data: average, statistics.

Points: center point, k-medians, .

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.

Graphs: Sparse graph.

Data: average, statistics.

Points: center point, k-medians, .

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.

Graphs: Sparse graph.

Data: average, statistics.

Points: center point, k-medians, .

High Dimensional optimization.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.
Graphs: Sparse graph.
Data: average, statistics.
Points: center point, k-medians, .

High Dimensional optimization.

Gradient Descent. Convexity.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.

Graphs: Sparse graph.

Data: average, statistics.

Points: center point, k-medians, .

High Dimensional optimization.

Gradient Descent. Convexity.

Linear Algebra.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.

Graphs: Sparse graph.

Data: average, statistics.

Points: center point, k-medians, .

High Dimensional optimization.

Gradient Descent. Convexity.

Linear Algebra.

Eigenvalues.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.
Graphs: Sparse graph.
Data: average, statistics.
Points: center point, k-medians, .

High Dimensional optimization.

Gradient Descent. Convexity.

Linear Algebra.

Eigenvalues.
Semidefinite Programming.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.
Graphs: Sparse graph.
Data: average, statistics.
Points: center point, k-medians, .

High Dimensional optimization.

Gradient Descent. Convexity.

Linear Algebra.

Eigenvalues.
Semidefinite Programming.

Dueling Subroutines. Duality.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.
Graphs: Sparse graph.
Data: average, statistics.
Points: center point, k-medians, .

High Dimensional optimization.

Gradient Descent. Convexity.

Linear Algebra.

Eigenvalues.
Semidefinite Programming.

Dueling Subroutines. Duality.

Lower bounder, upper bounder.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.
Graphs: Sparse graph.
Data: average, statistics.
Points: center point, k-medians, .

High Dimensional optimization.

Gradient Descent. Convexity.

Linear Algebra.

Eigenvalues.
Semidefinite Programming.

Dueling Subroutines. Duality.

Lower bounder, upper bounder.
Upper uses lower's evidence to find better solutions.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.
Graphs: Sparse graph.
Data: average, statistics.
Points: center point, k-medians, .

High Dimensional optimization.

Gradient Descent. Convexity.

Linear Algebra.

Eigenvalues.
Semidefinite Programming.

Dueling Subroutines. Duality.

Lower bounder, upper bounder.
Upper uses lower's evidence to find better solutions.
Lower uses upper's evidence to prove better lower bounds.

Other Algorithmic Techiniques

Sketching:

Large stream of data: $a_1, a_2, \ldots$

Find digest.
Graphs: Sparse graph.
Data: average, statistics.
Points: center point, k-medians, .

High Dimensional optimization.

Gradient Descent. Convexity.

Linear Algebra.

Eigenvalues.
Semidefinite Programming.

Dueling Subroutines. Duality.

Lower bounder, upper bounder.
Upper uses lower's evidence to find better solutions.
Lower uses upper's evidence to prove better lower bounds.

CS270: Administration.

1. Staff:
   Satish Rao

CS270: Administration.

1. Staff:
   Satish Rao
   Benjamin Weitz
2. Piazza. Log in! Pay attention to “bypass email preferences” especially.
3. Assessment.
   3.1 Homeworks (40%).

CS270: Administration.

1. Staff:
   Satish Rao
   Benjamin Weitz
2. Piazza. Log in! Pay attention to "bypass email preferences" especially.
3. Assessment.
   3.1 Homeworks (40%).
       Homework 1 out tonight/tomorrow.

CS270: Administration.

1. Staff:
   Satish Rao
   Benjamin Weitz
2. Piazza. Log in! Pay attention to “bypass email preferences” especially.
3. Assessment.
   3.1 Homeworks (40%).
   Homework 1 out tonight/tomorrow.
   3.2 1 Takehome Midterm (25 %)

CS270: Administration.

1. Staff:
   Satish Rao
   Benjamin Weitz
2. Piazza. Log in! Pay attention to “bypass email preferences” especially.
3. Assessment.
   3.1 Homeworks (40%).
       Homework 1 out tonight/tomorrow.
   3.2 1 Takehome Midterm (25 %)
   3.3 Project (35%)

CS270: Administration.

1. Staff:
   Satish Rao
   Benjamin Weitz
2. Piazza. Log in! Pay attention to “bypass email preferences” especially.
3. Assessment.
   3.1 Homeworks (40%).
       Homework 1 out tonight/tomorrow.
   3.2 1 Takehome Midterm (25 %)
   3.3 Project (35%)
       Groups of 2 or 3.

CS270: Administration.

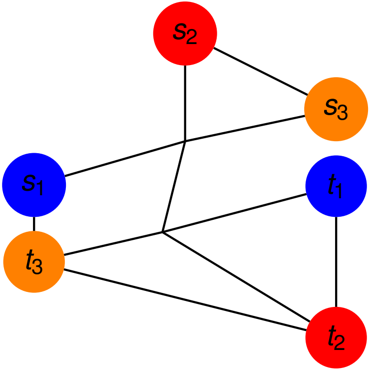
1. Staff:
   Satish Rao
   Benjamin Weitz
2. Piazza. Log in! Pay attention to “bypass email preferences” especially.
3. Assessment.
   3.1 Homeworks (40%).
   Homework 1 out tonight/tomorrow.
   3.2 1 Takehome Midterm (25 %)
   3.3 Project (35%)
   Groups of 2 or 3.
   Connect research to class.

CS270: Administration.

1. Staff:
   Satish Rao
   Benjamin Weitz
2. Piazza. Log in! Pay attention to “bypass email preferences” especially.
3. Assessment.
   3.1 Homeworks (40%).
   Homework 1 out tonight/tomorrow.
   3.2 1 Takehome Midterm (25 %)
   3.3 Project (35%)
   Groups of 2 or 3.
   Connect research to class.
   Or explore/digest a topic from class.
   3.4 No Discussion this week.

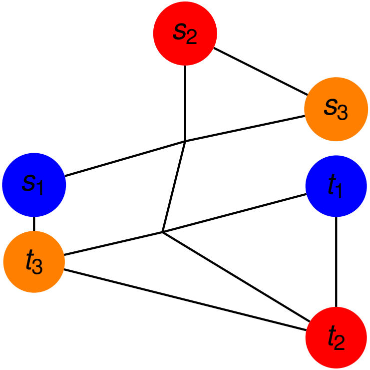
Path Routing.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths connecting s_i and t_i and minimize max load on any edge.



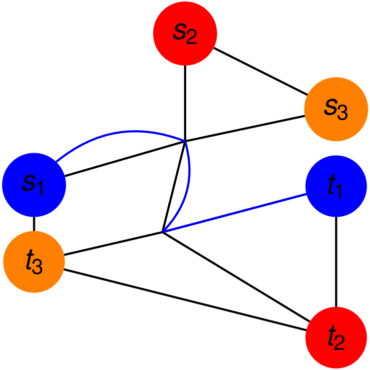
Path Routing.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths connecting s_i and t_i and minimize max load on any edge.



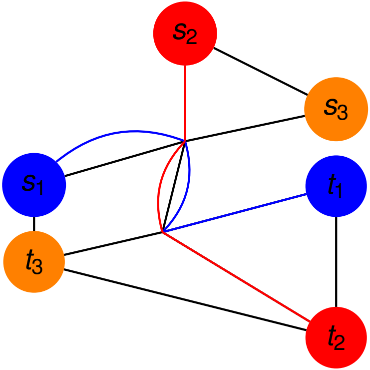
Path Routing.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths connecting s_i and t_i and minimize max load on any edge.



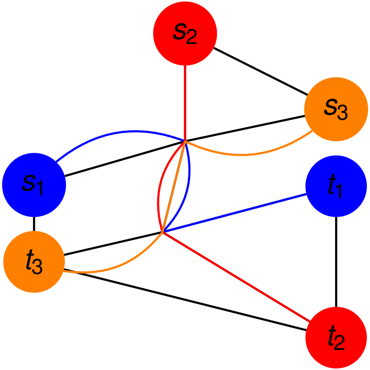
Path Routing.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths connecting s_i and t_i and minimize max load on any edge.



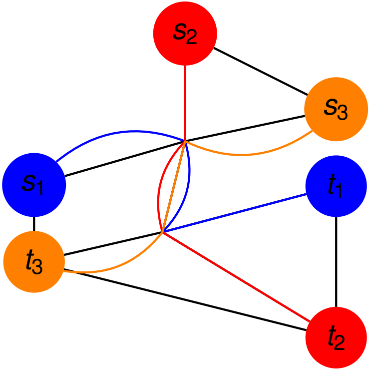
Path Routing.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths connecting s_i and t_i and minimize max load on any edge.



Path Routing.

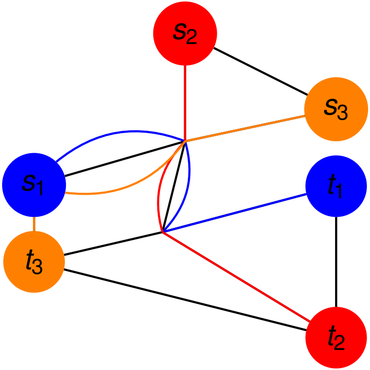
Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths connecting s_i and t_i and minimize max load on any edge.



Value: 3

Path Routing.

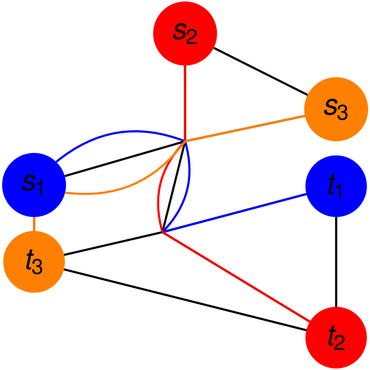
Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths connecting s_i and t_i and minimize max load on any edge.



Value: 3

Path Routing.

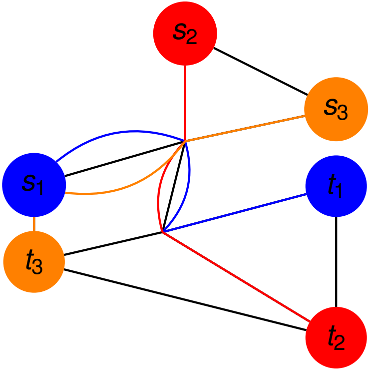
Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths connecting s_i and t_i and minimize max load on any edge.



~~Value: 3~~

Path Routing.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths connecting s_i and t_i and minimize max load on any edge.



~~Value: 3~~

Value: 2

Terminology

Routing: Paths $p_1, p_2, \ldots, p_k$, p_i connects s_i and t_i.

Terminology

Routing: Paths $p_1, p_2, \ldots, p_k$, p_i connects s_i and t_i.

Congestion of edge, e: $c(e)$

Terminology

Routing: Paths $p_1, p_2, \ldots, p_k$, p_i connects s_i and t_i.

Congestion of edge, e: $c(e)$

number of paths in routing that contain e.

Terminology

Routing: Paths $p_1, p_2, \ldots, p_k$, p_i connects s_i and t_i.

Congestion of edge, e: $c(e)$
number of paths in routing that contain e.

Congestion of routing:

Terminology

Routing: Paths $p_1, p_2, \ldots, p_k$, p_i connects s_i and t_i.

Congestion of edge, e: $c(e)$
number of paths in routing that contain e.

Congestion of routing: maximum congestion of any edge.

Terminology

Routing: Paths $p_1, p_2, \ldots, p_k$, p_i connects s_i and t_i.

Congestion of edge, e: $c(e)$
number of paths in routing that contain e.

Congestion of routing: maximum congestion of any edge.

Find routing that minimizes congestion (or maximum congestion.)

Algorithms?

Route along any path.

Algorithms?

Route along any path.

Feasible...

Algorithms?

Route along any path.

Feasible...but is it as good as possible?

Algorithms?

Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

Algorithms?

Route along any path.

Feasible...but is it as good as possible?

How far from optimal could it be?

(A) It is optimal!

Algorithms?

Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

Algorithms?

Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

Algorithms?

Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)

Algorithms?

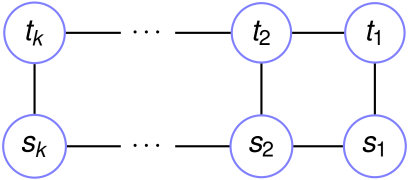
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Algorithms?

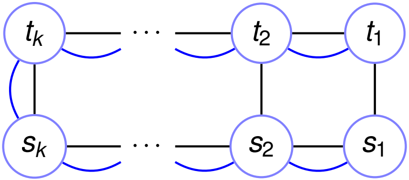
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Algorithms?

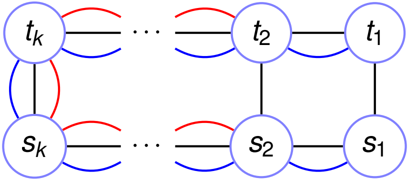
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Algorithms?

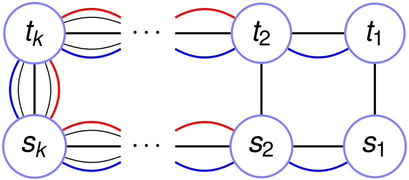
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Algorithms?

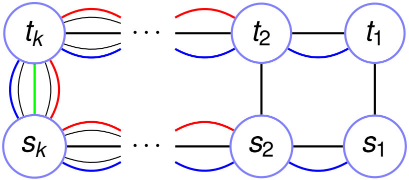
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Algorithms?

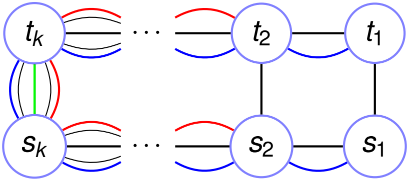
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Value: k.

Algorithms?

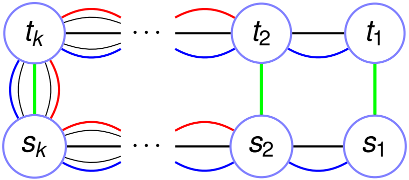
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Value: k.

Opt: 1.

Algorithms?

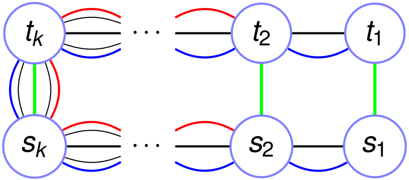
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Value: k.

Opt: 1.

Stupid..

Algorithms?

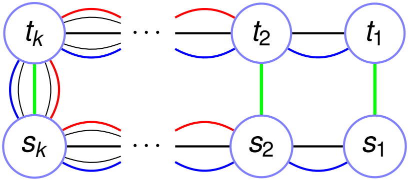
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Value: k.

Opt: 1.

Stupid..but this could be depth first search lexicographically!

Algorithms?

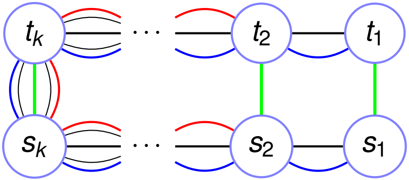
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Value: k.

Opt: 1.

Stupid..but this could be depth first search lexicographically!
Route along shortest path!

Algorithms?

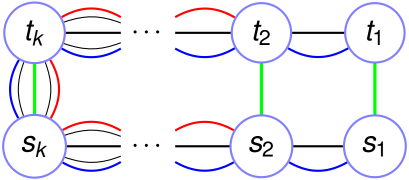
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Value: k.

Opt: 1.

Stupid..but this could be depth first search lexicographically!
Route along shortest path! Duh.

Algorithms?

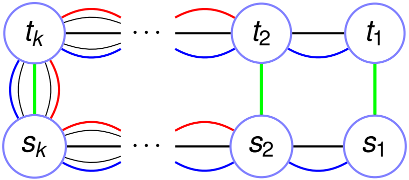
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Value: k.

Opt: 1.

Stupid..but this could be depth first search lexicographically!
Route along shortest path! Duh.
Optimal use of "resources"

Algorithms?

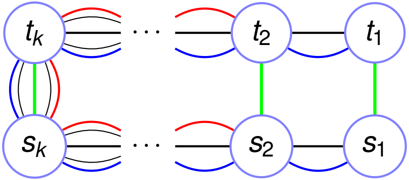
Route along any path.
Feasible...but is it as good as possible?
How far from optimal could it be?

(A) It is optimal!

(B) A factor of two.

(C) A factor of k, in general.

(C)



Value: k.

Opt: 1.

Stupid..but this could be depth first search lexicographically!
Route along shortest path! Duh.
Optimal use of “resources” ..or edges.

Shortest Path Routing.

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

- s_i, t_i routed along p_i
- $\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

- where $c(e)$ congestion of edge.

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why?

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?

(B) $\sum_i \ell(p_i) > \sum_e c(e)$?

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A).

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges.

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges. Edge e used by $c(e)$ paths.

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges. Edge e used by $c(e)$ paths.

Both count "uses"

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges. Edge e used by $c(e)$ paths.

Both count "uses" $\rightarrow$ Sums are the same.

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges. Edge e used by $c(e)$ paths.

Both count "uses" $\rightarrow$ Sums are the same.

$\sum_i \ell(p_i)$

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges. Edge e used by $c(e)$ paths.

Both count "uses" $\rightarrow$ Sums are the same.

$\sum_i \ell(p_i) = \sum_i \sum_{e \in p_i} 1$

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges. Edge e used by $c(e)$ paths.

Both count "uses" $\rightarrow$ Sums are the same.

$\sum_i \ell(p_i) = \sum_i \sum_{e \in p_i} 1 = \sum_e \sum_{p_i \ni e} 1$

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges. Edge e used by $c(e)$ paths.

Both count "uses" $\rightarrow$ Sums are the same.

$\sum_i \ell(p_i) = \sum_i \sum_{e \in p_i} 1 = \sum_e \sum_{p_i \ni e} 1 = \sum_e c(e)$

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges. Edge e used by $c(e)$ paths.

Both count "uses" $\rightarrow$ Sums are the same.

$\sum_i \ell(p_i) = \sum_i \sum_{e \in p_i} 1 = \sum_e \sum_{p_i \ni e} 1 = \sum_e c(e)$

Shortest path routing minimizes total congestion

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges. Edge e used by $c(e)$ paths.

Both count "uses" $\rightarrow$ Sums are the same.

$\sum_i \ell(p_i) = \sum_i \sum_{e \in p_i} 1 = \sum_e \sum_{p_i \ni e} 1 = \sum_e c(e)$

Shortest path routing minimizes total congestion !

Shortest Path Routing.

Minimizes $\sum_i \ell(p_i)$.

s_i, t_i routed along p_i

$\ell(p_i)$ is number of edges in p_i.

Also minimizes total congestion: $\sum_e c(e)$

where $c(e)$ congestion of edge.

Why? Let $\ell(p_i)$ be the length of path p_i.

(A) $\sum_i \ell(p_i) = \sum_e c(e)$?
(B) $\sum_i \ell(p_i) > \sum_e c(e)$?
(C) $\sum_i \ell(p_i) < \sum_e c(e)$?

(A). Proof?

Path i uses $\ell(p_i)$ edges. Edge e used by $c(e)$ paths.

Both count "uses" $\rightarrow$ Sums are the same.

$\sum_i \ell(p_i) = \sum_i \sum_{e \in p_i} 1 = \sum_e \sum_{p_i \ni e} 1 = \sum_e c(e)$

Shortest path routing minimizes total congestion ! !

Shortest Path Routing and Congestion.

Minimize each path length minimizes total congestion.

Shortest Path Routing and Congestion.

Minimize each path length minimizes total congestion.

Also minimizes average: $\frac{1}{m}\sum_e c(e)$.

Shortest Path Routing and Congestion.

Minimize each path length minimizes total congestion.

Also minimizes average: $\frac{1}{m}\sum_e c(e)$. Just a scaling!

Shortest Path Routing and Congestion.

Minimize each path length minimizes total congestion.

Also minimizes average: $\frac{1}{m}\sum_e c(e)$. Just a scaling!

Average load is lower bound on the lowest max congestion!

Shortest Path Routing and Congestion.

Minimize each path length minimizes total congestion.

Also minimizes average: $\frac{1}{m}\sum_e c(e)$. Just a scaling!

Average load is lower bound on the lowest max congestion!

Shortest path routing minimizes average load.

Shortest Path Routing and Congestion.

Minimize each path length minimizes total congestion.

Also minimizes average: $\frac{1}{m}\sum_e c(e)$. Just a scaling!

Average load is lower bound on the lowest max congestion!

Shortest path routing minimizes average load.

Does it minimize maximum load?

One problem...

How far from optimal?

One problem...

How far from optimal?

Optimal?

One problem...

How far from optimal?

Optimal? Factor 2?

One problem...

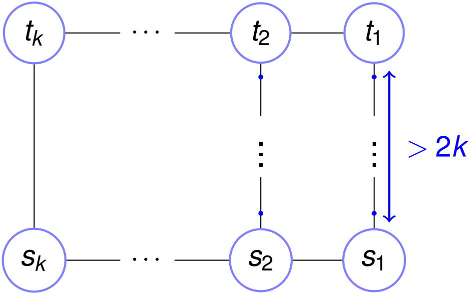
How far from optimal?

Optimal? Factor 2? Factor k?

One problem...

How far from optimal?

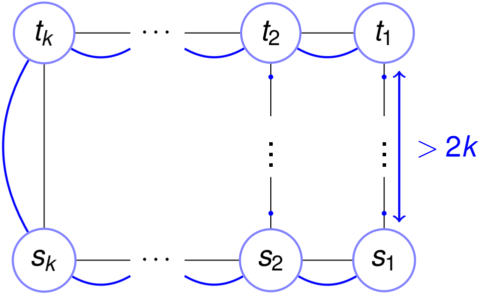
Optimal? Factor 2? Factor k?



One problem...

How far from optimal?

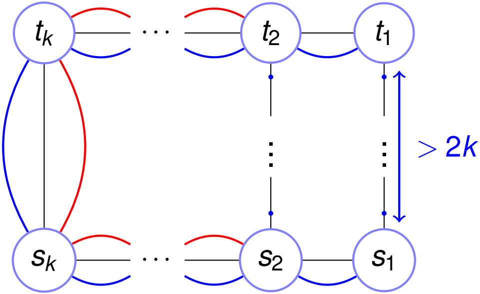
Optimal? Factor 2? Factor k?



One problem...

How far from optimal?

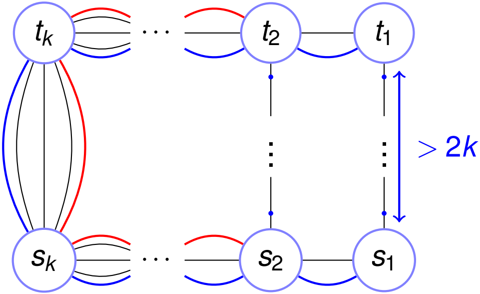
Optimal? Factor 2? Factor k?



One problem...

How far from optimal?

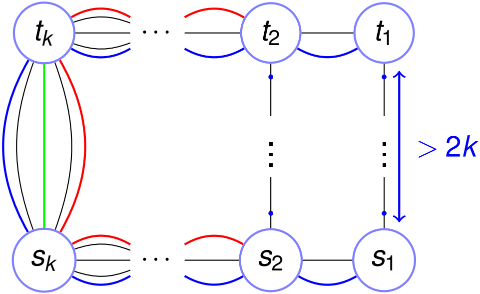
Optimal? Factor 2? Factor k?



One problem...

How far from optimal?

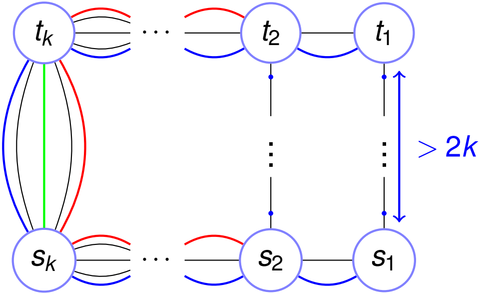
Optimal? Factor 2? Factor *k*?



One problem...

How far from optimal?

Optimal? Factor 2? Factor k?

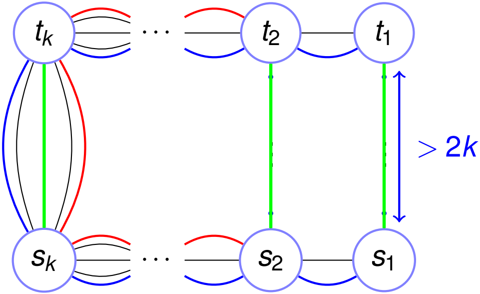


Value: k.

One problem...

How far from optimal?

Optimal? Factor 2? Factor k?



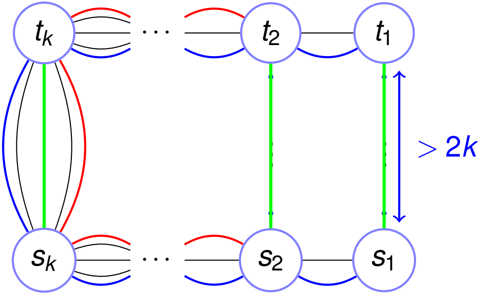
Value: k.

Opt: 1.

One problem...

How far from optimal?

Optimal? Factor 2? Factor k?



Value: k.

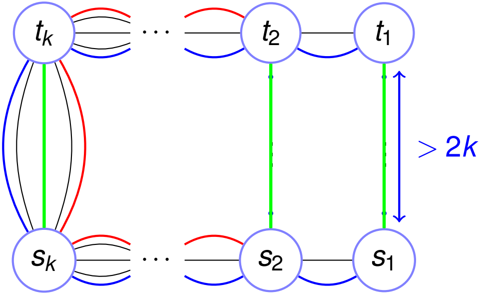
Opt: 1.

Does minimize average load though,

One problem...

How far from optimal?

Optimal? Factor 2? Factor k?



Value: k.

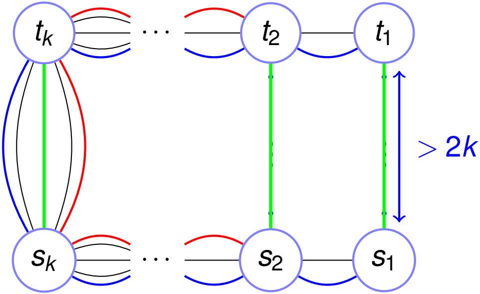
Opt: 1.

Does minimize average load though, FWIW.

One problem...

How far from optimal?

Optimal? Factor 2? Factor k?



Value: k.

Opt: 1.

Does minimize average load though, FWIW.

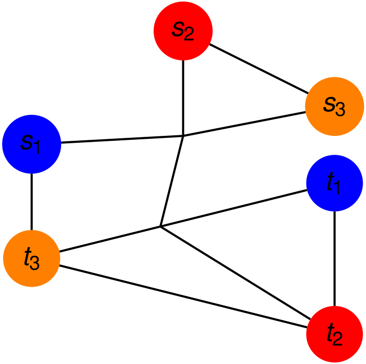
Any suggestions?

Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.

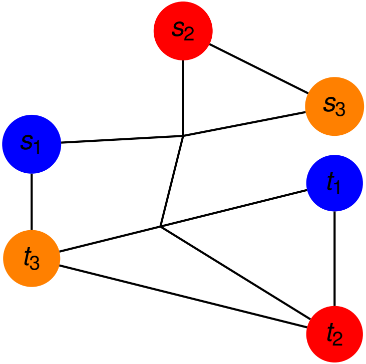
Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of “toll” to edges to maximize total toll for connecting pairs.



Another problem.

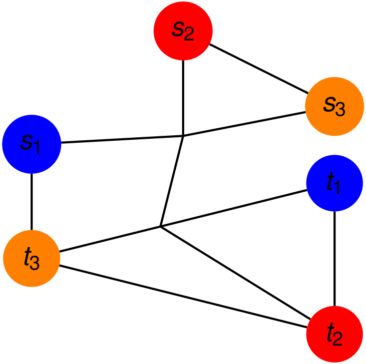
Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.



Assign $\frac{1}{11}$ on each of 11 edges.

Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.

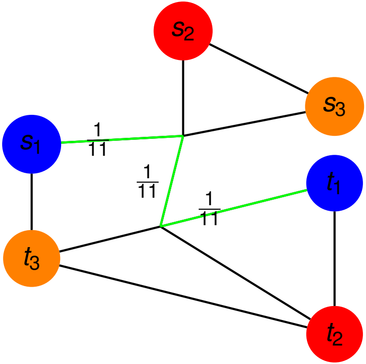


Assign $\frac{1}{11}$ on each of 11 edges.

Toll paid:

Another problem.

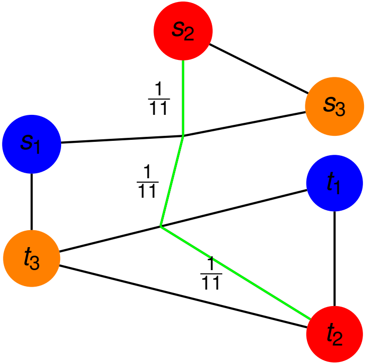
Given $G = (V,E)$, $(s_1,t_1),\ldots,(s_k,t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.



Assign $\frac{1}{11}$ on each of 11 edges.
Toll paid: $\frac{3}{11}$

Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.

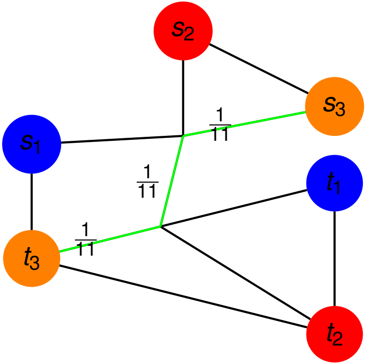


Assign $\frac{1}{11}$ on each of 11 edges.

Toll paid: $\frac{3}{11} \quad + \frac{3}{11}$

Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of “toll” to edges to maximize total toll for connecting pairs.

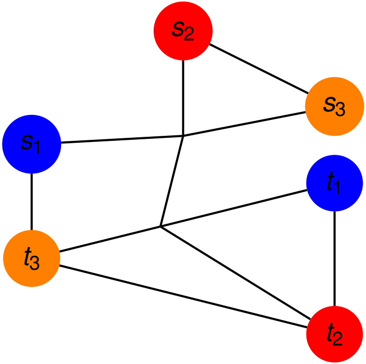


Assign $\frac{1}{11}$ on each of 11 edges.

Toll paid: $\frac{3}{11} \quad + \frac{3}{11} \ + \frac{3}{11}$

Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.

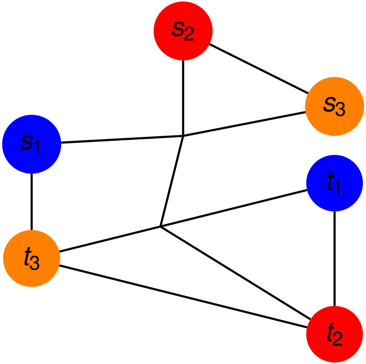


Assign $\frac{1}{11}$ on each of 11 edges.

Toll paid: $\frac{3}{11} \quad + \frac{3}{11} \ + \frac{3}{11} = \frac{9}{11}$

Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.



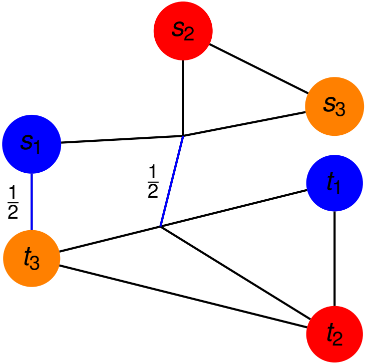
Assign $\frac{1}{11}$ on each of 11 edges.

Toll paid: $\frac{3}{11} + \frac{3}{11} + \frac{3}{11} = \frac{9}{11}$

Can we do better?

Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.



Assign $\frac{1}{11}$ on each of 11 edges.

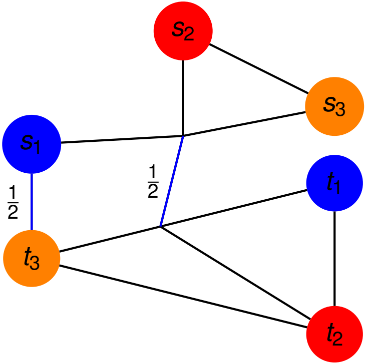
Toll paid: $\frac{3}{11} + \frac{3}{11} + \frac{3}{11} = \frac{9}{11}$

Can we do better?

Assign $1/2$ on these two edges.

Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.



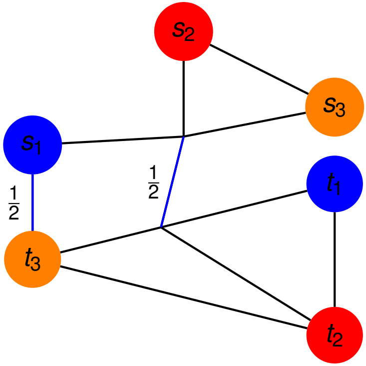
Assign $\frac{1}{11}$ on each of 11 edges.
Toll paid: $\frac{3}{11} + \frac{3}{11} + \frac{3}{11} = \frac{9}{11}$

Can we do better?

Assign $1/2$ on these two edges.
Toll paid:

Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.



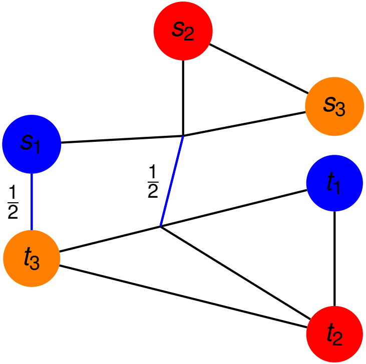
Assign $\frac{1}{11}$ on each of 11 edges.
Toll paid: $\frac{3}{11} \quad + \frac{3}{11} \; + \frac{3}{11} = \frac{9}{11}$

Can we do better?

Assign $1/2$ on these two edges.
Toll paid: $\frac{1}{2} + \frac{1}{2} + \frac{1}{2}$

Another problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll for connecting pairs.



Assign $\frac{1}{11}$ on each of 11 edges.
Toll paid: $\frac{3}{11} + \frac{3}{11} + \frac{3}{11} = \frac{9}{11}$

Can we do better?

Assign $1/2$ on these two edges.
Toll paid: $\frac{1}{2} + \frac{1}{2} + \frac{1}{2} = \frac{3}{2}$

Toll problem and Routing problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll paid to connecting pairs.

Toll problem and Routing problem.

Given $G=(V,E)$, $(s_1,t_1),\ldots,(s_k,t_k)$, find a set of k paths assign one unit of “toll” to edges to maximize total toll paid to connecting pairs.

Possible solution: $\frac{1}{m}$ on each edge.

Toll problem and Routing problem.

Given $G=(V,E)$, $(s_1,t_1),\ldots,(s_k,t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll paid to connecting pairs.

Possible solution: $\frac{1}{m}$ on each edge.

Toll collected: $\geq \frac{\sum_i \ell(p_i)}{m}$.

Toll problem and Routing problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll paid to connecting pairs.

Possible solution: $\frac{1}{m}$ on each edge.

Toll collected: $\geq \frac{\sum_i \ell(p_i)}{m}$.

Familiar?

Toll problem and Routing problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll paid to connecting pairs.

Possible solution: $\frac{1}{m}$ on each edge.

Toll collected: $\geq \frac{\sum_i \ell(p_i)}{m}$.

Familiar? Consider.

Toll problem and Routing problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll paid to connecting pairs.

Possible solution: $\frac{1}{m}$ on each edge.

Toll collected: $\geq \frac{\sum_i \ell(p_i)}{m}$.

Familiar? Consider.

Find $d : e \to R$ with $\sum_e d(e) = 1$ which maximizes

$$\sum_i d(s_i, t_i).$$

$d(s_i, t_i)$- shortest path between s_i and t_i under $d(\cdot)$.

Toll problem and Routing problem.

Given $G = (V, E)$, $(s_1, t_1), \ldots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll paid to connecting pairs.

Possible solution: $\frac{1}{m}$ on each edge.

Toll collected: $\geq \frac{\sum_i \ell(p_i)}{m}$.

Familiar? Consider.

Find $d : e \rightarrow R$ with $\sum_e d(e) = 1$ which maximizes

$$\sum_i d(s_i, t_i).$$

$d(s_i, t_i)$- shortest path between s_i and t_i under $d(\cdot)$.

Digression?

Toll problem and Routing problem.

Given $G = (V, E)$, $(s_1, t_1), \dots, (s_k, t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll paid to connecting pairs.

Possible solution: $\frac{1}{m}$ on each edge.

Toll collected: $\geq \frac{\sum_i \ell(p_i)}{m}$.

Familiar? Consider.

Find $d : e \to R$ with $\sum_e d(e) = 1$ which maximizes

$$\sum_i d(s_i, t_i).$$

$d(s_i, t_i)$- shortest path between s_i and t_i under $d(\cdot)$.

Digression? $d(e)$ suggests a weighted average.

Toll problem and Routing problem.

Given $G=(V,E)$, $(s_1,t_1),\ldots,(s_k,t_k)$, find a set of k paths assign one unit of “toll” to edges to maximize total toll paid to connecting pairs.

Possible solution: $\frac{1}{m}$ on each edge.

Toll collected: $\geq \frac{\sum_i \ell(p_i)}{m}$.

Familiar? Consider.

Find $d : e \rightarrow R$ with $\sum_e d(e) = 1$ which maximizes

$$\sum_i d(s_i,t_i).$$

$d(s_i,t_i)$- shortest path between s_i and t_i under $d(\cdot)$.

Digression? $d(e)$ suggests a weighted average.

Remember uniform average congestion is lower bound on congestion of routing!

Toll problem and Routing problem.

Given $G=(V,E)$, $(s_1,t_1),\ldots,(s_k,t_k)$, find a set of k paths assign one unit of "toll" to edges to maximize total toll paid to connecting pairs.

Possible solution: $\frac{1}{m}$ on each edge.

Toll collected: $\geq \frac{\sum_i \ell(p_i)}{m}$.

Familiar? Consider.

Find $d: e \rightarrow R$ with $\sum_e d(e) = 1$ which maximizes

$$\sum_i d(s_i,t_i).$$

$d(s_i,t_i)$- shortest path between s_i and t_i under $d(\cdot)$.

Digression? $d(e)$ suggests a weighted average.

Remember uniform average congestion is lower bound on congestion of routing!

Any toll solution value (weighted average congestion) is lower bound on path routing value (max congestion).

Proving lower bound: notation.

$d(e)$ - toll assigned to edge e.

Proving lower bound: notation.

$d(e)$ - toll assigned to edge e.
$d(p)$ - total toll assigned to path p.

Proving lower bound: notation.

$d(e)$ - toll assigned to edge e.
$d(p)$ - total toll assigned to path p.
$d(u, v)$ - total assigned to shortest path between u and v.

Proving lower bound: notation.

$d(e)$ - toll assigned to edge e.
$d(p)$ - total toll assigned to path p.
$d(u, v)$ - total assigned to shortest path between u and v.
$d(x)$ - polymorpic

Proving lower bound: notation.

$d(e)$ - toll assigned to edge e.
$d(p)$ - total toll assigned to path p.
$d(u, v)$ - total assigned to shortest path between u and v.
$d(x)$ - polymorpic
x could be edge, path, or pair.

Proving lower bound: notation.

$d(e)$ - toll assigned to edge e.
$d(p)$ - total toll assigned to path p.
$d(u, v)$ - total assigned to shortest path between u and v.
$d(x)$ - ~~polymorpic~~
x could be edge, path, or pair.

Proving lower bound: notation.

$d(e)$ - toll assigned to edge e.
$d(p)$ - total toll assigned to path p.
$d(u, v)$ - total assigned to shortest path between u and v.
$d(x)$ - ~~polymorpic~~ polymorphic
x could be edge, path, or pair.

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\sum_i d(p_i) \quad = \quad \sum_i \sum_{e \in p_i} d(e)$$

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\begin{aligned} \sum_i d(p_i) &= \sum_i \sum_{e \in p_i} d(e) \\ &= \sum_e \sum_{i : e \ni p_i} d(e) \end{aligned}$$

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\begin{aligned}
\sum_i d(p_i) &= \sum_i \sum_{e \in p_i} d(e) \\
&= \sum_e \sum_{i : e \ni p_i} d(e) \\
&= \sum_e d(e) \sum_{i : e \ni p_i} 1
\end{aligned}$$

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\begin{aligned}
\sum_i d(p_i) &= \sum_i \sum_{e \in p_i} d(e) \\
&= \sum_e \sum_{i : e \ni p_i} d(e) \\
&= \sum_e d(e) \sum_{i : e \ni p_i} 1 \\
&= \sum_e d(e)c(e)
\end{aligned}$$

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\begin{aligned}
\sum_i d(p_i) &= \sum_i \sum_{e \in p_i} d(e) \\
&= \sum_e \sum_{i : e \ni p_i} d(e) \\
&= \sum_e d(e) \sum_{i : e \ni p_i} 1 \\
&= \sum_e d(e)c(e)
\end{aligned}$$

A path uses "volume" $d(p_i)$.

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\begin{aligned}
\sum_i d(p_i) &= \sum_i \sum_{e \in p_i} d(e) \\
&= \sum_e \sum_{i : e \ni p_i} d(e) \\
&= \sum_e d(e) \sum_{i : e \ni p_i} 1 \\
&= \sum_e d(e)c(e)
\end{aligned}$$

A path uses "volume" $d(p_i)$.

Volume on edge is $d(e)c(e)$.

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\begin{aligned}
\sum_i d(p_i) &= \sum_i \sum_{e \in p_i} d(e) \\
&= \sum_e \sum_{i: e \ni p_i} d(e) \\
&= \sum_e d(e) \sum_{i: e \ni p_i} 1 \\
&= \sum_e d(e)c(e)
\end{aligned}$$

A path uses "volume" $d(p_i)$.

Volume on edge is $d(e)c(e)$.

$\sum_i d(p_i) = \sum_e d(e)c(e)$.

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\begin{aligned}
\sum_i d(p_i) &= \sum_i \sum_{e \in p_i} d(e) \\
&= \sum_e \sum_{i: e \ni p_i} d(e) \\
&= \sum_e d(e) \sum_{i: e \ni p_i} 1 \\
&= \sum_e d(e)c(e)
\end{aligned}$$

A path uses "volume" $d(p_i)$.

Volume on edge is $d(e)c(e)$.

$\sum_i d(p_i) = \sum_e d(e)c(e)$.

$\max_e c(e) \geq \sum_e d(e)c(e)$

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\begin{aligned}
\sum_i d(p_i) &= \sum_i \sum_{e \in p_i} d(e) \\
&= \sum_e \sum_{i : e \ni p_i} d(e) \\
&= \sum_e d(e) \sum_{i : e \ni p_i} 1 \\
&= \sum_e d(e)c(e)
\end{aligned}$$

A path uses "volume" $d(p_i)$.

Volume on edge is $d(e)c(e)$.

$\sum_i d(p_i) = \sum_e d(e)c(e)$.

$\max_e c(e) \geq \sum_e d(e)c(e) = \sum_i d(p_i)$

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\begin{aligned}
\sum_i d(p_i) &= \sum_i \sum_{e \in p_i} d(e) \\
&= \sum_e \sum_{i : e \ni p_i} d(e) \\
&= \sum_e d(e) \sum_{i : e \ni p_i} 1 \\
&= \sum_e d(e)c(e)
\end{aligned}$$

A path uses "volume" $d(p_i)$.

Volume on edge is $d(e)c(e)$.

$\sum_i d(p_i) = \sum_e d(e)c(e)$.

$\max_e c(e) \geq \sum_e d(e)c(e) = \sum_i d(p_i) \geq \sum_i d(s_i, t_i)$.

Proving lower bound.

Routing solution: p_i connects (s_i, t_i) and has length $d(p_i)$.

$c(e)$ - congestion on edge e under routing.
Max $c(e)$?
$\max_e c(e) \geq \sum_e c(e)d(e)$ since $\sum_e d(e) = 1$.

$$\sum_e c(e)d(e) = \sum_i d(p_i)$$

$$\begin{aligned}
\sum_i d(p_i) &= \sum_i \sum_{e \in p_i} d(e) \\
&= \sum_e \sum_{i : e \ni p_i} d(e) \\
&= \sum_e d(e) \sum_{i : e \ni p_i} 1 \\
&= \sum_e d(e)c(e)
\end{aligned}$$

A path uses "volume" $d(p_i)$.

Volume on edge is $d(e)c(e)$.

$\sum_i d(p_i) = \sum_e d(e)c(e)$.

$\max_e c(e) \geq \sum_e d(e)c(e) = \sum_i d(p_i) \geq \sum_i d(s_i, t_i)$.
Routing solution cost $\geq$ Any toll solution cost.

Toll is lower bound.

From before:
Max bigger than minimum weighted average:
$\max_e c(e) \geq \sum_e c(e)d(e)$
Total length is total congestion: $\sum_e c(e)d(e) = \sum_i d(p_i)$

Toll is lower bound.

From before:
Max bigger than minimum weighted average:
$\max_e c(e) \geq \sum_e c(e)d(e)$
Total length is total congestion: $\sum_e c(e)d(e) = \sum_i d(p_i)$
Each path, p_i, in routing has length $d(p_i) \geq d(s_i, t_i)$.

Toll is lower bound.

From before:
Max bigger than minimum weighted average:
$\max_e c(e) \geq \sum_e c(e)d(e)$
Total length is total congestion: $\sum_e c(e)d(e) = \sum_i d(p_i)$
Each path, p_i, in routing has length $d(p_i) \geq d(s_i, t_i)$.

$$\max_e c(e) \geq \sum_e c(e)d(e) = \sum_i d(p_i) \geq \sum_i d(s_i, t_i).$$

Toll is lower bound.

From before:
Max bigger than minimum weighted average:
$\max_e c(e) \geq \sum_e c(e)d(e)$
Total length is total congestion: $\sum_e c(e)d(e) = \sum_i d(p_i)$
Each path, p_i, in routing has length $d(p_i) \geq d(s_i, t_i)$.

$$\max_e c(e) \geq \sum_e c(e)d(e) = \sum_i d(p_i) \geq \sum_i d(s_i, t_i).$$

Toll is lower bound.

From before:
Max bigger than minimum weighted average:
$\max_e c(e) \geq \sum_e c(e)d(e)$
Total length is total congestion: $\sum_e c(e)d(e) = \sum_i d(p_i)$
Each path, p_i, in routing has length $d(p_i) \geq d(s_i, t_i)$.

$$\max_e c(e) \geq \sum_e c(e)d(e) = \sum_i d(p_i) \geq \sum_i d(s_i, t_i).$$

Toll is lower bound.

From before:
Max bigger than minimum weighted average:
$\max_e c(e) \geq \sum_e c(e)d(e)$
Total length is total congestion: $\sum_e c(e)d(e) = \sum_i d(p_i)$
Each path, p_i, in routing has length $d(p_i) \geq d(s_i, t_i)$.

$$\max_e c(e) \geq \sum_e c(e)d(e) = \sum_i d(p_i) \geq \sum_i d(s_i, t_i).$$

A toll solution is lower bound on any routing solution.

Toll is lower bound.

From before:
Max bigger than minimum weighted average:
$\max_e c(e) \geq \sum_e c(e)d(e)$
Total length is total congestion: $\sum_e c(e)d(e) = \sum_i d(p_i)$
Each path, p_i, in routing has length $d(p_i) \geq d(s_i, t_i)$.

$$\max_e c(e) \geq \sum_e c(e)d(e) = \sum_i d(p_i) \geq \sum_i d(s_i, t_i).$$

A toll solution is lower bound on any routing solution.
Any routing solution is an upper bound on a toll solution.

Shall we continue?

Algorithm.

Assign tolls.

Algorithm.

Assign tolls.
How to route?

Algorithm.

Assign tolls.

How to route? Shortest paths!

Algorithm.

Assign tolls.
How to route? Shortest paths!
Assign routing.

Algorithm.

Assign tolls.
How to route? Shortest paths!
Assign routing.
How to assign tolls?

Algorithm.

Assign tolls.
How to route? Shortest paths!
Assign routing.
How to assign tolls? Higher tolls on congested edges.

Algorithm.

Assign tolls.
How to route? Shortest paths!
Assign routing.
How to assign tolls? Higher tolls on congested edges.
Toll: $d(e) \propto 2^{c(e)}$.

Algorithm.

Assign tolls.
How to route? Shortest paths!
Assign routing.
How to assign tolls? Higher tolls on congested edges.
Toll: $d(e) \propto 2^{c(e)}$.

Equilibrium:

Algorithm.

Assign tolls.
How to route? Shortest paths!
Assign routing.
How to assign tolls? Higher tolls on congested edges.
Toll: $d(e) \propto 2^{c(e)}$.

Equilibrium:
The shortest path routing has $d(e) \propto 2^{c(e)}$.

Algorithm.

Assign tolls.
How to route? Shortest paths!
Assign routing.
How to assign tolls? Higher tolls on congested edges.
Toll: $d(e) \propto 2^{c(e)}$.

Equilibrium:
The shortest path routing has $d(e) \propto 2^{c(e)}$.

The routing does not change, the tolls do not change.

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.

$$c_{opt} \quad \geq \quad \sum_i d(s_i, t_i) = \sum_e d(e)c(e)$$

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.

$$\begin{aligned} c_{opt} &\geq \sum_i d(s_i, t_i) = \sum_e d(e) c(e) \\ &= \sum_e \frac{2^{c(e)}}{\sum_{e'} 2^{c(e')}} c(e) \end{aligned}$$

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.

$$\begin{aligned} c_{opt} &\geq \sum_i d(s_i, t_i) = \sum_e d(e)c(e) \\ &= \sum_e \frac{2^{c(e)}}{\sum_{e'} 2^{c(e')}} c(e) = \frac{\sum_e 2^{c(e)} c(e)}{\sum_e 2^{c(e)}} \end{aligned}$$

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.

$$
\begin{aligned}
c_{opt} &\geq \sum_i d(s_i, t_i) = \sum_e d(e) c(e) \\
&= \sum_e \frac{2^{c(e)}}{\sum_{e'} 2^{c(e')}} c(e) = \frac{\sum_e 2^{c(e)} c(e)}{\sum_e 2^{c(e)}}
\end{aligned}
$$

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.

$$
\begin{aligned}
c_{opt} &\geq \sum_i d(s_i, t_i) = \sum_e d(e)c(e) \\
&= \sum_e \frac{2^{c(e)}}{\sum_{e'} 2^{c(e')}} c(e) = \frac{\sum_e 2^{c(e)} c(e)}{\sum_e 2^{c(e)}} \quad \text{Let } c_t = c_{max} - 2\log m. \\
&\geq \frac{\sum_{e:c(e)>c_t} 2^{c(e)} c(e)}{\sum_{e:c(e)>c_t} 2^{c(e)} + \sum_{e:c(e)\leq c_t} 2^{c(e)}}
\end{aligned}
$$

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.

For e with $c(e) \leq c_{max} - 2\log m$; $2^{c(e)} \leq 2^{c_{\max} - 2\log m} = \frac{2^{c_{max}}}{m^2}$.

$$
\begin{aligned}
c_{opt} &\geq \sum_i d(s_i, t_i) = \sum_e d(e)c(e) \\
&= \sum_e \frac{2^{c(e)}}{\sum_{e'} 2^{c(e')}} c(e) = \frac{\sum_e 2^{c(e)} c(e)}{\sum_e 2^{c(e)}} \quad \text{Let } c_t = c_{max} - 2\log m. \\
&\geq \frac{\sum_{e:c(e)>c_t} 2^{c(e)} c(e)}{\sum_{e:c(e)>c_t} 2^{c(e)} + \sum_{e:c(e)\leq c_t} 2^{c(e)}}
\end{aligned}
$$

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.
For e with $c(e) \le c_{max} - 2\log m$; $2^{c(e)} \le 2^{c_{\max} - 2\log m} = \frac{2^{c_{max}}}{m^2}$.

$$
\begin{aligned}
c_{opt} &\ge \sum_i d(s_i, t_i) = \sum_e d(e)c(e) \\
&= \sum_e \frac{2^{c(e)}}{\sum_{e'} 2^{c(e')}} c(e) = \frac{\sum_e 2^{c(e)} c(e)}{\sum_e 2^{c(e)}} \quad \text{Let } c_t = c_{max} - 2\log m. \\
&\ge \frac{\sum_{e:c(e)>c_t} 2^{c(e)} c(e)}{\sum_{e:c(e)>c_t} 2^{c(e)} + \sum_{e:c(e)\le c_t} 2^{c(e)}} \\
&\ge \frac{(c_t) \sum_{e:c(e)>c_t} 2^{c(e)}}{(1+\frac{1}{m}) \sum_{e:c(e)>c_t} 2^{c(e)}}
\end{aligned}
$$

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.
For e with $c(e) \leq c_{max} - 2\log m$; $2^{c(e)} \leq 2^{c_{\max} - 2\log m} = \frac{2^{c_{max}}}{m^2}$.

$$
\begin{aligned}
c_{opt} &\geq \sum_i d(s_i, t_i) = \sum_e d(e)c(e) \\
&= \sum_e \frac{2^{c(e)}}{\sum_{e'} 2^{c(e')}} c(e) = \frac{\sum_e 2^{c(e)} c(e)}{\sum_e 2^{c(e)}} \quad \text{Let } c_t = c_{max} - 2\log m. \\
&\geq \frac{\sum_{e:c(e)>c_t} 2^{c(e)} c(e)}{\sum_{e:c(e)>c_t} 2^{c(e)} + \sum_{e:c(e)\leq c_t} 2^{c(e)}} \\
&\geq \frac{(c_t) \sum_{e:c(e)>c_t} 2^{c(e)}}{(1 + \frac{1}{m}) \sum_{e:c(e)>c_t} 2^{c(e)}} \\
&\geq \frac{(c_t)}{1 + \frac{1}{m}} = \frac{c_{\max} - 2\log m}{(1 + \frac{1}{m})}
\end{aligned}
$$

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.
For e with $c(e) \leq c_{max} - 2\log m$; $2^{c(e)} \leq 2^{c_{\max} - 2\log m} = \frac{2^{c_{max}}}{m^2}$.

$$
\begin{aligned}
c_{opt} &\geq \sum_i d(s_i, t_i) = \sum_e d(e)c(e) \\
&= \sum_e \frac{2^{c(e)}}{\sum_{e'} 2^{c(e')}} c(e) = \frac{\sum_e 2^{c(e)} c(e)}{\sum_e 2^{c(e)}} \quad \text{Let } c_t = c_{max} - 2\log m. \\
&\geq \frac{\sum_{e:c(e)>c_t} 2^{c(e)} c(e)}{\sum_{e:c(e)>c_t} 2^{c(e)} + \sum_{e:c(e)\leq c_t} 2^{c(e)}} \\
&\geq \frac{(c_t) \sum_{e:c(e)>c_t} 2^{c(e)}}{(1 + \frac{1}{m}) \sum_{e:c(e)>c_t} 2^{c(e)}} \\
&\geq \frac{(c_t)}{1 + \frac{1}{m}} = \frac{c_{\max} - 2\log m}{(1 + \frac{1}{m})}
\end{aligned}
$$

Or $c_{max} \leq (1 + \frac{1}{m}) c_{opt} + 2\log m$.

How good is equilibrium?

Path is routed along shortest path and $d(e) \propto 2^{c(e)}$.
For e with $c(e) \leq c_{max} - 2\log m$; $2^{c(e)} \leq 2^{c_{\max} - 2\log m} = \frac{2^{c_{max}}}{m^2}$.

$$
\begin{aligned}
c_{opt} &\geq \sum_i d(s_i, t_i) = \sum_e d(e) c(e) \\
&= \sum_e \frac{2^{c(e)}}{\sum_{e'} 2^{c(e')}} c(e) = \frac{\sum_e 2^{c(e)} c(e)}{\sum_e 2^{c(e)}} \\
&\geq \frac{\sum_{e:c(e)>c_t} 2^{c(e)} c(e)}{\sum_{e:c(e)>c_t} 2^{c(e)} + \sum_{e:c(e)\leq c_t} 2^{c(e)}} \\
&\geq \frac{(c_t) \sum_{e:c(e)>c_t} 2^{c(e)}}{(1+\frac{1}{m}) \sum_{e:c(e)>c_t} 2^{c(e)}} \\
&\geq \frac{(c_t)}{1+\frac{1}{m}} = \frac{c_{\max} - 2\log m}{(1+\frac{1}{m})}
\end{aligned}
$$

Let $c_t = c_{max} - 2\log m$.

Or $c_{max} \leq (1+\frac{1}{m}) c_{opt} + 2\log m$.
(Almost) within $2\log m$ of optimal!

The end: sort of.

Getting to equilibrium.

Maybe no equilibrium!

Getting to equilibrium.

Maybe no equilibrium!

Approximate equilibrium:

Getting to equilibrium.

Maybe no equilibrium!

Approximate equilibrium:

Each path is routed along a path with length
within a factor of 3 of the shortest path and $d(e) \propto 2^{c(e)}$.

Getting to equilibrium.

Maybe no equilibrium!

Approximate equilibrium:

Each path is routed along a path with length
within a factor of 3 of the shortest path and $d(e) \propto 2^{c(e)}$.

Lose a factor of three at the beginning.

Getting to equilibrium.

Maybe no equilibrium!

Approximate equilibrium:

Each path is routed along a path with length
within a factor of 3 of the shortest path and $d(e) \propto 2^{c(e)}$.

Lose a factor of three at the beginning.
$c_{opt} \geq \sum_i d(s_i, t_i) \geq \frac{1}{3} \sum_e d(p_i)$.

Getting to equilibrium.

Maybe no equilibrium!

Approximate equilibrium:

Each path is routed along a path with length
within a factor of 3 of the shortest path and $d(e) \propto 2^{c(e)}$.

Lose a factor of three at the beginning.
$c_{opt} \geq \sum_i d(s_i, t_i) \geq \frac{1}{3} \sum_e d(p_i)$.
We obtain $c_{max} = 3(1 + \frac{1}{m})c_{opt} + 2 \log m$.

Getting to equilibrium.

Maybe no equilibrium!

Approximate equilibrium:

Each path is routed along a path with length
within a factor of 3 of the shortest path and $d(e) \propto 2^{c(e)}$.

Lose a factor of three at the beginning.
$c_{opt} \geq \sum_i d(s_i, t_i) \geq \frac{1}{3} \sum_e d(p_i)$.
We obtain $c_{max} = 3(1 + \frac{1}{m})c_{opt} + 2 \log m$.

This is worse!

Getting to equilibrium.

Maybe no equilibrium!

Approximate equilibrium:

Each path is routed along a path with length
within a factor of 3 of the shortest path and $d(e) \propto 2^{c(e)}$.

Lose a factor of three at the beginning.
$c_{opt} \geq \sum_i d(s_i, t_i) \geq \frac{1}{3} \sum_e d(p_i)$.
We obtain $c_{max} = 3(1 + \frac{1}{m})c_{opt} + 2 \log m$.

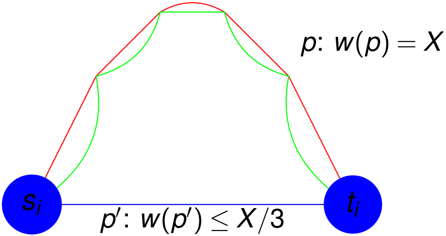
This is worse!
What do we gain?

An algorithm!

Algorithm: reroute paths that are off by a factor of three.
(Note: $d(e)$ recomputed every rerouting.)

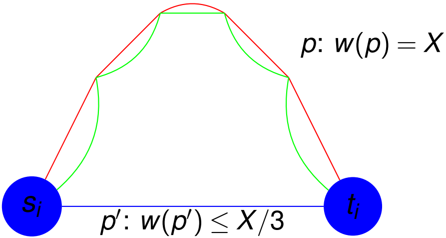
An algorithm!

Algorithm: reroute paths that are off by a factor of three.
(Note: $d(e)$ recomputed every rerouting.)



An algorithm!

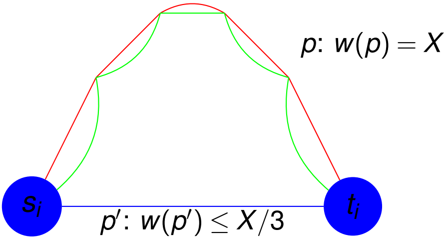
Algorithm: reroute paths that are off by a factor of three.
(Note: $d(e)$ recomputed every rerouting.)



Potential function: $\sum_e w(e)$, $w(e) = 2^{c(e)}$

An algorithm!

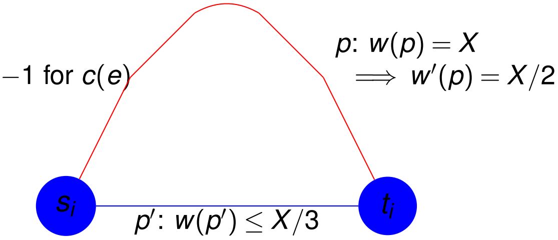
Algorithm: reroute paths that are off by a factor of three.
(Note: $d(e)$ recomputed every rerouting.)



Potential function: $\sum_e w(e)$, $w(e) = 2^{c(e)}$
Moving path:

An algorithm!

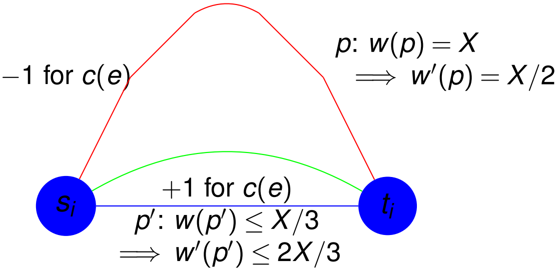
Algorithm: reroute paths that are off by a factor of three.
(Note: $d(e)$ recomputed every rerouting.)



Potential function: $\sum_e w(e)$, $w(e) = 2^{c(e)}$
Moving path:
Divides $w(e)$ along long path (with $w(p)$ of X) by two.

An algorithm!

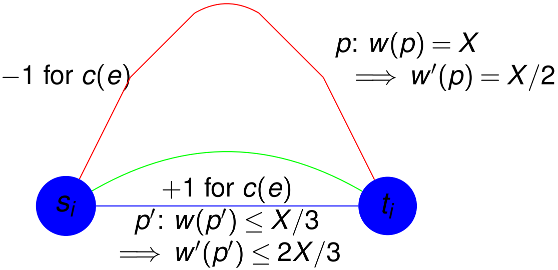
Algorithm: reroute paths that are off by a factor of three.
(Note: $d(e)$ recomputed every rerouting.)



Potential function: $\sum_e w(e)$, $w(e) = 2^{c(e)}$
Moving path:
Divides $w(e)$ along long path (with $w(p)$ of X) by two.
Multiplies $w(e)$ along shorter ($w(p) \leq X/3$) path by two.

An algorithm!

Algorithm: reroute paths that are off by a factor of three.
(Note: $d(e)$ recomputed every rerouting.)

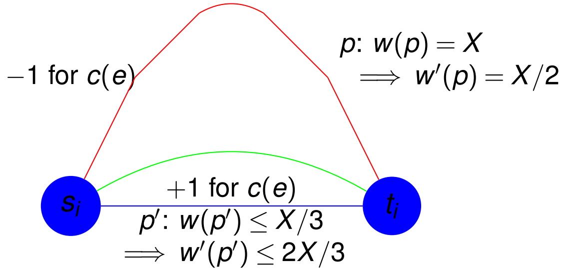


Potential function: $\sum_e w(e)$, $w(e) = 2^{c(e)}$
Moving path:
Divides $w(e)$ along long path (with $w(p)$ of X) by two.
Multiplies $w(e)$ along shorter ($w(p) \leq X/3$) path by two.

$$-\frac{X}{2} + \frac{X}{3} = -\frac{X}{6}.$$

An algorithm!

Algorithm: reroute paths that are off by a factor of three.
(Note: $d(e)$ recomputed every rerouting.)



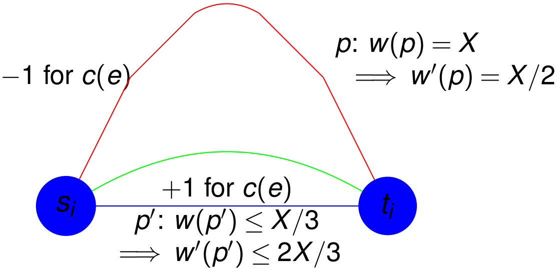
Potential function: $\sum_e w(e)$, $w(e) = 2^{c(e)}$
Moving path:
Divides $w(e)$ along long path (with $w(p)$ of X) by two.
Multiplies $w(e)$ along shorter ($w(p) \leq X/3$) path by two.

$$-\frac{X}{2} + \frac{X}{3} = -\frac{X}{6}.$$

Potential function decreases.

An algorithm!

Algorithm: reroute paths that are off by a factor of three.
(Note: $d(e)$ recomputed every rerouting.)



Potential function: $\sum_e w(e)$, $w(e) = 2^{c(e)}$
Moving path:
Divides $w(e)$ along long path (with $w(p)$ of X) by two.
Multiplies $w(e)$ along shorter ($w(p) \leq X/3$) path by two.

$$-\frac{X}{2} + \frac{X}{3} = -\frac{X}{6}.$$

Potential function decreases. $\implies$ termination and existence.

Tuning...

Tuning...

Replace $d(e) = (1+\varepsilon)^{c(e)}$.

Tuning...

Replace $d(e) = (1+\varepsilon)^{c(e)}$.

Replace factor of 3 by $(1+2\varepsilon)$

Tuning...

Replace $d(e) = (1+\varepsilon)^{c(e)}$.

Replace factor of 3 by $(1+2\varepsilon)$

$c_{max} \leq (1+2\varepsilon)c_{opt} + 2\log m/\varepsilon$.. (Roughly)

Tuning...

Replace $d(e) = (1+\varepsilon)^{c(e)}$.

Replace factor of 3 by $(1+2\varepsilon)$

$c_{max} \leq (1+2\varepsilon)c_{opt} + 2\log m/\varepsilon$.. (Roughly)

Fractional paths?

Wrap up.

Dueling players:

Wrap up.

Dueling players:
Toll player raises tolls on congested edges.

Wrap up.

Dueling players:
Toll player raises tolls on congested edges.
Congestion player avoids tolls.

Wrap up.

Dueling players:
Toll player raises tolls on congested edges.
Congestion player avoids tolls.

Converges to near optimal solution!

Wrap up.

Dueling players:
Toll player raises tolls on congested edges.
Congestion player avoids tolls.

Converges to near optimal solution!

A lower bound is “necessary” (natural),

Wrap up.

Dueling players:
Toll player raises tolls on congested edges.
Congestion player avoids tolls.

Converges to near optimal solution!

A lower bound is “necessary” (natural),
and helpful (mysterious?)!

Done for the day.....

...see you on Thursday.