Primal-Dual Algorithms for Weighted Abstract Path and Cut Packing

M Martens J Matuschke ST McCormick B Peis (M Skutella)

ZIB; Tor Vergata Rome; UBC; RWTH Aachen; TU Berlin

DIMACS, 20 Sept 2014

S. Thomas McCormick
Sauder School of Business
University of British Columbia

Outline

1. Combinatorial Optimization
   - Integral LPs

Outline

1. Combinatorial Optimization
   - Integral LPs
2. Hoffman's Models
   - Packing problems
   - Path models
   - Cut models
   - Blocking

Outline

1. Combinatorial Optimization
   - Integral LPs
2. Hoffman's Models
   - Packing problems
   - Path models
   - Cut models
   - Blocking
3. Algorithms
   - Primal-Dual Algorithm
   - P-D for path and cut packing

Outline

1. Combinatorial Optimization
   - Integral LPs
2. Hoffman's Models
   - Packing problems
   - Path models
   - Cut models
   - Blocking
3. Algorithms
   - Primal-Dual Algorithm
   - P-D for path and cut packing
4. Extensions
   - Flows over Time
   - Parametric Capacities

Outline

Outline

One Goal of Combinatorial Optimization

Much combinatorial optimization is around which LPs have guaranteed integer optimal solutions.

- Total unimodularity (TUM) — mostly network flow

One Goal of Combinatorial Optimization

Much combinatorial optimization is around which LPs have guaranteed integer optimal solutions.

- Total unimodularity (TUM) — mostly network flow
- Total dual integrality (TDI) — e.g., submodular RHSs

One Goal of Combinatorial Optimization

Much combinatorial optimization is around which LPs have guaranteed integer optimal solutions.

- Total unimodularity (TUM) — mostly network flow
- Total dual integrality (TDI) — e.g., submodular RHSs
- One early success: "A generalization of max flow-min cut" by A.J. Hoffman (*Math Prog* 1974):

One Goal of Combinatorial Optimization

Much combinatorial optimization is around which LPs have guaranteed integer optimal solutions.

- Total unimodularity (TUM) — mostly network flow
- Total dual integrality (TDI) — e.g., submodular RHSs
- One early success: "A generalization of max flow-min cut" by A.J. Hoffman (*Math Prog* 1974):
  - The first paper to formalize the notion that was later named TDI (by Edmonds and Giles) . . .

One Goal of Combinatorial Optimization

Much combinatorial optimization is around which LPs have guaranteed integer optimal solutions.

- Total unimodularity (TUM) — mostly network flow
- Total dual integrality (TDI) — e.g., submodular RHSs
- One early success: "A generalization of max flow-min cut" by A.J. Hoffman (*Math Prog* 1974):
  - The first paper to formalize the notion that was later named TDI (by Edmonds and Giles) . . .
  - . . . to show that a general model of max flow is still integral.

One Goal of Combinatorial Optimization

Much combinatorial optimization is around which LPs have guaranteed integer optimal solutions.

- Total unimodularity (TUM) — mostly network flow
- Total dual integrality (TDI) — e.g., submodular RHSs
- One early success: "A generalization of max flow-min cut" by A.J. Hoffman (*Math Prog* 1974):
  - The first paper to formalize the notion that was later named TDI (by Edmonds and Giles) . . .
  - . . . to show that a general model of max flow is still integral.
- Here we proceed in this same spirit.

Non-TUM but Integral Network LPs

There are non-TUM ways to formulate some network flow problems that still guarantee integrality:

- Max flow via flows on paths — packing paths into arcs.

Non-TUM but Integral Network LPs

There are non-TUM ways to formulate some network flow problems that still guarantee integrality:

- Max flow via flows on paths — packing paths into arcs.
- Dual of Dijkstra shortest path is packing cuts into arcs.

Non-TUM but Integral Network LPs

There are non-TUM ways to formulate some network flow problems that still guarantee integrality:

- Max flow via flows on paths — packing paths into arcs.
- Dual of Dijkstra shortest path is packing cuts into arcs.
- Recall that s–t paths and cuts are blockers of each other, i.e., paths are minimal arc subsets that hit every cut, and vice versa.

Non-TUM but Integral Network LPs

There are non-TUM ways to formulate some network flow problems that still guarantee integrality:

- Max flow via flows on paths — packing paths into arcs.
- Dual of Dijkstra shortest path is packing cuts into arcs.
- Recall that s–t paths and cuts are blockers of each other, i.e., paths are minimal arc subsets that hit every cut, and vice versa.
- These formulations do *not* in general work for weighted versions.

Non-TUM but Integral Network LPs

There are non-TUM ways to formulate some network flow problems that still guarantee integrality:

- Max flow via flows on paths — packing paths into arcs.
- Dual of Dijkstra shortest path is packing cuts into arcs.
- Recall that s–t paths and cuts are blockers of each other, i.e., paths are minimal arc subsets that hit every cut, and vice versa.
- These formulations do *not* in general work for weighted versions.
  - E.g., if we put general "rewards" on paths, then Max Weighted Path Flow is NP Hard.

Natural Questions

Question 1: Can we generalize to abstract versions of path and cut packing while maintaining integrality?

Natural Questions

Question 1: Can we generalize to abstract versions of path and cut packing while maintaining integrality?

Question 2: Both max flow-type path packing and dual-Dijkstra cut packing have all-one objective vectors, and are known to be fractional and NP Hard with general objectives. For which more general objectives are we still guaranteed integrality?

Natural Questions

Question 1: Can we generalize to abstract versions of path and cut packing while maintaining integrality?

Question 2: Both max flow-type path packing and dual-Dijkstra cut packing have all-one objective vectors, and are known to be fractional and NP Hard with general objectives. For which more general objectives are we still guaranteed integrality?

Question 3: Can we find polynomial algorithms for these abstract weighted path and cut packing problems?

Outline

Packing problems

A generic packing problem has

- A finite set E of elements

Packing problems

A generic packing problem has

- A finite set E of elements
- A family $\mathcal{D}$ of subsets of E, i.e., $D \in \mathcal{D} \implies D \subseteq E$.

Packing problems

A generic packing problem has

- A finite set E of elements
- A family $\mathcal{D}$ of subsets of E, i.e., $D \in \mathcal{D} \implies D \subseteq E$.
- A vector $u \in \mathbb{Z}^E$ of capacities on elements.

Packing problems

A generic packing problem has

- A finite set E of elements
- A family $\mathcal{D}$ of subsets of E, i.e., $D \in \mathcal{D} \implies D \subseteq E$.
- A vector $u \in \mathbb{Z}^E$ of capacities on elements.
- A vector $r \in \mathbb{Z}^{\mathcal{D}}$ of rewards on subsets.

Packing problems

A generic packing problem has

- A finite set E of elements
- A family $\mathcal{D}$ of subsets of E, i.e., $D \in \mathcal{D} \implies D \subseteq E$.
- A vector $u \in \mathbb{Z}^E$ of capacities on elements.
- A vector $r \in \mathbb{Z}^{\mathcal{D}}$ of rewards on subsets.
- The decision is to choose a weight y_D to put on each $D \in \mathcal{D}$ such that the total weight packed into e is at most u_e $\forall$ $e \in E$.

Packing problems

A generic packing problem has

- A finite set E of elements
- A family $\mathcal{D}$ of subsets of E, i.e., $D \in \mathcal{D} \implies D \subseteq E$.
- A vector $u \in \mathbb{Z}^E$ of capacities on elements.
- A vector $r \in \mathbb{Z}^{\mathcal{D}}$ of rewards on subsets.
- The decision is to choose a weight y_D to put on each $D \in \mathcal{D}$ such that the total weight packed into e is at most u_e $\forall$ $e \in E$.
- And among such feasible packings, find one that maximizes $r^T y$.

Packing problems

A generic packing problem has

- A finite set E of elements
- A family $\mathcal{D}$ of subsets of E, i.e., $D \in \mathcal{D} \implies D \subseteq E$.
- A vector $u \in \mathbb{Z}^E$ of capacities on elements.
- A vector $r \in \mathbb{Z}^{\mathcal{D}}$ of rewards on subsets.
- The decision is to choose a weight y_D to put on each $D \in \mathcal{D}$ such that the total weight packed into e is at most $u_e \; \forall \; e \in E$.
- And among such feasible packings, find one that maximizes $r^T y$.
- We are usually interested in finding *integer* optimal solutions.

Packing problems

A generic packing problem has

- A finite set E of elements
- A family $\mathcal{D}$ of subsets of E, i.e., $D \in \mathcal{D} \implies D \subseteq E$.
- A vector $u \in \mathbb{Z}^E$ of capacities on elements.
- A vector $r \in \mathbb{Z}^{\mathcal{D}}$ of rewards on subsets.
- The decision is to choose a weight y_D to put on each $D \in \mathcal{D}$ such that the total weight packed into e is at most $u_e \; \forall \; e \in E$.
- And among such feasible packings, find one that maximizes $r^T y$.
- We are usually interested in finding *integer* optimal solutions.
- This generic problem has many applications, e.g., flow is packing paths into arcs, connectivity is packing trees into edges, etc.

Packing as an LP

- Now formulate a packing problem as an LP (it's more natural to make packing the dual):

Packing as an LP

- Now formulate a packing problem as an LP (it's more natural to make packing the dual):
  - put dual packing variable y_D on each $D \in \mathcal{D}$;

Packing as an LP

- Now formulate a packing problem as an LP (it's more natural to make packing the dual):
  - put dual packing variable y_D on each $D \in \mathcal{D}$;
  - put primal weight x_e on each element $e \in E$.

Packing as an LP

- Now formulate a packing problem as an LP (it's more natural to make packing the dual):
  - put dual packing variable y_D on each $D \in \mathcal{D}$;
  - put primal weight x_e on each element $e \in E$.
- The dual linear programs are:

$$\text{(D)} \quad \max \sum_{D} r_D y_D \qquad \text{s.t.} \sum_{D \ni e} y_D \le u_e \quad \forall e \in E \qquad y \ge 0$$

$$\text{(P)} \quad \min \sum_{e} u_e x_e \qquad \text{s.t.} \sum_{e \in D} x_e \ge r_D \quad \forall D \in \mathcal{D} \qquad x \ge 0$$

Packing as an LP

- Now formulate a packing problem as an LP (it's more natural to make packing the dual):
  - put dual packing variable y_D on each $D \in \mathcal{D}$;
  - put primal weight x_e on each element $e \in E$.
- The dual linear programs are:

$$\text{(D)} \quad \max \sum_D r_D y_D \quad \text{s.t.} \sum_{D \ni e} y_D \le u_e \quad \forall e \in E, \quad y \ge 0$$

$$\text{(P)} \quad \min \sum_e u_e x_e \quad \text{s.t.} \sum_{e \in D} x_e \ge r_D \quad \forall D \in \mathcal{D}, \quad x \ge 0$$

"packing subsets into elements"

Packing as an LP

- Now formulate a packing problem as an LP (it's more natural to make packing the dual):
  - put dual packing variable y_D on each $D \in \mathcal{D}$;
  - put primal weight x_e on each element $e \in E$.
- The dual linear programs are:

(D) $$\begin{aligned} \max \quad & \sum_D r_D y_D \\ \text{s.t.} \quad & \sum_{D \ni e} y_D \le u_e \quad \forall e \in E \\ & y \ge 0 \end{aligned}$$

"packing subsets into elements"

(P) $$\begin{aligned} \min \quad & \sum_e u_e x_e \\ \text{s.t.} \quad & \sum_{e \in D} x_e \ge r_D \quad \forall D \in \mathcal{D} \\ & x \ge 0 \end{aligned}$$

"covering subsets by elements"

Packing as an LP

- Now formulate a packing problem as an LP (it's more natural to make packing the dual):
  - put dual packing variable y_D on each $D \in \mathcal{D}$;
  - put primal weight x_e on each element $e \in E$.
- The dual linear programs are:

$$\begin{aligned} \text{(D)} \quad \max \quad & \sum_{D} r_D y_D \\ \text{s.t.} \quad & \sum_{D \ni e} y_D \le u_e \quad \forall e \in E \\ & y \ge 0 \end{aligned} \qquad \begin{aligned} \text{(P)} \quad \min \quad & \sum_{e} u_e x_e \\ \text{s.t.} \quad & \sum_{e \in D} x_e \ge r_D \quad \forall D \in \mathcal{D} \\ & x \ge 0 \end{aligned}$$

Big Question: When do these LPs have guaranteed integer optimal solutions?

An example packing LP

- Consider:

$$\max \mathbb{1}^T y$$

$$\text{s.t.} \quad \begin{pmatrix} 1 & & 1 & & & 1 & & & \\ 1 & 1 & & & 1 & & & & \\ & 1 & 1 & & 1 & 1 & & & \\ & 1 & & 1 & & & & 1 & \\ & & 1 & 1 & 1 & 1 & & 1 & \\ & & 1 & 1 & & & 1 & & \\ & & & & 1 & 1 & 1 & 1 & \\ & & & & 1 & & 1 & & 1 \\ & & & & & 1 & & 1 & 1 \end{pmatrix} y \leq \begin{pmatrix} 1 \\ 5 \\ 5 \\ 8 \\ 4 \\ 7 \\ 9 \\ 3 \\ 6 \end{pmatrix}$$

$$y \geq 0.$$

An example packing LP

- Consider:

$$\max \mathbb{1}^T y$$

$$\text{s.t.} \quad \begin{pmatrix} 1 & & 1 & & & 1 & & & \\ 1 & 1 & & & 1 & & & & \\ & 1 & 1 & & 1 & 1 & & & \\ & 1 & & 1 & & & & 1 & \\ & & 1 & 1 & 1 & 1 & & 1 & \\ & & 1 & 1 & & & 1 & & \\ & & & & 1 & 1 & 1 & 1 & \\ & & & & 1 & & 1 & & 1 \\ & & & & & 1 & & 1 & 1 \end{pmatrix} y \le \begin{pmatrix} 1 \\ 5 \\ 5 \\ 8 \\ 4 \\ 7 \\ 9 \\ 3 \\ 6 \end{pmatrix}$$

$$y \ge 0.$$

- Does this LP have an integer optimal solution?

An example packing LP

- Consider:

$$\max \mathbb{1}^T y$$

$$\text{s.t.} \quad \begin{pmatrix} 1 & & 1 & & & 1 & & & \\ 1 & 1 & & & 1 & & & & \\ & 1 & 1 & & 1 & 1 & & & \\ & 1 & & 1 & & & & 1 & \\ & & 1 & 1 & 1 & 1 & & 1 & \\ & & 1 & 1 & & & 1 & & \\ & & & & 1 & 1 & 1 & 1 & \\ & & & & 1 & & 1 & & 1 \\ & & & & & 1 & & 1 & 1 \end{pmatrix} y \leq \begin{pmatrix} 1 \\ 5 \\ 5 \\ 8 \\ 4 \\ 7 \\ 9 \\ 3 \\ 6 \end{pmatrix}$$

$$y \geq 0.$$

- Does this LP have an integer optimal solution?
- What if we change the RHS u? The objective r?

More on the example

- This LP has an integer optimal solution: $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 3\ 0\ 0)$ of value 12.

More on the example

- This LP has an integer optimal solution: $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 3\ 0\ 0)$ of value 12.
- In fact, it can be shown that this LP has integer optimal solutions for *any* RHS u.

More on the example

- This LP has an integer optimal solution: $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 3\ 0\ 0)$ of value 12.
- In fact, it can be shown that this LP has integer optimal solutions for *any* RHS u.
- The same holds true for some objectives r:

More on the example

- This LP has an integer optimal solution: $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 3\ 0\ 0)$ of value 12.
- In fact, it can be shown that this LP has integer optimal solutions for *any* RHS u.
- The same holds true for some objectives r:
  - E.g., $r = (4\ 3\ 2\ 3\ 1\ 1\ 3\ 2\ 4)$ has integer optimal solution $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 0\ 0\ 3)$ of value 40 for the given RHS u, and this is true for any integral u.

More on the example

- This LP has an integer optimal solution: $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 3\ 0\ 0)$ of value 12.
- In fact, it can be shown that this LP has integer optimal solutions for *any* RHS u.
- The same holds true for some objectives r:
  - E.g., $r = (4\ 3\ 2\ 3\ 1\ 1\ 3\ 2\ 4)$ has integer optimal solution $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 0\ 0\ 3)$ of value 40 for the given RHS u, and this is true for any integral u.
- But not all objectives r:

More on the example

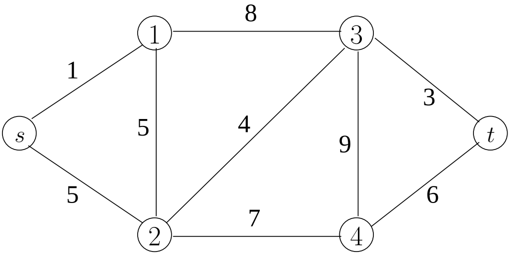
- This LP has an integer optimal solution: $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 3\ 0\ 0)$ of value 12.
- In fact, it can be shown that this LP has integer optimal solutions for *any* RHS u.
- The same holds true for some objectives r:
  - E.g., $r = (4\ 3\ 2\ 3\ 1\ 1\ 3\ 2\ 4)$ has integer optimal solution $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 0\ 0\ 3)$ of value 40 for the given RHS u, and this is true for any integral u.
- But not all objectives r:
  - E.g., $r = (0\ 9\ 0\ 0\ 9\ 0\ 0\ 9\ 0)$ has fractional optimal solution $y^* = (0\ 4.5\ 0\ 0\ 0.5\ 0\ 0\ 3.5\ 2.5)$ with value 76.5 for the given RHS u.

More on the example

- This LP has an integer optimal solution: $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 3\ 0\ 0)$ of value 12.
- In fact, it can be shown that this LP has integer optimal solutions for *any* RHS u.
- The same holds true for some objectives r:
  - E.g., $r = (4\ 3\ 2\ 3\ 1\ 1\ 3\ 2\ 4)$ has integer optimal solution $y^* = (1\ 4\ 0\ 4\ 0\ 0\ 0\ 0\ 3)$ of value 40 for the given RHS u, and this is true for any integral u.
- But not all objectives r:
  - E.g., $r = (0\ 9\ 0\ 0\ 9\ 0\ 0\ 9\ 0)$ has fractional optimal solution $y^* = (0\ 4.5\ 0\ 0\ 0.5\ 0\ 0\ 3.5\ 2.5)$ with value 76.5 for the given RHS u.
- How do I know that the first two objectives are "good" for all RHS?

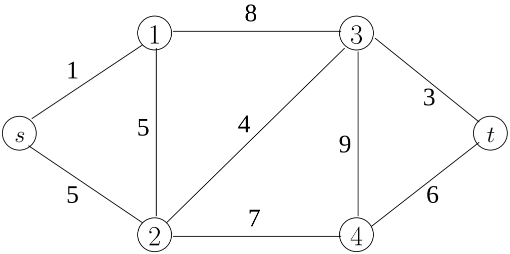
How the example was constructed

- Consider the following graph:



How the example was constructed

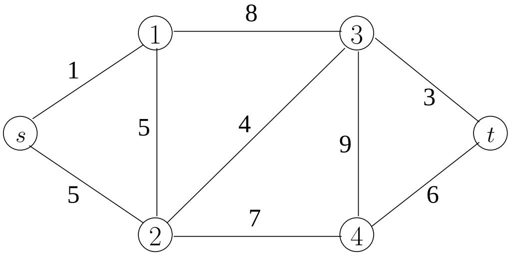
- Consider the following graph:



- There is a 1–1 correspondence between E and the nine edges of this graph.

How the example was constructed

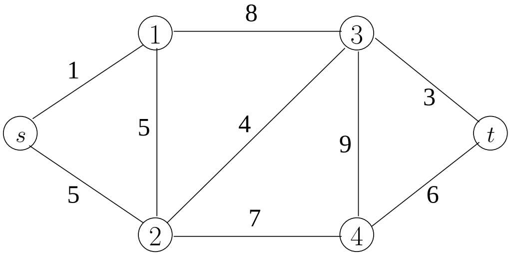
- Consider the following graph:



- There is a 1–1 correspondence between E and the nine edges of this graph.
- There is a 1–1 correspondence between the 9 interesting s–t cuts in this graph and the columns of the constraint matrix.

How the example was constructed

- Consider the following graph:



- There is a 1–1 correspondence between E and the nine edges of this graph.
- There is a 1–1 correspondence between the 9 interesting s–t cuts in this graph and the columns of the constraint matrix.
- Why does this lead to integer optimal LP solutions?

The primal covering LP

- Recall that the primal covering LP has variables x_e ...

The primal covering LP

- Recall that the primal covering LP has variables x_e ...
- ...and constraints $\sum_{e \in D} x_e \geq 1$ for all $D \in \mathcal{D}$.

The primal covering LP

- Recall that the primal covering LP has variables x_e ...
- ...and constraints $\sum_{e \in D} x_e \geq 1$ for all $D \in \mathcal{D}$.
- Imagine that x is 0–1, so that it picks out a subset of edges.

The primal covering LP

- Recall that the primal covering LP has variables x_e ...
- ... and constraints $\sum_{e \in D} x_e \geq 1$ for all $D \in \mathcal{D}$.
- Imagine that x is 0–1, so that it picks out a subset of edges.
- What subsets of edges hit every s–t cut?

The primal covering LP

- Recall that the primal covering LP has variables x_e ...
- ... and constraints $\sum_{e \in D} x_e \geq 1$ for all $D \in \mathcal{D}$.
- Imagine that x is 0–1, so that it picks out a subset of edges.
- What subsets of edges hit every s–t cut?
- The s–t paths are the minimal edge subsets hitting every s–t cut, i.e., the s–t paths are the blocker of the s–t cuts.

The primal covering LP

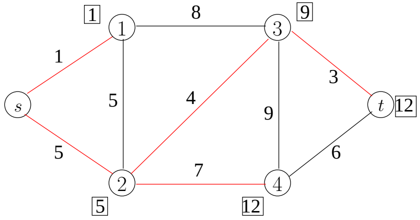
- Recall that the primal covering LP has variables x_e ...
- ... and constraints $\sum_{e \in D} x_e \geq 1$ for all $D \in \mathcal{D}$.
- Imagine that x is 0–1, so that it picks out a subset of edges.
- What subsets of edges hit every s–t cut?
- The s–t paths are the minimal edge subsets hitting every s–t cut, i.e., the s–t paths are the blocker of the s–t cuts.
- Therefore the primal LP is just Shortest Path.

The primal covering LP

- Recall that the primal covering LP has variables x_e ...
- ... and constraints $\sum_{e \in D} x_e \geq 1$ for all $D \in \mathcal{D}$.
- Imagine that x is 0–1, so that it picks out a subset of edges.
- What subsets of edges hit every s–t cut?
- The s–t paths are the minimal edge subsets hitting every s–t cut, i.e., the s–t paths are the blocker of the s–t cuts.
- Therefore the primal LP is just Shortest Path.
- And in fact Dijkstra's Algorithm gives an integer optimal solution to this form of Shortest Path.

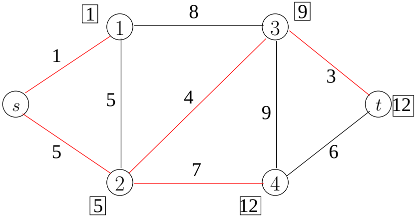
Going back to the dual packing LP

- Here is the Dijkstra solution with its shortest path tree:

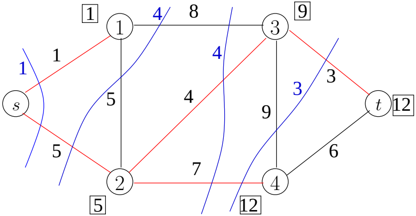


Going back to the dual packing LP

- Here is the Dijkstra solution with its shortest path tree:



- Recall that we can greedily construct a tight cut packing that proves that this shortest path tree is optimal:



Generalizing this behavior

- Since we know that Dijkstra, and this greedy cut packing, work for any non-negative capacities u, we know that we get integer optimal solutions for all RHS u.

Generalizing this behavior

- Since we know that Dijkstra, and this greedy cut packing, work for any non-negative capacities u, we know that we get integer optimal solutions for all RHS u.
- It is very cool that this random-looking constraint matrix always has an integer optimal solution *with the special objective vector* $\mathbb{1}$.

Generalizing this behavior

- Since we know that Dijkstra, and this greedy cut packing, work for any non-negative capacities u, we know that we get integer optimal solutions for all RHS u.
- It is very cool that this random-looking constraint matrix always has an integer optimal solution *with the special objective vector* $\mathbb{1}$.
- LPs such as this where you get guaranteed integer optimal solutions for all RHSs, but only for some special objective vectors, are called Totally Dual Integral, or TDI.

Generalizing this behavior

- Since we know that Dijkstra, and this greedy cut packing, work for any non-negative capacities u, we know that we get integer optimal solutions for all RHS u.
- It is very cool that this random-looking constraint matrix always has an integer optimal solution *with the special objective vector* $\mathbb{1}$.
- LPs such as this where you get guaranteed integer optimal solutions for all RHSs, but only for some special objective vectors, are called Totally Dual Integral, or TDI.
- A natural question here is whether we can generalize this sort of example to a broader class of packing LPs with 0–1 constraint matrices.

Generalizing this behavior

- Since we know that Dijkstra, and this greedy cut packing, work for any non-negative capacities u, we know that we get integer optimal solutions for all RHS u.
- It is very cool that this random-looking constraint matrix always has an integer optimal solution *with the special objective vector* $\mathbb{1}$.
- LPs such as this where you get guaranteed integer optimal solutions for all RHSs, but only for some special objective vectors, are called Totally Dual Integral, or TDI.
- A natural question here is whether we can generalize this sort of example to a broader class of packing LPs with 0–1 constraint matrices.
- Hoffman did it . . .

Alan Hoffman's Answers to Q1, Q2 for Paths

The Weighted Abstract Flow model:

- We are given a finite set of elements E (nodes/arcs/mixed)

Alan Hoffman's Answers to Q1, Q2 for Paths

The Weighted Abstract Flow model:

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e

Alan Hoffman's Answers to Q1, Q2 for Paths

The Weighted Abstract Flow model:

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{P}$ of paths, where

Alan Hoffman's Answers to Q1, Q2 for Paths

The Weighted Abstract Flow model:

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{P}$ of paths, where
  - $P \in \mathcal{P}$ means that $P \subseteq E$

Alan Hoffman's Answers to Q1, Q2 for Paths

The Weighted Abstract Flow model:

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{P}$ of paths, where
  - $P \in \mathcal{P}$ means that $P \subseteq E$
  - each $P \in \mathcal{P}$ has a linear order $<_P$ (could have $e <_P f$ but $f <_Q e$)

Alan Hoffman's Answers to Q1, Q2 for Paths

The Weighted Abstract Flow model:

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{P}$ of paths, where
  - $P \in \mathcal{P}$ means that $P \subseteq E$
  - each $P \in \mathcal{P}$ has a linear order $<_P$ (could have $e <_P f$ but $f <_Q e$)
    - Make artificial s with $s <_P e$ and t with $e <_P t \; \forall \; e \in P$ and define, e.g., $(s, f]_P = \{e \in P \mid e \leq f\}$.

Alan Hoffman's Answers to Q1, Q2 for Paths

The Weighted Abstract Flow model:

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{P}$ of paths, where
  - $P \in \mathcal{P}$ means that $P \subseteq E$
  - each $P \in \mathcal{P}$ has a linear order $<_P$ (could have $e <_P f$ but $f <_Q e$)
    - Make artificial s with $s <_P e$ and t with $e <_P t \ \forall \ e \in P$ and define, e.g., $(s, f]_P = \{e \in P \mid e \leq f\}$.
  - each $P \in \mathcal{P}$ has a per flow unit reward r_P (the *weight* of P)

Alan Hoffman's Answers to Q1, Q2 for Paths

The Weighted Abstract Flow model:

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{P}$ of paths, where
  - $P \in \mathcal{P}$ means that $P \subseteq E$
  - each $P \in \mathcal{P}$ has a linear order $<_P$ (could have $e <_P f$ but $f <_Q e$)
    - Make artificial s with $s <_P e$ and t with $e <_P t \ \forall \ e \in P$ and define, e.g., $(s, f]_P = \{e \in P \mid e \leq f\}$.
  - each $P \in \mathcal{P}$ has a per flow unit reward r_P (the *weight* of P)
- E and $\mathcal{P}$ are connected by a Crossing Axiom (F & F):
  If $e \in P \cap Q$, then
  $P \times_e Q := \text{argmax}\{r_V \mid V \in \mathcal{P},\ V \subseteq (s, e]_P \cup [e, t)_Q\}$ is well-defined.

Alan Hoffman's Answers to Q1, Q2 for Paths

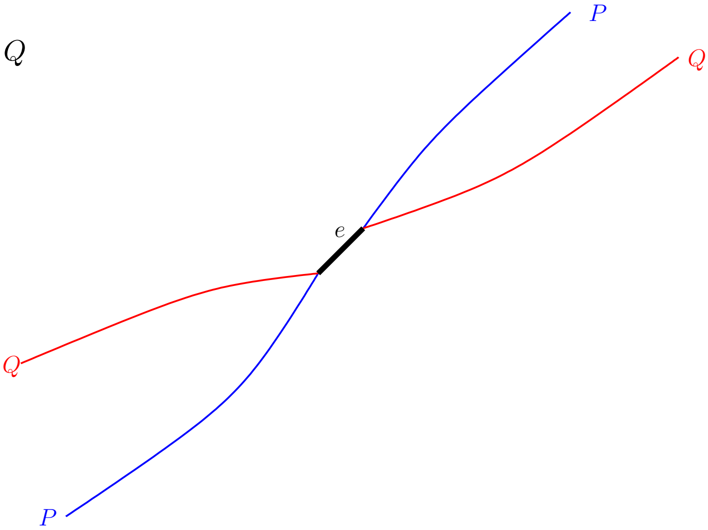
The Weighted Abstract Flow model:

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{P}$ of paths, where
  - $P \in \mathcal{P}$ means that $P \subseteq E$
  - each $P \in \mathcal{P}$ has a linear order $<_P$ (could have $e <_P f$ but $f <_Q e$)
    - Make artificial s with $s <_P e$ and t with $e <_P t \; \forall \; e \in P$ and define, e.g., $(s, f]_P = \{e \in P \mid e \leq f\}$.
  - each $P \in \mathcal{P}$ has a per flow unit reward r_P (the *weight* of P)
- E and $\mathcal{P}$ are connected by a Crossing Axiom (F & F): If $e \in P \cap Q$, then $P \times_e Q := \text{argmax}\{r_V \mid V \in \mathcal{P}, \; V \subseteq (s, e]_P \cup [e, t)_Q\}$ is well-defined.
- r satisfies a kind of supermodularity:

$$r_{P \times_e Q} + r_{Q \times_e P} \geq r_P + r_Q.$$

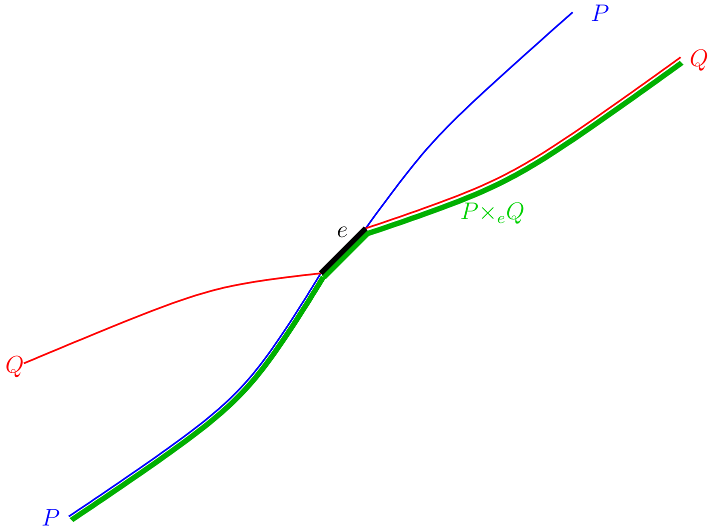
Picture of Crossing Axiom

$e \in P \cap Q$



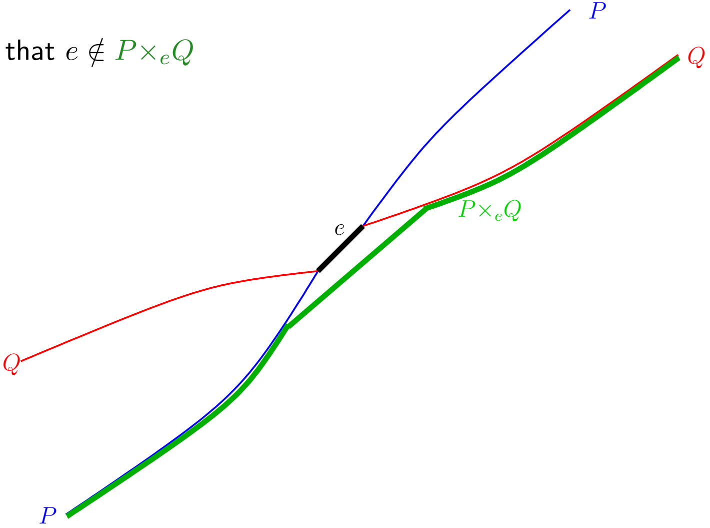
Picture of Crossing Axiom

$P \times_e Q$



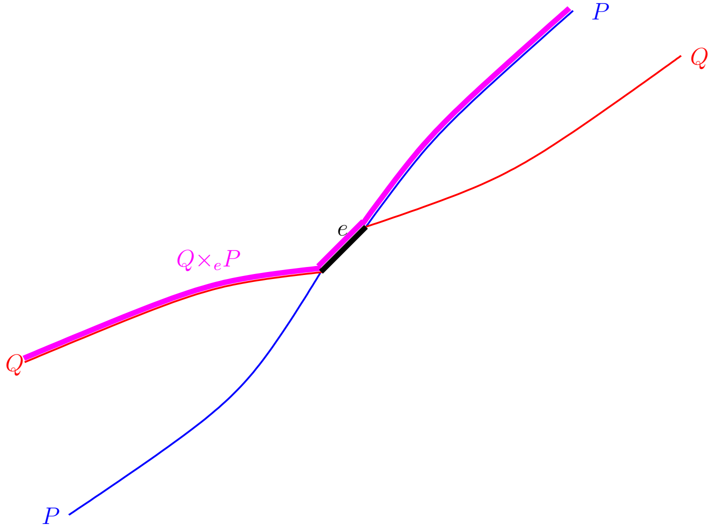
Picture of Crossing Axiom

Possible that $e \notin P \times_e Q$



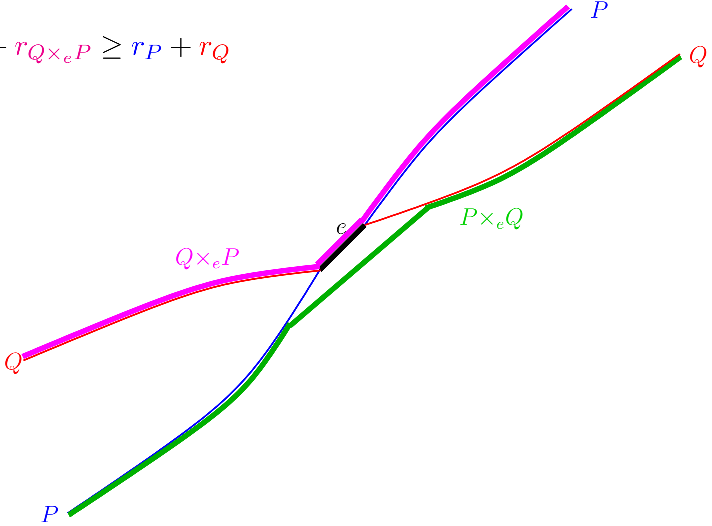
Picture of Crossing Axiom

$Q \times_e P$



Picture of Crossing Axiom

$$r_{P\times_e Q} + r_{Q\times_e P} \geq r_P + r_Q$$



The Weighted Abstract Flow linear programs

- The *Weighted Abstract Flow (WAF)* problem associated with E and $\mathcal{P}$ puts

The Weighted Abstract Flow linear programs

- The *Weighted Abstract Flow (WAF)* problem associated with E and $\mathcal{P}$ puts
  - flow variable x_P on each $P \in \mathcal{P}$;

The Weighted Abstract Flow linear programs

- The *Weighted Abstract Flow (WAF)* problem associated with E and $\mathcal{P}$ puts
  - flow variable x_P on each $P \in \mathcal{P}$;
  - weight y_e on each element $e \in E$.

The Weighted Abstract Flow linear programs

- The *Weighted Abstract Flow (WAF)* problem associated with E and $\mathcal{P}$ puts
  - flow variable x_P on each $P \in \mathcal{P}$;
  - weight y_e on each element $e \in E$.
- The dual linear programs are:

$$
\text{(P)} \quad \max \sum_{P} r_P x_P \qquad\qquad \text{(D)} \quad \min \sum_{e} u_e y_e
$$

$$
\text{s.t.} \sum_{P \ni e} x_P \le u_e \quad \forall e \in E \qquad\qquad \text{s.t.} \sum_{e \in P} y_e \ge r_P \quad \forall P \in \mathcal{P}
$$

$$
x \ge 0 \qquad\qquad\qquad\qquad y \ge 0
$$

The Weighted Abstract Flow linear programs

- The *Weighted Abstract Flow (WAF)* problem associated with E and $\mathcal{P}$ puts
  - flow variable x_P on each $P \in \mathcal{P}$;
  - weight y_e on each element $e \in E$.
- The dual linear programs are:

$$\text{(P)} \quad \max \sum_{P} r_P x_P \quad \text{s.t.} \sum_{P \ni e} x_P \leq u_e \quad \forall e \in E, \quad x \geq 0$$

$$\text{(D)} \quad \min \sum_{e} u_e y_e \quad \text{s.t.} \sum_{e \in P} y_e \geq r_P \quad \forall P \in \mathcal{P}, \quad y \geq 0$$

"packing paths into elements"

The Weighted Abstract Flow linear programs

- The *Weighted Abstract Flow (WAF)* problem associated with E and $\mathcal{P}$ puts
  - flow variable x_P on each $P \in \mathcal{P}$;
  - weight y_e on each element $e \in E$.
- The dual linear programs are:

$$\text{(P)} \quad \max \sum_P r_P x_P \quad \text{s.t.} \sum_{P \ni e} x_P \le u_e \quad \forall e \in E, \quad x \ge 0$$

"packing paths into elements"

$$\text{(D)} \quad \min \sum_e u_e y_e \quad \text{s.t.} \sum_{e \in P} y_e \ge r_P \quad \forall P \in \mathcal{P}, \quad y \ge 0$$

"covering paths by elements"

The Weighted Abstract Flow linear programs

- The *Weighted Abstract Flow (WAF)* problem associated with E and $\mathcal{P}$ puts
  - flow variable x_P on each $P \in \mathcal{P}$;
  - weight y_e on each element $e \in E$.
- The dual linear programs are:

$$
\begin{aligned}
\text{(P)} \quad \max \quad & \sum_{P} r_P x_P \\
\text{s.t.} \quad & \sum_{P \ni e} x_P \le u_e \quad \forall e \in E \\
& x \ge 0
\end{aligned}
\qquad
\begin{aligned}
\text{(D)} \quad \min \quad & \sum_{e} u_e y_e \\
\text{s.t.} \quad & \sum_{e \in P} y_e \ge r_P \quad \forall P \in \mathcal{P} \\
& y \ge 0
\end{aligned}
$$

If $\mathcal{P}$ is just s–t paths in a max flow network, and $r \equiv 1$, then this is just the usual formulation of Max Flow/Min Cut using path-flow variables.

The Weighted Abstract Flow linear programs

- The *Weighted Abstract Flow (WAF)* problem associated with E and $\mathcal{P}$ puts
  - flow variable x_P on each $P \in \mathcal{P}$;
  - weight y_e on each element $e \in E$.
- The dual linear programs are:

$$\text{(P)} \quad \max \sum_P r_P x_P \qquad\qquad \text{(D)} \quad \min \sum_e u_e y_e$$

$$\text{s.t.} \sum_{P \ni e} x_P \le u_e \quad \forall e \in E \qquad\qquad \text{s.t.} \sum_{e \in P} y_e \ge r_P \quad \forall P \in \mathcal{P}$$

$$x \ge 0 \qquad\qquad y \ge 0$$

Theorem (Hoffman '74)

When r and u are integral, (P) and (D) have integral optimal solutions.

Notes on Weighted Abstract Flow

- In 1974 there were a lot of papers being written on minor variations of Max Flow Min Cut.

Notes on Weighted Abstract Flow

- In 1974 there were a lot of papers being written on minor variations of Max Flow Min Cut.
- Alan's paper captured all these variations in one fell swoop.

Notes on Weighted Abstract Flow

- In 1974 there were a lot of papers being written on minor variations of Max Flow Min Cut.
- Alan's paper captured all these variations in one fell swoop.
- Alan's model was motivated by the (rarely read) *original* paper by Ford and Fulkerson on MF/MC.

Notes on Weighted Abstract Flow

- In 1974 there were a lot of papers being written on minor variations of Max Flow Min Cut.
- Alan's paper captured all these variations in one fell swoop.
- Alan's model was motivated by the (rarely read) *original* paper by Ford and Fulkerson on MF/MC.
- The possibility of supermodular r is interesting:

Notes on Weighted Abstract Flow

- In 1974 there were a lot of papers being written on minor variations of Max Flow Min Cut.
- Alan's paper captured all these variations in one fell swoop.
- Alan's model was motivated by the (rarely read) *original* paper by Ford and Fulkerson on MF/MC.
- The possibility of supermodular r is interesting:
    - It means that the model includes transportation problems (and hence min-cost flow)

Notes on Weighted Abstract Flow

- In 1974 there were a lot of papers being written on minor variations of Max Flow Min Cut.
- Alan's paper captured all these variations in one fell swoop.
- Alan's model was motivated by the (rarely read) *original* paper by Ford and Fulkerson on MF/MC.
- The possibility of supermodular r is interesting:
  - It means that the model includes transportation problems (and hence min-cost flow)
  - Alan remarked in a 2010 email to me "when I first wrote the paper with the [super]modular r (rather than all 1's), I put in the r because it came free".

Notes on Weighted Abstract Flow

- In 1974 there were a lot of papers being written on minor variations of Max Flow Min Cut.
- Alan's paper captured all these variations in one fell swoop.
- Alan's model was motivated by the (rarely read) *original* paper by Ford and Fulkerson on MF/MC.
- The possibility of supermodular r is interesting:
  - It means that the model includes transportation problems (and hence min-cost flow)
  - Alan remarked in a 2010 email to me "when I first wrote the paper with the [super]modular r (rather than all 1's), I put in the r because it came free".
  - Alan earlier verbally told me that he put in the supermodular r because he wanted to imitate the nice things that Jack Edmonds was doing.

NETFLOW '93

- In a talk at NETFLOW '93 (San Miniato, Italy) Alan asked:

NETFLOW '93

- In a talk at NETFLOW '93 (San Miniato, Italy) Alan asked:

  Is there a reasonable way using only [the WAF assumptions] and some modest oracle to find a generalized max flow and a min cut? Since it is known ['74 paper] that, for integral capacities, there is an optimum flow which is integral, it would even be progress to find an algorithm which increases a given flow by one unit, if the given flow is not optimum.

NETFLOW '93

- In a talk at NETFLOW '93 (San Miniato, Italy) Alan asked:

  > *Is there a reasonable way using only [the WAF assumptions] and some modest oracle to find a generalized max flow and a min cut? Since it is known ['74 paper] that, for integral capacities, there is an optimum flow which is integral, it would even be progress to find an algorithm which increases a given flow by one unit, if the given flow is not optimum.*

- At lunch afterwards Bill Pulleyblank accosted some of us and said something like "surely some of you young guys should be able to answer Alan's question".

NETFLOW '93

- In a talk at NETFLOW '93 (San Miniato, Italy) Alan asked:

  > *Is there a reasonable way using only [the WAF assumptions] and some modest oracle to find a generalized max flow and a min cut? Since it is known ['74 paper] that, for integral capacities, there is an optimum flow which is integral, it would even be progress to find an algorithm which increases a given flow by one unit, if the given flow is not optimum.*

- At lunch afterwards Bill Pulleyblank accosted some of us and said something like "surely some of you young guys should be able to answer Alan's question".
- As a bonus, Bill relayed to us Alan's concrete suggestion for an oracle for the max flow ($r \equiv 1$) version: You send the oracle a subset S of the elements, and it tells you whether there is a path P with $P \subseteq S$ (and $<_P$) or not.

Alan Hoffman's Answers to Q1, Q2 for Cuts

- We are given a finite set of elements E (nodes/arcs/mixed)

Alan Hoffman's Answers to Q1, Q2 for Cuts

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e

Alan Hoffman's Answers to Q1, Q2 for Cuts

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{L}$ of cuts, where

Alan Hoffman's Answers to Q1, Q2 for Cuts

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{L}$ of cuts, where
  - $D \in \mathcal{L}$ means that $D \subseteq E$

Alan Hoffman's Answers to Q1, Q2 for Cuts

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{L}$ of cuts, where
  - $D \in \mathcal{L}$ means that $D \subseteq E$
  - $\mathcal{L}$ is a lattice with partial order $\preceq$ and operations $\wedge$ and $\vee$ satisfying

Alan Hoffman's Answers to Q1, Q2 for Cuts

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{L}$ of cuts, where
  - $D \in \mathcal{L}$ means that $D \subseteq E$
  - $\mathcal{L}$ is a lattice with partial order $\preceq$ and operations $\wedge$ and $\vee$ satisfying
    - $D_i \prec D_j \prec D_k \implies D_i \cap D_k \subseteq D_j$ (consecutive), and

Alan Hoffman's Answers to Q1, Q2 for Cuts

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{L}$ of cuts, where
  - $D \in \mathcal{L}$ means that $D \subseteq E$
  - $\mathcal{L}$ is a lattice with partial order $\preceq$ and operations $\wedge$ and $\vee$ satisfying
    - $D_i \prec D_j \prec D_k \implies D_i \cap D_k \subseteq D_j$ (consecutive), and
    - $(D_i \wedge D_j) \cup (D_i \vee D_j) \subseteq D_i \cup D_j$ (submodular).

Alan Hoffman's Answers to Q1, Q2 for Cuts

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{L}$ of cuts, where
  - $D \in \mathcal{L}$ means that $D \subseteq E$
  - $\mathcal{L}$ is a lattice with partial order $\preceq$ and operations $\wedge$ and $\vee$ satisfying
    - $D_i \prec D_j \prec D_k \implies D_i \cap D_k \subseteq D_j$ (consecutive), and
    - $(D_i \wedge D_j) \cup (D_i \vee D_j) \subseteq D_i \cup D_j$ (submodular).
  - each $D \in \mathcal{L}$ has a per unit reward r_D (the *weight* of D)

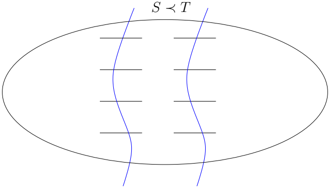
Alan Hoffman's Answers to Q1, Q2 for Cuts

- We are given a finite set of elements E (nodes/arcs/mixed)
  - Each $e \in E$ has capacity u_e
- And a family $\mathcal{L}$ of cuts, where
  - $D \in \mathcal{L}$ means that $D \subseteq E$
  - $\mathcal{L}$ is a lattice with partial order $\preceq$ and operations $\wedge$ and $\vee$ satisfying
    - $D_i \prec D_j \prec D_k \implies D_i \cap D_k \subseteq D_j$ (consecutive), and
    - $(D_i \wedge D_j) \cup (D_i \vee D_j) \subseteq D_i \cup D_j$ (submodular).
  - each $D \in \mathcal{L}$ has a per unit reward r_D (the *weight* of D)
- r satisfies a kind of supermodularity:

$$r_{D_i \wedge D_j} + r_{D_i \vee D_j} \geq r_{D_i} + r_{D_j}.$$

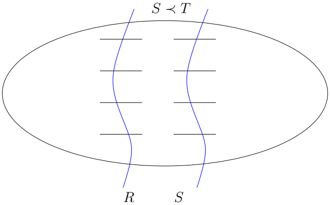
Understanding the Cut Axioms

Ordinary cuts are partially ordered:

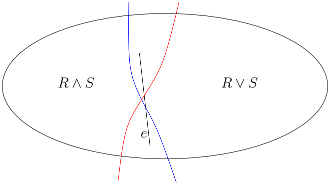


Understanding the Cut Axioms

Ordinary cuts are partially ordered:

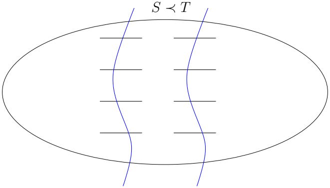


Ordinary cuts have meet and join, and submodularity:

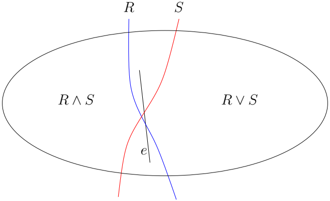


Understanding the Cut Axioms

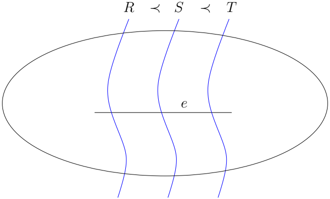
Ordinary cuts are partially ordered:



Ordinary cuts have meet and join, and submodularity:



Ordinary cuts are consecutive ($e \in R \cap T \implies e \in S$):



The Weighted Abstract Cut Packing linear programs

- The *Weighted Abstract Cut Packing (WACP)* problem associated with E and $\mathcal{L}$ puts

The Weighted Abstract Cut Packing linear programs

- The *Weighted Abstract Cut Packing (WACP)* problem associated with E and $\mathcal{L}$ puts
  - packing variable x_D on each $D \in \mathcal{L}$;

The Weighted Abstract Cut Packing linear programs

- The *Weighted Abstract Cut Packing (WACP)* problem associated with E and $\mathcal{L}$ puts
  - packing variable x_D on each $D \in \mathcal{L}$;
  - weight y_e on each element $e \in E$.

The Weighted Abstract Cut Packing linear programs

- The *Weighted Abstract Cut Packing (WACP)* problem associated with E and $\mathcal{L}$ puts
  - packing variable x_D on each $D \in \mathcal{L}$;
  - weight y_e on each element $e \in E$.
- The dual linear programs are:

$$\text{(P)} \quad \max \sum_{D} r_D x_D \qquad \text{s.t.} \sum_{D \ni e} x_D \le u_e \quad \forall e \in E \qquad x \ge 0$$

$$\text{(D)} \quad \min \sum_{e} u_e y_e \qquad \text{s.t.} \sum_{e \in D} y_e \ge r_D \quad \forall D \in \mathcal{L} \qquad y \ge 0$$

The Weighted Abstract Cut Packing linear programs

- The *Weighted Abstract Cut Packing (WACP)* problem associated with E and $\mathcal{L}$ puts
  - packing variable x_D on each $D \in \mathcal{L}$;
  - weight y_e on each element $e \in E$.
- The dual linear programs are:

$$\text{(P)} \quad \max \sum_D r_D x_D \quad \text{s.t.} \sum_{D \ni e} x_D \le u_e \quad \forall e \in E, \quad x \ge 0$$

$$\text{(D)} \quad \min \sum_e u_e y_e \quad \text{s.t.} \sum_{e \in D} y_e \ge r_D \quad \forall D \in \mathcal{L}, \quad y \ge 0$$

"packing cuts into elements"

The Weighted Abstract Cut Packing linear programs

- The *Weighted Abstract Cut Packing (WACP)* problem associated with E and $\mathcal{L}$ puts
  - packing variable x_D on each $D \in \mathcal{L}$;
  - weight y_e on each element $e \in E$.
- The dual linear programs are:

$$\text{(P)} \quad \max \sum_D r_D x_D \quad \text{s.t.} \sum_{D \ni e} x_D \le u_e \quad \forall e \in E, \quad x \ge 0$$

"packing cuts into elements"

$$\text{(D)} \quad \min \sum_e u_e y_e \quad \text{s.t.} \sum_{e \in D} y_e \ge r_D \quad \forall D \in \mathcal{L}, \quad y \ge 0$$

"covering cuts by elements"

The Weighted Abstract Cut Packing linear programs

- The *Weighted Abstract Cut Packing (WACP)* problem associated with E and $\mathcal{L}$ puts
  - packing variable x_D on each $D \in \mathcal{L}$;
  - weight y_e on each element $e \in E$.
- The dual linear programs are:

$$\text{(P)} \quad \max \sum_{D} r_D x_D \quad \text{s.t.} \sum_{D \ni e} x_D \le u_e \quad \forall e \in E, \quad x \ge 0$$

$$\text{(D)} \quad \min \sum_{e} u_e y_e \quad \text{s.t.} \sum_{e \in D} y_e \ge r_D \quad \forall D \in \mathcal{L}, \quad y \ge 0$$

If $\mathcal{L}$ is just s–t cuts in a max flow network, and $r \equiv 1$, then this is just the usual blocking dual formulation of Dijkstra shortest path.

The Weighted Abstract Cut Packing linear programs

- The *Weighted Abstract Cut Packing (WACP)* problem associated with E and $\mathcal{L}$ puts
  - packing variable x_D on each $D \in \mathcal{L}$;
  - weight y_e on each element $e \in E$.
- The dual linear programs are:

$$
\text{(P)} \quad \max \sum_{D} r_D x_D \qquad\qquad \text{(D)} \quad \min \sum_{e} u_e y_e
$$

$$
\text{s.t.} \sum_{D \ni e} x_D \le u_e \quad \forall e \in E \qquad\qquad \text{s.t.} \sum_{e \in D} y_e \ge r_D \quad \forall D \in \mathcal{L}
$$

$$
x \ge 0 \qquad\qquad y \ge 0
$$

Theorem (Hoffman & Schwartz '76)

When r and u are integral, (P) and (D) have integral optimal solutions.

Other applications

Lattice polyhedra would not be so interesting unless they included interesting applications other than Shortest Path:

- Dilworth's Theorem (chains and antichains in posets) and various Greene-Kleitman generalizations.

Other applications

Lattice polyhedra would not be so interesting unless they included interesting applications other than Shortest Path:

- Dilworth's Theorem (chains and antichains in posets) and various Greene-Kleitman generalizations.
- Shortest Path in hypergraphs.

Other applications

Lattice polyhedra would not be so interesting unless they included interesting applications other than Shortest Path:

- Dilworth's Theorem (chains and antichains in posets) and various Greene-Kleitman generalizations.
- Shortest Path in hypergraphs.
- Polymatroids and intersections of polymatroids.

Other applications

Lattice polyhedra would not be so interesting unless they included interesting applications other than Shortest Path:

- Dilworth's Theorem (chains and antichains in posets) and various Greene-Kleitman generalizations.
- Shortest Path in hypergraphs.
- Polymatroids and intersections of polymatroids.
- Min-cost arborescence.

Other applications

Lattice polyhedra would not be so interesting unless they included interesting applications other than Shortest Path:

- Dilworth's Theorem (chains and antichains in posets) and various Greene-Kleitman generalizations.
- Shortest Path in hypergraphs.
- Polymatroids and intersections of polymatroids.
- Min-cost arborescence.
- Our example with $r = (4\ 3\ 2\ 3\ 1\ 1\ 3\ 2\ 4)$ has integer optimal solutions for all RHS u because this r is supermodular: each $r_D = 6 - \#$ edges crossing D.

Other applications

Lattice polyhedra would not be so interesting unless they included interesting applications other than Shortest Path:

- Dilworth's Theorem (chains and antichains in posets) and various Greene-Kleitman generalizations.
- Shortest Path in hypergraphs.
- Polymatroids and intersections of polymatroids.
- Min-cost arborescence.
- Our example with $r = (4\ 3\ 2\ 3\ 1\ 1\ 3\ 2\ 4)$ has integer optimal solutions for all RHS u because this r is supermodular: each $r_D = 6 - \#$ edges crossing D.
- Our example with $r = (0\ 9\ 0\ 0\ 9\ 0\ 0\ 9\ 0)$ can have a fractional solution because this r is not supermodular.

Other applications

Lattice polyhedra would not be so interesting unless they included interesting applications other than Shortest Path:

- Dilworth's Theorem (chains and antichains in posets) and various Greene-Kleitman generalizations.
- Shortest Path in hypergraphs.
- Polymatroids and intersections of polymatroids.
- Min-cost arborescence.
- Our example with $r = (4\ 3\ 2\ 3\ 1\ 1\ 3\ 2\ 4)$ has integer optimal solutions for all RHS u because this r is supermodular: each $r_D = 6 - \#$ edges crossing D.
- Our example with $r = (0\ 9\ 0\ 0\ 9\ 0\ 0\ 9\ 0)$ can have a fractional solution because this r is not supermodular.
- Etc, etc ...

Blocking Carries Over

Suppose that $\mathcal{L}$ is a clutter, i.e., if $R, S \in \mathcal{L}$, then $R \not\subset S$ and $S \not\subset R$ (edge sets of ordinary cuts are a clutter). Then

Blocking Carries Over

Suppose that $\mathcal{L}$ is a clutter, i.e., if $R, S \in \mathcal{L}$, then $R \not\subset S$ and $S \not\subset R$ (edge sets of ordinary cuts are a clutter). Then

Theorem (Hoffman '78)

If $\mathcal{L}$ is a submodular clutter, then the blocker of $\mathcal{L}$ is an abstract path system.

Blocking Carries Over

Suppose that $\mathcal{L}$ is a clutter, i.e., if $R, S \in \mathcal{L}$, then $R \not\subset S$ and $S \not\subset R$ (edge sets of ordinary cuts are a clutter). Then

Theorem (Hoffman '78)

If $\mathcal{L}$ is a submodular clutter, then the blocker of $\mathcal{L}$ is an abstract path system.

Thus Weighted Abstract Flow and Weighted Abstract Cut Packing carry over the blocking relationship of ordinary s–t paths and cuts.

Blocking Carries Over

Suppose that $\mathcal{L}$ is a clutter, i.e., if $R, S \in \mathcal{L}$, then $R \not\subset S$ and $S \not\subset R$ (edge sets of ordinary cuts are a clutter). Then

Theorem (Hoffman '78)

If $\mathcal{L}$ is a submodular clutter, then the blocker of $\mathcal{L}$ is an abstract path system.

Thus Weighted Abstract Flow and Weighted Abstract Cut Packing carry over the blocking relationship of ordinary s–t paths and cuts.

What remains now is Q3:

Are there polynomial algorithms for solving Weighted Abstract Flow and Cut Packing?

Outline

The Primal-Dual Algorithm

- Recall the Primal-Dual (Successive Shortest Path, SSP) Algorithm for max flow at min cost.

The Primal-Dual Algorithm

- Recall the Primal-Dual (Successive Shortest Path, SSP) Algorithm for max flow at min cost.
- It greedily pushes flow on the cheapest (shortest) augmenting path.

Primal-Dual Algorithm:

The Primal-Dual Algorithm

- Recall the Primal-Dual (Successive Shortest Path, SSP) Algorithm for max flow at min cost.
- It greedily pushes flow on the cheapest (shortest) augmenting path.

Primal-Dual Algorithm:

Set $x = 0$, $\pi = 0$.

The Primal-Dual Algorithm

- Recall the Primal-Dual (Successive Shortest Path, SSP) Algorithm for max flow at min cost.
- It greedily pushes flow on the cheapest (shortest) augmenting path.

Primal-Dual Algorithm:

Set $x = 0$, $\pi = 0$.
While augmenting paths remain do

End

The Primal-Dual Algorithm

- Recall the Primal-Dual (Successive Shortest Path, SSP) Algorithm for max flow at min cost.
- It greedily pushes flow on the cheapest (shortest) augmenting path.

Primal-Dual Algorithm:

Set $x = 0$, $\pi = 0$.
While augmenting paths remain do
  Use Shortest Path to compute the subnetwork $\mathcal{S}$
    of min-cost augmenting paths (dual change).

End

The Primal-Dual Algorithm

- Recall the Primal-Dual (Successive Shortest Path, SSP) Algorithm for max flow at min cost.
- It greedily pushes flow on the cheapest (shortest) augmenting path.

Primal-Dual Algorithm:

Set $x = 0$, $\pi = 0$.
While augmenting paths remain do
    Use Shortest Path to compute the subnetwork $\mathcal{S}$
        of min-cost augmenting paths (dual change).
    Use Max Flow to augment all paths in $\mathcal{S}$ (primal change).
End

The Primal-Dual Algorithm

- Recall the Primal-Dual (Successive Shortest Path, SSP) Algorithm for max flow at min cost.
- It greedily pushes flow on the cheapest (shortest) augmenting path.

Primal-Dual Algorithm:

Set $x = 0$, $\pi = 0$.
While augmenting paths remain do
    Use Shortest Path to compute the subnetwork $\mathcal{S}$
        of min-cost augmenting paths (dual change).
    Use Max Flow to augment all paths in $\mathcal{S}$ (primal change).
End

- Each iteration maintains that x and π are optimal for current flow value, so when x becomes a max flow, it is optimal.

A Technical Detail

- Complementary slackness $\implies$ if a dual variable > 0, the primal constraint must stay tight.

A Technical Detail

- Complementary slackness $\implies$ if a dual variable > 0, the primal constraint must stay tight.
- Thus P-D solves a restricted problem in inner iterations where some elements in R must stay tight.

A Technical Detail

- Complementary slackness $\implies$ if a dual variable > 0, the primal constraint must stay tight.
- Thus P-D solves a restricted problem in inner iterations where some elements in R must stay tight.
- But otherwise, the advantage of P-D is that it replaces the complicated objective $r^T x$ with a simple objective $\mathbb{1}^T x$.

A Technical Detail

- Complementary slackness $\implies$ if a dual variable > 0, the primal constraint must stay tight.
- Thus P-D solves a restricted problem in inner iterations where some elements in R must stay tight.
- But otherwise, the advantage of P-D is that it replaces the complicated objective $r^T x$ with a simple objective $\mathbb{1}^T x$.
- Due to R, the solution to the restricted dual could have -1 values in it, so the dual update need not be monotone.

P-D for Path and Cut Packing 1

P-D for Path and Cut Packing 1

Path Packing

- $\max$ instead of $\min \implies$ must start with max weight paths.

P-D for Path and Cut Packing 1

Path Packing

- $\max$ instead of $\min$ $\implies$ must start with max weight paths.

Cut Packing

- $\max$ instead of $\min$ $\implies$ must start with max weight cuts.

P-D for Path and Cut Packing 1

Path Packing

- $\max$ instead of $\min \implies$ must start with max weight paths.
- Define λ as the weight of the current highest-reward path; initially $\lambda = \max_P r_P = r_{\max}$.

Cut Packing

- $\max$ instead of $\min \implies$ must start with max weight cuts.

P-D for Path and Cut Packing 1

Path Packing

- $\max$ instead of $\min$ $\implies$ must start with max weight paths.
- Define λ as the weight of the current highest-reward path; initially $\lambda = \max_P r_P = r_{\max}$.

Cut Packing

- $\max$ instead of $\min$ $\implies$ must start with max weight cuts.
- Define λ as the weight of the current highest-reward cut; initially $\lambda = \max_D r_D = r_{\max}$.

P-D for Path and Cut Packing 1

Path Packing

- $\max$ instead of $\min \implies$ must start with max weight paths.
- Define λ as the weight of the current highest-reward path; initially $\lambda = \max_P r_P = r_{\max}$.
- Relax $y(P) \geq r_P$ to $y(P) \geq r_P - \lambda$.

Cut Packing

- $\max$ instead of $\min \implies$ must start with max weight cuts.
- Define λ as the weight of the current highest-reward cut; initially $\lambda = \max_D r_D = r_{\max}$.

P-D for Path and Cut Packing 1

Path Packing

- $\max$ instead of $\min \implies$ must start with max weight paths.
- Define λ as the weight of the current highest-reward path; initially $\lambda = \max_P r_P = r_{\max}$.
- Relax $y(P) \geq r_P$ to $y(P) \geq r_P - \lambda$.

Cut Packing

- $\max$ instead of $\min \implies$ must start with max weight cuts.
- Define λ as the weight of the current highest-reward cut; initially $\lambda = \max_D r_D = r_{\max}$.
- Relax $y(D) \geq r_D$ to $y(D) \geq r_D - \lambda$.

P-D for Path and Cut Packing 1

Path Packing

- $\max$ instead of $\min \implies$ must start with max weight paths.
- Define λ as the weight of the current highest-reward path; initially $\lambda = \max_P r_P = r_{\max}$.
- Relax $y(P) \geq r_P$ to $y(P) \geq r_P - \lambda$.
- [When $\lambda = r_{\max}$, $x = y = 0$ is optimal.]

Cut Packing

- $\max$ instead of $\min \implies$ must start with max weight cuts.
- Define λ as the weight of the current highest-reward cut; initially $\lambda = \max_D r_D = r_{\max}$.
- Relax $y(D) \geq r_D$ to $y(D) \geq r_D - \lambda$.

P-D for Path and Cut Packing 1

Path Packing

- $\max$ instead of $\min \implies$ must start with max weight paths.
- Define λ as the weight of the current highest-reward path; initially $\lambda = \max_P r_P = r_{\max}$.
- Relax $y(P) \geq r_P$ to $y(P) \geq r_P - \lambda$.
- [When $\lambda = r_{\max}$, $x = y = 0$ is optimal.]

Cut Packing

- $\max$ instead of $\min \implies$ must start with max weight cuts.
- Define λ as the weight of the current highest-reward cut; initially $\lambda = \max_D r_D = r_{\max}$.
- Relax $y(D) \geq r_D$ to $y(D) \geq r_D - \lambda$.
- [When $\lambda = r_{\max}$, $x = y = 0$ is optimal.]

P-D for Path and Cut Packing 1

Path Packing

- $\max$ instead of $\min \implies$ must start with max weight paths.
- Define λ as the weight of the current highest-reward path; initially $\lambda = \max_P r_P = r_{\max}$.
- Relax $y(P) \geq r_P$ to $y(P) \geq r_P - \lambda$.
- [When $\lambda = r_{\max}$, $x = y = 0$ is optimal.]
- Now decrease λ to 0, keeping optimality $\implies$ when $\lambda = 0$ we are optimal.

Cut Packing

- $\max$ instead of $\min \implies$ must start with max weight cuts.
- Define λ as the weight of the current highest-reward cut; initially $\lambda = \max_D r_D = r_{\max}$.
- Relax $y(D) \geq r_D$ to $y(D) \geq r_D - \lambda$.
- [When $\lambda = r_{\max}$, $x = y = 0$ is optimal.]

P-D for Path and Cut Packing 1

Path Packing

- $\max$ instead of $\min \implies$ must start with max weight paths.
- Define λ as the weight of the current highest-reward path; initially $\lambda = \max_P r_P = r_{\max}$.
- Relax $y(P) \geq r_P$ to $y(P) \geq r_P - \lambda$.
- [When $\lambda = r_{\max}$, $x = y = 0$ is optimal.]
- Now decrease λ to 0, keeping optimality $\implies$ when $\lambda = 0$ we are optimal.

Cut Packing

- $\max$ instead of $\min \implies$ must start with max weight cuts.
- Define λ as the weight of the current highest-reward cut; initially $\lambda = \max_D r_D = r_{\max}$.
- Relax $y(D) \geq r_D$ to $y(D) \geq r_D - \lambda$.
- [When $\lambda = r_{\max}$, $x = y = 0$ is optimal.]
- Now decrease λ to 0, keeping optimality $\implies$ when $\lambda = 0$ we are optimal.

P-D for Path and Cut Packing 2

P-D for Path and Cut Packing 2

Path Packing

- Martens & Mc.

P-D for Path and Cut Packing 2

Path Packing

- Martens & Mc.

Cut Packing

- Mc & Peis.

P-D for Path and Cut Packing 2

Path Packing

- Martens & Mc.
- "Finished": see IPCO Bertinoro 2008.

Cut Packing

- Mc & Peis.

P-D for Path and Cut Packing 2

Path Packing

- Martens & Mc.
- “Finished”: see IPCO Bertinoro 2008.

Cut Packing

- Mc & Peis.
- “Finished”: see IPCO Yorktown Hts 2011.

P-D for Path and Cut Packing 2

Path Packing

- Martens & Mc.
- "Finished": see IPCO Bertinoro 2008.
- For fixed λ, focus on subnetwork of paths with $\text{gap}(P) = y(P) - r_P + \lambda = 0$.

Cut Packing

- Mc & Peis.
- "Finished": see IPCO Yorktown Hts 2011.

P-D for Path and Cut Packing 2

Path Packing

- Martens & Mc.
- "Finished": see IPCO Bertinoro 2008.
- For fixed λ, focus on subnetwork of paths with $\text{gap}(P) = y(P) - r_P + \lambda = 0$.

Cut Packing

- Mc & Peis.
- "Finished": see IPCO Yorktown Hts 2011.
- For fixed λ, focus on subnetwork of cuts with $\text{gap}(D) = y(D) - r_D + \lambda = 0$.

P-D for Path and Cut Packing 2

Path Packing

- Martens & Mc.
- "Finished": see IPCO Bertinoro 2008.
- For fixed λ, focus on subnetwork of paths with $\text{gap}(P) = y(P) - r_P + \lambda = 0$.
- Lemma: this subnetwork still satisfies the axioms.

Cut Packing

- Mc & Peis.
- "Finished": see IPCO Yorktown Hts 2011.
- For fixed λ, focus on subnetwork of cuts with $\text{gap}(D) = y(D) - r_D + \lambda = 0$.

P-D for Path and Cut Packing 2

Path Packing

- Martens & Mc.
- "Finished": see IPCO Bertinoro 2008.
- For fixed λ, focus on subnetwork of paths with $\text{gap}(P) = y(P) - r_P + \lambda = 0$.
- Lemma: this subnetwork still satisfies the axioms.

Cut Packing

- Mc & Peis.
- "Finished": see IPCO Yorktown Hts 2011.
- For fixed λ, focus on subnetwork of cuts with $\text{gap}(D) = y(D) - r_D + \lambda = 0$.
- Lemma: this subnetwork still satisfies the axioms.

P-D for Path and Cut Packing 2

Path Packing

- Martens & Mc.
- "Finished": see IPCO Bertinoro 2008.
- For fixed λ, focus on subnetwork of paths with $\text{gap}(P) = y(P) - r_P + \lambda = 0$.
- Lemma: this subnetwork still satisfies the axioms.
- *But* $R = \{e \mid y_e > 0\}$ is restricted to be tight, i.e., $\sum_{P \ni e} x_P = u_e$.

Cut Packing

- Mc & Peis.
- "Finished": see IPCO Yorktown Hts 2011.
- For fixed λ, focus on subnetwork of cuts with $\text{gap}(D) = y(D) - r_D + \lambda = 0$.
- Lemma: this subnetwork still satisfies the axioms.

P-D for Path and Cut Packing 2

Path Packing

- Martens & Mc.
- "Finished": see IPCO Bertinoro 2008.
- For fixed λ, focus on subnetwork of paths with $\text{gap}(P) = y(P) - r_P + \lambda = 0$.
- Lemma: this subnetwork still satisfies the axioms.
- *But* $R = \{e \mid y_e > 0\}$ is restricted to be tight, i.e., $\sum_{P \ni e} x_P = u_e$.

Cut Packing

- Mc & Peis.
- "Finished": see IPCO Yorktown Hts 2011.
- For fixed λ, focus on subnetwork of cuts with $\text{gap}(D) = y(D) - r_D + \lambda = 0$.
- Lemma: this subnetwork still satisfies the axioms.
- *But* $R = \{e \mid y_e > 0\}$ is restricted to be tight, i.e., $\sum_{D \ni e} x_D = u_e$.

P-D for Path and Cut Packing 3

P-D for Path and Cut Packing 3

Path Packing

- Solve $\mathrm{gap}(P) = 0$ subnetwork using extension of Mc '95 Abstract MF.

P-D for Path and Cut Packing 3

Path Packing

- Solve $\mathrm{gap}(P) = 0$ subnetwork using extension of Mc '95 Abstract MF.

Cut Packing

- Solve $\mathrm{gap}(D) = 0$ subnetwork using extension of A. Frank '99 Abstract SP.

P-D for Path and Cut Packing 3

Path Packing

- Solve $\mathrm{gap}(P) = 0$ subnetwork using extension of Mc '95 Abstract MF.
- Since restr. subnetwork is MF, it's blocked by a Min Cut l.

Cut Packing

- Solve $\mathrm{gap}(D) = 0$ subnetwork using extension of A. Frank '99 Abstract SP.

P-D for Path and Cut Packing 3

Path Packing

- Solve $\mathrm{gap}(P) = 0$ subnetwork using extension of Mc '95 Abstract MF.
- Since restr. subnetwork is MF, it's blocked by a Min Cut l.

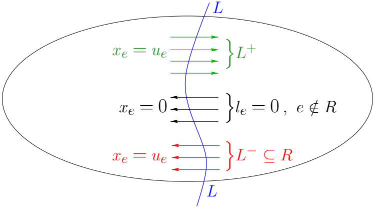
Cut Packing

- Solve $\mathrm{gap}(D) = 0$ subnetwork using extension of A. Frank '99 Abstract SP.
- Since restr. subnetwork is cut pack, it's blocked by a SP l.

P-D for Path and Cut Packing 3

Path Packing

- Solve $\text{gap}(P) = 0$ subnetwork using extension of Mc '95 Abstract MF.
- Since restr. subnetwork is MF, it's blocked by a Min Cut l.
- Here l is 0, ± 1:



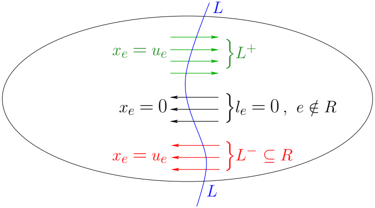
Cut Packing

- Solve $\text{gap}(D) = 0$ subnetwork using extension of A. Frank '99 Abstract SP.
- Since restr. subnetwork is cut pack, it's blocked by a SP l.

P-D for Path and Cut Packing 3

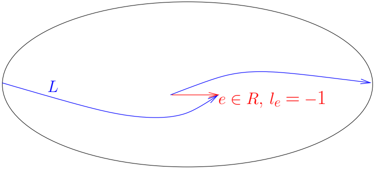
Path Packing

- Solve $\text{gap}(P) = 0$ subnetwork using extension of Mc '95 Abstract MF.
- Since restr. subnetwork is MF, it's blocked by a Min Cut l.
- Here l is 0, ± 1:



Cut Packing

- Solve $\text{gap}(D) = 0$ subnetwork using extension of A. Frank '99 Abstract SP.
- Since restr. subnetwork is cut pack, it's blocked by a SP l.
- Here l is 0, ± 1:



P-D for Path and Cut Packing 4

P-D for Path and Cut Packing 4

Path Packing

- Restricted subnetwork uses original x, auxiliary dual l.

P-D for Path and Cut Packing 4

Path Packing

- Restricted subnetwork uses original x, auxiliary dual l.

Cut Packing

- Restricted subnetwork uses original x, auxiliary dual l.

P-D for Path and Cut Packing 4

Path Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.

Cut Packing

- Restricted subnetwork uses original x, auxiliary dual l.

P-D for Path and Cut Packing 4

Path Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.

Cut Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.

P-D for Path and Cut Packing 4

Path Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$
  $\Longrightarrow \text{gap}'(P) \longleftarrow \text{gap}(P) + \theta(l(P) - 1)$.

Cut Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.

P-D for Path and Cut Packing 4

Path Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$

$\Longrightarrow$ $\text{gap}'(P) \longleftarrow \text{gap}(P) + \theta(l(P) - 1)$.

Cut Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$

$\Longrightarrow$ $\text{gap}'(D) \longleftarrow \text{gap}(D) + \theta(l(D) - 1)$.

P-D for Path and Cut Packing 4

Path Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$
  $\Longrightarrow \text{gap}'(P) \longleftarrow \text{gap}(P) + \theta(l(P) - 1)$.
- Lemma: θ is always an integer.

Cut Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$
  $\Longrightarrow \text{gap}'(D) \longleftarrow \text{gap}(D) + \theta(l(D) - 1)$.

P-D for Path and Cut Packing 4

Path Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$
  $\Longrightarrow \text{gap}'(P) \longleftarrow \text{gap}(P) + \theta(l(P) - 1)$.
- Lemma: θ is always an integer.

Cut Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$
  $\Longrightarrow \text{gap}'(D) \longleftarrow \text{gap}(D) + \theta(l(D) - 1)$.
- Lemma: θ is always an integer.

P-D for Path and Cut Packing 4

Path Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$
  $\Longrightarrow \text{gap}'(P) \longleftarrow \text{gap}(P) + \theta(l(P) - 1)$.
- Lemma: θ is always an integer.
  - If θ is determined by $\text{gap}'(P) \geq 0$ as $\theta = \text{gap}(P)/(1 - l(P))$ then $l(P) = 0$.

Cut Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$
  $\Longrightarrow \text{gap}'(D) \longleftarrow \text{gap}(D) + \theta(l(D) - 1)$.
- Lemma: θ is always an integer.

P-D for Path and Cut Packing 4

Path Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$

  $\Longrightarrow \text{gap}'(P) \longleftarrow \text{gap}(P) + \theta(l(P) - 1)$.
- Lemma: θ is always an integer.
  - If θ is determined by $\text{gap}'(P) \geq 0$ as $\theta = \text{gap}(P)/(1 - l(P))$ then $l(P) = 0$.

Cut Packing

- Restricted subnetwork uses original x, auxiliary dual l.
- Thus x is automatically updated.
- Update $y' \longleftarrow y + \theta l$
  $\lambda' \longleftarrow \lambda - \theta$

  $\Longrightarrow \text{gap}'(D) \longleftarrow \text{gap}(D) + \theta(l(D) - 1)$.
- Lemma: θ is always an integer.
  - If θ is determined by $\text{gap}'(D) \geq 0$ as $\theta = \text{gap}(D)/(1 - l(D))$ then $l(D) = 0$.

P-D for Path and Cut Packing 5

P-D for Path and Cut Packing 5

Path Packing

- Each solve of Restr. Abstract MF is polynomial.

P-D for Path and Cut Packing 5

Path Packing

- Each solve of Restr. Abstract MF is polynomial.

Cut Packing

- Each solve of Restr. Abstract Cut Pack is polynomial.

P-D for Path and Cut Packing 5

Path Packing

- Each solve of Restr. Abstract MF is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.

Cut Packing

- Each solve of Restr. Abstract Cut Pack is polynomial.

P-D for Path and Cut Packing 5

Path Packing

- Each solve of Restr. Abstract MF is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.

Cut Packing

- Each solve of Restr. Abstract Cut Pack is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.

P-D for Path and Cut Packing 5

Path Packing

- Each solve of Restr. Abstract MF is polynomial.
- x stays same at most n consecutive solves $\Longrightarrow$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.

Cut Packing

- Each solve of Restr. Abstract Cut Pack is polynomial.
- x stays same at most n consecutive solves $\Longrightarrow$ $O(nr_{\max})$ solves.

P-D for Path and Cut Packing 5

Path Packing

- Each solve of Restr. Abstract MF is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.

Cut Packing

- Each solve of Restr. Abstract Cut Pack is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.

P-D for Path and Cut Packing 5

Path Packing

- Each solve of Restr. Abstract MF is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.
- Make weakly polynomial via bit scaling $\implies$ sensitivity analysis.

Cut Packing

- Each solve of Restr. Abstract Cut Pack is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.

P-D for Path and Cut Packing 5

Path Packing

- Each solve of Restr. Abstract MF is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.
- Make weakly polynomial via bit scaling $\implies$ sensitivity analysis.

Cut Packing

- Each solve of Restr. Abstract Cut Pack is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.
- Make weakly polynomial via bit scaling $\implies$ sensitivity analysis.

P-D for Path and Cut Packing 5

Path Packing

- Each solve of Restr. Abstract MF is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.
- Make weakly polynomial via bit scaling $\implies$ sensitivity analysis.
- Theorem: This algorithm solves Weighted Abstract Flow in weakly polynomial time.

Cut Packing

- Each solve of Restr. Abstract Cut Pack is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.
- Make weakly polynomial via bit scaling $\implies$ sensitivity analysis.

P-D for Path and Cut Packing 5

Path Packing

- Each solve of Restr. Abstract MF is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.
- Make weakly polynomial via bit scaling $\implies$ sensitivity analysis.
- Theorem: This algorithm solves Weighted Abstract Flow in weakly polynomial time.

Cut Packing

- Each solve of Restr. Abstract Cut Pack is polynomial.
- x stays same at most n consecutive solves $\implies$ $O(nr_{\max})$ solves.
- This gives a *pseudo-polynomial* bound.
- Make weakly polynomial via bit scaling $\implies$ sensitivity analysis.
- Theorem: This algorithm solves Weighted Abstract Cut Packing in weakly polynomial time.

Outline

Abstract Flows over Time

- Question: can we extend the max flow version of WAF to flows over time ("dynamic flows") with time horizon T?

Abstract Flows over Time

- Question: can we extend the max flow version of WAF to flows over time ("dynamic flows") with time horizon T?
- Answer (J. Matuschke): Yes, via extending Ford and Fulkerson's ideas about flows over time.

Abstract Flows over Time

- Question: can we extend the max flow version of WAF to flows over time ("dynamic flows") with time horizon T?
- Answer (J. Matuschke): Yes, via extending Ford and Fulkerson's ideas about flows over time.
- For ordinary networks, can compute flows over time via a time-expanded network.

Abstract Flows over Time

- Question: can we extend the max flow version of WAF to flows over time ("dynamic flows") with time horizon T?
- Answer (J. Matuschke): Yes, via extending Ford and Fulkerson's ideas about flows over time.
- For ordinary networks, can compute flows over time via a time-expanded network.
  - Ok, but the size of the time-expanded network is pseudo-polynomial in T :-(

Abstract Flows over Time

- Question: can we extend the max flow version of WAF to flows over time ("dynamic flows") with time horizon T?
- Answer (J. Matuschke): Yes, via extending Ford and Fulkerson's ideas about flows over time.
- For ordinary networks, can compute flows over time via a time-expanded network.
  - Ok, but the size of the time-expanded network is pseudo-polynomial in T :-(
- **F&F idea:** Compute a max-reward flow in a (polynomial-sized) static network, then repeat this flow over time.

Abstract Flows over Time

- Question: can we extend the max flow version of WAF to flows over time ("dynamic flows") with time horizon T?
- Answer (J. Matuschke): Yes, via extending Ford and Fulkerson's ideas about flows over time.
- For ordinary networks, can compute flows over time via a time-expanded network.
    - Ok, but the size of the time-expanded network is pseudo-polynomial in T :-(
- **F&F idea:** Compute a max-reward flow in a (polynomial-sized) static network, then repeat this flow over time.
- Same idea works for abstract networks, but need to repeat path flows over time.

The Static Abstract Network

- Now each element e has a time delay τ_e, so it takes time $\tau(P) = \sum_{e \in P} \tau_e$ for flow to traverse path P.

The Static Abstract Network

- Now each element e has a time delay τ_e, so it takes time $\tau(P) = \sum_{e \in P} \tau_e$ for flow to traverse path P.
- What is our reward for putting flow x_P on P in a static abstract network?

The Static Abstract Network

- Now each element e has a time delay τ_e, so it takes time $\tau(P) = \sum_{e \in P} \tau_e$ for flow to traverse path P.
- What is our reward for putting flow x_P on P in a static abstract network?
- We can repeat flow x_P until time $r(P) \equiv T - \tau(P)$.

The Static Abstract Network

- Now each element e has a time delay τ_e, so it takes time $\tau(P) = \sum_{e \in P} \tau_e$ for flow to traverse path P.
- What is our reward for putting flow x_P on P in a static abstract network?
- We can repeat flow x_P until time $r(P) \equiv T - \tau(P)$.
- In order to maximize abstract flow over time, we want to repeat as much flow as possible as long as possible, i.e., $\max \sum_P r(P) x_P$.

The Static Abstract Network

- Now each element e has a time delay τ_e, so it takes time $\tau(P) = \sum_{e \in P} \tau_e$ for flow to traverse path P.
- What is our reward for putting flow x_P on P in a static abstract network?
- We can repeat flow x_P until time $r(P) \equiv T - \tau(P)$.
- In order to maximize abstract flow over time, we want to repeat as much flow as possible as long as possible, i.e., $\max \sum_P r(P) x_P$.

Lemma

This $r(P)$ is supermodular.

The Static Abstract Network

- Now each element e has a time delay τ_e, so it takes time $\tau(P) = \sum_{e \in P} \tau_e$ for flow to traverse path P.
- What is our reward for putting flow x_P on P in a static abstract network?
- We can repeat flow x_P until time $r(P) \equiv T - \tau(P)$.
- In order to maximize abstract flow over time, we want to repeat as much flow as possible as long as possible, i.e., $\max \sum_P r(P) x_P$.

Lemma

This $r(P)$ is supermodular.

- Thus we can solve max abstract flow over time in polynomial time (modulo lots of details).

Reconsidering Supermodularity of r

- Recall that no "real" application of supermodularity of r was known.

Reconsidering Supermodularity of r

- Recall that no "real" application of supermodularity of r was known.
  - It is needed for transportation problems, but they use modular r.

Reconsidering Supermodularity of r

- Recall that no “real” application of supermodularity of r was known.
  - It is needed for transportation problems, but they use modular r.
- This application to max abstract flow finally gives us an application where the supermodularity was really necessary.

Parametric Abstract Flow?

- There is a big literature on MF/MC with capacities parametric in scalar λ with lots of applications.

Parametric Abstract Flow?

- There is a big literature on MF/MC with capacities parametric in scalar λ with lots of applications.
- General result by Topkis: Consider $\min f(x, \lambda)$ with x on a lattice. Suppose that

Parametric Abstract Flow?

- There is a big literature on MF/MC with capacities parametric in scalar λ with lots of applications.
- General result by Topkis: Consider $\min f(x, \lambda)$ with x on a lattice. Suppose that
    1. $f(x, \lambda)$ is submodular in x for each fixed λ, and

Parametric Abstract Flow?

- There is a big literature on MF/MC with capacities parametric in scalar λ with lots of applications.
- General result by Topkis: Consider $\min f(x, \lambda)$ with x on a lattice. Suppose that
  1. $f(x, \lambda)$ is submodular in x for each fixed λ, and
  2. For all $x \preceq y$ and $\lambda_1 \leq \lambda_2$ we have Decreasing Differences $f(y, \lambda_2) - f(y, \lambda_1) \leq f(x, \lambda_2) - f(x, \lambda_1)$.

Parametric Abstract Flow?

- There is a big literature on MF/MC with capacities parametric in scalar λ with lots of applications.
- General result by Topkis: Consider $\min f(x, \lambda)$ with x on a lattice. Suppose that
  1. $f(x, \lambda)$ is submodular in x for each fixed λ, and
  2. For all $x \preceq y$ and $\lambda_1 \leq \lambda_2$ we have Decreasing Differences $f(y, \lambda_2) - f(y, \lambda_1) \leq f(x, \lambda_2) - f(x, \lambda_1)$.
- Then there are monotone optimal solutions $x^*(\lambda)$ such that for $\lambda_1 \leq \lambda_2$ we have $x^*(\lambda_1) \preceq x^*(\lambda_2)$.

Parametric Abstract Flow?

- There is a big literature on MF/MC with capacities parametric in scalar λ with lots of applications.
- General result by Topkis: Consider $\min f(x, \lambda)$ with x on a lattice. Suppose that
  1. $f(x, \lambda)$ is submodular in x for each fixed λ, and
  2. For all $x \preceq y$ and $\lambda_1 \leq \lambda_2$ we have Decreasing Differences $f(y, \lambda_2) - f(y, \lambda_1) \leq f(x, \lambda_2) - f(x, \lambda_1)$.
- Then there are monotone optimal solutions $x^*(\lambda)$ such that for $\lambda_1 \leq \lambda_2$ we have $x^*(\lambda_1) \preceq x^*(\lambda_2)$.
- When we specialize to MF/MC we find that min cuts are nested in λ (when parametric capacities satisfy (2)).

Parametric Abstract Flow?

- There is a big literature on MF/MC with capacities parametric in scalar λ with lots of applications.
- General result by Topkis: Consider $\min f(x,\lambda)$ with x on a lattice. Suppose that
  1. $f(x,\lambda)$ is submodular in x for each fixed λ, and
  2. For all $x \preceq y$ and $\lambda_1 \leq \lambda_2$ we have Decreasing Differences $f(y,\lambda_2) - f(y,\lambda_1) \leq f(x,\lambda_2) - f(x,\lambda_1)$.
- Then there are monotone optimal solutions $x^*(\lambda)$ such that for $\lambda_1 \leq \lambda_2$ we have $x^*(\lambda_1) \preceq x^*(\lambda_2)$.
- When we specialize to MF/MC we find that min cuts are nested in λ (when parametric capacities satisfy (2)).
- Gallo, Grigoriadis, and Tarjan (GGT) considered such a class, and showed that you can compute *all* min cuts in $O(1)$ Push-Relabel time

Parametric Abstract Flow?

- There is a big literature on MF/MC with capacities parametric in scalar λ with lots of applications.
- General result by Topkis: Consider $\min f(x,\lambda)$ with x on a lattice. Suppose that
    1. $f(x,\lambda)$ is submodular in x for each fixed λ, and
    2. For all $x \preceq y$ and $\lambda_1 \le \lambda_2$ we have Decreasing Differences $f(y,\lambda_2) - f(y,\lambda_1) \le f(x,\lambda_2) - f(x,\lambda_1)$.
- Then there are monotone optimal solutions $x^*(\lambda)$ such that for $\lambda_1 \le \lambda_2$ we have $x^*(\lambda_1) \preceq x^*(\lambda_2)$.
- When we specialize to MF/MC we find that min cuts are nested in λ (when parametric capacities satisfy (2)).
- Gallo, Grigoriadis, and Tarjan (GGT) considered such a class, and showed that you can compute *all* min cuts in $O(1)$ Push-Relabel time
    - Extended by Gusfield and Martel; Mc; F. Granot, Mc, Queyranne, Tardella; . . .

Does Submodularity Generalize?

- In his book, Topkis shows some extensions to transportation problems under some circumstances, as submodularity of more general problems than MF/MC was not evident.

Does Submodularity Generalize?

- In his book, Topkis shows some extensions to transportation problems under some circumstances, as submodularity of more general problems than MF/MC was not evident.
- But it's not hard to show that in fact min-cost flow dual objective value is submodular in the node potentials (using $\min$ and $\max$ as $\vee$ and $\wedge$); proved by Murota.

Does Submodularity Generalize?

- In his book, Topkis shows some extensions to transportation problems under some circumstances, as submodularity of more general problems than MF/MC was not evident.
- But it's not hard to show that in fact min-cost flow dual objective value is submodular in the node potentials (using $\min$ and $\max$ as $\vee$ and $\wedge$); proved by Murota.
- This plus Topkis gives that GGT-structured parametric capacities lead to monotone dual optimal solutions for min-cost flow.

Does Submodularity Generalize?

- In his book, Topkis shows some extensions to transportation problems under some circumstances, as submodularity of more general problems than MF/MC was not evident.
- But it's not hard to show that in fact min-cost flow dual objective value is submodular in the node potentials (using $\min$ and $\max$ as $\vee$ and $\wedge$); proved by Murota.
- This plus Topkis gives that GGT-structured parametric capacities lead to monotone dual optimal solutions for min-cost flow.
- WAF generalizes min-cost flow, hmmmm ...

Does Submodularity Generalize?

- In his book, Topkis shows some extensions to transportation problems under some circumstances, as submodularity of more general problems than MF/MC was not evident.
- But it's not hard to show that in fact min-cost flow dual objective value is submodular in the node potentials (using $\min$ and $\max$ as $\vee$ and $\wedge$); proved by Murota.
- This plus Topkis gives that GGT-structured parametric capacities lead to monotone dual optimal solutions for min-cost flow.
- WAF generalizes min-cost flow, hmmmm . . .
- Matuschke and Peis conjecture that we can show GGT-type results also for max flow versions of abstract flow.

Does Submodularity Generalize?

- In his book, Topkis shows some extensions to transportation problems under some circumstances, as submodularity of more general problems than MF/MC was not evident.
- But it's not hard to show that in fact min-cost flow dual objective value is submodular in the node potentials (using $\min$ and $\max$ as $\vee$ and $\wedge$); proved by Murota.
- This plus Topkis gives that GGT-structured parametric capacities lead to monotone dual optimal solutions for min-cost flow.
- WAF generalizes min-cost flow, hmmmm ...
- Matuschke and Peis conjecture that we can show GGT-type results also for max flow versions of abstract flow.
- Then parametric abstract flows over time :-) ?

Conclusions

1. We found the first combinatorial polynomial algorithms for Weighted Abstract Flow and Cut Packing.

Conclusions

1. We found the first combinatorial polynomial algorithms for Weighted Abstract Flow and Cut Packing.
2. Can we get a combinatorial faster, or even strongly polynomial algorithm? Maybe some version of *Min Mean Cycle*?

Conclusions

1. We found the first combinatorial polynomial algorithms for Weighted Abstract Flow and Cut Packing.
2. Can we get a combinatorial faster, or even strongly polynomial algorithm? Maybe some version of *Min Mean Cycle*?
3. Gröflin and Hoffman extended lattice polyhedra to 0, ± 1 matrices and to a version with sub- and super-modular interchanged; can we adapt our algorithm for these?

Conclusions

1. We found the first combinatorial polynomial algorithms for Weighted Abstract Flow and Cut Packing.
2. Can we get a combinatorial faster, or even strongly polynomial algorithm? Maybe some version of *Min Mean Cycle*?
3. Gröflin and Hoffman extended lattice polyhedra to 0, ± 1 matrices and to a version with sub- and super-modular interchanged; can we adapt our algorithm for these?
4. Typically for such problems, figuring out how to represent the problem is a big hurdle; here we suppressed details of the oracles we are using.

Conclusions

1. We found the first combinatorial polynomial algorithms for Weighted Abstract Flow and Cut Packing.
2. Can we get a combinatorial faster, or even strongly polynomial algorithm? Maybe some version of *Min Mean Cycle*?
3. Gröflin and Hoffman extended lattice polyhedra to 0, ± 1 matrices and to a version with sub- and super-modular interchanged; can we adapt our algorithm for these?
4. Typically for such problems, figuring out how to represent the problem is a big hurdle; here we suppressed details of the oracles we are using.
5. Could we further extend this idea to solve, e.g., Schrijver's general framework for TDI problems?

Conclusions

1. We found the first combinatorial polynomial algorithms for Weighted Abstract Flow and Cut Packing.
2. Can we get a combinatorial faster, or even strongly polynomial algorithm? Maybe some version of *Min Mean Cycle*?
3. Gröflin and Hoffman extended lattice polyhedra to 0, ± 1 matrices and to a version with sub- and super-modular interchanged; can we adapt our algorithm for these?
4. Typically for such problems, figuring out how to represent the problem is a big hurdle; here we suppressed details of the oracles we are using.
5. Could we further extend this idea to solve, e.g., Schrijver's general framework for TDI problems?
6. One can make a good career out of answering open questions in Alan's papers :-)

Dedication

I dedicate this talk to
Alan Hoffman's 90th birthday,
and to his long and fruitful career.

Questions?

Comments?