Maximum-Life Routing Schedule

Peng-Jun Wan

wan@cs.iit.edu

Outline

- Problem Description
- Min-Cost Routing
- Ellipsoid Algorithm
- Price-Directive Algorithm
- Flow-Based Algorithm

A Motivating Example

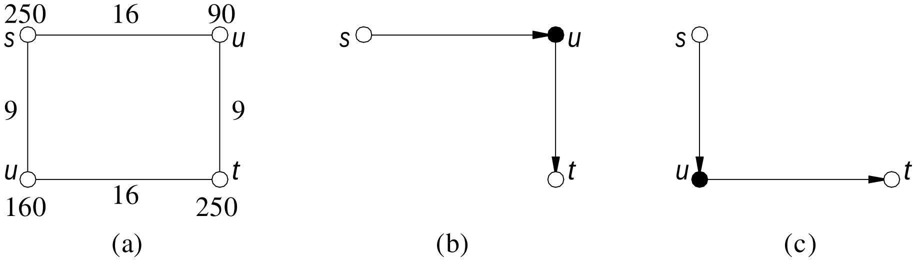

Figure: Consider the unicast from s to t in (a). If only one path in either (b) or (c) is used, the life is 10. On the other hand, we can use both paths for 10 time units each to achieve an overall life of 20.

Network model

- Communication topology: $D = (V, A; c)$
- Adjustable transmission power
- Power consumption: same as in the previous chapter
  - Receiving power consumption is ignored

Communication Tasks

- Concurrent Unicasts
- Aggregation
- Broadcast
- Multicast

Communication Tasks

- Concurrent Unicasts
- Aggregation
- Broadcast
- Multicast

$\mathcal{R}$: a collection of routes for a given communication task

Routing Schedule

- $b \in \mathbb{R}_+^V$: energy budget function
- A routing schedule is a set of pairs $(H_i, x_i) \in \mathcal{R} \times \mathbb{R}_+$ for $i = 1, \cdots, m$ satisfying that

$$\sum_{i=1}^{m} p_{H_i}(u)\, x_i \leq b(u), \forall u \in V.$$

Routing Schedule

- $b \in \mathbb{R}_+^V$: energy budget function
- A routing schedule is a set of pairs $(H_i, x_i) \in \mathcal{R} \times \mathbb{R}_+$ for $i = 1, \cdots, m$ satisfying that

$$\sum_{i=1}^{m} p_{H_i}(u) x_i \leq b(u), \forall u \in V.$$

- The life (or length) of this schedule is $\sum_{i=1}^{m} x_i$.

Max-Life Routing Schedule (**MLRS**): finding a routing schedule of maximum life.

Max-Life Routing Schedule

Max-Life Routing Schedule (**MLRS**): finding a routing schedule of maximum life.

$$\begin{array}{ll} \max & \sum_{H \in \mathcal{R}} x_H \\ s.t. & \sum_{H \in \mathcal{R}} x_H p_H(u) \leq b(u), \forall u \in V; \\ & x_H \geq 0, \forall H \in \mathcal{R}. \end{array}$$

Max-Life Routing Schedule

Max-Life Routing Schedule (**MLRS**): finding a routing schedule of maximum life.

$$\begin{array}{ll} \max & \sum_{H\in\mathcal{R}} x_H \\ s.t. & \sum_{H\in\mathcal{R}} x_H p_H(u) \leq b(u), \forall u \in V; \\ & x_H \geq 0, \forall H \in \mathcal{R}. \end{array}$$

- $|V| = n$ constraints $\Rightarrow \exists$ an optimal solution using at most n routes.

Max-Life Routing Schedule (**MLRS**): finding a routing schedule of maximum life.

$$\begin{array}{ll} \max & \sum_{H\in\mathcal{R}} x_H \\ s.t. & \sum_{H\in\mathcal{R}} x_H p_H(u) \leq b(u), \forall u \in V; \\ & x_H \geq 0, \forall H \in \mathcal{R}. \end{array}$$

- $|V| = n$ constraints $\Rightarrow \exists$ an optimal solution using at most n routes.
- # of variables $|\mathcal{R}|$ is exponential $\Rightarrow$ standard LP solvers are not practical.

Summary on Algorithms

- Ellipsoid Algorithm (EA)
- Price-Directive Algorithm (PDA)
- Flow-Based Algorithm (FBA)

	EA	PDA	FBA
Conc. Unicasts	exact	$1+\varepsilon$	exact
Aggregation	exact	$1+\varepsilon$	exact
Broadcast	$2H(n-1)-1$	$(1+\varepsilon)(2H(n-1)-1)$	N/A
Multicast	$O(k^{\varepsilon})$	$O(k^{\varepsilon})$	N/A

Roadmap

- Problem Description
- **Min-Cost Routing**
- Ellipsoid Algorithm
- Price-Directive Algorithm
- Flow-Based Algorithm

Min-Cost Routing

- $y \in \mathbb{R}_+^V$: a price function
- H: a subgraph of D

$$\text{cost of } H \text{ w.r.t. } y = \sum_{u \in V} y(u)\, p_H(u)$$

Min-Cost Routing

- $y \in \mathbb{R}_+^V$: a price function
- H: a subgraph of D

$$\text{cost of } H \text{ w.r.t. } y = \sum_{u \in V} y(u)\, p_H(u)$$

Min-Cost Routing (**MCR**): find an $H \in \mathcal{R}$ of minimum cost w.r.t y.

Min-Cost Routing

- $y \in \mathbb{R}_+^V$: a price function
- H: a subgraph of D

$$\text{cost of } H \text{ w.r.t. } y = \sum_{u \in V} y(u)\, p_H(u)$$

Min-Cost Routing (**MCR**): find an $H \in \mathcal{R}$ of minimum cost w.r.t y.

A generalization of **Min-Power Routing**

(Approximation) Algorithms for MCR

By applying the algorithms developed in the previous chapter for **MPR**, we immediately have the following algorithmic results:

1. Concurrent Unicasts: polynomial
2. Aggregation: polynomial
3. Broadcast: $(2H(n-1)-1)$-approximation algorithm
4. Multicast: $O(k^{\varepsilon})$-approximation algorithm for any fixed $\varepsilon > 0$

MCR vs. Dual of MLRS

- Dual of **MLRS**:

$$\begin{array}{ll} \min & \sum_{u \in V} b(u)\, y(u) \\ s.t. & \sum_{u \in V} p_H(u)\, y(u) \geq 1, \forall H \in \mathcal{R} \\ & y(u) \geq 0, \forall u \in V \end{array}$$

MCR vs. Dual of MLRS

- Dual of **MLRS**:

$$\begin{array}{ll} \min & \sum_{u \in V} b(u)\, y(u) \\ s.t. & \sum_{u \in V} p_H(u)\, y(u) \geq 1, \forall H \in \mathcal{R} \\ & y(u) \geq 0, \forall u \in V \end{array}$$

- **MCR**: separation problem of the dual of **MLRS**

Max-Life vs. Min-Cost

- *opt*: life of a max-life routing schedule.
- For any price function $y \in \mathbb{R}_{+}^{V}$, let

$$\alpha(y) = \min_{H \in \mathcal{R}} \sum_{u \in V} p_H(u)\, y(u) : \text{ min-cost of routes in } \mathcal{R} \text{ w.r.t.} y,$$

$$\beta(y) = \sum_{u \in V} b(u)\, y(u) : \text{ total energy cost w.r.t. } y.$$

Max-Life vs. Min-Cost

- *opt*: life of a max-life routing schedule.
- For any price function $y \in \mathbb{R}_+^V$, let

$$\alpha(y) = \min_{H \in \mathcal{R}} \sum_{u \in V} p_H(u)\, y(u): \text{ min-cost of routes in } \mathcal{R} \text{ w.r.t.} y,$$

$$\beta(y) = \sum_{u \in V} b(u)\, y(u): \text{ total energy cost w.r.t. } y.$$

Lemma

For any $y \in \mathbb{R}_+^V$, $\alpha(y) \leq \frac{\beta(y)}{opt}$. In addition, there exists some $y \in \mathbb{R}_+^V$ such that $\alpha(y) = \frac{\beta(y)}{opt}$.

Max-Life vs. Min-Cost

First Part: Trivial if $\alpha(y) = 0$. So, we assume that $\alpha(y) > 0$. Then, $\frac{y}{\alpha(y)}$ is a feasible solution of the dual LP. Hence

$$opt \leq \beta\left(\frac{y}{\alpha(y)}\right) = \frac{\beta(y)}{\alpha(y)} \Rightarrow \alpha(y) \leq \frac{\beta(y)}{opt}.$$

First Part: Trivial if $\alpha(y) = 0$. So, we assume that $\alpha(y) > 0$. Then, $\frac{y}{\alpha(y)}$ is a feasible solution of the dual LP. Hence

$$opt \leq \beta\left(\frac{y}{\alpha(y)}\right) = \frac{\beta(y)}{\alpha(y)} \Rightarrow \alpha(y) \leq \frac{\beta(y)}{opt}.$$

Second Part: Suppose y is an optimal solution to dual LP. Then,

$$opt = \beta(y) \text{ and } \alpha(y) = 1 \Rightarrow \alpha(y) = \frac{\beta(y)}{opt}.$$

Roadmap

- Problem Description
- Min-Cost Routing
- **Ellipsoid Algorithm**
- Price-Directive Algorithm
- Flow-Based Algorithm

Ellipsoid Method with (Approximate) Separation Oracle

$\mathcal{N}$: a network class

Theorem

Suppose that there is a polynomial (respectively, a polynomial μ-approximation) algorithm for **MCR** *for a communication task restricted to $\mathcal{N}$. Then, there is a polynomial (respectively, a polynomial μ-approximation) algorithm for* **MLRS** *for the same communication task restricted to $\mathcal{N}$.*

Ellipsoid Algorithm for MLRS

	Ellipsoid Algorithm
Conc. Unicasts	exact
Aggregation	exact
Broadcast	$2H(n-1)-1$
Multicast	$O(k^{\varepsilon})$

Drawback: very slow practically

Roadmap

- Problem Description
- Min-Cost Routing
- Ellipsoid Algorithm
- **Price-Directive Algorithm**
- Flow-Based Algorithm

Basic Idea

An iterative algorithm, in each iteration:

- Set the prices of the nodes with low residue energy relatively higher
- Nodes with low residue energy are protected from getting drained of energy quickly
- Nodes with high residue energy are enforced to contribute more energy

Basic Idea

An iterative algorithm, in each iteration:

- Set the prices of the nodes with low residue energy relatively higher
- Nodes with low residue energy are protected from getting drained of energy quickly
- Nodes with high residue energy are enforced to contribute more energy

Challenge: how to choose the prices properly?

Parameters

- $\mathcal{A}$: a μ-approximation algorithm for **MCR**
  - if $\mu = 1$, the algorithm $\mathcal{A}$ is optimal for **MCR**
- ε: a constant parameter $\in (0, 1)$
- output: an $(1 + \varepsilon)\,\mu$-approximation.

Variables

- $\mathcal{H}$: the set of chosen routes;
- x_H for each $H \in \mathcal{H}$: the duration of H;

Variables

- $\mathcal{H}$: the set of chosen routes;
- x_H for each $H \in \mathcal{H}$: the duration of H;
- $z \in \mathbb{R}_+^V$: the energy consumption percentage vector defined by

$$z(u) = \frac{\sum_{H \in \mathcal{H}} x_H p_H(u)}{b(u)}, \forall u \in V;$$

- ϕ: the maximum energy consumption percentage $\max_{u \in V} z(u)$;

Variables

- $\mathcal{H}$: the set of chosen routes;
- x_H for each $H \in \mathcal{H}$: the duration of H;
- $z \in \mathbb{R}_+^V$: the energy consumption percentage vector defined by

$$z(u) = \frac{\sum_{H \in \mathcal{H}} x_H p_H(u)}{b(u)}, \forall u \in V;$$

- ϕ: the maximum energy consumption percentage $\max_{u \in V} z(u)$;
- $y \in \mathbb{R}_+^V$: the price vector;
- β: the total energy cost $\sum_{u \in V} b(u) y(u)$.

Outline of PDA

$\mathcal{H} \leftarrow \emptyset; \forall u \in V, z(u) \leftarrow 0; \phi \leftarrow 0;$
$\forall u \in V, y(u) \leftarrow \frac{1}{b(u)}; \beta \leftarrow n;$

repeat

  compute an $H \in \mathcal{R}$ using $\mathcal{A}$ on (D, y);
  $t \leftarrow \min_{v \in V} b(v) / p_H(v);$
  if $H \in \mathcal{H}$ then $x_H \leftarrow x_H + t$,
    else $\mathcal{H} \leftarrow \mathcal{H} \cup \{H\}$ and $x_H \leftarrow t$;
  $\forall u \in V, \ z(u) \leftarrow z(u) + t \frac{p_H(u)}{b(u)};$
  $\phi \leftarrow \max_{u \in V} z(u);$
  $\forall u \in V, \ y(u) \leftarrow y(u) \left(1 + \varepsilon t \frac{p_H(u)}{b(u)}\right);$
  $\beta \leftarrow \sum_{u \in V} b(u) y(u);$

until $0 < \phi \leq \frac{1+\varepsilon}{\varepsilon} \ln \frac{\beta}{n};$
Output $\{(H, x_H / \phi) : H \in \mathcal{H}\}$.

Running Time And Approximation Bound

Theorem

The algorithm **PDA** *produces an* $(1+\varepsilon)\,\mu$*-approximation in at most* $K = n\left\lceil \frac{(1+\varepsilon)\ln n}{(1+\varepsilon)\ln(1+\varepsilon)-\varepsilon} \right\rceil$ *iterations.*

Running Time And Approximation Bound

Theorem

The algorithm **PDA** *produces an* $(1+\varepsilon)\,\mu$*-approximation in at most* $K = n\left\lceil \frac{(1+\varepsilon)\ln n}{(1+\varepsilon)\ln(1+\varepsilon)-\varepsilon} \right\rceil$ *iterations.*

	PDA
Conc. Unicasts	$1+\varepsilon$
Aggregation	$1+\varepsilon$
Broadcast	$(1+\varepsilon)\,(2H\,(n-1)-1)$
Multicast	$O\left(k^{\varepsilon}\right)$

Notations

- *opt*: life of an optimal solution

Notations

- opt: life of an optimal solution
- $\mathcal{H}_0$, z_0, ϕ_0, y_0 and β_0: initial values of $\mathcal{H}$, z, ϕ, y and β resp.

Notations

- *opt*: life of an optimal solution
- $\mathcal{H}_0$, z_0, ϕ_0, y_0 and β_0: initial values of $\mathcal{H}$, z, ϕ, y and β resp.
- $\mathcal{H}_j$, z_j, ϕ_j, y_j and β_j: values of $\mathcal{H}$, z, ϕ, y and β resp. at the end of the j-th iteration for each $j \geq 1$

Notations

- *opt*: life of an optimal solution
- $\mathcal{H}_0$, z_0, ϕ_0, y_0 and β_0: initial values of $\mathcal{H}$, z, ϕ, y and β resp.
- $\mathcal{H}_j$, z_j, ϕ_j, y_j and β_j: values of $\mathcal{H}$, z, ϕ, y and β resp. at the end of the j-th iteration for each $j \geq 1$
- H_j: route selected in the j-th iteration

Notations

- opt: life of an optimal solution
- $\mathcal{H}_0$, z_0, ϕ_0, y_0 and β_0: initial values of $\mathcal{H}$, z, ϕ, y and β resp.
- $\mathcal{H}_j$, z_j, ϕ_j, y_j and β_j: values of $\mathcal{H}$, z, ϕ, y and β resp. at the end of the j-th iteration for each $j \geq 1$
- H_j: route selected in the j-th iteration
- t_j: value of t computed in the j-th iteration

Notations

- opt: life of an optimal solution
- $\mathcal{H}_0$, z_0, ϕ_0, y_0 and β_0: initial values of $\mathcal{H}$, z, ϕ, y and β resp.
- $\mathcal{H}_j$, z_j, ϕ_j, y_j and β_j: values of $\mathcal{H}$, z, ϕ, y and β resp. at the end of the j-th iteration for each $j \geq 1$
- H_j: route selected in the j-th iteration
- t_j: value of t computed in the j-th iteration
- $\tau_j = \max_{u \in V} y_j(u) b(u)$: maximum energy cost of all nodes at the end of j-th iteration

Upper Bound on ϕ_j

Claim: $\phi_j \leq \log_{1+\varepsilon} \tau_j$ for each $j \geq 1$.

Upper Bound on ϕ_j

Claim: $\phi_j \leq \log_{1+\varepsilon} \tau_j$ for each $j \geq 1$.

Lemma

For any $\varepsilon > 0$ and $0 \leq t \leq 1$, $t \leq \log_{1+\varepsilon}(1 + \varepsilon t)$.

Upper Bound on ϕ_j

Claim: $\phi_j \leq \log_{1+\varepsilon} \tau_j$ for each $j \geq 1$.

Lemma

For any $\varepsilon > 0$ and $0 \leq t \leq 1$, $t \leq \log_{1+\varepsilon} (1 + \varepsilon t)$.

$$z_j(u) - z_{j-1}(u) \leq \log_{1+\varepsilon} (1 + \varepsilon (z_j(u) - z_{j-1}(u))) = \log_{1+\varepsilon} \frac{y_j(u)}{y_{j-1}(u)},$$

$$\Rightarrow z_j(u) \leq \log_{1+\varepsilon} \frac{y_j(u)}{y_0(u)} = \log_{1+\varepsilon} (y_j(u) b(u)) \leq \log_{1+\varepsilon} \tau_j.$$

Running Time

Prove by *contradiction*: assume $> K$ iterations.

Running Time

Prove by *contradiction*: assume $> K$ iterations.

- Among the first K iterations some node v appears as a "bottleneck" node in at least K/n iterations.

Running Time

Prove by *contradiction*: assume $> K$ iterations.

- Among the first K iterations some node v appears as a "bottleneck" node in at least K/n iterations.
- In each of K/n iterations, $y(v)$ is increased by a factor of $1+\varepsilon$.

Running Time

Prove by *contradiction*: assume $> K$ iterations.

- Among the first K iterations some node v appears as a "bottleneck" node in at least K/n iterations.
- In each of K/n iterations, $y(v)$ is increased by a factor of $1+\varepsilon$.

$$y_K(v) \geq y_0(v)(1+\varepsilon)^{K/n} = \frac{(1+\varepsilon)^{K/n}}{b(v)}$$

$$\Rightarrow \tau_K \geq y_k(v)\, b(v) \geq (1+\varepsilon)^{K/n}$$

$$\Rightarrow \frac{\phi_K}{\ln \frac{\beta_K}{n}} \leq \frac{\log_{1+\varepsilon} \tau_K}{\ln \frac{\tau_K}{n}} = \frac{1}{\ln(1+\varepsilon) - \frac{\ln n}{\log_{1+\varepsilon} \tau_K}} \leq \frac{1+\varepsilon}{\varepsilon}$$

Running Time

Prove by *contradiction*: assume $> K$ iterations.

- Among the first K iterations some node v appears as a "bottleneck" node in at least K/n iterations.
- In each of K/n iterations, $y(v)$ is increased by a factor of $1+\varepsilon$.

$$y_K(v) \geq y_0(v)(1+\varepsilon)^{K/n} = \frac{(1+\varepsilon)^{K/n}}{b(v)}$$

$$\Rightarrow \tau_K \geq y_k(v)\, b(v) \geq (1+\varepsilon)^{K/n}$$

$$\Rightarrow \frac{\phi_K}{\ln \frac{\beta_K}{n}} \leq \frac{\log_{1+\varepsilon} \tau_K}{\ln \frac{\tau_K}{n}} = \frac{1}{\ln(1+\varepsilon) - \frac{\ln n}{\log_{1+\varepsilon} \tau_K}} \leq \frac{1+\varepsilon}{\varepsilon}$$

- By the stopping rule, the number of iterations $\leq K$, which is a contradiction.

Correctness of The output Solution

k: number of iterations.

Claim: By the end of the j-th iteration for $1 \leq j \leq k$, the energy consumption percentage of each node u is $z_j(u)$, i.e.,

$$z_j(u) = \frac{\sum_{H \in \mathcal{H}_j} x_H p_H(u)}{b(u)}, \forall u \in V.$$

Correctness of The output Solution

k: number of iterations.

Claim: By the end of the j-th iteration for $1 \leq j \leq k$, the energy consumption percentage of each node u is $z_j(u)$, i.e.,

$$z_j(u) = \frac{\sum_{H \in \mathcal{H}_j} x_H p_H(u)}{b(u)}, \forall u \in V.$$

Therefore, the final scaling by a factor ϕ_k results in a feasible solution.

Lower Bound on t_j

Claim: $t_j \geq \frac{1}{\varepsilon\mu}\frac{\beta_j - \beta_{j-1}}{\beta_{j-1}} opt$ for each $1 \leq j \leq k$,

Lower Bound on t_j

Claim: $t_j \geq \frac{1}{\varepsilon\mu}\frac{\beta_j-\beta_{j-1}}{\beta_{j-1}} opt$ for each $1 \leq j \leq k$,

$$\begin{aligned}
\beta_j &= \sum_{u\in V} b(u)\, y_j(u) \\
&= \sum_{u\in V} b(u)\, y_{j-1}(u)\left(1+\varepsilon t_j \frac{p_{H_j}(u)}{b(u)}\right) \\
&= \sum_{u\in V} b(u)\, y_{j-1}(u) + \varepsilon t_j \left(\sum_{u\in V} p_{H_j}(u)\, y_{j-1}(u)\right) \\
&= \beta_{j-1} + \varepsilon t_j \left(\sum_{u\in V} p_{H_j}(u)\, y_{j-1}(u)\right) \\
&\leq \beta_{j-1} + \varepsilon t_j \cdot \mu \frac{\beta_{j-1}}{opt}.
\end{aligned}$$

Approximation Bound

$$\sum_{j=1}^{k} t_j \geq \frac{opt}{\varepsilon\mu} \sum_{j=1}^{k} \frac{\beta_j - \beta_{j-1}}{\beta_{j-1}} \geq \frac{opt}{\varepsilon\mu} \ln \frac{\beta_k}{\beta_0} = \frac{opt}{\varepsilon\mu} \ln \frac{\beta_k}{n},$$

Approximation Bound

$$\sum_{j=1}^{k} t_j \geq \frac{opt}{\varepsilon\mu} \sum_{j=1}^{k} \frac{\beta_j - \beta_{j-1}}{\beta_{j-1}} \geq \frac{opt}{\varepsilon\mu} \ln \frac{\beta_k}{\beta_0} = \frac{opt}{\varepsilon\mu} \ln \frac{\beta_k}{n},$$

So,

$$\frac{\sum_{j=1}^{k} t_j}{\phi_k} \geq \frac{1}{\varepsilon\mu} \frac{\ln \frac{\beta_k}{n}}{\phi_k} opt \geq \frac{1}{\varepsilon\mu} \frac{\varepsilon}{1+\varepsilon} opt = \frac{opt}{(1+\varepsilon)\,\mu}.$$

Roadmap

- Problem Description
- Min-Cost Routing
- Ellipsoid Algorithm
- Price-Directive Algorithm
- **Flow-Based Algorithm**

Single Flow

- $D = (V, A)$: a digraph with two distinct nodes s and t
- $f \in \mathbb{R}_+^A$ is an $s - t$ flow in D if

$$f\left(\delta^{out}(v)\right) = f\left(\delta^{in}(v)\right), \forall v \in V \setminus \{s, t\} \ \ (\textit{flow conservation law})$$

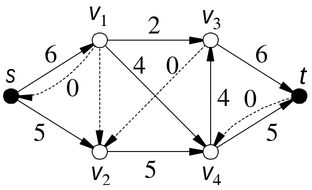

Figure: An an $s - t$ flow of value 11.

- Value of f: $val(f) = f\left(\delta^{out}(s)\right) - f\left(\delta^{in}(s)\right)$.

Single Flow

- $D = (V, A)$: a digraph with two distinct nodes s and t
- $f \in \mathbb{R}_+^A$ is an $s - t$ flow in D if

$$f\left(\delta^{out}(v)\right) = f\left(\delta^{in}(v)\right), \forall v \in V \setminus \{s, t\} \;\; (\textit{flow conservation law})$$

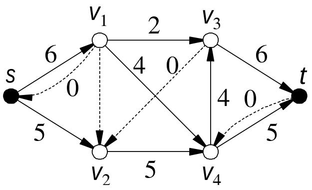

Figure: An an $s - t$ flow of value 11.

- Value of f: $val(f) = f\left(\delta^{out}(s)\right) - f\left(\delta^{in}(s)\right)$.
- f is subject to an arc-capacity $z \in \mathbb{R}_+^A$ if $f \leq z$.

Flow Decomposition

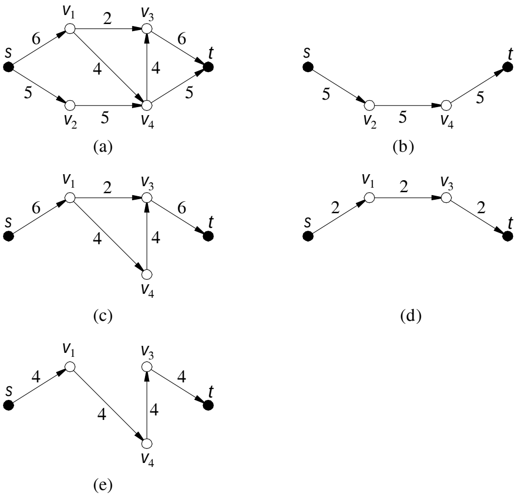

Figure: Any $s - t$ flow of value L can be decomposed into at most $|A|$ $s - t$ paths of total value L and possibly some circuits.

Maximum Flow

Maximum Flow: finding an $s - t$ flow f subject to a given arc-capacity $z \in \mathbb{R}_+^A$ such that $val(f)$ is maximized.

Maximum Flow

Maximum Flow: finding an $s - t$ flow f subject to a given arc-capacity $z \in \mathbb{R}_+^A$ such that $val(f)$ is maximized.

Solvable in polynomial time by flow-augmentation algorithms.

Multiflow

- Given k commodities with s_i, t_i being the source and sink, resp., for commodity i.
- $\mathcal{F}_i$: the set of s_i–t_i flows.
- A k-flow is a sequence $\langle f_1, f_2, \cdots, f_k \rangle$ with $f_i \in \mathcal{F}_i$ $\forall 1 \leq i \leq k$.

Multiflow

- Given k commodities with s_i, t_i being the source and sink, resp., for commodity i.
- $\mathcal{F}_i$: the set of s_i–t_i flows.
- A k-flow is a sequence $\langle f_1, f_2, \cdots, f_k \rangle$ with $f_i \in \mathcal{F}_i$ $\forall 1 \leq i \leq k$.
- A k-flow $\langle f_1, f_2, \cdots, f_k \rangle$ is subject to an arc-capacity $z \in \mathbb{R}_+^A$ if $\sum_{i=1}^k f_i \leq z$.

Maximum Concurrent Multiflow

$$\begin{array}{ll} \max & L \\ s.t. & f_i \in \mathcal{F}_i, \forall 1 \leq i \leq k \\ & val\left(f_i\right) = L, \forall 1 \leq i \leq k \\ & \sum_{i=1}^{k} f_i \leq z \end{array}$$

Concurrent Unicasts

- Given k unicasts are treated as k commodities.
- A k-flow $\langle f_1, f_2, \cdots, f_k \rangle$ is subject to an energy budget $b \in \mathbb{R}_+^V$ if

$$\sum_{e \in \delta^{out}(v)} c(e) \left(\sum_{i=1}^{k} f_i(e) \right) \leq b(v), \forall v \in V.$$

Concurrent Unicasts

- Given k unicasts are treated as k commodities.
- A k-flow $\langle f_1, f_2, \cdots, f_k \rangle$ is subject to an energy budget $b \in \mathbb{R}_+^V$ if

$$\sum_{e \in \delta^{out}(v)} c(e) \left(\sum_{i=1}^{k} f_i(e) \right) \leq b(v), \forall v \in V.$$

- MLRS corresponds to maximum concurrent multiflow subject to energy budget b

MLRS for Concurrent Unicasts

Step 1: Solve the LP

$$\begin{array}{ll} \max & L \\ s.t. & f_i \in \mathcal{F}_i, \forall 1 \leq i \leq k \\ & val\left(f_i\right) = L, \forall 1 \leq i \leq k \\ & \sum_{e \in \delta^{out}(v)} c\left(e\right) \left(\sum_{i=1}^{k} f_i\left(e\right)\right) \leq b\left(v\right), \forall v \in V \end{array}$$

MLRS for Concurrent Unicasts

Step 1: Solve the LP

$$
\begin{array}{ll}
\max & L \\
s.t. & f_i \in \mathcal{F}_i, \forall 1 \leq i \leq k \\
 & val\left(f_i\right) = L, \forall 1 \leq i \leq k \\
 & \sum_{e \in \delta^{out}(v)} c\left(e\right)\left(\sum_{i=1}^{k} f_i\left(e\right)\right) \leq b\left(v\right), \forall v \in V
\end{array}
$$

Step 2: Decompose each f_i into at most $|A|$ s_i–t_i paths of total value L and discarding the rest circuits if there is any.

Fractional Arborescence Packing

- $D = (V, A)$: a digraph with a "root" node s
- $\mathcal{T}$: collection of spanning arborescences rooted at s
- A *fractional* s-arborescence *packing* in D subject to given arc-capacity $z \in \mathbb{R}_+^A$ is a set of k pairs $(T_j, \lambda_j) \in \mathcal{T} \times \mathbb{R}_+$ satisfying that

$$\sum_{1 \leq j \leq k, e \in T_j} \lambda_j \leq z(e), \forall e \in A.$$

- The value of this packing is $\sum_{j=1}^{k} \lambda_j$.

Maximum Fractional Arborescence Packing

Maximum Fractional Arborescence Packing: finding a fractional s-arborescence packing in D subject to a given arc-capacity $z \in \mathbb{R}_+^A$ whose value is is maximized.

Maximum Fractional Arborescence Packing

Maximum Fractional Arborescence Packing: finding a fractional s-arborescence packing in D subject to a given arc-capacity $z \in \mathbb{R}_+^A$ whose value is is maximized.

Gabow-Manu algorithm

- a greedy algorithm
- using at most $|A|$ spanning s-arborescences.

A Min-Max Relation

$mflow(u, z)$: the value of a maximum $s - u$ flow in D subject to z, $\forall u \in V \setminus \{s\}$

Theorem

The value of a maximum fractional s-arborescence packing in D subject to z is equal to

$$\min_{u \in V \setminus \{s\}} mflow(u, z).$$

MLRS for Aggregation

Step 1: Compute an "optimal" arc-capacity $z \in \mathbb{R}_+^A$ by solving the LP:

$$
\begin{array}{ll}
\max & L \\
s.t. & \sum_{e \in \delta^{out}(v)} c(e) z(e) \leq b(v), \forall v \in V \\
& val(f_u) = L, \forall u \in V \setminus \{s\} \\
& f_u \in \mathcal{F}_u, \forall u \in V \setminus \{s\} \\
& f_u \leq z, \forall u \in V \setminus \{s\}
\end{array}
$$

MLRS for Aggregation

Step 1: Compute an "optimal" arc-capacity $z \in \mathbb{R}_{+}^{A}$ by solving the LP:

$$\begin{array}{rl} \max & L \\ s.t. & \sum_{e \in \delta^{out}(v)} c(e) z(e) \leq b(v), \forall v \in V \\ & val(f_u) = L, \forall u \in V \setminus \{s\} \\ & f_u \in \mathcal{F}_u, \forall u \in V \setminus \{s\} \\ & f_u \leq z, \forall u \in V \setminus \{s\} \end{array}$$

Step 2: Compute a maximum fractional packing of spanning inward s-arborescences subject to z using the Gabow-Manu algorithm.

Correctness

- *opt*: life of a max-life routing schedule
- L: the value of the LP in Step 1

Correctness

- *opt*: life of a max-life routing schedule
- *L*: the value of the LP in Step 1

Then,

1. $opt \leq L$

Correctness

- *opt*: life of a max-life routing schedule
- L: the value of the LP in Step 1

Then,

1. $opt \leq L$
2. the life of the output solution in Step 2 $\geq L$ by the min-max relation,