

The weighted ring arc-loading
problem with integer splitting: a
polynomial-time algorithm

Jinjiang Yuan

Zhengzhou University

Sanming Zhou

The University of Melbourne

ORSUM, November 12, 2004

Outline

Loading problems

Loading problems for rings

Weighted ring arc-loading problem with integer splitting

Part 1

Loading Problems

background

A **communication network** can be modelled by a graph, in which vertices represent processors, memory modules or routers and edges represent **bidirectional** links.

Given a network and a set of communication requests, design a **transmission route** (directed path) for each request such that high load on arcs/edges is avoided, where the **load on an arc** is the number of routes traversing the arc in its direction, and the **load on an edge** is the number of routes traversing the edge in either direction.

routing

Let $G = (V, E)$ be a connected graph. A communication **request** on G is an ordered pair (s, t) of distinct vertices, where s is the **source** and t is the **destination**. Let

$$D = \{(s_1, t_1), (s_2, t_2), \dots, (s_m, t_m)\}$$

be a set of requests. For each request (s_i, t_i) , we need to design a **directed path** P_i of G from s_i to t_i . A collection

$$\mathcal{P} = \{P_i : i = 1, 2, \dots, m\}$$

of such directed paths is called a **routing** for (G, D) .

loads on edges

The **load on an edge** with respect to \mathcal{P} is the number of directed paths in \mathcal{P} which pass through the edge in either direction. Thus, the load on an edge e is

$$\ell(e; \mathcal{P}) = |\{i : P_i \text{ uses } e \text{ in either direction, } 1 \leq i \leq m\}|.$$

Denote by $L(G, D; \mathcal{P})$ the **maximum load on edges** of G with respect to \mathcal{P} , that is,

$$L(G, D; \mathcal{P}) = \max_{e \in E} \ell(e; \mathcal{P}).$$

edge-loading problem

Given (G, D) , find a routing \mathcal{P} for (G, D) such that $L(G, D; \mathcal{P})$ is as small as possible.

Call

$$L(G, D) = \min_{\mathcal{P}} L(G, D; \mathcal{P})$$

the **edge-congestion** of (G, D) , where the minimum is over all routings \mathcal{P} for (G, D) .

loads on arcs

An **arc** of G is an edge with one of the two possible directions. The **load on an arc** with respect to \mathcal{P} is the number of directed paths in \mathcal{P} which pass through the arc in its direction. Thus, the load on an arc a is

$$\ell(a; \mathcal{P}) = |\{i : P_i \text{ uses } a \text{ in its direction, } 1 \leq i \leq m\}|.$$

Denote by $\vec{L}(G, D; \mathcal{P})$ the **maximum load on arcs** of G with respect to \mathcal{P} , that is,

$$\vec{L}(G, D; \mathcal{P}) = \max_{a \in A} \ell(a; \mathcal{P})$$

where A is the set of arcs of G .

arc-loading problem

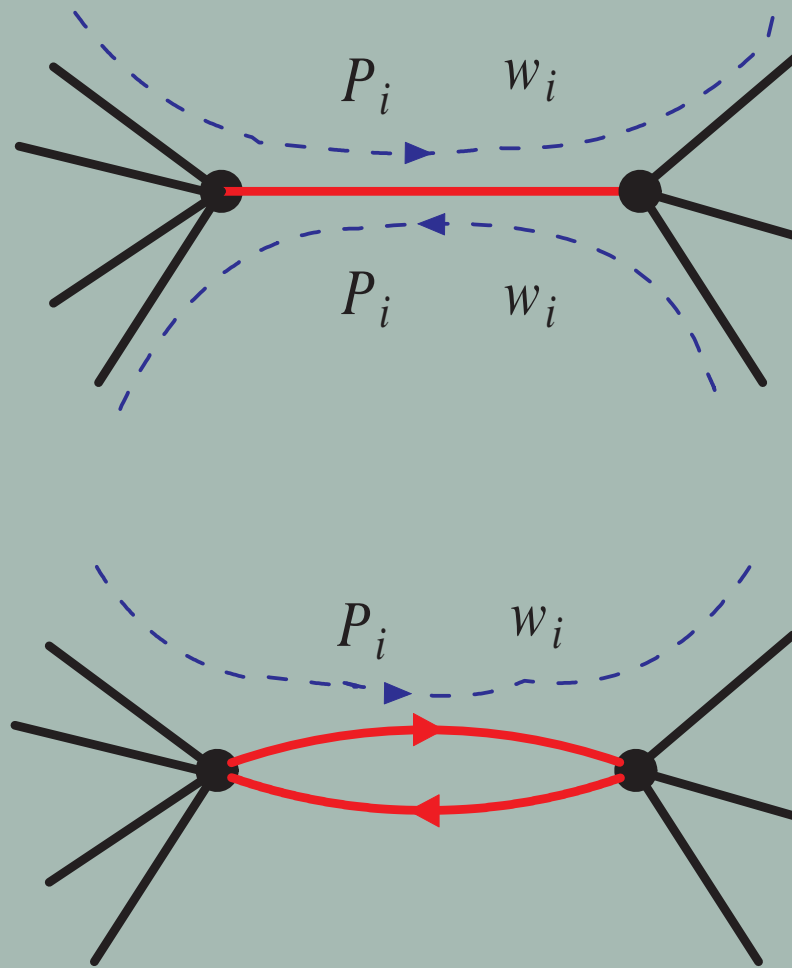
Given (G, D) , find a routing \mathcal{P} for (G, D) such that $\vec{L}(G, D; \mathcal{P})$ is as small as possible.

Call

$$\vec{L}(G, D) = \min_{\mathcal{P}} \vec{L}(G, D; \mathcal{P})$$

the **arc-congestion** of (G, D) , where the minimum is over all routings \mathcal{P} for (G, D) .

edge-loading problem \neq arc-loading problem



Above: load on an edge; Bottom: load on an arc

weighted requests

A request may be associated with a nonnegative integer **weight**, which usually represents the traffic demand. Let

$$D_w = \{(s_1, t_1; w_1), (s_2, t_2; w_2), \dots, (s_m, t_m; w_m)\}$$

be a set of weighted requests on G , where w_i is the integer weight for (s_i, t_i) . Let

$$\mathcal{P} = \{P_i : i = 1, 2, \dots, m\}$$

be a routing for (G, D_w) .

loads

The **load on an edge** e with respect to \mathcal{P} is the total weight of requests that are routed through e in either direction, that is,

$$\ell_w(e; \mathcal{P}) = \sum \{w_i : P_i \text{ uses } e \text{ in either direction, } 1 \leq i \leq m\}.$$

Similarly, the **load on an arc** a is

$$\ell_w(a; \mathcal{P}) = \sum \{w_i : P_i \text{ uses } a \text{ in its direction, } 1 \leq i \leq m\}.$$

$$L(G, D_w; \mathcal{P}) = \max_{e \in E} \ell_w(e; \mathcal{P}), \quad \vec{L}(G, D_w; \mathcal{P}) = \max_{a \in A} \ell_w(a; \mathcal{P})$$

$$L(G, D_w) = \min_{\mathcal{P}} L(G, D_w; \mathcal{P}), \quad \vec{L}(G, D_w) = \min_{\mathcal{P}} \vec{L}(G, D_w; \mathcal{P})$$

weighted edge-loading problem

Given (G, D_w) , find a routing \mathcal{P} for (G, D_w) such that $L(G, D_w; \mathcal{P})$ achieves the minimum, that is, $L(G, D_w; \mathcal{P}) = L(G, D_w)$.

weighted arc-loading problem

Given (G, D_w) , find a routing \mathcal{P} for (G, D_w) such that $\vec{L}(G, D_w; \mathcal{P})$ achieves the minimum, that is, $\vec{L}(G, D_w; \mathcal{P}) = \vec{L}(G, D_w)$.

weighted edge-loading problem \neq weighted arc-loading problem

Part 2

Loading Problems for Rings

why rings?

Rings are perhaps the most widely used structure for interconnection networks.

The North America standard for data transmission on optical fibre networks is **SONET** (**S**ynchronous **O**ptical **NET**work) ring.

non-weighted case

Theorem 1 (*Frank, 1985*) The *edge-loading problem for rings* can be solved in polynomial time.

Proof: Use a result of Okamura and Seymour on multicommodity flows.

Theorem 2 (*Wilfong and Winkler, 1998*) The *arc-loading problem for rings* can be solved in polynomial time.

Proof: Formulate an IP and apply an LP rounding technique.

weighted case

Theorem 3 (Cosares and Saniee, 1994) The *weighted edge-loading problem for rings* is NP-complete.

Proof: Reduction from the **Partition Problem**.

Cosares and Saniee also gave a 2-approximation algorithm.

Theorem 4 (Becchetti, Ianni and Marchetti-Spaccamela, 2002) The *weighted arc-loading problem for rings* is NP-complete.

Proof: Similar reduction.

approximation algorithms

For the **weighted edge-loading problem for rings**, Schrijver, Seymour and Winkler (1998) gave a polynomial time approximation algorithm, which produces a routing with load at most

$$(\text{minimum load}) + 1.5 \max_i w_i.$$

The error term could be a significant deviation from the optimum when the minimum load is small relative to $\max_i w_i$.

PTAS

An optimisation problem is said to have a **PTAS** (polynomial time approximation scheme) if, for any rational $\varepsilon > 0$, there is a polynomial time approximation algorithm with approximation ratio at most $1 + \varepsilon$. For minimisation problems, this means

$$\frac{\text{solution given by the algorithm}}{\text{optimal solution}} \leq 1 + \varepsilon.$$

PTAS exists for the **weighted edge-loading problem for rings** (Khanna, 1997)

PTAS exists for the **weighted arc-loading problem for rings** (Becchetti, Ianni and Marchetti-Spaccamela, 2002)

splitting: another approach

As an alternative approach, we may **split** the weight for each request into two **integral parts**, and route them clockwise and anticlockwise respectively.

Problem 1 *Is there a polynomial time algorithm for the loading problem if split is allowed?*

Theorem 5 (Myung, 2001) *The **weighted edge-loading problem for rings with integer splitting** can be solved in polynomial time.*

How about the **weighted arc-loading problem for rings**?

complexity: a summary

Non-weighted Edge-loading	Non-weighted Arc-loading
Polynomial (Frank, 85)	Polynomial (Wilfong & Winkler, 98)

	Weighted Edge-loading	Weighted Arc-loading
Non-splitting	NP-complete (Cosares & Saniee, 94)	NP-complete (Becchetti, etc., 02)
Splitting	Polynomial (Myung, 01)	???

complexity: a summary

Non-weighted Edge-loading	Non-weighted Arc-loading
Polynomial (Frank, 85)	Polynomial (Wilfong & Winkler, 98)

	Weighted Edge-loading	Weighted Arc-loading
Non-splitting	NP-complete (Cosares & Saniee, 94)	NP-complete (Becchetti, etc., 02)
Splitting	Polynomial (Myung, 01)	Polynomial (Yuan & Zhou, 03)

Part 3

Weighted Ring Arc-loading Problem with Integer Splitting

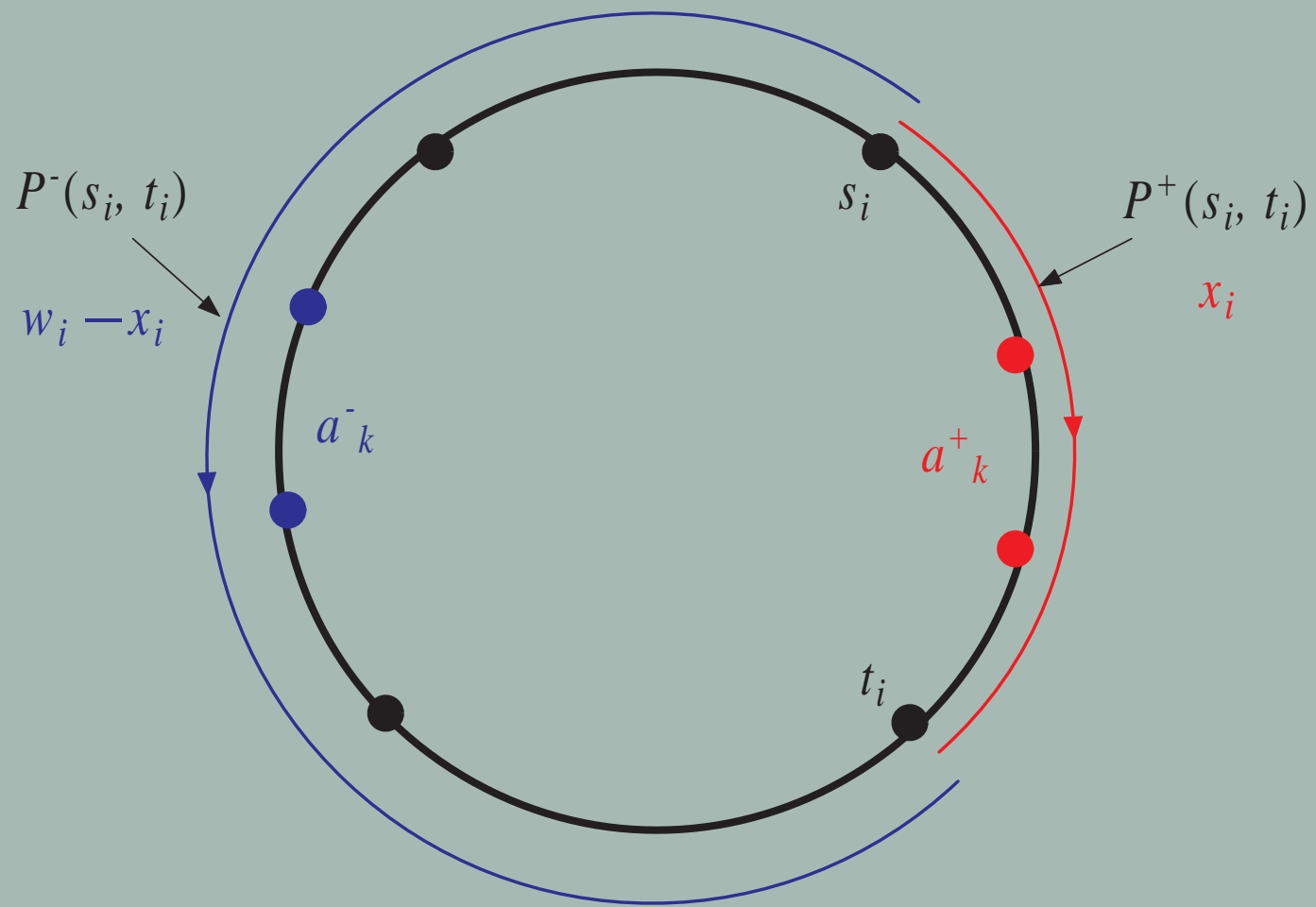
notation

Let C_n be a ring with n vertices v_0, v_1, \dots, v_{n-1} labelled clockwise. Subscripts are taken modulo n in the sequel.

For $k = 0, 1, \dots, n - 1$, denote

$$a_k^+ = (v_k, v_{k+1}), \quad a_k^- = (v_{k+1}, v_k).$$

For distinct vertices s, t , let $P^+(s, t)$ denote the directed (s, t) -path clockwise around C_n , and $P^-(s, t)$ the directed (s, t) -path anticlockwise around C_n .



a_k^+ , a_k^- , $P^+(s_i, t_i)$ and $P^-(s_i, t_i)$

Let

$$D_w = \{(s_1, t_1; w_1), (s_2, t_2; w_2), \dots, (s_m, t_m; w_m)\}$$

be any set of weighted requests on C_n , where w_i 's are **positive integers**.

A routing for D with integer splitting determines uniquely a vector

$$\mathbf{x} = (x_1, x_2, \dots, x_m)$$

and vice versa: route x_i along $P^+(s_i, t_i)$ and $w_i - x_i$ along $P^-(s_i, t_i)$, where each x_i is an integer with $0 \leq x_i \leq w_i$. Call \mathbf{x} an **integral routing**. In general, if we do not require that all x_i 's are integers, then call \mathbf{x} a **routing**.

For each $k = 0, 1, \dots, n - 1$, the loads on a_k^+ , a_k^- under \mathbf{x} are

$$L(\mathbf{x}, a_k^+) = \sum \{x_i : P^+(s_i, t_i) \text{ passes through } a_k^+\}$$

$$L(\mathbf{x}, a_k^-) = \sum \{w_i - x_i : P^-(s_i, t_i) \text{ passes through } a_k^-\}$$

respectively.

The maximum load on arcs of C_n is

$$L(\mathbf{x}) = \max \left\{ \max_{0 \leq k \leq n-1} L(\mathbf{x}, a_k^+), \max_{0 \leq k \leq n-1} L(\mathbf{x}, a_k^-) \right\}.$$

WRALP

Weighted Ring Arc-loading Problem with Integer Splitting

Instance A ring C_n and a set

$$D = \{(s_1, t_1; w_1), (s_2, t_2; w_2), \dots, (s_m, t_m; w_m)\}$$

of integrally weighted requests on C_n .

Objective Find an integral routing \mathbf{x} for D such that $L(\mathbf{x})$ is minimised.

polynomial solvability

Theorem 6 (Yuan and Zhou, 2003) The *Weighted Ring Arc-Loading Problem with Integer Splitting* can be solved in polynomial time.

Proof: Combinatorial methods plus the LP rounding technique, which was first used by Wilfong and Winkler in dealing with the non-weighted arc-loading problem for rings.

Taking each $(s_i, t_i; w_i)$ as w_i non-weighted requests and applying Wilfong and Winkler's algorithm, we get a **pseudo-polynomial algorithm**. (The LP problem involved has $\sum w_i$ variables, but $\sum(\lceil \log w_i \rceil + 1)$ will contribute to the input size.)

integer programming formulation

$$\text{WRALP: } \begin{cases} \min L(\mathbf{x}) \\ 0 \leq x_i \leq w_i, \quad 1 \leq i \leq m \\ x_i \text{ an integer, } \quad 1 \leq i \leq m. \end{cases}$$

LP relaxation

$$\text{LPR: } \begin{cases} \min L(\mathbf{x}) \\ 0 \leq x_i \leq w_i, \quad 1 \leq i \leq m. \end{cases}$$

feasible solutions of WRALP \leftrightarrow integral routings
feasible solutions of LPR \leftrightarrow routings

a relevant LP problem

Denote $W = \sum_{1 \leq i \leq m} w_i$. For any real number α with $0 \leq \alpha \leq W$, formulate the following LP:

$$\text{LPR}_\alpha: \begin{cases} \min L(\mathbf{x}) \\ \sum_{1 \leq i \leq m} x_i = \alpha \\ 0 \leq x_i \leq w_i, \quad 1 \leq i \leq m. \end{cases}$$

Let L_{OPT} , L_* and L_α denote the optimal objective values of WRALP, LPR and LPR_α , respectively.

Lemma 1 *As a function of α , L_α is convex on the interval $[0, W]$.*

A feasible solution $\mathbf{x} = (x_1, x_2, \dots, x_m)$ of LPR is called a **flush routing** if $\sum_{1 \leq i \leq m} x_i$ is an integer.

Lemma 2 *Let $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_m^*)$ be an optimal solution of LPR, and let $\alpha^* = \sum_{1 \leq i \leq m} x_i^*$. Let \mathbf{x} be an optimal solution of LPR $_{\lfloor \alpha^* \rfloor}$ if $L_{\lfloor \alpha^* \rfloor} \leq L_{\lceil \alpha^* \rceil}$, and an optimal solution of LPR $_{\lceil \alpha^* \rceil}$ otherwise. Then \mathbf{x} is a flush routing and $L(\mathbf{x}) \leq L_{\text{OPT}}$.*

Proof Clearly, \mathbf{x}^* is an optimal solution of LPR $_{\alpha^*}$. So

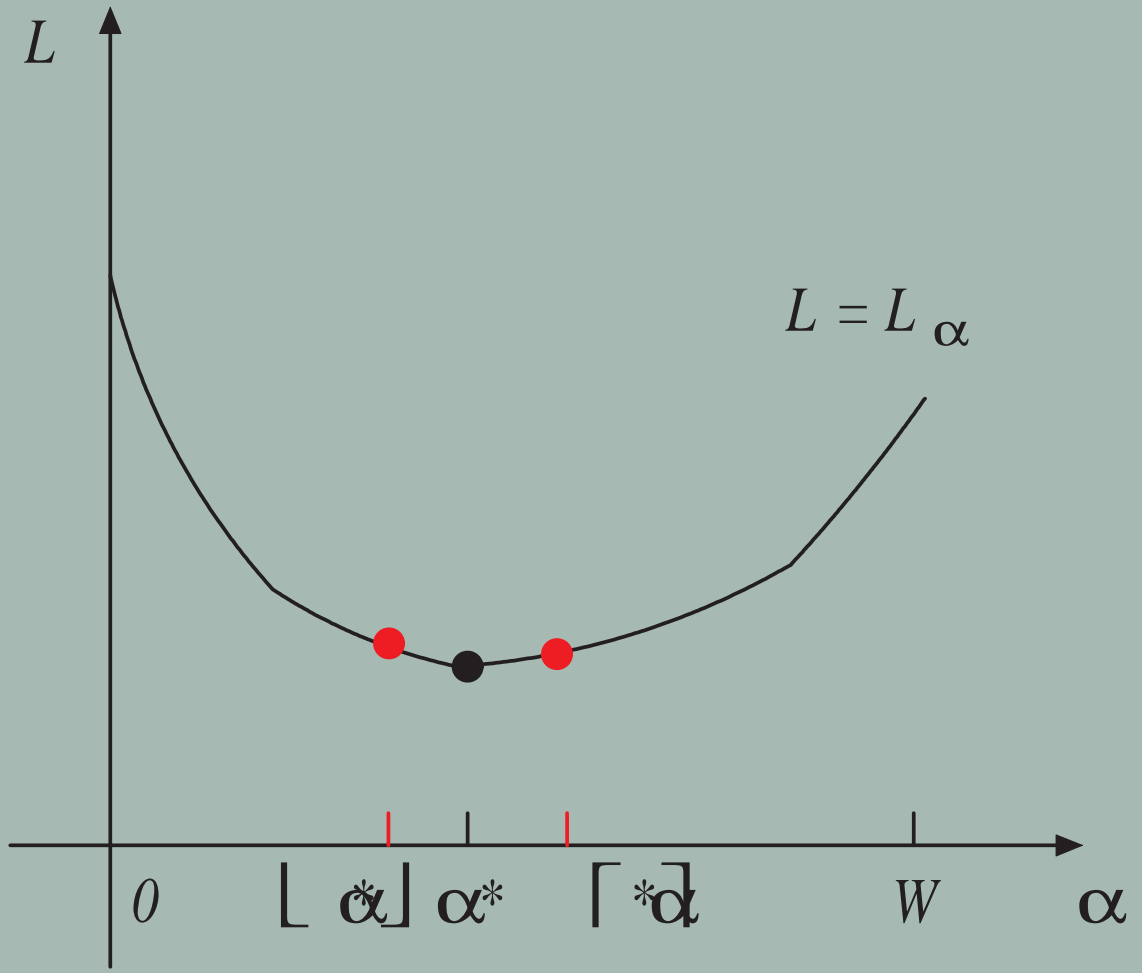
$$L_{\alpha^*} = L(\mathbf{x}^*) = L_* = \min\{L_\alpha : 0 \leq \alpha \leq W\}.$$

Since L_α is convex by Lemma 1, this implies

$$\min\{L_{\lfloor \alpha^* \rfloor}, L_{\lceil \alpha^* \rceil}\} = \min\{L_\alpha : \alpha \text{ is an integer and } 0 \leq \alpha \leq W\}.$$

But $\min\{L_\alpha : \alpha \text{ is an integer and } 0 \leq \alpha \leq W\} \leq L_{\text{OPT}}$, so

$$\min\{L_{\lfloor \alpha^* \rfloor}, L_{\lceil \alpha^* \rceil}\} \leq L_{\text{OPT}}.$$

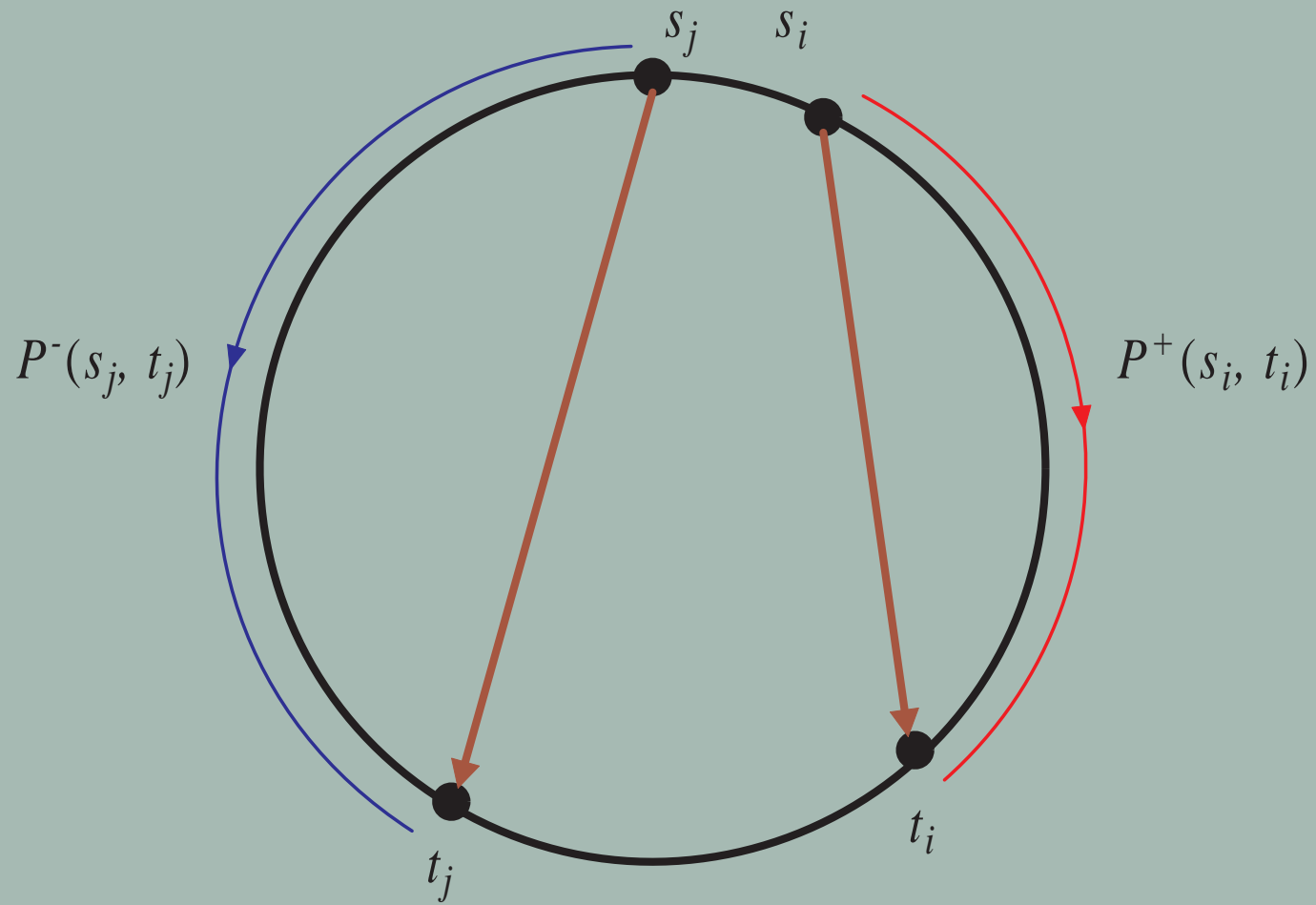


Convexity of L_α

Since each of LPR , $LPR_{\lfloor \alpha^* \rfloor}$ and $LPR_{\lceil \alpha^* \rceil}$ can be solved in polynomial time, by Lemma 2 a flush routing \mathbf{x} with $L(\mathbf{x}) \leq L_{OPT}$ can be found in polynomial time.

Two distinct requests (s_i, t_i) and (s_j, t_j) are said to be **parallel** if $P^+(s_i, t_i)$ and $P^-(s_j, t_j)$, or $P^+(s_j, t_j)$ and $P^-(s_i, t_i)$, intersect in at most one vertex. (Wilfong and Winkler)

A request $(s_i, t_i; w_i)$ is **split** by a routing $\mathbf{x} = (x_1, x_2, \dots, x_m)$ if $0 < x_i < w_i$.



Parallel requests

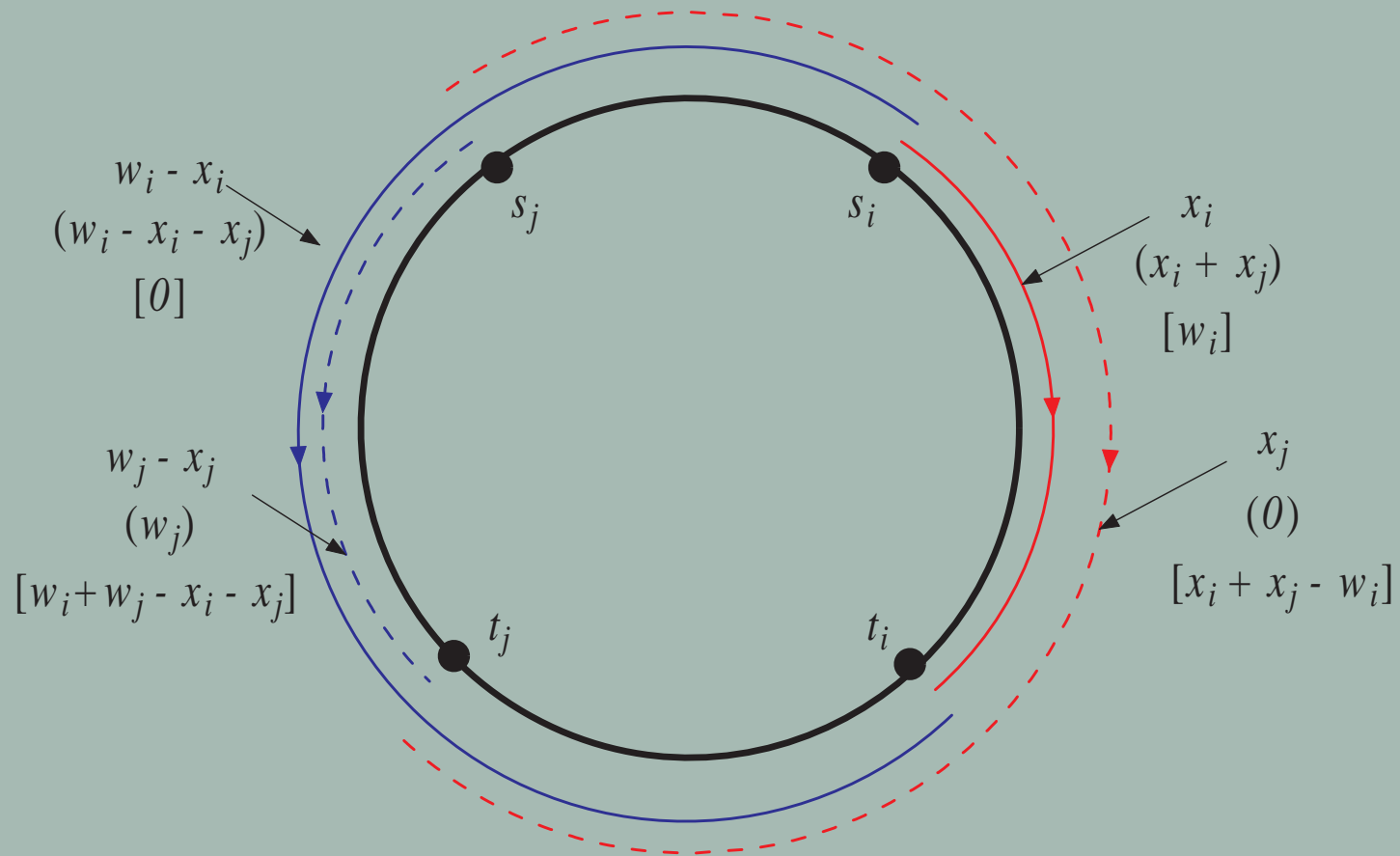
Lemma 3 Given a flush routing $\mathbf{x} = (x_1, x_2, \dots, x_m)$, we can find in polynomial time a flush routing $\mathbf{y} = (y_1, y_2, \dots, y_m)$ such that $L(\mathbf{y}) \leq L(\mathbf{x})$ and no two parallel requests are both split by \mathbf{y} .

Proof Suppose (s_i, t_i) and (s_j, t_j) are parallel requests which are both split by \mathbf{x} . Without loss of generality we may suppose that the directed paths $P^+(s_i, t_i)$ and $P^-(s_j, t_j)$ intersect in at most one vertex. If $x_i + x_j \leq w_i$, then define

$$y_i = x_i + x_j, \quad y_j = 0, \quad y_k = x_k \text{ for } k \neq i, j;$$

and if $x_i + x_j > w_i$, then define

$$y_i = w_i, \quad y_j = x_i + x_j - w_i, \quad y_k = x_k \text{ for } k \neq i, j.$$



The coordinates of y are in parentheses (when $x_i + x_j \leq w_i$) and brackets (when $x_i + x_j > w_i$).

In both cases, $\mathbf{y} = (y_1, y_2, \dots, y_m)$ is a flush routing with

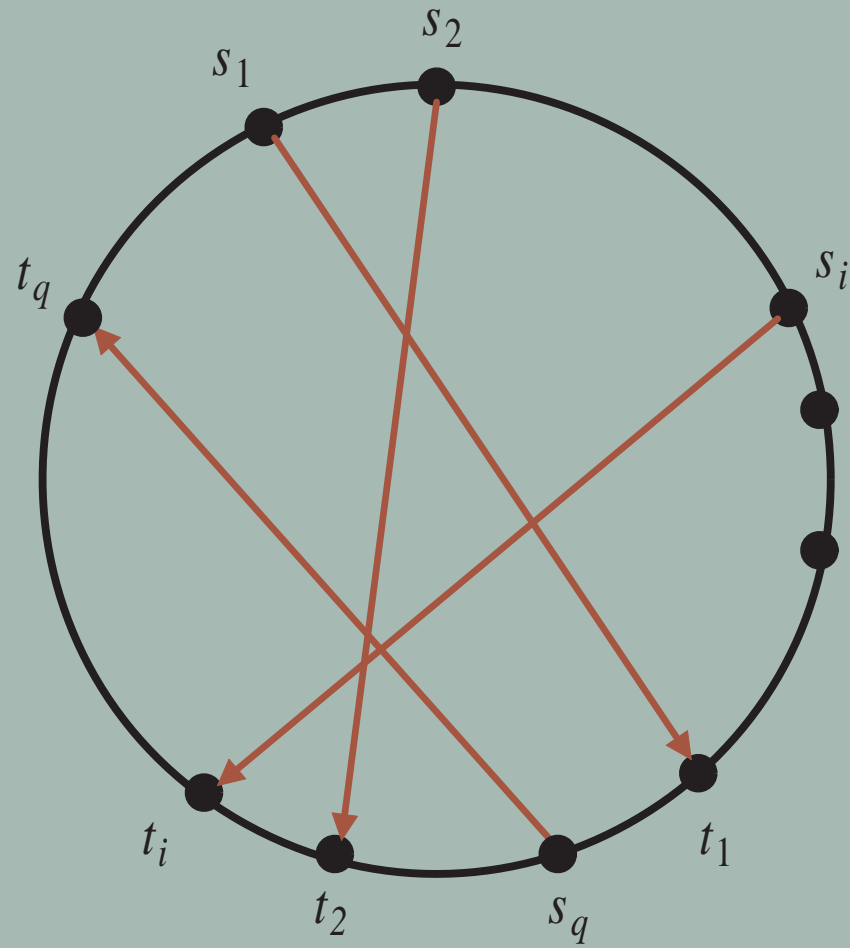
$$\sum_{1 \leq i \leq m} y_i = \sum_{1 \leq i \leq m} x_i$$

and one of (s_j, t_j) and (s_i, t_i) is not split by \mathbf{y} . Also, under \mathbf{y} the arcs in $P^+(s_i, t_i)$ and $P^-(s_j, t_j)$ have the same loads as before and other arcs have the same or reduced loads. Thus, $L(\mathbf{y}) \leq L(\mathbf{x})$ and \mathbf{y} splits less number of requests than \mathbf{x} . If there exist no parallel requests both split by \mathbf{y} , we are done; otherwise repeat the process above. After at most m such processes, we then get the desired flush routing, and the proof of Lemma 3 is complete.

Proof of Theorem 6 (sketch) By Lemma 3 and the comments after the proof of Lemma 2, we can find in polynomial time a flush routing $\mathbf{y} = (y_1, y_2, \dots, y_m)$ such that $L(\mathbf{y}) \leq L_{\text{OPT}}$ and no two parallel requests are both split by \mathbf{y} . In particular, for each $k = 0, 1, \dots, n - 1$, at most one request with source v_k is split. Hence the number of requests split by \mathbf{y} , denoted by q , is at most n . Without loss of generality, we may suppose that the requests split by \mathbf{y} are

$$(s_1, t_1; w_1), (s_2, t_2; w_2), \dots, (s_q, t_q; w_q)$$

and that the sources s_1, s_2, \dots, s_q of them are ordered clockwise on the ring C_n . By the properties of \mathbf{y} above, any two of such requests are not parallel. Thus, the destinations t_1, t_2, \dots, t_q of them must be in clockwise order as well.

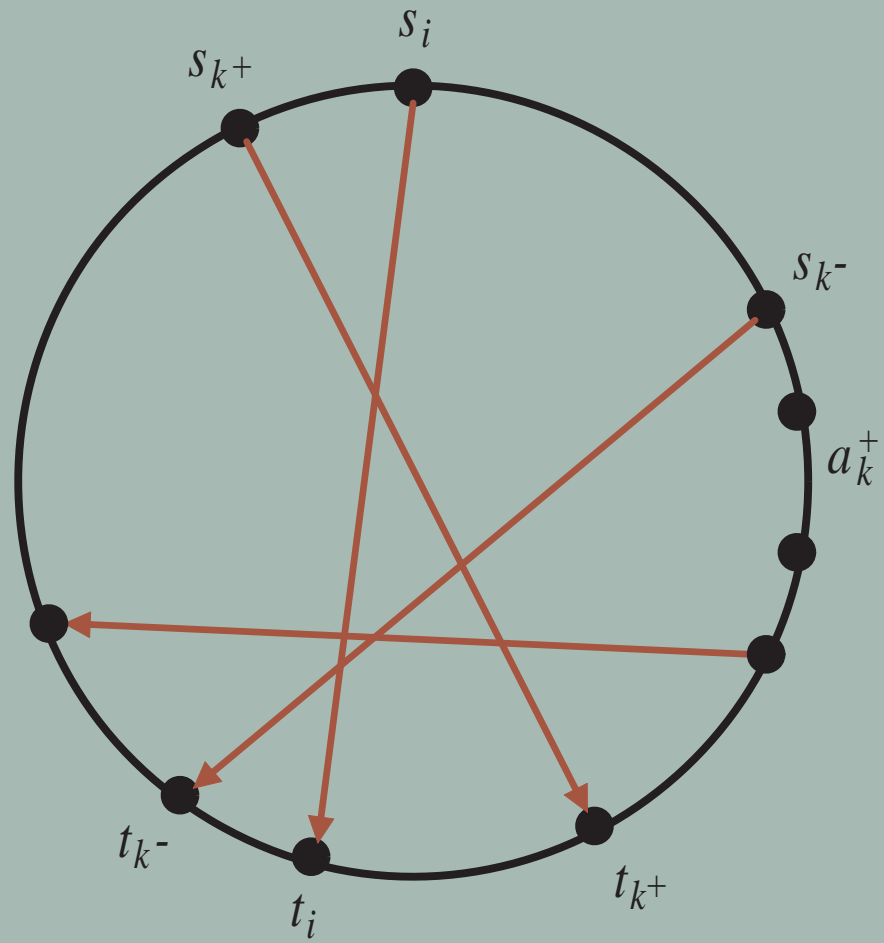


Consistent orders of s_1, s_2, \dots, s_q and t_1, t_2, \dots, t_q

For i, j with $1 \leq i, j \leq q$, denote

$$[i, j]_q = \begin{cases} \{i, i + 1, \dots, j\}, & \text{if } i \leq j \\ \{1, 2, \dots, q\} \setminus \{j + 1, j + 2, \dots, i - 1\}, & \text{if } i > j. \end{cases}$$

For $0 \leq k \leq n - 1$, if there exists i with $1 \leq i \leq q$ such that $a_k^+ \in P^+(s_i, t_i)$, then define k^+ (k^-) to be such a subscript i such that the length of $P^+(s_i, v_k)$ is maximum (minimum). (Note that k^+ can be greater than, less than, or equal to k^- . For example, if t_q is between s_1 and t_1 and $v_k \neq t_q$ is between s_1 and t_q , then we have $k^+ > k^-$.) If there exists no i with $1 \leq i \leq q$ such that $a_k^+ \in P^+(s_i, t_i)$, then define $[k^+, k^-]_q = \emptyset$. This latter case occurs if and only if, for some i , s_{i+1} is after t_i and $v_k \neq s_{i+1}$ is between t_i and s_{i+1} (subscripts of s and t modulo q) with respect to the cyclic order v_0, v_1, \dots, v_{n-1} of the vertices of C_n .



Definitions of k^+ and k^-

Since the sources and the destinations of the requests split by y are ordered in the same direction, for each $i = 1, 2, \dots, q$, we have

$$\begin{aligned} i \in [k^+, k^-]_q &\iff a_k^+ \in P^+(s_i, t_i) \\ i \notin [k^+, k^-]_q &\iff a_k^- \in P^-(s_i, t_i). \end{aligned} \tag{1}$$

Let $z_1 = \lfloor y_1 \rfloor$ if $y_1 - \lfloor y_1 \rfloor \leq 1/2$ and $z_1 = \lfloor y_1 \rfloor + 1$ otherwise.

Define recursively

$$z_j = \begin{cases} \lfloor y_j \rfloor, & \text{if } 1 \leq j \leq q \text{ and } y_j - \lfloor y_j \rfloor - \sum_{1 \leq i \leq j-1} (z_i - y_i) \leq \frac{1}{2} \\ \lfloor y_j \rfloor + 1, & \text{if } 1 \leq j \leq q \text{ and } y_j - \lfloor y_j \rfloor - \sum_{1 \leq i \leq j-1} (z_i - y_i) > \frac{1}{2} \\ y_j, & \text{if } q + 1 \leq j \leq m. \end{cases} \tag{2}$$

Then $\mathbf{z} = (z_1, z_2, \dots, z_m)$ is an integral routing for D . By induction one can prove that

$$-\frac{1}{2} \leq \sum_{1 \leq i \leq j} (z_i - y_i) < \frac{1}{2} \quad (3)$$

for each $j = 1, 2, \dots, q$. In particular,

$$-\frac{1}{2} \leq \sum_{1 \leq i \leq q} (z_i - y_i) < \frac{1}{2}.$$

However, $\sum_{1 \leq i \leq q} (z_i - y_i)$ is an integer because both \mathbf{y} and \mathbf{z} are flush routings. So we must have

$$\sum_{1 \leq i \leq q} (z_i - y_i) = 0. \quad (4)$$

For each $k = 0, 1, \dots, n - 1$, from (1) we have

$$L(\mathbf{z}, a_k^+) = L(\mathbf{y}, a_k^+) + \sum_{i: i \in [k^+, k^-]_q} (z_i - y_i)$$

$$L(\mathbf{z}, a_k^-) = L(\mathbf{y}, a_k^-) + \sum_{i: i \notin [k^+, k^-]_q} (y_i - z_i).$$

From this and by using (4) we obtain

$$L(\mathbf{z}, a_k^+) - L(\mathbf{y}, a_k^+) = L(\mathbf{z}, a_k^-) - L(\mathbf{y}, a_k^-) = \sum_{i: i \in [k^+, k^-]_q} (z_i - y_i).$$

Again by (4) we have

$$\sum_{i: i \in [k^+, k^-]_q} (z_i - y_i) = \sum_{1 \leq i \leq k^-} (z_i - y_i) - \sum_{1 \leq i \leq k^+ - 1} (z_i - y_i) \quad (5)$$

no matter whether $k^+ \leq k^-$ or $k^+ > k^-$. However, from (3) the right hand side of (5) is strictly less than 1. Thus, we have

$$L(\mathbf{z}, a_k^+) - L(\mathbf{y}, a_k^+) = L(\mathbf{z}, a_k^-) - L(\mathbf{y}, a_k^-) < 1.$$

Since this is true for each $k = 0, 1, \dots, n - 1$, it follows that

$$L(\mathbf{z}) < L(\mathbf{y}) + 1 \leq L_{\text{OPT}} + 1.$$

But $L_{\text{OPT}} \leq L(\mathbf{z})$ and \mathbf{z} is an integral routing, so $L(\mathbf{z}) = L_{\text{OPT}}$ and \mathbf{z} is an optimal routing for the Weighted Ring Arc-Loading Problem with Integer Splitting. Since \mathbf{y} can be found in polynomial time, from the proof above \mathbf{z} can be found in polynomial time as well.

polynomial algorithm for WRALP

1. Solve LPR and obtain an optimal solution \mathbf{x}^* .
2. Using Lemma 2 and solving $\text{LPR}_{\lfloor \alpha^* \rfloor}$ and $\text{LPR}_{\lceil \alpha^* \rceil}$, obtain a flush routing \mathbf{x} with $L(\mathbf{x}) \leq L_{\text{OPT}}$.
3. Applying the procedure in the proof of Lemma 3, turn \mathbf{x} into a flush routing \mathbf{y} with $L(\mathbf{y}) \leq L_{\text{OPT}}$ such that no parallel requests are both split by \mathbf{y} .
4. Rounding \mathbf{y} to obtain \mathbf{z} as in (2). Then \mathbf{z} is an optimal integral routing for WRALP.

The majority of the computational time is for solving the three LP problems in Steps 1-2.

Thank You !