Packet Routing Problems on Plane Grids

Ignasi Sau Valls

Mascotte project, CNRS/I3S-INRIA-UNSA, France

Joint work with Omid Amini, Florian Huc and Janez Žerovnik

AEOLUS Workshop on Scheduling

Outline

- Introduction
  - Statement of the problem
  - Preliminaries
  - Example
- Permutation routing algorithm for triangular grids
  - Description
  - Correctness
  - Optimality
- Permutation routing algorithm for hexagonal grids
- (ℓ, k)-routing algorithms
- Conclusions

(ℓ, k)-routing

- The (ℓ, k)**-routing** problem is a **packet routing** problem.
- Each processor is the **origin of at most** ℓ **packets** and the **destination of no more than** k **packets**.
- The goal is to **minimize the number of time steps** required to route all packets to their respective destinations.
- **Permutation routing** is the particular case when $\ell = k = 1$

(ℓ, k)-routing

- The (ℓ, k)**-routing** problem is a **packet routing** problem.
- Each processor is the **origin of at most** ℓ **packets** and the **destination of no more than** k **packets**.
- The goal is to **minimize the number of time steps** required to route all packets to their respective destinations.
- **Permutation routing** is the particular case when $\ell = k = 1$

Permutation Routing

Statement of the problem

- **Input:**
  - a directed graph $G = (V, E)$ (the *host* graph),
  - a subset $S \subseteq V$ of nodes,
  - and a permutation $\pi : S \rightarrow S$.
    Each node $u \in S$ wants to send a packet to $\pi(u)$.
- **Output:** Find for each pair $(u, \pi(u))$, a path form u to $\pi(u)$ in G.
- **Constraints:**
  - At each step, a packet can either move or stay at a node.
  - No arc can be crossed by two packets at the same step.
  - Cohabitation of multiple packets at the same node is allowed.
- **Goal: minimize the number of time steps** required to route all packets to their respective destinations.

Statement of the problem

- **Input:**
  - a directed graph $G = (V, E)$ (the *host* graph),
  - a subset $S \subseteq V$ of nodes,
  - and a permutation $\pi : S \rightarrow S$.
    Each node $u \in S$ wants to send a packet to $\pi(u)$.
- **Output:** Find for each pair $(u, \pi(u))$, a path form u to $\pi(u)$ in G.
- **Constraints:**
  - At each step, a packet can either move or stay at a node.
  - No arc can be crossed by two packets at the same step.
  - Cohabitation of multiple packets at the same node is allowed.
- **Goal:** **minimize the number of time steps** required to route all packets to their respective destinations.

Statement of the problem

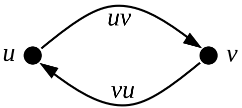
- **Input:**
  - a directed graph $G = (V, E)$ (the *host* graph),
  - a subset $S \subseteq V$ of nodes,
  - and a permutation $\pi : S \to S$.
    Each node $u \in S$ wants to send a packet to $\pi(u)$.
- **Output:** Find for each pair $(u, \pi(u))$, a path form u to $\pi(u)$ in G.
- **Constraints:**
  - At each step, a packet can either move or stay at a node.
  - No arc can be crossed by two packets at the same step.
  - Cohabitation of multiple packets at the same node is allowed.
- **Goal: minimize the number of time steps** required to route all packets to their respective destinations.

Statement of the problem

- **Input:**
  - a directed graph $G = (V, E)$ (the *host* graph),
  - a subset $S \subseteq V$ of nodes,
  - and a permutation $\pi : S \to S$.
    Each node $u \in S$ wants to send a packet to $\pi(u)$.
- **Output:** Find for each pair $(u, \pi(u))$, a path form u to $\pi(u)$ in G.
- **Constraints:**
  - At each step, a packet can either move or stay at a node.
  - No arc can be crossed by two packets at the same step.
  - Cohabitation of multiple packets at the same node is allowed.
- **Goal: minimize the number of time steps** required to route all packets to their respective destinations.

Assumptions

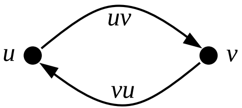
- We consider the **store-and-forward** and **Δ-port** model.
- **Full duplex link**: packets can be sent in the two directions of the link **simultaneously**.



- If the network is **half-duplex** → **2 factor approximation algorithm** from an optimal algorithm for the full-duplex case, by introducing *odd-even* steps.

Assumptions

- We consider the **store-and-forward** and **Δ-port** model.
- **Full duplex link**: packets can be sent in the two directions of the link **simultaneously**.



- If the network is **half-duplex** $\rightarrow$

  2 factor approximation algorithm from an optimal algorithm for the full-duplex case, by introducing *odd-even* steps.

Previous work

-The permutation routing problem has been studied in:

- Mobile Ad Hoc Networks
- Cube-Connected Cycle Networks
- Wireless and Radio Networks
- All-Optical Networks
- Reconfigurable Meshes...

-But, optimal algorithms (in the worst case):

- 2-circulant graphs, square grids.
- Triangular grids: **Two-terminal routing**
  (only one message to be sent)

-In this talk we describe **optimal permutation routing algorithms for triangular and hexagonal grids**.

Previous work

-The permutation routing problem has been studied in:

- Mobile Ad Hoc Networks
- Cube-Connected Cycle Networks
- Wireless and Radio Networks
- All-Optical Networks
- Reconfigurable Meshes...

-But, optimal algorithms (in the worst case):

- 2-circulant graphs, square grids.
- Triangular grids: **Two-terminal routing** (only one message to be sent)

-In this talk we describe **optimal permutation routing algorithms for triangular and hexagonal grids**.

Previous work

-The permutation routing problem has been studied in:

- Mobile Ad Hoc Networks
- Cube-Connected Cycle Networks
- Wireless and Radio Networks
- All-Optical Networks
- Reconfigurable Meshes...

-But, optimal algorithms (in the worst case):

- 2-circulant graphs, square grids.
- Triangular grids: **Two-terminal routing**
  (only one message to be sent)

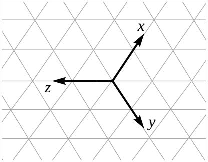
-In this talk we describe **optimal permutation routing algorithms for triangular and hexagonal grids**.

Permutation Routing on Triangular Grids

Notation and preliminary results

Nocetti, Stojmenović and Zhang [IEEE TPDS'02]:

Representation of the relative address of the nodes on a generating system ***i***, ***j***, ***k*** on the directions of the three axis x, y, z.



- This address **is not unique**, but we have that, being (a, b, c) and (a', b', c') the addresses of two $D - S$ pairs,

$$(a, b, c) = (a', b', c') \quad \Leftrightarrow \quad \exists \text{ an integer } d \text{ such that}$$

$$a' = a + d,$$

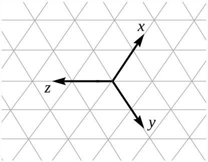
$$b' = b + d,$$

$$c' = c + d.$$

Notation and preliminary results

Nocetti, Stojmenović and Zhang [IEEE TPDS'02]:

Representation of the relative address of the nodes on a generating system ***i***, ***j***, ***k*** on the directions of the three axis x, y, z.



- This address **is not unique**, but we have that, being (a, b, c) and (a', b', c') the addresses of two $D - S$ pairs,

$$(a, b, c) = (a', b', c') \quad \Leftrightarrow \quad \exists \text{ an integer } d \text{ such that}$$

$$a' = a + d,$$
$$b' = b + d,$$
$$c' = c + d.$$

Notation and preliminary results (2)

- A relative address $D - S = (a, b, c)$ is of the ***shortest path form*** if
  - there is a path C from S to D, $C = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$,
  - and C has the shortest length over all paths going from S to D.

Theorem (*NSZ'02*)

*An address (a, b, c) is of the **shortest path form** if and only if*

i) ***at least one component is zero*** *(that is, $abc = 0$),*

ii) *and **any two components do not have the same sign** (that is, $ab \leq 0,\ ac \leq 0,$ and $bc \leq 0$).*

Notation and preliminary results (2)

- A relative address $D - S = (a, b, c)$ is of the ***shortest path form*** if
  - there is a path C from S to D, C=$a\mathbf{i}$+$b\mathbf{j}$+$c\mathbf{k}$,
  - and C has the shortest length over all paths going from S to D.

Theorem (*NSZ'02*)

*An address (a, b, c) is of the **shortest path form** if and only if*

i) ***at least one component is zero*** *(that is, $abc = 0$),*

ii) *and **any two components do not have the same sign** (that is, $ab \leq 0,\ ac \leq 0,$ and $bc \leq 0$).*

Notation and preliminary results (3)

Corollary (*NSZ'02*)

*Any address has a **unique** shortest path form.*

Corollary (*NSZ'02*)

If $D - S = (a, b, c)$, then the shortest path form is one of those:

$$(0, b-a, c-a),$$
$$(a-b, 0, c-b),$$
$$(a-c, b-c, 0),$$

and thus:

$$|D-S| = \min(|b-a| + |c-a|, |a-b| + |c-b|, |a-c| + |b-c|).$$

Notation and preliminary results (3)

Corollary (*NSZ'02*)

Any address has a ***unique*** *shortest path form.*

Corollary (*NSZ'02*)

If $D - S = (a, b, c)$*, then the shortest path form is one of those:*

$$(0, b-a, c-a),$$
$$(a-b, 0, c-b),$$
$$(a-c, b-c, 0),$$

and thus:

$$|D - S| = \min(|b-a| + |c-a|, |a-b| + |c-b|, |a-c| + |b-c|).$$

Notation and preliminary results (4)

- Given a packet p and its relative address (a, b, c) *in the shortest path form*,

$$\ell_p := |a| + |b| + |c|,$$

$$\ell_{max} := \max_p(\ell_p)$$

- Trivial **lower bound**:

  Any permutation routing algorithm needs at least ℓ_{max} routing steps.

Notation and preliminary results (4)

- Given a packet p and its relative address (a, b, c) *in the shortest path form*,

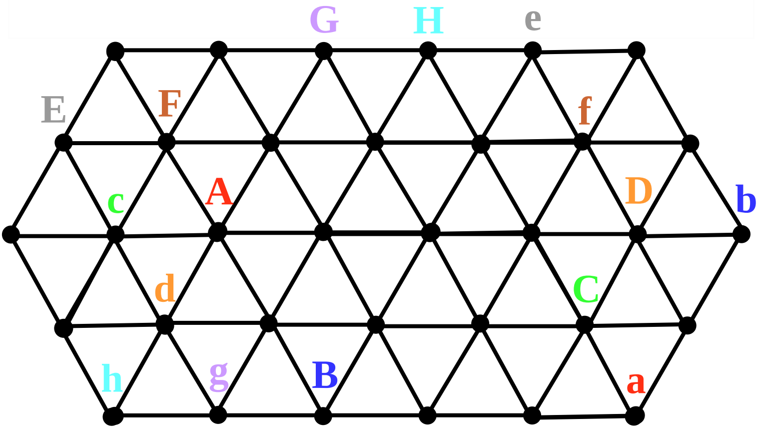
$$\ell_p := |a| + |b| + |c|,$$

$$\ell_{max} := \max_p(\ell_p)$$

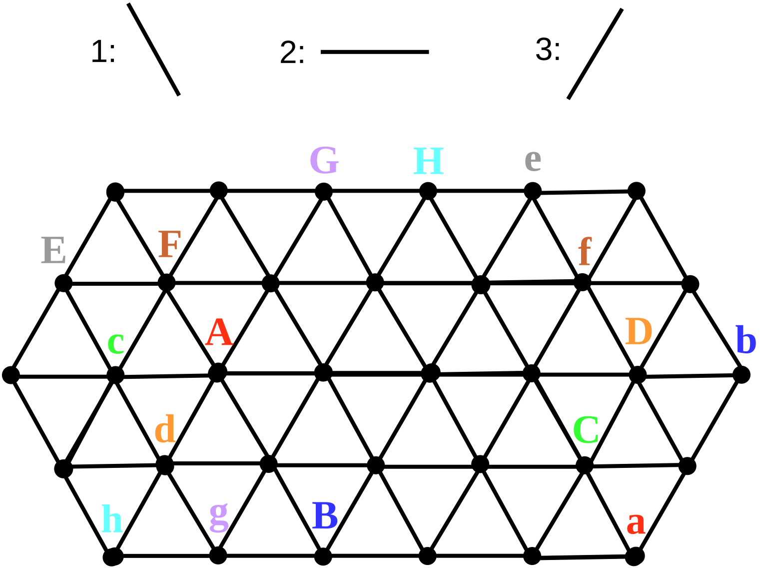
- Trivial **lower bound**:

  Any permutation routing algorithm needs at least ℓ_{max} routing steps.

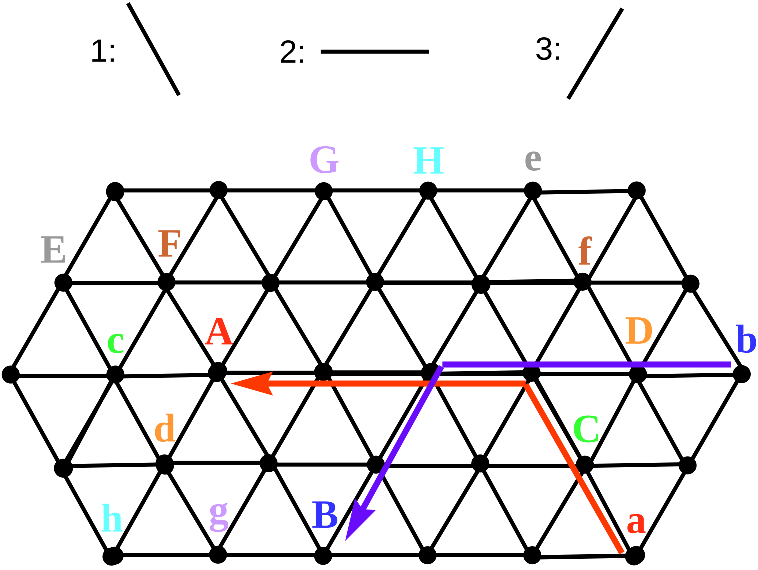
Example of an instance



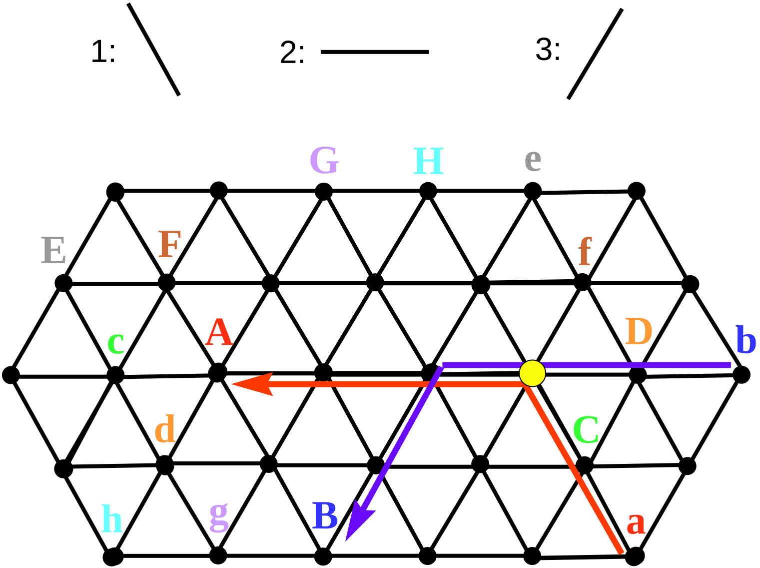
A non-optimal intuitive algorithm



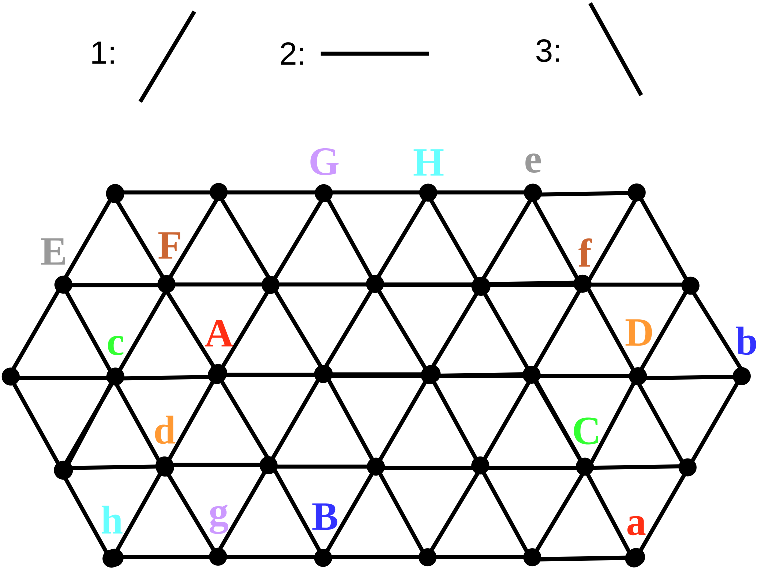
A non-optimal intuitive algorithm (2)



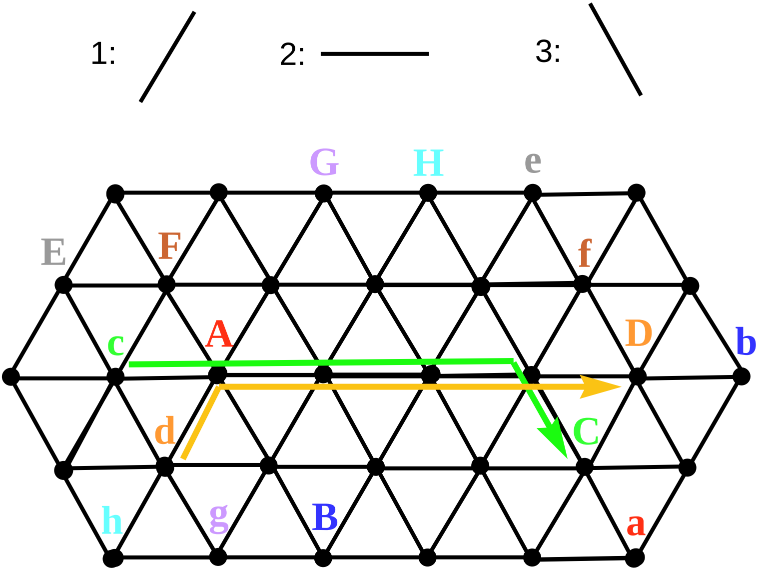
A non-optimal intuitive algorithm (3)



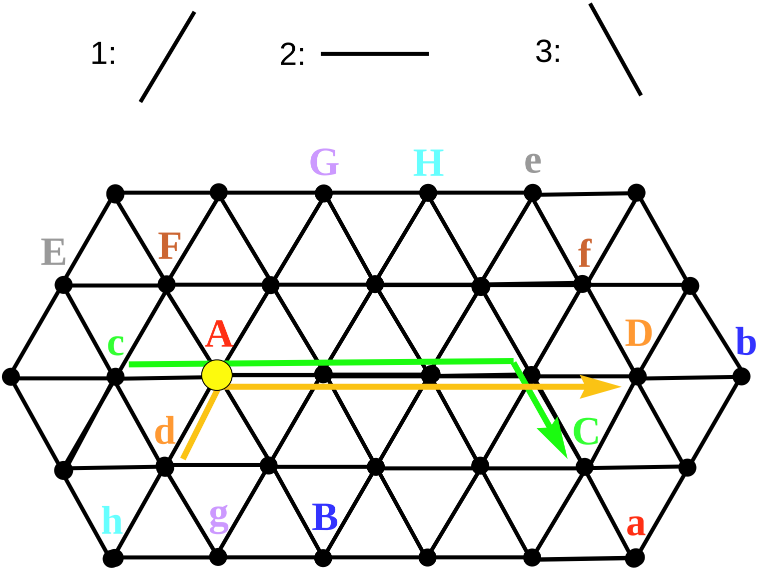
Another non-optimal intuitive algorithm



Another non-optimal intuitive algorithm (2)



Another non-optimal intuitive algorithm (3)



Description of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*:**
  - **a)** If there are packets with **negative components**, send them **immediately** along the direction of this component.
  - **b)** If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the first* packet of each queue.

Description of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*: Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*: At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*:**

  a) If there are packets with **negative components**, send them **immediately** along the direction of this component.

  b) If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Description of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.

- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.

- ***Transmission phase*:**

  a) If there are packets with **negative components**, send them **immediately** along the direction of this component.

  b) If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Description of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*:**
  - **a)** If there are packets with **negative components**, send them **immediately** along the direction of this component.
  - **b)** If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Description of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.

- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.

- ***Transmission phase*:**

  a) If there are packets with **negative components**, send them **immediately** along the direction of this component.

  b) If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Routing of the packets according to $\mathcal{A}$

- Algorithm $\mathcal{A}$ defines for each packet **two directions of movement** (except if a packet has only one non-zero component)
- For instance:
  - if the packet address is of the type $(-, 0, +) \rightarrow$ this packet goes first in the direction $-x$, and after in $+z$

    $\rightarrow$ We symbolize this rule by the arrow 

  - the routing of the address $(+, -, 0)$ is represented by 

Routing of the packets according to $\mathcal{A}$

- Algorithm $\mathcal{A}$ defines for each packet **two directions of movement** (except if a packet has only one non-zero component)
- For instance:
  - if the packet address is of the type $(-, 0, +) \rightarrow$
    this packet goes first in the direction $-x$, and after in $+z$

    $\rightarrow$ We symbolize this rule by the arrow 
  - the routing of the address $(+, -, 0)$ is represented by 

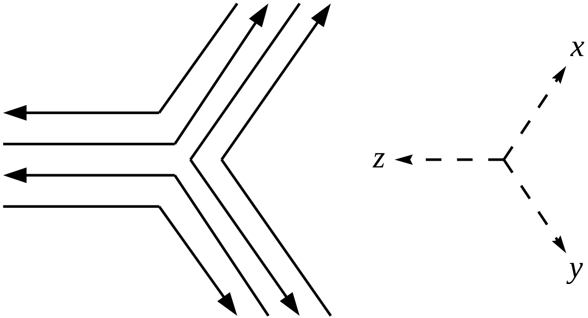
Routing of the packets according to $\mathcal{A}$

- Algorithm $\mathcal{A}$ defines for each packet **two directions of movement** (except if a packet has only one non-zero component)
- For instance:
  - if the packet address is of the type $(-, 0, +) \rightarrow$
    this packet goes first in the direction $-x$, and after in $+z$

    $\rightarrow$ We symbolize this rule by the arrow 
  - the routing of the address $(+, -, 0)$ is represented by 

Routing the packets (2)

In this figure all the routing rules are summarized:



Correctness of Algorithm $\mathcal{A}$

At each node u of the network:

- ***Preprocessing*****:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*****:** At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*****:**

  a) If there are packets with **negative components**, send them **immediately** along the direction of this component.

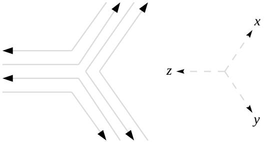
  b) If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Correctness (2)

- *Key observation:*

  Packets can only **wait**, possibly, during their **last direction**.

  - this is because if two packets meet when their first direction is not finished yet, they must have the same origin node → contradiction.



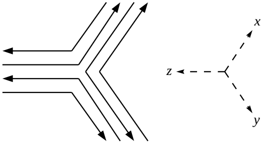
- Thus, in **a)** there can be **at most one packet with negative component at each outgoing edge** → there is no ambiguity.

Correctness (2)

- *Key observation:*

  Packets can only **wait**, possibly, during their **last direction**.

  - this is because if two packets meet when their first direction is not finished yet, they must have the same origin node $\rightarrow$ contradiction.



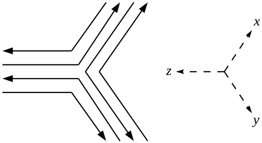
- Thus, in **a)** there can be **at most one packet with negative component at each outgoing edge** $\rightarrow$ there is no ambiguity.

Correctness (2)

- *Key observation:*

  Packets can only **wait**, possibly, during their **last direction**.

  - this is because if two packets meet when their first direction is not finished yet, they must have the same origin node → contradiction.



- Thus, in **a)** there can be **at most one packet with negative component at each outgoing edge** → there is no ambiguity.

Correctness (3)

At each node u of the network:

- ***Preprocessing*:** Initially, if there is a packet at u, compute the relative address $D - S$ of the message in the shortest path form, and add this information to the message.
- ***Reception phase*:** At each step, when a packet is received at u, its relative address is updated.
- ***Transmission phase*:**
  - **a)** If there are packets with **negative components**, send them **immediately** along the direction of this component.
  - **b)** If not, for each outgoing edge order the packets according to **decreasing number of remaining steps**, and send *the* **first** packet of each queue.

Correctness (4)

- All the packets in in **b)** are moving along their **last direction**
  - their negative component is already finished, otherwise they would be in **a)**

- Thus, since each node is the destination of at most one packet, in **b)** *the* **packet with maximum remaining length at each outgoing edge is unique.**

Correctness (4)

- All the packets in in **b)** are moving along their **last direction**
  - their negative component is already finished, otherwise they would be in **a)**
- Thus, since each node is the destination of at most one packet, in **b)** *the* **packet with maximum remaining length at each outgoing edge is unique**.

Optimality

- Using this algorithm, at each step **all the packets with maximum remaining distance move**

  $\longrightarrow$ every step the maximum remaining distance over all packets decreases by one

  $\longrightarrow$ the total running time is at most ℓ_{max}, **meeting the lower bound**.

- It is a **distributed**, **oblivious** and **translation invariant** algorithm.

Optimality

- Using this algorithm, at each step **all the packets with maximum remaining distance move**

  $\longrightarrow$ every step the maximum remaining distance over all packets decreases by one

  $\longrightarrow$ the total running time is at most ℓ_{max}, **meeting the lower bound**.

- It is a **distributed**, **oblivious** and **translation invariant** algorithm.

Optimality

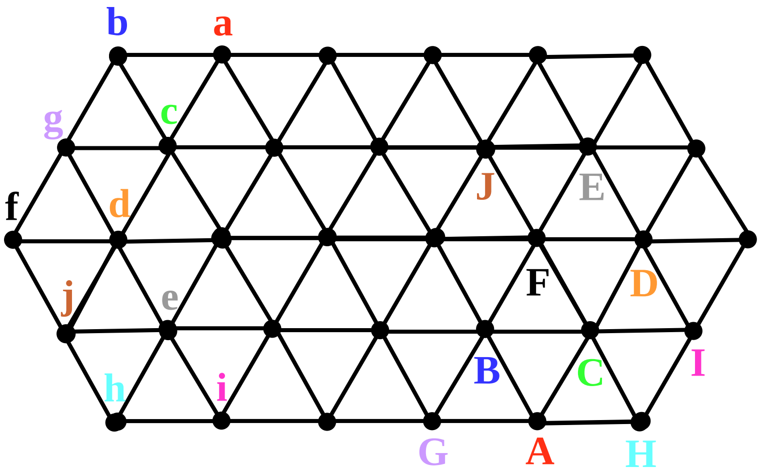
- Using this algorithm, at each step **all the packets with maximum remaining distance move**

  $\longrightarrow$ every step the maximum remaining distance over all packets decreases by one

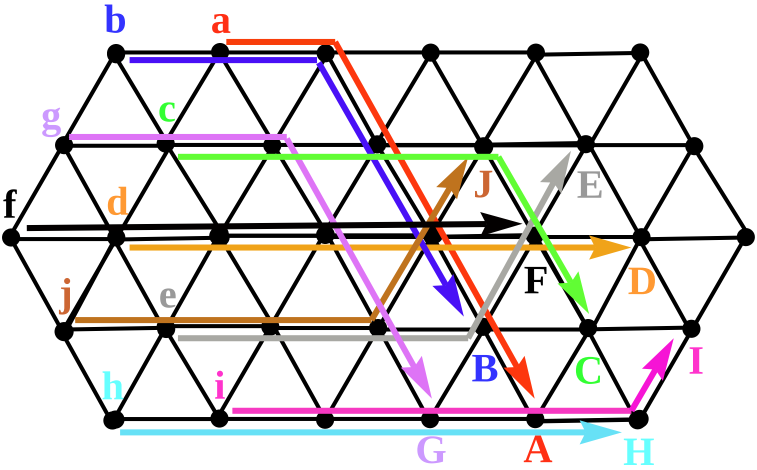
  $\longrightarrow$ the total running time is at most ℓ_{max}, **meeting the lower bound**.

- It is a **distributed**, **oblivious** and **translation invariant** algorithm.

Final example



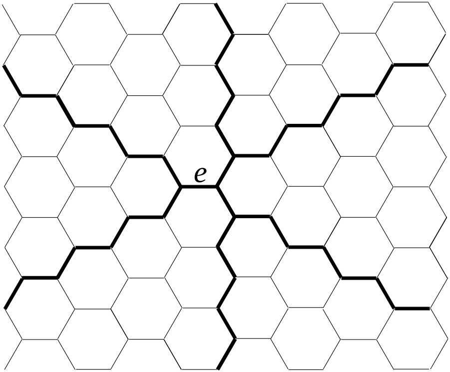
Final example (2)



Permutation Routing on Hexagonal Grids

Hexagonal grid

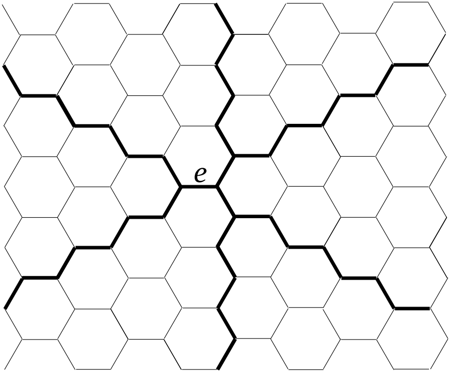
- One can define 3 types of *zigzag chains*:



- Any shortest path uses at most 2 types of zigzag chains

Hexagonal grid

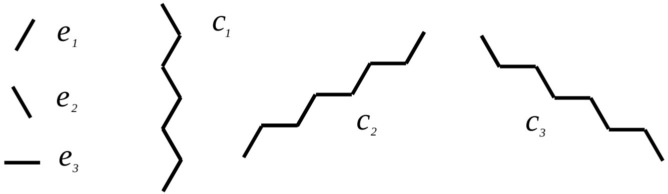
- One can define 3 types of *zigzag chains*:



- Any shortest path uses at most 2 types of zigzag chains

Idea

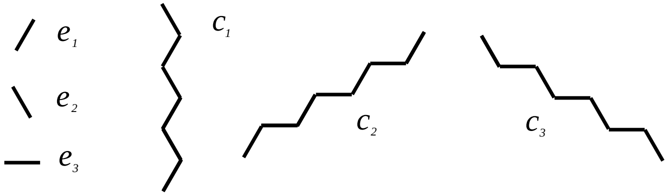
- There are 3 types of edges and 3 types of chains:



- Each edge belongs to exactly 2 different chains, and conversely each chain is made of 2 types of edges.
- Any 2 chains of different type intersect exactly on one edge.
- We can define 2 phases in such a way that at each phase, each type of chain uses only one type of edge.

Idea

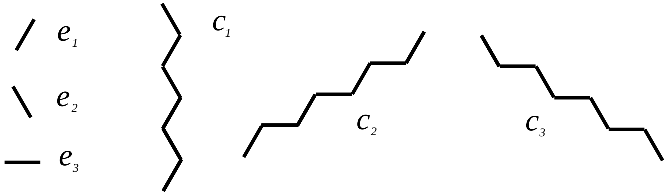
- There are 3 types of edges and 3 types of chains:



- Each edge belongs to exactly 2 different chains, and conversely each chain is made of 2 types of edges.
- Any 2 chains of different type intersect exactly on one edge.
- We can define 2 phases in such a way that at each phase, each type of chain uses only one type of edge.

Idea

- There are 3 types of edges and 3 types of chains:



- Each edge belongs to exactly 2 different chains, and conversely each chain is made of 2 types of edges.
- Any 2 chains of different type intersect exactly on one edge.
- We can define 2 phases in such a way that at each phase, each type of chain uses only one type of edge.

Optimal algorithm

At each node of the network:

1) During the **first step**, move all packets along the direction of their **negative component**. If a packet's address has only a positive component, move it along this direction.
2) From now on, **change alternatively** between Phase 1 and Phase 2.
3) At each step (the same for both phases):
   a) If there are packets with negative components, send them immediately along the direction of this component.
   b) If not, for each outgoing edge order the packets according to decreasing number of remaining steps, and send the first packet of each queue.

Running time

- Every 2 steps (one of Phase 1 and one of Phase 2) the maximum remaining distance over all packets decreases by one.
- During the first step all packets decrease their remaining distance by one.
- Thus, the total running time is $1 + 2(\ell_{max} - 1) = 2\ell_{max} - 1$.
- It can also be proved that $2\ell_{max} - 1$ is a lower bound.
- Thus, this algorithm is optimal.

Running time

- Every 2 steps (one of Phase 1 and one of Phase 2) the maximum remaining distance over all packets decreases by one.
- During the first step all packets decrease their remaining distance by one.
- Thus, the total running time is $1 + 2(\ell_{max} - 1) = 2\ell_{max} - 1$.
- It can also be proved that $2\ell_{max} - 1$ is a lower bound.
- Thus, this algorithm is optimal.

Running time

- Every 2 steps (one of Phase 1 and one of Phase 2) the maximum remaining distance over all packets decreases by one.
- During the first step all packets decrease their remaining distance by one.
- Thus, the total running time is $1 + 2(\ell_{max} - 1) = 2\ell_{max} - 1$.
- It can also be proved that $2\ell_{max} - 1$ is a lower bound.
- Thus, this algorithm is optimal.

(ℓ, k)-Routing

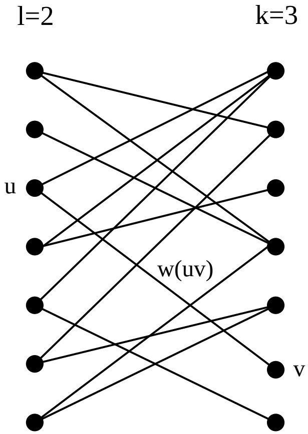
Algorithm (in any grid)

- **Recall**: each node can send at most ℓ packets and receive at most k packets
- **Idea**: represent the request set as a weighted bipartite graph H:
  - split each vertex of the original graph
  - u and v are adjacent if u wants to send a packet to v
  - for each edge uv, let $w(uv)$ be the length of a shortest path from u to v on the grid

Algorithm (in any grid)

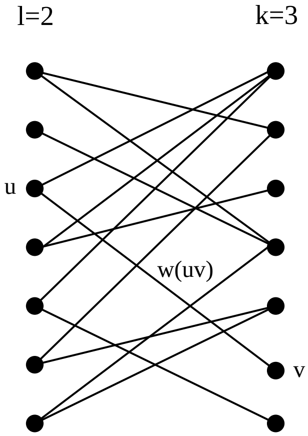
- **Recall**: each node can send at most ℓ packets and receive at most k packets
- **Idea**: represent the request set as a weighted bipartite graph H:
  - split each vertex of the original graph
  - u and v are adjacent if u wants to send a packet to v
  - for each edge uv, let $w(uv)$ be the length of a shortest path from u to v on the grid

Example



- **Fact**: each matching in H corresponds to an instance of a permutation routing problem $\rightarrow$ it can be solved optimally

Example



- **Fact**: each matching in H corresponds to an instance of a permutation routing problem $\longrightarrow$ it can be solved optimally

New problem

- **Problem**: find $m := \max\{\ell, k\}$ matchings in H: $M_1, \ldots, M_m$
- Let $c(M_i) := \max\{w(e) | e \in M_i)\}$, $i = 1, \ldots, m$
- **Objective function**:

$$\min \sum_{i=1}^{m} c(M_i)$$

- **Fact**: $\min \sum_{i=1}^{m} c(M_i)$ is the running time of routing a (ℓ, k)-routing instance using this algorithm
- But we suspect that this problem is **NP-complete**... ☹

New problem

- **Problem**: find $m := \max\{\ell, k\}$ matchings in H: $M_1, \ldots, M_m$
- Let $c(M_i) := \max\{w(e) | e \in M_i)\}$, $i = 1, \ldots, m$
- **Objective function**:

$$\min \sum_{i=1}^{m} c(M_i)$$

- **Fact**: $\min \sum_{i=1}^{m} c(M_i)$ is the running time of routing a (ℓ, k)-routing instance using this algorithm
- But we suspect that this problem is **NP-complete**... ☹

New problem

- **Problem**: find $m := \max\{\ell, k\}$ matchings in H: $M_1, \ldots, M_m$
- Let $c(M_i) := \max\{w(e) | e \in M_i)\}$, $i = 1, \ldots, m$
- **Objective function**:

$$\min \sum_{i=1}^{m} c(M_i)$$

- **Fact**: $\min \sum_{i=1}^{m} c(M_i)$ is the running time of routing a (ℓ, k)-routing instance using this algorithm
- But we suspect that this problem is **NP-complete**... ☹

Summary and further research

- We have described optimal permutation routing algorithms for triangular and hexagonal grids
- We have also optimal algorithms for the $(1, k)$-routing problem
- It remains to solve the $(\ell - k)$-routing
- Permutation routing on 3-circulant graphs is still a challenging open problem...

Summary and further research

- We have described optimal permutation routing algorithms for triangular and hexagonal grids
- We have also optimal algorithms for the $(1, k)$-routing problem
- It remains to solve the $(\ell - k)$-routing
- Permutation routing on 3-circulant graphs is still a challenging open problem...

Thanks!