Efficient Algorithms for Computing Shortest Paths in \( k \)-Loop Networks

D. J. Guan
(Collaborated with M. Y. Chen)
Department of Computer Science
National Sun Yat-Sen University
Kaohsiung, Taiwan 80424

guan@cse.nsysu.edu.tw

July 26, 2009
Contents

1. double-loop and triple-loop networks

2. Shortest path in directed double-loop networks

3. Shortest path in undirected double-loop networks

4. Shortest path in directed double-loop networks

5. Shortest path in directed hyper-L triple-loop networks
**$k$-loop networks**

$G(n; h_1, h_2, \ldots, h_k)$

Vertexes: $\mathbb{Z}_n = \{0, 1, \ldots, n - 1\}$.

Directed $k$-loop networks:
Edges: $E_i = \{(u, v) \mid v = u + h_i, u \in \mathbb{Z}_n\}, i = 1, 2, \ldots, k$.

Undirected $k$-loop networks:
Edges: $E_i = \{(u, v) \mid v = u \pm h_i, u \in \mathbb{Z}_n\}, i = 1, 2, \ldots, k$. 
\[ G(13; 3, 5) \]
Properties of a $k$-loop network

1. more reliable

2. smaller diameter

3. strongly connected iff $\gcd(n, h_1, h_2, \ldots, h_k) = 1$

4. $G(n, h_1, h_2, \ldots, h_k) \cong G(n, h'_1, h'_2, \ldots, h'_k)$ iff

\[ \exists a, \gcd(a, n) = 1, \ h_i \equiv ah'_i, \quad i = 1, 2, \ldots, k. \]
Message routing

Find a path to send a message from its source to its destination.

A message routing algorithm is *optimal* if it always sends the message by using one of the shortest paths.
Optimal message routing strategies

- pre-compute shortest paths and store a routing table at each node
  - few graphs have compact routing tables,
    most graphs need $O(n)$ space at each node
  - difficult for fault-tolerant routing

- dynamic computation of shortest paths
  - can be adopted to fault-tolerant routing

- very efficient algorithm ($O(1)$ time with pre-processing) for computing shortest paths
  - can be adopted to fault-tolerant routing
  - need only $O(1)$ spaces.
An optimal message routing algorithm for directed double-loop networks

At each vertex \( u \), execute

\[
\text{route}(v: \text{destination}, m: \text{message}) \]

\[
\text{if } (v = u) \{ \\
\quad \text{receive the message } m; \\
\} \text{ else } \{ \\
\quad \text{solve } h_1 x \equiv v - u \pmod{n} \text{ for } x; \\
\quad \text{solve } h_2 y \equiv v - u \pmod{n} \text{ for } y; \\
\quad \text{if } (x = \infty \text{ or } w_2 x > w_1 y) \\
\qquad \text{send the message } m \text{ to } (u + h_2) \pmod{n} \\
\text{else} \\
\qquad \text{send the message } m \text{ to } (u + h_1) \pmod{n} \\
\}
Solve

\[ h_1 x \equiv v - u \pmod{n} \]
\[ h_2 y \equiv v - u \pmod{n} \]

separately, instead of solving the optimization problem:

\[ \text{minimize } w_1 a + w_2 b \]
subject to

\[ h_1 a + h_2 b \equiv v - u \pmod{n} \]
Solve $h_1 x \equiv v - u \pmod{n}$

Let $d_1 = \gcd(n, h_1)$.

If $d_1$ does not divide $(v - u) \pmod{n}$, then the equation has no solutions.

Otherwise, solve

$$
\left( \frac{h_1}{d_1} \right)^x \equiv \frac{(v - u) \mod{n}}{d_1} \pmod{\frac{n}{d_1}}
$$

$$
x = \frac{(v - u) \mod{n}}{d_1} \left( \frac{h_1}{d_1} \right)^{-1} \mod{\frac{n}{d_1}}
$$

$$
\left( \frac{h_1}{d_1} \right)^{-1} \text{ is the inverse of } \left( \frac{h_1}{d_1} \right) \text{ in } \mathbb{Z}_{\frac{n}{d_1}}.
$$
Note that the inverses \( \left( \frac{h_1}{d_1} \right)^{-1} \) and \( \left( \frac{h_2}{d_2} \right)^{-1} \) do not depend on \( u \) or \( v \), and therefore, it can be precomputed.
Correctness of our algorithm

If $w_1x < w_2y$ then any shortest path from $u$ to $v$ contains at least one edge in $E_1$.
If $w_1x > w_2y$ then any shortest path from $u$ to $v$ contains at least one edge in $E_2$.

If both the equation $h_1x \equiv v - u \pmod{n}$ and the equation $h_2y \equiv v - u \pmod{n}$ have no solutions, then any path from $u$ to $v$ contains at least one edge in $E_1$ and at least one edge in $E_2$.

If the shortest path from $u$ to $v$ contains at least one edge in $E_i, i = 1, 2$, then there exists a shortest path from $u$ to $v$ whose first edge is an edge in $E_i$.
The routing algorithm is an optimal message routing algorithm, and, with $O(\log n)$ pre-processing time, it need only

- 2 subtraction,
- 2 divisions,
- 2 multiplications,
- 4 modulo operations, and
- 1 comparison

to determine the next vertex on the shortest path to which the message must be sent.
Computing shortest paths in constant time

With $O(\log n)$ pre-processing time, the shortest path from any pair of vertex $u$ and $v$ can be computed in constant time for directed and undirected double-loop networks, and some 3-loop networks.

Since these networks are vertex transitive, the shortest path from $u$ to $v$ is the same as the shortest path from 0 to $v - u$. 
Represent the network in \( \mathbb{Z}^2 \)

\[
\begin{array}{cccccc}
7 \\
2 \\
10 \\
5 & 8 & 11 & 1 & 4 \\
0 & 3 & 6 & 9 & 12 \\
\end{array}
\]

\( G(13; 3, 5), \ \bar{d} = 5 \)

\((x, y) \mapsto (xh_1 + yh_2) \mod n\)
In directed double-loop networks, Wong and Coppersmith has shown that the minimum distance diagram must be an L-shape region.
Minimum distance diagram for undirected double-loop networks

\[
\begin{array}{cccccc}
12 & 2 & 5 & \\
4 & 7 & 10 & 0 & 3 & 6 & 9 \\
8 & 11 & 1 &
\end{array}
\]

\[G(13; 3, 5), \quad \vec{d} = 3\]
$G(20; 3, 7), \quad \bar{d} = 5$
Algorithm for shortest path in undirected double-loop networks

We first show that, with $O(\log n)$ pre-processing time, the distance from $u$ to $v$ in a undirected double-loop network $G(n; h_1, h_2)$ can be computed in $O(1)$ time.

Given an undirected double-loop network $G(n; h_1, h_2)$, let

$$f : \mathbb{Z}^2 \rightarrow V \quad f(x, y) = xh_1 + yh_2 \mod n$$

Define

$$S_v = \{(x, y) \in \mathbb{Z}^2 \mid xh_1 + yh_2 \equiv v \pmod{n}\}.$$
Theorem 1 \( S_0 \) is a lattice of rank 2.
1. For every $x, y \in S_0$, $x + y \in S_0$.

2. $(0, 0)$ is the identity element.

3. For every $x, y, z \in S_0$, $(x + y) + z = x + (y + z)$.

4. The inverse of $(x, y)$ is $(-x, -y)$.

Therefore, $S_0$ is an additive subgroup of $\mathbb{Z}^2$. 
Packed basis for $S_0$

There is a packed basis $\{a, b\}$ for $S_0$ of a double-loop network [Žerovnik and Pisanski 1993].

$$\max\{|a|, |b|\} \leq \min\{|a + b|, |a - b|\}$$

They also show that the region

$$M = \{(\alpha a, \beta b) | 0 \leq \alpha, \beta < 1\}$$

contains each vertex in $V$ exactly once.
An algorithm for undirected double-loop networks

- Find an element \((s, t) \in S_v\).

- Solve \(\alpha a + \beta b = (s, t)\) for \((\alpha, \beta)\).

- Find the shortest vector \((\bar{x}, \bar{y})\) in

\[
T_v = \{(s, t) - pa - qb \mid p \in \{\lfloor \alpha \rfloor, \lceil \alpha \rceil\}, \ q \in \{\lfloor \beta \rfloor, \lceil \beta \rceil\} \}.
\]
Correctness of the algorithm

Theorem 2 \((\bar{x}, \bar{y})\) is a solution to

\[ xh_1 + yh_2 \equiv v \pmod{n} \]

with minimum \(|x| + |y|\).
Good for undirected double-loop networks, but not for directed double-loop networks.
Algorithm for directed double-loop networks

Let \( \{a, b\} \) be a packed basis.

Assume that \( a \) is in the first quadrant.

- Find an element \((s, t) \in S_v\).
- Solve \( \alpha a + \beta b = (s, t) \) for \((\alpha, \beta)\).
- Find the shortest vector \((\bar{x}, \bar{y}) \) in

\[
T_v = \{(s, t) - (p - 1)a - qb \mid p \in \lfloor \alpha \rceil, \lceil \alpha \rceil, \quad q \in \lfloor \beta \rceil, \lceil \beta \rceil \}.
\]
Algorithms for directed triple-loop networks

We would like to generalize the algorithm for directed double-loop networks to directed triple-loop networks.

However, the lattice $S_0$ of $k$-loop networks has rank $k$, and finding a shortest non-zero element in $S_0$ is NP-complete if $k > 2$.

For a special class of directed triple-loop network, the hyper-L networks, the generalized algorithm works well.
Hyper-L, $l = 10$, $m = 5$, $n = 3$
Theorem 3 (C. Y. Chen, F. K. Hwang, J. S. Lee, S. J. Shih)
A necessary and sufficient condition for the existence of a Hyper-L directed triple-loop network with parameters \( l, m, n, \ l > m + n \), is, \( \gcd(l^2 - mn, m^2 + ln, n^2 + lm) = 1 \).

Theorem 4 There exists a basis such that the hyper-parallelogram generated by the basis contains at most 8 minimum distance diagrams.

\[ a = (l - m - n, \ l - m - n, \ l - m - n) \]

\[ b = (l, \ - m, \ - n) \]

\[ c = (n, \ - l, \ m) \]
Algorithm for hyper-L directed triple-loop networks

Let \( \{a, b, c\} \) be the basis described above.

- Find an element \( (r, s, t) \in S_v \).
- Solve \( \alpha a + \beta b + \gamma c = (r, s, t) \) for \( (\alpha, \beta, \gamma) \).
- Find the shortest vector \( (\bar{x}, \bar{y}, \bar{z}) \) in
  \[
  T_v = \{(r, s, t) - (p - 1)a - qb - tc\},
  \]
  where \( p \in \{\lfloor \alpha \rfloor, \lceil \alpha \rceil\} \), \( q \in \{\lfloor \beta \rfloor, \lceil \beta \rceil\} \), \( t \in \{\lfloor \gamma \rfloor, \lceil \gamma \rceil\} \).
Conclusions

With $O(\log n)$ pre-processing time, a shortest path of

1. a directed double-loop network can be computed in $O(d)$ time, where $d$ is the distance;
2. an undirected double-loop network can be computed in $O(1)$ time;
3. a directed double-loop network can be computed in $O(1)$ time.
4. a hyper-L, a special class of triple-loop network with parameters $l, m, n$, can be computed in $O(1)$ time.

These algorithms are useful in message routing in the networks.