Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 359 (2006) 69–76
Theoretical Computer Science
www.elsevier.com/locate/tcs

An efficient algorithm to find a double-loop network that realizes a given L-shape☆

Chiuyuan Chen[a,*], James K. Lan[a], Wen-Shiang Tang[b]

[a]*Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan*
[b]*Department of Communication Engineering, National Chiao Tung University, Hsinchu 300, Taiwan*

Received 11 April 2005; received in revised form 19 January 2006; accepted 30 January 2006

Communicated by D.-Z. Du

Abstract

Double-loop networks have been widely studied as an architecture for local area networks. It is well known that the minimum distance diagram of a double-loop network yields an L-shape. Given a positive integer N, it is desirable to find a double-loop network with its diameter being the minimum among all double-loop networks with N nodes. Since the diameter of a double-loop network can be easily computed from its L-shape, one method is to start with a desirable L-shape and then find a double-loop network to realize it. This is a problem discussed by many authors [F. Aguiló, M.A. Fiol, An efficient algorithm to find optimal double loop networks, Discrete Math. 138 (1995) 15–29, R.C. Chan, C.Y. Chen, Z.X. Hong, A simple algorithm to find the steps of double-loop networks, Discrete Appl. Math. 121 (2002) 61–72, C.Y. Chen, F.K. Hwang, The minimum distance diagram of double-loop networks, IEEE Trans. Comput. 49 (2000) 977–979, P. Esqué, F. Aguiló, M.A. Fiol, Double commutative-step diagraphs with minimum diameters, Discrete Math. 114 (1993) 147–157] and it has been open for a long time whether this problem can be solved in $O(\log N)$ time. In this paper, we will provide a simple and efficient $O(\log N)$-time algorithm for solving this problem.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Local area network; Double-loop network; Diameter; L-shape; Algorithm

1. Introduction

A *double-loop network* $DL(N; s_1, s_2)$ has N nodes $0, 1, \ldots, N-1$ and $2N$ links of two types:

$$s_1\text{-links: } i \to i + s_1 \pmod N, \quad i = 0, 1, \ldots, N-1,$$
$$s_2\text{-links: } i \to i + s_2 \pmod N, \quad i = 0, 1, \ldots, N-1.$$

Throughout this paper, N denotes the number of nodes in the network. Double-loop networks have been widely studied as an architecture for local area networks. See [2,11,12,15] for surveys of these networks. A double-loop network is *strongly connected* if for each ordered pair u, v of nodes, there is a path from u to v. Let gcd() denote the greatest common divisor; for example, $\gcd(9, 2, 5) = 1$. Fiol et al. [9] proved that a double-loop network $DL(N; s_1, s_2)$ is strongly connected if and only if $\gcd(N, s_1, s_2) = 1$.

☆ This research was partially supported by the National Science Council of the Republic of China under the Grant NSC94-2115-M-009-006.
* Corresponding author. Tel.: +886 3 5731767.
E-mail address: cychen@mail.nctu.edu.tw (C. Chen).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.01.048

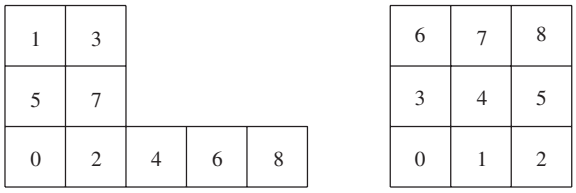

Fig. 1. Examples of the L-shapes.

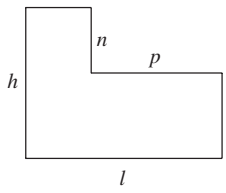

Fig. 2. An L-shape and its parameters l, h, p, n.

When $DL(N; s_1, s_2)$ is strongly connected, we can talk about its minimum distance diagram (MDD). This diagram gives a shortest path from node u to node v for any u, v. Since $DL(N; s_1, s_2)$ is node-symmetric, it suffices to give a shortest path from node 0 to any other node. Let 0 occupy cell (0,0). Then v occupies cell (i, j) if and only if $is_1 + js_2 \equiv v \pmod N$ and $i + j$ is the minimum among all (i', j') satisfying the congruence, where $\equiv$ means congruent modulo N. That is, if v occupies cell (i, j), then a shortest path from 0 to v is achieved through taking i s_1-links and j s_2-links (in any order). The MDD of $DL(N; s_1, s_2)$ includes every node exactly once (in case of two shortest paths, the convention is to choose the cell with the smaller row index, i.e., the smaller j). Note that in a cell (i, j), i is the column index and j is the row index.

Wong and Coppersmith [16] proved that the MDD of a double-loop network is always an L-shape (see Fig. 1); a rectangle is considered a degeneration. An L-shape can be determined by four geometric parameters l, h, p, n as shown in Fig. 2. These four parameters are the lengths of four of the six segments on the boundary of the L-shape. For example, $DL(9; 2, 5)$ in Fig. 1 has $l = 5$, $h = 3$, $p = 3$, and $n = 2$. Clearly,

$$N = \ell h - pn.$$

It was proven in [9,10,4] that there exists a double-loop network $DL(N; s_1, s_2)$ realizing the L-shape(l, h, p, n) if and only if

$$l > n, \quad h \geqslant p \quad \text{and} \quad \gcd(l, h, p, n) = 1. \tag{1.1}$$

The *diameter* $d(N; s_1, s_2)$ of a double-loop network $DL(N; s_1, s_2)$ is the largest distance between any pair of nodes. It represents the maximum transmission delay between any two nodes. The diameter of a double-loop network $DL(N; s_1, s_2)$ can be easily computed from its L-shape(l, h, p, n) by the equation

$$d(N; s_1, s_2) = \max\{l + h - p,\ l + h - n\} - 2.$$

Let $d(N)$ denote the optimal diameter of a double-loop network with N nodes. Wong and Coppersmith [16] showed that $d(N) \geqslant \lceil \sqrt{3N} \rceil - 2$. Given a positive integer N, it is desirable to find a double-loop network with N nodes such that its diameter is $d(N)$. This is a problem discussed by many authors; see [1,3–5,7–10,13,16]. Since the diameter of a double-loop network can be readily computed from its L-shape, one method is to start with a desirable L-shape and then find a double-loop network to realize it. This is Problem 2 described below.

The following two problems have been discussed by many authors:

Problem 1. Given a double-loop network $DL(N; s_1, s_2)$, find its L-shape(l, h, p, n) and its diameter.

Problem 2. Given an L-shape(l, h, p, n), find a double-loop network $DL(N; s_1, s_2)$ that realizes it.

For Problem 1, Cheng and Hwang [5] have proposed a very elegant O(log N)-time solution. As for Problem 2, three algorithms have been proposed in the following literature: the Smith normalization method [1,8], the sieve method [4], and the Chan–Chen–Hong's algorithm (the CCH algorithm in short) [3]. In particular, the Smith normalization method is based on finding the Smith normal form of a matrix and it requires matrix operations [14]; see [3] for an explicit algorithm of this method. The CCH algorithm is based on the Smith normalization method, but unlike the Smith normalization method, it does not require any matrix operation and thus greatly simplifies the computation. Both the Smith normalization method and the CCH algorithm take O((log N)^2) time; see [3]. The sieve method is based on the sieve method in number theory; it uses the Euclidean algorithm and is very easy to be implemented. The exact time complexity of the sieve method is not known; however, Chan et al. [3] showed that it is upper bounded by O($P_{\eta(N)}$ log N), where $\eta(N)$ is the number of prime factors of N and P_i is the ith prime (i.e., $P_1 = 2$, $P_2 = 3$, etc.).

It has been open for a long time whether Problem 2 can be solved in O(log N) time. In this paper, we will show that there exists a family of L-shapes such that for each L-shape in this family, we have $P_{\eta(N)} \geqslant \sqrt{\log N}$. We will also improve the CCH algorithm to derive a simple and efficient O(log N)-time algorithm for solving Problem 2. We now summarize current results of Problems 1 and 2 in the following tables.

Algorithm for Problem 1	Worst-case time complexity
Cheng–Hwang's algorithm [5]	O(log N)

Algorithm for Problem 2	Worst-case time complexity
Smith normalization method [1,8]	O((log N)^2)
Sieve method [4]	O($P_{\eta(N)}$ log N)
CCH algorithm [3]	O((log N)^2)
Our algorithm	O(log N)

This paper is organized as follows. In Section 2, we show that there exists a family of L-shapes such that for each L-shape in this family, $P_{\eta(N)} \geqslant \sqrt{\log N}$. In Section 3, we describe our O(log N)-time algorithm. The concluding remarks are given in Section 4.

2. The time complexity of the sieve method

Given an L-shape, Chen and Hwang [4] (see also [12]) proposed the following method, which is based on the sieve method in number theory, to find a double-loop network that realizes the given L-shape.

The Sieve Method [4]
Input: An L-shape(l, h, p, n) that satisfies (1.1).
Output: A double-loop network $DL(N; s_1, s_2)$ that realizes the given L-shape(l, h, p, n).
1. Let $N = lh - pn$ and $k = 0$.
2. Let $a_k = kn + h$ and $b_k = kl + p$.
3. If gcd(N, a_k, b_k) $= 1$, then return N, $s_1 = a_k \pmod N$, $s_2 = b_k \pmod N$ and stop the algorithm; otherwise, increase k by 1 and go to Step 2.

We now prove that there exists a family of L-shapes such that for each L-shape in this family, we have $P_{\eta(N)} \geqslant \sqrt{\log N}$. First a lemma.

Lemma 1. *Let $P_1, P_2, \ldots, P_t$ be the smallest t primes, where $t \geqslant 2$ and $P_1 < P_2 < \cdots < P_t$. If $N = P_1 \times P_2 \times \cdots \times P_t$, then $P_{\eta(N)} \geqslant \sqrt{\log N}$.*

Proof. Let $N = P_1 \times P_2 \times \cdots \times P_t$. Then $\eta(N) = t$ and $N \leqslant (P_t)! \leqslant P_t^{P_t}$. Hence $\log N \leqslant P_t \log P_t$ and $\log\log N \leqslant \log P_t + \log\log P_t \leqslant 2\log P_t$. So $\log P_t \geqslant (\log\log N)/2 = \log\sqrt{\log N}$. Hence $P_{\eta(N)} = P_t \geqslant \sqrt{\log N}$. □

Theorem 2. *There exists a family of L-shapes such that for each L-shape in this family, $P_{\eta(N)} \geqslant \sqrt{\log N}$ and the sieve method has to execute the Euclidean algorithm at least $P_{\eta(N)}$ times to solve Problem* 2.

Proof. First we construct an L-shape. Let t be an integer in $\{2, 3, \ldots, 100\,000\}$. Let $P_1, P_2, \ldots, P_t$ be the smallest t primes and suppose $P_1 < P_2 < \cdots < P_t$. Let

$$d = P_1 \times P_2 \times \cdots \times P_{t-1}.$$

It is not difficult to verify that for each t in $\{2, 3, \ldots, 100\,000\}$, $P_t \leqslant 2P_{t-1}$ holds and thus we have $P_t < d$ and $\lfloor 2d/P_t \rfloor \geqslant 1$. Since $2d$ is not divisible by P_t, $\lceil 2d/P_t \rceil$ and $\lfloor 2d/P_t \rfloor$ are two consecutive integers. Thus

$$\lceil 2d/P_t \rceil - \lfloor 2d/P_t \rfloor = 1 \tag{2.1}$$

and

$$\gcd(\lceil 2d/P_t \rceil, \lfloor 2d/P_t \rfloor) = 1. \tag{2.2}$$

Set

$$(l, h, p, n) = (P_t\lceil 2d/P_t \rceil - d, d, d, P_t\lfloor 2d/P_t \rfloor - d).$$

It is easy to see that $l > 0,\ h > 0,\ p \geqslant 0,\ n \geqslant 0,\ l \geqslant p$ and $h \geqslant n$ hold. Thus (l, h, p, n) forms an L-shape.

We claim that there exists a double-loop network $DL(N; s_1, s_2)$ realizing the L-shape (l, h, p, n). To prove this claim, we have to prove that the three constraints $l > n,\ h \geqslant p$ and $\gcd(l, h, p, n) = 1$ in (1.1) hold. It is easy to see that $l > n$ and $h \geqslant p$ hold. By the definition of P_t and d,

$$\gcd(P_t, d) = 1. \tag{2.3}$$

Hence

$$\begin{aligned}\gcd(l, h, p, n) &= \gcd(P_t\lceil 2d/P_t \rceil, P_t\lfloor 2d/P_t \rfloor, d)\\ &= \gcd(\lceil 2d/P_t \rceil, \lfloor 2d/P_t \rfloor, d) \quad \text{(by (2.3))}\\ &= 1 \quad \text{(by (2.2)).}\end{aligned}$$

We now prove that for the L-shape(l, h, p, n), $P_{\eta(N)} \geqslant \sqrt{\log N}$ occurs and the sieve method has to execute the Euclidean algorithm at least $P_{\eta(N)}$ times to solve Problem 2. Recall that $N = lh - pn$. Thus $N = dP_t(\lceil 2d/P_t \rceil - \lfloor 2d/P_t \rfloor) \overset{(2.1)}{=} dP_t = P_1 \times P_2 \times \cdots \times P_t$. By Lemma 1, $P_{\eta(N)} \geqslant \sqrt{\log N}$. Let F be the set of prime factors of N and let F_k be the set of prime factors of $\gcd(a_k, b_k)$. Then $F = \{P_1, P_2, \ldots, P_t\}$. $F_0 = \{P_1, P_2, \ldots, P_{t-1}\}$ since $\gcd(a_0, b_0) = \gcd(h, p) = d$. $F_1 = \{P_t\}$ since $\gcd(a_1, b_1) = \gcd(n + h, l + p) = \gcd(P_t\lfloor 2d/P_t \rfloor, P_t\lceil 2d/P_t \rceil) \overset{(2.2)}{=} P_t$. Recall that $P_1, P_2, \ldots, P_t$ are the smallest t primes and $P_1 = 2$, $P_2 = 3$, $P_3 = 5$, and so on. It was proved in [4] that if $f \in F$ appears in F_k for some k and k_f is the smallest such k, then f appears in every fth k after k_f. Therefore

$P_1 \in F$ appears in $\gcd(a_0, b_0),\ \gcd(a_2, b_2),\ \gcd(a_4, b_4),\ \gcd(a_6, b_6),$ and so on;
$P_2 \in F$ appears in $\gcd(a_0, b_0),\ \gcd(a_3, b_3),\ \gcd(a_6, b_6),\ \gcd(a_9, b_9),$ and so on;
$P_3 \in F$ appears in $\gcd(a_0, b_0),\ \gcd(a_5, b_5),\ \gcd(a_{10}, b_{10}),\ \gcd(a_{15}, b_{15}),$ and so on;
...
$P_{t-1} \in F$ appears in $\gcd(a_0, b_0),\ \gcd(a_{P_{t-1}}, b_{P_{t-1}}),\ \gcd(a_{2\times P_{t-1}}, b_{2\times P_{t-1}}),$ and so on;
$P_t \in F$ appears in $\gcd(a_1, b_1),\ \gcd(a_{1+P_t}, b_{1+P_t}),\ \gcd(a_{1+2\times P_t}, b_{1+2\times P_t}),$ and so on.

Thus the first k such that $\gcd(N, a_k, b_k) = 1$ is P_t. Since $P_t = P_{\eta(N)}$ and since each iteration of the sieve method involves the Euclidean algorithm, the sieve method has to execute the Euclidean algorithm at least $P_{\eta(N)}$ times to solve Problem 2. Since t can be chosen arbitrarily from the set $\{2, 3, \ldots, 100\,000\}$, we have this theorem. □

3. Our algorithm

Let us describe the CCH algorithm first.

The CCH algorithm [3]
Input: An L-shape(l, h, p, n) that satisfies (1.1).
Output: A double-loop network $DL(N; s_1, s_2)$ that realizes the given L-shape(l, h, p, n).
1. Find $r_1 = \gcd(l, -n)$.
2. Find integers α_1 and β_1 such that $\alpha_1 l + \beta_1(-n) = r_1$.
3. Find $r_2 = \gcd(r_1, -\alpha_1 p + \beta_1 h)$.
4. Find integers α_2 and β_2 such that $\alpha_2 r_1 + \beta_2(-\alpha_1 p + \beta_1 h) = r_2$ and $\gcd(\beta_2, r_2) = 1$.
5. Return $N = lh - pn$, $s_1 = \alpha_2 n - \beta_2 h \pmod N$, $s_2 = \alpha_2 l - \beta_2 p \pmod N$ and stop the algorithm.

For example, let $(l, h, p, n) = (5, 3, 3, 2)$. Then $r_1 = 1$, $\alpha_1 = 1$, $\beta_1 = 2$, $r_2 = 1$, $\alpha_2 = -2$ and $\beta_2 = 1$. Thus $N = 9$, $s_1 = -7 \pmod 9 = 2$, $s_2 = -13 \pmod 9 = 5$. It can be verified from Fig. 1 that double-loop network $DL(9; 2, 5)$ realizes the L-shape(5,3,3,2).

Recall that $N = lh - pn$. The CCH algorithm takes $O((\log N)^2)$ time because: Steps 1–3 and 5 take $O(\log N)$ time and Step 4 takes $O((\log N)^2)$ time [3]. *Our algorithm is the same as the CCH algorithm except the implementation of Step* 4. In the CCH algorithm, Step 4 was implemented by an $O((\log N)^2)$-time algorithm, while in our algorithm, Step 4 is implemented by an $O(\log N)$-time algorithm. We now describe the details.

It was proved in [3] that

Lemma 3 (*Chan et al. [3]*). *If* α, a, β, b *are integers* (*not all zero*) *such that* $\alpha a + \beta b = 1$, *then* $\gcd(a, \beta) = 1$.

It is well known that

Lemma 4 (*Cormen et al. [6]*). *If* a *and* b *are integers*, *not both zero*, *then there exist integers* α *and* β *such that* $\alpha a + \beta b = \gcd(a, b)$. *Moreover*, *if* $|a| \geqslant |b|$, *then* α *and* β *can be found in* $O(\log |b|)$ *time*.

Step 4 of the CCH algorithm is based on the following theorem [3]:

Theorem 5 (*Chan et al. [3]*). *If* a *and* b *are integers*, *not both zero*, *then there exist integers* x *and* y *such that* $xa + yb = \gcd(a, b)$ *and* $\gcd(y, \gcd(a, b)) = 1$.

We now prove that

Theorem 6. *If* a *and* b *are integers*, *not both zero*, *then there exist integers* x *and* y *such that* $xa + yb = \gcd(a, b)$ *and* $\gcd(y, \gcd(a, b)) = 1$. *Moreover*, *if* $|a| \geqslant |b|$, *then* x *and* y *can be found in* $O(\log |b|)$ *time*.

Proof. It was proved in [3] that if a and b are integers, not both zero, then there exist integers x and y such that $xa + yb = \gcd(a, b)$ and $\gcd(y, \gcd(a, b)) = 1$. For completeness of this paper, we describe the proof here. Set $r = \gcd(a, b)$ for easy writing. By Lemma 4, there exist integers α and β such that

$$\alpha a + \beta b = r.$$

If $\gcd(\beta, r) = 1$, then we are done. In the following, assume that $\gcd(\beta, r) = k > 1$. Let r' be the largest integer such that

$$r' \mid r \quad \text{and} \quad \gcd(r', k) = 1. \tag{3.1}$$

Then either $r' = 1$ or $r' > 1$. In the former case, every prime factor of r is also a prime factor of k. In the latter case, every prime factor of r is either a prime factor of k or a prime factor of r'. Let

$$a' = a/r \quad \text{and} \quad b' = b/r. \tag{3.2}$$

Note that $\gcd(r', \beta) = 1$; otherwise, we will have $\gcd(\beta, r) > k$. Since $\alpha a + \beta b = r$, we have

$$\alpha a' + \beta b' = 1.$$

By Lemma 3, we have $\gcd(a', \beta) = 1$. Since $\gcd(a', \beta) = 1$ and $k \mid \beta$, we have $\gcd(a', k) = 1$. Since $k \mid \beta$ and $\gcd(r', k) = 1$ and $\gcd(a', k) = 1$, we have

$$\gcd(\beta - r'a', k) = 1. \tag{3.3}$$

Since $\gcd(r', \beta) = 1$ and $r' \mid r'a'$, we have

$$\gcd(\beta - r'a', r') = 1. \tag{3.4}$$

Recall that either $r' = 1$ or $r' > 1$. In the former case, by (3.3) and (3.4) and the fact that every prime factor of r is also a prime factor of k, we have

$$\gcd(\beta - r'a', r) = 1.$$

In the latter case, by (3.3) and (3.4) and the fact that every prime factor of r is either a prime factor of k or a prime factor of r', we also have

$$\gcd(\beta - r'a', r) = 1.$$

Let

$$x = \alpha + r'b' \quad \text{and} \quad y = \beta - r'a'. \tag{3.5}$$

Then $xa + yb = (\alpha + r'b')a + (\beta - r'a')b = r$ and $\gcd(y, r) = \gcd(\beta - r'a', r) = 1$. From the above, if a and b are integers, not both zero, then there exist integers x and y such that $xa + yb = \gcd(a, b)$ and $\gcd(y, \gcd(a, b)) = 1$.

We now prove that if $|a| \geqslant |b|$, then x and y can be found in $O(\log |b|)$ time.

Claim 1.

$$r' = \frac{r}{\gcd(k^{\lfloor \log_2 r \rfloor}, r)} \textit{ is the largest integer satisfying } (3.1).$$

Proof of Claim 1. Recall that $k|r$. Assume that

$$k = p_1^{s_1} p_2^{s_2} \cdots p_m^{s_m},$$

where p_i's are distinct prime factors of k. Also assume that

$$r = p_1^{t_1} p_2^{t_2} \cdots p_m^{t_m} p_{m+1}^{t_{m+1}} p_{m+2}^{t_{m+2}} \cdots p_n^{t_n},$$

where p_j's are distinct prime factors of r. Note that when $p_{m+1}, p_{m+2}, \ldots, p_n$ do not exist (this case occurs when k contains every prime factor of r), we will simply say that $p_{m+1}^{t_{m+1}} p_{m+2}^{t_{m+2}} \cdots p_n^{t_n} = 1$. It is clear that if r' is the largest integer satisfying (3.1), then

$$r' = p_{m+1}^{t_{m+1}} p_{m+2}^{t_{m+2}} \cdots p_n^{t_n}.$$

Thus it suffices to prove that

$$\frac{r}{\gcd(k^{\lfloor \log_2 r \rfloor}, r)} = p_{m+1}^{t_{m+1}} p_{m+2}^{t_{m+2}} \cdots p_n^{t_n}.$$

Note that

$$k^{\lfloor \log_2 r \rfloor} = p_1^{\lfloor \log_2 r \rfloor s_1} p_2^{\lfloor \log_2 r \rfloor s_2} \cdots p_m^{\lfloor \log_2 r \rfloor s_m}.$$

Since 2 is the smallest prime, we have

$$t_i \leqslant \lfloor \log_2 r \rfloor \quad \text{for all } i, \ \ 1 \leqslant i \leqslant n.$$

Therefore

$$t_i \leqslant \lfloor \log_2 r \rfloor s_i \quad \text{for all } i, \quad 1 \leqslant i \leqslant m.$$

Thus

$$\gcd(k^{\lfloor \log_2 r \rfloor}, r) = p_1^{t_1} p_2^{t_2} \cdots p_m^{t_m}.$$

So

$$\frac{r}{\gcd(k^{\lfloor \log_2 r \rfloor}, r)} = \frac{p_1^{t_1} p_2^{t_2} \cdots p_m^{t_m} p_{m+1}^{t_{m+1}} p_{m+2}^{t_{m+2}} \cdots p_n^{t_n}}{p_1^{t_1} p_2^{t_2} \cdots p_m^{t_m}} = p_{m+1}^{t_{m+1}} p_{m+2}^{t_{m+2}} \cdots p_n^{t_n}. \qquad \square$$

The following is an algorithm for finding x and y such that $xa + yb = r$ and $\gcd(y, r) = 1$.

ALGORITHM-NEW-EUCLIDEAN
Input: Integers a and b, not both zero, and $r = \gcd(a, b)$.
Output: Integers x and y such that $xa + yb = r$ and $\gcd(y, r) = 1$.
1. Find integers α and β such that $\alpha a + \beta b = r$.
2. Find $k = \gcd(\beta, r)$.
3. If $k = 1$, then let $x = \alpha$, let $y = \beta$, return x and y, and stop this algorithm.
4. If $k \geqslant 2$, find $r' = r/\gcd(k^{\lfloor \log_2 r \rfloor}, r)$.
5. Let $a' = a/r$, $b' = b/r$, $x = \alpha + r'b'$ and $y = \beta - r'a'$. Return x and y.

The correctness of ALGORITHM-NEW-EUCLIDEAN follows from Claim 1, (3.2) and (3.5). We now analyze the time complexity of ALGORITHM-NEW-EUCLIDEAN. We claim that

Claim 2. *If* $|a| \geqslant |b|$, *then ALGORITHM-NEW-EUCLIDEAN takes* $O(\log |b|)$ *time*.

Proof of Claim 2. Since $r = \gcd(a, b)$, we clearly have

$$r \leqslant |b|. \tag{3.6}$$

By Lemma 4, Step 1 takes $O(\log |b|)$ time. By (3.6) and by Lemma 4, Step 2 takes $O(\log |b|)$ time. It is clear that Steps 3 and 5 take $O(1)$ time. In Step 4, computing $k^{\lfloor \log_2 r \rfloor}$ takes $O(\log \lfloor \log_2 r \rfloor) = O(\log \log |b|)$ time. Once $k^{\lfloor \log_2 r \rfloor}$ is known, by (3.6) and by Lemma 4, finding $\gcd(k^{\lfloor \log_2 r \rfloor}, r)$ takes $O(\log |b|)$ time. Hence finding $r' = r/\gcd(k^{\lfloor \log_2 r \rfloor}, r)$ takes $O(\log \log |b|) + O(\log |b|) + 1 = O(\log |b|)$ time, where $+1$ is for the division. Thus Step 4 takes $O(\log |b|)$ time. We have this claim. $\square$

We now have Theorem 6. $\square$

We now describe our algorithm. It is the same as the CCH algorithm except the implementation of Step 4.

Our algorithm
Input: An L-shape(l, h, p, n) that satisfies (1.1).
Output: A double-loop network $DL(N; s_1, s_2)$ that realizes the given L-shape(l, h, p, n).
1. Find $r_1 = \gcd(l, -n)$.
2. Find integers α_1 and β_1 such that $\alpha_1 l + \beta_1(-n) = r_1$.
3. Find $r_2 = \gcd(r_1, -\alpha_1 p + \beta_1 h)$.
4. Uses ALGORITHM-NEW-EUCLIDEAN to find integers α_2 and β_2 such that $\alpha_2 r_1 + \beta_2(-\alpha_1 p + \beta_1 h) = r_2$ and $\gcd(\beta_2, r_2) = 1$.
5. Return $N = lh - pn$, $s_1 = \alpha_2 n - \beta_2 h \pmod N$, $s_2 = \alpha_2 l - \beta_2 p \pmod N$ and stop the algorithm.

Theorem 7. *Our algorithm is correct and it takes* $O(\log N)$ *time*.

Proof. The correctness of our algorithm follows from the correctness of the CCH algorithm and Theorem 6. We now analyze the time complexity. Recall that Steps 1–3 and 5 of the CCH algorithm take $O(\log N)$ time. Thus Steps 1–3

and 5 of our algorithm take O(log N) time. It suffices to prove that Step 4 of our algorithm takes O(log N). Recall that $N = lh - pn$. When Step 4 is performed, inputs to ALGORITHM-NEW-EUCLIDEAN are $a = r_1$, $b = -\alpha_1 p + \beta_1 h$ and $r = \gcd(r_1, -\alpha_1 p + \beta_1 h)$. Since $r_1 = \gcd(l, -n)$, we have $0 < r_1 \leqslant N$ and

$$\min\{|a|, |b|\} = \min\{|r_1|, |-\alpha_1 p + \beta_1 h|\} \leqslant |r_1| = r_1 \leqslant N.$$

By Theorem 6, Step 4 takes O(log N) time. We now have this theorem. □

4. The concluding remarks

This paper considers the problem of finding a double-loop network that realizes a given L-shape. It is well known that if a and b are integers, not both zero, then there exist integers α and β such that $\alpha a + \beta b = \gcd(a, b)$; moreover, if $|a| \geqslant |b|$, then α and β can be found in O(log $|b|$) time. It was proved in [3] that if a and b are integers, not both zero, then there exist integers x and y such that $xa + yb = \gcd(a, b)$ and $\gcd(y, \gcd(a, b)) = 1$. In this paper, we showed that if $|a| \geqslant |b|$, then x and y can be found in O(log $|b|$) time. Based on this result, we improve the CCH algorithm to an O(log N)-time algorithm.

References

[1] F. Aguiló, M.A. Fiol, An efficient algorithm to find optimal double loop networks, Discrete Math. 138 (1995) 15–29.
[2] J.-C. Bermond, F. Comellas, D.F. Hsu, Distributed loop computer networks: a survey, J. Parallel Distribut. Comput. 24 (1995) 2–10.
[3] R.C. Chan, C.Y. Chen, Z.X. Hong, A simple algorithm to find the steps of double-loop networks, Discrete Appl. Math. 121 (2002) 61–72.
[4] C.Y. Chen, F.K. Hwang, The minimum distance diagram of double-loop networks, IEEE Trans. Comput. 49 (2000) 977–979.
[5] Y. Cheng, F.K. Hwang, Diameters of weighted double loop networks, J. Algorithms 9 (1988) 401–410.
[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., The MIT Press, Cambridge, MA, pp. 856–862.
[7] P. Erdös, D.F. Hsu, Distributed loop networks with minimum transmission delay, Theoret. Comput. Sci. 100 (1992) 223–241.
[8] P. Esqué, F. Aguiló, M.A. Fiol, Double commutative-step diagraphs with minimum diameters, Discrete Math. 114 (1993) 147–157.
[9] M.A. Fiol, M. Valero, J.L.A. Yebra, I. Alegre, T. Lang, Optimization of double-loop structures for local networks, in: Proc. XIX Internat. Symp. MIMI'82, Paris, France, 1982, pp. 37–41.
[10] M.A. Fiol, J.L.A. Yebra, I. Alegre, M. Valero, A discrete optimization problem in local networks and data alignment, IEEE Trans. Comput. C-36 (1987) 702–713.
[11] F.K. Hwang, A survey on double-loop networks, in: F. Roberts, F.K. Hwang, C. Monma (Eds.), Reliability of Computer and Communication Networks, AMS Series, 1991, pp. 143–151.
[12] F.K. Hwang, A complementary survey on double-loop networks, Theoret. Comput. Sci. A 263 (2001) 211–229.
[13] F.K. Hwang, Y.H. Xu, Double loop networks with minimum delay, Discrete Math. 66 (1987) 109–118.
[14] M. Newman, Integral Matrices, Pure and Applied Mathematics Series, Vol. 45, Academic Press, New York, 1972.
[15] J.M. Peha, F.A. Tobagi, Analyzing the fault tolerance of double-loop networks, IEEE Trans. Network. 2 (1994) 363–373.
[16] C.K. Wong, D. Coppersmith, A combinatorial problem related to multimodule memory organizations, J. Assoc. Comput. Mach. 21 (1974) 392–402.