

# P2P ネットワークにおける一般化 Kautz ダイグラフに基づく分散ハッシュ表を用いた検索アルゴリズム

## On Lookup Algorithm with a Generalized Kautz-based Distributed Hash Table

岡下 綾<sup>▼</sup> 有次 正義<sup>◆</sup>

柴田 幸夫<sup>▲</sup>

Aya OKASHITA Masayoshi ARITSUGI  
Yukio SHIBATA

P2P (Peer-to-Peer) ネットワークではデータを動的なネットワーク上のノードに分散させるため、データを効率良く検索することが本質的に重要な課題となる。そこで、ネットワーク上の各ノードに経路情報として分散ハッシュ表 (DHT) を持たせることにより、位置に依存せず効率的にデータを検索するアプローチが注目を集めている。本稿では一般化 Kautz ダイグラフに基づく定数サイズの DHT を提案する。各ノードは他の二つのノードに関する情報を持つだけで、Koorde よりも拡張性の高いネットワークの上で、ノード数  $n$  に対してホップ数  $O(\log n)$  の検索を実現する。

In peer-to-peer (P2P) networks, it is important to efficiently locate nodes assigned to data items. This paper proposes lookup algorithm with a distributed hash table (DHT) based on generalized Kautz digraphs as a routing table. This is a family of constant-degree routing networks of logarithmic diameter. Generalized Kautz digraphs have an optimal diameter and higher scalability than de Bruijn digraphs. In our method, each node needs to store routing information about only 2 other nodes. Our algorithm guarantees  $O(\log n)$  hops, for the number of nodes  $n$ .

### 1. はじめに

Peer-to-Peer (P2P) ネットワークはノード同士が対等に通信を行う自律分散ネットワークである。P2P ネットワークは高価な中央サーバを必要とせず、容易に柔軟で耐故障性に優れたネットワークを構築できる。ノードは自由に参加/離脱を行い、ネットワークトポロジは動的に変化する。すべてのノードに対等の権限が与えられ、ネットワークに接続されているすべてのノードが持つデータにアクセスすることが可

能となる。通信は個々のノード間で直接行われるため、プライバシーの保護も容易である。

P2P ネットワークにおいて、各ノードは隣接するノードの情報のみを持ち、ネットワーク全体の情報を持つノードは存在しない。そのため、要求されたデータを持つノードを検索する問題は本質的に重要である。データを検索するためには、隣接するノードに問合せ (クエリ) をリレーで渡していく方法を採用する。

近年、ネットワーク上の各ノードに経路情報として分散ハッシュ表 (DHT) を持たせることにより、位置に依存せず効率的にデータを検索するアプローチが注目を集めている [2]。DHT を用いる検索は、オーバーレイネットワーク上の検索となる。オーバーレイネットワークに様々なダイグラフのサブクラスを埋め込むことにより、ダイグラフのルーティングアルゴリズムを利用して、確実かつ効率的に検索を行うことができる。DHT の主な評価指標に、次数とホップ数がある。次数は各ノードに隣接するノードの数であり、ホップ数はデータを検索するために必要な仲介ノードの数の期待値である。次数とホップ数には一般にトレードオフの関係がある。

本稿では、一般化 Kautz ダイグラフ [4, 7] に基づく DHT を用いた分散検索アルゴリズムを提案する。これは Chord DHT プロトコル [3] の族に含まれ、類似の検索アルゴリズムとして Koorde [5] が存在する。Chord は ring と finger table に基づいた DHT を用いることにより、次数  $\log n$  に対して  $O(\log n)$  のホップ数を実現している。Koorde は Chord ring に de Bruijn ダイグラフ [6] を埋め込むことにより、各ノードはわずか二つの他のノードの情報を保持するだけで  $O(\log n)$  のホップ数を実現している。

de Bruijn ダイグラフ、Kautz ダイグラフ [9] は次数とノード数に対して最小の直径を持つダイグラフであり、Moore bound [8] に対して良い値を持つことが知られている。直径とは、最も遠いノード間の距離であり、直径が小さいほど性質の良いネットワークトポロジであるといえる。検索のホップ数はネットワークの直径の値に大きく依存する。

本稿で取り扱う一般化 Kautz ダイグラフは Kautz ダイグラフ [9] を代数的な構成法から一般化したダイグラフであり、Kautz ダイグラフの性質を失わないまま、任意のノード数に対応している。一般化 Kautz ダイグラフはノード数に対して最小かそれに近い直径を持つ。一般化 Kautz ダイグラフは Kautz ダイグラフを含んだ、より大きいダイグラフのクラスであることから、本稿で提案するアルゴリズムは Kautz ダイグラフに対しても適用できる。

### 2. 一般化 Kautz ダイグラフとルーティングアルゴリズム

#### 2.1 一般化 Kautz ダイグラフ

次数  $d$ 、ノード数  $n$  の一般化 Kautz ダイグラフ  $G_K(d, n)$  のノード集合を  $V$ 、弧集合を  $A$  で表す。一般化 Kautz ダイグラフは  $V = \{0, 1, \dots, n-1\}$ 、 $A = \{(x, y) \mid y \equiv -dx - i \pmod{n}, 1 \leq i \leq d\}$  として定義される [4, 7]。直径  $D$  は  $\lceil \log_d n \rceil$  以上  $\lfloor \log_d n \rfloor + 1$  以下であり、ノード数に対して最小かほぼ最小となる [8]。一般化 Kautz ダイグラフは  $n = d^D + d^{D-1}$  ならば Kautz ダイグラフと同型であり、 $n = d^D$  ならば一般化 de Bruijn ダイグラフや de Bruijn ダイグラフと同型である。一般化 Kautz ダイグラフは Kautz ダイグラフを含み、一般化 de Bruijn ダイグラフと共通集合を持つダイグラフのサブクラスである。図 1 に次数 2,

▼ 学生会員 群馬大学大学院工学研究科博士後期課程  
[aya@msc.cs.gunma-u.ac.jp](mailto:aya@msc.cs.gunma-u.ac.jp)

◆ 正会員 群馬大学工学部情報工学科  
[aritsugi@cs.gunma-u.ac.jp](mailto:aritsugi@cs.gunma-u.ac.jp)

▲ 非会員 群馬大学工学部情報工学科  
[shibata@msc.cs.gunma-u.ac.jp](mailto:shibata@msc.cs.gunma-u.ac.jp)

ノード数12の一般化Kautzダイグラフ  $G_K(2,12)$  を示す。

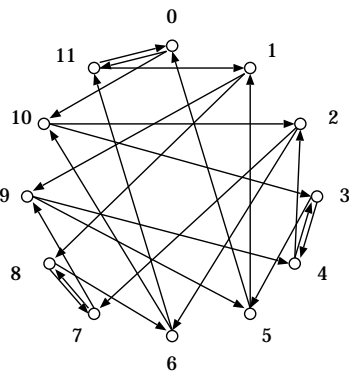


図1 一般化 Kautz ダイグラフ  $G_K(2,12)$

Fig.1 Generalized Kautz digraph  $G_K(2,12)$

## 2.2 一般化 Kautz ダイグラフにおけるルーティング

ダイグラフ  $G$  において、ノード列  $u_0, u_1, \dots, u_l$  に対して、弧  $(u_i, u_{i+1})$ ,  $0 \leq i \leq l-1$  が存在するならば、 $u_0, u_1, \dots, u_l$  を  $G$  における長さ  $l$  の  $(u_0, u_l)$  ウォーク、または単にウォークと呼ぶ。次数  $d$ 、ノード数  $n$  の一般化 Kautz ダイグラフ  $G_K(d, n)$  におけるノード  $x$  から  $y$  への経路は、長さが最小のウォークを特定することにより得られる。ノード  $x$  から  $y$  への長さ  $l$  のウォークを  $x = u_0, u_1, \dots, u_{l-1}, u_l = y$  とすると、合同式 (1) を得る。合同式はすべて  $n$  を法とする。

$$\begin{aligned} u_1 &\equiv -du_0 - r_0 \\ u_2 &\equiv -du_1 - r_1 \\ &\vdots \\ u_{l-1} &\equiv -du_{l-2} - r_{l-2} \\ u_l &\equiv -du_{l-1} - r_{l-1} \end{aligned} \quad (1)$$

ノード  $x$  から  $y$  への最短経路を求めることは、合同式 (1) を満足する最小の  $l$  の値と  $r_0, r_1, \dots, r_{l-1}$ ,  $r_i \in \{1, 2, \dots, d\}$ ,  $0 \leq i \leq l-1$  の値を求めることと等価である。

一般化 Kautz ダイグラフにおけるノード  $x$  からノード  $y$  への長さ  $l$  の経路が存在するための必要十分条件を補題 1 に示す。

### 補題 1

合同式 (1) が解を持つための必要十分条件は以下の条件を満たす整数  $t$  が存在することである。

1.  $l$  が奇数ならば、 $1 \leq t \leq d^l, t \equiv (-d)^l x - y \pmod{n}$
2.  $l$  が偶数ならば、 $-d^l + 1 \leq t \leq 0, t \equiv (-d)^l x - y \pmod{n}$  □

$1 \leq l \leq \log n$  に対して、 $t \equiv (-d)^l x - y \pmod{n}$  の値を求めることにより、条件を満たす最小の  $l$  の値を見つけることができる。得られた最小の  $l$  に対する  $r_i$ ,  $0 \leq i \leq l-1$ , は長さ  $l$  のウォークに一意に対応する。得られるウォークの長さは最短なので、同じノードを重複して通らない。

## 3. 分散ハッシュ表と検索アルゴリズム

### 3.1 分散ハッシュ表

ネットワーク上の各ノードとデータには SHA-1[1] などのハッシュ関数を用いて 0 から  $N-1$  までのハッシュ値を識別子としてそれぞれに割り当てる。ノードの識別子は端末の IP

アドレスやポート番号などから、データの識別子はファイル名やファイル情報などからハッシュ値として得る。ノードに割り当てられる識別子を NID, データに割り当てられる識別子を KID と表記する。NID には  $N$  を, KID には  $K$  を識別子の頭に付けることにより双方を区別する。  $N$  個の識別子で構成される論理的な識別子円に, 割り当てられている参加ノードの数を  $n$  とする。各識別子  $id$  に対して, 識別子円上で反時計回りに先に存在する最初のノードと, その NID を  $id$  の forward と呼び, forward( $id$ ) で表す。また,  $id$  に対して, 識別子円上で反時計回りに  $id$  自身が  $id$  より先に存在する最初のノードと, その NID を  $id$  の successor と呼び, successor( $id$ ) で表す。また  $id$  に対して, 識別子円上で反時計回りに  $id$  より手前に存在する最初のノードと, その NID を  $id$  の predecessor と呼び, predecessor( $id$ ) で表す。KID  $id$  を持つデータは, successor( $id$ ) を識別子として持つノードに割り当てられる。

本稿を通じて, 識別子に関する計算はすべて  $N$  を法とし, 識別子の取り得る範囲は 0 以上  $N-1$  以下とする。また,  $t = x \pmod{N}$  と表記された場合,  $0 \leq t < N$  と仮定する。

識別子空間において, 反時計回りの向きで連続する識別子の集合を区間と呼び, 閉区間表記  $[, ]$  と开区間表記  $(, )$  を用いて表す。識別子  $x$  から  $y$  までの区間は  $[x, y] = \{x, x+1, x+2, \dots, y-1, y\}$  となる。 $(x-1, y]$ ,  $[x, y+1)$ ,  $(x-1, y+1)$  も  $[x, y]$  と同じ区間を表す。区間  $[x, y]$  に含まれる識別子の数を  $|[x, y]|$  で表す。

すべての識別子に対してノードが割り当てられていれば検索経路は最短となるため, 検索に必要なホップ数は, 検索を実行したノードとデータを保持するノードの間の距離に等しくなる。しかし, ネットワークに多くのノードがいつでも参加できるように, またハッシュ値の衝突を防ぐためにも, 識別子円の状態は疎にしておかなければならない。よって  $N$  は十分大きい値を取り, ノードが割り当てられない多くの NID (仮想ノード) が存在する。

KID  $k$  を持つデータを検索するために, 一般化 Kautz ダイグラフの弧を辿ることにより successor( $k$ ) を発見する。識別子円に埋め込まれた一般化 Kautz ダイグラフは多くの仮想ノードを含むため, 理論上の最短経路に含まれた仮想ノードを辿ることはできない。そこで, 仮想ノードの NID  $x$  に対する predecessor( $x$ ) を辿ることで, ルーティングを行う。

forward ポインタのみを辿ることにより検索中のデータに確実に到達できるが, 最悪の場合にはすべてのノードを辿る必要がある。よって, forward ポインタで構成される識別子円上に次数 2 の一般化 Kautz ダイグラフの弧を埋め込むことにより,  $O(\log n)$  のホップ数を実現する。

本稿で提案する DHT では, 識別子  $x$  を持つノードに次の二つのノードへのポインタ (IP アドレスやポート番号などの情報) を持たせる。

1. forward( $x$ ),
2. predecessor( $-2x-1$ ) .

次数 2 の一般化 Kautz ダイグラフにおいて, ノード  $x$  は二つのノード  $-2x-1$  と  $-2x-2$  への弧を持つため, predecessor( $-2x-1$ ) と predecessor( $-2x-2$ ) の両方へのポインタが必要であるが, predecessor( $-2x-2$ ) = predecessor( $-2x-1$ ) である場合が多く, それ以外の場合には predecessor( $-2x-2$ ) =  $-2x-2$  であるので,

predecessor(-2x-1)のみを持たせる. predecessor(-2x-1)から forward ポインタを辿ることで predecessor(-2x-2)に到達できる. 図 2 に仮想ノードを含まない状態の, 一般化 Kautz ダイグラフに基づいた分散ハッシュ表を示す.

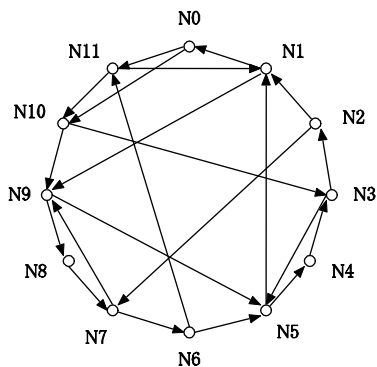


図 2 一般化 Kautz ダイグラフに基づく分散ハッシュ表

Fig. 2 A Generalized Kautz-based DHT

### 3.2 経路計算

関数 find\_route(x: NID, k: KID)は, 一般化 Kautz ダイグラフにおけるノード x から k までの仮想ノードの存在を考慮しない理論上の最短経路を計算し, ノード x が Lookup を送信するノードの識別子を返す. 図 3 に, 関数 find\_route を示す. ノード列  $u_0, u_1, \dots, u_l$  が KID k を検索するための理論上の経路であるとき, 各ノード  $u_i$  を turning point と呼ぶ.

```

Require: x : NID, k : KID
Ensure: 次の turning point の NID
for l=2 to D do
  if l is even then
     $t \leftarrow 2^l x - k + 2^l - 1 \pmod{N}$ 
    if  $0 \leq t \leq 2^l - 1$  then
       $t_0 \leftarrow t$  の最左ビット
       $r \leftarrow 2 - t_0$ 
      return  $(-2x - r \pmod{N})$ 
    end if
  else // l is odd
     $t \leftarrow -2^l x - k - 1 \pmod{N}$ 
    if  $0 \leq t \leq 2^l - 1$  then
       $t_0 \leftarrow t$  の最左ビット
       $r \leftarrow t_0 + 1$ 
      return  $(-2x - r \pmod{N})$ 
    end if
  end if
end for
    
```

図 3 関数 find\_route

Fig. 3 Function find\_route

### 3.3 検索アルゴリズム

図 4 に検索アルゴリズム Lookup を示す.

```

Require: k : KID, p : 次の turning point の NID
Ensure: successor(k) : NID
x : Lookup を実行するノードの NID
if  $k \in (x, forward]$  then
  return ( forward )
else if  $p == x$  then
  // x が turning point である場合
   $p \leftarrow find\_route(x, k)$ 
  return ( predecessor(-2x-1).Lookup(k, p) )
else if  $p \in (x, forward]$  then
   $p \leftarrow find\_route(p, k)$ 
  return ( forward.Lookup(k, p) )
else if predecessor(-2x-1)  $\notin (x, p]$  then
  // predecessor(-2x-1) が x よりも p から遠い場合
  return ( forward.Lookup(k, p) )
else
  return ( predecessor(-2x-1).Lookup(k, p) )
end if
    
```

図 4 アルゴリズム: Lookup

Fig. 4 Algorithm: Lookup

アルゴリズム Lookup(k: KID, p: NID)において, 引数 k は検索データの識別子であり, p は経路上の次の turning point の識別子を表す.

アルゴリズム Lookup に必要なホップ数に関する補題を以下に示す.

**補題 2:**

アルゴリズム Lookup に必要なホップ数の期待値は  $3\lceil \log N \rceil$  である.

**証明**

まず, データ検索の流れを説明する. 検索を開始するノード i が KID k を検索するとき, 最初に自分自身がデータを持っていないことを確認し, predecessor(-2i-1).Lookup(k, find\_route(i, k)) を実行する.

検索の過程で, Lookup(k, p)を渡されたノード x は次の処理を行う. ノード x の forward が successor(k)であれば, forward を返す. ノード x が turning point ならば, 次の turning point を計算して, predecessor(-2x-1)に Lookup(k, p)を渡す. turning point p が forward であるか, または  $x = predecessor(p)$  ならば, 次の turning point の NID を p として, forward に Lookup(k, p)を渡す.

以上の条件を満たさない場合は, 仮想ノード x からノード p へのルーティングのシミュレートに対応した処理を行う.  $s = predecessor(x)$ ,  $s' = forward(s)$ ,  $p = predecessor(-2x-2) = -2x-2 \pmod{N}$  とし, s から p に到達するために必要なホップ数について考察する. predecessor(-2s'-1)が区間(x, p]に存在しない場合, つまり, 仮想ノード x から s' までの距離がある程度大きい場合には, forward ポインタを辿って Lookup を渡すことにより p に到達する. そうでない場合には, predecessor(-2x-1)へ Lookup(k, p)を渡す.

Lookup アルゴリズムに必要なホップ数を求めるために, turning point に仮想ノードが含まれ, 最もホップ数が大きくなる場合を考察する. 図 5 に仮想ノード x からノード predecessor(-2x-2)へのルーティングをシミュレートする

実際の Lookup クエリの流れを示す。点線は理論上の経路であり、実線は実際に辿る経路を表す。

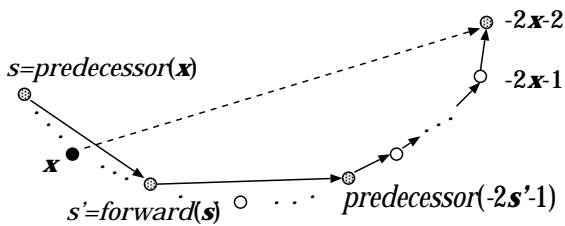


図5 仮想ノード x から predecessor(-2x-2)へのルーティング

Fig.5 Routing from Virtual Node x to predecessor(-2x-2)

図5において、 $s=predecessor(x)$ から $p=predecessor(-2x-2)$ に到達するために、 $s$ はまず $s'=forward(s)$ にLookup( $k, p$ )を渡す。次に、 $s'$ は $predecessor(-2s'-1)$ にLookup( $k, p$ )を渡す。一般化 Kautz ダイグラフの定義より、 $predecessor(-2s'-1)$ が区間 $(s', p]$ に存在するための必要十分条件は、 $|s', p| > 2|(x, s')|$ であることである。よって、 $predecessor(-2s'-1)$ から $p$ に到達するまでに forward ポインタを辿る回数 $|predecessor(-2s'-1), p|$ の値は高々 $u = (-2s'-1 - (-2x-2)) \pmod N = 2(x-s') + 1 \pmod N$ となる。N個の識別子に対してn個のノードが識別子円上にランダムに配置されているため、 $|(s, s')|$ の期待値は $N/n-1$ であり、

$$|(x, s')| \text{の期待値は } n \sum_{i=0}^{N/n-1} i / N = (N/n-1)/2 \text{ となる。}$$

$x-s' \pmod N$ の期待値は $|x, s'|+2$ であることから、 $u = 2(|x, s'|+2) + 1 = N/n + 1$ がいえ。よって、 $|predecessor(-2s'-1), predecessor(-2x-2)|$ の期待値は $un/N = (N/n+1)n/N \cong 1$ となる。故に、仮想ノード x からノード  $predecessor(-2x-2)$ に到達するために必要なホップ数の期待値は、ノード  $s=predecessor(x)$ から $s'=forward(s)$ へ1回、ノード  $s'$ から $predecessor(-2s'-1)$ へ1回、 $predecessor(-2s'-1)$ から $predecessor(-2x-2)$ へのホップ数の期待値は1回なので3となる。

$|s', p| \leq 2|(x, s')|$ である場合には、 $s'$ は $p$ に到達するために、forward ポインタのみを辿る。このとき forward ポインタを辿る回数は区間 $(s', p]$ に存在するノードの数である。 $|s', p| \leq 2|(x, s')|$ かつ $|x, s'|$ の期待値が $(N/n-1)/2$ であることから、区間 $(s', p]$ に存在する識別子の数の期待値は $N/n+1$ となる。よって、forward ポインタを辿る回数の期待値は $(N/n+1)n/N = 1+n/N \cong 1$ となる (Nは十分大きい値を取る)。故に、ノード  $s$ からノード  $p$ に到達するために必要なホップ数の期待値はノード  $s=predecessor(x)$ から $s'=forward(s)$ へ1回、 $s'$ から $p$ までのホップ数の期待値は1回なので2となる。

よって、アルゴリズム Lookup に必要なホップ数の期待値は $3\lceil \log N \rceil$ となる。□

補題2より、次の系が得られる。

系1:

アルゴリズム Lookup に必要なホップ数は $O(\log N)$ である。

□

アルゴリズム Lookup は $O(\log N)$ のホップ数を必要とするが、Nは非常に大きい数であることが想定される。次に、

$O(\log n)$ のホップ数でデータを検索するための手法について考察する。通常、Nの値は非常に大きい値を取り、仮想ノードの数も多くなることから、仮想ノード x からある turning point p への移動中に、後で辿るべき別の turning point p' に対する predecessor(p')に到達してしまう場合も少なくない。ここでは証明は省略するが、そのような場合には、pではなくp'からの経路を再計算して辿ることにより、ホップ数を $O(\log N)$ から $O(\log n)$ にまで削減できる。

4. まとめと今後の課題

本稿では、一般化 Kautz ダイグラフに基づく定数サイズの分散ハッシュ表と、ノード数 n に対して $O(\log n)$ のホップ数でデータを検索するアルゴリズムを提案した。今後の課題としては、分散ハッシュ表のサイズに関する一般化を行い、耐故障性について考察することなどが挙げられる。

【文献】

- [1] FIPS Pub 180-2, "Secure Hash Standard (SHS)," U.S. DoC/NIST, Aug. 2002.
- [2] H.Balakrishnan, M.F.Kaashoek, D.Karger, R.Morris and I.Stoica, "Looking up data in P2P systems," CACM, Vol.46, No.2, pp.43-48, Feb. 2003.
- [3] I.Stoica, R.Morris, D.Liben-Nowell, D.R.Karger, M.F.Kaashoek, F.Dabek and H.Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet Applications," IEEE/ACM Trans. Networking, Vol.11, No.1, pp.17-32, Feb. 2003.
- [4] M.Imase and M.Itoh, "A design for directed graphs with minimum diameter," IEEE Trans. Computers, Vol.32, No.8, pp.782-784, Aug. 1983.
- [5] M.F.Kaashoek and D.R.Karger, "Koorde: a simple degree-optimal distributed hash table," In Proc. of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS), Mar. 2003.
- [6] N.de Bruijn, "A combinatorial problem," Proc. Kon. Nederl. Akad. Wetensch, Ser. A, 49, pp.758-764, 1946.
- [7] S.Reddy, D.Pradhan and J.Kurl, "Direct graphs with minimum diameter and maximal connectivity," School of Engineering, Oakland Univ. Tech. rep., Jul. 1980.
- [8] W.Bridges and S.Toueg, "On the impossibility of directed Moore graphs," J. Combinatorial Theory, Ser. B, 29, pp.339-341, 1980.
- [9] W.Kautz, "Bounds on directed (d,k)graphs," Theory of cellular logic networks and machines, SRI Project 7258, AFCRL 68-0668 Final Report, pp.20-28, 1968.

岡下 綾 Aya OKASHITA

群馬大学大学院工学研究科博士後期課程在学中。システムレベルの故障診断に関する研究に従事。電子情報通信学会学生会員。日本データベース学会学生会員。

有次 正義 Masayoshi ARITSUGI

群馬大学工学部情報工学科助教授。1991九州大学工学部情報工学科卒。1996同大学院博士後期課程了。博士(工学)。データベースシステム、分散並列データ処理等に興味を持つ。

柴田 幸夫 Yukio SHIBATA

群馬大学工学部教授。グラフ理論、特に de Bruijn graph, グラフの積、システムの故障診断等の研究に従事。電子情報通信学会、情報処理学会、IEEE、AMS 会員。