社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－MPS－56 (11)
2005／9／22

パンケーキグラフの直径を求める耐障害性のある並列計算システム

浅井章吾† 鴻池祐輔†
品野勇治† 金子敬一†

n パンケーキグラフは，n 種類の記号で作られる順列をそれぞれ頂点とし，順列の前方反転によって移ることが可能な順列間を辺で結んだグラフである．n パンケーキグラフは $n!$ 個の頂点を持つので，頂点数に対する多項式時間のアルゴリズムでは，直径を求めるのが困難な問題として知られている．これまでに，基本的な計算方法は提案されているが，実装面では $n = 15$ の規模の計算が限界であった．さらに大規模なパンケーキグラフの直径計算を行うためには，十分なスケーラビリティを持つ並列システムの構築が不可欠である．本研究では，大規模なパンケーキグラフの直径計算が行える耐障害性のある並列システムを開発し，遊休計算機を利用して実際に $n = 16$ の直径を求めた．

A Fault-Tolerant Parallel System for Computing the Diameters of Pancake Graphs

SHOGO ASAI †, YUUSUKE KOUNOIKE †, YUJI SHINANO † and KEIICHI KANEKO †

An n-pancake graph is a graph whose vertices are the permutations of n symbols. Two vertices are connected by an edge if and only if the corresponding permutations can be transitive by the prefix reverses. Since an n-pancake graph has $n!$ vertices, it is known to be a hard problem to compute its diameter by using an algorithm with the polynomial order of the number of vertices. A fundamental approach of the diameter computation has been proposed. However, the computation of the diameter of an n-pancake graph with $n = 15$ is the limit in practice. Hence, it is indispensable to establish a parallel system with enough scalability and fault-tolerance to compute the diameters of the larger pancake graphs. Therefore, in this study, we have developed a such system, and computed the diameter of the 16-pancake graph by using idling computers.

1. はじめに

パンケーキグラフ (*pancake graph*) は，1 から n までの n 種類の記号で作られる順列をそれぞれ頂点とし，順列の前方反転によって移ることが可能な順列の間を辺で結んだグラフである．n によって異なるグラフが作れることから，***n* パンケーキグラフ (*n-pancake graph*)** と呼ぶ．パンケーキグラフは，対称性，再帰性，次数と直径に対する頂点数の多さといった利点を多く持つことから，並列計算機システムにおける相互結合網のモデルとして注目されている [1)~3)]．パンケーキグラフを相互結合網のモデルとして利用することを考えたとき，グラフの直径は通信遅延を示す尺度の一つとなる [5),7)]．

n パンケーキグラフの直径を求めるには，ある頂点から全頂点への最短距離を求めればよいが，頂点数や辺数に依存するアルゴリズムでは，計算時間と記憶容量が指数的に増加するため，すぐに現実的には解けなくなる．そこで，鴻池ら [8)] はパンケーキグラフの再帰性に注目して，最短距離を求める必要がある頂点だけについて調べる手法を提案し，それまで未知であった $n = 14, 15$ のパンケーキグラフの直径を与えた．

鴻池らが $n = 15$ のパンケーキグラフの直径を求めた際にも，手動による並列計算を行った．しかし，さらに大規模なパンケーキグラフの直径を求めるためには，並列システムとして，多くの計算機を並列実行することが必要であった．また，計算のための並列システムは，計算終了時間の予測できない長時間の計算に耐える，耐障害性のあるシステムである必要があった．本研究では，プラットフォームとして，Condor[4)] と MW(Master-Worker)[6)] フレームワークを利用することで，パンケーキグラフの直径計算のための耐障害性のある並列計算システムを開発した．さらに，開発したシステムを利用することで，これまでに求められていなかった $n = 16$ のパンケーキグラフの直径を求めた．

2. 用語と記号の定義

ここでは，説明に必要な用語と記号を定義する．*例*を用いた詳細については 8) を参照されたい．

† 東京農工大学工学部情報コミュニケーション工学科
Department of Computer, Information and Communication Sciences, Faculty of Engineering, Tokyo University of Agriculture and Technology

1からnまでのn種類の記号で作られる順列の全体を集合をS_nとする．最小のパンケーキを記号1，最大を記号nとして，一番上のパンケーキから順に対応する記号を並べた順列$\pi \in S_n$によりn枚のパンケーキの山を表す．整列された山に対応する順列$(1,2,...,n)$を$\boldsymbol{e}_n$とする．順列$\pi \in S_n$の最初の$k(2 \leq k \leq n)$個の記号の順序を反転し，残りをそのままの順序とした順列を$\sigma \in S_n$とすると，順列πから順列σへの変換を**順列πに対するk個の前方反転**という．また，$\pi^k = \sigma$と表すこととする．本稿では順列の前方反転だけを扱うので，単に順列をk個反転すると書いたときも順列をk個前方反転することを意味する．順列πをx_1個反転した結果をさらにx_2個反転した結果に対応する$(\pi^{x_1})^{x_2}$を$\pi^{(x_1,x_2)}$のように表すこととする．さらに$\boldsymbol{x} = (x_1, x_2, ..., x_m)$とすると，$\pi^{\boldsymbol{x}}$によって連続する$x_1, x_2, ..., x_m$個の反転を表すこととする．$\pi^{\boldsymbol{x}} = \boldsymbol{e}_n$となるとき，$\boldsymbol{x}$を$\pi$を整列する手順と呼ぶ．

与えられた順列$\pi \in S_n$を整列する手順の最小の反転回数を順列πを引数とする関数$f(\pi) = \min\{|\boldsymbol{x}| : \pi^{\boldsymbol{x}} = \boldsymbol{e}_n\}$で表すこととする．$n$枚のパンケーキの山を整列するのに必要な反転回数の最大を，nを引数とする関数$f(n) = \max\{f(\pi) : \pi \in S_n\}$で表す．慣習的に同じ関数$f$が使われるが，引数によって意味が異なるので注意が必要である．

順列$\pi \in S_n$をそれぞれ頂点πとし，$\sigma = \pi^k$となる頂点σと頂点πの間を辺で結んだグラフを**パンケーキグラフ**と呼ぶ．nによって異なるグラフが作れることから，それぞれのパンケーキグラフのことは**nパンケーキグラフ**と呼び，P_nで表す．

P_nの頂点で割り当てられた順列の最後の記号がjである頂点と，その頂点間を結ぶP_nの辺を抜き出すことで作られる部分グラフをP_n^jのことを**部分パンケーキグラフ**と呼ぶ．各部分パンケーキグラフの頂点は重複しないから，P_nはn個の部分パンケーキグラフと，部分パンケーキグラフをまたぐ辺からなる．この性質をパンケーキグラフの**再帰性**と呼ぶ．

1からnまでのn種類の記号に対して一つの置換を決め，nパンケーキグラフの各頂点に割り当てられた順列をこの置換により書き換えたグラフは，元のパンケーキグラフと同型である．このことをパンケーキグラフの**対称性**という．

一般にグラフにおいて，ある頂点から別の頂点までの経路のうち，通過する辺の数が最も少ない経路を2頂点間の**最短経路**といい，その経路における辺の数を**最短距離**という．任意の2頂点間の組合わせのうち，もっとも長い最短距離をグラフの**直径**という．

最短距離を調べる頂点の一方に$\boldsymbol{e}_n$を選ぶことで，nパンケーキグラフの直径を求めることと，$f(n)$を求めることは等価となる．本稿では，ある頂点$\pi \in S_n$と頂点$\boldsymbol{e}_n$との最短距離のことを単にπの**距離**と表すこととする．

3. 直径を求める方法

3.1 部分計算の依存関係

まず，順列$\pi = \boldsymbol{e}_{n-1}^{\boldsymbol{x}} \in S_{n-1}$について順列$\sigma_k \in S_n(1 \leq k \leq n)$を式(1)と定義する．このとき$f(\sigma_k)$について式(2)が成り立つ．

$$\sigma_k = \begin{cases} ((\boldsymbol{e}_n)^n)^{\boldsymbol{x}} & k = 1 \\ ((\boldsymbol{e}_n)^{(k,n)})^{\boldsymbol{x}} & 2 \leq k \leq n-1 \\ \boldsymbol{e}_n^{\boldsymbol{x}} & k = n \end{cases} \quad (1)$$

$$f(\sigma_k) \leq \begin{cases} f(\pi)+1 & k = 1 \\ f(\pi)+2 & 2 \leq k \leq n-1 \\ f(\pi) & k = n \end{cases} \quad (2)$$

$\pi \in S_{n-1}$に対し，$\sigma_k \in S_n$を求める変換を$T_k(\pi)$とし，集合$S \subseteq S_{n-1}$の要素すべてについて$T_k(\pi)$を求めた集合を$T_k(S)$とする．また，集合S_n^mを$f(\pi) = m$となる$\pi \in S_n$の集合とし，$k < 0$又は$k > f(n)$となるkについては，S_n^kは空集合とする．このとき集合$\overline{S}_n^m$を式(3)と定義する．

$$\begin{aligned} \overline{S}_n^m &= T_1(S_{n-1}^{m-1}) \\ &\cup T_2(S_{n-1}^{m-2}) \cup \cdots \cup T_{n-1}(S_{n-1}^{m-2}) \\ &\cup T_n(S_{n-1}^m) \end{aligned} \quad (3)$$

これは，S_nのうち上界値がmである頂点集合である．

このとき式(2)により$\pi \in \overline{S}_n^m$について$f(\pi) \leq m$が成り立つ．また，$\overline{S}_n^m$とS_n^m, S_nの間には以下の関係が成り立つ．

$$S_n = \bigcup_{k=0}^{f(n-1)+2} \overline{S}_n^k \quad (4)$$

$$S_n^m \subseteq \bigcup_{k=m}^{f(n-1)+2} \overline{S}_n^k \quad (5)$$

式(3)，式(4)，式(5)より，部分計算の依存関係は**図1**のようになる．図1では，集合間の位置関係により，ある集合は，その真上及び左上の集合すべてに依存していることを表している．また，図1の左下及び右上の空白部分は空集合であることを表している．ある集合の下側が空集合かどうかは，直径がわかって初

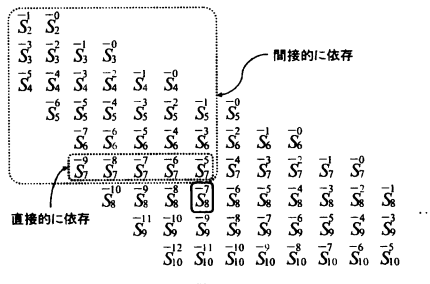

図1 $\overline{S}_n^m$間の依存関係
Fig. 1 The dependency relation between $\overline{S}_n^m$

めて判断できる．

このような関係によって，$S_1 = S_1^0 = \{e_1\}$ から，変換と距離計算を再帰的に繰り返す事で任意の $\overline{S}_n^m$ を導く事ができる．このとき，導き出したい $\overline{S}_n^m$ と無関係な頂点を無視することができる．たとえば，$\overline{S}_7^9$ を導きたい時は，$f(7) \geq f(6) = 7$ であることを使って，図1の $\overline{S}_7^7$ より左側の列についてだけ距離を求めればよい．

ある頂点 π と，その上界値 u がわかっているとき，実際にその頂点を無視するべきかを判断するためには，図1のような依存関係の図において，その頂点がどの列に属しているか知る必要がある．そこで，$n = |\pi|$ と u から次の式によって得られる数を，依存関係の図上での列番号とする．

$$column = n \times 2 - u \qquad (6)$$

上式の u の部分に暫定直径を入れて算出した列番号が，注目している頂点の列番号と等しいか小さい場合は，その頂点を無視することが出来る．

直径 $f(n)$ を求めるためには，まず全ての $\pi \in \overline{S}_n^{f(n-1)+2}$ について $f(\pi)$ を求める．この時，$f(\pi) = f(n-1)+2$ となるような π が有るか無いかによって $f(n)$ は以下のようになる．

- $f(\pi) = f(n-1)+2$ となる π が有る場合：$f(n) = f(n-1)+2$ であり，このような π が見つかった時点で計算を終了できる．
- そのような π が無い場合：$f(n) \leq f(n-1)+1$ であり，もし既に $f(\pi) = f(n-1)+1$ となるような π が見つかっていれば，$f(n) = f(n-1)+1$ として計算を終了できる．そうでなければ $\pi \in \overline{S}_n^{f(n-1)+1}$ のなかから $f(\pi) = f(n-1)+1$ となる π を探し，見つかれば $f(n) = f(n-1)+1$，見つからなければ $f(n) = f(n-1)$ である．

なお，個々の最短距離の求め方として，鴻池らと同じA*探索による最短経路探索を用いた．詳細については，8) を参照されたい．

3.2 MW フレームワークによるアルゴリズム

並列システムは MW フレームワークを用いて Master-Worker 方式で作成した．MW を用いることで，システムに耐障害性を持たせることが出来る．システムへの入力としては，n,$f(n-1)$, 計算中断時間,Master が確保する子問題数を与える．n は，直径を求めたいパンケーキグラフの大きさである．n より一つ小さいパンケーキグラフの直径である $f(n-1)$ を与えることにより，無駄な列挙を抑えることが出来る．計算中断時間とは，子問題の規模 (距離計算が必要な頂点数) のばらつきに対応するために，Worker での計算を中断するまでの時間である．Master のアルゴリズムを**図2**に，Worker のアルゴリズムを**図3**に示す．これらの図では，頂点 π とその距離の上界値 u の組合せを1つの子問題とし，$[\pi, u]$ と表現している．

3.3 実装上の仕様

3.2では，1つの頂点を $[\pi, u]$ と表現していた．しかし，実際には，$[\pi, f(\pi), k]$ という形で扱っている．計算に必要になった時点で，$[\pi, f(\pi), k]$ を $[\sigma_k, f(\pi)+\alpha]$ と置き換える．σ_k, α は式 (1), (2) より求める．この表現を用いると，k を変化させることで別の頂点を表すことができるので，メモリ使用量とメモリの確保・開放にかかる時間を減らすことができる．

また，図2および図3では，子問題を1つの頂点として扱っているが，実際には複数の頂点をまとめて扱っている．これによって，MasterPool 内の子問題数が極端に多くなるのを防いでいる．

4. 計算実験

作成したシステムを用いて，実際に16パンケーキグラフの直径を求めた．最大で，Pentium3 1GHz Dual, 1GB PC133 ECC SDRAM の計算機16台と Pentium2 400MHz 256MB SDRAM の計算機17台の計49CPU を用いて計算を行った．33日に及ぶ計算の結果，$\boldsymbol{f(16) = 18}$ であることがわかった．なお，33日の間 49CPU を全て使っていたわけではない．途中で一般ユーザのジョブにより計算を中断していた CPU もある．また，メンテナンスの都合により，Pentium2 のマシンは途中で全て止めている．

実際に18回の反転が必要であった長さ16の順列をいくつか示す．

$(1, 15, 9, 11, 8, 10, 12, 7, 13, 5, 2, 16, 4, 14, 6, 3)$
$(6, 10, 4, 14, 2, 13, 16, 12, 8, 11, 7, 9, 5, 1, 3, 15)$
$(13, 9, 15, 2, 6, 4, 7, 11, 8, 12, 10, 14, 1, 16, 5, 3)$

5. おわりに

本研究では，鴻池らが P_{15} の直径を求めるときに用いた手法を並列化させ，大規模なパンケーキグラフの直径計算が可能な，耐障害性のある並列計算システムを作成した．また，実際にこのシステムにより，遊休計算機を利用して16パンケーキグラフの直径を求めた．

本研究では，$n = 16$ までのパンケーキグラフの直径を求めたが，より大きな n についても直径を求めたいと考えている．これは，n が一つ増えることによる計算量の増加からすると，一見難しいようにも思える．しかし，PC の高性能化・低価格化を考えると，$n = 17$ までは，計算機環境を整えて求めることが十分可能だと思われる．それ以上の直径を求めるためには，やはりアルゴリズムの改良が必要であろう．

また，既知の直径は，$n = 3, 6, 11$ の時だけ $f(n) = f(n-1)+2$ となっている．$n > 11$ では，まだこのような n は見つかっていない．このような n が，どこで出てくるかということにも興味がある．

```
MasterPool := {[(1,2),0], [(2,1),1]}
while (|MasterPool| < Master が確保する子問題数) {
  [π,u] を MasterPool より取り出す（幅優先）
  for (k = 1; k ≤ |π|+1; ++k) {
      σ_k := T_k(π) を作成し，式 (2) により上界値を算出
      列番号を式 (6) によって算出し，必要と判断されたら MasterPool へ追加
  }
}
while ( ((|MasterPool| != 0) || (実行中の Worker がある))
      && (MW が空の Worker を検出した || MW が Worker からの戻りを検出した) ) {
  if (空の Worker を検出) {
    [π,u] を MasterPool より取り出す
    [π,u], を検出した Worker へ送信（初期化時のみ n, f(n−1), 計算中断時間も送信）
  }
  if (ある Worker からの戻りを検出) {
    検出した WorkerPool の全ての子問題を受信し MasterPool へ追加
    送られてきた暫定直径が，Master が持っているものより大きければ暫定直径を更新
    このとき，暫定直径が f(n−1)+2 であれば，直ちに終了
  }
  (Master は待ち状態となる)
}
暫定直径の値を直径として出力
```

図 2 Master のアルゴリズム
Fig. 2 Algorithm for the Master

```
while (MW により，Master からの受信を検出) {
  受信した子問題 [π,u] を WorkerPool へ追加
  while (WorkerPool が空でない && (子問題受信後の経過時間 < 計算中断時間)) {
    [π,u] を WorkerPool より取り出す（深さ優先）
    f(π) を計算し，暫定直径より大きければ，暫定直径を更新
    for (k = 1; k ≤ |π|+1; ++k) {
        σ_k := T_k(π) を作成し，式 (2) により上界値を算出
        列番号を式 (6) によって算出し，必要と判断されたら WorkerPool へ追加
    }
  }
  暫定直径と WorkerPool の子問題を全て Master へ送信
  (この Worker は実行待ち状態となる)
}
```

図 3 Worker のアルゴリズム
Fig. 3 Algorithm for the Worker

参 考 文 献

1) Akl, S. G. and Qiu, K.: Fundamental Algorithms for the Star and Pancake Interconnection Networks with Applications to Computational Geometry, *Networks*, Vol.23, pp.215–225 (1993).
2) Bass, D. W. and Sudborough, I. H.: Pancake Problems with Restricted Prefix Reversals and Some Corresponding Cayley Networks, *Journal of Parallel and Distributed Computing*, Vol. 63, No. 3, pp. 327–336 (2003).
3) Berthomé, P., Ferreira, A. and Perennes, S.: Optimal Information Dissemination in Star and Pancake Networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol.7, No.12, pp. 1292–1300 (1996).
4) Condor Team: Condor Project Homepage. http://www.cs.wisc.edu/condor/.
5) Kumar, V., Grama, A., Gupta, A. and Karypis, G.: *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Benjaming/Cummings (1994).
6) MW project: MW Homepage. http://www.cs.wisc.edu/condor/mw/.
7) Quinn, M. J.: *Parallel Computing: Theory and Practice, second edition*, McGraw-Hill (1994).
8) 鴻池祐輔, 金子敬一, 品野勇治: n ≧ 14 パンケーキグラフの直径計算, 情報処理学会論文誌：数理モデルと応用, Vol. 46, No. SIG10(TOM12), pp. 48–56 (2005).