パンケーキグラフの直径計算

鴻池祐輔[†] 金子敬一[†] 品野勇治[†]

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，14 パンケーキグラフの直径でさえ求めることが困難な問題として知られている．実際に，これまでに求められている直径は $n = 13$ までである．本稿では，Heydari らが 13 パンケーキグラフの直径を求めるときに使った手法を基本として，不要な探索を行わないように発展させた手法を提案し，$n = 14$，15 のパンケーキグラフの直径を与えた．

Computing the Diameters of Pancake Graphs

Yuusuke Kounoike,[†] Keiichi Kaneko[†] and Yuji Shinano[†]

The *n-pancake graph* is a graph whose vertices are labeled by permutations on n symbols. There is an edge between two vertices when the label of one vertex is derived from the other by some prefix reversal. Finding the diameter of the n-pancake graph is known to be a very hard problem, because n-pancake graph has $n!$ vertices. Actually, $n = 13$ has been the maximum size of the n-pancake graph of which diameter was known so far. Heydari et al. found the diameter of the 13-pancake graph by using their own method. In this paper, we extend their method and give diameters of 14- and 15-pancake graphs that are previously unknown by using it.

1. はじめに

パンケーキ整列問題（*pancake sorting problem*）は，文献 7) で提案された以下のような問題である．

> 「あるレストランのシェフはいいかげんな性格で，彼の作るパンケーキは必ず全部がばらばらの大きさである．そのため，パンケーキの山をお客さんのテーブルまで持って行く前に，上に行くほど小さくなるように並べ替えなくてはならない．並べ替えは，一番上から何枚かをまとめて反転させることを，枚数を変えながら必要なだけ繰り返すことで行う．n 枚のパンケーキで作られる山のうち最悪の場合について，並べ替えるのに必要な反転の回数（n の関数として）は，どうなるであろうか？」

この問題は，順列の前方反転によって表されることから**前方反転問題（*prefix reversal problem*）**と呼ばれることもある．

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[1)]．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．n パンケーキグラフは，$n!$ 個の頂点を持つ，$n-1$ 次の正則グラフである．パンケーキ整列問題とパンケーキグラフの直径を求める問題は等価な問題である．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)~5),9),14)]．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 1 つとなる[13),15)]．

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．しかし，n パンケーキグラフは $n!$ 個の頂点を持つので，ダイクストラ法などの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こととなり，すぐに現実的には解けなくなってしまう．そこで，本研究で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

† 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

手法を提案する．提案する手法は Heydari ら[12] が 13 パンケーキグラフの直径を求めるときに使った手法を基本として，不要な探索を行わないように発展させたものである．

これまで 13 パンケーキグラフまでの直径については既知であったが，$n \geq 14$ のパンケーキグラフの直径は未知であった．本稿では，初めて $n = 14$，15 のパンケーキグラフの直径を与えた．

本稿は，次の構成で説明する．まず，2 章で問題を定式化する．その後 3 章で，既存の研究と本稿との関連を明確にするとともに，主たる結果の位置づけを示す．4 章では基本となっている Heydari らの手法を概観し，本稿での提案手法を示す．5 章では，実験の結果とその計算時間などを示す．最後に 6 章で結論と今後の展望について述べる．

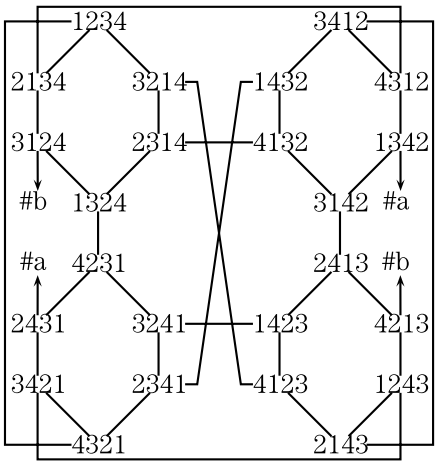

図 1 4 パンケーキグラフ
Fig. 1 4-pancake graph.

2. 問題の定式化

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 S_n とする．与えられた n 枚のパンケーキの山を，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 $\pi \in S_n$ で表す．整列された山に対応する順列 $(1, 2, \ldots, n)$ を $\boldsymbol{e}_n$ とする．本稿では $n = 4$ のときを例として用いるが，このときは順列の表し方として記号を続けて書くこととする．たとえば $\boldsymbol{e}_4$ は 1234 と表記する．

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．順列を k 個前方反転することが，パンケーキ整列問題でパンケーキの山を上から一番上から順に連続する k 枚のパンケーキの順序を反転することと対応する．本稿では順列の前方反転のみを扱うので，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 $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ とすると，$\pi^{\boldsymbol{x}}$ によって連続する $x_1, x_2, \ldots, x_m$ 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 π を整列する手順と呼ぶ．たとえば，$\pi = 2143$ とすると，この順列を 2 個前方反転すると $\pi^2 = 1243$ となる．$\boldsymbol{x} = (2, 4, 2, 4)$ とすると $\boldsymbol{x}$ は π を整列する手順である．

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．パンケーキ整列問題で求める値である，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．例として 4 パンケーキグラフ（P_4）を**図 1** に示す．図の中で#a，#b の記号はそれぞれ同じ記号につながる辺である．

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 とすると，P_n^j は P_{n-1} と同型である．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ことから，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．P_4 の例でいえば，P_4^4 は 1234，2134，3214，3124，2314，1324 の 6 個の頂点とその間を結ぶ 6 本の辺を持つ部分グラフである．図 1 では，P_4^4 は左上の六角形の部分である．

1 から n までの n 種類の記号に対して 1 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．P_4 の例で，1,2,3,4 をそれぞれ 3,4,1,2 と置換するとちょうど図 1 の左右を反転し

た結果になる．また，1,2,3,4 をそれぞれ 4,3,2,1 と置換すると上下を反転した結果になる．

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その 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 頂点の組合せのうち，最も長い最短距離をグラフの**直径**という．たとえば P_4 では 3421 と 4132 との間，1234 と 2413 との間などが最短距離 4 で最大であるので，P_4 の直径は 4 である．パンケーキグラフは対称性を持つので，任意の 2 頂点の組合せをすべて調べる必要はなく，ある頂点から他の頂点への最短距離の最大を調べれば十分である．先の例でいえば，1,2,3,4 をそれぞれ 4,3,1,2 とする置換を用いると，頂点 3421 は頂点 1234 に対応し，頂点 4132 は頂点 2413 に対応する．したがって，先ほどの 2 組の頂点対は最短距離が同じであることが明らかである．

$\boldsymbol{e}_n$ を割り当てた頂点を選ぶことで，n パンケーキグラフの直径を求めることと，パンケーキ問題における $f(n)$ を求めることは等価となる．P_4 の例でいえば，1234 からの最短距離が最大となるのは，4231，2413，3142 の 3 頂点であり，この 3 つの順列がパンケーキ整列問題の $n = 4$ のときの最悪の場合で，$f(n) = 4$ となる．

3. 既存の研究との関連

$f(n)$ の具体的な値については，Garey ら[8] が $n \leq 7$ について，Robbin[16] が $8 \leq n \leq 9$ について，Heydari ら[12] が $10 \leq n \leq 13$ について，これとは別に Cohen ら[6] が $10 \leq n \leq 12$ について，それぞれ求めている．これらの結果と本研究で求めた $14 \leq n \leq 15$ を合わせて，$f(n)$ の値を**表 1** に示す．

$f(n)$ の下界については Garey ら[8] が $n+1 \leq f(n)$ であることを示した．また，Gates ら[10] は長さ n が 16 の倍数である順列 χ_n を提案し，$f(\chi_n)$ の下界によって $(17/16)n \leq f(n)$ であることを示し，$(19/16)n \leq f(\chi_n)$ であろうと予想した．しかし，Heydari ら[12] が χ_n を $(9/8)n + 2$ 回で整列する手順を発見したことによって，この予想は誤りであることが分かっている．また，Heydari らは同時に長さ n が 14 の倍数である順列 φ_n を提案している．$(15/14)n \leq f(\varphi_n) \leq (8/7)n - 1$ であることから，$(15/14)n \leq f(n)$ であることを示した．さらに $f(\varphi_n) = (8/7)n - 1$ であろうと予想している．

$f(n)$ の上界については，Garey ら[8] が $f(n) \leq 2n-6$ であることを示し，Gates ら[10] と Györi ら[11] がほぼ同時期に，$f(n) \leq 5(n+1)/3$ であることを示した．Gates ら[10] は「焦げたパンケーキ問題（*burnt pancake problem*）」という，それぞれのパンケーキの片面が焦げていると考え，最終的にすべて焦げた面が下になるようにするという条件を追加した問題を提案している．焦げたパンケーキ問題において，Cohen ら[6] は，サイズは整列されているがすべて焦げた面が上になっている場合が最悪のケースなのではないかと予想し，$(23/14)n + c$（c は小さな定数）で焦げた面を下にする手順を示した．焦げたパンケーキ問題における必要な反転の回数の最大を $g(n)$ とすると，$f(n) \leq g(n)$ となることから，この予想が正しい場合，$f(n) \leq (23/14)n + c$ となる．さらに Heydari ら[12] が $3(n+1)/2$ 回の手順を考案したことにより，Cohen らの予想が正しい場合は，$f(n) \leq 3(n+1)/2$ ということになる．

表 1 $f(n)$ の値
Table 1 The concrete values of $f(n)$.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	**14**	**15**
$f(n)$	0	1	3	4	5	7	8	9	10	11	13	14	15	**16**	**17**

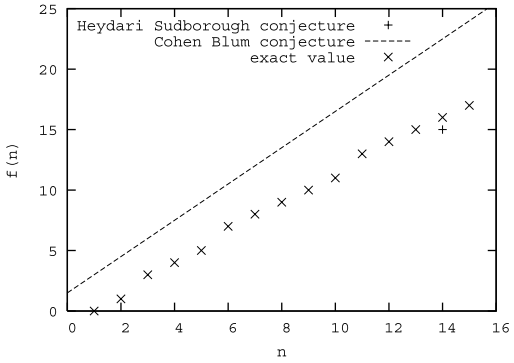

図 2 $f(n)$ の値と予想
Fig. 2 Exact and conjectured values of $f(n)$.

先に示した $f(14)$，$f(15)$ は，**図 2** に示すように，これらの予想を覆すものとはならなかった．

4. 直径計算の方法

4.1 Heydari らの手法

Heydari ら[12] は P_{12} の各頂点の最短距離を利用することで P_{13} のうち一部の頂点についてだけ最短距離を求めて $f(13)$ を得る手法を提案している．文献 12) では $f(13)$ を求める方法として説明しているが，ここでは一般的に $f(n)$ を求める手法として説明する．また，文献 12) では $f(n) \leq f(n-1)+2$ であることと，$\pi \in S_n \setminus S'_n$（$X \setminus Y = \{x : x \in X, x \notin Y\}$ とする）

表 2 $S_4'^k$ の要素
Table 2 The elements of $S_4'^k$.

k	$S_4'^k$ の要素
1	4132, 4312, 4231, 4321, 4123, 4213
2	3412, 1432, 2431, 3421, 2413, 1423
3	1342, 3142, 3241, 2341, 1243, 2143
4	1234, 2134, 3214, 3124, 2314, 1324

について $f(\pi) \leq f(n-1)+1$ であることについての説明がないが，本稿ではこれらの点を補足している．

全順列の集合 S_n を記号 n の位置によって，n 個に分割する部分集合 $S_n'^k$ を式 (1) で定義する．ただし，$\pi(k)$ は順列 π の k 番目の記号を表すこととする．また，例として $S_4'^k$ の要素を**表 2** に示す．

$$S_n'^k = \{\pi \in S_n : \pi(k) = n\} \quad (1 \leq k \leq n) \quad (1)$$

すべての順列 $\pi \in S_{n-1}$ について整列する手順を求めてあるとすると，$S_n'^n$ に含まれる順列は S_{n-1} と同じ手順で整列することができる．したがって，$\pi \in S_n'^n$ については，$f(\pi) \leq f(n-1)$ となる．また，$\pi \in S_n'^1$ については最初に n 個前方反転することで，$\pi \in S_n'^k$ $(2 \leq k \leq n-1)$ については k 個，n 個と続けて前方反転することによって，$S_n'^n$ に含まれる順列に移動することができる．したがって，それぞれ $f(\pi) \leq f(n-1)+1$，$f(\pi) \leq f(n-1)+2$ となる．また，$\pi \in S_n'^k$ $(2 \leq k \leq n-1)$ のうち，順列 $\pi^{(k,n)}$ の最後の記号 n を除いた順列 $\sigma \in S_{n-1}$ が $f(n-1)-1$ 回以下で整列できるような π については，$f(\pi) \leq f(n-1)+1$ である．

$\pi \in S_n'^k$ $(2 \leq k \leq n)$ のうち，$f(\sigma) = f(n-1)$ となるような順列の集合を S_n' とすると，$\pi \in S_n'$ のうち 1 つでも $f(\pi) = f(n-1)+2$ となれば，$f(n) = f(n-1)+2$ である．また，すべての $\pi \in S_n'$ について $f(\pi) \leq f(n-1)+1$ であれば，$f(n) \leq f(n-1)+1$ である．1 つでも $f(\pi) = f(n-1)+1$ となるような順列 π がある場合，$f(n) = f(n-1)+1$ となる．たとえば $n=4$ について考えると，$f(3)=3$ であり，$f(\pi)=3$ となる順列は 132 の 1 つだけである．したがって，$S_4' = \{2431, 3241\}$ となるが，どちらの順列も 3 回の反転で整列することができる．順列 4231 などが 4 回の反転を必要とすることから $f(4)=4$ である．S_n' は，S_n 全体に比べて小さいので，S_n 全体について最短距離を求める方法より少ない計算で $f(n)$ を得ることができる．

この方法で $f(13)$ を求めるためには，P_{12} で最短距離が直径と等しい頂点の列挙が必要となるが，文献 12) ではその列挙について特に述べていないことから，すべての順列 $\pi \in S_{12}$ について $f(\pi)$ を計算しているものと思われる．

4.2 本研究での提案手法

4.2.1 基本アルゴリズム

Heydari らの手法では P_{n-1} において最短距離が直径と等しいような頂点を列挙しておく必要がある．しかし，そのような頂点を列挙することは，直径を求めることと同様に困難である．

本稿で提案する手法は，最短距離を求める頂点の数を抑えてこの列挙を行うことを目的としている．これにより，Heydari らの手法よりもさらに少ない計算で $f(n)$ を得ることができる．

Heydari らの手法では記号 n の位置によって S_n を分割したが，本手法では部分パンケーキグラフ P_n^k によって分割する．この分割と Heydari らの手法での分割は，本質的には同じであるが，パンケーキグラフの再帰性と対称性から直感的に分かりやすい．

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 (2) と定義する．このとき $f(\sigma_k)$ について式 (3) が成り立つ．

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

$$f(\sigma_k) \leq \begin{cases} f(\pi)+1 & k=1 \\ f(\pi)+2 & 2 \leq k \leq n-1 \\ f(\pi) & k=n \end{cases} \quad (3)$$

頂点 σ_k はそれぞれ部分パンケーキグラフ P_n^k に含まれる．また，パンケーキグラフの再帰性と対称性から，P_{n-1} における $\boldsymbol{e}_{n-1}$ と π の位置関係と，各部分パンケーキグラフにおける $\boldsymbol{e}_n$ と σ_n，$(\boldsymbol{e}_n)^n$ と σ_1，$(\boldsymbol{e}_n)^{(k,n)}$ と σ_k の位置関係が等しくなる．P_4 の例を**図 3** に示す．

$\pi \in S_{n-1}$ に対し，$\sigma_k \in S_n$ を求める変換を $T_k(\pi) = \sigma_k$ 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，$k < 0$ または $k > f(n)$ 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 (4) と定義する．例として，$n=4$ でのそれぞれの要素を**表 3** に示す．

$$\begin{aligned} \overline{S}_n^m = {} & T_1(S_{n-1}^{m-1}) \\ & \cup T_2(S_{n-1}^{m-2}) \cup \cdots \cup T_{n-1}(S_{n-1}^{m-2}) \\ & \cup T_n(S_{n-1}^m) \end{aligned} \quad (4)$$

このとき式 (3) 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

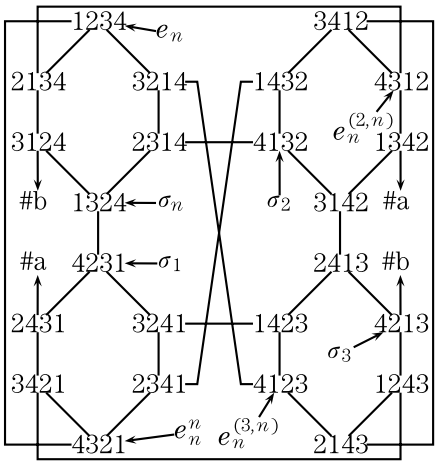

図 3 σ_k の位置
Fig. 3 The positions of σ_k.

表 3 $\overline{S}_4^m$ の要素
Table 3 The elements of $\overline{S}_4^m$.

m	$\overline{S}_4^m$ の要素
0	1234
1	2134, 3214, 4321
2	3124, 2314, 3421, 2341, 4312, 4123
3	1324, 2431, 3241, 3412, 1342, 1423, 2143
4	4231, 1432, 3142, 2413, 1243
5	4132, 4213

関係が成り立つ．

$$S_n = \bigcup_{k=0}^{f(n-1)+2} \overline{S}_n^k \tag{5}$$

$$S_n^m \subseteq \bigcup_{k=m}^{f(n-1)+2} \overline{S}_n^k \tag{6}$$

すべての $\pi \in \overline{S}_n^{f(n-1)+2}$ について $f(\pi)$ を求めた結果，$f(\pi) = f(n-1)+2$ となるような π があるかないかによって，$f(n)$，$S_n^{f(n)}$ は以下のようになる．

- $f(\pi) = f(n-1)+2$ となる π がある場合：$f(n) = f(n-1)+2$ であり，$S_n^{f(n)}$ はすべて列挙されている．
- そのような π がない場合：$f(n) \le f(n-1)+1$ であり，$f(\pi) = f(n-1)+1$ となる π がある場合は $f(n) = f(n-1)+1$ である．$S_n^{f(n)}$ を列挙するには，さらに $\overline{S}_n^{f(n-1)+1}$ をについて最短距離を求める必要がある．

後者の場合は，さらに $\overline{S}_n^{f(n-1)+1}$ を求める必要がある．さらに一般に $0 \le m \le f(n)$ とすると，S_n^m を完全に列挙するためには $\overline{S}_n^m$ が必要である．$\overline{S}_n^m$ は S_{n-1}^{m-2}，S_{n-1}^{m-1}，S_{n-1}^m から生成される．これらの集合は，再帰的に $S_{n-k}^{l \ge m-2k}$ を求めていくことで得ることができる．$S_1 = S_1^0 = \{\boldsymbol{e}_n\}$ のように事前に列挙しておくことが容易なほど $n-k$ が小さくなるか，S_{n-k}^l が空集合となる $l < 0$ になるまで再帰を繰り返す．

```
procedure make_problem(n, m)
{S_n^k から S̄_n^m を式 (4) により求める}
begin
  if n − 1 > c then begin
    make_problem(n − 1, m − 2);
    solve(n − 1, m − 2);
  end;
  S̄_n^m := 式 (4);
end

procedure solve(n, m)
{S̄_n^m を解き，S_n^m に追加していく}
begin
  foreach π in S̄_n^m do begin
    f(π) を求める
    S_n^{f(π)} := S_n^{f(π)} ∪ {π};
  end
end

procedure diameter(n)
{f(n) を求める}
begin
  make_problem(n, f(n − 1) + 2);
  solve(n, f(n − 1) + 2);
  if S_n^{f(n−1)+2} is not empty then
    f(n) = f(n − 1) + 2;
  else
    f(n) = f(n − 1) + 1;
end
```

図 4 $f(n)$ を求めるアルゴリズム
Fig. 4 Algorithm for computing $f(n)$.

図 4 にこのアルゴリズムを擬似コードで示す．ただし，S_c については事前に S_c^k $(k \le f(c))$ を求めておくこととし，diameter$(c+1)$，diameter$(c+2)$ と順番に呼び出すものとする．また，このアルゴリズムにおいて $c = 7$ としたときに S_n^m を求めていく順番を **表 4** に示す．

diameter 手続きを $c+1$ から昇順で呼び出すことで，solve(n, m) が呼ばれる時点では，$k > m$ について solve(n, k) をすでに呼び出していることになる．したがって，solve(n, m) 手続きが終了すると，式 (6) により S_n^m 全体が列挙されたことになる．

4.2.2 個々の最短距離の求め方

本研究では個々の最短距離の求め方として，A*探索による最短経路探索を用いた．A*探索の詳しい説

表 4 S_n^m を求める順番
Table 4 Computation order of S_n^m.

n	$f(n)$	S_n^m を求める順番
8	9	$S_7^8 \to S_8^{10} = \phi$
9	10	$S_7^7 \to S_8^9 \to S_9^{11} = \phi$
10	11	$S_7^6 \to S_8^8 \to S_9^{10}$ $\to S_{10}^{12} = \phi$
11	13	$S_7^5 \to S_8^7 \to S_9^9 \to S_{10}^{11}$ $\to S_{11}^{13} \neq \phi$
12	14	$S_{11}^{13} \to S_{12}^{15} = \phi$
13	15	$S_7^4 \to S_8^6 \to S_9^8 \to S_{10}^{10}$ $\to S_{11}^{12} \to S_{12}^{14} \to S_{13}^{16} = \phi$

明は文献 17) などがあるので，ここでは簡単に概略を示す．

A*探索では，始点から終点までの経路を求める．始点に隣接する頂点を列挙し，列挙した頂点のうち始点からの距離と終点までの距離の見積りの和が最も小さい頂点について，同じことを繰り返す．列挙された頂点に終点が見つかるとその経路を探索の結果とする．このとき，距離の見積りが実際の距離の下界を与える場合，A*探索により見つかる経路が最短であることが保証される．本研究では以下に示す $\tilde{f}(\pi)$ を距離の見積りとして使用した．

$\pi \in S_n$ を，$\pi = \pi_1\pi_2\cdots\pi_n$ とする．$\pi_{i+1} = \pi_i + 1$ または $\pi_{i+1} = \pi_i - 1$ のとき，π_i と π_{i+1} が連結している，と呼ぶ．また，$\pi_n = n$ のときも連結していることとする．π の中で連結の数を π の連結度とする．このとき，1 回の反転では連結度はたかだか 1 しか増えない．また，$\boldsymbol{e}_n$ の連結度は n であることから，$\tilde{f}(\pi)$ を n から π の連結度を引いたものとすると，$\tilde{f}(\pi)$ は $f(\pi)$ の下界を与える．$\tilde{f}(\pi)$ を距離の見積りとして使うことにより，A*探索で見つかる経路が最短経路であることが保証される．

4.2.3 実装と並列化

S_n^m と $\overline{S}_n^m$ は n が大きくなるにつれて，多くの要素を持つようになる．そこで本研究ではこれらの集合を，ファイルとして保存することにした．それぞれのファイルは 1 行に 1 つの順列をテキストとして持つ．さらにこのファイルを圧縮した状態で保存することとした．

計算を行うプログラムについては，図 4 の各手続きをそれぞれ別のプログラムとして作成した．solve 手続きに相当するプログラムは，$\overline{S}_n^m$ を保存したファイルの内容を標準入力として与えると，各要素の $f(\pi)$ の値によって適切なファイルに保存する．このプログラムによって得られた S_n^m を使い，make_problem 手続きに相当するプログラムが S_n^m を作成する．diameter 手続きに相当するプログラムが他の 2 つのプログラムを適切な順序で呼び出すことで，$f(n)$ を求める．

n が大きくなると，$\overline{S}_n^m$ は非常に多くの要素を持つようになるが，solve 手続きで行う最短経路の探索は，それぞれ完全に独立した問題である．そこで，本研究では $\overline{S}_n^m$ を適当に分割して，それぞれ別の計算機で計算するという単純な並列化を行った．並列化の基準として，$\overline{S}_n^m$ が 10 億以上の要素を持つ場合に 1,000 万個ずつに分割し，各計算機に適当に割り当てることとした．

A*探索の性質上，それぞれの最短経路問題を解くのに必要な計算時間は問題ごとに異なる．そのため，同じ 1,000 万個の最短経路問題であっても，計算が完了するまでの時間にはばらつきがある．動的に問題を割り当てることで，すべての計算機を効率良く使用することもできるが，本研究ではそこまでは考えず，計算を開始する前に静的に割り当てるものとした．

5. 計算実験の結果

本稿で提案した方法を実装し，P_{14}，P_{15} の直径を求めた結果を以下に示す．$f(14)$ までは Pentium 4 2.6 CGHz，1 GB Dual Channel DDR-SDRAM の計算機で計算した．その結果 $f(15)$ の計算には長時間かかることが予想されたため，4.2.3 項に示した方法で，Pentium3 1 GHz Dual，1 GB PC133 ECC SDRAM の計算機 16 台を加えた 17 台で計算した．

実験の結果得られた $f(14)$，$f(15)$ の値はすでに表 1 に示したとおりである．$f(n)$ の計算の過程で得られた S_n^m の要素数を**表 5** に示す．また，実際に 17 回の反転が必要であった長さ 15 の順列をいくつか示す．

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

これらの順列はそれぞれ次の手順により，17 回の前方反転で整列することができる．

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

今回の実験では事前に P_7 の全頂点について最短経路を求めておくこととした．$c = 7$ から始め，$f(8)$，$f(9)$ と順に $f(n)$ を求める各段階でかかった時間と，最短距離を求めた問題数を**表 6** に示す．問題数の比較対象として P_n の頂点数を最後の列に示した．

表 6 から，P_n の頂点数と比べてかなり少ない数の最短経路問題で，$f(n)$ が求められていることが分かる．また，n が大きくなるにつれて解いた最短経路

表 5 S_n^m の要素数
Table 5 The number of elements in S_n^m.

S_n^m	$\lvert S_n^m\rvert$	S_n^m	$\lvert S_n^m\rvert$	S_n^m	$\lvert S_n^m\rvert$
S_7^0	1	S_8^7	15,011	S_{11}^{10}	14,250,471
S_7^1	6	S_8^8	8,520	S_{11}^{11}	9,123,648
S_7^2	30	S_8^9	455	S_{11}^{12}	956,354
S_7^3	149	S_9^6	38,015	S_{11}^{13}	6
S_7^4	543	S_9^7	93,585	S_{12}^{12}	111,050,066
S_7^5	1,357	S_9^8	132,697	S_{12}^{13}	13,032,704
S_7^6	1,903	S_9^9	79,379	S_{12}^{14}	167
S_7^7	1,016	S_9^{10}	5,804	S_{13}^{14}	186,874,852
S_7^8	35	S_{10}^8	919,365	S_{13}^{15}	2,001
S_8^4	1,191	S_{10}^9	1,309,756	S_{14}^{16}	24,732
S_8^5	4,281	S_{10}^{10}	814,678		
S_8^6	10,561	S_{10}^{11}	73,232		

表 6 $f(n)$ の計算にかかった時間と問題数
Table 6 Computing time and the number of problems for $f(n)$.

n	計算時間	解いた問題数	P_n の頂点数
8	0.06 sec	210	40,320
9	1.0 sec	9,316	362,880
10	14.1 sec	117,996	3,628,800
11	3.5 min	1,425,037	39,916,800
12	0.07 sec	60	479,001,600
13	54 min	18,221,452	6,227,020,800
14	15.5 hour	249,271,566	87,178,291,200
*15	7 day	3,640,943,222	1,307,674,368,000

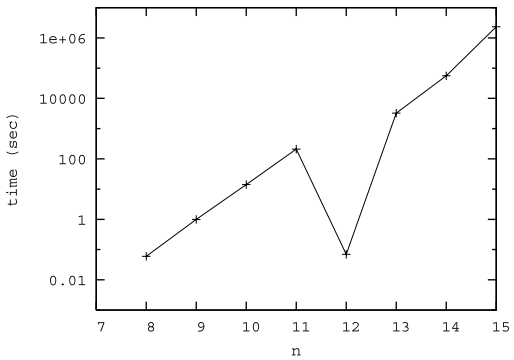

図 5 $f(n)$ の計算時間
Fig. 5 Computing time for $f(n)$.

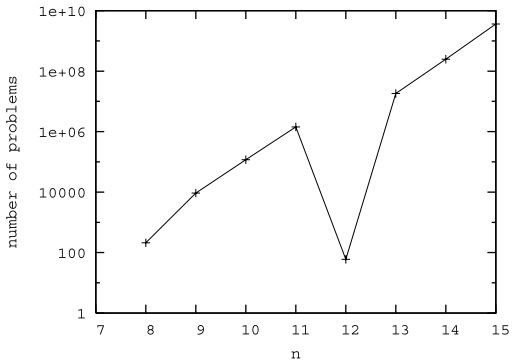

図 6 $f(n)$ の計算に必要な問題数
Fig. 6 The number of problems for $f(n)$.

問題の数が増え，それに合わせて計算時間も増えている．ただし，$n = 12$ ではわずか 60 個と少ない数の最短経路問題で直径を計算できている．これは，直前の $n = 11$ で $f(11) = f(10) + 2 = 13$ となり，$f(11)$ の計算が完了した時点で S_{11}^{13} の列挙が完了しているためである．

表 6 において，$n = 15$ については並列化した結果の計算時間であるので，それ以前の結果とは単純には比較できない．そこで $n = 14$ について並列でも計算し表 6 の逐次計算結果との比較を行った．4.2.3 項で述べた方法では並列化を行うのは図 4 の solve 手続きの一部だけであり，その他の部分では逐次計算を行っている．$n = 14$ について並列計算した場合，逐次に計算する部分が 6.5 時間，並列に計算する部分が 1.5 時間，合計 8 時間であった．全部を逐次で計算した場合の計算時間が 15.5 時間であることから，並列化した部分を逐次で計算すると 9 時間かかることが分かる．逐次で 9 時間かかる計算が並列化により 1.5 時間と 1/6 に短縮されている．

表 6 の $n = 15$ の計算時間のうち，逐次に計算する部分がおよそ 3 日，並列で計算する部分がおよそ 4 日であった．$n = 14$ の計算と同じ並列化効率と仮定すると，並列で計算する部分についても逐次で計算すると 24 日かかると予想される．つまり，逐次計算の部分と合わせて全体で 27 日の計算時間がかかるものと予想される．

n を横軸に，計算時間と問題数を縦軸に対数でとったものをそれぞれ**図 5** と**図 6** に示す．ただし，図 5 のうち $n = 15$ については先に示した予想計算時間を用いている．グラフから計算時間・問題数ともに指数的に増加していることが分かる．$n = 12$ が例外的であることは先に述べたとおりである．4.2.3 項で述べたように順列をファイルとして保存するため，問題数の増加にともない計算に必要なディスク容量も指数的に増加する．$f(14)$ の計算が完了した時点で 21 GB 必要であった．一方，必要な情報を順次ディスクへと書き込んでいく設計であるので，計算に必要なメモリは問題数には依存しない．ただし，n が大きくなると探索空間の拡大にともない個々の最短経路問題を解く A*探索に必要な記憶容量も増加することとなる．

6. おわりに

本稿では Heydari らが P_{13} の直径を求めるときに用いた方法を発展させて，$n!$ 個の頂点のうち一部の頂点について最短経路を求めるだけで P_n の直径を求める方法を提案した．また，この方法により P_{14} と P_{15} の直径を求めた．

本稿では $n = 15$ までのパンケーキグラフの直径を求めたが，さらなる改良により，より大きな n についても直径を求めたいと考えている．特に，Heydari らの手法と本稿で提案した手法では最短距離を求める必要のある頂点が異なることが非常に興味深い．

図 2 を見ても分かるように，$f(n)$ の下界に比べて上界は Cohen らの予想が正しかったときの $f(n) \leq 3(n+1)/2$ としてもまだ余裕がある．このことから，Cohen らの予想はおそらく正しいものであるが，さらに上界を改善する余地があるのではないかと考えられる．

また，3 章で紹介した χ_n，φ_n で大きな n について $f(\chi_n)$，$f(\varphi_n)$ を求めることにも興味がある．その結果によっては $f(n)$ の下界を更新できる可能性がある．さらに，本稿で提案した手法を焦げたパンケーキ整列問題に適用することで，焦げたパンケーキグラフの直径についても求めていきたいと考えている．

参考文献

1) Akers, S.B. and Krishnamurthy, B.: A Group-Theoretic Model for Symmetric Interconnection Networks, *IEEE Trans. Comput.*, Vol.38, pp.555–566 (1989).
2) 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).
3) Akl, S.G. and Qiu, K.: A Novel Routing Scheme on the Star and Pancake Interconnection Networks and its Applications, *Parallel Computing*, Vol.19, pp.95–101 (1993).
4) 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, pp.327–336 (2003).
5) Berthomé, P., Ferreira, A. and Perennes, S.: Optimal Information Dissemination in Star and Pancake Networks, *IEEE Trans. Parallel and Distributed Systems*, Vol.7, No.12, pp.1292–1300 (1996).
6) Cohen, D.S. and Blum, M.: On the problem of sorting burnt pancakes, *Discrete Appl. Math.*, Vol.61, No.2, pp.105–120 (1995).
7) Dweighter, H.: *Amer. Math. Monthly*, Vol.82, p.1010 (1975).
8) Garey, M.R., Johnson, D.S. and Lin, S.: *Amer. Math. Monthly*, Vol.84, p.296 (1977).
9) Garfgano, L., Vaccaro, U. and Vozella, A.: Fault Tolerant Routing in the Star and Pancake Interconnection Networks, *Information Processing Letters*, Vol.45, pp.315–320 (1993).
10) Gates, W.H. and Papadimitriou, C.H.: Bounds for sorting by prefix reversals., *Discrete Math.*, Vol.27, pp.47–57 (1979).
11) Györi, E. and Turán, G.: Stack of Pancakes, *Studia Sci. Math. Hungar.*, Vol.13, pp.133–137 (1978).
12) Heydari, M.H. and Sudborough, I.H.: On the diameter of the pancake network, *J.Algorithms*, Vol.25, No.1, pp.67–94 (1997).
13) Kumar, V., Grama, A., Gupta, A. and Karypis, G.: *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Benjaming/Cummings (1994).
14) Qiu, K., Meijer, H. and Akl, S.G.: Parallel Routing and Sorting on the Pancake Network, *Proc. International Conference on Computing and Information*, Lecture Notes in Computer Science, Vol.497, pp.360–371, Springer Verlag (1991).
15) Quinn, M.J.: *Parallel Computing: Theory and Practice*, 2nd edition, McGraw-Hill (1994).
16) Robbin, D.P.: Personal Communication (1977). cited in Ref. 10).
17) Russell, S. and Norvig, P.（著），古川康一（監訳）：エージェントアプローチ人工知能，共立出版 (1997).

(平成 16 年 4 月 19 日受付)
(平成 16 年 5 月 28 日再受付)
(平成 16 年 8 月 3 日再々受付)
(平成 16 年 8 月 12 日採録)

鴻池 祐輔（学生会員）

1978 年生．2003 年東京農工大学大学院工学研究科電子情報工学専攻博士前期課程修了．現在，同大学院工学研究科電子情報工学専攻博士後期課程在学中．主に整数計画問題の解法とその並列化に興味を持つ．オペレーションズ・リサーチ学会会員．

金子　敬一（正会員）

1962 年生．1985 年東京大学工学部計数工学科卒業．1987 年同大学大学院工学系研究科情報工学修士課程修了．同大学計数工学科助手，千葉大学情報工学科講師を経て，現在，東京農工大学情報コミュニケーション工学科助教授．博士（工学）．関数プログラミング，ディペンダブルコンピューティング，マルチメディア教育等の研究に従事．日本ソフトウェア科学会，電子情報通信学会，ACM 各会員．

品野　勇治（正会員）

1961 年生．1997 年東京理科大学大学院工学研究科経営工学専攻博士課程修了，博士（工学）．同年，東京理科大学助手．1999 年東京農工大学講師，2004 年同大学助教授（現職）．主に，数理計画法の理論と応用の研究，組合せ最適化問題に対する並列・分散アルゴリズムとその実装に関する研究に従事．オペレーションズ・リサーチ学会，IEEE，ACM 各会員．