17パンケーキグラフの直径計算

品野　勇治　金子　敬一

すべて大きさの異なる17枚のパンケーキに対する整列問題を解くことは，17パンケーキグラフの直径を求めることと同値である．17パンケーキグラフは，次数16の対称グラフである．しかしながら，頂点数が300兆を超えるため，頂点数や辺数に依存する単純なアルゴリズムによって直径を求めようとしても，計算時間と記憶容量の制限により，現実的にはまったく歯が立たない．ここでは，アルゴリズムの改良とともに，100台以上のPCとCondor-MWを用いた並列計算によって17パンケーキグラフの直径計算に挑んだ様子を紹介する．その結果については，本文を参照されたい．

キーワード：パンケーキ整列問題，Condor-MW，前方反転

1. はじめに

パンケーキ整列問題（*pancake sorting problem*）は，Jacob Goodmanによって提案された次のような問題である[3]．なお，文献[3]の著者名であるHarry Dweighterは，Hurried Waiterを洒落たペンネームである．

> 「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び替えなくてはならない．並び替えは，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，並び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までの整数からな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*）とよび，本稿ではP_nと表す．図1に4パンケーキグラフ(P_4)の例を示す．nパンケーキグラフは，$n!$個の頂点をもつ，$n-1$次の正則グラフである．グラフの直径は，各辺の距離を1とし，任意の2頂点間の最短路の距離（最短距離）の最大値で与えられる．パンケーキグラフにおいて，整列された記号列に対応する頂点と他のすべての頂点への最短距離の最大値が，パンケーキ整列問題の解となる．パンケーキグラフは対称性をもつので，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．

パンケーキ整列問題の解，つまりパンケーキグラフの直径の上下界値を与える論文がいくつか知られている[4]～[7]（その中でも特に有名なのは，マイクロソフト社ビル・ゲイツ会長の論文[5]である）．また，直径の示す数列そのものに対する関心ももたれており，*On-Line Encyclopedia of Integer Sequences*[10]には，$n=13$までの数列（数列名はSorting by prefix rever-

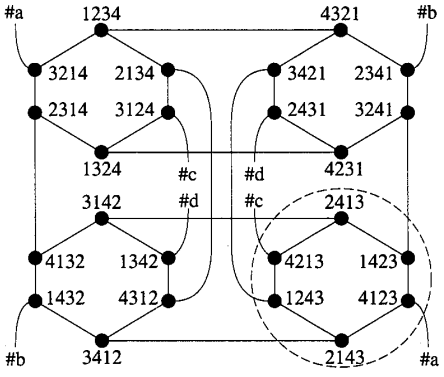

図1　4パンケーキグラフ(P_4)の例

しなの　ゆうじ，かねこ　けいいち
東京農工大学 大学院共生科学技術研究院
〒184-8588 小金井市中町2-24-16

sal) が示されている．しかし，$n=14$ 以上に関しては示されていない．

n パンケーキグラフの直径を求める際，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，Heydari ら[7]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，鴻池ら[8]は不要な探索を行わないように発展させた解法により，$n=14$，15 のパンケーキグラフの直径を与えた．この際，$n=15$ の直径の計算には，手動による並列計算を行ったが，直径を与える可能性のあるパンケーキの並びをすべてディスクへ保持するアルゴリズムであったため，$n=15$ での計算時に利用したディスク容量が，圧縮ファイルを使ったにもかかわらず，21 G バイトを超えた．そこで，浅井ら[9]はアルゴリズムを再設計するとともに，Condor-MW を用いたパンケーキの直径計算専用の並列プログラムを開発し，$n=16$，17 パンケーキグラフの直径を求めた．本稿では，この 17 パンケーキグラフの直径計算を紹介する．

2. パンケーキグラフの直径計算

2.1 用語

1 から n までの整数からなる順列全体を S_n で表す．最小のパンケーキを 1，最大のパンケーキを n とし，上に行くほど小さくなるように整列されたパンケーキの山を $\boldsymbol{e}_n=(1, 2, \cdots, n)$ で表す．n 枚のパンケーキの山に対する x 枚の反転は，その山に対応する順列 $\boldsymbol{\pi}\in S_n$ に対する x 個の前方反転 $\boldsymbol{\pi}^x$ で表すことができる．また，順列 $\boldsymbol{\pi}\in S_n$ に対する x_1 個の前方反転によって得られる順列 $\boldsymbol{\pi}^{x_1}$ に対して，さらに x_2 個を前方反転して得られる順列 $(\boldsymbol{\pi}^{x_1})^{x_2}$ を $\boldsymbol{\pi}^{(x_1, x_2)}$ と表す．さらに，$\boldsymbol{x}=(x_1, x_2, \cdots, x_m)$ とすると，順列 $\boldsymbol{\pi}\in S_n$ に対する，$x_1, x_2, \cdots, x_m$ 個の連続する反転を $\boldsymbol{\pi}^{\boldsymbol{x}}$ で表す．

任意の順列 $\boldsymbol{\pi}\in S_n$ に対して，$\boldsymbol{\pi}^{\boldsymbol{x}}=\boldsymbol{e}_n$ となる連続する反転 $\boldsymbol{x}$ を，$\boldsymbol{\pi}$ を整列する系列とよぶ．また，任意の順列 $\boldsymbol{\pi}\in S_n$ に対して，これを整列する系列のうち，その最小の反転回数を $\boldsymbol{f}(\boldsymbol{\pi})=\min\{|\boldsymbol{x}||\boldsymbol{\pi}^{\boldsymbol{x}}=\boldsymbol{e}_n\}$ と表す．さらに，S_n 中の順列を整列するのに必要な反転回数の最大値を $\boldsymbol{f}(n)=\max\{\boldsymbol{f}(\boldsymbol{\pi})|\boldsymbol{\pi}\in S_n\}$ で表す．慣習として，同じ $\boldsymbol{f}$ が使用されており，引数の型によって意味が変わる点に注意が必要である．

n パンケーキグラフ P_n の頂点である順列のうち，最後の整数が j であるような頂点から導出される部分グラフは，P_{n-1} と同型になる．このグラフを $P_n(j)$ と表し，部分パンケーキグラフと呼ぶ．例えば，図 1 の鎖線円内の部分グラフは，$P_4(3)$ となる．これらの部分パンケーキグラフは，互いに素であるため，P_n は，n 個の部分パンケーキグラフ $P_n(j)(j=1, 2, \cdots, n)$ とその間の辺からなる．すなわち，P_n は，再帰的に構成されている．また，P_n は，対称グラフであるから，頂点 $\boldsymbol{e}_n$ から他のすべての頂点への距離の最大値 $\boldsymbol{f}(n)$ を求めることと，P_n の直径を求めることは同値となる．

2.2 計算

まず，S_{n-1} 中の順列 $\boldsymbol{\pi}=\boldsymbol{e}_{n-1}^{x}$ に対して，S_n 中の順列 $\boldsymbol{\sigma}_k$ を以下のように定義する．

$$\boldsymbol{\sigma}_k=\begin{cases}((\boldsymbol{e}_n)^n)^x & k=1,\\ ((\boldsymbol{e}_n)^{(k,\,n)})^x & 2\le k\le n-1,\\ \boldsymbol{e}_n^x & k=n.\end{cases}$$

このとき，$\boldsymbol{f}(\boldsymbol{\sigma}_k)$ について，以下の関係が成り立つ．

$$\boldsymbol{f}(\boldsymbol{\sigma}_k)\le\begin{cases}\boldsymbol{f}(\boldsymbol{\pi})+1 & k=1,\\ \boldsymbol{f}(\boldsymbol{\pi})+2 & 2\le k\le n-1,\\ \boldsymbol{f}(\boldsymbol{\pi}) & k=n.\end{cases}$$

任意の $\boldsymbol{\pi}\in S_{n-1}$ に対して，$\boldsymbol{\sigma}_k\in S_n$ を求める変換を $T_k(\boldsymbol{\pi})$ と表す．このとき，先の関係の様子を図 2 に示す．さらに，$S\subseteq S_{n-1}$ に対して，$T_k(S)=\{T_k(\boldsymbol{\pi})|\boldsymbol{\pi}\in S\}$ と定義する．また，$S_n^m=\{\boldsymbol{\pi}|\boldsymbol{\pi}\in S_n, \boldsymbol{f}(\boldsymbol{\pi})=m\}$ とする．ただし，$m<0, m>\boldsymbol{f}(n)$ なる m に対しては，$S_n^m=\emptyset$ と定める．ここで，$\bar{S}_n^m=T_1(S_{n-1}^{m-1})\cup T_2(S_{n-1}^{m-2})\cup\cdots\cup T_{n-1}(S_{n-1}^{m-2})\cup T_n(S_{n-1}^{m})$ により $\bar{S}_n^m$ を定めると，この集合の要素となる順列 $\boldsymbol{\pi}$ は，$\boldsymbol{f}(\boldsymbol{\pi})\le m$ という性質を満たす．この $\bar{S}_n^m$ を**候補頂点集合** (candidate vertex set) とよぶ．

S_n, S_n^m，および $\bar{S}_n^m$ の間に以下の関係が成り立つ．

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

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

以上の関係をまとめると $\bar{S}_n^m$ は，$\bar{S}_{n+1}^0, \bar{S}_{n+1}^1, \cdots,$

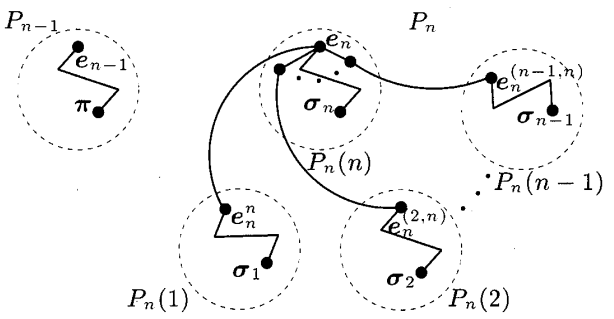

図 2 $\boldsymbol{f}(\boldsymbol{\pi})$ と $\boldsymbol{f}(\boldsymbol{\sigma}_k)$ との関係

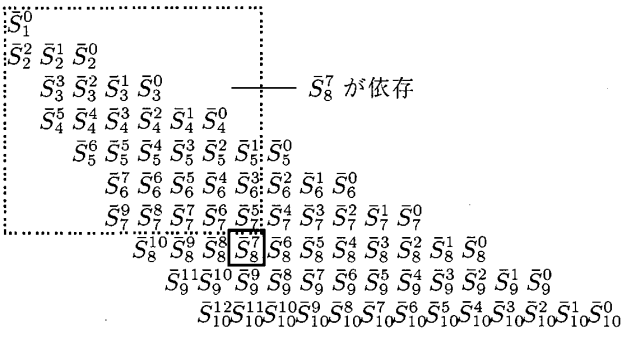

図3　$\bar{S}_n^m$ の依存関係

$\bar{S}_{n+1}^{m+2}$ に直接影響を与えている．間接的な影響まで考慮して，この依存関係を表現すると図3のようになる．すなわち，図3中のある集合は，その真上，および左上の集合すべてに依存している．例えば，$\bar{S}_8^7$ は，$\bar{S}_7^9$, $\bar{S}_7^8$, …, $\bar{S}_7^5$ に直接依存しており，$\bar{S}_1^0$, $\bar{S}_2^2$, …, $\bar{S}_6^3$ に間接的に依存している．さらに，図3の空白部分は，対応する集合が空であることを示している．ある集合 $\bar{S}_n^m$ の下に位置する集合が空か否かは，P_n の直径である $f(n)$ を計算した後，$f(n)<m$ であるか否かで判定することができる．以上の関係を用いて，$S_1=S_1^0=\{e_1\}$ から，変換と距離計算を再帰的に反復することで，任意の $\bar{S}_n^m$ を求める．

P_n の直径 $f(n)$ を求めるには，以下のような手順で計算を実行する．

1. まず，すべての $\pi\in\bar{S}_n^{f(n-1)+2}$ に対して，$f(\pi)$ を計算する．もし，$f(\pi)=f(n-1)+2$ を満たす π が存在するならば，ただちに $f(n)=f(n-1)+2$ となり，計算を終了する．
2. そのような π が存在しない場合，すでに $f(\pi)=f(n-1)+1$ を満たす $\pi\in\bar{S}_n^{f(n-1)+2}$ が見つかっていれば $f(n)=f(n-1)+1$ となり，計算を終了する．
3. そのような π がなかった場合は，すべての $\pi\in\bar{S}_n^{f(n-1)+1}$ に対して，$f(\pi)$ を計算する．もし，$f(\pi)=f(n-1)+1$ を満たす π が存在すれば，$f(n)=f(n-1)+1$ となり，存在しなければ $f(n)=f(n-1)$ となる．

個々の π に対して，$f(\pi)$ の計算にはA*アルゴリズムを用いている．A*アルゴリズムでは，距離の下界を用いて探索をすすめる．そのため，任意の順列 $\pi\in S_n$ に対して，Gatesらによって提案されているブロックという概念[5]を利用して，e_n までの距離の下界を計算している．

鴻池らによる実装[8]では，図3において，各列を左から右へと求めていく．それぞれの列では，上の集合に属する順列すべてに対する距離計算を済ませてから，その下の集合に対する計算を実行している．このような順序で計算を進めることで，集合間の依存関係から直径の計算に明らかに無関係な順列に対する距離計算を回避することができる．しかしながら，大きなパンケーキグラフに対しては，集合の要素数が爆発してしまい，二次記憶装置を利用した実装を必要とし，15パンケーキグラフの直径計算が限界となっていた．

一方，浅井らの実装[9]では，ある順列 π に対して距離を求めた後，直ちにその下の行に位置する順列のうち先の順列 π に対する変換で導かれるものの距離を確定する．鴻池らの実装が幅優先探索に類似しているのに対して，浅井らのものは深さ優先探索に類似する．深さ優先の場合は，直径計算に不要な順列まで探索してしまうという欠点がある．しかしながら，P_n の暫定直径 u を利用して $\bar{S}_n^u$ の上および右に位置する集合に属する順列は探索しないように工夫している．この方法は，$f(n)>u$ のときには，鴻池の手法では探索しないで済む順列に対しても距離計算を行ってしまうという冗長性をもつ．しかしながら，$f(n)=u$ となった後では，鴻池の手法と同一の順列だけに対する距離計算で済む．経験的に $f(n-1)+1$ となる暫定直径を与える順列は容易に見つかるため，上記の工夫により，不要な距離計算を大幅に抑えることに成功している．

3. Condor-MW を用いた並列計算

ここでは，Condor-MW を簡単に紹介し，Condor-MW を用いた直径計算の概要および，$f(17)$ の計算結果を示す．

3.1 Condor-MW

Condor は，計算時間を要するプログラムの実行と計算資源を管理するシステムである[2]．実行するプログラムはジョブとして管理され，ジョブには並列プログラムも含む．また，実行環境に対しては，動的に計算資源の追加・削除を許す．Condor は，High Throughput Computing (HTC) 環境の実現に向けて開発されている．High Performance Computing (HPC) 環境が短時間（1秒間等）に実行されるオペレーション数による評価での性能を求めるのに対して，HTC 環境は，長時間（月や年を単位とする計算）に実行可能なオペレーション数を増やすことを目標にし

ている．そのため，利用されていない遊休計算資源を効果的に利用するための種々の機能が実装されている．Condorプロジェクトは，Wisconsin大学で1988年から始まり，現在でも活発に開発が行われている．

MW（Master-Worker）は，Condorの動的に計算資源が変化する分散環境上で，Master-Worker型の並列プログラムを開発するためのツールである．MWでは，MWTaskクラスを継承して定義する部分問題クラスのインスタンスを1つの処理単位とする（本稿では，MWTaskクラスのインスタンスを単にMWTaskと記述する）．Masterは，MWTaskを管理し，空きのWorkerができれば順次MWTaskをWorkerへ割当てて処理する．計算機資源の管理はCondorが行い，空きの計算資源があれば，Workerの数を自動的に増やす．また，MWTaskを割当てた計算機に障害があると，自動的にWorkerの障害は検知され，実行されていたMWTaskは正常なWorkerへ自動的に移される．つまり，Workerに対する耐障害性はCondor-MWの機能として実現される．

3.2 Condor-MWによる直径計算の概要

nパンケーキグラフの直径計算に際しては，$f(n-1)$を暫定直径として与えて計算を開始する．Condor-MWを利用するに当たり，候補頂点集合の頂点1つを現すクラスをMWTaskクラスのサブクラスとして定義する．i番目のMWTaskを$(\bar{S}^*_*)_i$と表現する．Masterは，MWTask保存領域（Task Poolとよぶ）と暫定直径および，それを与えた頂点をもつ．また，MasterのTask Poolに保持されるMWTaskの数は，できる限り一定数（本稿ではこの数をpとする）を保持するように制御する．

初期化　Masterにおいて，候補頂点集合の生成規則に従って，幅優先探索により$(\bar{S}^*_*)_1, (\bar{S}^*_*)_2, \cdots, (\bar{S}^*_*)_p$のMWTaskを生成する（図4参照）．生成されたら，空きのWorkerがなくなるまで，各MWTaskと暫定直径をWorkerへ転送する（図5参照）．

Workerにおける通常処理　各Workerでは，受け取った頂点$(\bar{S}^*_*)_i$（MWTask）から候補頂点集合の生成規則に従って，深さ優先探索により$\bar{S}^*_*$の頂点が得られるまで生成し，その頂点と頂点$(1, 2, \cdots, n)$の最短距離をA^*アルゴリズムにより求め直径とする．$(\bar{S}^*_*)_i$から生成されるすべての候補頂点集合の頂点が調べ終わると，最長の距離を与えた頂点とその距離

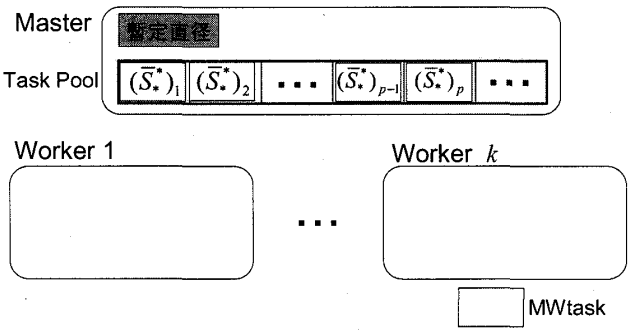

図4　Masterで初期MWTaskが生成された状態

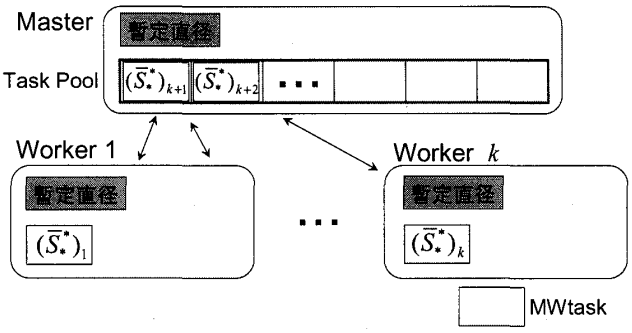

図5　すべてのWorkerへのMWTaskの割当

（直径）をMasterへ戻す．

Masterでの処理　あるWorkerへ送ったMWTaskにより定義される候補頂点から生成された直径を受け取ったら，Master内に保存している暫定直径と比較し，Workerからの直径が長い場合には，Master内の暫定直径とそれを与えた頂点を保存する．空きとなったWorkerへは，新たなMWTaskを割り当てる．Masterにはチェックポイント機構を実装している．つまり，Masterでは，実行時に指定された時間が経過するごとに，Task Pool中の計算待ちのMWTaskとWorkerによって計算中のMWTaskの情報をディスクへ保存する．これにより，Masterの動作している計算機に障害が生じた場合には，ディスクに保存された状態から計算が再開できる（Worker側の障害に対しては，Condor-MWが対処する）．

負荷分散　各MWTaskによって示される候補頂点から生成される候補頂点の数には大きな差がある．つまり，各MWTaskの計算負荷には大きな差がある．よって，これまでに述べた処理だけでは，MasterのTask Poolが空となり，一部のWorkerのみが計算している状況が生じる．そこで，Workerでの探索が，実行時に指定された最大探索時間を越えた場合には，候補頂点の列挙を中断し，それまでに得られている距

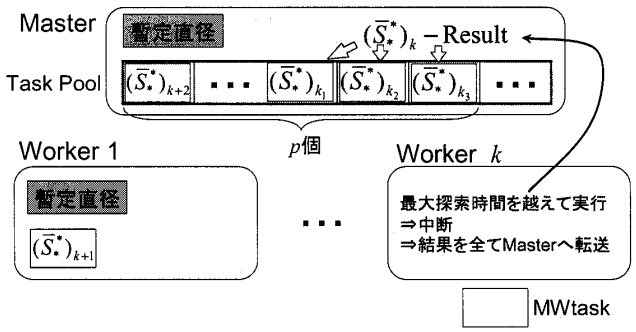

図6　Worker 処理の中断と Master での MWTask 生成

表1　PC クラスタの構成

Master/Worker	CPU	メモリ	PC 数	コネクション
Master	Opteron 1.8GHz dual	2GB	1	1000BASE-T
Worker	Opteron 1.8GHz dual	2GB	107	

離の最大値（暫定直径）およびそれを与えた頂点と，列挙中の候補頂点集合をすべて Master へ戻す．Master では，候補頂点集合が戻された場合には，戻された候補頂点数と実行待ち MWTask 数が p を下回る場合には，幅優先探索により MWTask の数を p まで増やす（図6）．このようにして，Master 中の Task Pool に保存されている MWTask 数を p 以上に保ち，Worker が空きとならないように並列化している．

3.3　$f(17)$ の計算結果

$f(17)$ の計算には，表1に示した PC クラスタを利用した．Master に維持する MWTask 数（p の値）は 1,024 とし，Master 側でのチェックポイント，および，Worker での計算を中断する時間は 10 分に設定した．

図7は，時間の経過とともに，動作させた Worker 数を示す．Worker 数が，若干上下している部分は，ネットワークスイッチの障害に起因している．Condor-MW の機能により，自動的に実行中の MWTask は，他の Worker へ転送され計算が継続していた．

計算途中で，Worker 数が倍程度に増えている．これは，当初の Worker 数では PC クラスタの利用期限内に計算が終了しないと予測し，利用可能な PC 台数を増やしたためである．この予測は，列挙すべき全頂点数の内，どれだけの頂点との間の最短路計算が終了したか（計算不要と判断された頂点も終了頂点として含めている）を，ログに出力しているので可能である．具体的には，最短路計算が必要となる残りの頂点数が

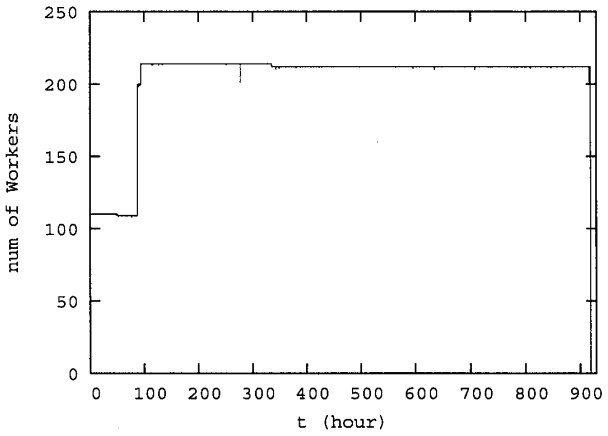

図7　Worker 数の遷移

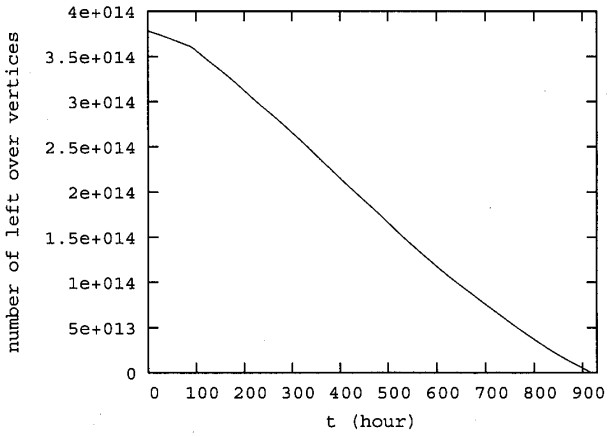

図8　残り頂点数の遷移

わかるので，それを図8に示すように時間軸にプロットし，その傾きを利用した．図8において，Worker を追加した前後の部分は，ほぼ直線と見なされるので，ある程度の精度で計算終了時刻が予測できることがわかる．

これまでに示したように環境を変化させながら，38日と7時間の計算を行った結果，$f(17)=19$ という結果を，浅井ら[9]は求めた．

4.　おわりに

本稿に示した結果を含めて，得られている n パンケーキグラフの直径を表2に示す．文献[10]の中で，Brain Goodwin 氏は，直径が2増える部分は，図9のような三角形の左端に，これまでのところ一致しているとコメントしている．しかしながら，University of Texas at Dallas の Ivan Hal Sudborough 教授との私的な情報交換によれば，このコメントが示す予測は正しくないとのことである．いずれにしても，次に直径が2増える部分を知りたいと思う．

本稿で紹介したプログラムでは，計算時間は比較的

表2　n-パンケーキグラフの直径

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
直径	0	1	3	4	5	7	8	9	10	11	13	14	15	16	17	18	19

0
1
3　4　5
7　8　9　10　11
13　14　15　16　17　18　19
21　22　23　24　25　26　27　28　29

図9　Brain Goodwin 氏のコメント

正確に予測できる．予測では，$f(18)$の計算には，$f(17)$の計算時間の20倍程度の計算時間が必要となる．本稿では，PCクラスタ環境を用いたが，Condor-MWは計算グリッドの環境として知られており，世界規模での動作環境が存在する．本稿で紹介したプログラムは，そのようなグリッド環境でも動作し，耐障害性，ウォームスタート（計算中断時点からの計算の再開）の機能は実現されている．つまり，計算資源さえあれば，$f(18)$の計算も可能な状況にはなっている．『多くの計算資源を使って，$f(18)$を求めて何の役に立つの？』という感想をおもちの方も多いとは思うが，遊休資源で延々と動かすことができるプログラムなので，電源が入っていて遊んでいるような計算機が存在するCondor環境があるなら，このような計算への利用も有効だと思う．このような計算に利用させていただけそうなCondor環境をご存知の方は，是非ご連絡下さい．

参考文献

[1] S. B. Akers and B. Krishnamurthy: "A Group-Theoretic Model for Symmetric Interconnection Networks," *IEEE Transactions on Computers*, **38**, 4, pp. 555-566, 1989.

[2] Condor Project Homepage: http://www.cs.wisc.edu/condor/

[3] H. Dweighter: *Amer. Math. Monthly*, **82**, p. 1010, 1975.

[4] M. R. Garey, D. S. Johnson and S. Lin: *Amer. Math. Monthly*, **84**, p. 296, 1977.

[5] W. H. Gates and C. H. Papadimitriou: "Bounds for sorting by prefix reversals," *Discrete Math.*, **27**, pp. 47-57, 1979.

[6] D. S. Cohen and M. Blum: "On the problem of sorting burnt pancakes," *Discrete Appl. Math.*, **61**, 2, pp. 105-120, 1995.

[7] M. H. Heydari and I. H. Sudborough: "On the diameter of the pancake network," *J. Algorithms*, **25**, 1, pp. 67-94, 1997.

[8] 鴻池祐輔，金子敬一，品野勇治："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，**46**, SIG 10 (TOM 12), pp. 48-56, 2005.

[9] 浅井章吾，鴻池祐輔，品野勇治，金子敬一："PCクラスタを用いた16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，**47**, SIG 14 (TOM 15), pp. 71-79, 2005.

[10] On-Line Encyclopedia of Integer Sequences: http://www.research.att.com/njas/sequences/Seis.html