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

2004－ARC－157　(31)
2004－HPC－ 97　(31)
2004／3／3

グリッド技術を用いた進化系統樹推定の並列化

山本　洋[†]　中田秀基[††,†]
下平英寿[†]　松岡　聡[†,†††]

進化系統樹の推定では最尤法を用いた手法が最も優れた推定法の 1 つとされているが、最尤法の計算量は大きく、種の個数が増えると系統樹の個数は莫大となるため全系統樹の尤度を求めることは事実上不可能となる。系統樹の構成要素であるスプリットの尤度を最尤法によって計算し、スプリットの尤度を用いた行列計算によって系統樹の尤度を近似計算する手法が提案されている。しかし、種の個数がさらに増大すると、近似計算であってもすべての系統樹に対して行うことは困難になる。本研究では、系統樹の推定を系統樹空間における探索問題とみなし、最適化手法を適用することで、近似計算の対象となる系統樹の個数を削減する。また、グリッドミドルウェアを用いたマスタ・ワーカ方式を採用し、尤度計算および最適化手法の並列実行を可能にした。生物 9 種の系統樹推定において 16 ワーカを用いた結果、 64.0 倍の性能向上が得られた。

Parallelization of phylogenetic tree inference using Grid technology

YO YAMAMOTO ,[†] HIDEMOTO NAKADA ,[††,†]
HIDETOSHI SHIMODAIRA [†] and SATOSHI MATSUOKA[†,†††]

The maximum likelihood method is considered as one of the most reliable methods for phylogenetic tree inference. But if the number of species increases, it becomes impossible to calculate all phylogenetic trees, since the number of the trees increses explosively. An approximation method using *split* decomposition is proposed. It reduces calculation cost drastically, although, the calculation cost for larger number of species is still too high. We propose a method to reduce the cost using combinatorial optimization technique. We also parallelize it in a master-worker style using Grid Middlewares. The 64.0 times speedup is obtained as the result of using 16 workers in the problem of 9 species.

1. はじめに

現在地球上に生息する全ての生物は 1 つの共通祖先から進化してきたものであり、1 本の巨大な系統樹の中のどこかに位置づけられるはずである。このように多様な生物を系統樹の中に位置づけることが、進化系統樹の推定である。生物の系統関係を明らかにすることは、多様な生物が進化してきた機構を明らかにするためにも必要なことである。

従来の生物系統学では、生物の形態を比較することによってなされてきたが、形態レベルでは客観的基準が乏しく、結論が一致しないことが多かった。それに対し、生物の DNA に代表される遺伝情報を用いた、客観的なモデルに基づいて推定を行う分子系統学の研究がなされている。その最も優れた推定方法の 1 つが最尤法を用いた手法であるが、計算量が大きいため小規模の問題にしか適用できいことが問題であった。

本研究では、最尤法による信頼性を保ちつつ、計算効率の向上とグリッド環境への分散を行い高速化することを目的としている。

† 東京工業大学
Tokyo Institute of Technology
†† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
††† 国立情報学研究所
National Institute of Informatics

2. 系統樹推定と問題点

系統樹とは図 1 のような進化の分岐を表す木である。最尤法では、図 2 のような塩基配列などで表される生物の DNA 配列の座位配列 x_k に対する系統樹の尤度 $L(x_k)$ を計算し、その尤度の積 $\prod_k L(x_k)$ を DNA 配列から導かれる系統樹の尤度とみなす。$L(x_k)$ は非線形最適化により導かれ、反復法を用いて計算されるため計算量が大きく、$\prod_k L(x_k)$ の計算には種数に応じて表 1 の T_{ave} 程度の時間がかかる。系統樹の尤度を比較し、その尤度の大きいものほど系統樹としての信頼性が高いと評価する。しかし、系統樹の個数は

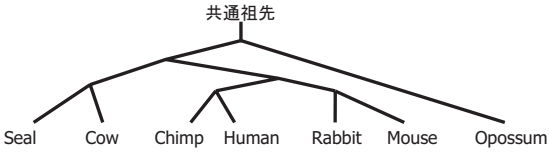

図 1 系統樹

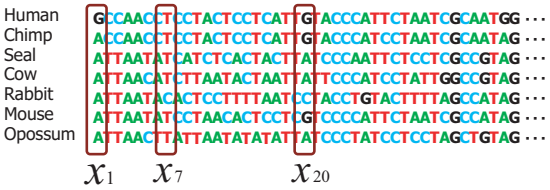

図 2 DNA の塩基配列

種数 n に対し $((2n-5)!)/(2^{n-3}(n-3)!) = O(2^n n!)$ 個と莫大であるために、種数が増加すると全系統樹に対して尤度を計算することは現実的に不可能である。

この問題に対し、尤度の近似計算を行う手法[3] が提案されている。系統樹を構成する枝を特にスプリットと呼び、種数を n とすると系統樹は $n-3$ 個のスプリットの集合に分解できる。このスプリットの尤度を最尤法を用いて全て求め、その尤度を基に行列計算で系統樹の尤度を近似する手法である。DNA 配列の長さ m に対し計算量は $O(m^3 + nm)$ で、最尤法と比べ非常に短時間で計算できる。全スプリットの個数は $2^{n-1} - (n+1) = O(2^n)$ 個であるために最尤法による尤度計算の個数を大幅に削減して実行時間を短縮できる。しかし、系統樹の個数は莫大であるために、種数が増加すると、この近似計算を用いても計算が困難となる。

3. 提案手法の概要

本研究では、最尤法を用いた系統樹推定の高速化を目標としている。系統樹の構成要素であるスプリットを用いて系統樹の尤度を近似計算する手法を検証した。その上で、系統樹の近似計算の回数を削減するために最適化手法を適用するとともに、グリッド上での実行を目指し Ninf[2], Jojo[1] を用いて並列化を行い、PC クラスタ上で評価を行った。その設計について説明する。

本プログラムでは、最尤法を用いた尤度計算の並列実行とスプリットを用いた系統樹の尤度の合成や最適化手法およびその並列実行を提供する。その際の実行の分岐は図 3 のようになる。系統樹のスプリット分解を用いたプログラムを利用する場合には、処理の過程は「最尤法を用いたスプリットの尤度計算部分」と「スプリットを用いた合成・探索計算部分」の 2 つに大きく分けられる。前者では、最尤法を用いて系統樹、またはスプリットの尤度を逐次または並列に計算する。

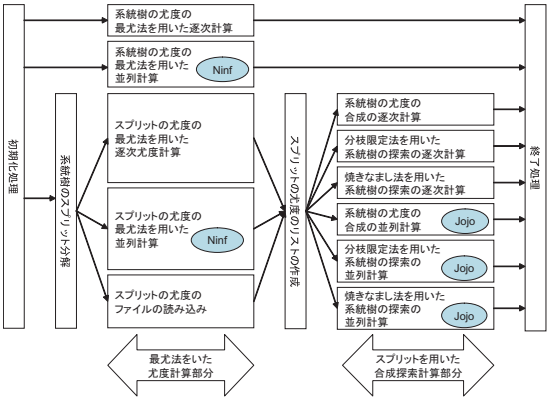

図 3 実行形式による分岐

また、一度計算したスプリットの尤度はファイルに出力され、このファイルを入力として与えることで、最尤法を用いたスプリットの尤度の計算を省略することもできる。後者では、全ての系統樹の尤度をスプリットの合成により求めるか、分枝限定法または焼きなまし法を用いて、尤度の上位の系統樹とその尤度を求めるかのいずれかの処理を逐次または並列に実行する。

それぞれの部分の並列化ではマスタ・ワーカ方式を採用し、通信ライブラリとして「最尤法を用いた尤度計算部分」では、GridRPC システムの 1 つである Ninf を、「スプリットを用いた合成・探索計算部分」では java による階層的な分散実行環境を提供する Jojo を用いた。

4. 系統樹推定の最適化

スプリットの合成を用いても、種数の大きい場合 $O(2^n n!)$ 個の全系統樹の尤度を求めることは不可能である。そこで、最大尤度の系統樹の探索の効率の向上を目指し、最適化手法を適用した。

4.1 分枝限定法の適用

各系統樹は $n-3$ 個のスプリットの組と対応することから、スター型系統樹に 1 つずつ合成可能なスプリットを付加していくことで、系統樹を生成することができる。各段階で合成可能なスプリットは複数あるので、それによって分岐することにより、図 4 のように末端が系統樹と対応する木が形成される。

分枝限定法では、この木を探索木として探索木の節で枝狩り判定を行う。この枝狩り判定により、最大尤度となる系統樹の存在しない節の探索を取りやめることが可能となるため、探索回数の節約が期待できる。

枝狩りでは、節における尤度の上界値を計算し、暫定解の尤度と比較する。もし上界値の方が大きければ、その節を展開して暫定解を更新できる可能性があるので、その節を展開して探索を続行する。一方、上界値が暫定解の尤度よりも小さければ、その節を展開して

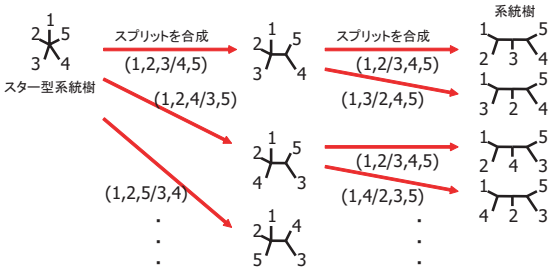

図 4　スプリットの合成による探索木

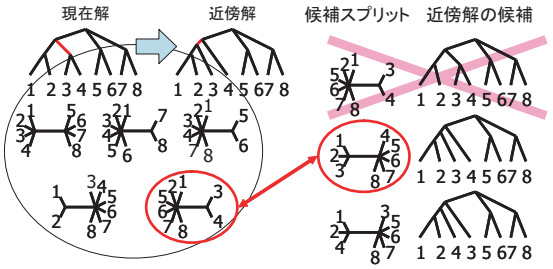

図 5　焼きなまし法の近傍解

も暫定解を更新できないため探索を取りやめる。

尤度の上界値は、その節のスプリットの組と合成可能なスプリットを全て合成したときの合成尤度とした。

4.2　焼きなまし法の適用

焼きなまし法で用いる近傍解は、図 5 のようにスプリットを基に導くような設計とした。まず、現在解である系統樹を表すスプリットの集合から無作為にスプリットを 1 つ削除する。残ったスプリットの集合に合成可能な 3 つのスプリットのうち、先ほど削除したスプリットを除いた 2 つの中からランダムに選び、スプリットの集合と合成する。この合成によって得られる系統樹を近傍解とした。また温度冷却アルゴリズムは冷却パラメータ α を用いた次の式とした。

$$T_{\mathrm{next}} = \alpha T_{\mathrm{current}} \quad (0 \ll \alpha < 1)$$

5.　系統樹推定の並列化

5.1　最尤法を用いた尤度計算の並列化

マスタは尤度計算すべき系統樹またはスプリットをプールに生成し、ワーカに 1 つずつ送信する。ワーカは最尤法を用いて尤度を計算し、マスタに尤度を返信する。マスタは返信された尤度を集計するとともに、プールが空でなければ新たな系統樹またはスプリットをワーカに送信する。すべての尤度を計算したら終了する。

5.2　合成による系統樹の尤度計算の並列化

マスタは合成すべき系統樹のプールから系統樹を 1 つずつ取り出し、系統樹と対応するスプリットの組をワーカに送信する。スプリットの組を受信したワーカ

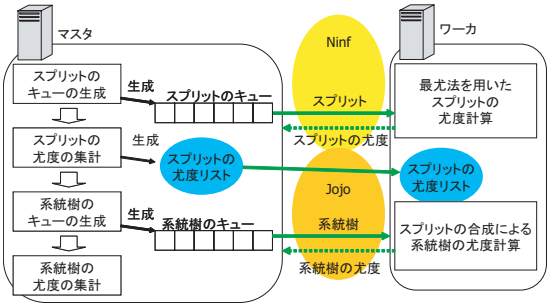

図 6　最尤法および合成による尤度計算の並列化モデル

は、スプリットの尤度リストを基に合成により系統樹の尤度を近似計算し、尤度をマスタに返信する。マスタは受信した尤度を系統樹と対応付けして集計するとともに、プールされた系統樹が残っていれば新たな系統樹と対応するスプリットの組をワーカに送信する。すべての系統樹の尤度を合成して終了する。略図を図 6 に示す。

5.3　分枝限定法の並列化

分枝限定法の並列化では、探索する木を複数のワーカに分割して割り振ることで、並列に探索する。しかし、探索木をどのように分割して各ワーカに割り当てるかによって、ロードバランスの悪化や、余計な探索が増加するという懸念もある。

というのも、分枝限定法の探索木では、同じ深さの節を比べてもその節を展開して得られる子問題の個数は大きく異なってくる。その上、節の展開の是非は枝狩り判定によって決まるため、実行時まで分からない。従って、ワーカに静的に部分木を割り振ったのでは、各ワーカの処理する問題の個数にばらつきが生じ、ロードバランスが悪い。そのため本研究では、各ワーカの持つ問題数に上限値を定めた。

マスタはまず探索木の根で 1 回分枝操作を行い、それにより生じた子問題を各ワーカに 1 つずつ割り振る。各ワーカは受け取った問題に対して分枝限定法を行い、終了したらマスタに次の問題を要求する。また、ワーカが持つ問題の数が上限値を超えた場合には、ワーカは一定数の問題を残し、残りをマスタに返却する。マスタは、ワーカに問題を割り当てた際に、その問題を自分のプールから削除し、ワーカから問題を返却された際には、それを自分のプールに追加する。マスタのプールが空になり、全てのワーカが次の問題をマスタに要求する状態になったら、処理を終了する。

5.4　焼きなまし法の並列化

焼きなまし法の並列化では、各ワーカがそれぞれ個別の温度を持ち独立に探索を行うレプリカ交換法を採用した。各ワーカは交換周期に基づき、マスタに現在解の尤度を送信して温度交換を要求する。このワーカをワーカ i としよう。マスタは尤度を受信すると、ワーカ i の温度を確認し、それよりも小さくて最大の

温度を持つワーカ j に対して、交換要求があったことを温度、尤度と共に通知する。通知を受けたワーカ j は、自らの温度、尤度と受信した温度、尤度に基づき解交換判定を行う。交換要求を棄却する場合には、棄却した旨をマスタに報告し、マスタはワーカ i に交換要求が棄却されたことを通知する。交換要求が受理された場合には、ワーカ j は温度を更新し、マスタに交換の受理を報告する。マスタは、マスタが保持する全ワーカの温度のリストにあるワーカ i とワーカ j の温度を交換し、ワーカ i にワーカ j が保持していた温度を送信する。ワーカ i は自らの温度をマスタから受信した温度に更新する。交換の結果よらず次の交換周期に至るまで、各ワーカは探索を行う。

6. 評　　価

6.1 評価問題

評価すべき事柄として以下を取り上げた。

スプリットの合成による系統樹の尤度計算　本研究で用いるスプリットの合成の有効性を検証するため、スプリットの尤度を最尤法を用いて計算し、その結果を元に系統樹の尤度を合成する手法の近似精度を評価する。また、この近似計算を行うことにより得られる性能向上について評価する。

最適化手法の導入による計算量の削減　最適化手法を用いて全系統樹の尤度を計算することなく最大尤度の系統樹を発見することによる合成回数の減少量について検証する。

並列化によるスケーラビリティ　一般に並列実行では、ノード数が増加するとノード間の通信が増加する上、マスタ・ワーカ方式ではマスタの負荷が増加するため、ノード数を増やしても一定速度で頭打ちになる。各並列化によるオーバーヘッド、およびワーカ数を増加した際に得られる性能向上、スケーラビリティを評価する。性能では、逐次での実行と複数のワーカを用いた場合の時間の逆比（速度比）

$$\mathrm{SpeedUp} = \frac{T_{\mathrm{serial}}}{T_{\mathrm{parallel}}}$$

で評価しており、特にマスタ 1 台・ワーカ 1 台を同一のノードに割り当てた場合の速度比をオーバーヘッドとしている。また、ワーカ数 N_{workers} におけるスケーラビリティは

$$\mathrm{Scalability} = \frac{\mathrm{SpeedUp}}{N_{\mathrm{workers}}}$$

として評価した。

最尤法による尤度計算プログラムとして paml[4] の塩基配列を用いた尤度計算プログラムを用いた。問題に用いた生物種はアザラシ、牛、ウサギ、オポッサム、マウス、ホモサピエンス、ジュゴン、アルマジロ、ラットで、種数に応じて前から用いる。各生物の遺伝子データは NCBI からダウンロードしたミトコンドリアの塩基配列を用いており、配列長は 3392 である。

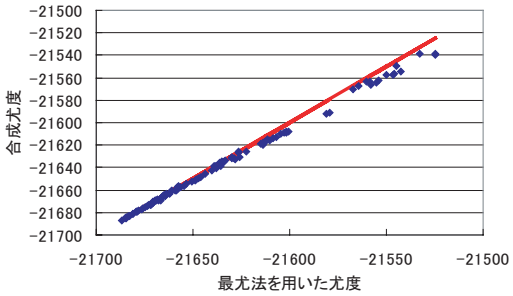

図 7　近似精度

表 1　最尤法による系統樹推定の予測時間とスプリット分解を用いた場合の時間

n	N_{tree}	T_{ave}	T_{paml}	T_{comp}
5	15	37 秒	9 分 30 秒	3 分 7 秒
6	105	85 秒	2 時間 30 分	6 分 55 秒
7	945	149 秒	1 日 15 時間	35 分 22 秒
8	10395	241 秒	29 日	2 時間 44 分
9	135135	330 秒	1 年 5ヶ月	18 時間 41 分

6.2 評価環境

評価環境として東京工業大学 情報理工学研究科 数理・計算科学専攻 下平研究室の abacus クラスタを用いた。スペックは CPU : AMD Athlon MP 2800+ × 2 、メモリ : 1024MB である。逐次計算ノード、並列実行のマスタ、ワーカのいずれもこのノードを用い、ワーカの台数を 2, 4, 8, 16 台として、ネットワークには 100base-T Ethernet を用いて評価した。

6.3 スプリットの合成による系統樹の尤度計算

スプリットの合成手法の近似精度を評価した結果を図 7 に示す。問題は 6 種で評価している。全 105 個の系統樹に対して、横軸に最尤法を用いて計算した尤度をとり、縦軸にスプリット合成して求めた尤度をとった散布図である。なお、直線は $y = x$ のグラフを表しており、散布図がこの直線に近いほど優れた近似であることを示している。この結果から、この近似手法は優れた方法であると考えられる。

続いて、スプリットの合成で近似計算することで得られる実行時間の短縮効果を評価した。種数 n の全系統樹に対し、スプリットの合成を用いた場合の時間 T_{comp} を最尤法を用いて尤度を計算した場合の予測時間 T_{paml} （系統樹 1 個当たりの最尤法の平均実行時間 $T_{\mathrm{ave}}\times$ 系統樹数 N_{tree}）と比較すると表 1 となり、大幅に高速化することができた。

スプリットの合成により系統樹の尤度を計算する場合には、「最尤法を用いたスプリットの尤度計算」と、「スプリットの合成による系統樹の尤度計算」の 2 段階を行うことになる。この 2 つの実行時間の内訳を

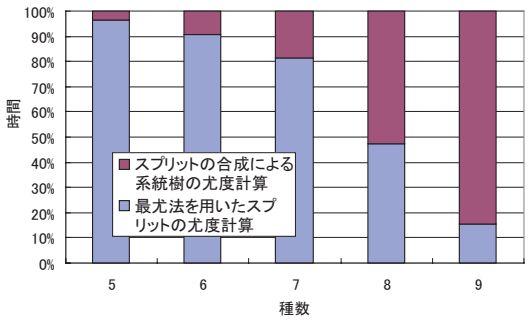

図 8　スプリット合成における実行時間の内訳

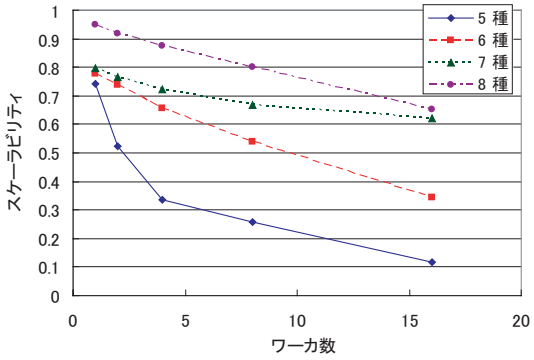

図 9　並列合成のスケーラビリティ

表すと図 8 のようになる。

これは種数 n に応じてスプリットの個数は $O(2^n)$ であるが、系統樹の個数は $O(2^n n!)$ であるために、種数が増加すると後者の時間の占める割合が増加することを表している。

そこで、前者では並列化を行い、後者では並列化および最適化手法の導入を行い、実行時間の短縮に努めた。

6.4　最適化手法の導入による計算量の削減

6.4.1　分枝限定法

分枝限定法では、通常の全探索と比べてどれだけ探索回数を減少できるかに注目して検証した。枝狩り判定時にその節におけるスプリットと合成可能な全スプリットの合成を行うため、探索木の末端における系統樹の尤度の合成計算と同程度の計算コストがかかる。したがって、末端での合成回数を数えるだけでなく、節での合成回数も数え、双方の回数の和である全合成回数を比較した。その結果、7, 8, 9 種で全合成回数をそれぞれ最大 83.4, 93.5, 95.3 %削減できた。

6.4.2　焼きなまし法

焼きなまし法では、終了条件によって探索回数は異なるため、ここでは、全系統樹の尤度を事前に調べ、その上位 1 %以内の解に 10 回至ることを終了条件とし、それまでの探索回数で検証した。また、確率的な探索を行うため、同じパラメータでも実行ごとに異なる探索となるので、同じパラメータで 3 回実行し、その平均値で検証する。パラメータとして用いたのは、初期温度と冷却パラメータ、初期解である。

7 種で初期温度、冷却パラメータを変更して検証した結果、初期温度を小さくすると収束までの探索回数は少なくなるが、冷却パラメータを大きくしなければ収束する確率が小さくなることが分かった。この結果から、初期温度を 10 、冷却パラメータを 0.99 にしたところ、合成回数を平均 95.2 %削減でき、ほぼ収束することが確認できた。

6.5　並列化によるスケーラビリティ

最尤法による尤度計算の並列化を 5, 6 種で評価したところ、並列化によって生じるオーバーヘッドは 6 種の場合で 12 %ほどもあることが分かった。これは決して十分小さいオーバーヘッドとは言えないだろう。これだけのオーバヘッドが生じる理由としては、様々な要因が考えられるが、種数の増加に伴いオーバーヘッドが増加したことから、本プログラムで行う系統樹の読み込み部分におけるオーバーヘッドではないかと考えられ、改善の必要がある。

性能向上では 8 ワーカで 6.5 倍から 7.5 倍の性能を、16 ワーカで 12.7 倍の性能を発揮できた。スケーラビリティについてみると、5 種、6 種の場合のいずれも常に 80 %を超える値を得られた。これは並列化によるオーバーヘッドが 12 %あることからすると、よくスケールしていると考えられる。

スプリットの合成による系統樹の尤度計算の並列化では、5-8 種で評価を行った。オーバーヘッドは 5 種で 30 %ほどにもなったが、 8 種では 5 %に抑えられた。8 種 16 台では 10.5 倍の速度向上が得られ、スケーラビリティは図 9 のようになった。

6.5.1　並列分枝限定法

分枝限定法の評価では、5-9 種で評価した。並列化によるオーバーヘッドは問題サイズが 6 種以上であれば 3-4 %程度であることが分かり、十分小さいものと考えられる。また、スケーラビリティを図 10 に示す。このグラフには、問題サイズが大きいほど台数効果がよく得られることが現れている。この評価では、ワーカ数が 16 台までしか実験をしていないため、問題サイズを十分大きくした場合に台数効果がどこまでスケーラブルなのか、適切な判断は得がたい。

6.5.2　並列焼きなまし法

並列焼きなまし法（レプリカ交換法）では、最大温度を 5, 10, 20, 50, 100, 200, 500 、最低温度を 1 とし、各ワーカに割り当てる温度はこの最大温度と最低温度の区間を等比で分割するような温度とした。例えば、最高温度が 8 で、ワーカ数が 4 であったら、各ワーカに割り当てる温度は 1, 2, 4, 8 となる。このような温度の割り当てで、ワーカ数を 4, 8, 12, 16 で種数 7 で実験を行い、全ワーカのうちいずれかのワー

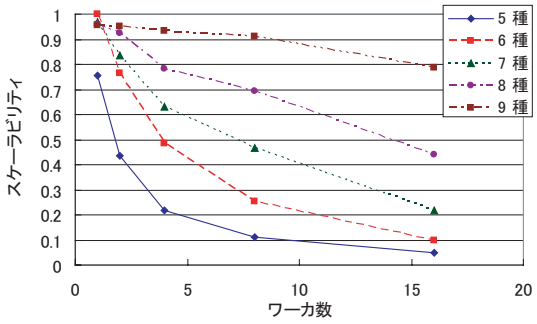

図 10 分枝限定法の並列化スケーラビリティ

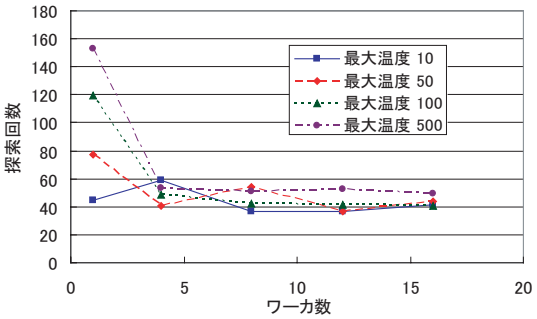

図 11 レプリカ交換法：種数 7 での結果

カが尤度の上位 1 %の系統樹を 10 回推移することを終了条件とし、その探索回数で評価した。その結果、図 11 が得られた。

この結果からは、レプリカ交換法での台数効果はほとんど得られないと考えられる。しかし、レプリカ交換法ではどのパラメータの場合にも全て収束した上、最大温度をどのような値に設定した場合にも、60 回以下の探索回数で収束しており、94 %以上削減できている。従って、レプリカ交換法では、この程度の問題規模ではワーカ数の増大による性能向上は期待できないものの、収束性の向上が得られたり、温度パラメータのチューニングに煩わされる手間が軽減できると考えられる。性能の向上については、より大きな問題で検証する必要がある。

7. おわりに

系統樹の構成要素であるスプリットの尤度を最尤法によって計算し、スプリットの尤度を用いて系統樹の尤度を簡単な行列計算によって近似計算する手法を検証した。近似の精度は十分に高く、全体の計算コストを大幅に短縮できるため、スプリットの尤度合成による系統樹の尤度計算が有効であることを立証した。

また、スプリットの性質を用いて最適化手法を導入し、探索効率の向上を目指した。尤度の合成計算の回数で評価したところ、生物 9 種の系統樹推定において、分枝限定法で 95.3 %、生物 7 種の系統樹推定において、焼きなまし法で 95.2 %減少させることに成功した。

最尤法およびスプリットの合成による尤度計算、最適化手法をグリッド上に適用することを目指し、計算機クラスタ上で並列実行を可能にした。16 台のワーカを用いた結果、生物 8 種の系統樹の推定において最尤法を用いた尤度計算は並列化で最大 12.7 倍、スプリットの合成は並列化で最大 10.5 倍、生物 9 種の系統樹の推定において分枝限定法は並列化で最大 12.6 倍の性能向上が得られた。また、焼きなまし法は並列化（レプリカ交換法）により、合成回数を少なく保ったまま収束性を安定させることができた。

今後の課題には他の最適化手法として遺伝的アルゴリズムの適用とその並列化や、他の尤度計算プログラムを利用可能にすることが挙げられる。また、大規模環境での実行で十分スケールすることを目指し、階層化による対応の検証を行う。問題規模に応じて実行時間が長期化することもあり、ノードの故障によって計算を中止させないフォールトトレランスについても取り組む必要がある。多くの系統学研究者に利用してもらうべく、簡単に利用可能なポータルサービスの実現も検討している。

謝辞 本研究の一部は、科学技術振興機構の計算科学技術活用型特定研究開発推進事業 (ACT-JST) 研究開発課題「コモディティグリッド技術によるテラスケール大規模数理最適化」の援助、および文部科学省科学研究費若手研究（A) 14702061「多重リサンプリングを用いたモデル信頼集合の構成法の開発とその応用」の援助による。

参 考 文 献

1) Hidemoto Nakada, Satoshi Matsuoka, and Satoshi Sekiguchi. A java-based programming environment for hierarchical grid: Jojo. In *CCGrid 2004*.
2) Hidemoto Nakada, Mitsuhisa Sato, and Satoshi Sekiguchi. Design and implementations of ninf: towards a global computing infrastructure. In *Future Generation Computing Systems, Metacomputing Issue*, Vol. 15, pp. 649–658, 1999.
3) Hidetoshi Shimodaira. Multiple comparisons of log-likelihoods and combining nonnested models with applications to phylogenetic tree selection. *Comm. in Statist., Part A–Theory Meth.*, Vol. 30, pp. 1751–1772, 2001.
4) Z. Yang. Paml: a program package for phylogenetic analysis by maximum likelihood. In *CABIOS*, Vol. 13, pp. 555–556, 1997.