グリッド向け実行環境 Jojo を用いた遺伝的アルゴリズムによる蛋白質構造決定

中田秀基[†1,†2] 中島直敏[†4] 小野功[†4]
松岡聡[†2,†3] 関口智嗣[†1]
小野典彦[†4] 楯真一[†5]

Java はグリッド上でのプログラミング言語として、1) クラスファイルのポータビリティによりアーキテクチャヘテロな環境への対応が容易、2) マルチスレッドによるレイテンシの隠蔽が期待できる、などの点で有望である。また、遺伝的アルゴリズムは実問題に応用範囲が広く、並列化や実行粒度の調節が容易なことから、グリッド上のアプリケーションとして適していると考えられる。われわれはこれらの点に着目し、Java による遺伝的アルゴリズム実行支援環境 jPop-GA の開発をすすめている。しかし、グリッド上で遺伝的アルゴリズムの効率的な実行を可能にする並列化方式については指針を欠いていた。

本稿では、遺伝的アルゴリズムのグリッド上での並列化に関する指針を得るべく、実アプリケーションである遺伝的アルゴリズムによる核磁気共鳴分光法による蛋白質構造解析を Java で実装し、これを Java 向けグリッド実行環境 Jojo を用い、2 つの並列化指針に基づいて並列化した。さらにそれぞれの実装に対して問題サイズを変えて評価を行い並列化手法に関する指針を得た。

Protein structure optimization using Genetic Algorithm on Jojo

Hidemoto Nakada,[†1,†2] Naotoshi Nakajima,[†4] Isao Ono,[†4]
Satoshi Matsuoka,[†2,†3] Satoshi Sekiguchi,[†1] Norihiko Ono[†4]
and Shin'ich Tate [†5]

Java language is suitable for Grid environment due to its 1) portability among heterogeneous architechtures, 2) integrated multi-thread capability that can effectively hide latency. Genetic Algorithm (GA) is a good candidate as an application area for the Grid, because its affinity for parallel execution. From these viewpoints, we are developping a Java-based programming framework for GA called jPoP-GA. However, we did not have concrete knowledge on effective parallel implementation method for GA.

In this paper, we implemneted a real GA application on top of Java-based Grid programming environment Jojo in several parallelization methods. As an application, we deployed protein 3-dimensional structure optmization using NMR spectroscopy. We performed several experiments on a Grid environment and obtained knowledge on parallelization of GA applications.

1. はじめに

Java はグリッド上でのプログラミング言語として、1) クラスファイルのポータビリティによりヘテロなシステムへの対応が容易、2) マルチスレッドによるレイテンシの隠蔽が期待できる、などの点で有望である。

†1 産業技術総合研究所 National Institute of Advanced Industrial Science and Technology (AIST)
†2 東京工業大学 Tokyo Institute of Technology
†3 国立情報学研究所 National Institute of Information
†4 徳島大学 The University of Tokushima
†5 生物分子工学研究所 Biomolecular Engineering Research Institute

グリッド上アプリケーションの候補の 1 つとして遺伝的アルゴリズムがある。遺伝的アルゴリズムは実問題に応用範囲が広く、並列化や実行粒度の調節が容易なことから、グリッド上のアプリケーションとして適していると考えられる。

われわれはこれらの点に着目し、Java による遺伝的アルゴリズム実行支援環境 jPop-GA[1)] の開発をすすめている。しかし、グリッド上で遺伝的アルゴリズムの効率的な実行を可能にする並列化方式については指針を欠いていた。

本稿では、上記の点に関する指針を得るべく、遺伝的アルゴリズムによる核磁気共鳴分光法での蛋白質構造解析を Java で実装し、これを Java 向けグリッド

実行環境 Jojo[2] でいくつかの手法を用いて並列化を行った。その結果、遺伝的アルゴリズムの並列化手法に関する指針を得た。

以降の本稿の構成は次のとおりである。2 では、遺伝的アルゴリズムを概説し、特に本稿で使用する遺伝的アルゴリズム手法のひとつである MGG について詳述する。さらに、本稿の対象アプリケーションである核磁気共鳴分光法での蛋白質構造決定問題について述べる。3 では、MGG の並列化手法を考察しその得失を議論する。4 で 3 で示した手法の実装を述べる。5 では実装したシステムの評価を行い、考察する。6 で結論と今後の課題を述べる。

2. 遺伝的アルゴリズムと蛋白質構造決定

2.1 遺伝的アルゴリズム

遺伝的アルゴリズムとは、生物の進化機構を模して最適化を行うアルゴリズムである。遺伝的アルゴリズムでは解の候補を個体と呼び、個体の集合に対してある手順で世代交代を行うことで、最良の個体を求める。世代交代は一般に、複製選択、交叉、突然変異、生存選択で構成される。複製選択とは、個体群から交叉、突然変異の対象になる個体を選択する過程である。交叉とは、複数 (通常 2 つ) の親個体から、なんらかの方法で親個体の性質を併せ持つ次世代の個体を生成することである。突然変異はとは、1 つの個体に対してなんらかの乱数的な処理を行って変更を加えることである。生存選択とは生成された個体群と前世代の個体群から、新しい世代の個体群を選択する手法である。

2.2 MGG(Minimal Generation Gap) 法

遺伝的アルゴリズムの探索効率は、問題を個体にコーディングする手法や、交叉、突然変異の設計だけでなく、世代交代の手法によっても大きく左右される。世代交代手法のポイントは、探索序盤における初期収束を避けること、探索後半にける進化的停滞を抑制すること、の 2 点である。MGG(Minimal Generation Gap) 法[3] はこれらの点に留意して設計された世代交代手法であり、その名のとおり世代間の相違が小さいことに特徴がある。

一般的な GA では、個体集合中のすべての個体を交叉の対象とし、交叉してできた個体と従来の個体から、何らかの方法で次の世代の個体集合を選択する。これに対して、MGG では 1 世代あたり 1 組の個体だけを交叉の対象とし、1 組の個体を生成して前世代の個体の組と入れ替え、新しい世代を生成する。したがって、各世代間の相違は最高で 2 個体だけである。ただし 1 組の個体から 1 組の個体を生成する際には、複数回 (通常数十回以上) の交叉を行い、そのなかから次世代に残す 2 個体を選び出す。この操作がローカルに強力な探索を行うことに相当するため、通常の GA よりも効率のよい探索が実現できる。さらに、世代間の格差が小さく個体群の多様性が保たれるため、初期収束や進化的停滞の少ない探索が可能になる。

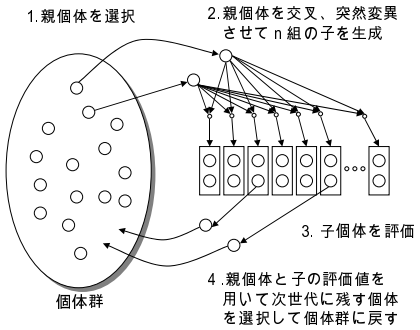

図 1 MGG による世代交代

次世代に残す 2 個体の選択手法にはいくつかの方法がある。文献 [3] では、最良 1 個体とランキングに基づくルーレットホイール選択による 1 個体を選択するとしているが、文献 [4] では最良 2 個体を選択している。本稿では最良 2 個体を用いる事とする。

MGG による世代交代の手順を以下にまとめる (図 1)。

(1) 個体群から交叉の対象となる親個体を 1 組選択する。
(2) 1 組の親個体に対して交叉を行い、複数組の子個体を生成する。さらに生成した個体に対して突然変異を行う。
(3) 子個体を評価する。
(4) すべての子個体と親個体から、次世代に残す個体の組を選択して個体群に戻す。

2.3 核磁気共鳴分光法による蛋白質構造決定

蛋白質は生体において重要な役割を果たしている。蛋白質の機能はその立体構造に依存しているため、蛋白質の立体構造決定は分子生物学において最も重要な課題のひとつである。現在用いられている立体構造決定には、核磁気共鳴分光法と X 線結晶解析法がある。

対象蛋白質を結晶化させなければならない X 線結晶解析法に対して、核磁気共鳴分光法は結晶化することの難しい蛋白質をも対象にすることができる。反面、核磁気共鳴分光法による立体構造決定は自動化されておらず、専門的な知識を持つ人間が作業しても構造を得るのに数ヶ月かかるという問題点がある。

2.4 遺伝的アルゴリズムによる核磁気共鳴分光法

われわれは核磁気共鳴分光法による蛋白質構造決定における、もっとも人月を要する過程である NOE(Nclear Overhause Effect) 帰属と呼ばれる過程を、遺伝的アルゴリズムを用いて自動化する研究を行っている [4]。開発は Java 言語と C 言語の双方で行っているが [5]、本稿では Java 言語で開発された版を用いる。

個体に相当するのは 2 面角で表現された立体分子

構造である。各個体の評価値は、個体が表現する立体分子構造をとった際に得られるはずの NOE シグナルが、実際に得られた NOE シグナルをどの程度説明しているか、および、個体が表現する立体分子構造に物理的な矛盾がないか、の両者を総合的に判断して決定する。交叉としては一様交叉を用い、2 つの個体の対応する 2 面角を互いに交換した。確率は 50%とした。突然変異としては、確率 0.01 で変数を選択し、その値に－ 1 度から 1 度までの一様乱数値を加えた。

3. MGG 法の並列化

MGG の過程において並列化できるポイントは大別して世代内と世代間の 2 つがある。

3.1 世代内並列化

世代交代で必要となる計算は、複製選択、交叉、突然変異、生存選択で構成される。生存選択の際には、個体の評価値計算が必要になる。

MGG では、複製選択は単純な乱数選択、生存選択の手法も最良 2 個体であるため、主な計算は、交叉、突然変異、生存選択のための個体評価、である。これらの各過程の計算量は、問題自身、問題のコーディング手法、交叉や突然変異の設計に依存して大きく変化するので一概には言えないが、一般には個体評価の計算量がドミナントである。

MGG では 1 世代につき数十個の子個体を生み出し、これらをそれぞれ評価する。これらの子個体の評価は完全に独立しており、並列に行うことができる。

3.2 世代間並列化

MGG における各世代の処理は、個体群中の 1 組の個体だけを対象とする計算過程である。したがって世代 n で処理対象とする個体組と、世代 $n+1$ で処理対象とする個体組が重複していない場合には、世代 n と世代 $n+1$ の処理は並行して行うことができる。

ここで実装を容易にするには、ある世代で処理対象とする個体組を選択する際に、計算が終了していない個体組を選択の対象からはずすことが考えられる。この方法は、個体群のサイズが重畳して実行する世代数に対して十分に大きければ、実用上問題はないと予測される。

しかし、並列化を世代間並列のみで行う場合には、プロセッサ台数が増大するにしたがって上記の条件を満たすことが難しくなる。したがって、ある程度以上の並列化を試みる場合には世代内並列と組み合わせることが必要になる。

4. 並列化 MGG 法の実装

4.1 グリッド向け実行環境 Jojo

実装には Java によるグリッド向け実行環境である Jojo[2] を用いた。Jojo は、複数のクラスタ計算機からなるグリッド環境に適した実行システムで、Globus[6] および SSH による安全性、プログラムコードの自動ステージング、ローカルファイルシステムの提供、カスケード式のリモート起動、直感的でマルチスレッド環境に適したメッセージパッシング API、といった特徴を持つ。

Jojo では、各ノード上で単一の代表オブジェクトが稼動するモデルをとる。ノード間の通信は、各ノード上の代表オブジェクトに対するメッセージパッシングで行われる。特徴的なのは、メッセージの送信がユーザによって明示的に行われるのに対して受信は暗黙裡に行われることである。Jojo の API にはメッセージを受信するための命令はない。ユーザコード中ではメッセージ受信時に起動するべきハンドラメソッドを指定する。メッセージが到着すると、新たなスレッドが生成され、そのスレッドでハンドラが起動される。したがって複数のメッセージを連続して受信した場合には複数のメソッドが同時にハンドラメソッドを実行することになる。このようにメッセージの送受信が完全に非同期になっていることで、通信と計算のオーバラップが可能になっている。

メッセージの送信には、柔軟な並列処理を可能にするために、返答の処理方法の異なる 4 つの呼び出しモードが用意されている。中でも返答受信時に、受信した返答に対して実行するハンドラを登録するタイプである `callWithContext` は通信と計算のオーバラップに貢献することが期待できる。

Jojo はリモートサイト上に自動的にシステムプログラムとユーザプログラムをアップロードして実行する機能を持つ。したがってリモートサイト上に Java の VM がインストールされているだけでよく、システムをインストールする手間がかからない。

4.2 MGG の並列実装

本稿では、MGG の並列実行はマスタ・ワーカ方式で行った。マスタで個体群全体を管理し、複数のワーカに対して処理を依頼する。

マスタにはワーカへの参照を保持したプールを用意する。ワーカに処理を依頼する際には、ワーカへの参照をプールから取り出して、そのワーカに処理を依頼する。評価が終了して返り値が返ってくると、そのワーカへの参照を再びプールに投入する。この機構は Jojo の `callWithContext` 呼び出しを用いて実装した。

評価する際にプールにワーカへの参照が残っていなかった場合には、そこでブロックしてすでに投入したジョブが終了するのを待つ。このように、終了したワーカには即座にジョブが投入され、セルフスケジューリングによる動的な負荷分散が実現される。

本稿では 2 つの方針で並列実行を実装した。1 つ目の**評価のみモデル**は評価部のみを並列実行するモデルである (図 2)。この場合、マスタで交叉による子個体を生成および突然変異処理を行い、評価部のみをワーカに依頼する。マスタとワーカとの間では個体がやり

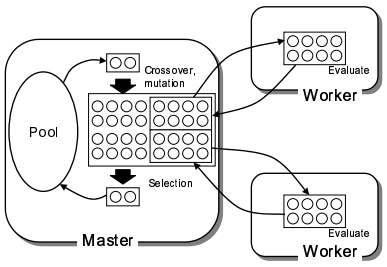

図 2 評価のみモデル

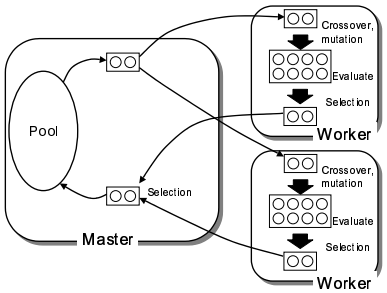

図 3 交叉評価モデル

とりされ、ワーカでの計算時間は個体の評価時間となる。マスタ・ワーカ間通信を効率化するため、複数の個体の評価を同時に依頼することもできる。この場合は、通信時間も評価時間も同時に評価する個体の数に比例する。

2 つ目の方針である**交叉評価モデル**は、マスタでは親個体の選択のみを行い、ワーカで交叉、突然変異、評価、さらには生存選択の一部まで行う方法である (図 3)。このモデルでも、ワーカで処理する個体数を変更することができる。マスタからワーカに送られるのは親個体組であり、ワーカからマスタに送られるのはワーカで生成した個体のうちの最良 2 個体となる。したがって、通信時間はワーカで処理する個体の数に依存せず一定である。これに対して評価時間はワーカ処理する個体の数に比例する。このため、生成個体の多くすることで、計算時間に対する通信時間を削減することができる。

さらにそれぞれに対して、世代間並列実行を実装した。本実装では、ある世代の親個体組として使用中の個体にはロックをかけ、次世代以降の親個体として選択できないようにした。世代の評価が終了すると、親個体を置換してロックを解除する。

5. 評価実験と考察

5.1 遺伝的アルゴリズムによる核磁気共鳴分光法の特徴

ここで、本稿の対象アプリケーションである、核磁気共鳴分光法の NOE 帰属問題の特徴を整理しておく。

各個体のデータ量は、対象とする蛋白質のサイズ (残基) にほぼ比例し、評価にかかるコストは蛋白質の残基数の二乗にほぼ比例する☆。交叉および突然変異にかかる計算コストは残基の数に比例するが、評価のコストと比較するとはるかに小さく、ほぼ無視できる。

表 1 核磁気共鳴分光法プログラムの特徴

	サイズ [Kbyte]	評価時間 [ms]
13 残基	2.8	55
27 残基	5.9	358

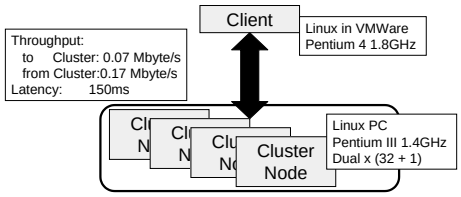

図 4 評価環境

評価で用いる 13 残基の蛋白質と 27 残基の蛋白質について、Java のシリアライザでシリアライズした際のサイズと、後述の評価環境のワーカ計算機を用いて単独で計算した際の個体一つあたりの評価時間を表 1 に示す。

13 残基の個体を用いる場合、MGG の 1 世代における交叉回数を 100 とし (すなわち子個体は 200 生み出される)、10000 世代実行すると、単一 CPU で実行する場合の実行時間は $55[ms] * 200 * 10000 = 110000[s] \approx 30.6[h]$ となる。

5.2 評価環境

評価環境としては図 4 に示す WAN 環境を使用した。サーバとしては産総研内にあるクラスタ計算機を使用し、クライアントとしては CATV ネットワークでインターネットに接続された PC を使用した。クライアント上でマスタプログラムを実行し、サーバ上でワーカプログラムを実行する。通信路としては、Jojo がサポートしている ssh による標準入出力ストリームを用いている。実験は数回行い最良値を採用した。

5.3 評価のみモデルと交叉評価モデル

まず、評価のみモデルと交叉評価モデルを比較するために、ワーカの数と同時評価する個体数を (4,10,20) と変化させて実験を行い 10 世代交代にかかった時間を計測した。プログラムスタート時の挙動の影響を防ぐために、5 世代目から 14 世代目の世代交代を計測している。実験には 13 残基の蛋白質を用いた。

図 5 に評価のみモデルの結果を示す。横軸がワーカの数であり、縦軸が経過時間である。ワーカ数 2 までは実行時間の短縮が見られるものの、それ以降はほとんど短縮が見られない。これは、13 残基の蛋白質ではワーカ側での評価時間と比較して通信時間が長大で

☆ 評価コストはカットオフの導入によってオーダを減じることが可能だと思われるが、現行の実装では実現されていない

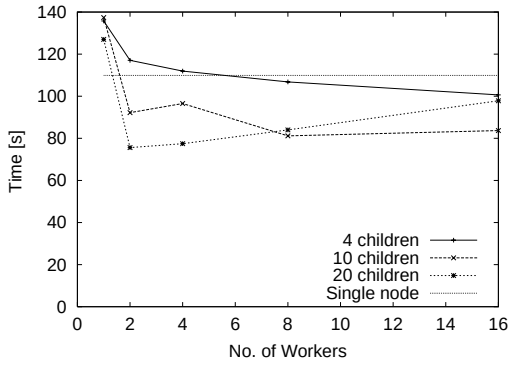

図 5 評価のみモデルの実行結果 (13 残基)

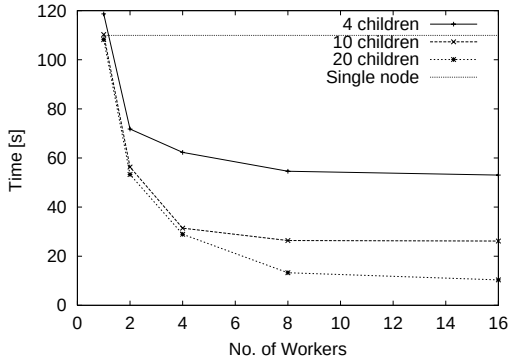

図 6 交叉評価モデルの実行結果 (13 残基)

あるため、マスタの通信がボトルネックとなるためであると考えられる。また、同時評価する個体数による影響は後述の交差評価モデルと比較すると小さい。しかし基本的には同時評価個体数が大きいほど性能が高い。これは、評価のみモデルでは通信する個体数と計算する個体数が一致することから、同時評価個体数を増減しても通信と計算の比率は変化しないものの、通信回数が減少することで通信のセットアップタイムが減少するためであると思われる。

図 6 に交叉評価モデルの結果を示す。同時評価個体数によってワーカ数に対する性能向上の度合いが変化することがわかる。同時評価個体数が 4 の場合には、ワーカ数 2 までは性能が向上しているがそれ以降は性能向上がみられない。同様に 10 の場合にはワーカ数 4 まで性能向上している。20 の場合にはワーカ数 16 まで性能向上が見られる。これは、交差評価モデルでは、同時評価個体数によって通信時間を削減することができるからである。通信時間が減少することで、クライアントがボトルネックになることがなくなるため、性能が向上しているのであると思われる。

5.4 世代間並列実行の効果

次に、世代間並列実行の効果を検証するために交叉評価モデルにおいて世代間並列を行った場合と、行わ

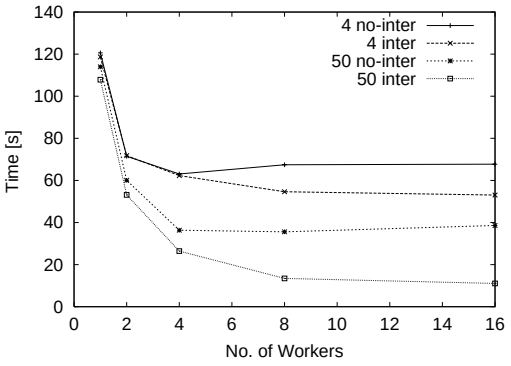

図 7 世代間並列実行の効果

なかった場合の比較を行った。同時評価する個体数は、4 と 50 とした。図 7 に結果のグラフを示す。図中凡例中の数字は同時評価個体数であり、no-inter は世代間並列を行った場合を、inter は行った場合を示す。

どちらの場合にも世代間並列を行ったほうが高速になっていることがわかる。これは、世代間並列を行わない場合には各世代の終了時に、すべてのワーカにジョブがない状態になってしまい、ワーカの計算資源を有効に利用できないためであると考えられる。

同時評価個体数 4 の場合は、ワーカ数 4 までほとんど差がないが、ワーカ数 8,16 では世代間並列を行ったほうが高速になっている。これに対して同時評価個体数 50 の場合には、ワーカ数 2 から 16 まで一貫して世代間並列を行ったほうが高速になっている。これは、同時評価個体数が増えると世代あたりのジョブ数が減少するため、ワーカ間での負荷の不均衡がより生じやすい状態になるためであると考えられる。

5.5 対象分子サイズの影響

対象分子のサイズが大きくなると、通信量はサイズに比例するのに対し計算量はサイズの二乗に比例するため、計算量と通信量の比率が変化する。この影響を見るために、対象分子の残基数が 13 の場合と 27 の場合について、実験を行った。

図 8 に評価のみモデルの結果を、図 9 に交叉評価モデルの結果を示す。横軸はプロセッサ数、縦軸はワーカに用いたノードで単体プログラムで実行した際の性能に対する性能の向上率である。同時個体評価数は 50 とした。

図 8 から、分子サイズが小さい場合には性能向上がほとんど見られないことがわかる。これに対して分子サイズが大きい場合にはワーカ数 4 までは性能が向上している。これは通信量と計算量の比率が変化し、通信時間が計算時間に対して小さくなったためであると考えられる。ワーカ数 8 および 16 での性能向上率はワーカ数 4 の場合とほとんど変わらない。これはプロセッサ数が増えたため再びクライアントの通信がボトルネックになったためであると考えられる。

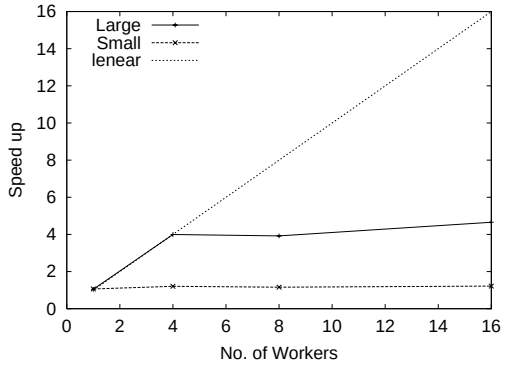

図 8 対象分子サイズの影響 (評価のみモデル)

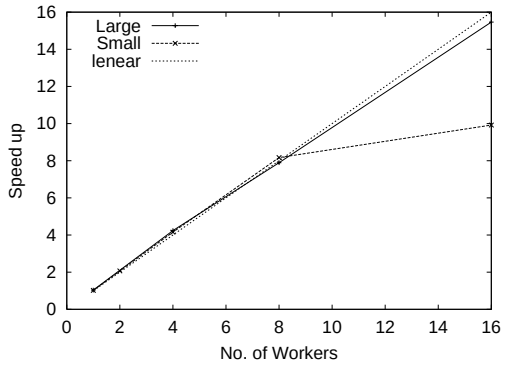

図 9 対象分子サイズの影響 (交叉評価モデル)

交叉評価モデルの結果 (図 9) では分子サイズが小さい場合でもかなりの性能向上が見られるものの、ワーカ数 16 では性能向上が鈍化している。これに対して分子サイズが大きい場合にはワーカ数 16 までほぼリニアに性能が向上している。

5.6 考　　察

実験の結果から、MGG の並列化手法として交叉評価モデルが優れていることが明らかになった。交叉評価モデルは評価のみモデルに対して実装が若干複雑になるが、それを考慮に入れても交叉評価モデルで実装することが望ましいことがわかった。

また、交叉評価モデルにおいては、同時評価個数は多いほど効率的になることがわかった。ただし同時評価個体数を増やすためには世代間並列実装が不可欠であることも示唆された。

ただし、ワーカの数が多い場合には同時評価個体数を増やしすぎると MGG の探索効率への影響が懸念される。3.2 で述べたとおり、世代間並列を行って複数の世代を同時に処理すると、親個体選択の対象が限定されてしまう。同時に処理できる世代数は、同時評価個体数とワーカの数に比例するので、ワーカの数が増えると同時に処理できる世代数が増大し、親固体選択が限定される効果が無視できなくなる可能性がある。この効果がどの程度探索効率に影響を与えるかは不明だが、ワーカの数に応じて同時評価個体数を調整する必要があると思われる。

6. おわりに

本稿では、遺伝的アルゴリズムの並列実装方針に関する知見を得るべく、核磁気共鳴分光法を用いた蛋白質構造決定プログラムを Java 向けグリッド実行環境 Jojo 上に並列実装し、評価実験を行った。その結果並列化手法としては交叉評価モデルと世代間並列が有効であることを確認した。

今後の課題としては以下が挙げられる。

- 本実装の有効性を検証するべく、より大規模な実験を行う。本稿で用いた構造決定プログラムは速度の点で最適化されておらず、大規模分子を対象とすることは難しい。今後最適化が必要である。
- 本稿の交差評価モデルは MGG の生存選択に最良 2 個体を用いることに依存している。この点の妥当性に関して検討が必要である。
- 今回得た指針に基づいて GA フレームワーク jPoP-GA の実装を変更する。また他の GA アプリケーションを同様に実装して、手法の一般性を確認する。

参 考 文 献

1) 秋山智宏, 中田秀基, 松岡聡, 関口智嗣: Grid 環境に適した並列組み合わせ最適化システムの提案, 情報処理学会 HPC 研究会 2002-HPC-91 (2002).
2) 中田秀基, 松岡聡, 関口智嗣: グリッド環境に適した Java 用階層型実行環境 Jojo の設計と実装, 情報処理学会 HPC 研究会 2002-HPC-92 (2002).
3) 佐藤浩, 小野功, 小林重信: 遺伝的アルゴリズムにおける世代交代モデルの提案と評価, 人工知能学会誌, Vol. 12, No. 5, pp. 734–744 (1997).
4) Ono, I., Fujiki, H., Ootsuka, M., Nakashima, N., Ono, N. and Tate, S.: Global Optimization of Protein 3-Dimensional Structures in NMR by A Genetic Algorithm, *Proc. 2002 Congress on Evolutionary Computation*, pp. 303–308 (2002).
5) 小野功, 今出広明, 中田秀基, 小野典彦, 松岡聡, 関口智嗣, 楯真一: 蛋白質立体構造の進化的解析のための Ninf 版並列 MGG とその性能評価, 情報処理学会 HPC 研究会 2002-HPC (to appear) (2003).
6) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications and High Performance Computing*, Vol. 11, No. 2, pp. 115–128 (1997).