Vol.2010-ARC-192 No.8
Vol.2010-HPC-128 No.8
2010/12/16

遺伝的アルゴリズムの並列計算システム向けフレームワークの提案

山中亮典[†1] 吉見真聡[†2] 廣安知之[†3]
三木光範[†2] 横内久猛[†3]

本研究では，ユーザが実行する遺伝的アルゴリズムの論理モデルを様々な並列計算環境において稼働させるためのフレームワークを提案する．実装モデルとしてクライアント・サーバ方式を採用することで，ユーザの稼働させたい理論モデルを実行することが可能である．並列計算環境の計算資源を十分に利用するためには，本来であれば高度なプログラミング技術が必要であるが，遺伝的アルゴリズムの開発ユーザがこのフレームワークを利用し，あらかじめ用意した評価部のテンプレートを利用することにより，並列計算環境を意識することなく，設定した遺伝的アルゴリズムの論理モデルを高速に実行することが可能となる．本稿では，フレームワークの実装にC#によるWCF（Windows Communication Foundation）を用い，システムを構築し，検討を行った．実装したフレームワークをテスト関数に適用した結果，2種類の記述を追加するのみでGAの並列化を実現できることを確認した．

Propose A Framework of Genetic Algorithm in Parallel Computing System

Ryosuke Yamanaka,[†1] Masato Yoshimi,[†2]
Tomoyuki Hiroyasu,[†3] Mitsunori Miki[†2]
and Hisatake Yokouchi[†3]

We proposed the framework to use logical model of genetic algorithm (GA) in various parallel environments. We adopted a client-server model for implemented model, and users are able to execute their logical model. It requires high technique for users to use the resource in a parallel environment. When users developed their code with the proposed framework and the prepared template of evaluation part, users can derive parallel code of GA which can use parallel environment efficiently. In this paper, we implemented the framework with WCF (Windows Communication Foundation) on C# language.As a result of using this framework for test function, we added only two definitions and developed the parallel code of GA.

1. はじめに

一般のパーソナルコンピュータ（PC）を用いた様々な規模のPCクラスタと，マルチコアプロセッサなどの様々な構成を持つハードウェアが普及してきた．そのような現状の中で，複数の計算資源を有する並列計算環境を有効に利用するための様々なアルゴリズムが開発されている．大規模な進化計算の分野では，最適化問題を解くために様々なアルゴリズムが提案されているが，通常それらのアルゴリズムを用いて良好な解を得るには，膨大な計算時間が必要であるため，対象問題によっては現実的な時間内に解を求めることが難しい場合がある．そのため，計算量を削減，もしくは高速に処理することが課題となっている．本稿では，最適化手法の1つである遺伝的アルゴリズム（Genetic Algorithm: GA）に関して，処理を高速化させるための実行方法について検討する．

GAは，生物の進化を参考にした進化計算と呼ばれる最適化手法の1つであり，トラス構造物の重量最適化問題[1)]，タンパク質の立体構造予測問題[2)]，ハイブリッドロケットエンジン（Hybrid Rocket Engine: HRE）の概念設計[3)]など数多くの分野に応用され，研究が進められている．

GAの計算時間を短縮するためには，並列計算環境下での実行が有効である．GAはそのアルゴリズム自体に並列性を内包するために多くの並列モデルが存在する．たとえば，グリッド環境を想定したGAの並列モデル[4)]やGAによるタンパク質構造決定など，並列性能を向上させるためにGAの探索性能を変化・向上させるような並列化モデルが提案されている．また，並列計算環境を利用しやすくするための研究も行われており，高度なプログラミング技術を要求されるCell Broadband Engineを利用するためのオフロード機構が提案されている[5)]．このような，ユーザのプロダクティビティを向上させるフレームワークをGAに適用することで，困難な並列化プログラミングの負担を軽減させることが可能である

†1 同志社大学 工学部
Faculty of Engineering, Doshisha University
†2 同志社大学 理工学部
Faculty of Science and Engineering, Doshisha University
†3 同志社大学 生命医科学部
Faculty of Department of Life and Medical Science, Doshisha University

© 2010 Information Processing Society of Japan

Vol.2010-ARC-192 No.8
Vol.2010-HPC-128 No.8
2010/12/16

と考えられる．ここでは，論理モデルとしての並列モデルと実際の並列計算環境において実行するための実装モデルを区別して考える．論理モデルは，逐次モデルとしての Simple GA[6] を基にした並列分散モデルで，分割母集団モデルなどがあげられるが，この論理モデルを逐次処理することも可能である．一方，実際の並列計算環境において GA を並列処理するためには，実装モデルが必要である．例えば，小野らのモデル[7] では，実装モデルとして，交叉，突然変異，評価を含んだマスタ・スレーブ型を提案し，効果をあげている．この例では，高い並列処理が実行可能ではあるが，実装モデルは論理モデルと強く結びついているために，それ以外の論理モデルを実装することはできない．ユーザが設定した論理モデルを変化させず実装するためには，実装モデルとしてマスタ・スレーブモデルを採用することが現実的である．

マスタ・スレーブモデルは，GA の評価部を分散処理するモデルであるため，評価部をうまく計算資源に配分する仕組みが必要である．しかしながら，近年の GPGPU やメニーコアを有した計算サーバにおいては，これらの計算資源を有効に利用するためには，ハードウエアに併せて GA の遺伝子情報を計算資源に配分する必要がある．本来であればこれらを行うためには，高度なプログラミング技術が必要であり，GA 開発ユーザには敷居が高い原因となっている．

そこで本研究では，実装モデルとしてマスタ・スレーブモデルを利用した GA 用のフレームワークを提案する．本フレームワークでは，GA 開発ユーザは任意の論理モデルを利用して，評価部以外の部分を実装する．各並列計算環境に応じた評価部の実装のテンプレートを用意し，各問題の実装と組み合わせることにより，評価部の実装を行う．このテンプレートを利用することで，GA 開発ユーザは，通信や並列計算環境に応じたスケジューラなどの知識がなくとも，実行する並列計算環境に適した遺伝的アルゴリズムを構築することが可能である．

本稿では，C#による WCF（Windows Communication Foundation）を用い，シンプルなシステムを構築し，基礎的な検討を行った．実装したフレームワークをテスト関数に適用した結果，2 種類の記述を追加するのみで GA の並列化を実現できることを確認した．

2. 遺伝的アルゴリズムと並列モデル

2.1 遺伝的アルゴリズム

遺伝的アルゴリズム（GA）は生物が環境に適応して進化していく過程を工学的に模倣した最適化アルゴリズムの一つである．自然界における生物の進化過程においては，ある世代を形成している個体の集団の中で，環境に適応した個体がより高い確率で生き残り，次世代に子を残す．この生物進化のメカニズムをモデル化し，環境に対して最もよく適応した個体，すなわち評価関数に対して最適値を与えるような解を計算機上で求めることが GA の概念である．

GA では母集団の各個体に対して交叉，突然変異といった遺伝的操作を適用し，新しい個体を生成する．その後，新しい個体に対する評価を行い，優れた個体を選択し，次世代に残す．これら一連の操作を，定められた終了条件まで繰り返すことで，解探索を行う．

GA の各個体は染色体によって特徴を持ち，染色体は遺伝子の集まりから構成される．GA では 1 つの染色体で 1 つの個体を表す．最適化問題の設計変数の値が染色体へとコーディングされ，GA の各操作は染色体あるいは遺伝子に対して行われる．GA は**図 1** に示す流れに沿って行われる．以下，それぞれの操作について簡単に説明する．

- 初期集団の生成
  予め定められた数だけランダムに個体を生成する．
- 評価
  各個体の持つ染色体を設計変数に変換し，解くべき最適化問題の評価値を計算する．その結果をもとに各個体の適合度を求める．
- 選択
  各個体の適合度に基づき，次世代母集団の候補を形成する．
- 交叉
  選択された個体群の中から 2 つの個体を選び出し，その 2 個体を親個体として 2 つの子個体を生成する．子個体の生成は，親個体の染色体を一部組み替えることによって行う．
- 突然変異
  染色体上のある遺伝子を一定確率で変化させる．突然変異によって，母集団内に新しい遺伝子を持つ個体を生成する．
- 終了判定
  予め定められた終了条件に基づいて GA を終了させる．

大規模問題に対して GA を適用した場合，評価に用いる目的関数が非常に複雑になる．そのため，評価計算に膨大な計算時間がかかる問題がある．

© 2010 Information Processing Society of Japan

Vol.2010-ARC-192 No.8
Vol.2010-HPC-128 No.8
2010/12/16

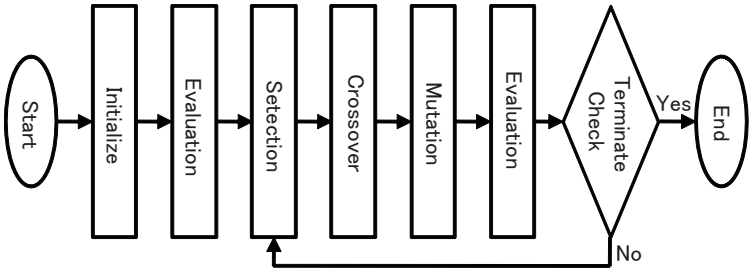

図 1 GA のフローチャート
Fig. 1 Flowchart of GA.

2.2 GA の並列モデル

GA の膨大な計算時間を短縮するため，並列処理に関する議論がなされ，多くの並列モデルが提案されている[8]．GA の並列モデルは論理モデルと実装モデルを区別して考える必要がある．すなわち，論理モデルは並列的に処理を行うモデルであるが，実際には逐次処理を行うことも可能である．この際，逐次モデルと比較して解探索の性能を向上させる論理モデルも存在することに留意すべきである．一方，実装モデルは，実際に並列処理を実行するために採用するモデルであり，実装モデルによっては，論理的には逐次モデルを採用することも可能な場合も存在し，実装モデルの制約で，論理モデルの変更を余儀なくする場合も存在する．本稿では，実装モデルとしてマスタ・スレーブモデルを使用する．マスタ・スレーブモデルは，GA における処理のうち，計算時間の大部分を占める評価計算を並列化するモデルであり，**図 2** に示す構成をとる．ユーザ端末がマスタ，クラスタ内に存在する計算ノードがスレーブの役割を担う．マスタが GA の遺伝的操作を行い，スレーブが個体の評価計算を行う．マスタからスレーブへの通信では各個体情報が送信され，スレーブからマスタへの通信で評価後の個体情報が送信される．マスタ・スレーブモデルを実装モデルとして採用することで，ユーザがいかなる GA の論理モデルを採用しても基本的には対応可能であるため，提案するフレームワークに適していると考えられる．

3. 提案フレームワーク

本稿で提案するフレームワークは，利用者が特別な並列化プログラミング技術を有する必要なく GA の実装モデルとしてのマスタ・スレーブモデルの並列処理を実現することを目的としている．**図 3** に提案フレームワークの概要を示す．GA 開発ユーザは GA の任意の

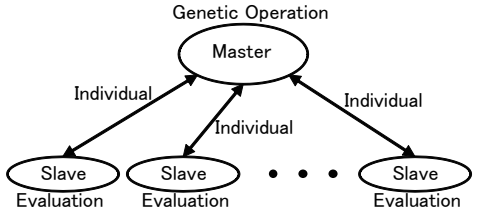

図 2 マスタ・スレーブモデル
Fig. 2 Model of master and slaves.

論理モデルを構築し，評価部以外の部分を実装する．評価部においては，対象問題の評価部のコードと，あらかじめ用意されているテンプレートを利用して実装する．このテンプレートを利用することで，特殊な通信と評価タスクのスケジューリングの実装を GA 開発ユーザから遮蔽することが可能である．

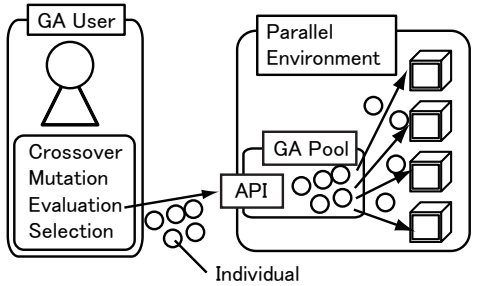

図 3 提案フレームワークの概要
Fig. 3 Concept of the framework.

上記で説明したとおり，並列計算環境においては，各遺伝子が各計算資源に送られ評価を行う．これらを効率的に行うためには，送信する遺伝子の

- 効率的な転送
- 効率的な遺伝子の静的再配置
- 効率的なバランシングを行うための動的再配置

が必要である．提案フレームワークにおける評価部のテンプレートでは，計算資源の情報を何らかの方法で取得し，この情報を利用して上記の仕様を満たす実装を行うものとする．

© 2010 Information Processing Society of Japan

Vol.2010-ARC-192 No.8
Vol.2010-HPC-128 No.8
2010/12/16

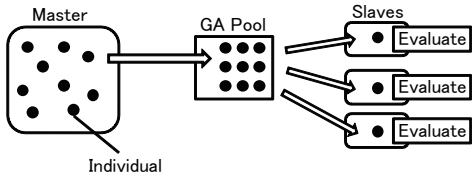

図 4 個体の送信方法
Fig. 4 Sending method of individuals.

表 1 個体クラス（ソースコード）
Table 1 Individual class(source code).

```
1 class Individual {
2    public int[] gene;
3    public double fitness;
4 }
```

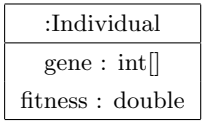

図 5 個体クラス
Fig. 5 Individual class.

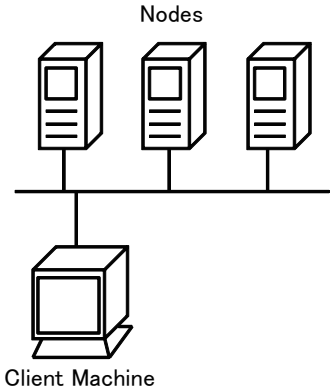

図 6 クラスタ構成
Fig. 6 Architecture of cluster.

4. Windows Communication Foundation による実装と評価

4.1 Windows Communication Foundation による実装

本稿で提案するフレームワークの実装には，C#による WCF（Windows Communication Foundation）を用いた．WCF はクライアント・サーバ方式のアプリケーションを作成するためのプログラミングモデルである．WCF では，関数をサービスという形式で定義しサーバーに配布することで，クライアントから独立した関数を呼び出すことが出来る．サービスとは具体的には，DLL（Dynamic Link Library）形式で提供される．すなわち，評価関数を DLL 形式で定義し，各計算ノードに配布する．そして，マスタから定義した DLL を呼び出すことで並列化を実現する．本実装に WCF を用いた理由として以下の点が挙げられる．

- マスタとスレーブが互いに独立
- 評価関数のみをスレーブに設置可能

マスタとスレーブの間でソースコードや実行ファイルを共有しないため，拡張性に優れている．また，DLL 形式で関数のみをスレーブに設置できるため，スレーブが評価のみを実行するマスタ・スレーブモデルに適していると考えられる．

4.2 GA の設計

提案するフレームワームでは，個体を表現する形式を統一する必要があるため，本稿では，GA を実装する際の個体表現の方法を**表 1** および**図 5** のように定義する．また，GA における母集団は，上記クラスの配列として定義する．個体の通信方法については，**図 4** に示すように，フレームワークに全個体を渡し，各計算ノードに 1 個体ずつ送信する．

4.3 利 用 環 境

ユーザが利用する並列計算環境として**図 6** に示す，汎用プロセッサによって構築されたクラスタを想定している．本クラスタ環境はユーザ端末であるクライアントマシンと全ての計算ノードが直接ネットワークで接続されている必要がある．

4.4 実　　装

提案するフレームワークは，クラスライブラリの形式で作成しているため，ライブラリに定義された関数を用いることで容易に並列化が実現できる．ユーザがコーディングにおいて行う必要がある処理を以下に示す．

- クラスインスタンスの生成
- 母集団を引数に，並列化メソッドを呼び出す

なお，クラスインスタンスを生成する際に，引数として利用するコアの最大数および GA のパラメータを与えるものとする．

実装したフレームワークの利用例を**表 2** に示す．また，作成したフレームワークの持つ

© 2010 Information Processing Society of Japan

Vol.2010-ARC-192 No.8
Vol.2010-HPC-128 No.8
2010/12/16

表 2 フレームワークを利用した GA の一例
Table 2 Example of GA which uses the framework.

```
1 static int Main(string[] args) {
2     Framework f = new Framework(MAX_NODE_NUM, PARAMETER);
3     Individual[] population = new Individual[POPULATION_SIZE];
4     for(i = 0; i < MAX_GENERATION; i++) {
5         population = f.ParallelEvaluate(population);
6         // Genetic Operation
7     }
8     return 0;
9 }
```

変数およびメソッドを**表 3** および**表 4** に示す．表 2 の 2 行目と 5 行目がフレームワークの利用に際して必要な処理である．2 行目では利用する評価関数を引数に取りフレームワーククラスのインスタンスを生成する．5 行目で，Individual クラスの配列である母集団を引数に取る並列化関数を呼び出す．本フレームワークでは，以上の 2 つの処理だけで GA の並列化を実現できる．

4.5 評　　価

並列計算環境として，Windows HPC Server 2008 によって構成されたクラスタを用いた．クラスタを構成するマシンの構成を**表 5** に示す．

GA の評価関数として，JAXA 宇宙科学研究所宇宙輸送工学研究系[9] より公開されているハイブリッドロケットエンジン（HRE）概念設計最適化問題の目的関数評価モジュールを利用する．HRE では，燃焼室内での乱流境界層燃焼により推力を得ており，一般的な設計知識を得ることが難しい．そのため，HRE の概念設計を最適化問題とすることで設計知識を得る研究が行われている[3]．HRE 概念設計最適化を行う際の GA パラメータを**表 6** に示す．

表 3 関数名と機能
Table 3 List of functions.

関数名	機能
ParallelEvaluate	母集団を引数に取り，評価を行う

表 4 変数型と機能
Table 4 List of variables.

変数名	機能
max_core_num	使用する計算資源の最大数
parameter	ユーザが作成した GA のパラメータ

表 5 マシンの構成
Table 5 Architecture of machines.

OS	Windows Server 2008 HPC Edition
メモリ	8 GB
プロセッサ	AMD Opteron 2356 × 2
周波数	2.30 GHz
コア数	4

表 6 GA のパラメータ
Table 6 Parameter of GA.

パラメータ	値
母集団サイズ	64
染色体長	41
世代数	32

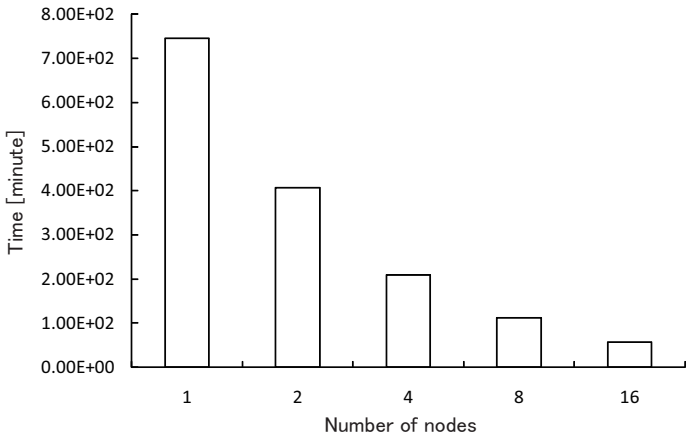

図 7 使用した計算ノード数と実行時間の関係
Fig. 7 Relation between number of node and execution time.

今回用いる GA では母集団を世代数分だけ評価するため，評価回数は 64 × 32=2048 回である．表 5 に示すマシンの 1 つのコア上で表 6 のパラメータを用いたて HRE の概念設計最適化問題評価モジュールを実行した場合，1 度の評価に要する時間は約 19.33 sec である．

HRE の概念設計最適化問題に，提案フレームワークを用いて実装した並列 GA の，実行

© 2010 Information Processing Society of Japan

Vol.2010-ARC-192 No.8
Vol.2010-HPC-128 No.8
2010/12/16

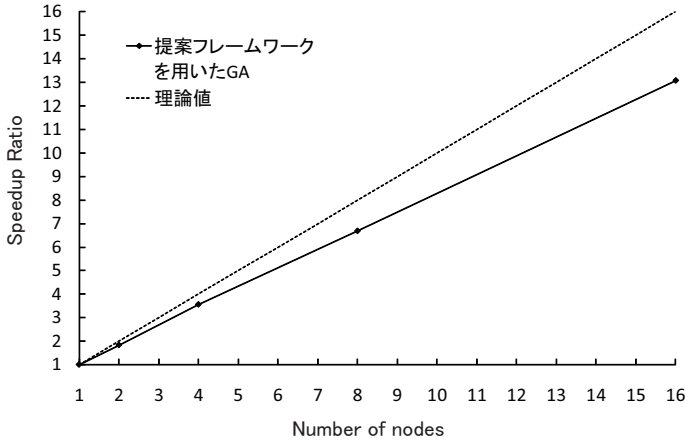

図 8 計算ノード 1 つを用いて計算した時間を 1 として正規化した場合の倍率
Fig. 8 Speed up ratio(compared with 1 node).

表 7 実験結果
Table 7 Experimental result.

計算ノード数	総実行時間 [min]	速度向上率
1	746	1.00
2	407	1.83
4	210	3.55
8	112	6.68
16	57.0	13.07

時間および計算ノード数 1 時の実行時間に対する速度向上率，計算ノード数と実行時間の関係グラフ，および計算ノード数に対する速度向上率をそれぞれ**表 7**，**図 7**，および**図 8** に示す．表 7 より，提案フレームワークを用いて実装した GA は，HRE 概念設計最適化問題に関して，計算ノードの増加に伴い実行時間が短縮されているため，並列性を確認できる．

図 7 の実行時間データから，計算ノードが増加するにつれ実行時間が短縮されることが確認できる．しかし，図 8 の速度向上率データを見ると，計算ノード数の増加に伴い，速度向上率が落ちている．実装したフレームワークでは，計算ノードの利用に関するスケジューリングを考慮せず，1 スレッド内の for 文で個体の送受信を行っているため，クラスタ内に存在する計算資源を有効利用出来ていないと考えられる．

5. まとめと今後の課題

本稿では，GA を並列計算環境で用いる際のフレームワークを提案し，実問題である HRE の概念設計最適化問題に対して適用した．その結果，提案フレームワークを用いて実装した GA において確かな並列性が確認できた．提案フレームワークの利用方法は，クラスインスタンスを生成し，クラスメソッドを利用するのみである．このような簡便な利用法で並列化が実現されるため，ユーザのプロダクティビティを向上することが出来る可能性を示した．

本稿では，属するコンピュータの CPU アーキテクチャが同一であるクラスタを評価環境として利用した．しかし，近年の CPU のマルチコア化およびアーキテクチャの多様化の傾向から，今後は様々なプロセッサで構成されたクラスタへの対応が必要であると考える．そのために提案フレームワークに追加するべきであろう機能を以下に示す．

- スケジューリング機能
- クラスタ内のプロセッサの構成を調べる機能
- 調べた構成に従って，個体の送信方法および送信先を最適化する機能

これらの機能を実装することで，多様なアーキテクチャで構築された，様々なクラスタで利用可能な共通のフレームワークの作成を目指す．

参 考 文 献

1) 薫田匡史，大森博司，河村拓昌：大型望遠鏡を支持するトラス構造物の多目的最適設計，日本建築学会大会学術講演梗概集， Vol.2008, pp.917–918 (2008).
2) 中田秀基，中島直敏， 小野功， 松岡聡，関口智嗣，小野典彦， 楯真一：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，情報処理学会研究報告. [ハイパフォーマンスコンピューティング]， Vol.2003, No.29, pp.155–160 (2003).
3) 小杉幸寛， 大山聖，藤井孝藏，金崎雅博：ハイブリッドロケットエンジンの概念設計最適化，2009 年度宇宙輸送シンポジウム (2009).
4) 吉井健吾，廣安知之，三木光範：グリッド環境を想定した多目的遺伝的アルゴリズムの並列モデルの提案およびその検討，情報処理学会研究報告. BIO, バイオ情報学, Vol.2006, No.135, pp.57–60 (2006).
5) 鎌田俊昭，西川由理，吉見真聡，天野英晴：Cell Broadband Engine 向けオフロード機構の提案，情報処理学会研究報告， Vol.2010, No.21, pp.1–6 (2010).

© 2010 Information Processing Society of Japan

Vol.2010-ARC-192 No.8
Vol.2010-HPC-128 No.8
2010/12/16

6) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley (1989).
7) 小野功，水口尚亮，中島直敏，小野典彦，中田秀基，松岡聡，関口智嗣，楯真一：Ninf-1/Ninf-G を用いた NMR 蛋白質立体構造決定のための遺伝アルゴリズムのグリッド化，情報処理学会論文誌．コンピューティングシステム， Vol.46, No.12, pp. 369–406 (2005).
8) 廣安知之：並列・分散 GA，GP，人工知能学会 人工知能学事典，共立出版 (2005).
9) JAXA 宇宙科学研究所宇宙輸送工学研究系：`http://flab.eng.isas.jaxa.jp/`.

© 2010 Information Processing Society of Japan