ハイパフォーマンス
コンピューティング 91−31
（２００２．８．２３）

動的な階層型システムにおける最適化計算法の検討

上川　純一 ††　廣安　知之†
三木　光範 †　谷村　勇輔 ††

本論文では，分散して存在するコンピュータ資源を利用するためのシステムとして DNAS を提案する．DNAS はノード間での相互の通信トポロジとしてツリー構造を採用している．DNAS はツリー構造を動的に変化させ，障害に対応できる機構を持つ．このシステム上で最適化手法であるランダムサーチと遺伝的アルゴリズムを実装した．ランダムサーチはノード数が多くなると探索時間が短縮可能であった．遺伝的アルゴリズムでもノード数を増加させるとより探索時間に改善は見られたが，ノード数に対してのスケールは良好な結果が得られなかった．

Optimization Computation in Dynamic Hierarchical System

Junichi Uekawa,†† Tomoyuki Hiroyasu,† Mitsunori Miki†
and Yusuke Tanimura ††

In this paper, DNAS, a system for utilizing computer resources that exist in distributed manner is proposed. DNAS uses tree structure for communication topoogy between nodes. We implemented optimization methods, such as genetic algorithm and random search on this system. Random search improved search time with increase in number of nodes. For genetic algorithm, search time improved with increase in node number, but the result was not as good as it was expected.

1. はじめに

本研究の目的はグリッド計算環境を含めた並列計算機上でアプリケーションを動作させるミドルウエアの構築にある．そのために，並列計算機のノード間でのネットワークの通信経路に論理的階層構造を構築する．また階層構造を動的に変化する機構を導入する事で，ノードの停止等に対応できるようにする．この構造上でターゲットとなるアプリケーションの一つである最適化計算を行えるようにする．

筆者らは Grid 計算環境において効率的に利用できるアプリケーションは一般にいくつかの共通の性質を持っていると考え，それらの性質を有するアプリケーションを GOCA(Grid Oriented Computing Apllication) と呼んでいる．

PC などの未使用の計算資源を有効利用しようとするプロジェクトは GOCA の一部であると考えられる．一般に利用されるコンピュータシステムにはアイドル時間が多く存在する．そのため，未使用の計算資源を有効活用して，計算アプリケーションを動かす事が可能である．SETI@home[3] や Folding@home[4] 等のプロジェクトは未使用のリソースを利用して問題を解決するための計算を行うプロジェクトである．これらのシステムにおいては中央サーバが問題やデータを各ノードに送信し，各ノードがその部分を計算し，中央サーバに結果を返送する．これらのプロジェクトでは各計算ノードが計算中に通信する必要がない問題を対象としている．

また，Grid RPC を利用したシステムにおいても GOCA を効率良く稼動させることが可能である．NetSolve や Ninf/G といった Grid RPC では GOCA を計算サーバーに登録しておくことで，クライアントからの要求に従いエージェントが Grid 上の GOCA サーバの情報を通知する．

しかしながら，これらのシステムの多くは各ノードが計算の途中で通信を行う必要のないタイプの GOCA を対象としている．そのため，各ノードが通信を必要とするタイプの GOCA には基本的には対応することができない．

遺伝的アルゴリズム[1](以下 GA) は最適化手法の一

† 同志社大学
Doshisha University
†† 同志社大学大学院
Graduate School of Doshisha University

つでアルゴリズム的に並列性を内在する手法であり，多くの並列モデルが存在する．分散遺伝的アルゴリズム[2](以下 DGA) はその一つであり，DGA は計算途中に通信を必要とする GOCA であると言える．なぜならば，その通信量は大きくなく，かつ頻繁に通信が行われるわけではない．かつ，通信が途中で失敗したとしても，計算を中止する必要がないアルゴリズムであるからである．

本研究では，DGA のようなノード間通信を含み，特定の問題に特化しないでコンピュータの資源を利用するためのミドルウェアシステム DNAS (分散ネットワークアプリケーションシステム Distributed Network Application System) を提案する．DNS 等の従来の階層構造を利用したシステムは，アプリケーションを特化していて，階層構造の変化についても静的に定義されている．DNAS は動的に変化する論理的階層構造を有している．

ネットワークの論理的な階層構造は，各ホストがネットワーク上で情報パケットを伝達する相手を制限することにより構成する．制限方法を送信相手との接続関係で表すと**図 1** のようにツリー状の階層構造になる．各ノードが情報中継サーバとして複数のクライアントからの情報を受け付け，それを処理して，そのサーバがさらに自分より上位のノードに伝達する．各ノードはサーバでもクライアントでもあるため servent ☆と呼ぶ．サーバ機能を分散することにより，中央ノードに負荷を集中せずに処理が行える．計算問題で，部分部分で計算を行いその部分解をあつめることにより最終的な解を出す事ができる種類の問題には有効な考え方である．階層構造にすることによりネットワーク通信が局所化できるなどの利点が考えられる．

本研究では階層構造の実現方法を説明し，ランダムサーチの実験について説明する．次に，その後階層構造をとるネットワーク上での GA のモデルについて解説する．さらに，実験を行い，今回構築した手法が実際に動作することを確認する．提案システムは Grid 計算環境での使用を志向したものであるが，本研究における数値実験では PC クラスタ内のノードにおいて実験を行っている．

2. 階層的構造システム上での遺伝的アルゴリズムの提案

2.1 階層的構造

既知のネットワークの中で，木構造を動的に生成し，

☆ serv-er + cli-ent = serv + ent

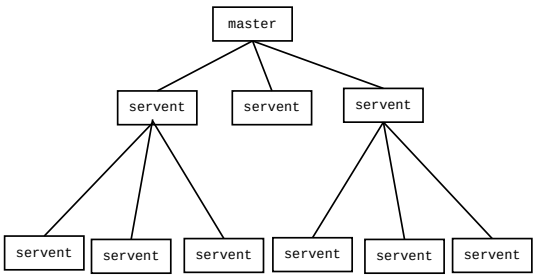

図 1 Tree structure within the cluster

情報の伝達を効率よく行うのが DNAS のシステム全体としての目的である．

階層構造を作るには，事前にユーザにより規定した階層構造を利用することもできる．そのように作成された階層構造は中間層にあるノードの障害に対して非常に弱い．しかし，そのような階層構造に対して，階層構造を停止したノードを動的に削除して作りなおすようなシステムを構築すれば，ノードの障害に耐えられるシステムを構築することが可能である．ただし，マスターノードが停止した場合にはこのシステムは停止する．

システムを階層的にすることにより，完全に分散したシステムと比べて利点が生まれる．もっとも顕著なものとしては，効率の良い情報の複製が行われる[5] ということがある．これは，階層的なシステムでは，一台のサーバからの情報を取得したノードがそれぞれサーバとして情報を提供することになるという構造自体の特性による．

情報は，下位 (マスターノードからの論理的距離が比較的遠いもの) のノードからより上位 (マスターノードへの論理的距離が比較的近いもの) のノードに対して伝達されていく．下位のノードからの情報を中間にある比較的上位のノードはさらに上位のノードに中継して行く．そして，最終的には最上位にあるマスターノードに情報が伝達される．上位にあり直接接続しているノードを uplink , 下位にあるノードを downlink と呼ぶ．

2.2 遺伝的アルゴリズム

GA では，最適化問題における問題の入力パラメータの数値変数を遺伝子型として扱う．その遺伝子型に対して，遺伝的操作として交叉と突然変移を定義する．交叉とは，二つの遺伝子型から次の世代の遺伝子型を導出する手法であり，突然変異とはそれ以外の要因でランダムな変化を起こす手法である．このようにして定義した遺伝子型それぞれを個体として扱う．複数の個体に対して複数世代にわたって交叉と突然変異を行

い，その中で適合度の低い個体を淘汰する処理を行う．適合度とは，各個体の評価値であり，この値が高いほど次世代に残る確率が高い．このようなアナロジーに基づいている遺伝的アルゴリズムは最適化問題の解法の一つとして利用されている．

分散遺伝的アルゴリズムは遺伝的アルゴリズムを，複数のコンピュータで分散的に行うモデルである．複数の個体群が別の島に存在しており，その島の内部で進化が起き，まれに個体が他の島に移住するモデルである．各島を各コンピュータが担当する．それぞれのコンピュータ上で独立して遺伝的アルゴリズムを行う．

従来の島モデルの DGA システムは，移住トポロジとしてリング構造等をとる．例えばリング状に島が並び，各島では独立して進化が進む．ある程度進化が進むと，隣接するノード間で個体の交換を行う．

3. DNAS システムの実装

DNAS システムの実装の方法について説明する．DNAS システムは階層的な構造をもち，かつ動的に変化するようなシステムである．DNAS システムでは，初期的な構造を構築する方法と，その構造を有効に維持するための手法が必要となる．

3.1 トポロジの作成

本システムは情報を一つの計算機ノードから別のノードに伝達する手法を実装している．ノードはツリー状になっており，ツリーの根をマスターノードと呼ぶ．接続しているノードのうち，マスターノードにより近いノードを「uplink」と呼ぶ．マスターノードからより遠くにあるノードを「downlink」と呼ぶ．本システムでは,「TAG」と「DATA」のペアがマスタノードへ向けて転送される．このようにしておくことで，データ形式の拡張性が維持できる．各 downlink は uplink へ，メッセージを一定時間 (例：15 秒) に一回送信するように設定される．各ノードのタイミングは非同期的である．

データは，簡単に扱うために平文で送られる．データは LDIF[6] に似た形式で，コロンで区切られた TAG と DATA のペアで構築されている．各ノードの通信データは，最初にホスト名を示す「:hostname:」という一行で始まる．それに続いて「TAG: DATA」の形でデータと TAG のペアが必要なだけ繰り返される．このデータ単位を情報パケットと呼ぶ．形式を**図 2** に示す．

3.2 DNAS トポロジ維持の仕組み

構造を維持するための仕組みについて説明する．Route-To，Seen-By，そして Data-Seen という TAG を定義する．

```
Info-packet :- meta-header each-host-info*
each-host-info: host-name host-info
host-name:- ":hostname:"
host-info:- tag-data*
tag-data :- "tag: data"
```

図 2 Information packet

```
while link (car(Route-To)) fails do
  Route-To=cdr(Route-To)
```

図 3 Algorithm to relink to uplink, when uplink is lost

Route-To 情報は，uplink から downlink に伝達される．uplink から master までの経路にある全ノードの名前の一覧となっている．各ノードは，downlink に対して，uplink から取得した Route-To 情報に自分のホスト名を追加したものを伝達する．

Seen-By は downlink から uplink に対して送信される情報で，あるパケットを過去見たことのあるホスト名のリストになっている．各ノードは自分自身のホスト名をパケットの Seen-By に追加する．

Data-Seen はある情報パケットが uplink に対して新しいパケットを送信する必要があるものなのか，それとも古いパケットで再送する必要が無いものなのかということを判断するための情報である．

以上の情報を利用すると，接続トポロジを動的に変化させることができる．この目的を達成するために二つの基本動作が定義される必要がある．そのアルゴリズムを次に提示する．

3.3 再接続のアルゴリズム

uplink の機能が停止しているとき，DNAS ノードは Route-To 情報を参考にして uplink の uplink がどれであるかということを発見しようとする．この相手をみつけて直接そのノードを uplink として利用することにより，ノードが uplink を失っている状態から脱出しようとする．さらに uplink の uplink にも障害がある場合，そのさらに uplink に接続を要求をする．この処理は再帰的に行われる．最終的にはマスターノードに到達する．仮にマスターに障害があり接続できない場合には，そこで停止する．このアルゴリズムは**図 3** に示したようになる．このアルゴリズムでは各ノードの同期をとる必要がない．

```
D=list of active downlinks
A=D[random]
B=D[random]
if A!=B then
  set Route-To[A]=B+Route-To[A]
```

図 4 Algorithm to relink on too many downlinks

3.4 ノードに接続している downlink が多すぎる場合の再接続アルゴリズム

多数のノードがあるノードに直接通信するとそのノードにておいての処理負荷が高くなる．既定の上限より多くの downlink が DNAS に直接接続している場合，DNAS は，downlink を再構成する．ノードは自分の downlink ノードを任意に選択し，他の downlink の downlink となるように再配置する．

ここで利用するアルゴリズムを**図 4** に示す．ダウンリンクのリストを D とする．D から選出したランダムなノードを A とする，また別のランダムに選択したノードを B とする．A に対しての Route-To の先頭に B を付加する．次に A は図 3 に示すアルゴリズムを用いて接続しようとするときに B を上流として認識する．この操作はループが出来る可能性があるので，それぞれの DNAS ノードにおいて同時に downlink 一台にしか適用しない．

4. 実　　験

4.1 データの転送のためのシステム

今回提案したシステムに，各ノード間でデータを転送し，お互いに情報を交換するために必要な API を準備した．libdnas-application ライブラリが DNAS のアプリケーション API で `DNAS_gatherinfo(TAG)` や `DNAS_sendinfo(TAG,DATA)` 命令等を提供する．これらの命令は目的によりアプリケーションが決定した TAG の文字列により識別できるデータを近隣の DNAS アプリケーションと共有するものである．`DNAS_sendinfo` 命令は TAG と DATA パラメータを与えると，自分のノード上の DNAS にデータを送信する．その情報は uplink へリレーされる．`DNAS_gatherinfo` 命令は uplink の DNAS が持っている TAG とホスト名で識別される情報を取得し，TAG が一致するものを抽出する．結果として他ノードが TAG を指定して `DNAS_sendinfo` で送信したデータを取得できる．このシステムを利用すれば，データの交換を行うアプリケーションを実装できる．また，それぞれのアプリケーションに別の TAG を割り当てれば，互いに独立にデータを送受信できる．

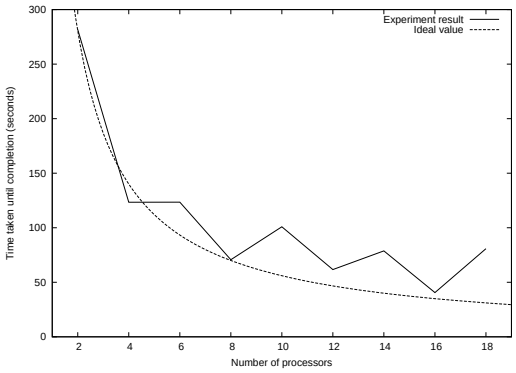

図 5 Random search time with 1 to 19 nodes. Time it takes until the system stops after finding the solution. Median of 20 trials.

4.2 ランダムサーチの実装

階層システム上で動作するランダムサーチアプリケーションを実装した．28 ビットの二進数の one-max 問題を解く．one-max 問題とは，二進数において全ビットが 1 となる解を持つ問題である．

DNAS システムの通信は 3 秒に 1 回行うようにした．乱数で問題空間の検索を行う．10000 試行ごとに進行状況をモニタするために，ランダムサーチプロセスはそのノード上の DNAS プロセスにその時点までで一番良かった解の情報を送信する．DNAS プロセスはその情報を上流ノードにリレーする．ランダムサーチプロセスは uplink の DNAS プロセスにあるデータを見る．そこには他ノードからの情報があり，最適解が他のノードで発見されたことが分かれば，自分も最適解を発見したことにする．最適解を発見したら，DNAS プロセスを介して他ノードにそれを伝達し，自分は終了する．システム全体の終了条件としては，全ノードにおいて計算が終了することとした．

実験を 1 台から 18 台までで行い，実行時間を計測した．**図 5** に結果を示す．結果をみると，台数が増えると早く解が見付かっているのがわかる．

あるノードで計算が終了しても，その結果が伝搬するのに時間がかかるために，全ノードが終了するためには，多少時間がかかる．今回の DNAS の設定では，3 秒サイクルで情報を伝達しているため，早い時点で解がみつかってもノード全体にその情報が伝わるために時間が必要であり，ノード数によるオーバヘッドが存在する．

ランダムサーチの実験に利用した PC クラスタは Gregor (**表 1**) である．

表 1　PC Cluster used in the experiment

Name	CPU	Network
Gregor	Pentium III 1GHz $x64$	100BASE-TX

4.3　分散遺伝的アルゴリズムの実装

今回のシステム上に GA を実装した．今回のモデルでは，各ノードがそれぞれ島となり，島内部での GA 操作を行う．さらに DNAS servent を介したノード間のメッセージ通信を利用して島間で個体の移住を実装する．今回の実装では各島には 100 個体が存在することとした．全個体に対して遺伝的操作を行って新しい世代の個体を作る．ここまでの処理を一世代と呼ぶ．あるノードで実行している GA プロセスに注目する．一定世代が過ぎたあと，DNAS servent には GA プロセスに存在する個体の情報を送信する．DNAS servent は uplink にその情報を一定間隔でリレーする．uplink の DNAS servent は，そのノードから来た情報をプールする．また，GA プロセスは自分の uplink の DNAS servent に存在するノードにプールされている個体情報を取得するための要求を送信し，個体情報を取得する．ここで取得するのは，他のノードの GA プロセスにある個体の中からランダムに選択して送信した個体の遺伝子の情報である．これらの個体を各ノードの GA プロセスは自分の個体群のなかに入れる．例えば，10 個体を取得したら，現在ある個体の中からランダムに選択した 10 個体を破棄して，その代わりに新規に取得した 10 個体を配置する．ノードが個体を uplink に転送し，上流リンクからデータを取得するという処理は各ノードが非同期的に行う．初期状態では個体情報が uplink にはプールされていないので，個体は取得できない．また，各ノードから送信された個体情報をある一定の TAG(“dga-randominfo”) で uplink は持ち，そのノードの情報を上書きすることにする．これにより DNAS servent によって個体数が際限無くプールされて増えることは無い．DNAS servent は事前に設定したタイミングでプールしているデータの交換を行う．例えば 3 秒に一回ノード間で情報を交換する．通信しあうノードが構成する階層が重層的に存在して，各ノードでランダムな個体を交換することになる．**図 6** に各 GA プロセスの作業の流れを示す．**図 7** に全体の流れを示す．

各ノードは，ノード上で実行する GA の島内に存在するエリート個体の情報を uplink に伝達する．これは，探索がどれくらい進んでいるか，という状況をモニタするためである．また，各ノードからランダムに

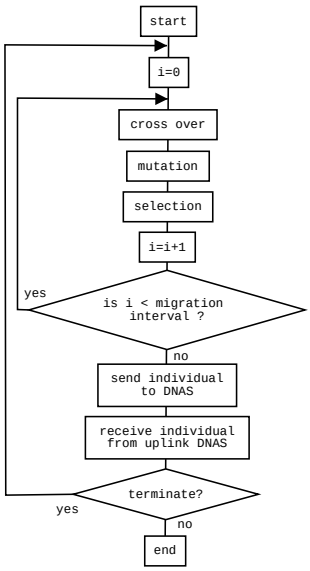

図 6　DGA Model, GA process side logic

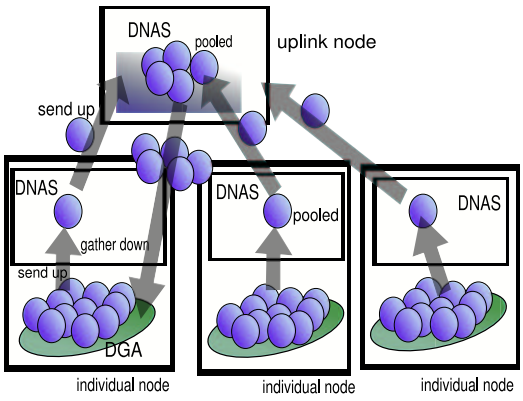

図 7　Implementation of DGA

個体を取得し，それを uplink に伝達する．この操作により，uplink には，自分から送信したランダムな個体と，他のノードから送信されたランダムな個体の情報がある．この情報を取得して，島上の個体の集合に追加し，探索に参加させる．それにより複数の島においての計算結果を統合して協調して探索を行うことが可能になると考えられる．

4.4　島モデルの島数と最適解の精度の変化に関する実験

ここで利用した対象問題は 5 次元の Rastrigin 関数の最大値を求めるものである．各変数は 32bit の精度の固定小数点を利用し，変数 x_i の範囲は $[-5.12, 5.12]$. であった．利用した関数は式 1 に示すものであった．

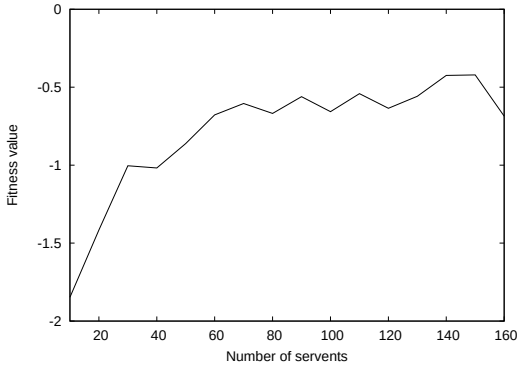

図 8 Fitness value after 2000 generation running with 10 to 160 nodes.

$$f = -\left(10n + \sum_{1}^{n} x_i^2 - 10cos(2\pi x_i)\right) \quad (1)$$

一島あたり 100 個体を利用している．エリートは 1 個体，染色体長は 160 ビット，選択はトーナメント選択，交叉は一点交叉，突然変移率は 0.2%で計算した．全個体数は島数 ×100 である．島数を増やして計算量を増大させると，一定時間の間に比較的よりよい解に到達できているようである．計算量を増加させれば単位時間でより良い解に到達している．

図 8 には 2000 世代が過ぎたあとの状態で，島数が 10 から 160 の間のそれぞれの場合の最良であった島の値を示す．結果は 20 回試行の平均である．ここでのトポロジは図 3 や図 4 に示す再接続ルールを用いて再構築した結果生成されたトポロジである．実験の最初の時点ではクライアントサーバ形式のトポロジから始まり，その段階が少しづつ変化する．各ノードは 4 台までの子ノードを持つという制約にする．台数が多いと性能は向上するが，多すぎても良くはなっていないという結果になった．そのため，DGA を DNAS 上に実装する際にはノード数に対するスケーラビリティを有するモデルの検討が必要である．

4.5 移住間隔の変化による影響

移住間隔を変化させた際の探索性能の変化を検討する．3000 世代目まで探索して計算の途中で停止した結果が**図 9** である．この実験においては，4 ノードが使用されている．結果を見ると，移住間隔が密であった方が早く解に向かっているという傾向がわかる．

5. 結　　論

本研究では動的に通信トポロジを変化して階層的構造を構成する DNAS システムの上のアプリケーションとして，分散遺伝的アルゴリズムを実装した．実装

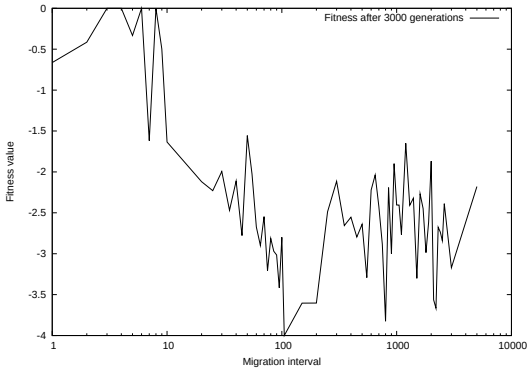

図 9 Fitness value after 3000 generations on 5-node set up, with 4 nodes representing islands of 100 populations each. Comparison is done based on different migration rates. Average of 5 tries.

したアプリケーションが実際に動作し，複数のノードが協調して動作すると，解の探索性能が向上するということが明らかとなった．従来の MPI 等を利用した計算システムと同じパラメータを利用していても良い性能が出ないため，調整が必要である．

今後は提案システムをハイパークラスタおよび Grid 環境に適用させ，アプリケーションの検討を行う．

謝辞　本研究は文部科学省学術フロンティア推進事業により支援されている．

参 考 文 献

1) Goldberg, D. E.: *Genetic algorithms in search, optimization and machine learning*, Addison Wesley, Reading, Massachusetts (1989).
2) Tanese, R.: Distributed Genetic Algorithms, *Proc. 3rd International Conference on Genetic Algorithms*, pp. 434–439 (1989).
3) SETI@home: search for extraterrestrial intelligence at home, http://setiathome.ssl.berkeley.edu/.
4) Folding@home: Folding@home Distributed Computing, http://folding.stanford.edu/.
5) Rodriguez, P., Spanner, C. and Biersack, E. W.: Analysis of Web Caching Architectures: Hierarchical and Distributed Caching, *IEEE/ACM Transactions on Networking*, Vol.9, No. 4, pp. 404–418 (2001).
6) Good, G.: *The LDAP Data Interchange Format (LDIF) - Technical Specification*, http://www.faqs.org/rfcs/rfc2849.html.