# 多資源計算環境下での遺伝的アルゴリズムのための ローカルサーチメカニズムを有するデータベースの提案

花田良子[†] 廣安知之[††] 三木光範[††]

近年，膨大な計算ノードを有する PC クラスタやグリッドを利用することにより，多資源環境下で遺伝的アルゴリズム（GA）を実行することが可能となってきた．大規模計算環境が利用可能な場合には GA に膨大な計算リソースを有効に扱え，効率良く探索できるメカニズムが必要となる．本研究では，計算資源の増加に対して探索性能のスケーラビリティを保証する 1 つのアプローチとして，GA のためのローカルサーチメカニズムを有するデータベースを提案する．提案するデータベースでは既探索領域の情報を保存しながら，未探索領域を重点的に探索し既探索領域の拡張を行うローカルサーチを行う．データベースを GA に組み込むことにより，既探索領域の大きさを定量的に把握することが可能である．また，計算資源あるいは計算コストを増加させれば必ず既探索領域が拡張し，有限の計算コストの下で全探索を保証して探索を終了することが可能となる．提案するデータベースを原始的なビットの問題，および連続最適化問題のテスト問題に適用した結果，計算コストを制限しない場合，全探索により得られた解が最適解であることを保証すること，計算コストを制限した場合においても従来の GA と同等の結果を得られることを示した．

# The Database with Local Search Mechanism for Genetic Algorithms under Large-scale Computing Environments

Yoshiko Hanada,[†] Tomoyuki Hiroyasu[††] and Mitsunori Miki[††]

Recently, GA that uses large-scale computer systems comprised of massive processors has become feasible because of the emergence of super PC clusters and Grid computation environments. Mechanisms to use massive computation resources laconically and to search effectively are necessary if large-scale computer systems are available. In this study, a new GA-specific database with the local search mechanism to assure the scalability of search performances against the number of computing resources is proposed. Our proposed database possesses information of space that has been already searched with limited number of individuals. At the same time the local search for the space that is not searched is applied using individuals stored in the database. To embed such mechanisms in a GA enables us to comprehend the quantitative rate of a searched region during the search. Using this information, the searched space can be expanded linearly as the number of computing resources increases. Moreover GA can be terminated under guarantee of the exhaustive search. First, our database was applied to primitive functions and shown to ensure an effective exhaustive search. This method was then applied to the test functions of continuous optimization problems under restricted computing costs. Through such experiments, it was clear that this method has the same performance as a conventional GA.

## 1. はじめに

近年，対象問題の大規模複雑化にともない，進化的戦略を用いた確率的な最適化手法が注目を集めており，その代表的なものに生物の進化を模倣した遺伝的アルゴリズム（Genetic Algorithms: GAs）[1] がある．GA は様々な問題に適用可能であり，複雑な問題に対しても有効な最適化手法であるとされている．一方で，早熟収束による局所解への収束，局所探索が不得手，高い計算負荷という問題が存在する．探索の問題を解決する方法として，Minimal Generation Gap（MGG）[2] などの世代交代モデルの利用，Linkage の同定とその情報を利用した探索手法[3),4)]，実数値 GA [5]，確率モデル GA [6),7)]，分散 GA [8] など様々な手法が提案されている．また，再初期化など複数回試行[9)~12)] による探索の強化などが図られている．計算負荷の問題につ

† 同志社大学大学院
Graduate School of Engineering, Doshisha University
†† 同志社大学工学部
Department of Engineering, Doshisha University

いては，一般的に GA は多点探索であることから並列処理化が容易であるため，並列計算機に実装を行うことで解決されている．

GA の並列化は高い計算負荷の問題の解決にとどまらず，計算規模を拡大することで解探索性能の向上にも用いられる．GA は並列化との親和性に加え，処理中の探索点情報が失われても対処できるという特徴を有するため，これまでに多資源計算環境におけるアプリケーションが開発され，成果をあげている[13)~16)]．GA をとりまく計算環境は，従来は限られた少数の資源であったものが，近年では膨大なノードを有する PC クラスタ，大規模な Grid などを利用することで膨大な資源が使用可能となってきている．しかしながら，膨大な資源を有する計算環境に対応することを考えた場合，次章で示すように，使用可能な資源数に対する探索性能のスケーラビリティが得られないといった問題が存在する．その 1 つの原因は探索の重複である．本研究では大規模計算環境が利用可能な場合には GA に膨大な計算資源を有効に扱え，効率良く探索できるメカニズムとして，探索の重複を回避するための既探索領域を記憶するデータベースを提案する．本研究の目標は，1) 膨大な計算環境での探索性能のスケーラビリティを考慮し，かつ限定された計算コストの中でも探索能力が保持できるメカニズムとしての既探索領域データベースの考案，2) 既探索領域データベースを利用したタブ・サーチ，未探索領域への再初期化の導入，3) 大規模環境での性能検証，の 3 点である．本論文では，スケーラビリティを「計算資源の増加に対する既探索領域の増加」と定義し，1) について議論する．

本論文では全既探索個体を限られたサイズで格納するためのデータベースでの個体表現，およびデータベース内の既探索個体の圧縮のメカニズムとしてのローカルサーチを提案する．提案するローカルサーチでは，既探索領域の情報を保存しながら，未探索領域を重点的に探索し既探索領域の拡張を行う．GA にこのようなローカルサーチメカニズムを有するデータベースを適用することにより，計算資源を増加させれば必ず既探索領域が拡張し，有限の計算コストの下で全探索を保証して探索を終了することが可能となる．

## 2. 多資源計算環境における GA の問題点

GA は並列処理との親和性が高い一方で，多資源計算環境に対応することを考えた場合，使用可能な資源数に対する探索性能のスケーラビリティが得られないという問題が存在する．大規模な計算環境が利用可能な場合には，その計算資源を利用する方法として，最

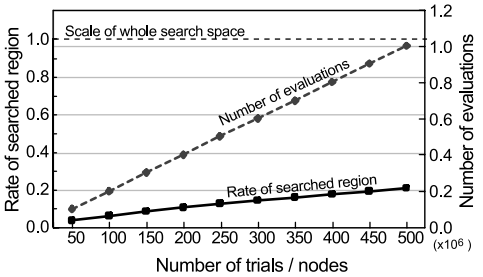



**図 1** 計算ノード数に対する既探索領域の増加
Fig. 1 Increases in searched region against the number of evaluations.

も単純なアプローチは母集団の個体数を増加させ並列度を向上させることである．しかしながら，一般に GA は個体を極度に増加させることで多様性の極端な増大により収束が遅くなるため，時間一定で比較した場合，個体数の増加が必ずしも解探索性能の向上につながらないといったデメリットを有する．一方で，適度な母集団サイズの複数試行を並行して実行するといった手法が考えられる．しかしながら，GA は少ない計算量で探索の早い段階で満足解が得られる解探索能力が高いアルゴリズムであるが，試行間での探索の重複，母集団の収束による母集団内での探索の重複が生じること，また，再初期化による複数回試行を行った場合には同じ解に収束する可能性があるといった問題がある．

**図 1** は 2 次元の Rastrigin 関数において，GA の試行数を増加させたときの既探索領域の大きさの割合の推移を示したものである．図中，横軸は試行数（計算ノード数），左縦軸は問題空間の大きさに対する既探索領域の割合，右縦軸は総評価計算回数を示している．本実験では，1 変数を 10 ビットで表しており，探索空間の大きさは $2^{20}$（$= 1.05 \times 10^6$）である．なお，世代交代モデルを Elitest Recombination（ER）[17)] 母集団サイズを 10，交叉には一様交叉を用い，1 回の交叉における生成子個体数を 20，突然変異率を 0.05（$= 1/L$：染色体長）とした．また，探索終了世代を 20 世代とし，1 試行につき 2,010 回の評価計算が行われる．500 試行で $1.0 \times 10^6$ 回の評価計算となり，ほぼ全探索に相当する評価計算が行われる．

図 1 から，試行数の増加に対し，総評価計算回数の増加と比較して既探索領域の割合が増加しないことが分かる．また，全探索に相当する評価計算を行ったとしても 20%程度しか探索ができておらず，探索に無駄が生じている．このような探索の重複の問題により，大規模計算環境への GA の適用を考えた場合，利用可能な計算資源が増加したとしても，有効に扱うことが

困難であると考えられる．本研究では，探索の重複を回避する解決策として，既探索領域を記憶する GA に特化したデータベースを提案する．

## 3. 既探索領域を保存するデータベース

個体データベースにおいて，1 個体の情報を 1 つの個体データとして扱った場合，膨大なデータ量が必要であり，個体の参照などで膨大な処理時間を浪費することとなるため，すべての個体を保存することは困難である．そこで，全既探索個体を限られたサイズで格納するためのデータベースの個体表現，および圧縮のメカニズムが必要である．本章ではデータベースの個体表現を説明する．

### 3.1 既探索領域の表現

全既探索個体を限られたサイズで格納するためのデータベースの個体表現について説明する．本研究では個体の遺伝子表現に $\{0,1\}$ を用いる GA を対象としており，データベースに保存される個体は $\{0,1\}$ のビット列で表現される．データベースの個体は，染色体に加え，既探索領域を示すマスクを持つ．マスクは染色体長と同じ長さのビット列である．染色体においてマスクの “1” の部分に対応する遺伝子座はすべてのパターンを探索したことを表す．**図 2** に，マスクを用いた個体の例を示す．提案するデータベースではこのようにマスクを有する個体を格納する．

**図 3** にデータベースの例を示す．図 3 において，マスク “0 1 1 0 1” の 5 ビットの個体 $x_3$ = “0 1 1 1 1” は，個体群 $X_3 = \{0 * * 1 *\}$ ($* = 0$ or 1) をすべて探索し終えたうえで最も評価値が高い解が “0 1 1 1 1” であることを示す．マスクの “1” の個数をハミング指標とよぶ．この個体のハミング指標は 3 である．

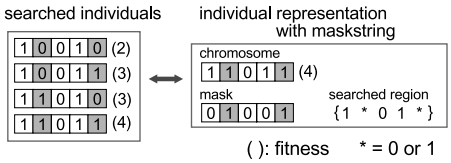



**図 2** データベースに保存される個体
Fig. 2 An example of the individual stored on the database.

データベースで 1 個体の情報を 1 つの個体データとして扱った場合，膨大な量のデータを格納することとなり，個体が探索されているかどうかデータベースに参照する場合などに多大な時間を要する．提案するデータベースではマスクを用いることにより既探索個体の情報を限られた個体で表現しているため，データベースの参照などの操作で処理時間を費やすことはない．

### 3.2 既探索領域の大きさの定量的な把握

提案するデータベースでは，既探索領域の大きさを定量的に示すことが可能である．染色体長を $L$，個体 $x$ の染色体を $(c_{x1}, c_{x2}, \cdots, c_{xL})$，マスクを $(m_{x1}, m_{x2}, \cdots, m_{xL})$ と表現する．$c_{xl}$，$m_{xl}$ の $l$ $(1 \leq l \leq L)$ を染色体，およびマスクそれぞれの遺伝子座とする．

#### 1 個体の既探索領域の大きさ

個体 $x$ の既探索領域 $X$ の大きさを $|X|$ と表記する．$|X|$ は集合 $X$ の要素数を示す．ハミング指標値が $h$ の個体の既探索領域のサイズは $2^h$ となる．たとえば，図 3 に示した $x_3$ について，$x_3$ の既探索領域 $X_3 = \{0 * * 1 *\}$ の大きさは要素数 $|X_3| = 2^3$ 個である．

#### $N$ 個体の既探索領域の大きさ

個体 $x_i$ $(1 \leq i \leq N)$ の既探索領域を $X_i$ とすると，$N$ 個体の既探索領域は和集合 $X_1 \cup X_2 \cup \cdots \cup X_N$ である．添え字 $i$ の集合を $I = \{1, 2, \cdots, N\}$ とすると，$N$ 個体の既探索領域の大きさは $\left|\bigcup_{i \in I} X_i\right|$ と記述でき，式 (1) で求めることができる．

$$\left|\bigcup_{i \in I} X_i\right| = \sum_{J \subseteq I, J \neq \phi} (-1)^{|J|-1} \left|\bigcap_{j \in J} X_j\right| \quad (1)$$

一般に，対象集合族の和集合の個数関数は求めにくい場合が多く，積集合の個数関数は具体的な閉じた式

| | Chromosome | Mask | Searched Region | Fitness | Hamming-Index |
|---|---|---|---|---|---|
| Individual $X_1$ | 1 0 1 1 1 | 1 0 0 0 1 | * 0 1 1 * | 4 | 2 |
| Individual $X_2$ | 1 0 0 1 1 | 0 0 0 1 1 | 1 0 0 * * | 3 | 2 |
| Individual $X_3$ | 0 1 1 1 1 | 0 1 1 0 1 | 0 * * 1 * | 4 | 3 |
| Individual $X_4$ | 1 1 0 0 1 | 0 0 0 0 0 | 1 1 0 0 1 | 3 | 0 |

* = 0 or 1

**図 3** データベースの例
Fig. 3 An example of the proposed database.

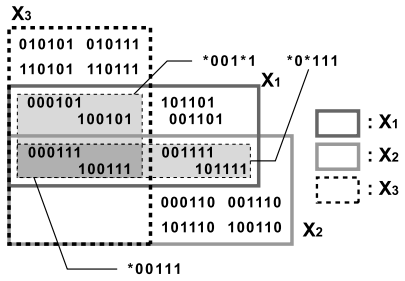



**図 4** 3 個体の和集合
Fig. 4 The union set of three individuals.

となる場合が多い．個体の既探索領域もこれにあたり，$N$ 個体の既探索領域の和集合の要素数が一定の規則で求められないのに対し，積集合の要素数は次に示す式 (3) で求めることができる．このことから，$N$ 個体の既探索領域を直接，和集合の要素数を求めるのではなく，式 (1) で示したように積集合から求める．

2 個体 $x_i$，$x_j$ についてマスクを考慮した距離 $d(x_i, x_j)$ を式 (2) のように定義する．

$$d(x_i, x_j) = \sum_{l=1}^{L} B|c_{x_i l} - c_{x_j l}|$$
$$B = \begin{cases} 1, & \text{if } m_{x_i l} = m_{x_j l} = 0 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

$N$ 個体の既探索領域の積集合の要素数は式 (3) のように求めることができる．なお，$[R]$ は，$R$ を実数，$z$ を整数としたとき，$z \le R < z+1$ ならば，$[R] = z$ を意味する．

$$|X_1 \cap X_2 \cap \cdots \cap X_N| = \begin{cases} 2^M, & \text{if } K = 0 \\ 0, & \text{otherwise} \end{cases}$$

$$K = \sum_{i=1}^{N-1} \sum_{j=(i+1)}^{N} d(x_i, x_j) \quad (3)$$

$$M = \sum_{l=1}^{L} \left[ \frac{1}{N} \sum_{i=1}^{N} m_{x_i l} \right]$$

以下に 3 個体 $x_1$，$x_2$ および $x_3$ の既探索領域の大きさ $|X_1 \cup X_2 \cup X_3|$ を求めた例，および**図 4** に $X_1 \cup X_2 \cup X_3$ を図示したものを示す．

**和集合の例**

- 各個体の要素数
  $X_1 = \{* \ 0 \ * \ 1 \ * \ 1\}, \quad |X_1| = 2^3$
  $X_2 = \{* \ 0 \ * \ 1 \ 1 \ *\}, \quad |X_2| = 2^3$
  $X_3 = \{* \ * \ 0 \ 1 \ * \ 1\}, \quad |X_3| = 2^3$
- 2 個体の積集合およびその要素数
  $X_1 \cap X_2 = \{* \ 0 \ * \ 1 \ 1 \ 1\}, \quad |X_1 \cap X_2| = 2^2$
  $X_2 \cap X_3 = \{* \ 0 \ 0 \ 1 \ 1 \ 1\}, \quad |X_2 \cap X_3| = 2^1$
  $X_3 \cap X_1 = \{* \ 0 \ 0 \ 1 \ * \ 1\}, \quad |X_3 \cap X_1| = 2^2$
- 3 個体の積集合およびその要素数
  $X_1 \cap X_2 \cap X_3 = \{* \ 0 \ 0 \ 1 \ 1 \ 1\}$
  $|X_1 \cap X_2 \cap X_3| = 2^1$
- 3 個体の和集合の要素数
  $|X_1 \cup X_2 \cup X_3| = |X_1| + |X_2| + |X_3| - |X_1 \cap X_2| - |X_2 \cap X_3| - |X_3 \cap X_1| + |X_1 \cap X_2 \cap X_3|$
  $= 24 - 10 + 2 = 16$

このように，本データベースではマスクを用いた個体表現を用いることで，探索中に得られた解がどの程度探索した結果であるか把握することを可能にしている．

## 4. データベースでの諸操作

データベースは GA で得られた良好な個体を前章で示したような個体表現で格納する．なお，新たに保存される個体はすべてのビットが “0” のマスク，ハミング指標が 0 の状態で保存されている．データベースサイズをある程度の大きさに限った場合，GA の探索の過程で発見された新たな個体を保存するためには，データベースの許容量を超えないようデータベース内の個体をマージする必要がある．このとき，既探索領域の情報を損なわないよう複数の個体情報を 1 つの個体情報としてマージするため，データベースの個体に対してローカルサーチを適用する．GA で得られた良好な解の近傍を集中的に探索することにより，解探索性能の向上も期待できる．

### 4.1 ローカルサーチ

提案するローカルサーチでは，ある 1 個体に対して，マスクの “0” を “1” にする操作を行う．対象個体にはデータベースの保存個体でハミング指標値が最小の個体を選ぶ．本研究におけるローカルサーチの戦略は，1) ハミング指標の小さい個体，2) データベース内の個体の各ビットのうち，分散値が最も大きなものから探索を行う．これは，次節で説明するデータベースの個体のマージの際に，ハミング指標ができるだけ等しいものが望ましいこと，分散値が小さなビットは良い適合度を与える高い要因であると考えられることなどの理由に起因している．

以下にローカルサーチの手順を示す．データベースに保存されている個体数を $N$，データベースの個体を $x_i$ $(1 \le i \le N)$ とする．

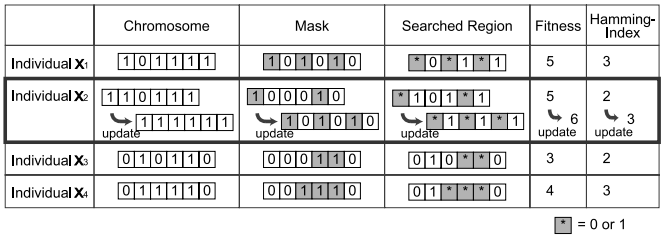



**図 5** 1max 問題におけるローカルサーチの例
Fig. 5 An example of the local search on 1-max problem.

**Step1.** 式 (4) に示すように個体 $x_i$ $(1 \leq i \leq N)$ の各遺伝子座 $l$ $(1 \leq l \leq L)$ の遺伝子の平均と 0.5 との差の絶対値 $a_l$ を求める．

$$a_l = \left| \frac{1}{N} \sum_{i=1}^{N} (c_{x_i l}) - 0.5 \right| \quad (1 \leq l \leq L) \quad (4)$$

**Step2.** $N$ 個の個体で，適合度が最大，かつハミング指標値が最小の個体 $x$ を探索点とする．

**Step3.** $m_{xl} = 0$ を満たす遺伝子座の中で，$a_l$ が最小となるような遺伝子座 $l^*$ を求める．

**Step4.** 探索点 $x$ とマスクが同一で，$c_{xl^*}$ のビットのみを反転した個体群 $X'$ を探索する．$x$ のハミング指標値を $h_x$ とすると，新たに探索する個体数は $2^{h_x}$ である．

**Step5.** $x$ と個体群 $X'$ の中で最も適合度が高い解 $x'$ のどちらか適合度の高い個体を $x$ と置き換える．$m_{xl^*} = 1$，$h_x = h_x + 1$ に更新する．もとの個体よりも適合度の高い解が得られた場合，その個体を探索アーカイブの最悪個体に上書きコピーする．$h_x$ が $L$ に達した場合，全探索が完了する．

上記の Step4 における個体群 $X'$ の探索は任意の計算ノード数の並列化が可能であり，大規模計算環境における遊休資源を利用することにより効率的な資源の使用が可能である．また，各計算ノード間で依存することなく領域探索が可能なため，通信量が少なく高スループットな計算が期待できる．計算資源あるいはコストをかけるほど既探索領域が拡大するため，探索を進めていくことで必ず最適解が得られる保証を有する．

**図 5** にローカルサーチの適用例を示す．ここでは，既探索領域を拡大する対象個体 $x_2$ において，遺伝子座 $l^* = 3$ について $X_2 = \{*\,1\,\underline{0}\,1\,*\,1\}$ の $c_{x_2 l^*}$ を反転させた個体群 $X_2' = \{*\,1\,\underline{1}\,1\,*\,1\}$ を探索し，$c_{x_2 l^*}$ を 1 に変更している．拡大後の既探索領域は $\{*\,1\,*\,1\,*\,1\}$ となる．

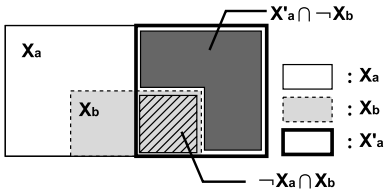



**図 6** Condition 3 におけるマージ
Fig. 6 The merge process with the Condition 3.

### 4.2 データベースにおけるマージ

ローカルサーチ後に，データベースにおいてマージが可能な個体の組合せがあればマージを適用する．マージが可能な組合せとしては次のようなものが考えられる．

**Condition1.** 個体 $x_a$，$x_b$ のマスクが同一で，式 (2) で定義した距離が $d(x_a, x_b) = 1$ を満たす場合，$m_{x_a l} = m_{x_b l} = 0$ で，$c_{x_a l} \neq c_{x_b l}$ となる遺伝子座を $l^*$ とすると，$x_a$，$x_b$ の適合度の高い個体のマスクの遺伝子座 $l^*$ のビットを “1” にし，もう一方の個体を削除する．図 5 において，個体 $x_1$ と更新後の個体 $x_2$ は Condition 1 でマージすることができる．マージを適用すると，更新後の個体 $x_2$ のマスクの左から 2 ビット目 $m_{x_2 2}$ を “1”，ハミング指標 $h_{x_2}$ を 4 に更新し，個体 $x_1$ をデータベースから削除する．

**Condition2.** 個体 $x_a$，$x_b$ について $m_{x_a l} = 1$，$m_{x_b l} = 0$ $(1 \leq l \leq L)$ となる遺伝子座が存在せず，$d(x_a, x_b) = 0$ となる場合，個体 $x_b$ の既探索領域 $X_b$ が個体 $x_a$ の既探索領域 $X_a$ を包含しているので個体 $x_a$ を削除する．

図 5 において，個体 $x_3$ と個体 $x_4$ は Condition 2 でマージすることができる．マージを適用すると，個体 $x_3$ が削除される．

**Condition3.** $X_a \cap X_b \neq \phi$ のような個体 $x_a$，$x_b$ が存在した場合には，この 2 個体を 1 個体の情報として**図 6** に示すようなマージを行う．どちらか一方の探索領域を Condition 2 のマージができるまで拡

張する操作を行い，他方を削除する．拡張する領域を $X'_a$ と表記すると，$X'_a \cap \neg X_b$ の領域が探索される．なお，重複領域の大きさ $|X_a \cap X_b|$ に対し，探索を行わなければならない領域の大きさ $|X'_a \cap \neg X_b|$ がある程度大きい場合はマージは行わない．$|X_a \cap X_b|$ と $|X'_a \cap \neg X_b|$ の比率はパラメータ $A$ とする．パラメータ $A$ は，探索空間の規模，許される計算コスト，計算時間などから妥当な値を与える必要がある．本論文における実験では $A = 8$ としている．

## 5. 数 値 実 験

提案したデータベースの有効性を検証する．ここではデータベースを GA に適用することにより，提案どおり既探索領域が表現されていること，計算資源に対する探索性能のスケーラビリティを保証することを示し，ローカルサーチを有するデータベースを組み込むことによって解探索能力が低下しないことを数値実験を通じて示す．本研究で提案しているデータベースと GA との利用方法については種々考えられるが，本論文では，GA で得られた良好な解をデータベースを保存し，それを用いてローカルサーチを行うといった簡単な適用法でデータベースの有効性を検討する．**図 7** にその概要を示す．この実装では，GA とローカルサーチが交互に適用されることになる．

まず，GA の母集団を初期化する．データベースは初期世代において個体数は 0 である．次に GA の母集団において，交叉，突然変異，選択などの遺伝的操作を行い，母集団の上位 1 個体をデータベースに保存する．ただし，すでにこの個体がデータベースで保存している既探索領域内に存在するときは格納しない．また，データベースの個体数があらかじめ定めた個体数に達しており，かつデータベースの最悪個体の適合度よりも良ければ，最悪個体と置き換える．データベースの最大許容個体数はパラメータとする．次にデータベースにおいて，ある 1 個体を対象としてローカルサーチを行う．ローカルサーチで対象個体よりも適合度の良いものが発見された場合，その個体のコピーを GA の母集団における最悪個体と置き換える．

上記のようにデータベースを GA に組み込み，計算コストを制限しない場合，および計算コストを制限した場合の解探索性能について検証する．対象問題として，原始的なビットの問題である 1max 問題，だまし問題の 1 つである 3-deceptive 問題 (5) [18]，連続関数のテスト問題の Rastrigin 関数 (6)，Schwefel 関数 (7)，Ridge 関数 (8)，および Griewank 関数 (9) を用いる．1max 問題は染色体中の 1 の個数が適合度である．1max 問題は代表的な GA のビットのテスト問題であり，これにより提案するデータベースの有効性を示す．3-deceptive 問題は，GA では探索が困難であり，データベースを有することで最適解を発見することが可能であることを示す．また，4 種の連続関数のテスト問題を通じて提案するデータベースの有効性を示す．

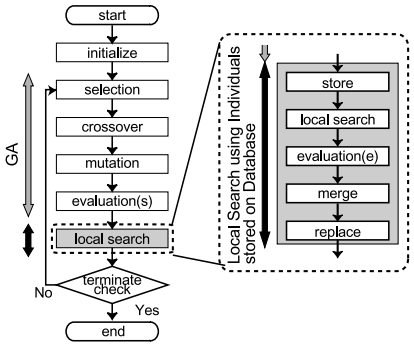



**図 7** 提案データベースを適用した GA の流れ
Fig. 7 The flow of GA using the proposed database.

$$F_{3\text{-}deceptive} = \sum_{i=1}^{N} f_i \tag{5}$$

$$f_i = \begin{cases} 0.9, & u_i = 0 \\ 0.8, & u_i = 1 \\ 0.7, & u_i = 2 \\ 1.0, & u_i = 3 \end{cases}$$

$$F_{Rastrigin} = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \tag{6}$$

$$x_i \in [-5.12, 5.12)$$

$$F_{Schwefel} = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{|x_i|}\right) \tag{7}$$

$$x_i \in [-5.12, 5.12)$$

$$F_{Ridge} = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \right)^2 \tag{8}$$

$$x_i \in [-64, 64)$$

$$F_{Griewank} = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \left( \cos\left(\frac{x_i}{\sqrt{i}}\right) \right) \tag{9}$$

$$x_i \in [-512, 512)$$

### 5.1 計算コストを制限しないときの探索性能

本研究で取り扱っている探索手法は GA であるので，比較的早い段階で準最適解，あるいは最適解を発見する可能性が高い．提案しているデータベースと組み合

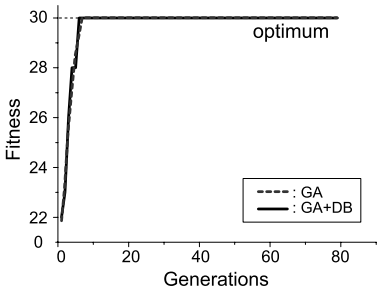



**図 8** 1max 問題における評価値の推移
Fig. 8 History of the fitness on 1-max problem.

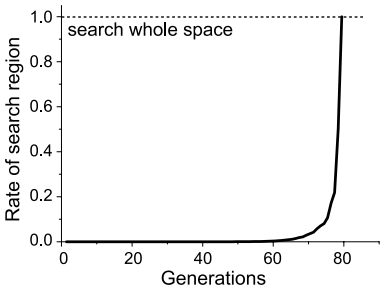



**図 9** 1max 問題における既探索領域の割合の推移
Fig. 9 History of the searched region rate on 1-max problem.

わせることで，その特徴に加えて，既探索領域を示し，かつ，GA で探索が困難な場合にも計算回数を増加させれば最適解を発見することが可能となる．本節では，30 ビットの 1max 問題，3-deceptive 問題を用いて，計算コストを制限しない場合のローカルサーチメカニズムを有するデータベースを組み込んだ GA の探索の特徴について検証する．全探索空間の大きさは $2^{30}$ であり，すべての解を評価し終えたときに探索が終了する．GA における世代交代モデルは ER モデルとした．交叉には一様交叉を用い，1 回の交叉における生成子個体数を 20 とし，突然変異率を 0.03 $(= 1/L)$ とした．母集団サイズを 20，データベースに保存できる個体数を 5 とした．

**図 8** および**図 9** に 1max 問題における適合度，および既探索領域の割合の推移を示す．これらの結果は 1 試行の例である．図 8 では，黒の実線が本データベースを GA に組み込んだ結果（GA+DB），グレーの点線が通常の GA の結果を示している．図 9 において，既探索領域の割合が 1.0 に達した場合，全探索が終了したことを意味する．

本手法は GA をベースとして探索を進めていくため，従来の GA と同等の解探索性能を持っていることが分かる．この問題では探索の序盤で最適解を得ているが，得られた解が最適解であるかについては，全探索することで保証している．また，図 9 に示すように，得られた解が全探索空間のうち，どの程度探索して得られた結果であるか，実行中に確認することが可能である．

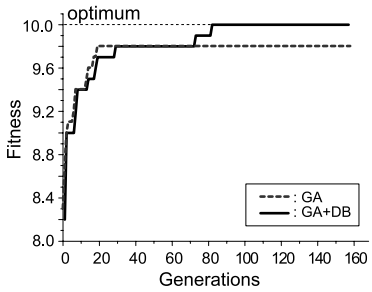



**図 10** 3-deceptive 問題における評価値の推移
Fig. 10 History of the fitness on 3-deceptive problem.

**図 10** に 3-deceptive 問題における適合度の推移を示す．3-deceptive 問題は GA の母集団が局所解へ収束しやすいよう設計された問題であり，GA で大域的最適解を得ることが難しい問題である．図 10 から，本手法は従来の GA と同様に探索初期で局所解に収束するが，探索を続けることによって最適解を得ていることが分かる．これはローカルサーチを有するデータベースを組み込むことによって計算資源あるいはコストをかけるほど既探索領域が拡大し，探索を進めていくことで必ず最適解が得られる保証を有するためである．

### 5.2 計算資源の増加に対するスケーラビリティ

提案するローカルサーチを有するデータベースの目的の 1 つは計算資源の増加に対して既探索領域をスケーラブルに増加させることである．そこで，既探索領域の増加のスケーラビリティを，本データベースを GA に導入することで検証する．提案するデータベースにおけるローカルサーチの並列化は**図 11** に示すような手法で探索領域を分割し，各計算ノードに割り当てることにより行われる．このとき，4.2 節に示したマージにおける Condition 1 の逆の操作を適用することが可能である．

本実験では，**表 1** に示す PC クラスタ環境において，1，2，4，8，16，32 ノードを用い，45 bit の 3-deceptive 問題を対象とし，1 時間の実行で得られた既探索領域の大きさを比較する．なお，この問題における全探索空間の大きさは $2^{45}$ である．GA における世代交代モデルには ER モデルを用いた．交叉には一様交叉を用い，1 回の交叉における生成子個体数を 20 とし，突然変異率を 0.02（$= 1/L$）とした．母集

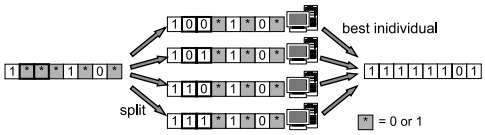



**図 11** 個体群の分割の例
Fig. 11 An example of splitting the individual.

**表 1** 実験環境
Table 1 Specification of machines used for the experiment.

| | |
|---|---|
| Processor | Dual Intel Xeon 2.4 GHz × 2 |
| Memory | 1 GB × 32 |
| OS | RedHat Linux 7.3 |
| Nodes | 32 nodes (64 processors) |
| Network | Myrinet2000 |

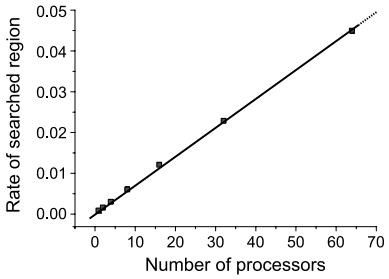



**図 12** 既探索領域のスケーラビリティ
Fig. 12 Scalability in searched region.

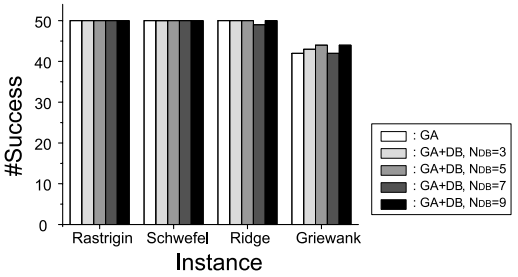



**図 13** 最適解を得た試行数
Fig. 13 Number of trials with the optimum.

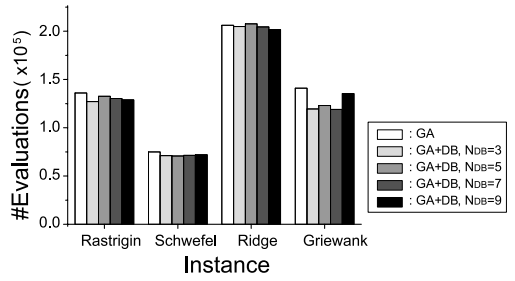



**図 14** 最適解を得た評価計算回数
Fig. 14 Number of evaluations for getting the optimum.

団サイズを 100，データベースに保存できる個体数を 10 とした．

**図 12** に既探索領域の割合を示す．これらは 5 試行の平均である．図 12 から，2 倍の計算ノードを利用することでほぼ 2 倍の既探索領域が得られており，提案どおり，既探索領域が計算資源の増加にともなって線形に増加していることが確認できる．

### 5.3 計算コストを制限したときの探索性能

本手法ではローカルサーチで多くの評価計算が必要である．計算資源あるいは計算コストが限られたときの懸念として，GA で十分探索ができず，従来の GA と比較して性能が低下するおそれがある．そこで，評価計算回数を制限したときの性能を検証する．

式 (6)，(7)，(8)，および (9) に示した 4 つの連続関数のテスト問題を用いて，GA およびデータベースを組み込んだ GA の解探索性能を比較する．いずれの問題も 10 次元の問題を用いている．最大の評価計算回数を $2.5 \times 10^5$ に制限した．GA における世代交代モデルには ER モデルを用いた．交叉には一様交叉を用い，1 回の交叉における生成子個体数を 20 とし，突然変異率を 0.01（$= 1/L$）とした．なお，染色体にはグレイコーディングによる $\{0, 1\}$ のビット列を用いている．母集団サイズを 200，データベースに保存できる個体数（$N_{DB}$）に 3，5，7，および 9 の 4 通りを用いた．本データベースでは，既探索領域を定量的に把握するために式 (1) で正確な大きさを求めている．この式では $N$ 個の領域の和集合を求める際に $2^N$ の組合せの積集合を求めており，データベースのサイズに対して指数的に増加するため，サイズが大きいと探索以外の計算で時間を浪費することになる．本論文では探索領域の大きさを定量的に示すため，また，サイズを小さくしても問題なく探索が可能であることを示すため，このような計算時間を圧迫しない程度の小さなサイズを用いている．なお，既探索領域の正確な大きさを必要とせず，既探索領域のみを把握するために提案するデータベースを用いる場合，データベースのサイズを小さな値に制限する必要はない．データベースのサイズを小さく限ることで既探索領域を幾分破棄する可能性があるが，本実装においては，初期段階で得られた解をもとに既探索領域を拡大するより，探索で得られた有望な解の付近を集中してローカルサーチを行うことで，より有効な解探索が行われると考えられる．

**図 13** および **図 14** に最適解を得た回数，最適解を得るのに必要とした評価計算回数の平均値を比較した結果を示す．これらの結果は 50 試行の結果である．

図 13 から，Rastrigin 関数，Schwefel 関数，Ridge 関数については，従来の GA，および本手法はすべての試行において最適解を得ており，Griewank 関数についてはデータベースを組み込む方が最適解を得た回数が多いことが分かる．また，図 14 より，データベースを組み込むことによって，ローカルサーチで多くの評価計算を必要とするにもかかわらず，わずかではあるが，従来の GA よりも最適解を得るまでに必要とした評価計算回数が少ないことが分かる．このことから，評価計算回数などの計算コストを制限した場合でも，本手法は良好な探索性能を示していることが分かる．

また，$N_{DB}$ が性能に与える影響については問題によって傾向が異なる．最適解を得た回数に着目すると，Ridge 関数については，データベースサイズが 5，7 のときと比較して 3，9 の方がわずかではあるが優れている．しかしながら，最適解を得るのに必要な評価計算回数はその逆となっている．一方，Schwefel 関数，Griewank 関数については，データベースサイズが 9 のときは 3，5，7 と比較して劣っているが，最適解を得た回数は多い．これより，計算コストを制限した場合，最小の評価計算回数で最適解を多く得る設定は存在しないが $N_{DB}$ に依存しない解探索ができているといえる．

## 6. おわりに

GA は少ない計算量で探索の早い段階で満足解が得られる解探索能力が高いアルゴリズムである．しかしながら，探索の重複といった問題が存在し，大規模計算環境への GA の適用を考えた場合，計算資源の増加に対する探索性能のスケーラビリティを保証することは困難である．本研究では，膨大な計算環境での探索性能のスケーラビリティを考慮し，かつ限定された計算コストの中でも探索能力が保持できるアプローチとして，GA のためのローカルサーチメカニズムを有するデータベースを提案した．探索の重複を回避するための GA に特化したデータベースでは，全既探索個体を限られたサイズで格納するため，その圧縮のメカニズムとしてローカルサーチを行う．ローカルサーチでは，既探索領域の情報を保存しながら，未探索領域を重点的に探索し既探索領域の拡張を行う．GA にデータベースを適用することにより，既探索領域の大きさを定量的に把握することが可能であり，計算資源あるいは計算コストを増加させれば必ず既探索領域が拡張する．

提案するデータベースを GA に組み込み，原始的なビットの問題である 1max 問題，だまし問題の 1 つ 3-deceptive 問題，および連続最適化問題のテスト問題に適用し，データベースの有効性を示した．また，スケーラビリティを「計算資源の増加に対する既探索領域の増加」と定義し，ローカルサーチを有するデータベースを GA に導入することでスケーラビリティを保証できるかについて PC クラスタ環境上で検証した．その結果，1) 計算コストを制限しない場合，既探索領域が表せていること，2) 全探索により得られた解が最適解であることを保証すること，3) GA が局所解に陥りやすい問題においても，ローカルサーチで探索を進めていくことにより最適解を得ること，4) データベースを GA に導入することによって既探索領域が計算資源の増加にともなって線形に増加すること，および，5) 計算コストを制限した場合においても従来の GA と同等の結果を得られることを示した．

今後，大規模計算環境においてその有効性を検証する予定である．提案したデータベースの適用方法については種々考えられるが，本データベースを利用したタブ・サーチ，GA の母集団が収束した場合の未探索領域への再初期化の導入を兼ねて議論する必要がある．本論文で提案したデータベースでのローカルサーチでは適用ごとに計算量が指数的に増大するといった問題があり，大規模な問題を対象とした場合には実装モデルに工夫を要すると考えている．その問題を解決した新たなデータベースについては次報で報告する．

## 参考文献


1) Goldberg, D.E.: *Genetic Algorithms in Search Optimization and Machine Learnig*, Addison-Wesley (1989).
2) Satoh, H., Yamamura, M. and Kobayashi, S.: Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation, *Proc. IIZUKA*, pp.494–497 (1996).
3) Kargupta, H.: SEARCH, polynomial complexity and the fast messy genetic algorithm, University of Illinois at Urbana-Champaign, Urbana, IL. IlliGAL Report, No.95008 (1995).
4) Harik, G.R.: Linkage learning in via probabilistic modeling in the ECGA, University of Illinois at Urbana-Champaign, Urbana, IL. IlliGAL Technical Report, No.99010 (1999).
5) Ono, I. and Kobayashi, S.: A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover, *Proc. 7th Int. Conf. on Genetic Algorithms*, pp.246–253 (1997).
6) Pelikan, M., Goldberg, D.E. and Lobo, F.: A Survey of Optimization by Building and Using

Probabilistic Models, Technical Report 99018, IlliGAL (1999).
7) Larranaga, P. and Lozano, J.A.: *Estimation of Distribution Algorithms, A New Tool for Evolutionary Computation*, Kluwer Academic Publishers (2001).
8) Tanese, R.: Distributed Genetic Algorithms, *Proc. 3rd International Conference on Genetic Algorithms*, pp.434–439 (1989).
9) Jansen, T.: On the Analysis of Dynamic Restart Strategies for Evolutionary Algorithms, *Proc. Parallel Problem Solving from Nature* (*PPSN VII*), *7th International Conference*, pp.33–43 (2002).
10) Fukunaga, A.S.: Restart Scheduling for Genetic Algorithms, Lecture Notes in Computer Science, Vol.1498, pp.357–369 (1998).
11) Luke, S.: When Short Runs Beat Long Runs, *Proc. Genetic and Evolutionary Computation Conference*, pp.74–80 (2001).
12) Maresky, J., et al.: Selectively Destructive Restart, *Proc. 6th International Conference on Genetic Algorithms*, pp.144–150 (1995).
13) 谷村勇輔，廣安知之，三木光範：グリッド計算環境でのマスターワーカシステムの構築，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG6(ACS6), pp.197–207 (2004).
14) Imade, H., et al.: A Grid-Oriented Genetic Algorithm for Estimating Genetic Networks by S-Systems, *Proc. SICE Annual Conf.*, pp.3317–3322 (2003).
15) Imade, H., et al.: A framework of grid-oriented genetic algorithms for large-scale optimization in bioinformatics, *Proc. Congress on Evolutionary Computation in Canberra*, Vol.1, pp.623–630 (2003).
16) 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 2002-HPC-93, pp.155–160 (March 2003).
17) Thierens, D. and Goldberg, D.E.: Elitist Recombination: An integrated selection recombination GA, *Proc. 1st IEEE Conference on Evolutionary Computation*, pp.508–512 (1994).
18) Pelikan, M., et al.: BOA: The Bayesian Optimization Algorithm, IlliGAL Report, No.99003 (1999).

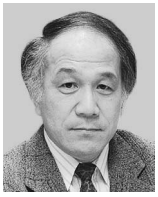
(平成 16 年 12 月 30 日受付)
(平成 17 年 6 月 7 日再受付)
(平成 17 年 6 月 18 日採録)

**花田　良子**（学生会員）

2004 年同志社大学大学院工学研究科修士課程修了．同年同志社大学大学院工学研究科博士課程入学．進化的計算，最適設計，並列処理等の研究に従事．

**廣安　知之**（正会員）

1997 年早稲田大学大学院理工学研究科後期博士課程修了．早稲田大学理工学部助手を経て，1998 年同志社大学工学部助手．2003 年より同志社大学工学部知識工学科助教授．進化的計算，最適設計，並列処理等の研究に従事．IEEE，電子情報通信学会，計測自動制御学会，日本機械学会，超並列計算研究会，日本計算工学会各会員．



**三木　光範**（正会員）

1950 年生．1978 年大阪市立大学大学院工学研究科博士課程修了，工学博士．大阪市立工業研究所研究員，金沢工業大学助教授を経て，1987 年大阪府立大学工学部航空宇宙工学科助教授，1994 年同志社大学工学部教授．進化的計算手法とその並列化，および知的なシステムの設計に関する研究に従事．著書は『工学問題を解決する適応化・知能化・最適化法』（技法堂出版）等多数．IEEE，米国航空宇宙学会，人工知能学会，日本機械学会，計算工学会，日本航空宇宙学会等会員．通産省産業技術審議会委員等歴任．超並列計算研究会代表．