Vol. 47 No. 4 情報処理学会論文誌 Apr. 2006

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改良

花田良子[†] 廣安知之[††] 三木光範[††]

GA は並列環境に適しているといわれているが，多資源計算環境に GA を適用した際には，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メカニズムが必要となる．本研究のアプローチはスケーラビリティを「計算コストおよび資源の増加に対する既探索領域の増加」と定義し，既探索領域データベース，およびローカルサーチを導入することによって，スケーラビリティを保証することである．前回提案したスキーマに基づく個体データベースでは，ローカルサーチで適用毎に必要とする評価計算回数が指数的に増大するといった問題があった．本論文では，その問題点を解決した新たなデータベースおよびローカルサーチを提案する．提案するデータベースでは，解空間を 2 次元表現にマッピングし，既探索個体を 2 次元平面上で表現する．また，既探索領域拡大のローカルサーチはその平面上で行うことで，適用毎の計算量の増加の問題点を解決する．

An Improvement of Database with Local Search Mechanisms for Genetic Algorithms in Large-Scale Computing Environments

Yoshiko HANADA,[†] Tomoyuki HIROYASU[††] and Mitsunori MIKI[††]

It is convinced that GAs are the suitable model for parallel environment. However, mechanisms to use massive computation resources laconically and to search effectively are necessary if large-scale computer systems are available. In our approach, we define the scalability as increases in search regions against the increase in computing resources or costs. Our target is to guarantee the scalability by applying GA-specific database with the local search mechanism. In our previous work, there was the drawback that computing costs increased exponentially in accordance with generations. In this study, we introduce new database and local search based on our previous work. Our database used the mapping method that represents whole search space as a two-dimensional plane. Searched individuals are expressed on this plane. By applying the local search on the plane, the drawback of increases in computing costs can be solved.

1. はじめに

近年，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．また，GA は多点探索であることから並列化が容易であり，これまでに評価計算あるいは遺伝的操作を並列に実行するマスタ・スレーブモデル[2]，あるいは島モデル[3] などで並列実装されてきた．

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，近年では数千の計算ノードからなる PC クラスタや数百万台規模のグリッド計算環境など，膨大な資源が利用可能となってきている．一般に GA は高い並列性に加え，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．そのため，これまでに多資源計算環境を対象とした並列モデルが開発されており[4)~8)]，GA の並列化の研究はこれらを対象としたアプリケーションが主流となっていくと考えられる．

大規模計算環境に GA を対応させる場合に考慮しなければならない点は，計算コストおよび資源の増加に対して，解探索性能を低下させることなく，スケーラビリティを保証することである．そのため，計算資源

† 同志社大学大学院/日本学術振興会特別研究員
Graduate School of Engineering, Doshisha University / JSPS Research Fellow
†† 同志社大学工学部
Department of Engineering, Doshisha University

の効率的な利用方法を考慮する必要がある．GA の大規模並列化の最も単純なアプローチは，母集団の個体数を増加させることで並列度を向上することである．また，適度な母集団サイズの GA を並行するといった手法が考えられる．しかしながら，前者は，収束が遅くなることで，時間一定で比較した場合には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こと，また，問題に対して適切な母集団サイズが存在し，それ以上多く設定した場合には性能の向上が飽和してしまうことが知られている[9]．後者については，探索に重複が生じ，不要な計算をしているノードが存在するといった問題がある[10]．これより，利用可能な計算資源が増加したとしても，GA は有効に扱うことが困難であると考えられる．

本研究のアプローチでは，スケーラビリティを「計算コスト，資源の増加に対する既探索領域の増加」と定義する．解探索性能を低下させることなくスケーラビリティを向上させるため，不要な計算が行われない仕組みとして，次の 3 点を考案する．また，目標とする並列化のモデルの概念図を**図 1** に示す．

1) 既探索領域を保存するデータベース
2) ノードを効率的に利用することを目的とした未探索領域を探索するローカルサーチ
3) 探索の重複を回避することを目的としたデータベースの応用としてのタブサーチ

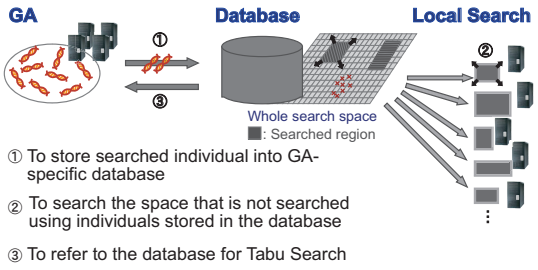


図 1 大規模計算環境における並列化の概念図
Fig. 1 Concept of large-scale parallelization

本研究はバイナリ表現を用いた GA を対象としており，有限の解空間上での既探索領域の表現を考案する．前回の報告[10] では，スキーマに基づく既探索領域の表現を用いたデータベース，および全既探索個体を限られたサイズで格納するための圧縮メカニズムとしてのローカルサーチを提案した．ローカルサーチでは，データベースの既探索領域の情報をもとに未探索領域を重点的に探索する．これにより，データベースを GA に組み込むことで，計算コストおよび資源の増加に伴い既探索領域が拡張することを示した．また，限定された計算コストの中でも探索能力が保持できることを示した．しかしながら，1 回のローカルサーチに必要とする計算量 (評価計算回数) が探索の進行とともに指数的に増大し，探索の早い段階で GA の探索を圧迫していた．そのため，評価計算回数を制限した場合には，GA で十分に探索ができないという問題があった．本論文では，前回の提案手法の探索特性を保持しつつ，計算量の増加の問題点を改善した新たな既探索領域データベース，およびローカルサーチを提案する．また，データベースの応用としてタブサーチの有効性を示す．

2. 既探索領域を保存するデータベース

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，全既探索個体を保存することは困難であり，限られたサイズで格納するための個体表現が必要となる．前回提案したデータベースではスキーマに基づいた個体表現を用いていた (付録 A.1)．そこでは，保存された既探索領域をもとに未探索領域の探索を行うローカルサーチにおいて，適用毎に必要とする計算量 (評価計算回数) が探索の進行とともに指数的に増大するといった問題があった．本論文では，計算量が指数的に増加しないような既探索個体の表現方法を考案する．なお，計算量および計算コストは解の評価計算回数として議論する．

考案する方法では解空間を 2 次元平面にマッピングし，個体を 2 次元平面上で表現する．未探索領域を探索するローカルサーチをこの平面上で行うことで，1 回のローカルサーチに必要とする計算量は線形に増加し，既探索領域の大きさはローカルサーチの適用回数に対して 2 次関数的に増加する．

2.1 データベースにおける個体の表現

本研究では個体の遺伝子表現に $\{0, 1\}$ を用いる GA を対象とする．提案するデータベースでは，T.Collins によって提案された多次元の解空間を 2 次元平面に表現するマッピング手法[11] を用いて，2 次元座標で表現された個体，および個体の集合を格納する．このマッピング手法では，染色体長 L の個体は，偶数番目の遺伝子座 ($2k$ 番目 $(1 \leq k \leq L/2)$の遺伝子座，左端の遺伝子座を 1) の遺伝子からなるビット列，および奇数番目の遺伝子座 ($2k$-1 番目 $(1 \leq k \leq L/2)$の遺伝子座) からなる長さ $L/2$ のビット列をそれぞれグレイコーディングして得られた整数値 x, y を用いて，2 次元座標 (x, y) と表現する．個体と座標が 1 対 1 に対応しており，全探索空間は $2^{L/2} \times 2^{L/2}$ の 2 次元平面で表

現される．**図 2** に個体を座標表現した例を示す．

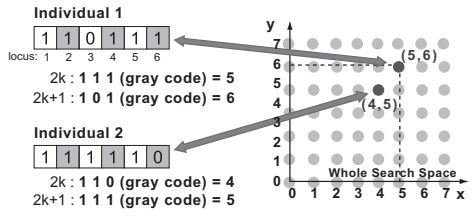


図 2 6 ビット問題の個体の表現
Fig. 2 Representation of an individual on 6 bits problem

2.2 既探索領域の表現

本データベースでは 2 次元平面上で矩形を成している既探索個体群を，対角に位置する 2 つの頂点 (x_{min}, y_{min})，(x_{max}, y_{max}) を用いて，まとめて保存する．また，2 点の座標に加えて，その既探索領域での最良の個体を保存する．**図 3** に 6 ビットの問題の解空間の 2 次元平面表現，および矩形で表現された既探索個体群の例を示す．また，**図 4** に図 3 の既探索領域を保存したデータベースを示す．図中では，(x_{min}, y_{min})-(x_{max}, y_{max}) と表現している．

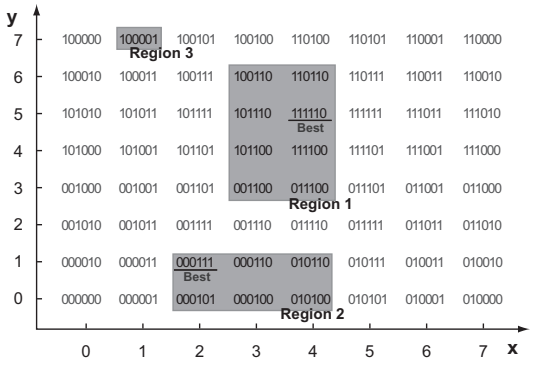


図 3 6 ビット問題の解空間の 2 次元平面
Fig. 3 The searched plane of 6 bits problem

	Searched region	Best individual	Size of region
Region1	(3,3)-(4,6)	111110	8 (4x2)
Region2	(2,0)-(4,1)	000111	6 (2x3)
Region3	(1,7)-(1,7)	100001	1 (1x1)

図 4 図 3 の領域を表現したデータベース
Fig. 4 Database representing searched region on Fig. 3

提案するデータベースでは個体群を座標の組で表現することにより，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等の操作で処理時間を費やすことはない．また，矩形の面積で既探索領域の大きさを定量的に表すことが可能である．

T.Collins のマッピング手法はグレイコーディングをベースとしており，各点において上下左右の点に対応する個体とのハミング距離がいずれも 1 である．局所的に 2 次元平面における近さがビット列での近さを表しており，似た構造を持つ個体群を連続した領域として扱えるといった点で，より小さなデータベースサイズで全既探索領域を把握するのに適していると考えられる．ただし，長さ L の，あるビット列 s とハミング距離が 1 のビット列に対応する点は L 個あるが，それらすべてが 2 次元平面上で s に近接しているのではなく☆，大域的には，解 s に対して 2 次元平面において近い解ほど，ビット列が近いということを表していない．そのため，解どうしが非常に似通っていても，別の領域に保存される可能性がある．

3. データベースの諸操作

3.1 ローカルサーチ

資源が膨大に存在するときには，GA が利用しきれない一部のノードを効率的に利用するためのローカルサーチを行う．ローカルサーチでは，データベースに保存された既探索個体をもとに，未探索領域を重点的に探索し，既探索領域の拡張を行う．

本研究で提案するローカルサーチは，データベースで保存されている既探索領域である 2 次元平面上の矩形を縦軸方向，横軸方向へ拡大するよう探索を進めていく．矩形 (x_{min}, y_{min})-(x_{max}, y_{max}) は以下のように縦軸，横軸方向に拡大される．

Rightward expansion:
個体群 $\{(x_{max}+1,\ y) \mid y_{min} \le y \le y_{max}\}$ を探索し，x_{max} を $x_{max}+1$ に更新する．
Leftward expansion:
個体群 $\{(x_{min}-1,\ y) \mid y_{min} \le y \le y_{max}\}$ を探索し，x_{min} を $x_{min}-1$ に更新する．
Upward expansion:
個体群 $\{(x,\ y_{max}+1) \mid x_{min} \le x \le x_{max}\}$ を探索し，y_{max} を $y_{max}+1$ に更新する．
Downward expansion:
個体群 $\{(x,\ y_{min}-1) \mid x_{min} \le x \le x_{max}\}$ を探索し，y_{min} を $y_{min}-1$ に更新する．

ローカルサーチの 1 ステップでは，各領域に対して縦軸，横軸方向に 1 だけ拡大する操作を行う．**図 5** にデータベースに保存されている一領域に対してローカルサーチを適用した例を示す．太線で囲んでいる領域

☆ 例えば，$s = 000\cdots00$ と左から $2k$ 番目 $(1 \le k \le L/2)$の遺伝子座における遺伝子だけが異なる点は，$y{=}2^{L/2-k}$ の軸について対称の位置に存在する．また，$2k$-1 番目 $(1 \le k \le L/2)$の遺伝子座は $x{=}2^{L/2-k}$ の軸について対称の位置に存在する．

が 1 ステップごとに，探索される個体群である．

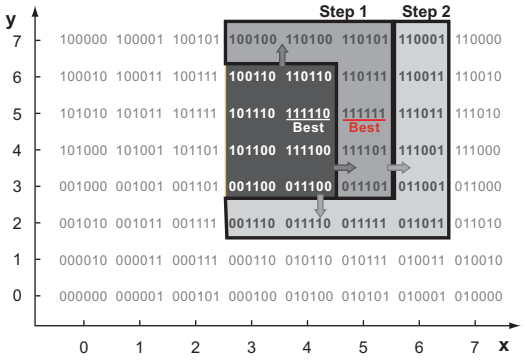


図 5 ローカルサーチの例
Fig. 5 An example of local search

縦軸，および横軸における矩形の拡大の方向は矩形のエッジ上での最良個体の評価値を用いて決定する．横軸方向の場合は，右側および左側エッジ上の最良個体について，評価値の良い方向に探索が進む．縦軸の拡大方向についてもこれと同様の方法で決定する．ローカルサーチの性能は 2 次元平面へのマッピング手法に依存するため，矩形の拡大の方向の決定は手法の特徴を考慮する必要がある．本研究で用いたマッピング手法では，局所的には 2 次元平面における近さがビット列での近さを表せているが，大域的には表せていない．矩形の拡大操作ではエッジ上の近傍が新たに探索されることになるため，ローカルサーチの過程において良好な解が得られている方向を優先して探索するといった点でエッジ上の最良個体を用いることが適切であると考えられる．

前回のローカルサーチでは，1 ステップで必要とする計算量が探索の進行とともに指数的に増大するといった問題があったが，本論文で提案するローカルサーチでは計算量は適用毎に線形に増加するため，評価計算回数を制限した場合でも，探索の後半でローカルサーチが GA の探索を圧迫することはない．

3.2 既探索領域のマージ

ローカルサーチでは，データベースで保存している各矩形領域に対して拡大する操作を行う．探索を進めていくにしたがい，矩形間の重なり，すなわち探索の重複が生じる．このような探索の重複を回避するため，複数の領域で重なりが生じた場合は次のように 1 つの領域にマージする．**図 6** にマージの例を示す．

Merge: $I_a \cap I_b \neq \phi$ となるような領域 I_a=(x^a_{min}, y^a_{min})-(x^a_{max}, y^a_{max})，領域 I_b=(x^b_{min}, y^b_{min})-(x^b_{max}, y^b_{max}) が存在したとき，$I_b \subset I_a$ となるまで I_a を拡張する操作を行い，他方を削除する．$x_1 < x_2$ のとき $min(x_1, x_2) = x_1$, $max(x_1, x_2) = x_2$ と表記すると，I_a は I'_a=($min(x^a_{min}, x^b_{min})$, $min(y^a_{min}, y^b_{min})$)-($max(x^a_{max}, x^b_{max})$, $max(y^a_{max}, y^b_{max})$) となり，領域 $I'_a \cap \neg(I_a \cup I_b)$ が探索される．

データベースにおいてマージ可能な組み合わせがない状態の場合，全既探索領域の大きさは，各データが示す既探索領域の大きさの単純な総和となる．

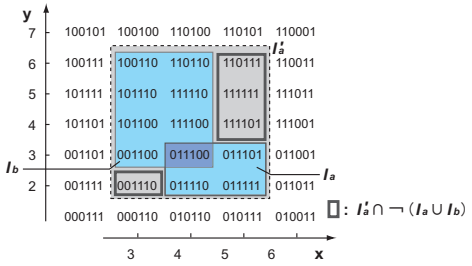


図 6 マージの例
Fig. 6 An example of the merge

3.3 GA へのデータベースの適用

本研究で考案するデータベースの目的は，GA による探索の一方で，データベースに保存される既探索領域の情報をもとに未探索領域を重点的に探索するローカルサーチを行うことである．この仕組みにより，計算コストの増加に対して，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．

ローカルサーチの適用のタイミング，対象となる領域数，また，一度に適用するステップ数など，保存している各領域へのローカルサーチの適用方法は任意であり，提案するデータベースの GA への適用については種々の方法が考えられる．ここでは，データベースで保存されている全領域に対して，ローカルサーチを適用する領域数についてはパラメータとし，N_{LS} で表現する．ローカルサーチで必要となる計算量および既探索領域の大きさは N_{LS} の増加に伴って増加する．数値実験で用いたデータベースの適用方法の詳細については次節で述べる．

分散環境においては，**図 7** に示すように，ローカルサーチを適用する領域数 N_{LS} を利用できる計算ノード数に設定し，各計算ノードに矩形領域を割り当てることで単純に並列化が可能である．これにより，既探索領域はノードの増加に対してスケーラブルに増加する．前回の手法と同様に，各計算ノード間での探索に

強い依存がないため，通信量が少なく高スループットな計算が期待でき，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．なお，ローカルサーチの適用のタイミングについては，分散環境の場合は，ヘテロな環境，通信のオーバヘッド等のことを考慮すると，GA と非同期であることが好ましいと考えられる．

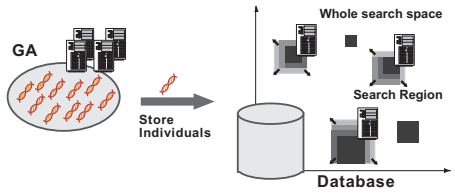


図 7 ローカルサーチの並列化
Fig. 7 Parallelization of local searches

4. 数値実験

提案するデータベースにおける個体の表現方法とそれを用いたローカルサーチを GA に組み込み，探索の特徴，計算コストを制限したときの探索性能，計算コストおよび資源の増加に対するスケーラビリティを検証する．本実験では，1 台のマシンおよび分散環境でデータベースを適用することの有効性を示す．1 台を用いた実験ではローカルサーチを適用する領域数 N_{LS} の増加，分散環境においては計算ノードの増加に対するスケーラビリティを検証する．

GA とデータベースの組み合わせ方には種々の方法が考えられるが，ここでは次のように，GA で得られた良好な解をデータベースに保存し，それを用いてローカルサーチを行うといった単純な適用法でデータベースの有効性を検証する．なお，データベースは初期世代において保存している領域数は 0 である．

【GA + LS のフロー】
Step 0 GA の母集団，データベースを初期化する．
Step 1 GA の母集団に対し，遺伝的オペレータを適用する．
Step 2 GA の母集団の最良個体がデータベースに存在しないならば，データベースに保存する．
Step 3 N_{LS} 個の領域に対してローカルサーチを k ステップ適用する．保存されている領域数が N_{LS} 以下の場合はすべての領域に対して適用する．そうでない場合は領域が小さいものを優先して適用する．また，その中でも評価値が高い領域を優先する．
Step 4 データベースを更新する．データベースにおいてマージが可能な領域の組合せがあれば，それを適用する．Step 1 へ戻る．

この適用方法では，マシン 1 台を用いた場合，GA とローカルサーチが交互に適用されることになる．一方，分散環境においては各計算ノードに領域を割り当て，ローカルサーチを適用するため，GA(Step1，2) とローカルサーチ (Step3，4) を非同期にする．分散環境における実装例については 4.4 節で述べる．

テスト問題として原始的なビットの問題である 1max 問題，式 (1)，(2) に示すだまし問題の 3-deceptive 問題，5-trap 問題[12]，および式 (3)〜(6) に示す連続関数のテスト問題を対象とする．いずれも最大化問題となるように記述している．3 ビット，あるいは 5 ビットの連続するビット列からなるパーティション毎に部分的な評価値を持つだまし問題 (1)，(2) において，N はパーティション数，u_i はパーティション内の 1 の個数を示す．これらの問題は実際に分散環境で解かれるべき問題ではないが，ここでは分散環境で実装した場合などにおける探索の特性について示すため，これらのテスト問題を用いた．計算コスト，資源に対する既探索領域のスケーラビリティなど，探索の特性はどのような問題においても同様のため，テスト問題を通して探索の特性を明らかにすることで，他の実問題にも対応することが可能となる．

$$F_{3-deceptive} = \sum_{i=1}^{N} f_i \tag{1}$$
$$f_i = \begin{cases} 0.9, & u_i = 0 \\ 0.8, & u_i = 1 \\ 0.7, & u_i = 2 \\ 1.0, & u_i = 3 \end{cases}$$

$$F_{5-trap} = \sum_{i=1}^{N} f_i \tag{2}$$
$$f_i = \begin{cases} 4 - u_i, & \text{if} \quad u_i < 5 \\ 5, & \text{otherwise} \end{cases}$$

$$F_{Rastrigin} = -\sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \tag{3}$$
$$x_i \in [-5.12, 5.12)$$

$$F_{Schwefel} = 418.98288n - \sum_{i=1}^{n} -x_i \sin\left(\sqrt{|x_i|}\right) \tag{4}$$
$$x_i \in [-5.12, 5.12)$$

$$F_{Ridge} = -\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2 \tag{5}$$
$$x_i \in [-64, 64)$$

$$F_{Griewank} = -1 - \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \left(\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) \tag{6}$$
$$x_i \in [-512, 512)$$

4.1 探索の特徴

性質の異なる 2 種の原始的なビットの問題を用いて，データベースを適用した場合の探索の特徴を検証する．また，前回提案したデータベースとの比較を行う．なお，前回の手法については，付録 A.1 にメカニズム，およびパラメータ設定等を記述している．本節では，20 ビットの 1max 問題，30 ビットの 3-deceptive 問題を用いる．全探索空間の大きさは前者は 2^{20}，後者は 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．最適解の評価値はそれぞれ，20 および 10 である．GA における世代交代モデルには ER(Elitest Recombination)[13] に準ずるモデルを用いた．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，突然変異率を $1/L$(L：染色体長)，母集団サイズを 20 とした．なお，ローカルサーチは毎世代，データベースに保存されている領域すべてに対して 1 ステップ適用している．

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，および 50 試行の平均値である．図中，黒の実線が本データベースを GA に組み込んだ結果，グレーの実線は前回提案したデータベースを GA に組み込んだ結果，および黒の点線が通常の 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．

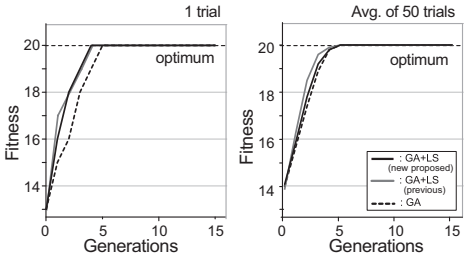


図 8 評価値の推移 (1max 問題)
Fig. 8 History of fitness on 1max

GA をベースとして探索を進めていくため，前回に報告したデータベースと同様に，従来の GA と同等の解探索性能を持っている．このように GA が大域的最適解を得やすい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 のように，探索の過程において得られた解が全探索空間中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．なお，本研究で提案するローカルサーチを有するデータベースは世代数に対して概ね 2 次関数的に増加していることが確認できる．

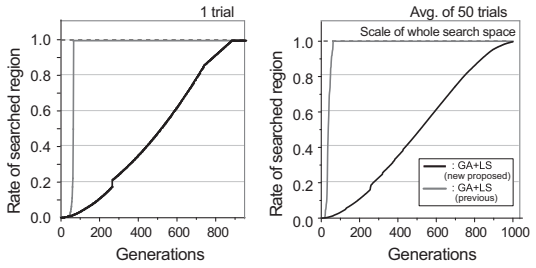


図 9 既探索領域の割合の推移 (1max 問題)
Fig. 9 History of searched region rate on 1max

図 10 および**図 11** に 3-deceptive 問題における世代数および評価計算回数に対する適合度の推移を示す．これらの結果は 1 試行の例，および 50 試行の平均値である．なお，既探索領域の割合の推移はここに示していないが，1max 問題と同様の傾向を示している．

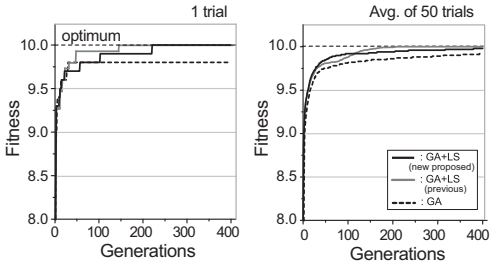


図 10 評価値の推移 (3-deceptive 問題)
Fig. 10 History of fitness on 3-deceptive

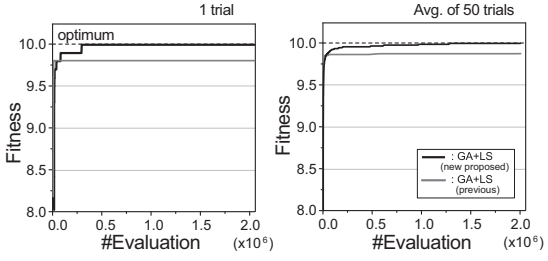


図 11 評価計算回数を制限したときの比較
Fig. 11 History of fitness on 3-deceptive with limited computation costs

図 10 から，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テスト問題であるが，本手法は，前回に報告したデータベースと同様にローカルサーチを行い，既探索領域を拡大することで必ず最適解が得られる保証を有している．前回のローカルサーチは適用毎の計算量が指数的に増大し，図 9

のように既探索領域が世代に対して指数的に増加していた．そのため，評価計算回数を限った場合にはデータベースを適用することにより，探索初期からローカルサーチが GA の探索を圧迫し，GA で十分な探索が行われない問題があった．本提案手法はローカールサーチでの計算量の増加を抑えることで，図 11 に示すように評価計算回数を限った場合でも，前回の手法と比較して GA の十分な探索を可能にしており，解探索の性能が向上していることがわかる．

4.2 計算コストを制限したときの探索性能

1 台のマシン上で評価計算回数を一定にしたときの，本提案データベースを用いた GA の解探索性能を検証する．本手法ではデータベースを適用することでローカルサーチにも多くの計算が必要となる．ここでは，バイナリ表現を用いた GA と比較することで，評価計算回数を限定した場合でも，探索性能が低下しないことを示す．また，前回のデータベースを用いた GA とも比較する．式 (3)，(4)，(5)，および (6) に示した 4 つの連続関数を用いる．いずれの問題も 10 次元の問題であり，最適解の評価値は 0.0 である．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 100 とする．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 $1/L$ とする．N_{LS}=1 とし，ローカルサーチは 1 つの領域に対して，毎世代，1 ステップ適用している．

図 12 に，評価計算回数を 0.5x10^5，1.0x10^5，1.5x10^5，2.0x10^5 および 1.0x10^6 に制限したときの最適解を得た回数を示す．これらは 50 試行の結果である．

図 12 から，本提案手法は評価計算回数を制限した場合においても，GA と同等の結果を得ていることがわかる．僅かではあるが，評価計算回数が比較的少ない場合において，本提案手法は GA と比較して最適解を多く得ている．一方，従来のデータベースを用いた GA は，本手法と比較して最適解への到達率が低い．これは，前節で示したようにローカルサーチにおける評価計算が GA の探索を圧迫しているためである．これより，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わかる．

4.3 計算コスト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一つは計算コストの増加に対して既探索領域を増加させることである．本手法では，計算ノード，あるいは N_{LS} を増加させることによって計算コストが増加する．ここでは，マシン 1 台を用いて，N_{LS} の増加に対する既探索領域の増加のスケーラビリティを検証する．

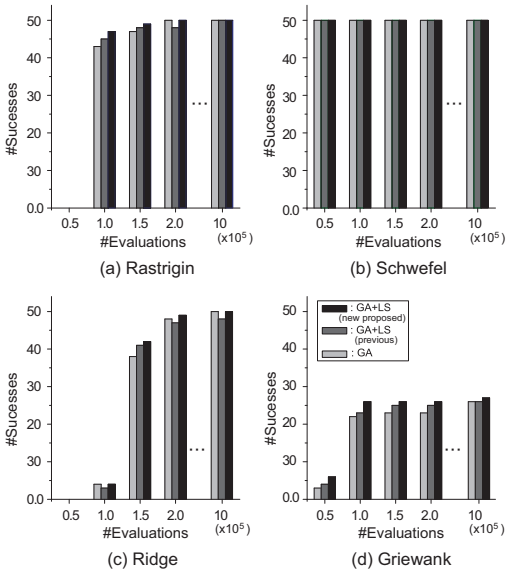


図 12 評価計算回数を制限したときの比較
Fig. 12 Comparing of 3 methods with limited computation costs

対象問題として，120 ビットの 3-deceptive 問題，100 ビットの 5-trap 問題，および 10 次元の Griewank 関数を用いる．最適解の評価値はそれぞれ 40，100，0.0 である．

図 13 に，N_{LS} を 5，10，20，30，および 40 に変化させたときの解探索性能，および評価計算回数の増加，既探索領域の割合を示す．なお，パラメータの設定は N_{LS} を除き，前節と同様である．3-deceptive 問題，5-trap 問題については 500 世代および 2000 世代，Griewank 関数については 4000 世代における最良値の 50 試行平均を示している．なお，Griewank 関数については最適解を得た回数についても示している．図 13 の下図では，左縦軸および黒線が解空間に対する既探索領域の割合 (%region)，右軸およびグレー線が評価計算回数 (#eval) を示している．

図 13 から，いずれの問題においても，データベースを適用することにより，N_{LS} を増加，すなわち計算コストを増加させることで既探索領域が増加していることがわかる．また，解探索性能が向上している．なお，下図では，N_{LS} の増加に対して評価計算回数および既探索領域がほぼ 2 次関数的に増加している．これは，本実験で用いたデータベースの適用方法ではローカルサーチを適用するステップ数を毎世代一定に

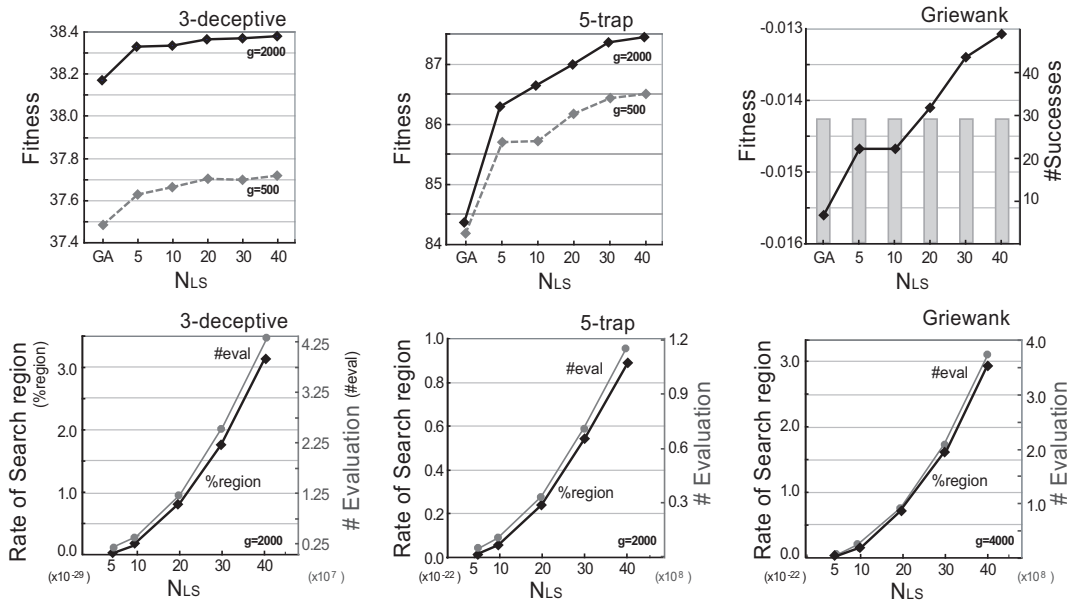


図 13 N_{LS} の増加に対する既探索領域，解探索性能の向上
Fig. 13 Improvement of search performance in accordance with N_{LS}

していることで，拡大に必要とする計算量がステップ数の増加に対して線形に増加し，各領域が 2 次関数的に増加するためである．

4.4 分散環境での実装例

分散環境におけるデータベースの適用方法は任意であるが，本実験では単純に，図 7 に示すように各計算ノードに矩形領域を割り当てることで並列化を行う．ここでは，分散環境へのジョブの投入等における通信などを考慮し，**図 14** に示すように GA とローカルサーチを非同期に行う．GA ではデータベースに毎世代最良個体を保存しつつ探索を進める．一方，分散環境において，データベースに保存している領域を遊休ノードに割り当て，各ノードはある程度ローカルサーチが進むと結果をデータベースに戻すといったことを繰り返す．

1 ノードでの実装では，各領域に対して 1 ステップのローカルサーチを適用していたが，ヘテロかつリソースの性能の予測が困難である分散環境では不向きであるため，各ノードで領域に対してローカルサーチを一定時間適用した後，結果を出力しデータベースを更新する．

本実験では，同志社大学情報メディア課が運営する分散環境を用いて，計算ノードの増加に対する既探索領域のスケーラビリティを検証する．この分散環境は，

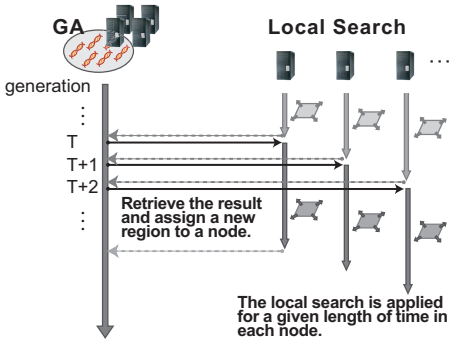


図 14 分散環境での実装
Fig. 14 Implementation on distributed environments

米 United Devices 社☆が開発する商用グリッドミドルウェアである Grid MP によって構築されたマスタ・ワーカ型の計算環境であり，サーバにジョブを投入すると，遊休ノードがそれを実行し，結果をサーバに返すといった形態をとる．**表 1** に環境の詳細を示す．表中の Local Machine で GA を行い，Node でローカルサーチを行う．そして，図 14 における黒矢印およびグレーの点線矢印の処理が Server を介して行われる．利用ノード数を 10，20，40，および 60 に変化させ，ノード数に対するスケーラビリティを検証する．

対象問題として，100 ビットの 5-trap 問題を用いる．GA のパラメータは前節の実験と同様である．各

☆ United Devices：http://www.ud.com

表 1 実験環境
Table 1 Specification of Machines Used for the Experiment

	#Nodes	Processor	Memory	OS
Local Machine	1	Intel Xeon 2.8GHz x2	1GB	RedHat ES 2.0
Server	1	Intel Pentium4 3.0GHz	1GB	RedHat ES 3.0
Node	60	Intel Pentium4 2.4GHz	512MB	Windows XP

ノードでは割り振られた領域に対してローカルサーチを一定時間適用し，結果をサーバに返す．本実験で用いる計算環境は比較的小さなテスト環境であるため，1 分間程度と設定している．なお，膨大なノードを有する環境を対象とした場合，計算ノードが頻繁に結果をサーバに返すことがサーバの大きな負担になるため，ローカルサーチの適用時間を長くした方が好ましい．GA を実行しているローカルのマシンは，サーバから結果を回収してデータベースを更新し，アイドル状態のノード分だけ新たな領域についてのローカルサーチを計算ジョブとしてサーバに投入する．

図 15 に，500 世代における最良解，および既探索領域の割合を示す．これらは 5 試行の平均である．なお，左図におけるグレーの点線は図 13 で示した 1 ノードを用いたときの 500 世代での結果を示している．

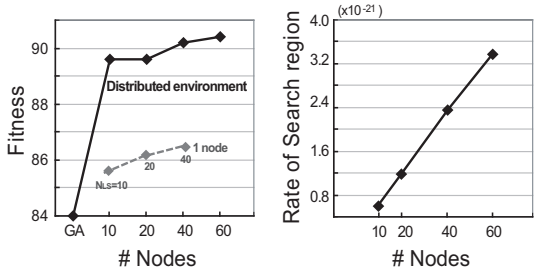


図 15 ノード数の増加に対する既探索領域，解探索性能の向上
Fig. 15 Improvement of search performance in accordance with #Nodes

これらの結果から，分散環境で実装することにより，より良好な解が得られ，ノード数の増加に対して解探索性能が向上していることがわかる．また，ローカルサーチの適用毎の計算量が一定であるため，既探索領域はノード数の増加に伴って線形に増加していることが確認できる．

5. 過去の探索の再利用としてのタブサーチ

本データベースは，スケーラビリティを保持するため，GA における探索の重複を回避することを目的としており，タブサーチのメカニズムを取り入れることにより実現する．従来からのタブサーチでは，近傍生成時に適用した要素間の操作をタブリストとして保持し，短期間あるいは長期間，その要素間の操作を禁止することで，局所探索での解遷移の循環を防いでいる．GA とタブサーチのメカニズムの組み合わせでは，多くの場合，膨大な解空間での解保持が困難であることから，上記のオペレーションベースの禁止を採用している．本研究で提案するデータベースは，探索解を直接保存しており，既探索情報をタブリストとして用いることが可能である．交叉や突然変異において既探索領域内の解生成を禁止することで，GA の試行内での探索の重複を回避する．

5.1 提案データベースを用いたタブサーチ

GA にデータベースを適用し，ローカルサーチで既探索領域を拡大する一方で，GA のオペレータで生成された解の評価時にデータベースを参照する．このとき，データベースで保持されている既探索領域 $\{I_1, I_2, \cdots, I_n\}$ をタブリストとして扱う．そして，解が既探索領域内に含まれる場合には，既探索領域外に解を強制することで，同一の解の再評価を回避する．**図 16** のように，解 s が領域 I_i 内に含まれた場合には，s を I_i の境界付近にランダムで生成した解 s' に置き換える．ここでは単純に，$s = (x, y)$，I_i=(x_{min}, y_{min})-(x_{max}, y_{max}) のとき，$(x, y_{min} - 1)$，$(x, y_{max} + 1)$，$(x_{min} - 1, y)$ および $(x_{max} + 1, y)$ の 4 点からランダムに選んで，s と置き換えている．この仕組みにより，既探索領域内の解を再評価するかわりに，新たな未探索解が生成される．

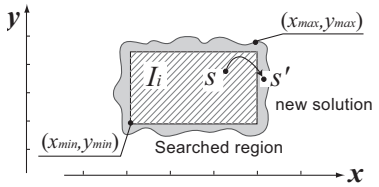


図 16 既探索外領域の解への置き換え
Fig. 16 Replacement of s with s'

5.2 タブサーチの有効性

解の評価時にタブリストとしてデータベースを用いることの有効性を示す．ここでは，GA にデータベースを適用し，ローカルサーチのみを行う探索と，ロー

カルサーチに加え，タブサーチメカニズムを取り入れた探索を比較する．以降，前者を GA+DB，後者を GA+DB+TS と記述する．10 次元の rastrigin 関数および griewank 関数の 2 つを対象として用いる．GA におけるパラメータは 4.2 節と同様である．N_{LS}=10 とし，ローカルサーチは 1 つの領域に対して，毎世代，1 ステップ適用している．

図 17 および**図 18** に，GA+DB と GA+DB+TS の探索の特徴を示す．図はそれぞれ，(a)GA における評価値，(b) データベースに保存されている解が再評価された回数の累積の推移を示している．これらは 1 試行の例である．

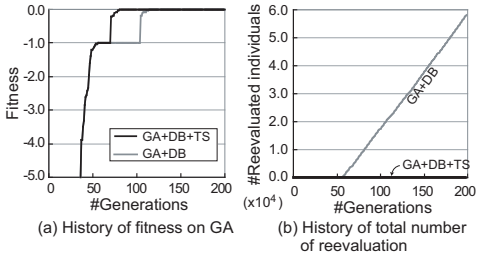


図 17 GA+DB+TS と GA+DB の比較 (rastrigin 関数)
Fig. 17 Comparison of GA+DB+TS with GA+DB on rastrigin function

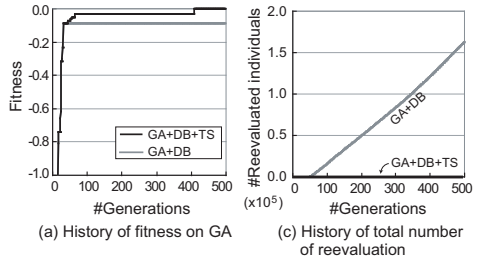


図 18 GA+DB+TS と GA+DB の比較 (griewank 関数)
Fig. 18 Comparison of GA+DB+TS with GA+DB on griewank function

評価値の推移において，griewank 関数に着目すると，GA+DB，GA+DB+TS ともに一度局所解に陥っているが，後者は TS で未探索領域に解を生成することで最適解を得ていることがわかる．また，通常の GA を行っている GA+DB では，探索が進むにしたがい，既探索解の再評価数が増加し，探索に重複が生じていることがわかる．これは，母集団の収束に起因する．一方，GA+DB+TS では，かならず未探索領域に解が生成されるため，データベースの既探索解が再評価されることはない．

図 19 は，両関数についてそれぞれ 4 種類の評価計算回数に制限したときの最適解を得た回数を比較した結果である．これらは独立した 50 試行の結果である．いずれの関数においても，TS を適用することにより，高い割合で最適解を得ており，探索性能が向上していることがわかる．

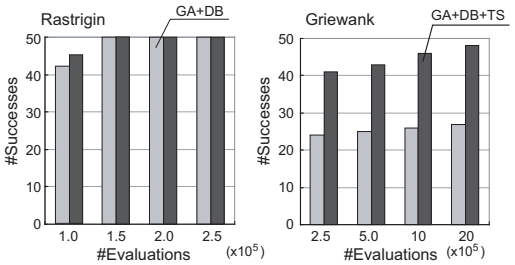


図 19 GA+DB+TS と GA+DB の比較 (最適解を得た試行数)
Fig. 19 Comparison of GA+DB+TS with GA+DB

なお，ここでは GA の 1 試行内におけるタブサーチの有効性を示したが，GA の母集団が収束したときに，母集団を未探索領域に初期化しリスタートを適用した場合には，試行間における探索重複が回避可能である．

6. おわりに

本論文で提案するデータベースでは，解空間を 2 次元表現にマッピングし，既探索個体を 2 次元平面上で表現する．また，既探索領域拡大のローカルサーチはその平面上で行うことで，前回の報告で提案したデータベースにおいて問題であった適用毎の計算量の増加の問題点を解決した．

提案したデータベースを GA に適用し，ローカルのマシン 1 台および分散環境において，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を対象として有効性を検証した．その結果，前回の手法と同様に，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，評価計算回数などの計算コストを制限した場合でも，バイナリ表現を用いた GA との比較して本手法は良好な探索性能を示すこと，計算コストおよび資源の増加に対する探索性能のスケーラビリティが保証されることを示した．一方，評価計算回数を制限した場合においても，ローカルサーチに要する評価計算が GA の探索を圧迫するといった前回の手法における問題を改善したことを示した．また，データベースをタブリストとして利用し，タブサーチを適用することで未探索領域への効果的な探索が可能であり，探索性能が向上することが分かった．

本研究ではテスト問題を用いて検証を行ったが，今後，実問題を対象とし，より規模の大きな環境で検証する予定である．

謝辞 本研究は，科学研究費補助金 特別研究員奨励費課題番号 187179 によるものである．

参 考 文 献

1) Goldberg, D.E.: *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley (1989).
2) Cantú-Paz, E.: Designing efficient master-slave parallel genetic algorithms, Technical report, University of Illinois at Urbana-Champaign, Urbana, IL. IlliGAL Technical Report No. 97004 (1997).
3) Tanese, R.: Distributed Genetic Algorithms, *Proc. 3rd International Conference on Genetic Algorithms*, pp.434–439 (1989).
4) 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， Vol.45, No.SIG6(ACS6), pp. 197–207 (2004).
5) Imade, H., Morishita, R., Ono, I., Okamoto, M.：A Grid-Oriented Genetic Algorithm for Estimating Genetic Networks by S-Systems, *Proc. of SICE Annual Conf.*, pp.3317–3322 (2003).
6) Imade, H., Morishita, R., Ono, I., Ono, N., Okamoto, M.：A framework of grid-oriented genetic algorithms for large-scale optimization in bioinformatics, *Proc. of The Congress on Evolutionary Computation in Canberra*, Vol.1, pp. 623–630 (2003).
7) 染谷博司：グリッド環境に適した遺伝的アルゴリズムによる最適化,「統計数理」, Vol.52, No.5, pp.275–279 (2004).
8) 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，先進的計算基盤システムシンポジウム SACSIS 2005 IPSJ Symposium Series， Vol. 2005, No. 5, pp. 143–151 (2005).
9) Nakano, R., Davidor, Y., Yamada, T.：Optimal population size under constant computation cost, *Proceedings of Parallel Problem Solving fron Nature, PPSN III*, pp.130–138 (1994).
10) 花田良子，廣安知之，三木光範：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，情報処理学会論文誌「数理モデル化と応用， Vol.47, No.SIG 1, pp.19–28 (2006).
11) Collins, T. D.: Understanding Evolutionary Computing: A Hands on Approach, Technical report, KMI-TR-48 (1997).
12) Pelikan, M., Goldberg, D. E., Cantú-Paz, E.：BOA: The Bayesian Optimization Algorithm, Technical report, University of Illinois at Urbana-Champaign, Urbana, IL. IlliGAL Technical Report No. 99003 (1999).
13) Thierens, D. and Goldberg, D.E.: Elitist Recombination: an integrated selection recombination GA, *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pp.508–512 (1994).

付 録

A.1 スキーマに基づく個体表現を用いたデータベース

A.1.1 データベースにおける既探索個体の表現

データベースの個体は，{0,1} のビット列で表現された染色体に加え，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 20** に，マスクを用いた個体の例を示す．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．

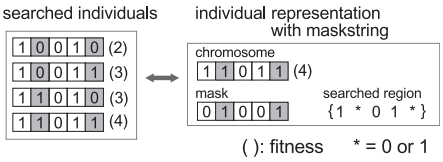


図 20 データベースに保存される個体
Fig. 20 Stored individuals in the database

個体 x におけるマスクの'1' の個数を h とすると，x が示す既探索領域 X の大きさ $|X|$ は 2^h である．N_{DB} 個体 $\{x_i|(1 \leq i \leq N_{DB})\}$ の既探索領域は，添え字 i の集合を I とすると，和集合 $\bigcup_{i \in I} X_i = X_1 \cup X_2 \cup \cdots \cup X_{N_{DB}}$ であり，その大きさは式 (7) で求めることができる．

$$\left|\bigcup_{i \in I} X_i\right| = \sum_{J \subseteq I, J \neq \phi} (-1)^{|J|-1} \left|\bigcap_{j \in J} X_j\right| \qquad (7)$$

N_{DB} はデータベースの最大許容個体数であり，パラメータとしている．本論文の検証実験では，$N_{DB} = 10$ とした．

A.1.2 ローカルサーチ

ローカルサーチの 1 ステップでは，次に示すように，データベースのある 1 個体に対して，マスクの'0' を'1' にする操作を行う．ただし，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．

【ローカルサーチのフロー】

Step1. 式 (8) に示すように個体 $x_i(1 \leq i \leq N_{DB})$ 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．

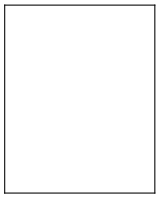
$$a_l = \left| \frac{1}{N_{DB}} \sum_{i=1}^{N_{DB}} (c_{x_i l}) - 0.5 \right| \quad (1 \leq l \leq L) \qquad (8)$$

Step2. データベースから，$|X_i|$ が最小の個体を選択し，x 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．

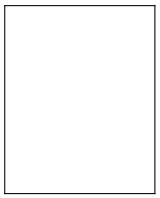
Step4. 探索点 x とマスクが同一で，c_{xl^*} のビットのみを反転した個体群 X' を探索し，$m_{xl^*} = 1$ とする．

(平成 17 年 11 月 18 日受付)
(平成 18 年 2 月 4 日採録)



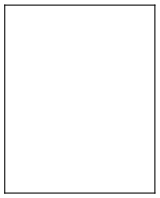
花田 良子（学生会員）

2004 年同志社大学大学院工学研究科修士課程修了．現在，同志社大学大学院工学研究科博士課程在学中．日本学術振興会特別研究員．進化的計算，最適設計，並列処理などの研究に従事．情報処理学会会員．



廣安 知之（正会員）

1997 年早稲田大学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 年より工学部知識工学科助教授．進化的計算，最適設計，並列処理などの研究に従事．IEEE，情報処理学会，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，情報処理学会，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．