集団構造に基づく
進化的計算手法の拡張

大谷 隆浩

目次

図目次

表目次

第 1 章

序論

1.1 進化的計算手法

最適化問題とは与えられた制約条件のもとで，指定された関数 (目的関数) の値が最小または最大となるような変数の値 (最適解) を求める問題である．配送計画や施設配置，生産スケジューリング，学校時間割の編成など，実社会における数理的問題の多くは最適化問題に帰着できる．そのため，この問題を解くための効果的なアルゴリズムを開発することは極めて重要である [1, 2].

最適化問題の中でも線形計画問題など特定の問題クラスにおいては，効率的に最適解を求めることができる．また，大域的単峰性のある関数であればニュートン法や最急降下法により容易に最適解を求めることができる．離散最適化問題においてもナップサック問題や巡回セールスマン問題など特定の問題では，分枝限定法や動的計画法を用いることで実用的な時間内で最適解を求められることが知られている．しかしながら最適化問題の多くは NP 困難に属するため，効率的に厳密な最適解を求めることは極めて難しい．そのため，最適ではないものの実用上十分な目的関数値を持つ解 (準最適解) を求めるための近似解法や発見的アルゴリズムが広く研究されている [3, 4, 5, 6].

最適化問題の準最適解を求めるための手法として，自然界に生育する生物の適応プロセスに着想を得た進化的計算手法が提案されている．進化的計算手法の処理概要を図 1.1 に示す．進化的計算では，候補解 (個体) の集合を保持し，評価値の高い個体が残るように集団を更新していく．集団の更新は，現在の集団から評価値の良い個体を選択し，それらの個体の情報を元に新たな個体を生成し，集団内の評価値の悪い個体と入れ替えるという流れで行われる．どの進化的計算手法においても基本的な処理手順は以上のとおりであるが，個体の選択方法や生成方法の違いによって様々なバリエーションがある．

進化的計算の中でも特に，生物の遺伝と進化の仕組みに着想を得た遺伝的アルゴリズム

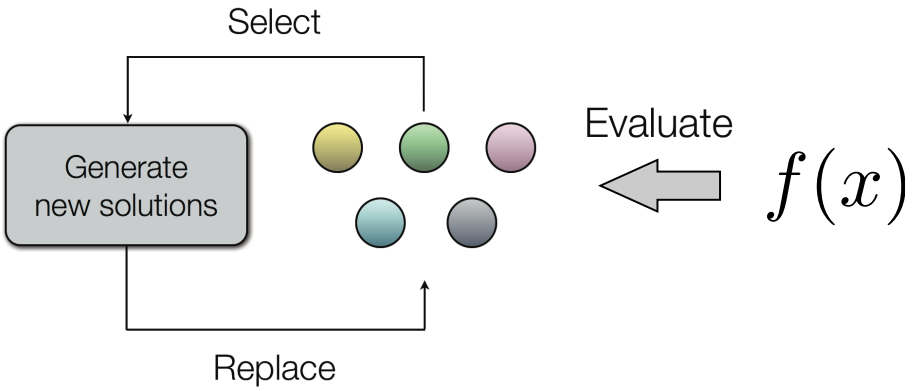


図 1.1　進化的計算の概念

[7, 8, 9] は数多くの実問題に応用され，高い性能を示している．その他にも，蟻の食料採集行動を参考にした Ant Colony Optimization (ACO) [10, 11, 12] や，鳥や魚の群れ行動を応用した Particle Swarm Optimization (PSO) [13] など様々なアルゴリズムが提案されている．

1.2　進化集団の構造

進化的計算はあらかじめ定められた関数の最適解を探索する問題においては一定の成功を収めていると言える．このような枠組みについて，進化する集団の構造という点に注目してみる．集団の構造には，複数の進化集団が形成する構造 (集団間構造と呼ぶ) と，単一集団内の個体が形成する構造 (集団内構造と呼ぶ) の 2 つのレベルがあると考えられる (図 1.2)．集団間構造は，基本的には独立して進化する集団の間でどのような情報がやり取りされるか，どの集団とどの集団が情報のやり取りを行うかを表現するものである．集団内構造は，集団内の各個体が探索空間上においてどのような位置関係をとるかを表現するものである．基本的な進化計算手法の枠組みについて集団構造を見てみると，単一の集団を用いているため集団間構造は無く，集団内構造は集団内の個体が最適解一点に集まるように進化するという単純な構造になっている．

集団構造を拡張した進化的計算手法についても広く研究されている．集団間構造を拡張した手法としては，分散遺伝的アルゴリズム (distributed GA: DGA) [14, 15] が広く知られている．この手法では，複数の集団が同時に進化を行うことで探索性能を向上させており，また各集団が基本的には独立して進化するために並列化効率が非常に高い．集団間の相互作用として，集団内の個体と他集団内の個体が交換される移住という処理がある．どの集団からどの集団へ移住を行うかを決定する集団間構造が探索性能に大きく影響する

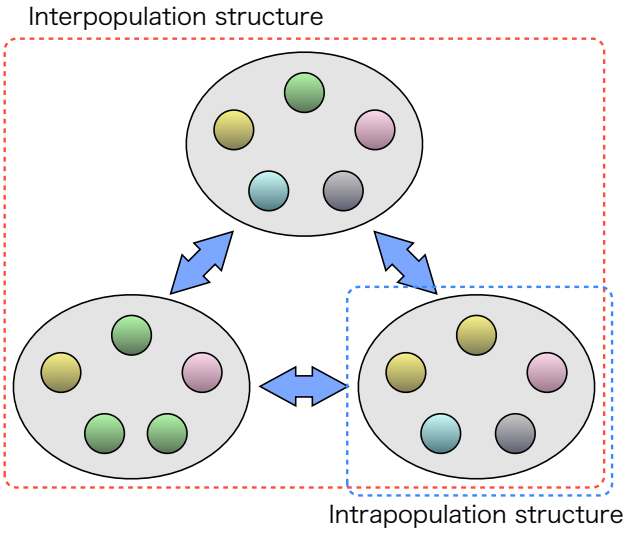


図 1.2　進化集団の構造

ことから，様々な構造が提案されている [15, 16].

集団間構造を拡張することで進化的計算の適用範囲を拡大した手法としては，共進化アルゴリズム (Coevolutionary Algorithm: CA)[17] がある．この手法では集団内の個体は他集団内の個体との相互作用に基づいて評価され，その評価結果によって各集団が共に進化してゆく．このような構造により，ゲームの戦略の進化など，解を相対的にしか評価できない問題にうまく適用することができる．

集団内構造を拡張した手法としては，集団内の個体を複数のサブ集団 (ニッチ，niche) に分けて探索を行うニッチング法 (niching) が広く研究されている [18, 19]．実応用においては多峰性を有する関数を最適化しなければならないことが多く，探索領域内の複数の最適解を網羅的に探索できるこのような手法がよく用いられる．また多目的最適化問題においてはパレート最適解と呼ばれる解の集合を発見しなければならないため，集団内の個体が各パレート解に分散して配置されるように進化させる必要がある．これを行うための手法も数多く提案されている [20, 21, 22, 23, 24].

集団内構造に関連する GA の理論的解析として，スキーマ定理 [7] が挙げられる．この定理は GA のうち最も基本的とされる単純 GA (simple GA) において，集団が探索空間をどのように探索していくかを示したものである．スキーマとは個体を構成するビットストリングの部分列を定式化したものである．スキーマ定理により，世代が進行するにつれ，集団全体の平均評価値よりも評価値の高いスキーマを持つ個体が指数関数的に増加していくことが示されている．このようなスキーマは特別にビルディングブロック [25] と呼ばれる．集団内構造の観点からは，探索空間内での個体同士の位置関係は共通のスキー

マをどの程度有しているかに対応付けられる．特に空間上において近くに位置する個体同士は，共通のスキーマを多く持っているといえる．スキーマが集団内で増殖し全個体が共通のスキーマを有するようになったとき，集団内の個体が一点に集まるような集団構造となる．

1.3　本研究の目的

本研究では進化する集団の集団間構造と集団内構造に基づき，進化的計算手法の拡張を行うことを目的とする．また，単純な最適化ではない問題クラスを対象とした効果的な手法を開発することで，進化的計算の適用可能範囲を拡大することを目指す．

まず集団間構造に着目し，複数の集団が密に相互作用し共に進化していく共進化アルゴリズム (CA) の改良を行う．一般的な進化的計算手法が対象とする問題では，各候補解の評価値はあらかじめ定められた目的関数に基づいて算出されるが，CA では各個体間の相互作用に基づいて評価値が算出される．このような構成により，通常の進化的計算手法を適用しづらい問題クラスを取り扱うことができるため，この手法の性能を向上することで進化的計算の適用範囲を拡大できると考えられる．

次に集団内構造に着目し，各個体が探索空間中の複数の最適解に対して均等に配置されるよう進化させる，多峰性関数の最適化を行う手法を開発する．実問題の多くは多峰性を有する関数最適化問題と考えられ，その場合，複数の最適解を正確に発見することが求められる．このことから，効果的な計算手法を開発することは応用面からも重要である．

1.4　本論文の構成

本論文の構成は次の通りである．

第 2 章では，集団間構造に着目し，共進化アルゴリズム (CA) の改良について述べる．本章では，確率モデルを用いた進化的計算手法である確率モデル構築型遺伝的アルゴリズム (Probabilistic Model-Building Genetic Algorithm: PMBGA) を CA に取り入れたアルゴリズムを，確率モデル構築型共進化アルゴリズム (Coevolutionary Algorithm with Probabilistic Model-Building: CA-PMB) と名付けることにする．その一実装として PMBGA の最も基本的な手法である PBIL (Population-Based Incremental Learning) を CA の枠組みに取り入れた CA-PBIL を示し，CA のベンチマークとしてよく用いられる非推移的ナンバーズ・ゲームを用いて性能評価を行う．

第 3 章では，集団内構造に着目し，多峰性関数最適化のための拡張手法として，孤立個体に基づく変異操作を導入した差分進化 (Differential Evolution: DE)[26, 27] を提案す

る．差分進化を用いる理由として，アルゴリズムが単純な処理のみで構成されるため実装と拡張が比較的簡単に行えること，多くの最適化問題において高い性能を示していることが挙げられる．8 つの 2 次元多峰性関数をベンチマークとして用いた計算機実験により，提案手法とこれまでに提案されている拡張手法の性能を比較する．

最後に 4 章で本論文をまとめ，今後の展望について述べる．

第 2 章

確率モデル構築型共進化アルゴリズムの開発

ゲームの戦略を進化させるような応用例では，戦略に対する絶対的な評価値を定めることは難しいため，相対評価のみに基づいて進化を行う共進化アルゴリズム (Coevolutionary Algorithm: CA) が提案されている．CA は集団間構造を拡張した手法であり，集団内の個体は他集団内の個体との相互作用に基づいて評価される．

本研究では，確率モデルを用いた進化的計算手法である確率モデル構築型遺伝的アルゴリズム (Probabilistic Model-Building Genetic Algorithm: PMBGA) [28] を CA に取り入れたアルゴリズムを，確率モデル構築型共進化アルゴリズム (Coevolutionary Algorithm with Probabilistic Model-Building: CA-PMB) と呼ぶことにする．PMBGA は解の生成確率を表現する確率モデルを保持し，これを更新しながら解空間を探索する手法の総称である．この手法の利点として，変数間の関係を明示的に取り扱った効果的な探索が行えることが挙げられる．よって，これを CA に組み込むことにより性能を向上させることができると考えられる．

本章では，PMBGA の最も基本的な手法である PBIL (Population-Based Incremental Learning) を CA の枠組みに取り入れた CA-PBIL (Coevolutionary Algorithm with Population-Based Incremental Learning) を実装し，CA のベンチマークとしてよく用いられる非推移的ナンバーズ・ゲームを用いて挙動の分析を行う．計算機実験により，進化の停滞や評価値の低い解への収束など，ゲームの非推移性に起因する不適切な共進化が起こりうることを示す．さらに，各集団の学習率が異なるように設定することにより，非推移性を応用した望ましい進化挙動が得られることを示す．

2.1　共進化アルゴリズム

共進化アルゴリズム (CA) は生態系における捕食被食，共生，寄生などのさまざまな共進化現象に着想を得た進化的計算の拡張手法である．予め定められた目的関数を用いて個体の評価を行うのではなく，複数の個体の相互作用に基づいて評価が行われる点に特徴がある．

一般的な進化的計算手法を効果的に適用できないような問題を，CA を用いることでうまく解決できる場合が多くある．一例として，クラス分類問題などのような解評価に訓練事例を要する問題が挙げられる．このような問題では，事例データの総数が膨大である場合，すべてのデータを用いた目的関数値の算出は計算コストが大きくなる．また，算出に用いるデータの数を限定するにしても，効果的な探索に必要となるデータを選び出すことが必ずしも容易ではない．これに対して CA では，問題に対する解を最適化するだけでなく，解を評価する訓練事例データの最適化も同時に行う．これにより，訓練事例の集団はより難しい事例データとなるように最適化され，解の集団はそのようなデータ集団に対しても適切に処理できるような強固な集団へと最適化される，というように適切な処理が行われる．

CA はまずソーティングネットワークの設計問題に適用され，遺伝的アルゴリズムを用いて作成されたネットワーク構成に比べて比較器の少ない効率的な構成を設計することに成功した [17]．この他にも，クラス分類問題に対するニューラルネットワークの進化 [29] や，ゲームプレイヤーの戦略の進化 [30]，移動ロボットにおける追跡・逃走行動の共進化 [31] などに応用されている．

標準的な CA は，遺伝的アルゴリズム (GA) の構成を参考に実装されている．GA では，候補解の集合を保持し，各解が評価値の高いものとなるように集合内の解を変更して

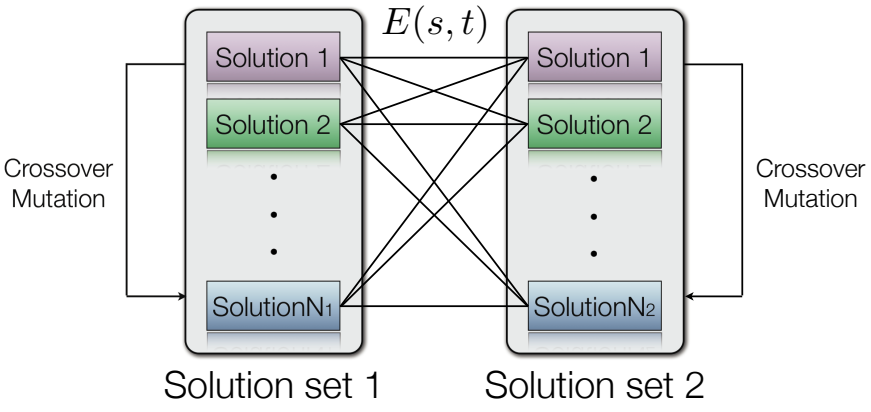


図 2.1　共進化アルゴリズムの構成

いく．変更の操作としては，集合内から評価値の高い解を複数選択し，その個体の情報をもとに新たな解を作り出す交叉や，集合内の解に微小な改変を行うことで新たな解を作り出す突然変異，個体集団から評価値の低い解を削除する選択処理がある．

CA と一般的な GA の違いは，解の評価値を求める際に予め定められた目的関数が用いられるのではなく，他の個体との相互作用によって得られる利得に基づいて評価値が算出されることである．本研究で対象とする CA の構成を図 2.1 に示す．この構成では 2 つの解集合 S^1 および S^2 を用いる．集合内の解 s は他方の集合内にある全ての解との相互作用に基づいて評価される．解 s の評価値 $F(s)$ は以下の式で与える．

$$F(s) = \begin{cases} \sum_{t \in S^2} E(s,t) & \text{if } s \in S^1 \\ \sum_{t \in S^1} E(s,t) & \text{if } s \in S^2 \end{cases} \tag{2.1}$$

ここで，$E(s,t)$ は解 s が解 t との相互作用により得られる利得であり，適用する問題に基づいて定められる．一般的な進化的計算では $F(s)$ が問題によって直接定義されるため，この点において CA と異なる．

クラス分類問題のような訓練事例データを用いるような応用例では，S^1 を分類器集合，S^2 を訓練事例集合とする．$s_1 \in S^1, s_2 \in S^2$ とすると，$E(s_1, s_2)$ は分類器 s_1 が訓練事例 s_2 を正しく分類できるかどうかを表し，正しく分類できた場合は 1，できなかった場合は 0，というように設定する．反対に $E(s_2, s_1)$ は，正しく分類できた場合は 0，できなかった場合は 1，というように設定する．このように利得を定義すると，$F(s_1)$ は事例集合 S^2 のうち正しく分類できる事例の数となり，値が大きいほど性能の高い分類器であるといえる．$F(s_2)$ は分類器集合 S^1 のうち s_2 を正しく分類できない解の数となり，値が大きいほど分類が難しい事例であるといえる．

ゲームへの応用例ではプレイヤーがとる戦略を解とみなし，$E(s,t)$ は戦略 s に基づいて行動するプレイヤーが戦略 t に基づいて行動するプレイヤーと対戦した時の勝敗，あるいは得られる点数を表す．

集団の構造として，本研究では 2 つの進化集団が相互作用する構造を用いた．クラス分類問題における分類器集団と訓練事例集団のように，多くの応用例では各集団が非対称性を持つため，このような構造を採用することが一般的である．ゲームの戦略を進化させるような例では単一の集団のみを用いることも可能である．この場合，各個体の評価値は集団内の他個体と対戦を行うことで算出される．しかしながら，単一集団では適切な進化挙動が得られなくなる場合があることが報告されている [32]．以上を踏まえ，本研究では 2 つの進化集団構造を用いることとした．

2.2　確率モデル構築型遺伝的アルゴリズム

確率モデル構築型遺伝的アルゴリズム (PMBGA) は分布推定アルゴリズム (Estimation of Distribution Algorithm: EDA) とも呼ばれる GA の拡張手法である．GA では新たな解は，交叉処理 (Crossover) と突然変異処理 (Mutation) により生成される．PMBGA ではこれらの処理が，解の生成確率を表現する確率モデルの構築と構築したモデルに基づく解の生成に置き換えられている．この手法の利点として，変数間の関係を明示的に取り扱った効果的な探索が行えることが挙げられる．

代表的な手法として，各変数値の割り当て確率を表す確率ベクトルの値を学習する Population-Based Incremental Learning (PBIL)[33]，compact GA (cGA) [34]，Univariate Marginal Distribution Algorithm (UMDA) [35] や，2 変数間の条件付き確率が連なったモデルを用いる Mutual Information Maximization Input Clustering (MIMIC) [36]，2 変数の同時確率を用い，カイ二乗統計量に基づいて変数間の独立性を判断する Bivariate Marginal Distribution Algorithm (BMDA) [37]，ベイジアンネットワークをモデルとして用いる Bayesian Optimization Algorithm (BOA) [38] などがある．

2.2.1　Population-based incremental learning

PBIL は PMBGA の比較的簡単な実装の 1 つである．図 2.2 にアルゴリズムの概念を示す．このアルゴリズムでは解をビットストリングとして扱う．さらに，確率モデルとしては各ビットに 1 が割り当てられる確率を表す確率ベクトル $\boldsymbol{p}$ を用意し，この値を更新してゆく．以下では，$p_i (i = 1, \ldots, L)$ を確率ベクトル $\boldsymbol{p}$ の i 番目の要素の値とする．p_i は

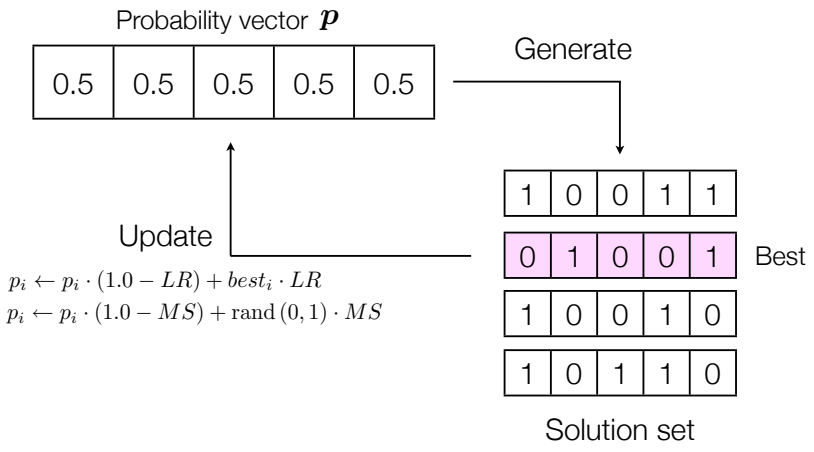


図 2.2　PBIL の概念

$p_i \leftarrow 0.5, i = 1, \ldots, L$

while Termination condition not met **do**

  $S \leftarrow N$ solutions generated by $\boldsymbol{p}$

  $best \leftarrow \mathrm{argmax}_{s \in S} F(s)$

  $p_i \leftarrow p_i \cdot (1.0 - LR) + best_i \cdot LR, i = 1, \ldots, L$

  for $i = 1, \ldots, L$ **do**

    if rand $< MP$ **then**

      $p_i \leftarrow p_i \cdot (1.0 - MS) + \mathrm{rand}(0, 1) \cdot MS$

    end if

  end for

end while

図 2.3　PBIL の処理手順

$[0, 1]$ の値をとり，生成される解の i ビット目に 1 が割り当てられる確率を表す．L は解のビット長である．

図 2.3 に PBIL の処理手順を示す．まず，確率ベクトルの各要素は 0.5 に初期化される．その後，計算時間や反復回数などによって定義される終了条件を満たすまで以下のように探索処理を繰り返す．

まず，確率ベクトル $\boldsymbol{p}$ にしたがって N 個の解を生成する．N は解のサンプル数を決めるアルゴリズムのパラメータである．次に，生成された解集合 S から評価値が最も大きい解 $best$ を取り出し，この解の情報に基づいて確率ベクトルの値を更新する．確率ベクトルの要素 p_i は以下の式に従って更新される．

$$p_i \leftarrow p_i \cdot (1.0 - LR) + best_i \cdot LR \tag{2.2}$$

ここで，$best_i$ は生成した解集合の中で最も高い評価値を持つ解の，i ビット目の値である．LR は学習率と呼ばれる学習の速度を決めるパラメータであり，$(0, 1]$ の値を取る[*1]．

最後に，各ビットごとに確率 MP で突然変異と呼ぶ処理を行う．これは確率ベクトルの値をランダムに変化させることで，局所解への収束を防ぐものである．この処理では確率ベクトルの要素 p_i は以下の式に従って更新される．

$$p_i \leftarrow p_i \cdot (1.0 - MS) + \mathrm{rand}\,(0, 1) \cdot MS \tag{2.3}$$

[*1] 文献 [33] では最も低い評価値をもつ解の情報も活用した更新式が示されているが，本論文では最良解のみを用いて更新を行う実装を用いる．

ここで，rand(0, 1) は 0 または 1 を等確率で出力する乱数である．MS は p_i の更新量を決めるパラメータであり $(0, 1]$ の値を取る．

2.3　提案手法: 確率モデル構築型共進化アルゴリズム

本研究では，GA に基づく処理を行う一般的な CA の構成を，PMBGA に基づく確率モデル構築型の構成に変更した，確率モデル構築型共進化アルゴリズム (CA-PMB) を提案する．CA-PMB の構成を図 2.4 に示す．CA-PMB では PMBGA と同様に確率モデルを保持し，モデルに従って個体を複数生成し，生成された個体の情報に従って評価値の高い個体の生成確率が上昇するようにモデルの更新を行う．PMBGA の特徴として，変数間の依存関係を明示的に取り扱った効果的な探索が行えることが挙げられる．また CA においては，表現力の高いモデルを用いることにより，他集団の構成を予測しそれに対抗する個体を効率的に生み出すようなモデルが構築されることが期待でき，これにより相乗的に性能を向上できると考えられる．

本論文では CA-PMB の一実装として，探索手法として PBIL アルゴリズムを用いる CA-PBIL を示す．図 2.5 に CA-PBIL の処理手順を示す．このアルゴリズムでは解集合 S^1，S^2 を生成するためにそれぞれ確率ベクトル $\boldsymbol{p}^1$，$\boldsymbol{p}^2$ を用意し，これを用いて解の生成を行う．基本となる処理手順は図 2.3 に示す PBIL アルゴリズムに基づいており，各解の評価を式 (2.1) に従って行う．

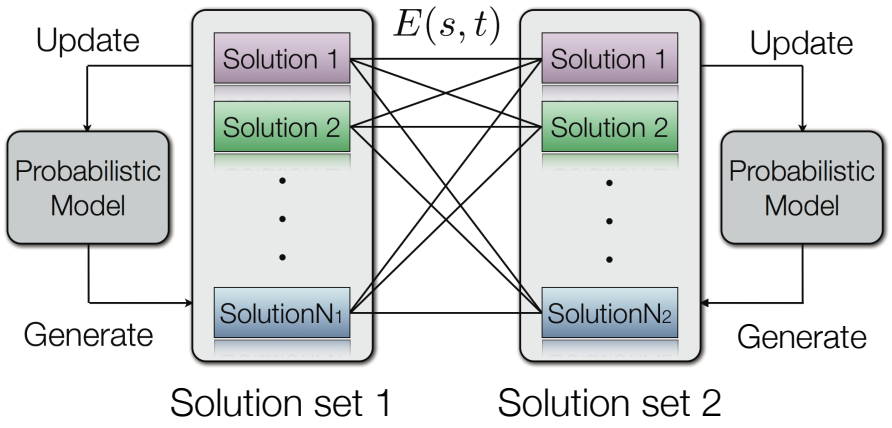


図 2.4　確率モデル構築型共進化アルゴリズムの構成

$p_i^1 \leftarrow 0.5, i = 1, \ldots, L_1$

$p_i^2 \leftarrow 0.5, i = 1, \ldots, L_2$

while Termination condition not met **do**

  $S^1 \leftarrow N_1$ solutions generated by $\boldsymbol{p}^1$

  $S^2 \leftarrow N_2$ solutions generated by $\boldsymbol{p}^2$

  $best^1 \leftarrow \mathrm{argmax}_{s \in S^1} F(s)$

  $best^2 \leftarrow \mathrm{argmax}_{s \in S^2} F(s)$

  $p_i^1 \leftarrow p_i^1 \cdot (1.0 - LR_1) + best_i^1 \cdot LR_1, i = 1, \ldots, L_1$

  $p_i^2 \leftarrow p_i^2 \cdot (1.0 - LR_2) + best_i^2 \cdot LR_2, i = 1, \ldots, L_2$

  for $i = 1, \ldots, L_1$ **do**

    if rand $< MP_1$ **then**

      $p_i^1 \leftarrow p_i^1 \cdot (1.0 - MS_1) + \mathrm{rand}(0, 1) \cdot MS_1$

    end if

  end for

  for $i = 1, \ldots, L_2$ **do**

    if rand $< MP_2$ **then**

      $p_i^2 \leftarrow p_i^2 \cdot (1.0 - MS_2) + \mathrm{rand}(0, 1) \cdot MS_2$

    end if

  end for

end while

図 2.5　CA-PBIL の処理手順

2.4　性能評価実験

2.4.1　非推移的ナンバーズ・ゲーム

本論文では，2 人対称ゼロ和ゲームである非推移的ナンバーズ・ゲーム (Intransitive numbers game) [39] を用いて CA-PBIL の挙動を分析する．非推移的ナンバーズ・ゲームは Watson と Pollack によって提案された，問題の非推移性が共進化アルゴリズムに及ぼす影響を調べるための抽象モデルである．非推移性とは解の強弱関係に推移律が成立しない性質をいう．代表的な例としてジャンケンゲームが挙げられる．ジャンケンではグーはチョキに勝ち，チョキはパーに勝つが，グーはパーに負けるため推移律が成立しない．

このような性質は進化の堂々巡りを発生させ，解の適切な共進化を妨げる．

ナンバーズ・ゲームをベンチマーク問題として用いる理由には，単純な問題で分析がしやすいこと，ベクトルの全成分の和を解の絶対的な良さを表す指標にできることが挙げられる．このことから，共進化アルゴリズムの比較評価や分析を行うためによく用いられている [39, 32].

2.4.1.1　定義

ナンバーズ・ゲームの解は n 次元のベクトルで表され，ベクトルの各要素は区間 $[0, k]$ の値を取る．さらに，解 s が解 t との相互作用により得られる利得 $E(s,t)$ を，以下のように定義する．

$$E(s,t) = sign\left(\sum_{i=1}^{n} g_i\right) \tag{2.4}$$

ただし，

$$g_i = \begin{cases} s_i - t_i & \text{if } h_i = \min_j h_j \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

$$h_i = |s_i - t_i| \tag{2.6}$$

とする．関数 $sign$ は入力が正の数ならば 1 を，負ならば -1 を，0 ならば 0 を出力する．

利得 $E(s,t)$ の定義より，ナンバーズ・ゲームにおける 2 つの間の利得は，差が最小である次元に注目し，その次元の値が大きい解が 1 (勝ちと呼ぶ)，小さい解が -1 (負けと呼ぶ) となり，両解共に同じ値ならば 0 (引き分けと呼ぶ) となる．ベクトルの各要素の値が大きいほどより多くの解から利得を得ることができるため，最大値 k を各要素に持つベクトル $(k, k, \ldots, k)$ が最適解である．また，ナンバーズ・ゲームでは推移律が成立しない．たとえば 2 次元ナンバーズ・ゲームにおける 3 つの解 $(3,6)$，$(1,5)$，$(4,4)$ を考えると，$(3,6)$ は $(1,5)$ に勝ち，$(1,5)$ は $(4,4)$ に勝つが，$(3,6)$ は $(4,4)$ に負ける．

上記のゲームに加えて，利得 $E(s,t)$ を以下のように変更したゲームも用いる．

$$E(s,t) = step\left(\sum_{i=1}^{n} g_i\right) \tag{2.7}$$

関数 $step$ は入力が 0 より大きければ 1 を，0 以下であれば 0 を出力する．以下ではこの利得関数を用いたゲームを改変型ナンバーズ・ゲームと呼び，元の定義によるゲームを原型ナンバーズ・ゲームと呼ぶ．

図 2.6 に 2 次元ナンバーズ・ゲームにおける勝敗を決定する領域を示す．この図は中央の座標に位置する解がどの領域上の解に勝つかを表している．中央の解は白色の領域にある解には勝ち，利得 1 を得る．灰色の領域にある解には負け，原型ナンバーズ・ゲームで

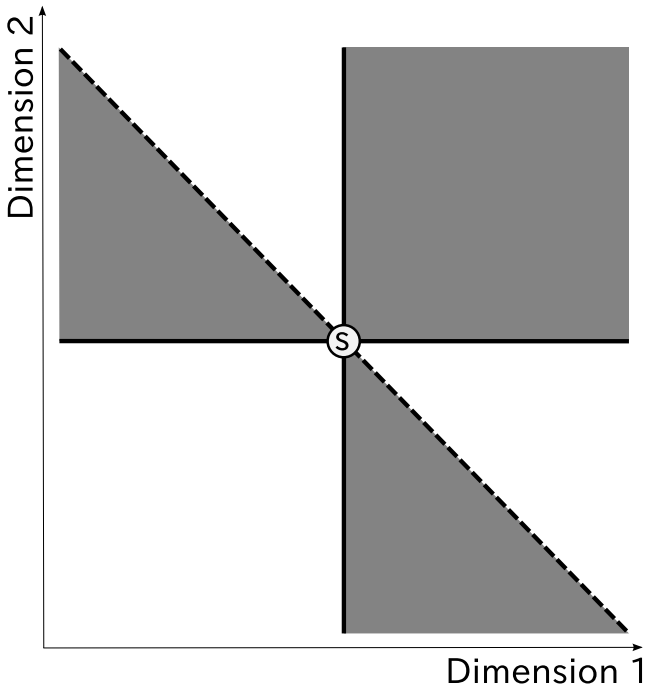


図 2.6　勝敗領域

は利得 -1 を，改変型ナンバーズ・ゲームでは利得 0 を得る．黒色の実線にある解とは引き分けとなり，利得 0 を得る．また，黒色の破線上にある解との勝敗はランダムに決定される．

2.4.1.2　ビットストリングによる解の表現

PBIL はビットストリングで表される解を扱うアルゴリズムであるため，ナンバーズ・ゲームに適用するためにはこれをベクトル表現に変換する方法を定義する必要がある．本研究では以下のような変換方法を用いる．次元数 n，要素の最大値 k のナンバーズ・ゲームに対する解を表現する場合を考える．このときは解のビット長を $L_1 = L_2 = nk$ とする．そして各次元の値は，1 ビット目から k ビット目までの総和を第 1 次元の値，$k+1$ ビット目から $2k$ ビット目までの総和を第 2 次元の値，というように割り当てる．

例として，次元数 $n = 2$，最大値 $k = 10$ のナンバーズ・ゲームを考える．この場合は解のビット長 $L_1 = L_2 = 20$ とし，1 ビット目から 10 ビット目までの総和が第 1 次元の値となり，11 ビット目から 20 ビット目までの総和が第 2 次元の値となる．

なお，本論文では次元数 $n = 2$，最大値 $k = 100$ のゲームに対して CA-PBIL を適用し，その挙動を分析する．この場合，解を表すビットストリングの長さは $L_1 = L_2 = 200$ となる．

2.4.2　挙動パターンの分析

まず，生成される解の平均値がどのように推移するか調べるために以下のように各パラメータを設定して計算機実験を行った．学習率 $LR_1 = LR_2 = 0.01$，サンプル数 $N_1 = N_2 = 100$ とし，突然変異確率は $MP_1 = MP_2 = 0.0$ として処理を行わない設定とした．反復回数を 10000 回とし，100 回の試行を行った．この実験では，各集合 S^1 および S^2 に含まれる解の平均値の推移を調べる．

2.4.2.1　原型ナンバーズ・ゲーム

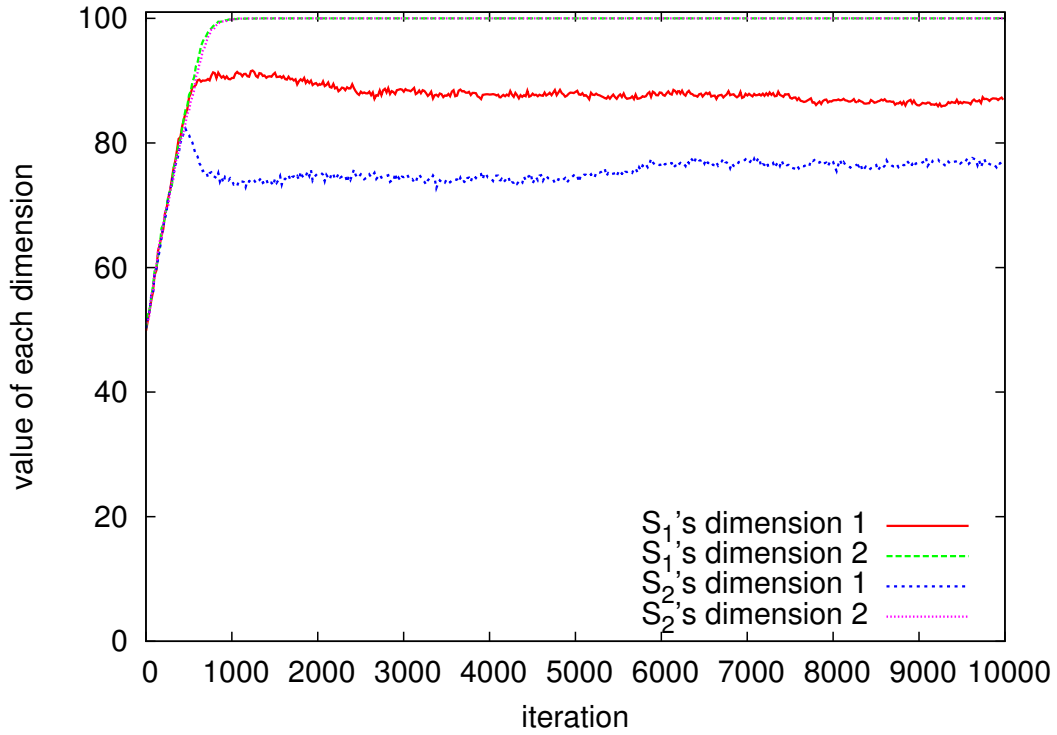
まず，原型ナンバーズ・ゲームにおいて現れる挙動パターンの分析を行った．このゲームにおいては，片方の次元の進化が停滞するパターンと，低い値へ収束するパターンが現れた．図 2.7 にそれぞれのパターンの例を示す．

図 2.7 (a) に進化が停滞するパターンの例を示す．序盤では各次元の値が徐々に大きくなっていくため，適切な共進化が起こっているが，中盤になると片方の次元において値が上昇しなくなる．これは過度の特殊化 (Over-specialization) [39] と呼ばれる現象である．一方の次元の差が他方に比べて小さくなると，差の小さい次元のみを用いて利得の決定が行われ，差の大きい次元が利得に影響しなくなる．よって差の小さい次元は進化するが，大きい次元の進化は停滞してしまう．

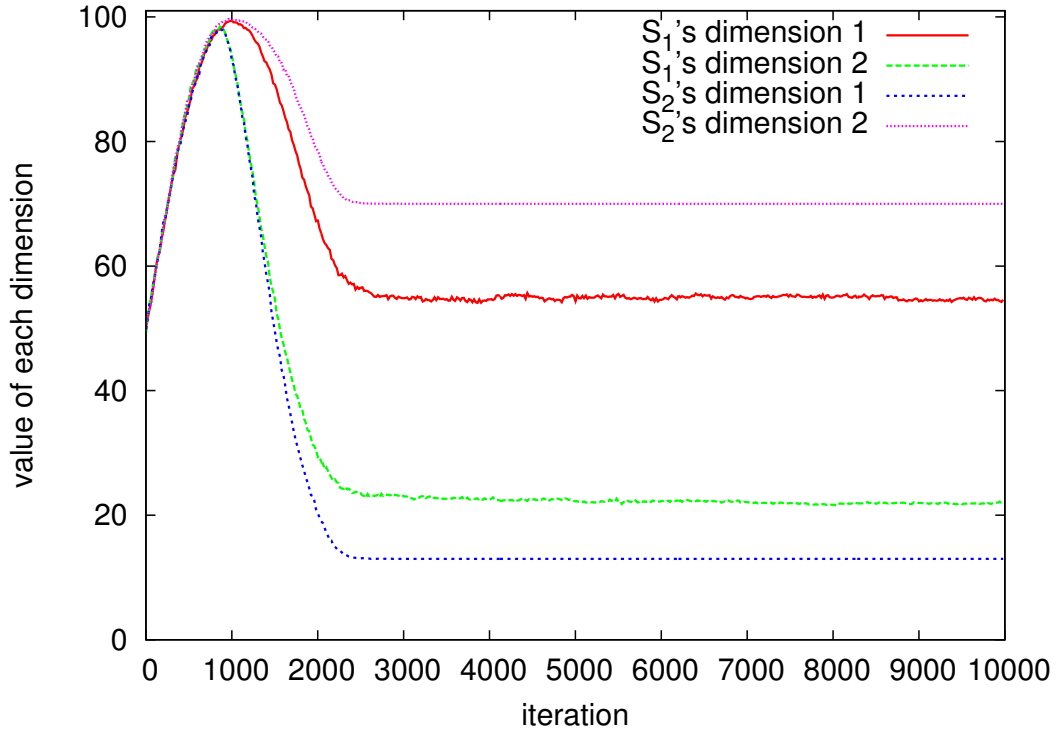
図 2.7 (b) に低い値への収束現象の様子を示す．このような挙動は，解の評価が他の解との相対的な指標に基づいて行われることが原因となって発生することから，相対主義 (Relativism) と呼ばれている [39]．この原因は次のように説明できる．例として，$a = (4, 7)$ と $b = (5, 5)$ の 2 つの解を考える．差の小さい次元は第 1 次元であるので，b が正の利得を得る．ここで，$a' = (3, 6)$ という a から変化した解を考える．a' と b を比べると差の小さい次元は第 2 次元であるので，a' が正の利得を得る．よって，a は a' に置き換わるべきである．その後，$b' = (4, 4)$ という解は a' との対戦により正の利得を得ることができるので，b は b' に置き換わるべきである．これを繰り返すことにより，それぞれの解は低い値へと進化していく．

2.4.2.2　改変型ナンバーズ・ゲーム

次に，改変型ナンバーズ・ゲームにおいて現れる挙動パターンの分析を行った．図 2.8, 2.9, 2.10, 2.11 に，生成される解の推移パターン例を示す．これらは最終的に現れる挙動ごとにわけて，それぞれの例を示した．平均値の推移を見ると，その挙動は探索の序盤・中盤・終盤で異なり，次のように 3 段階に分けることができる．



(a) 進化の停滞



(b) 低い値への収束

図 2.7　原型ナンバーズ・ゲームにおける推移パターン例

1. 適切な共進化 (反復回数 500 程度まで)
2. 片方の次元の進化が停滞する (反復回数 500〜2000 程度)
3. 4 つのパターンに分かれる (反復回数 2000〜)
   (a) 2. の状態が続く
   (b) 停滞していた成分が低い値に収束する
   (c) 停滞していた成分が高い値に収束する
   (d) 停滞していた成分が振動を始める

序盤では各次元の値が徐々に大きくなっていくため，適切な共進化が起こっているといえる．しかし，中盤になると過度の特殊化により片方の次元において値が上昇しなくなる．

その後，終盤では 4 つの挙動パターンが現れる．表 2.1 に各パターンの発生回数を示す．4 つのパターンのうち，進化の停滞と低い値への収束が多く起こっている．また，各パターンが発生するメカニズムを次節以降に示す．

2.4.2.3　進化の停滞

図 2.8 に進化の停滞の様子を示す．このパターンでは，過度の特殊化現象により停滞した次元の評価値がそのまま推移するこの状態では各個体の評価は停滞が起こっていない次元しか用いられなくなる．

2.4.2.4　低い値への収束

図 2.9 に低い値への収束現象の様子を示す．このパターンでは，過度の特殊化現象により停滞していた次元の評価値がある時点から減少しはじめ，最終的には 0 に近い値となる．

この現象が起こるメカニズムは次のように説明できる．図 2.9 において反復回数 1500 回目の部分に注目すると，$\boldsymbol{p}^1$ から生成される解の平均値は $(60, 100)$ であり，$\boldsymbol{p}^2$ では

表 2.1　各挙動パターンの発生回数

パターン	発生回数
過度の特殊化による進化の停滞	38
停滞していた次元が低い値に収束	49
停滞していた次元が高い値に収束	4
停滞していた次元が振動する	9

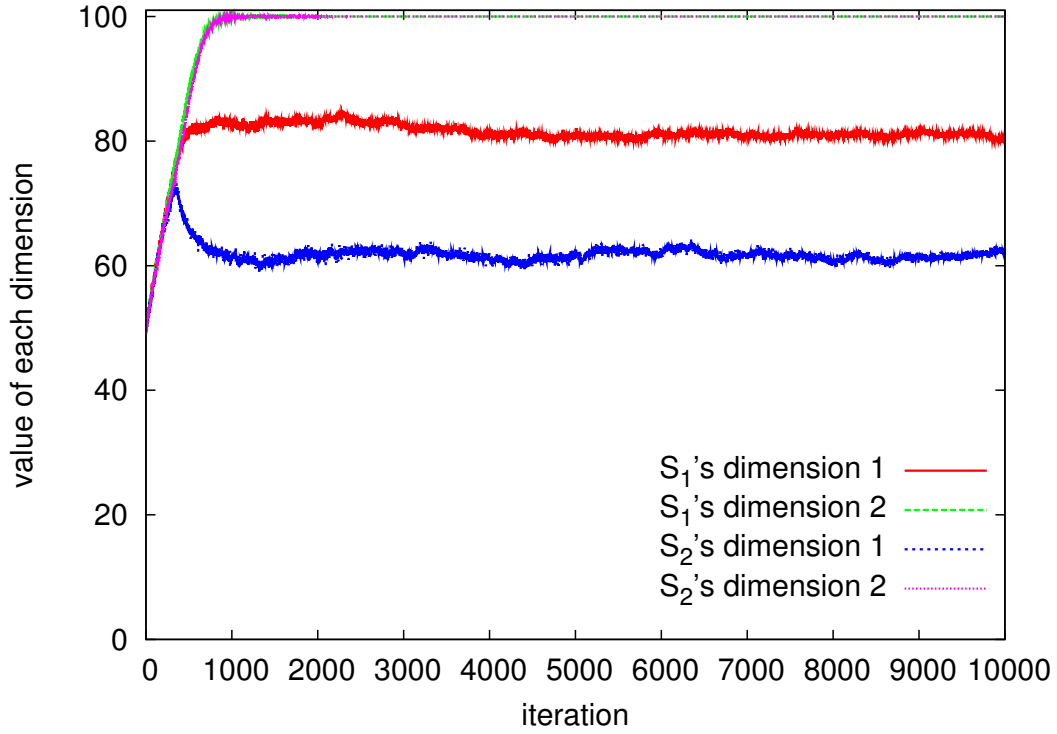


図 2.8　過度の特殊化による進化の停滞

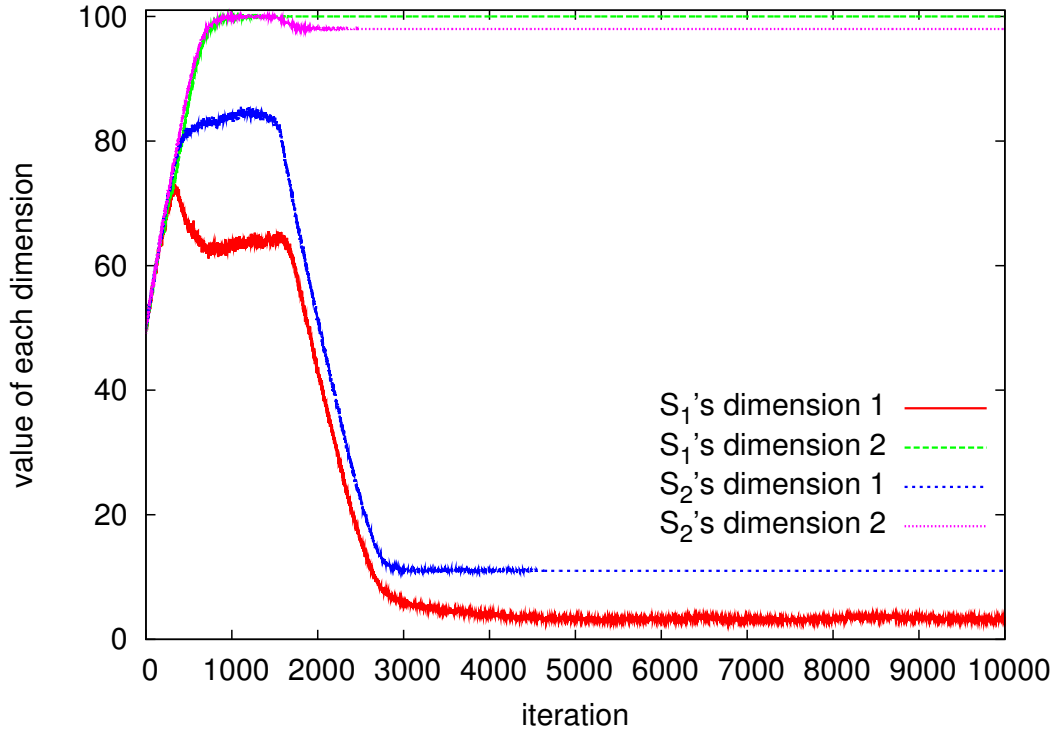


図 2.9　低い値への収束

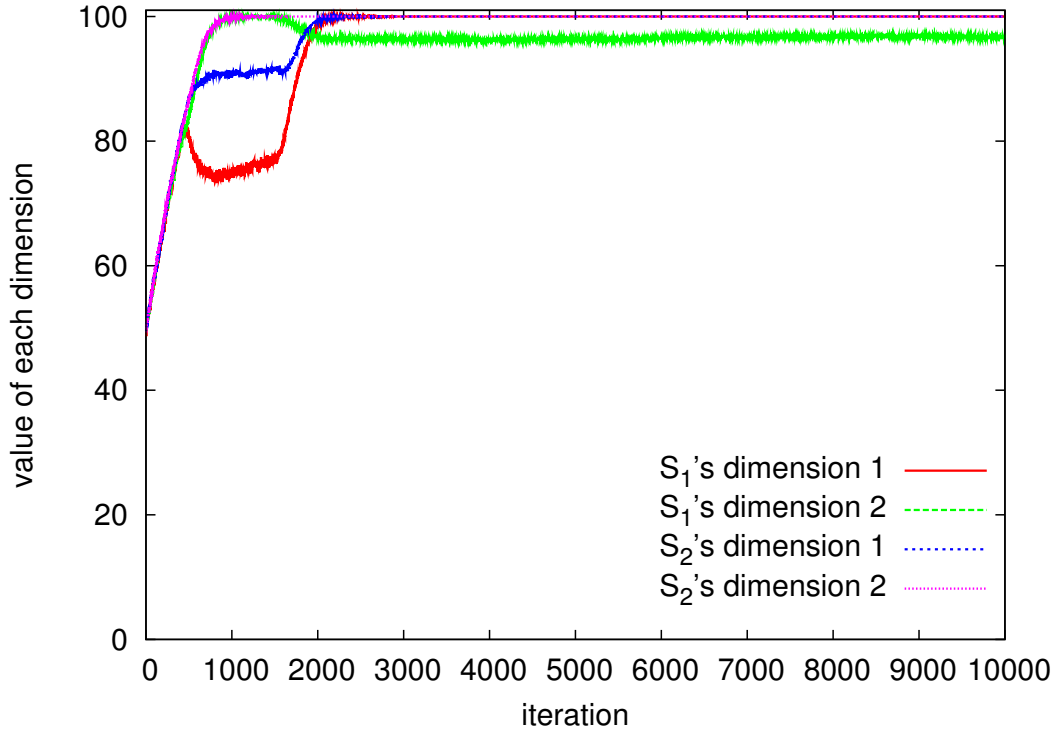


図 2.10　高い値への収束

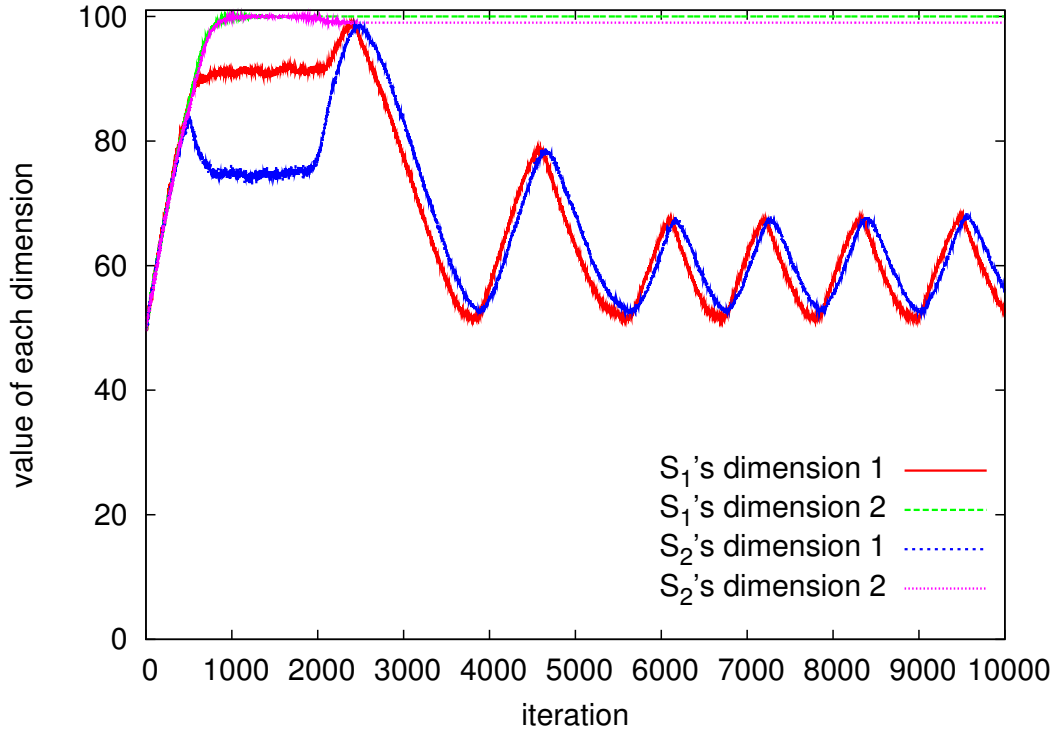


図 2.11　振動

$(80, 100)$ である．よって，第 2 次元で利得の決定が行われ勝敗は引き分けとなる．この状態では，S^2 において $(61, 98)$ のような解が生成されると，第 1 次元で利得の決定が行われるようになるため，この解は S^1 内の解に対して勝利することが期待される．そのため，S^2 においてはそのような値を取るような解が優位となり，確率ベクトルがそのように更新されていく．すると $\boldsymbol{p}^1$ では第 1 次元の期待値 60 がより低い値を取るように確率ベクトルを更新することで，第 2 次元で利得の決定が行われるようにする．

2.4.2.5　高い値への収束

図 2.10 に高い値への収束現象の様子を示す．このパターンでは，過度の特殊化現象により停滞していた次元の評価値がある時点から上昇しはじめ，最終的には 100 に近い値となる．

この現象も相対主義によるものであり，発生のメカニズムは次のように説明できる．図 2.10 において反復回数 1500 回目の部分に注目すると，$\boldsymbol{p}^1$ から生成される解の平均値は $(80, 100)$ であり，$\boldsymbol{p}^2$ では $(90, 100)$ である．よって第 2 次元で利得の決定が行われ，勝敗は引き分けとなる．この状態では，S^1 において $(91, 98)$ のような解が生成されると，第 1 次元で利得の決定が行われるようになるため，この解は S^2 内の解に対して勝利することが期待される．そのため，S^1 においてはそのような値を取るような解が優位となり，確率ベクトルがそのように更新されていく．すると $\boldsymbol{p}^2$ を第 1 次元の期待値 90 がより高い値を取るように確率ベクトルを更新することで，第 2 次元で利得の決定が行われるようにする．

2.4.2.6　振動

図 2.11 に振動現象の様子を示す．このパターンでは，過度の特殊化現象により停滞していた次元の評価値がある時点から振動をはじめる．なお，図 2.11 の例では高い値に移行してから振動しているが，低い値に移行してから振動状態に移ることもある．このパターンでは低い値への収束現象と，高い値への収束現象が交互に起こっていると考えられる．

2.4.3　突然変異処理の効果

ここでは，PBIL における突然変異処理を行った場合，アルゴリズムの挙動にどのよう影響があるか計算機実験により確認する．

まず，突然変異操作のみを繰り返した場合の挙動を調べる．この実験では，長さ $N = 100$ の確率ベクトルを用意し，これに対し式 (2.3) に示す突然変異操作を行い続け

た時の，各ビットの総和の期待値 $\sum_{i=1}^{N} p_i$ の推移を調べる．突然変異量 $MS = 0.005$，反復回数は 10000 回とし，各突然変異確率 MP ごとに 1 試行を行った．確率ベクトルの各要素の初期値は $p_i = 1.0 (i = 1, \ldots, N)$ の場合と，$p_i = 0.0$ の場合について調べた．図 2.12 に結果を示す．確率ベクトルの初期値にかかわらず，反復回数を重ねるごとに期待値が 50 に近づいているのがわかる．このことから，突然変異操作のみを繰り返すと $(k/2, k/2, \ldots, k/2)$ の解に収束するといえる．なお，このような効果は一般的な GA の突然変異においても発生することが知られている [39]．

次に，学習率 $LR_1 = LR_2 = 0.01$，サンプル数 $N_1 = N_2 = 100$，突然変異確率 $MP_1 = MP_2 = 0.001$，突然変異による更新量 $MS_1 = MS_2 = 0.005$，反復回数 50000 とした上で，アルゴリズムを実行した．

図 2.13 に各ゲームにおける挙動パターン例を示す．原型ゲームにおいては停滞パターンが現れる．改変型ゲームにおいては，特定の値へ収束することがなく，動的な挙動を繰り返す．これは突然変異処理により発生したランダム性により，前節で示した 4 つの挙動パターンが様々なタイミングで発生するためと考えられる．

また，突然変異確率を高く設定した場合にはすべての試行において停滞パターンが現れた．この場合，停滞する次元の値は対戦する集団との差を保ちながら 50 に近づいていく．これは，前述の突然変異操作の特性によるものである．図 2.14 に例を示す．なお，この例では突然変異確率 $MP = 0.05$ とした．

2.4.4　異なる学習率での共進化

ここでは，学習率 LR_1 と LR_2 を異なる値に設定した場合の挙動を調べる．まず改変型ナンバーズ・ゲームにおいて，学習率 $LR_1 = 0.01$，$LR = 0.02$，サンプル数 $N_1 = N_2 = 100$，突然変異確率 $MP_1 = MP_2 = 0.0$，反復回数 10000 回とした場合の試行を 100 回行い，第 3 段階における 4 つの各挙動パターンの発生回数を調べた．表 2.2 に結果を示す．このようなパラメータ設定では進化の停滞と高い値への収束のみが発生して

表 2.2　学習率が異なる場合の各挙動パターンの発生回数

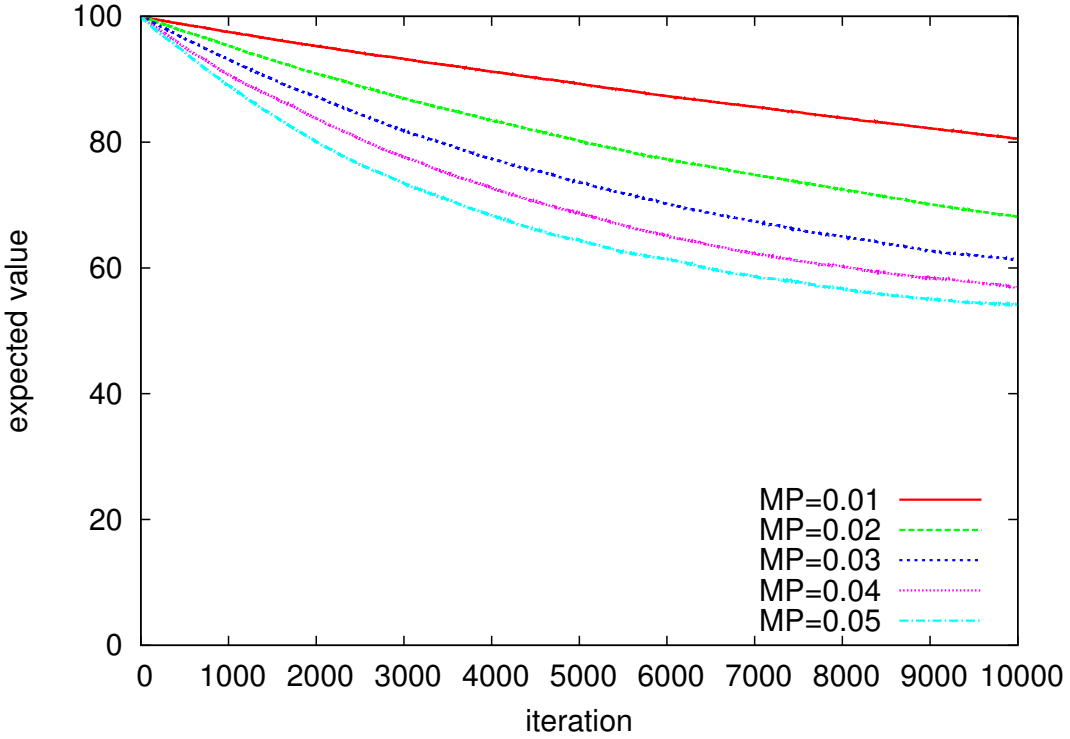
パターン	発生回数
過度の特殊化による進化の停滞	14
停滞していた次元が低い値に収束	0
停滞していた次元が高い値に収束	86
停滞していた次元が振動する	0

いる．特に望ましい挙動である高い値への収束の割合がかなり高い．

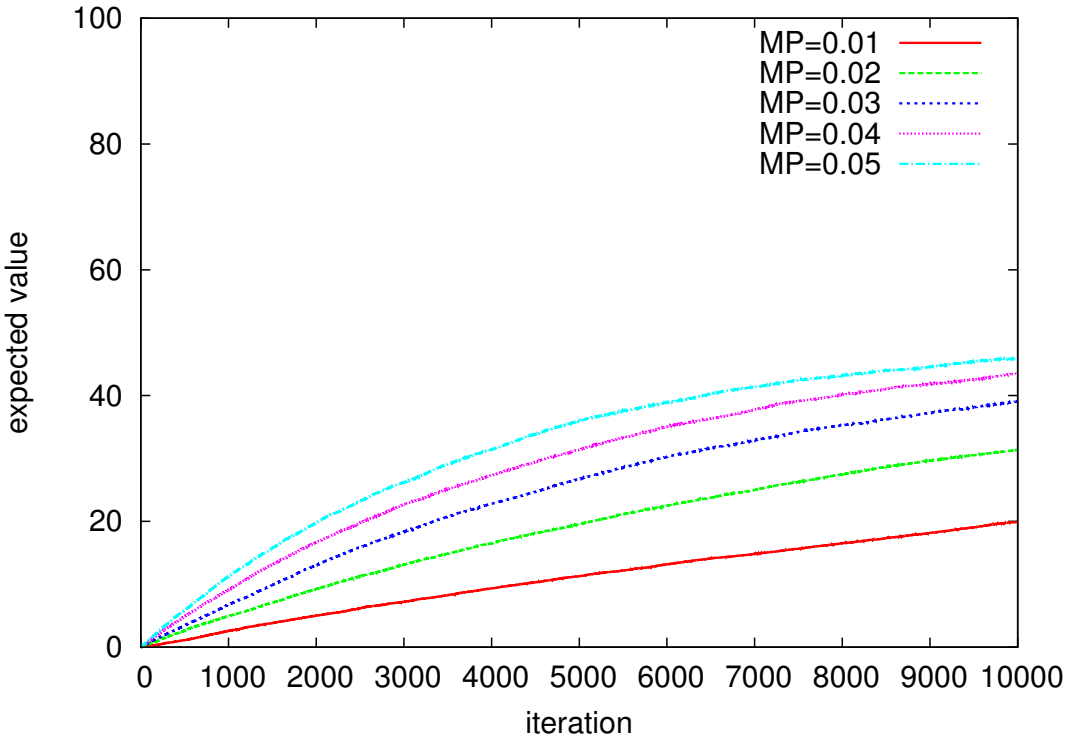
このメカニズムは次のように説明できる．まず，学習率が大きい S^2 の第 1 次元が 100 に収束し，第 2 次元は過度の特殊化現象により停滞したとする．ただし，第 2 次元については学習率が大きいことから S^1 よりも大きい平均値で停滞する．次に，学習率が小さい S^1 の第 1 次元の値が S^2 の進化に遅れて追従し，100 に近い値となる．すると，第 1 次元が 100 に近くなり，かつ第 2 次元では S^2 よりも値が小さい状況になる．この状況は高い値への収束現象を引き起こすものであり，よってこの現象が高い頻度で現れる．

次に，LR_2 を変えた時の探索性能の変化を調べる実験を行った．この実験では，各学習率において 100 回の試行を行ない，最適解 $(100, 100)$ を発見できた試行の回数で性能を評価する．パラメータは学習率 $LR_1 = 0.01$，サンプル数 $N_1 = N_2 = 100$ とし，突然変異率は $MP_1 = MP_2 = 0.0$ とした．学習率 LR_2 は 0.01 から 0.03 まで 0.001 刻みで変化させる．反復回数は 10000 回とした．

図 2.15 に LR_2 に対する最適解発見回数の変化を示す．学習率に差をつけることで最適解発見頻度が向上するが，大きく設定すると低下してしまう．これは学習率が大きいと初期収束を起こしてしまうためである．以上の結果から，異なる学習率を適切に設定することが探索性能を向上させる上で有効であると考えられる．

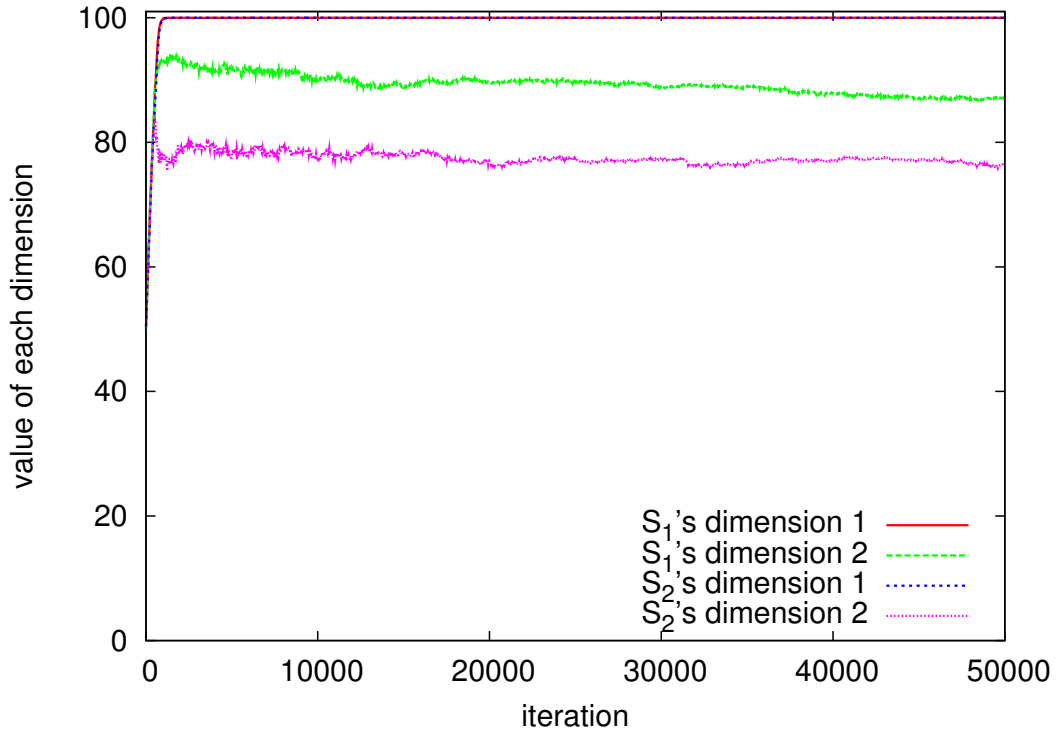


(a) 初期値 1.0

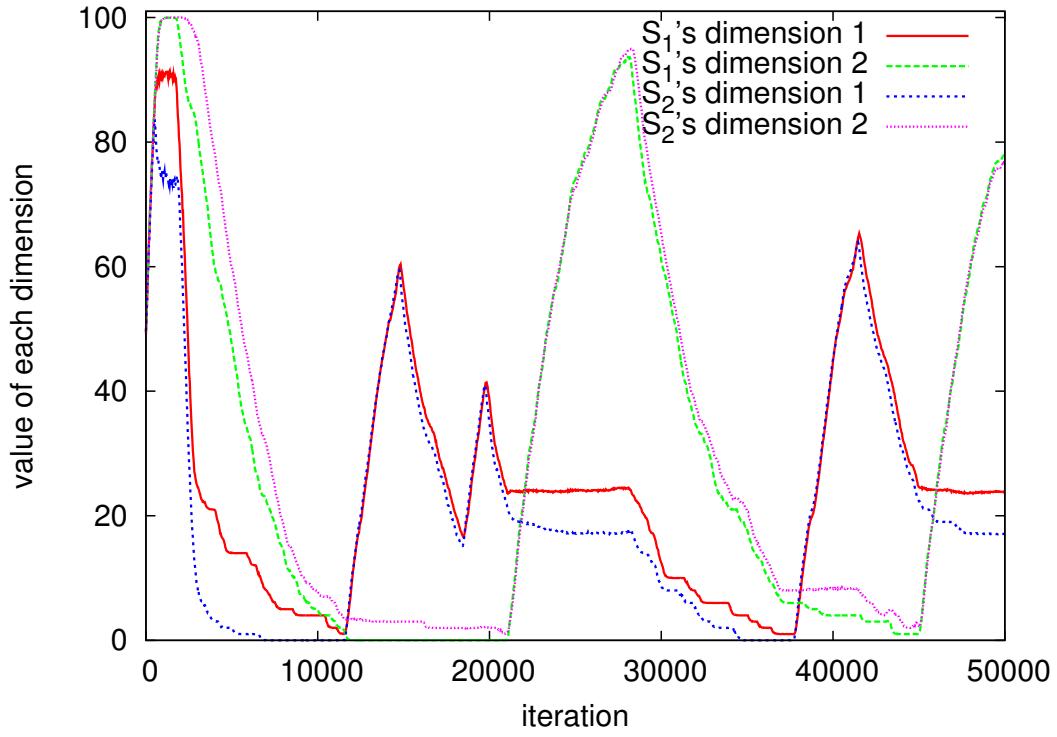


(b) 初期値 0.0

図 2.12　突然変異処理による収束



(a) 原型ナンバーズ・ゲーム



(b) 改変型ナンバーズ・ゲーム

図 2.13　突然変異処理を行った場合の挙動パターン例

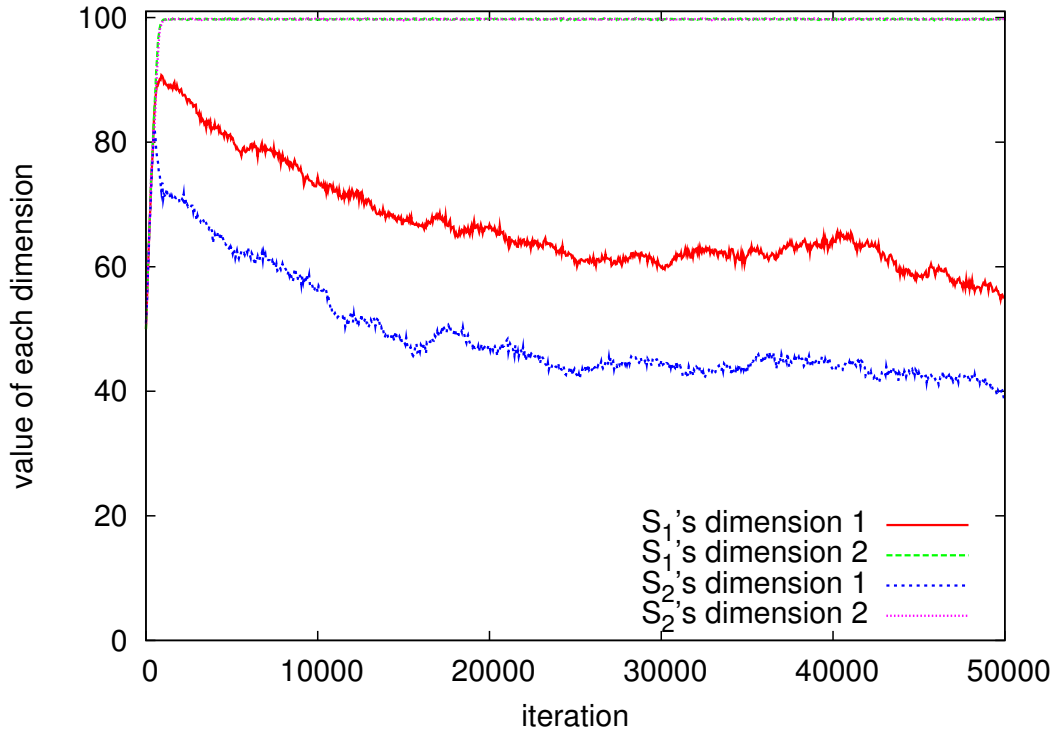


図 2.14　高確率で突然変異操作を行った場合の挙動パターン例

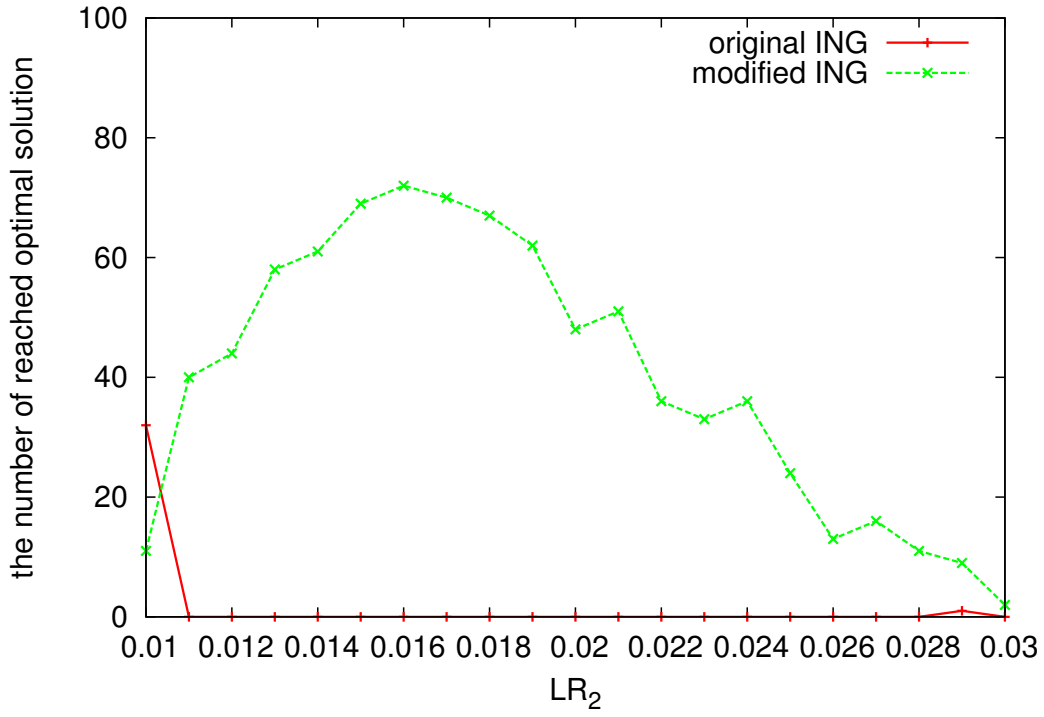


図 2.15　最適解 $(100, 100)$ の発見回数

$S \leftarrow$ Randomly initialized N solutions
while Termination condition not met **do**
  $S' \leftarrow \{\}$
  while $|S'| \neq N$ **do**
    $s_1, s_2 \leftarrow$ 2 solutions sampled with replacement from S using roulette selection
    $s'_1, s'_2 \leftarrow \mathrm{crossover}(s_1,\ s_2)$
    $\mathrm{mutation}(s'_1)$
    $\mathrm{mutation}(s'_2)$
    $S' \leftarrow S' \cup \{s'_1, s'_2\}$
  end while
  $S \leftarrow S'$
end while

図 2.16　遺伝的アルゴリズムの処理手順

2.4.5　通常の共進化アルゴリズムとの比較

本節では遺伝的アルゴリズム (GA) [7, 8, 9] を集団の更新に用いる通常の CA の挙動を分析し，提案手法である CA-PBIL の挙動と比較する．

2.4.5.1　遺伝的アルゴリズムの概要

図 2.16 に GA の処理手順を示す．このアルゴリズムにおいても PBIL と同様に，ビットストリングの個体を扱う．まず，ランダムに生成した N 個体により集団 S を構築する．その後，計算時間や反復回数などによって定義される終了条件を満たすまで以下のように探索処理を繰り返す．

次世代の個体は以下のように生成する．まず，集団 S から評価値に基づいたルーレット選択により，交叉を行う 2 個体 s_1，s_2 を復元抽出する．この際には，評価値をそのまま用いて選択確率を決定することもできる．しかしながら，適切な探索を行うために選択圧を調節することを目的として，元の評価値に適当な変換を行うスケーリング手法を用いることが一般的である．本実験では線形スケーリングを用いた．変換後の評価値を以下のように定義する．

$$F'_S(s) = a_S F(s) + b_S \tag{2.8}$$

ただし，

$$a_S = \frac{c}{\max_{t\in S} F(t) - \min_{t\in S} F(t)} \tag{2.9}$$

$$b_S = 1 - \frac{c}{\max_{t\in S} F(t) - \min_{t\in S} F(t)} \min_{t\in S} F(t) \tag{2.10}$$

とする．$c \geq 0$ はアルゴリズムの制御パラメータであり，選択圧の強さを表している．最小値 0 のときは，集団中の個体をランダムに選択することに対応する．

スケーリングした評価値に基づき，個体 s を次の確率で選択する．

$$p(s) = \frac{F'_S(s)}{\sum_{t\in S} F'_S(t)} \tag{2.11}$$

ルーレット選択により 2 個体を抽出したあと，これらの個体に対して交叉処理を行うことで新たな個体 s'_1，s'_2 を生成する．交叉の方法は，個々のビットを確率 0.5 で選択し，選択されたビットの値を一方の親から受け継ぎ，残りを他方の親から引き継ぐ一様交叉 [40] を用いた．さらに，生成した個体に対して突然変異処理を行う．この処理では，個体の各ビットを一定の確率 MP で反転させる．以上の処理を次世代集団 S' のサイズが N となるまで繰り返し，最後に，集団 S を S' に置き換える．

2.4.5.2　挙動パターンの分析

CA-PBIL の場合と同様に，生成される解の平均値がどのように推移するか調べるために以下のように各パラメータを設定して計算機実験を行った．突然変異確率は $MP_1 = MP_2 = 0.0$ として処理を行わない設定とした．世代数を 10000 とし，100 回の試行を行った．パラメータ c は，選択圧が弱い場合 $(c = 0.1)$ と強い場合 $(c = 4.0)$ について，それぞれ挙動を調べた．

図 2.17 に原型ナンバーズ・ゲームにおける進化挙動の例を示す．なお，各試行ごとに 10000 世代までアルゴリズムを実行したが 1000 世代以降では一定の値に留まるため，この図ではその時点までの推移を示している．CA-PBIL の場合に比べて探索初期の段階で一定の値に留まるが，これは評価値に基づくルーレット選択を繰り返すことで，集団内の多様性が急速に失われるためである．このため選択圧が弱い場合，進化の速度が遅くなるため最適解 $(100, 100)$ 付近に到達する前に一定の値に留まってしまう．$c = 0.1$ の場合，どの試行においても図 2.17 (a) のように停滞する挙動パターンとなった．

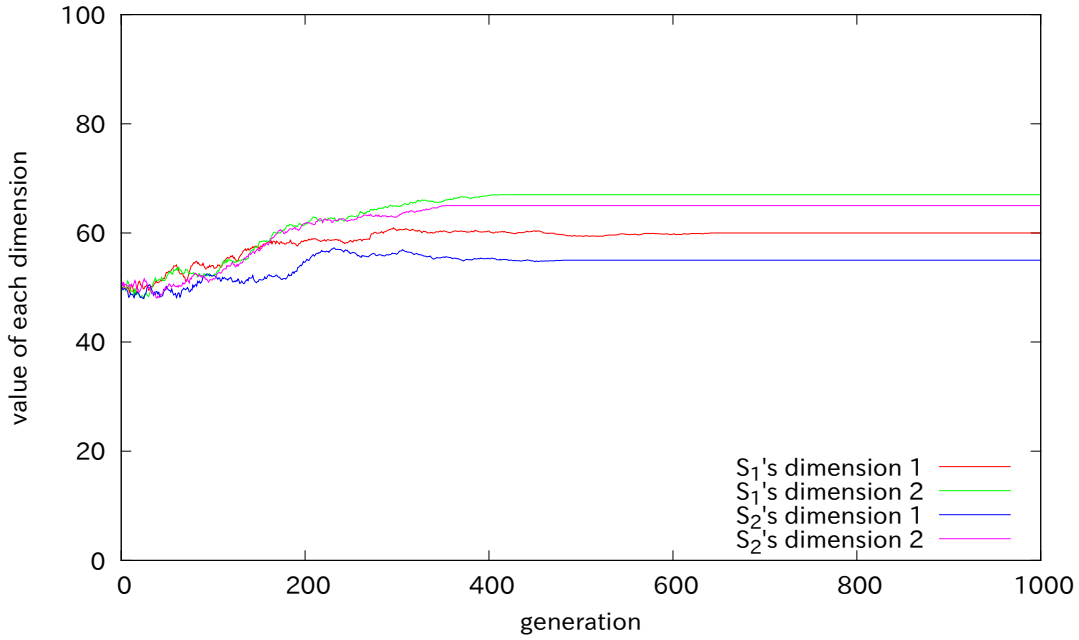
$c = 4.0$ と選択圧が強い場合，CA-PBIL の場合と同様に様々な挙動パターンが現れた．表 2.3 に原型および改変型ゲームにおける各パターンの発生回数を示す．なお，図 2.17 (b) のように 1 試行の間に低い値への収束パターンと高い値への収束パターンの両方が 1 度でも現れた場合，振動パターンとみなした．最終的には前述の選択の効果により，どの試行においても一定の値に留まった．

表 2.3　通常の共進化アルゴリズムにおける各挙動パターン発生回数

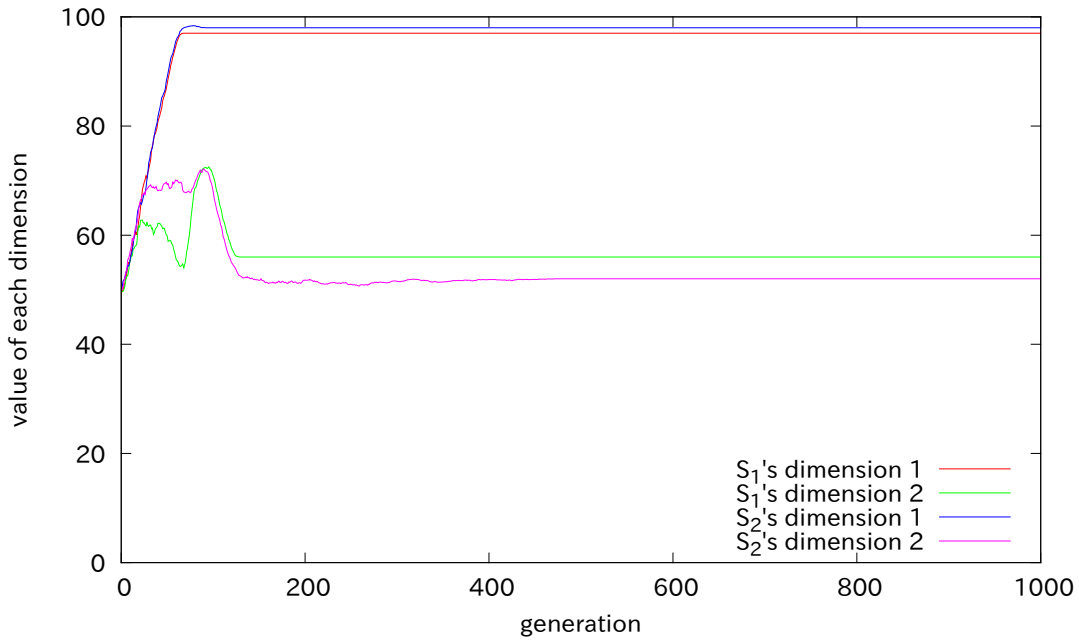
パターン	原型ゲーム	改変型ゲーム
過度の特殊化による進化の停滞	21	21
停滞していた次元が低い値に収束	31	40
停滞していた次元が高い値に収束	32	17
停滞していた次元が振動する	8	7
両方の次元が低い値に収束	8	15

CA では，CA-PBIL に比べて各挙動が一定の割合で発生している．特に原型ナンバーズ・ゲームにおいては，CA-PBIL はほとんどの試行において停滞パターンが現れたが，CA ではその他のパターンも多く現れている．また CA-PBIL ではほとんど現れない，両方の次元が低い値に収束するパターンがどちらのゲームにおいても高い割合で発生している．

また CA を用いた場合，最適解 $(100, 100)$ はどの試行においても発見できなかった．これは前述のとおり，ルーレット選択によって多様性が失われることが大きな要因である．選択圧 c を調整することではこれを回避することは難しい．c が小さい場合，図 2.17 (a) のように停滞してしまう．大きくしすぎると評価値が高い個体ばかりがルーレット選択により抽出されてしまい，同様に多様性は失われてしまう．また，突然変異処理を行うことで多様性を維持することも考えられるが，CA-PBIL と同様に動的な挙動を繰り返したり，停滞することによりうまく探索できない．その他に，集団サイズを大きくすることも多様性維持の方法として挙げられるが，評価値を算出するための計算量が増大するという問題がある．以上により，ナンバーズ・ゲームにおいては通常の CA よりも CA-PBIL の方が容易に最適解を発見でき，効果的であると考えられる．



(a) 選択圧が弱い場合 ($c = 0.1$)



(b) 選択圧が強い場合 ($c = 4.0$)

図 2.17　通常の共進化アルゴリズムの挙動例

2.5 まとめ

本章では PMBGA を CA に組み込んだ，確率モデル構築型共進化アルゴリズム (CA-PMB) を提案し，その一実装として，確率ベクトルをモデルとして用いる PBIL アルゴリズムを CA に組み込んだ CA-PBIL を示した．さらに，2 次元の非推移的ナンバーズ・ゲームを用いてアルゴリズムの挙動を実験的に分析した．実験結果より，GA をベースとした一般的な共進化アルゴリズムにおいて確認されていた過度の特殊化現象による進化の停滞が提案手法においても発生することがわかった．さらに，相対主義の影響による 4 パターンの挙動が現れることを示した．これにより不適切な方向への共進化や，値の振動による進化の停滞が見られた．さらに，2 つの集団で異なる学習率を適切に設定することで，この現象を応用した望ましい共進化を引き起こせることを示した．

今後の課題としては，ナンバーズ・ゲームの次元数を増やすなど広汎な問題設定において，多種のパラメータ設定を試すことに詳細なアルゴリズムの分析を行うことが挙げられる．また本研究では GA に基づく通常の CA との簡単な性能比較を行ったが，その拡張手法など他手法との性能比較や，ソーティングネットワーク設計問題など他の問題を用いた性能評価を行う必要がある．特に，異なる学習率により探索性能を向上させる方法が実問題においても有効であるかについては詳しく調べなければならない．さらには，今回示した CA-PBIL 以外の，より高い探索性能をもつ PMBGA 手法を用いたアルゴリズムを実装することが挙げられる．提案した CA-PMB は表現力の高い確率モデルを用いることにより，相乗的な探索性能向上が期待できる．CA-PBIL で用いているモデルは最も単純なものであるため，BOA[38] などを組込んだより複雑なモデルを用いる CA を実装し，実際にそのような効果が現れるのか評価を行う必要がある．

第 3 章

孤立個体に基づく変異操作を導入した差分進化の開発

本章では単一集団内における個体間の構造に着目し，集団内の個体が探索空間中にある複数の最適解に分散して配置するような進化手法について検討する．このような計算は多峰性関数最適化と呼ばれる．多峰性関数の最適化では解空間中に存在する複数の最適解を網羅的に発見する必要がある．機械学習におけるクラス分類問題 [41] や遠地地震波の解析問題 [42] のような実問題は，多峰性の高い最適化問題であると考えられる．これらは複数の大域的および局所的最適解を持ちえ，多くの場合においてそれらをなるべく多く正確に発見することが望ましい [43].

本研究では進化計算手法の一種である差分進化 (Differential Evolution: DE) [26, 27] に着目し，この手法を基として多峰性関数最適化を行える拡張手法を開発する．差分進化を用いる理由として，アルゴリズムが単純であるため実装と拡張が比較的簡単に行えること，多くの最適化問題において高い性能を示していることが挙げられる．

本章では多峰性関数最適化行うための差分進化の拡張として DE/isolated/1 を提案する．この手法では現在の集団の中で孤立している個体に基づいて新たな個体を生成する．このように設定することで，新しい個体は孤立個体の近くに生成される．その後，新たに生成された個体が現在の個体よりも良好な評価値を持っていれば，その個体に置き換わる．従って，候補解は各世代において集団内で孤立している領域に向かって積極的に移住を行うようになる．これにより，解空間内の各最適解に均等に個体を割り当てられると考えられる．

3.1　差分進化

本節では一般的な差分進化 (DE) の処理手順について説明する．なお以下の記述は文献 [44] を参考にしている．DE の処理手順を図 3.1 に示す．この手法では集団 X_c 内の個体 $x_{i,c}$ を順に選んで処理を行う．選ばれた個体を target 個体と呼ぶ．そして，変異操作と交叉操作によって新しい個体が生成される．その後選択操作によって，新しく生成した個体が target 個体よりも良い評価値を持っていれば，target 個体を新しい個体に置き換える．

ここでは，DE の処理中に扱われるいくつかの個体を以下のように呼ぶ．

- target 個体 $\vec{x}_{i,c}$: 現在の集団 X_c から選ばれた個体．
- donor 個体 $\vec{v}_i$: 変異操作によって生成される個体．
- base 個体: 変異操作の中で差分ベクトルにより摂動を与えられる個体．
- trial 個体 $\vec{u}_i$: donor 個体と target 個体の交叉によって生成される個体．

```
input:
  f(x⃗) - The function to be minimized
  F - Scaling factor
  Cr - Crossover rate
  NP - Population size
procedure:
  X_c ← Randomly initialized vectors
        [x⃗_{1,c}, x⃗_{2,c}, ··· , x⃗_{NP,c}]
  while (not termination condition) {
    X_n ← X_c
    for (i ← 1 to NP) {
      v⃗_i ← Mutation(i, X_c, F)
      u⃗_i ← Crossover(x⃗_{i,c}, v⃗_i, Cr)
      Selection(i, X_c, u⃗_i, X_n, f)
    }
    X_c ← X_n
  }
  Output the best vector in X_c
```

図 3.1　DE の処理手順

3.1.1　初期化

最初に，N 次元のベクトル (個体) の集合である初期集団を生成する．指定された各次元の最小値と最大値の範囲内で一様乱数によって個体を生成する．集団 X_c 内にある i 番目の個体 $\vec{x}_{i,c}$ が持つ j 番目の要素は以下の式に従って初期化される．

$$x_{i,j,c} = x_{j,\min} + rand_{i,j}\,[0,1] \cdot (x_{j,\max} - x_{j,\min}) \tag{3.1}$$

ここで，$x_{j,\min}$ は j 番目の要素の最小値であり，$x_{j,\max}$ は最大値である．$rand_{i,j}[0,1]$ は 0 から 1 の値を取る一様乱数であり，個体の要素ごとに異なる値を取る．

3.1.2　変異 (Mutation)

変異操作では現在の集団からランダムに選ばれた個体を用いて，donor 個体 $\vec{v}_i$ を生成する．図 3.2 に変異操作の概念を示す．一般的な DE では，DE/rand/1 と DE/rand/2 の 2 種類の演算が用いられる．DE/rand/1 は以下の式に従って donor 個体を生成する．

$$\vec{v}_i = \vec{x}_{r_1^i,c} + F \cdot \left(\vec{x}_{r_2^i,c} - \vec{x}_{r_3^i,c}\right) \tag{3.2}$$

また，DE/rand/2 は以下の式に従って個体を生成する．

$$\vec{v}_i = \vec{x}_{r_1^i,c} + F \cdot \left(\vec{x}_{r_2^i,c} - \vec{x}_{r_3^i,c}\right) + F \cdot \left(\vec{x}_{r_4^i,c} - \vec{x}_{r_5^i,c}\right) \tag{3.3}$$

個体番号 r_1^i，r_2^i，r_3^i，r_4^i，r_5^i は相互に異なる整数であり，$\{1, 2, \ldots, NP\} \setminus \{i\}$ からランダムに選ばれる．F はスケーリング係数と呼ばれるアルゴリズムの制御パラメータである．

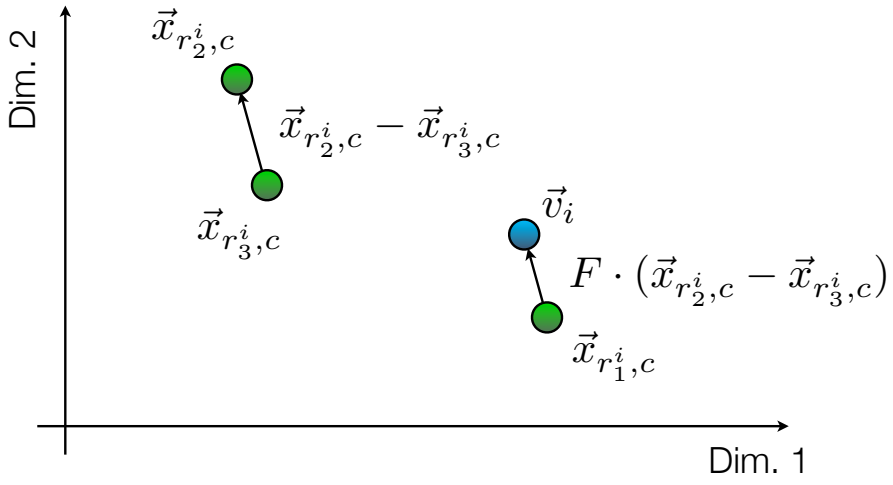


図 3.2　変異操作 (DE/rand/1) の概念

ランダムに選択した個体間の差分ベクトルを用いて新たな個体を生成することが DE の最も特徴的な点である．このようなベクトルを用いることで，探索の進行状況に合わせた適応的な個体生成が行える．探索の序盤では各個体は探索領域内に散らばって配置されるため，差分ベクトルは長くなりやすい．この場合，新しい個体は現在の個体から離れた位置に生成されるため，探索領域全体に渡る大域的な探索が行われる．逆に探索の終盤では各個体は最適解付近に集中しているため，差分ベクトルは短くなりやすい．この場合，新しい個体は現在の固体に近い位置に生成されるため，最適解付近での局所的な探索が行われる．

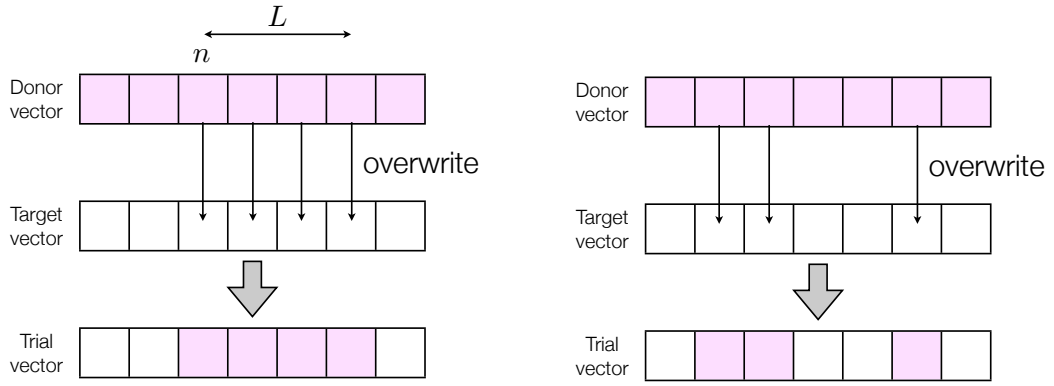
3.1.3　交叉 (Crossover)

交叉操作では target 個体 $\vec{x}_{i,c}$ と donor 個体 $\vec{v}_i$ を組み合わせることで，trial 個体 $\vec{u}_i$ を生成する．一般的な DE では *exponential* と *binomial* と呼ばれる 2 種類の処理のいずれかが用いられる．図 3.3 に各処理の概念を示す．

exponential 処理では，まず $[1, D]$ の範囲で整数 n をランダムに抽出する．D はベクトルの要素数である．そして，以下に示す処理により別の整数 $L \in [1, D]$ を抽出する．

$L \leftarrow 0$
`do {`
　$L \leftarrow L + 1$
`} while (`$rand[0,1] \leq Cr$ `and` $L < D$`)`
`Output L`

Cr はアルゴリズムの制御パラメータである (交叉率と呼ぶ)．この値を用い，trial 個体



(a) Exponential　　(b) Binomial

図 3.3　交叉操作の概念

$\vec{u}_i$ を以下の式に従って決定する．

$$u_{j,i} = \begin{cases} v_{j,i} & j = \langle n \rangle_D, \langle n+1 \rangle_D, \ldots, \langle n+L-1 \rangle_D \\ x_{j,i,c} & \text{otherwise} \end{cases} \tag{3.4}$$

ここで $\langle\rangle_D$ は D を法とした剰余関数である．

binomial 処理では，trial 個体の各要素は確率 Cr で donor 個体の値を引き継ぎ，確率 $1 - Cr$ で target 個体の値を引き継ぐ．Cr はアルゴリズムの制御パラメータである．trial 個体 $\vec{u}_i$ は以下の式で得られる．

$$u_{j,i} = \begin{cases} v_{j,i} & rand_{i,j}[0,1] \leq Cr \text{ or } j = j_{rand} \\ x_{j,i,c} & \text{otherwise} \end{cases} \tag{3.5}$$

ここで，$j_{rand} \in [1, 2, \ldots, D]$ は一様乱数によって選ばれた要素番号であり，$\vec{u}_i$ は少なくとも一つの要素を trial 個体 $\vec{v}_i$ から引き継ぐようになっている．

3.1.4　選択 (Selection)

選択処理では target 個体と trial 個体のどちらが次世代の集団に残るかを決定する．trial 個体の目的関数値が target 個体の値以下であれば，対応する target 個体を置き換える．次世代の個体 $\vec{x}_{i,n}$ は以下の式に従って決定する．

$$\vec{x}_{i,n} = \begin{cases} \vec{u}_i & f(\vec{u}_i) \leq f(\vec{x}_{i,c}) \\ \vec{x}_{i,c} & f(\vec{u}_i) > f(\vec{x}_{i,c}) \end{cases} \tag{3.6}$$

3.2　多峰性関数最適化向け差分進化の既存手法

多峰性関数の最適化問題を解くための DE の拡張手法はこれまでにもいくつか提案されている．Thomsen は CrowdingDE と SharingDE の 2 つの拡張手法を提案し，一般的に用いられる 14 のベンチマーク問題において CrowdingDE が良好な性能を示すことを報告している [45]．DE with local selection (DELS) [46] では現在の個体の近くに新しい個体を生成する．このような局所的な選択により，集団を複数の孤立に進化するニッチに分けることができる．Epitropakis らは現在の個体に隣接する個体の近くに新たな個体を生成する DE/nrand/1 と DE/nrand/2 を提案している [43]．さらに，制限付きトーナメント選択を導入した DE[47] や，種分化を導入した DE[48, 49] が提案されている．

以下では比較的単純な拡張により多峰性関数最適化を実現している DELS，DE/nrand/{1,2}，CrowdingDE について説明する．DELS と DE/nrand/{1,2} では変異操作を元の DE から変更しており，CrowdingDE では選択操作を変更している．

3.2.1　DE with local selection

DELS [46] では変異操作に用いる base 個体として，target 個体を用いる．これにより集団中の各個体は常にその個体自身からの変異個体と比較されることになる．

この変異操作は以下のように記述できる．

$$\vec{v}_i = \vec{x}_{i,c} + F \cdot \left(\vec{x}_{r_2^i,c} - \vec{x}_{r_3^i,c} \right) \tag{3.7}$$

3.2.2　DE/nrand/{1,2}

Epitropakis らは，現在の個体に隣接する個体の近くに新たな個体を生成する DE/nrand/1 と DE/nrand/2 を提案している [43]．DE/nrand/1 の変異操作は以下のように記述される．

$$\vec{v}_i = \vec{x}_{nearest,c} + F \cdot \left(\vec{x}_{r_1^i,c} - \vec{x}_{r_2^i,c} \right) \tag{3.8}$$

また，DE/nrand/2 の変異操作は以下のように記述される．

$$\vec{v}_i = \vec{x}_{nearest,c} + F \cdot \left(\vec{x}_{r_1^i,c} - \vec{x}_{r_2^i,c} \right) + F \cdot \left(\vec{x}_{r_3^i,c} - \vec{x}_{r_4^i,c} \right) \tag{3.9}$$

ただし，

$$nearest = \arg\min_{1 \le j \le NP, j \neq i} \|\vec{x}_{i,c} - \vec{x}_{i,c}\| \tag{3.10}$$

である．

3.2.3　CrowdingDE

CrowdingDE[45] では選択操作を変更している．この手法では，trial 個体 $\vec{u}_i$ とのユークリッド距離が最も短い個体を集団から抽出し，$\vec{u}_i$ の目的関数値が抽出した個体よりも小さければ置き換えを行う．この処理は以下のように記述できる．

$$\vec{x}_{z,n} = \begin{cases} \vec{u}_i & f(\vec{u}_i) \le f(\vec{x}_{z,c}) \\ \vec{x}_{z,c} & f(\vec{u}_i) > f(\vec{x}_{z,c}) \end{cases} \tag{3.11}$$

ただし，

$$z = \arg \min_{1 \le j \le NP} \|\vec{x}_{j,c} - \vec{u}_i\| \tag{3.12}$$

である．なお，1 度も抽出されなかった個体はそのまま次世代集団に残る．

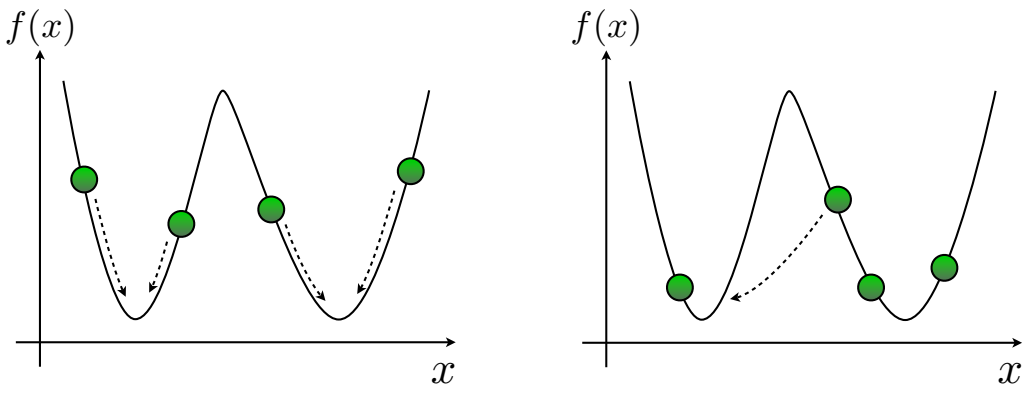
3.3　**提案手法**: DE/isolated/1

本章では孤立個体に基づく変異操作を導入した DE を提案する．この手法では現在の集団の中で孤立している個体に基づいて新たな個体を生成する．このように設定することで，新しい個体は孤立個体の近くに生成される．その後，新たに生成された個体が現在の個体よりも良好な評価値を持っていれば，その個体に置き換わる．従って，候補解は各世代において集団内で孤立している領域に向かって積極的に移住を行うようになる．これにより，解空間内の各最適解に均等に個体を割り当てられると考えられる．

本節では，多峰性関数の最適化に向けた DE の拡張である DE/isolated/1 を提案する．この手法の特徴は，変異操作中の base 個体として，集団内で孤立している個体を選ぶという点である．これにより，trial 個体は孤立個体の近くに生成される．従って，target 個体は選択操作により孤立している領域に向かって移住することになる．

図 3.4 に，一次元の問題における一般的なニッチング手法と提案手法の動作の比較を示す．一般的なニッチング手法では，同じ谷に配置された各個体はその谷を下り集まる傾向がある．これに対して提案手法では，各個体は積極的に他の谷に移住する．

図 3.5 に提案手法の擬似コードを示す．次の節では，この手法の設計指針について説明する．



(a) 既存手法　　(b) 提案手法

図 3.4　既存手法と提案手法の動作の比較

input:
  $f(\vec{x})$ - The function to be minimized
  F - Scaling factor
  Cr - Crossover rate
  NP - Population size
  N_d - The number of candidate
      differential vectors
  N_w - Threshold of rejected trials
procedure:
  $X \leftarrow$ Randomly initialized vectors
      $[\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_{NP}]$
  while (not termination condition) {
    for ($i \leftarrow 1$ to NP) {

$$\vec{v}_i \leftarrow \begin{cases} \vec{x}_{r_1^i} + F \cdot (\vec{x}_{r_2^i} - \vec{x}_{r_3^i}) & isolated = i \wedge n_w \geq N_w \\ \vec{x}_{isolated} + F \cdot (\vec{x}_{r_1^i} - \vec{x}'_{r_1^i}) & \text{otherwise} \end{cases}$$

$$\vec{u}_i \leftarrow \text{Crossover}(\vec{x}_i,\ \vec{v}_i,\ Cr)$$

$$\vec{x}_i \leftarrow \begin{cases} \vec{u}_i & f(\vec{u}_i) \leq f(\vec{x}_i) \\ \vec{x}_i & f(\vec{u}_i) > f(\vec{x}_i). \end{cases}$$

    }
  }
  Output X

図 3.5　DE/isolated/1 の擬似コード

3.3.1　孤立個体に基づく変異操作

DE/isolated/1 で用いられる変異操作では，集団内で孤立している個体を base 個体として選び出す．この処理は以下のように表される．

$$\vec{v}_i = \vec{x}_{isolated} + F \cdot \left(\vec{x}_{r_1^i} - \vec{x}'_{r_1^i} \right) \tag{3.13}$$

ここで，$isolated$ は孤立個体の個体番号であり，以下の式に従って決定する．

$$isolated = \arg\max_{1 \leq j \leq NP} \left\{ \min_{1 \leq k \leq NP, j \neq k} ||\vec{x}_j - \vec{x}_k|| \right\} \tag{3.14}$$

これは，集団内の隣接する個体への距離が最も大きいものが孤立個体として選ばれることを意味している．

差分ベクトルの第 2 項 $\vec{x}'_{r_1^i}$ は $\vec{x}_{r_1^i}$ に隣接する N_d 個の個体の中でランダムに選び出した個体である．$N_d \in [1, 2, \ldots, NP-1]$ はアルゴリズムの制御パラメータである．$N_d = NP - 1$ とした場合，target 個体も選択される可能性があるという違いはあるが，一般的な DE とほぼ同等な設定となる．第 2 項の設計意図は，trial 個体を確実に孤立個体の近くに生成するということである．一般的な DE では，差分ベクトルは集団内からランダムに抽出した個体から生成される．このような設定では，抽出された各個体が探索空間内の異なる谷に配置されている場合，探索空間内の山を飛び越えるようなベクトルが生成されてしまう．提案手法においてこのような設定を用いると，trial 個体が孤立個体の近くに生成されにくくなる．

孤立個体の決定方法については，式 (3.14) のように隣接する個体との距離を用いる他に，以下に示すような集団の重心からの距離を用いて決定する方法や，他個体との平均距離に基づいて決定する方法も考えられる．

- 集団の重心からの距離

$$isolated = \arg\max_{1 \le j \le NP} \left\{ \left| \frac{1}{NP} \sum_{k=1}^{NP} \vec{x}_k - \vec{x}_j \right| \right\} \tag{3.15}$$

- 他個体との平均距離

$$isolated = \arg\max_{1 \le j \le NP} \left\{ \frac{1}{NP} \sum_{k=1}^{NP} |\vec{x}_k - \vec{x}_j| \right\} \tag{3.16}$$

しかしながら，これらの方法は集団の重心に近い個体が孤立個体として選ばれにくい性質がある．このため，図 3.6 のような状況においても中心の個体は孤立個体として選択されず，他の領域に新たな個体が生成されてしまう．これを踏まえ，孤立個体の決定方法としては式 (3.14) を用いることとした．

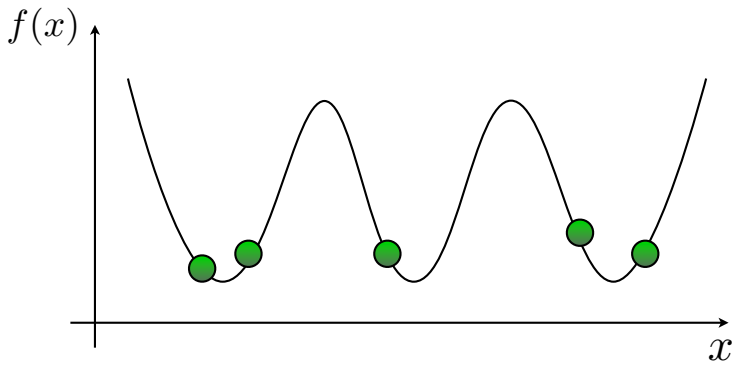


図 3.6　集団中心付近にある孤立個体の例

3.3.2　集団の即時更新

通常の DE では集団は各個体の次世代個体がすべて生成された後で更新される．言い換えれば，新しく生成される個体は別の記憶領域 X_n に保存され，その後，集団を格納する記憶領域 X_c を上書きするということである．提案手法においてこのような更新処理を行うと，次世代の集団は現世代での孤立個体の近くに一斉に集まってしまい，初期収束を引き起こす．

このような状況を引き起こさないように，集団を即座に更新するようにした．この更新処理では，選択処理によって target 個体が新しく生成された個体に置き換わる場合，即座に集団が更新される．言い換えれば，集団を格納するために 1 つ記憶領域 X のみを用い，すべての処理はこの記憶領域内の個体に対して操作を行うということである．従って，孤立個体に関する情報も即座に更新されることになる．

3.3.3　局所最適解からの脱出

孤立個体に基づいた変異操作について，探索の停滞を引き起こす状況が考えられる．図 3.7 のように孤立個体が局所最適な領域に配置されており，集団内の他個体がその領域よりも評価値の良い領域に配置されているとする．この場合，trial 個体は孤立個体の近くに生成され悪い評価値を持つため，すべての trial 個体は選択操作により棄却されてしまう．このような状況を避けるためには，孤立個体は局所最適な領域から別の領域に移動しなければならない．しかしながら，すべての trial 個体はその個体の近くに生成されるため，その領域から脱出することは難しい．

そこで脱出するための機構として，trial 個体が特定の回数棄却された場合には，孤立個体に対して通常の DE で用いる変異操作を適用することで局所解から脱出するようにし

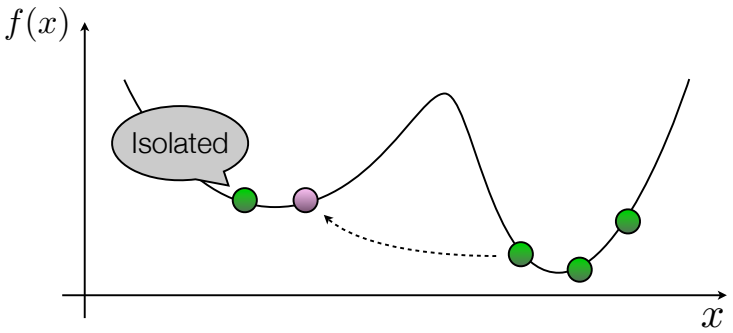


図 3.7　局所最適解での停滞

た．改良した変異操作は以下のように表される．

$$\vec{v}_i = \begin{cases} \vec{x}_{r_1^i} + F \cdot \left(\vec{x}_{r_2^i} - \vec{x}_{r_3^i} \right) & isolated = i \wedge n_w \geq N_w \\ \vec{x}_{isolated} + F \cdot \left(\vec{x}_{r_1^i} - \vec{x}'_{r_1^i} \right) & \text{otherwise} \end{cases} \tag{3.17}$$

ここで，$n_w (\geq 0)$ は現在の孤立個体を使用している間に棄却された trial 個体の数であり，集団内の個体が新たに生成された個体に置き換わった場合に 0 に初期化される．N_w は通常の変異操作を行うかどうかを決める閾値である．$N_w = \infty$ と設定することは式 (3.13) で表される元の変異操作を用いることに対応する．

3.4　性能評価実験

3.4.1　各手法の動作比較

まず提案手法 DE/isolated/1 が実際に積極的な移住を行なっているか確かめるために，以下のような 1 次元多峰性関数 (Deb 1) を用いて計算機実験を行った．

$$f(x) = -\sin^6(5\pi x) \tag{3.18}$$

図 3.8 に関数の形状を示す．最適解集合は $x^* = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ である．

提案手法に加え，前述の拡張手法 CrowdingDE[45]，DE with local selection (DELS) [46]，DE/nrand/1[43] についても実験を行い，各手法の動作を比較した．DE の制御パラメータは次のように設定した．集団サイズ NP を 100，世代数を 1000 とした．スケーリング係数と交叉率はそれぞれ $F = 0.5$，$Cr = 0.9$ とし，交叉操作には binomial 処理を用いた．DE/isolated/1 においては差分ベクトルの候補の数 $N_d = NP - 1$，棄却される trial 個体の閾値は $N_w = \infty$ とした．

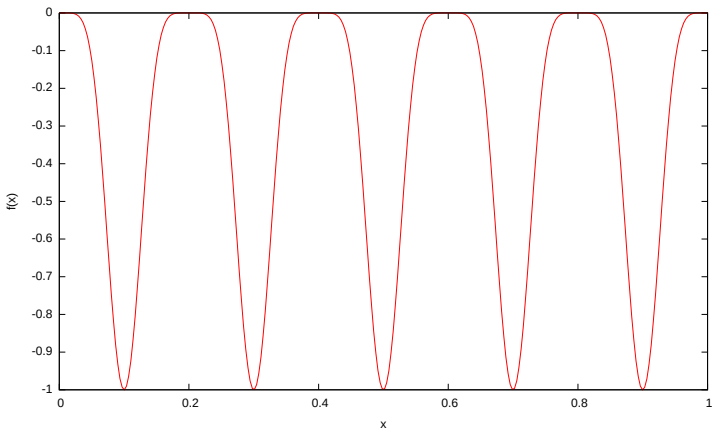
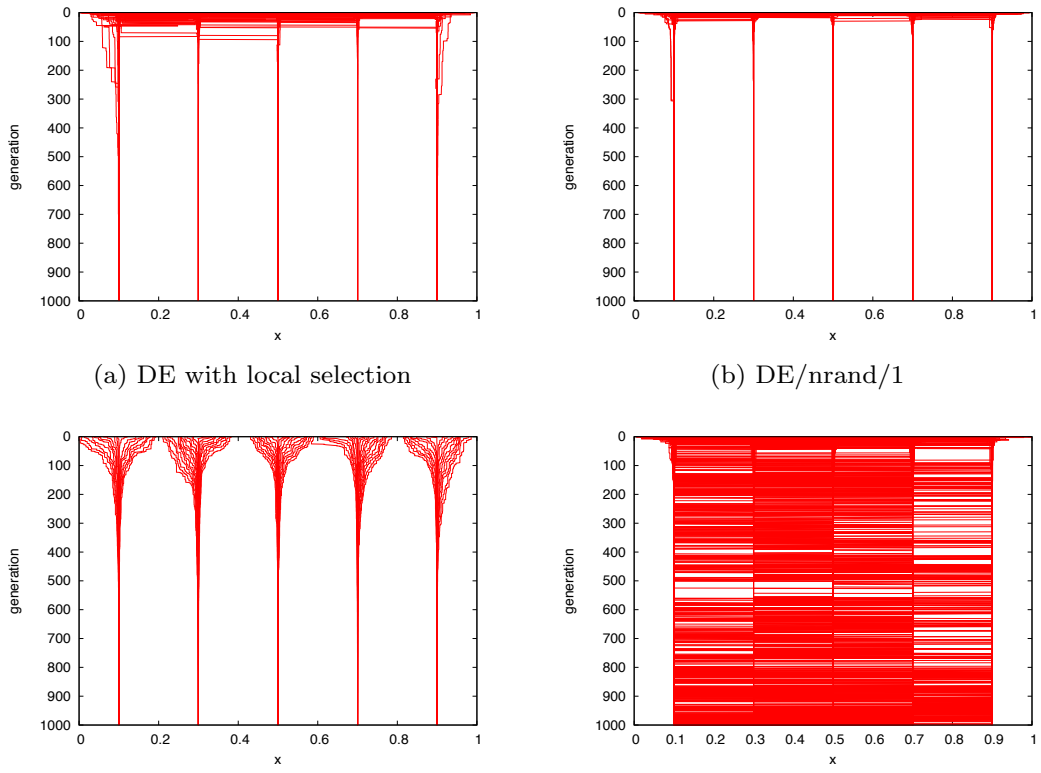


図 3.8　1 次元 Deb 1 関数



(a) DE with local selection　(b) DE/nrand/1

(c) CrowdingDE　(d) DE/isolated/1

図 3.9　各手法の進化軌跡

図 3.9 に各手法の進化軌跡の一例を示す．この図では集団中の各個体の進化軌跡を重ねて表示している．既存の拡張手法では集団中の各個体がニッチを形成し独立して探索を進めるため，最終的には各最適解に留まっている．これに対して提案手法では世代全体を通して最適解間での移住が行われている．

3.4.2　探索性能の比較

次に提案手法の性能を評価するために，8 つの 2 次元多峰性関数をベンチマークとして用い性能評価実験を行った．また，既存の DE の拡張手法，CrowdingDE [45]，DE with local selection (DELS) [46]，そして DE/nrand/{1,2} [43] と性能を比較した．なお，実験の設定は文献 [43] を参考にした．

3.4.2.1　ベンチマーク関数

文献 [43] で用いられているものと同じ 8 つの多峰性関数をベンチマークとして用いる．表 3.1 にこれらの関数と解の定義域を示す．また，図 3.10 に関数の形状を示す．

関数 F_1 と F_2 には少ない数の不規則に配置された大域的最適解がある．最適解の数はそれぞれ 3 と 4 である．関数 F_3 は 18 個の大域的最適解と 742 個の局所最適解を持つ．

表 3.1　ベンチマーク関数

Function	Mathematical formula	Domain
Branin	$F_1(\vec{x}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2$ $+10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	$x_1 \in [-5, 10]$ $x_2 \in [0, 15]$
Himmelblau	$F_2(\vec{x}) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$	$\vec{x} \in [-6, 6]^2$
Shubert	$F_3(\vec{x}) = \sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)$ $\cdot \sum_{i=1}^{5} i\cos\left((i+1)x_2 + i\right)$	$\vec{x} \in [-10, 10]^2$
Six-hump camel back	$F_4(\vec{x}) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2$ $+x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2$	$x_1 \in [-1.9, 1.9]$ $x_2 \in [-1.1, 1.1]$
Vincent	$F_5(\vec{x}) = -\frac{1}{2}\sum_{i=1}^{2}\sin\left(10\log(x_i)\right)$	$\vec{x} \in [0.25, 10]^2$
Deb 1	$F_6(\vec{x}) = -\frac{1}{2}\sum_{i=1}^{2}\sin^6(5\pi x_i)$	$\vec{x} \in [0, 1]^2$
Deb 3	$F_7(\vec{x}) = -\frac{1}{2}\sum_{i=1}^{2}\sin^6\left(5\pi\left(y_i^{\frac{3}{4}} - 0.05\right)\right)$	$\vec{x} \in [0, 1]^2$
Modified Rastrigin	$F_8(\vec{x}) = 20 + \sum_{i=1}^{2}\left(x_i^2 + 10\cos(2\pi x_i)\right)$	$\vec{x} \in [-5.12, 5.12]^2$

関数 F_5 には 6^2 個の大域的最適解があり，x の値が大きくなるにつれて隣り合う解同士の距離も大きくなる．関数 F_6 には 5^2 個の大域的最適解が均等に配置されており，局所解はない．関数 F_7 は同じ数の大域的最適解を持つが，各最適解間の距離は原点に近づくにつれて小さくなる．関数 F_8 は関数最適化問題のベンチマークとしてよく知られる Rastrigin 関数を改造したものであり，均等に配置された 4 つの大域的最適解を持ち，96 個の局所的最適解を持つ．

3.4.2.2　評価指標

各手法の性能を評価するために，発見率 (peak ratio, PR) と成功率 (success ratio, SR) の 2 つの評価指標を用いる．発見率は以下のように定義する．

$$\text{発見率} = \frac{\text{発見した大域的最適解の数}}{\text{関数が持つ大域的最適解の数}} \tag{3.19}$$

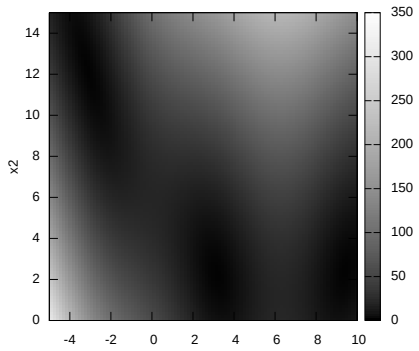
なお，ある最適解とのユークリッド距離が指定した距離 ε (精度レベルと呼ぶ) 以下である個体が集団内にあれば，その解は発見したとみなす．

成功率は以下のように定義する．

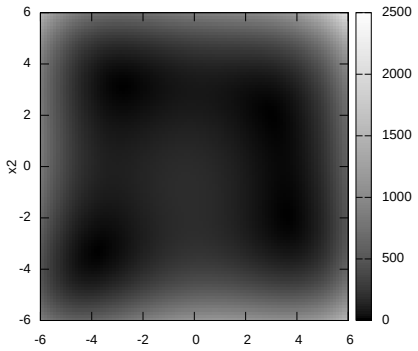
$$\text{成功率} = \frac{\text{全最適解を発見した試行の数}}{\text{全試行数}} \tag{3.20}$$

ある試行において全ての最適解を発見することは，その試行の最終世代における発見率が 1 となることに対応する．

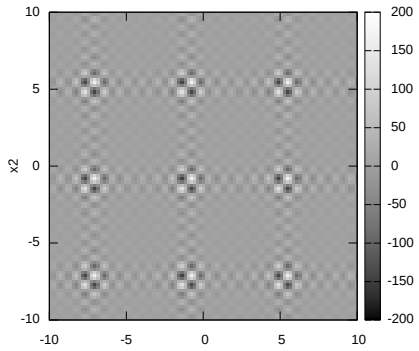
これらの定義に従って，最終世代における発見率の平均値と成功率によって各手法の性能を評価する．さらに文献 [43] と同じように，あらかじめ精度レベルを指定した上で，全



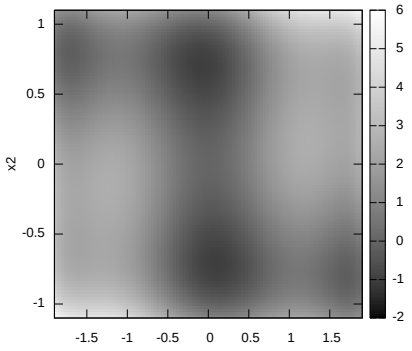
(a) Branin (F_1)



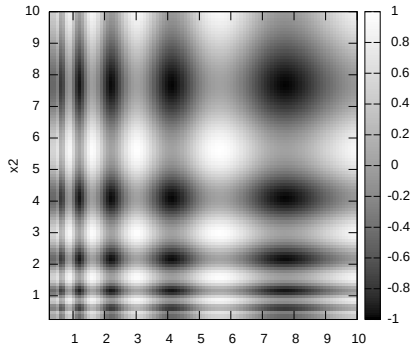
(b) Himmelblau (F_2)



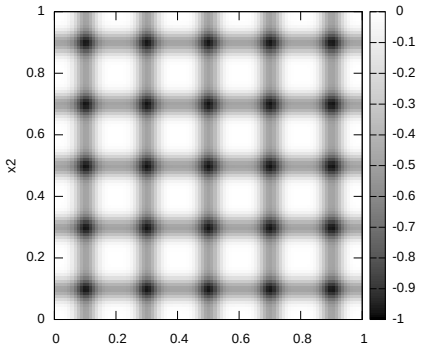
(c) Shubert (F_3)



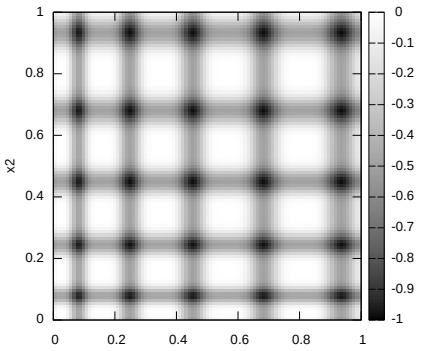
(d) Six-hump camel back (F_4)



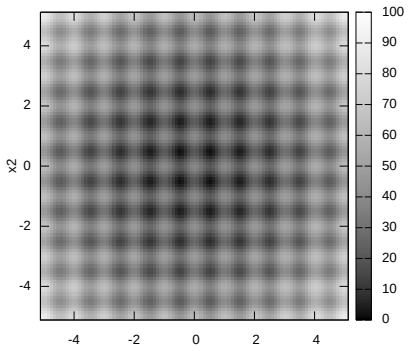
(e) Vincent (F_5)



(f) Deb 1 (F_6)



(g) Deb 3 (F_7)



(h) modified Rastrigin (F_8)

図 3.10　各関数の形状

ての大域的最適解を発見するまでに要する世代数についても評価した．また，各手法の実行中の挙動を調べるために各世代における発見率の推移について調べた．

3.4.2.3　パラメータの設定

各関数と各精度レベルにおいて 100 回の試行を行った．精度レベルは $\varepsilon = \{10^{-3}, 10^{-4}, \ldots, 10^{-8}\}$ とした．DE の制御パラメータは次のように設定した．集団サイズ NP を 100，世代数を 1000 とした．CrowdingDE, DELS, DE/nrand/1，そして DE/nrand/2 については一般的な設定である $F = 0.5$ と $Cr = 0.9$ の値を用いた．これらは文献 [43] と同様の設定である．すべての手法において交叉操作には binomial 処理を用いた．

提案手法である DE/isolated/1 においては，$F = 0.9$，$Cr = 0.9$ とした．F を他の手法に比べ大きい値に設定するのは，3.3.1 節で示した隣接個体に基づいた差分ベクトルの抽出方法では，そのベクトルの大きさが小さくなりやすいためである．差分ベクトルの候補の数 N_d は 5 とした．棄却される trial 個体の閾値は $N_w = 150$ とした．これらの値は事前実験の結果により決定したが，最適な値とは限らない．

3.4.2.4　実験結果

表 3.2 および 3.3 に各ベンチマーク関数における実験結果を示す．なお，表中の値で太字になっているものは，各精度レベルにおける最良の結果を表している．提案手法は精度レベル ε を厳しく設定した場合に，他の手法に比べ良好な性能を示している．レベルが $\varepsilon \leq 10^{-6}$ の時には，提案手法は 7 個の関数 $(F_1 - F_7)$ において最も良い性能を示している．

提案手法は特に Vincent 関数 (F_5) と Deb 3 関数 (F_7) において良好な性能を示している．これらの関数には探索空間内に大きい谷と小さい谷が混在している．初期集団の生成の際に個体のほとんどは大きい谷に配置され探索資源の配分に偏りが生じるため，既存の手法では小さい谷の中にある最適解を発見することが難しい．

modified Rastrigin 関数 (F_8) においては，提案手法の性能は他の手法に比べ多少低くなっている．これはいくつかの試行において全ての最適解を発見することに失敗しているためである．この原因は，集団内で孤立している個体が複数いる場合においても，隣接する個体との距離が最も大きいものだけを孤立個体として抽出しているためであると考えられる．この場合，孤立個体として選ばれなかった個体は選択処理によって失われる可能性がある．その他の理由としては，3.3.3 節で説明した局所解からの脱出機構が必要な個体まで移動させていることが考えられる．脱出機構を行うかどうかを決める閾値 N_w を小さく設定して実験を行ったところ，探索性能が低下することを確認している．しかしなが

ら，3.3.3 節で説明したとおり閾値を高く設定しすぎた場合においても，探索の停滞が生じるため性能は低下する．適切な設定は与えられた問題に依存しているため，N_w をアルゴリズムの実行中に調節する機構を導入することが，提案手法の改良として考えられる．

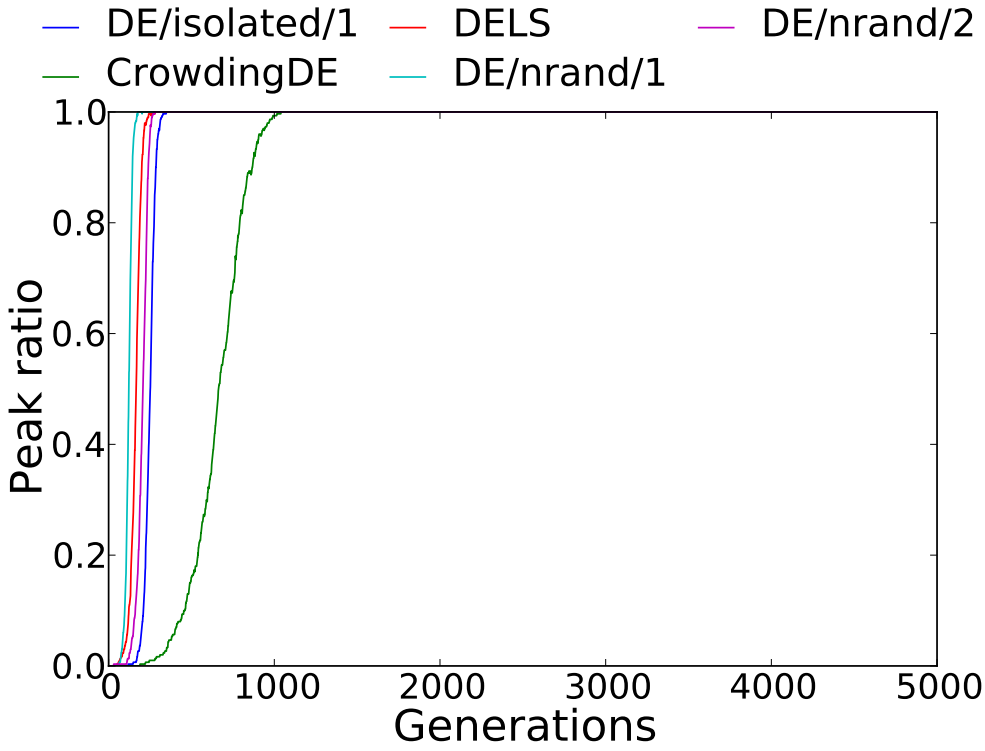
表 3.4 に，関数 F_1，F_2，F_4，F_8 における各手法の収束速度を示す．精度レベルは $\varepsilon = 10^{-3}$ とした．なお，これらの値はすべての最適解を発見できた試行の結果のみをもとに算出しており，いくつかの手法においては 100% の成功率に到達できていないことに注意が必要である (F_1 関数における CrowdingDE，及び F_8 における DE/isolated/1)．特に局所解を含む関数 F_4 と F_8 において，提案手法の収束速度は既存の手法よりも遅いことがわかる．これは閾値 N_w を小さく設定することによって改善できる可能性があるが，そのような設定では前述のとおり性能低下が起こりうる．

3.4.3　各手法の発見率の推移

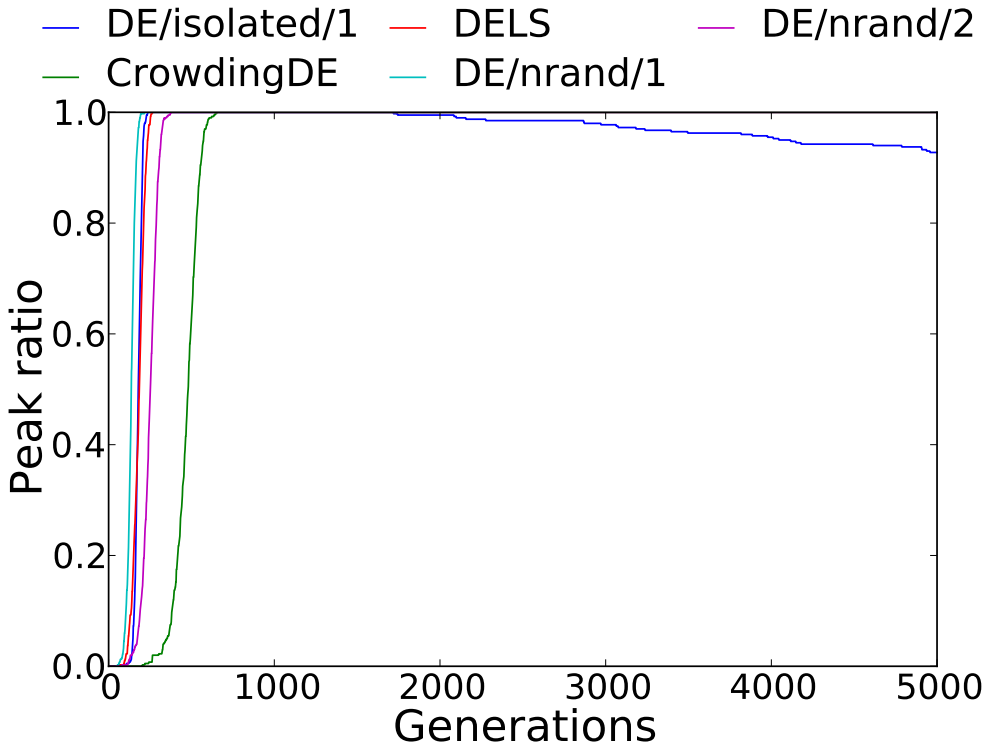
各手法の実行中の挙動を調べるために世代数を 5000 に設定した上で発見率の推移を調べた．手法の制御パラメータは前述の実験と同じ値を使用した．

図 3.11，3.12，3.13，3.14 に精度レベル $\varepsilon = 10^{-3}$ とした時の発見率の推移を示す．これは 100 回の試行の平均値である．なお，表中の値で太字になっているものは，各関数における最速の結果を表している．特に局所最適解を持つ関数 F_3，F_4，F_8 において，提案手法の収束速度が他の手法に比べ遅いことがわかる．なお表 3.4 の結果からもわかるように，CrowdingDE はどの関数においても比較的収束速度が遅い傾向がある．表 3.2，3.3 の結果において精度レベルを小さく設定した場合にこの手法の性能が低下しているのは，指定したレベルの距離まで収束できていないためである．

アルゴリズムを長期にわたって実行した場合，集団が最適解に収束した後はその配置を維持し続けるような挙動が望ましいといえる．提案手法である DE/isolatd/1 の推移に注目すると，Himmelblau 関数 (F_2) と Shubert 関数 (F_3) において 1000 世代程度までは高い発見率を保っているが，その後低下してしまう．この点においては CrowdingDE は収束した後の発見率の値は一定となっており，望ましい挙動を行なっている．その他の手法は特に Shubert 関数において顕著なように，発見率が 1 となる最適配置に収束した後に低下する場合があり，その配置を維持できていない．図 3.12 (a) の Shubert 関数の結果に注目すると，DE/nrand/1 は 400 世代程度で 1 に近い発見率を示している．しかしながら，その状態を維持できず，世代を重ねるごと低下している．そのため 1000 世代目の結果を切り出した表 3.2 の結果においては良好な値を示していない．以上のことから，発見した最適解を別の記憶領域に保持しておくなどの機構が必要といえる．

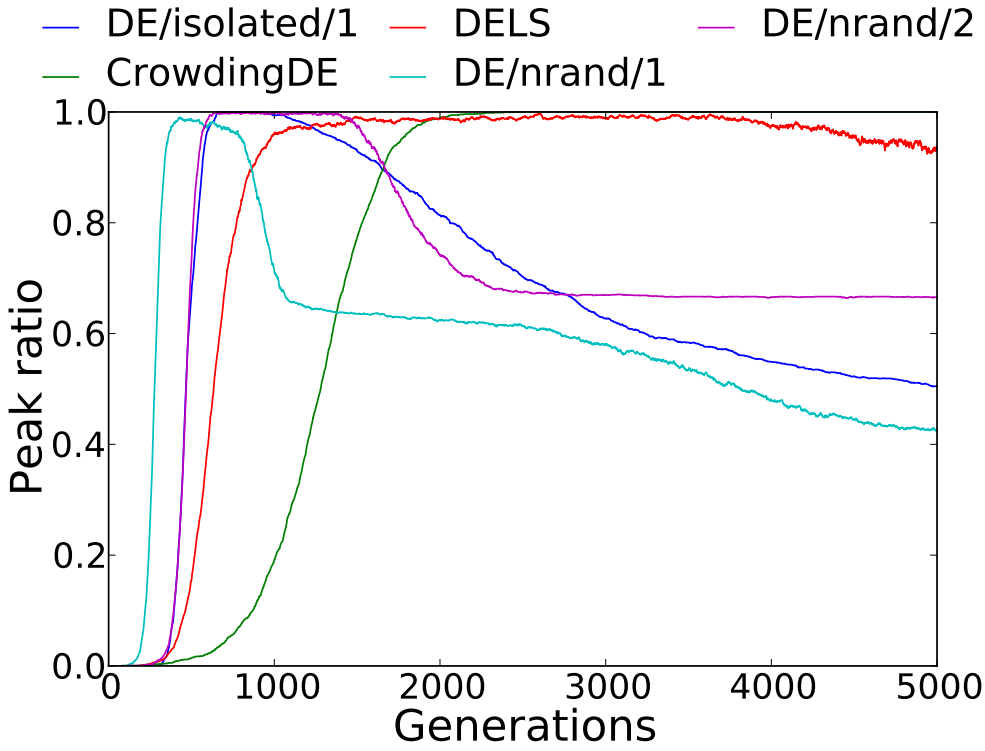


(a) Branin (F_1)

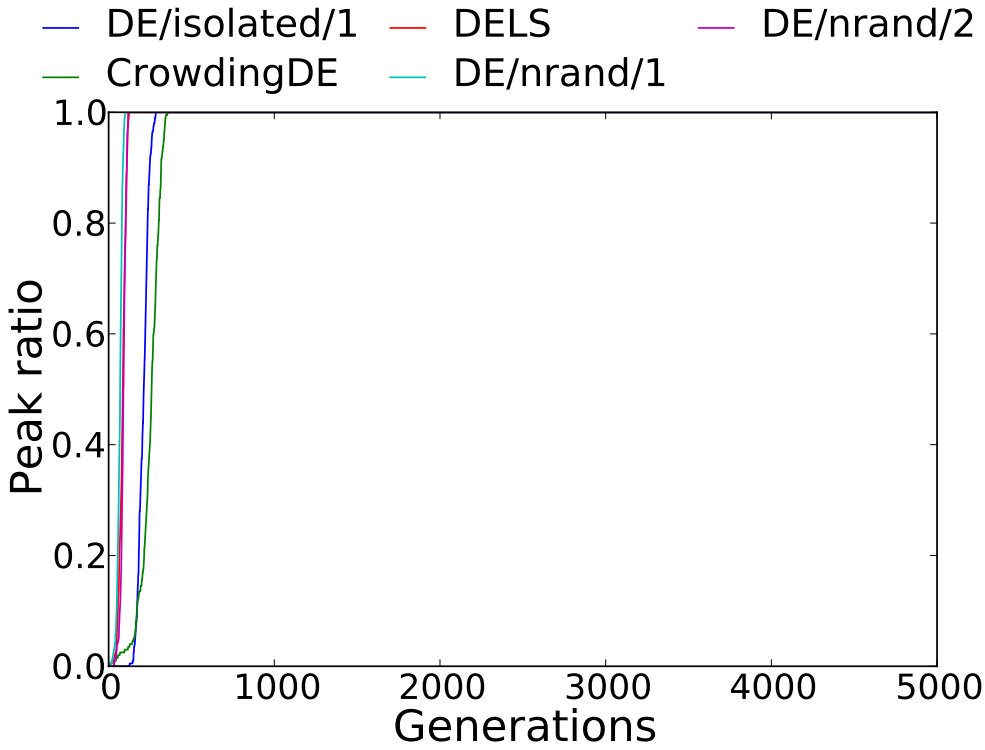


(b) Himmelblau (F_2)

図 3.11　F_1，F_2 関数における発見率の推移

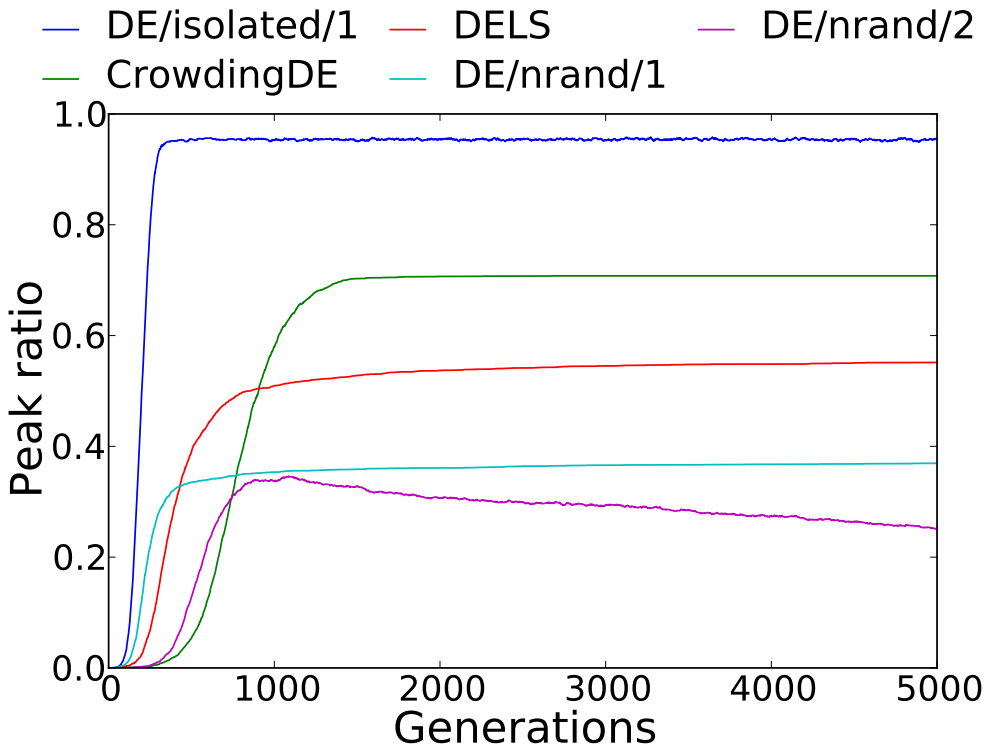


(a) Shubert (F_3)

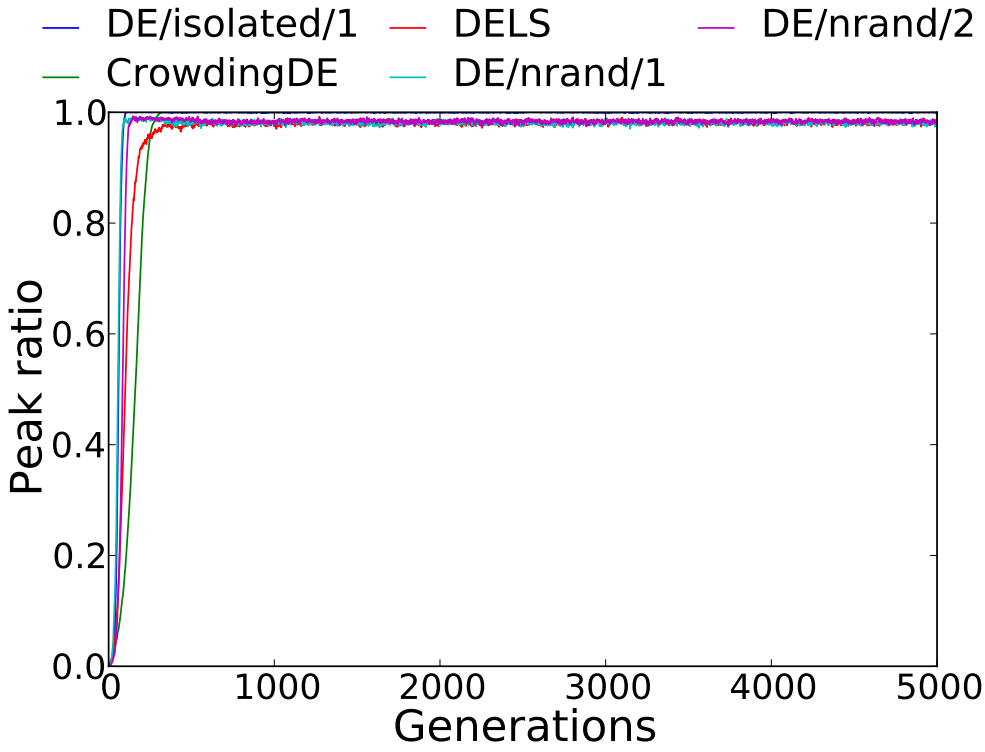


(b) Six-hump camel back (F_4)

図 3.12　F_3，F_4 関数における発見率の推移

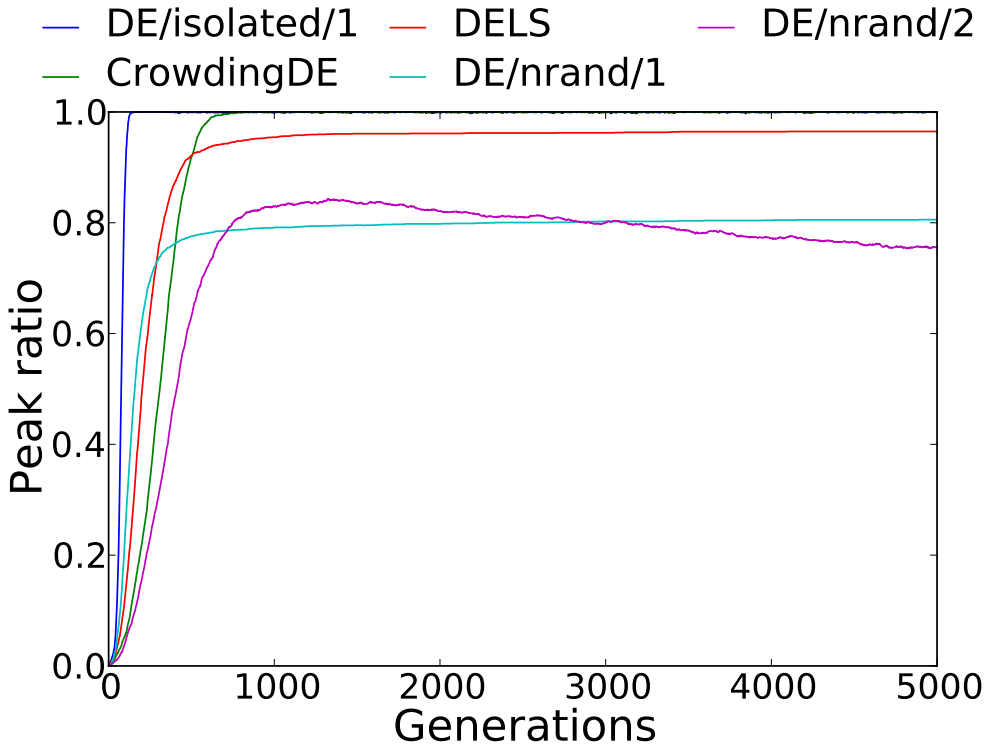


(a) Vincent (F_5)

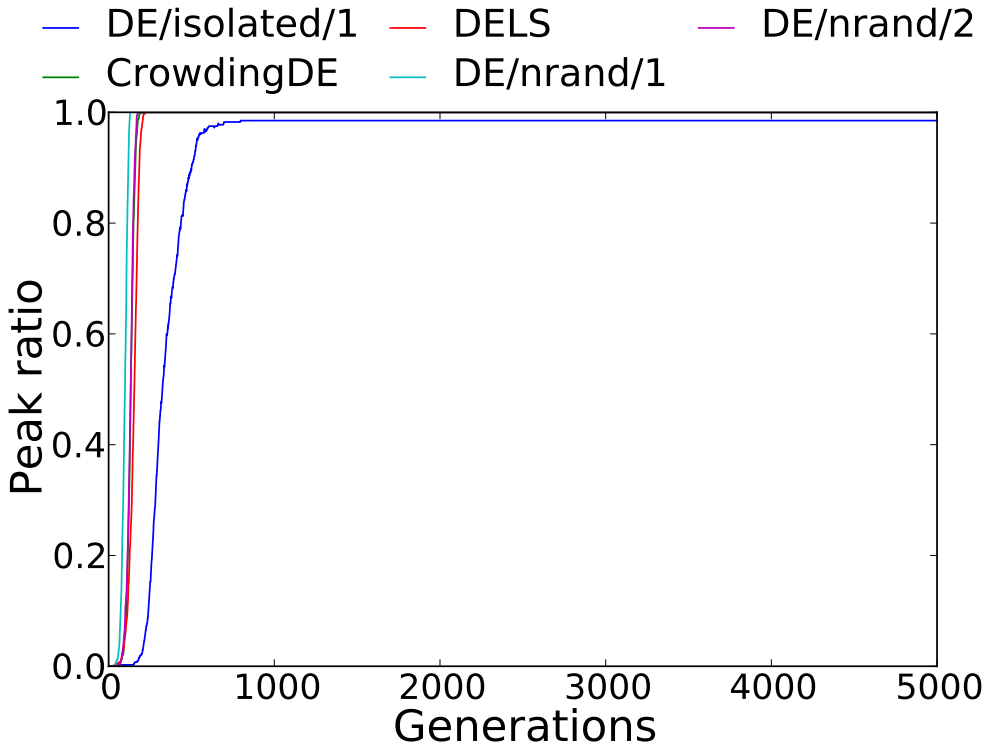


(b) Deb 1 (F_6)

図 3.13　F_5，F_6 関数における発見率の推移



(a) Deb 3 (F_7)



(b) modified Rastrigin (F_8)

図 3.14　F_7，F_8 関数における発見率の推移

3.5　まとめ

本論文では多峰性関数の最適化を行う DE の拡張手法として DE/isolated/1 を提案し，8 つの多峰性関数をベンチマークとして用い性能評価実験を行った．実験結果より，提案手法は既存の DE の拡張手法に比べ概ね良好な性能を示すことが分かった．

今後の課題としては，より高次元な問題や実問題を用いた実験を行うこと，制限付きトーナメント選択を導入した DE[47] や種分化を導入した DE[48, 49] など他の手法との性能比較を行うことがあげられる．これらの実験を通して，提案した機構の特性や，アルゴリズムの長所および短所について詳しく調べる必要がある．

また，提案手法を更に改良することも課題である．一つの改良案としては前述のとおり，幅広い種類の問題を効果的に解けるように閾値 N_w を自動的に調節する機構を組み込むことがある．3.4.3 節で示したように，発見した最適解を保持する機構も必要である．さらに，計算時間を削減することも重要である．式 (3.14) において集団中の孤立個体を抽出することは，時間計算量の大きい最近傍探索を行なっていることを意味する．そこで，kd-tree [50] のような効率的な探索アルゴリズムを導入することで高速化が可能である．また，locality sensitive hashing [51] のような近似アルゴリズムを用いることでさらに高速化できると考えられるが，この場合には近似誤差が探索に与える影響について注意しなければならない．

表 3.2　関数 F_1 – F_4 における性能比較

(a) Branin (F_1)

	DE/isolated/1		CrowdingDE		DELS		DE/nrand/1		DE/nrand/2	
ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
10^{-3}	**1.000**	**1.00**	0.993	0.98	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-4}	**1.000**	**1.00**	0.157	0.01	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-5}	**1.000**	**1.00**	0.003	0.00	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-6}	**1.000**	**1.00**	0.000	0.00	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-7}	**1.000**	**1.00**	0.000	0.00	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-8}	**1.000**	**1.00**	0.000	0.00	0.820	0.55	**1.000**	**1.00**	**1.000**	**1.00**

(b) Himmelblau (F_2)

	DE/isolated/1		CrowdingDE		DELS		DE/nrand/1		DE/nrand/2	
ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
10^{-3}	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-4}	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-5}	**1.000**	**1.00**	0.395	0.07	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-6}	**1.000**	**1.00**	0.003	0.00	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-7}	**1.000**	**1.00**	0.000	0.00	0.990	0.96	**1.000**	**1.00**	0.995	0.98
10^{-8}	**1.000**	**1.00**	0.000	0.00	0.173	0.02	**1.000**	**1.00**	0.883	0.76

(c) Shubert (F_3)

	DE/isolated/1		CrowdingDE		DELS		DE/nrand/1		DE/nrand/2	
ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
10^{-3}	0.992	0.93	0.164	0.00	0.953	0.41	0.718	0.01	**0.998**	**0.97**
10^{-4}	0.994	0.92	0.004	0.00	0.489	0.00	0.727	0.00	**0.999**	**0.99**
10^{-5}	**0.997**	0.94	0.000	0.00	0.014	0.00	0.733	0.01	**0.997**	**0.96**
10^{-6}	**0.994**	**0.91**	0.000	0.00	0.000	0.00	0.714	0.01	0.921	0.54
10^{-7}	**0.996**	**0.94**	0.000	0.00	0.000	0.00	0.712	0.01	0.166	0.00
10^{-8}	**0.994**	**0.93**	0.000	0.00	0.000	0.00	0.708	0.00	0.002	0.00

(d) Six-hump camel back (F_4)

	DE/isolated/1		CrowdingDE		DELS		DE/nrand/1		DE/nrand/2	
ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
10^{-3}	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-4}	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-5}	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-6}	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-7}	**1.000**	**1.00**	0.215	0.07	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-8}	**1.000**	**1.00**	0.000	0.00	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**

表 3.3　関数 F_5 – F_8 における性能比較

(a) Vincent (F_5)

	DE/isolated/1		CrowdingDE		DELS		DE/nrand/1		DE/nrand/2	
ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
10^{-3}	**0.966**	**0.60**	0.587	0.00	0.508	0.00	0.359	0.00	0.339	0.00
10^{-4}	**0.959**	**0.62**	0.065	0.00	0.431	0.00	0.331	0.00	0.240	0.00
10^{-5}	**0.957**	**0.58**	0.001	0.00	0.239	0.00	0.304	0.00	0.062	0.00
10^{-6}	**0.966**	**0.61**	0.000	0.00	0.029	0.00	0.268	0.00	0.001	0.00
10^{-7}	**0.955**	**0.57**	0.000	0.00	0.001	0.00	0.211	0.00	0.000	0.00
10^{-8}	**0.942**	**0.37**	0.000	0.00	0.000	0.00	0.084	0.00	0.000	0.00

(b) Deb 1 (F_6)

	DE/isolated/1		CrowdingDE		DELS		DE/nrand/1		DE/nrand/2	
ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
10^{-3}	**1.000**	**1.00**	**1.000**	**1.00**	0.980	0.58	0.984	0.63	0.988	0.73
10^{-4}	**1.000**	**1.00**	**1.000**	**1.00**	0.975	0.52	0.986	0.69	0.981	0.61
10^{-5}	0.999	0.98	**1.000**	**1.00**	0.965	0.36	0.984	0.66	0.985	0.69
10^{-6}	**1.000**	**1.00**	0.551	0.02	0.949	0.18	0.983	0.64	0.984	0.65
10^{-7}	**1.000**	**0.99**	0.008	0.00	0.893	0.06	0.979	0.61	0.984	0.64
10^{-8}	**1.000**	**1.00**	0.000	0.00	0.435	0.00	0.972	0.44	0.983	0.67

(c) Deb 3 (F_7)

	DE/isolated/1		CrowdingDE		DELS		DE/nrand/1		DE/nrand/2	
ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
10^{-3}	0.999	0.98	**1.000**	**1.00**	0.957	0.29	0.804	0.00	0.840	0.01
10^{-4}	**1.000**	**0.99**	0.638	0.00	0.898	0.03	0.749	0.00	0.286	0.00
10^{-5}	**1.000**	**1.00**	0.018	0.00	0.384	0.00	0.654	0.00	0.004	0.00
10^{-6}	**0.999**	**0.97**	0.000	0.00	0.008	0.00	0.478	0.00	0.000	0.00
10^{-7}	**0.999**	**0.97**	0.000	0.00	0.000	0.00	0.110	0.00	0.000	0.00
10^{-8}	**0.999**	**0.98**	0.000	0.00	0.000	0.00	0.004	0.00	0.000	0.00

(d) modified Rastrigin (F_8)

	DE/isolated/1		CrowdingDE		DELS		DE/nrand/1		DE/nrand/2	
ε	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
10^{-3}	0.985	0.97	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-4}	0.988	0.98	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-5}	0.985	0.97	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-6}	0.995	0.99	0.963	0.85	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-7}	0.980	0.97	0.053	0.00	**1.000**	**1.00**	**1.000**	**1.00**	**1.000**	**1.00**
10^{-8}	0.970	0.94	0.000	0.00	0.940	0.82	**1.000**	**1.00**	**1.000**	**1.00**

表 3.4　収束速度の比較

	DE/isolated/1		CrowdingDE		DELS		DE/nrand/1		DE/nrand/2	
	Ave.	St.D.	Ave.	St.D.	Ave.	St.D.	Ave.	St.D.	Ave.	St.D.
F_1	276.46	24.03	790.17	84.03	189.32	15.94	**135.32**	12.88	227.64	20.57
F_2	196.31	17.08	537.02	39.17	215.75	20.91	**158.32**	18.22	289.51	26.34
F_4	219.08	24.67	284.37	32.79	95.58	13.83	**73.85**	10.73	94.66	12.09
F_8	402.00	75.80	149.67	14.45	178.27	15.71	**113.98**	10.50	150.92	12.12

第 4 章

結論

本論文では，進化する集団の構造に基づき，進化的計算手法の拡張を行った．集団の構造には，複数の進化集団が形成する集団間構造と，集団内の個体が形成する集団内構造の 2 つのレベルがあると考えられる．集団間構造を拡張する手法として共進化アルゴリズムに PMBGA を取り入れた確率モデル構築型共進化アルゴリズムを提案し，集団内構造を拡張する手法としては多峰性関数最適化に向けた差分進化の拡張手法 DE/isolated/1 を提案した．それぞれの手法について，計算機実験を通して挙動の分析と探索性能の評価を行った．

第 2 章では，集団間構造の拡張として確率モデル構築型共進化アルゴリズムを提案し，その一実装として PBIL アルゴリズムを CA に組み込んだ CA-PBIL を示した．そして，非推移的ナンバーズ・ゲームを用いてアルゴリズムの挙動を実験的に分析した．実験結果より，問題の非推移性に起因する様々な進化挙動が引き起こされ，適切な共進化が妨げられる場合があることが分かった．本実験では単純なゲームを用いてこの現象を確認したが，実問題のほとんどは同様な性質を有していると考えられる．さらに，この性質を利用することで望ましい共進化挙動を引き起こす一方法として，2 つの集団で異なる学習率を設定するという方法を発見した．ただし，この方法の性能は利得関数をどのように定義するかに依存しており，必ずしも効果的に働くわけではない．実応用においてはその点も考慮し，利得関数の設計についても慎重に行う必要がある．

第 3 章では，集団内構造に着目し，各個体が探索空間中の複数の最適解に対して均等に配置されるよう進化させる，多峰性関数の最適化について検討した．本章では多峰性関数の最適化を行う差分進化の拡張手法として DE/isolated/1 を提案し，8 つの多峰性関数をベンチマークとして用い性能評価実験を行った．実験結果より，提案手法は既存の拡張手法に比べ概ね良好な性能を示すことが分かった．特に大小の谷が混在する関数において，既存手法に比べ大幅な性能向上に成功した．これは積極的な移住を繰り返すことで各最適

解に割り当てられる個体数の均等化を図るメカニズムがうまく働いたためであると考えられる．しかしながら，提案手法は元のアルゴリズムと比べて処理が煩雑になってしまい，実装が比較的難しいものとなった．それに伴い計算量も通常の差分進化に比べ多くなっている．また，アルゴリズムの挙動を決定するための制御パラメータが増えている点も短所としてあげられる．実応用を考えるとアルゴリズムの実装と利用の容易さは重要な要素であるため，探索性能を保ちつつ，より単純化された改良手法を検討するべきである．

本研究では進化的計算における集団間構造と集団内構造に着目し，これらを拡張した手法を開発した．研究を通して，集団構造を適切に制御することが探索性能を引き出す上で非常に重要であることを示した．集団間構造については，利得関数の定義や，どのような進化集団を用意し相互作用させるかなど，集団間の関係を適切に設定しなければならない．また集団内構造については，所望の構造を得るための進化方法について，適用する問題の特徴を考慮した上で適切なものを選定する必要がある．

この研究の発展として，集団間と集団内の拡張を同時に施した手法を開発することが挙げられる．本研究では集団間構造の拡張と集団内構造の拡張を独立に行ったが，本来これら 2 つのレベルの構造は相互に依存しており，その組合せが探索性能に大きく影響する．例えば共進化アルゴリズムにおいては一方の集団内にある個体の多様性を維持することで，性能が向上することが報告されている [52]．このことから，第 3 章で提案したような集団内構造を持つ手法を組み込むことで性能を向上させられると期待できる．また，分散遺伝的アルゴリズムにおいても集団間の関係だけでなく，各集団内でどのような探索を行うかが全体の性能に大きく影響することから，様々な拡張手法が研究されている [53, 54, 55]．これらの例にとどまらず，適切な集団構造を効果的に扱える手法を開発することで，大規模複雑な実問題の解決に進化的計算手法を応用できるようになると考えられる．

謝辞

本論文は，名古屋大学大学院情報科学研究科複雑系科学専攻に在籍する三年間での研究をまとめたものです．

本研究を遂行するにあたって，多くの方々にご指導，ご教示を頂きました．指導教官ならびに主査である有田隆也先生，鈴木麗璽先生には在籍中の三年間，熱心なご指導をして頂きました．また，副査である北栄輔先生には本論文の執筆にあたって貴重な時間を割いてご指導していただき，大変有益な御助言を頂きました．この場を借りて心から感謝申し上げます．また研究室の皆様とは楽しく，そして切磋琢磨しあうことで大変有意義な研究生活を過ごすことが出来ました．深く感謝致します．

最後になりましたが，御指導，御協力して頂いた教職員の皆様には三年間大変お世話になりました．ここに謹んで感謝の意を表します．

参考文献

[1] G. L. Nemhauser, A. H. G. R. Kan and M. J. Todd. *Optimization*, Vol. 1 of *Handbooks in Operations Research and Management Science*. North-Holland, Amsterdam, 1989.

[2] 久保幹雄, 田村明久, 松井知己. 応用数理計画ハンドブック. 朝倉書店, 2002.

[3] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons Inc., 1989.

[4] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. Optimization by simulated annealing. *Science*, Vol. 220, No. 4598, pp. 671–680, May 1983.

[5] F. Glover. Tabu search – Part I. *ORSA Journal on Computing*, Vol. 1, No. 3, pp. 190–206, 1989.

[6] F. Glover. Tabu search – Part II. *ORSA Journal on Computing*, Vol. 2, No. 1, pp. 4–32, 1990.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[8] J. H. Holland. Genetic algorithms. *Scientific American*, Vol. 267, No. 1, pp. 66–72, 1992.

[9] 棟朝雅晴. 遺伝的アルゴリズム: その理論と先端的手法. 森北出版, 2008.

[10] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[11] M. Dorigo, G. D. Caro and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, Vol. 5, No. 2, pp. 137–172, 1999.

[12] E. Bonabeau, M. Dorigo and G. Theraulaz. Inspiration for optimization from social insect behaviour. *Nature*, Vol. 406, pp. 39–42, 2000.

[13] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942–1948, 1995.

[14] R. Tanese. Distributed genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 434–439. Morgan Kaufmann, 1989.

[15] T. C. Belding. The distributed genetic algorithm revisited. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 114–121. Morgan Kaufmann, 1995.

[16] E. Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles*, Vol. 10, No. 2, 1998.

[17] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, Vol. 42, pp. 228–234, 1990.

[18] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the 2nd International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pp. 41–49. Lawrence Erlbaum Associates, 1987.

[19] G. R. Harik. Finding multimodal solutions using restricted tournament selection. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 24–31. Morgan Kaufmann, 1995.

[20] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 93–100. Lawrence Erlbaum Associates, 1985.

[21] C.M. Fonseca and P.J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 416–423. Morgan Kaufmann, 1993.

[22] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, Vol. 2, No. 3, pp. 221–248, 1994.

[23] E. Zitzler, M. Laumanns and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. In *Proceedings of the EUROGEN Conference*, pp. 95–100, 2002.

[24] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182–197, 2002.

[25] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[26] R. Storn and K. Price. Differential evolution: A simple and efficient heuristic

for global optimization over continuous spaces. *Journal of Global Optimization*, Vol. 11, No. 4, pp. 341–359, 1997.

[27] K. Price, R. M. Storn and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, 2005.

[28] M. Pelikan, D. E. Goldberg and F. G. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, Vol. 21, pp. 5–20, 2002.

[29] J. Paredis. Coevolutionary computation. *Artificial Life*, Vol. 2, No. 4, pp. 355–375, 1995.

[30] C. D. Rosin and R. K. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 373–380. Morgan Kaufmann, 1995.

[31] D. Floreano, S. Nolfi and F. Mondada. Competitive co-evolutionary robotics: From theory to practice. *From Animals to Animats*, Vol. 4, pp. 515–524, 1998.

[32] P. Funes and E. Pujals. Intransitivity revisited coevolutionary dynamics of numbers games. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pp. 515–521. ACM, 2005.

[33] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. *Technical Report CMU-CS-94-163*, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1994.

[34] G. R. Harik, F. G. Lobo and D. E. Goldberg. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 4, pp. 287–297, 1999.

[35] H. Mühlenbein and G. Paass. From recombination of genes to the estimation of distributions I. Binary parameters. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, Vol. 1141 of *Lecture Notes in Computer Science*, pp. 178–187. Springer, 1996.

[36] J. S. D. Bonet, C. L. Isbell and P. Viola. MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems*, Vol. 9, pp. 424–431, 1997.

[37] M. Pelikan and H. Mühlenbein. Marginal distributions in evolutionary algorithms. In *Proceedings of the International Conference on Genetic Algorithms*

Mendel'98, pp. 90–95, 1999.

[38] M. Pelikan, D. E. Goldberg and E. Cantú-Paz. BOA: The bayesian optimization algorithm. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, pp. 525–532. Morgan Kaufmann, 1999.

[39] R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pp. 702–709. Morgan Kaufmann, 2001.

[40] G. Sywerda. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 2–9. Morgan Kaufmann, 1989.

[41] S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, Urbana, IL, USA, 1995.

[42] K. Koper, M. Wysession and D. Wiens. Multimodal function optimization with a niching genetic algorithm: A seismological example. *Bulletin of the Seismological Society of America*, Vol. 89, No. 4, pp. 978–988, 1999.

[43] M. Epitropakis, V. Plagianakos and M. Vrahatis. Finding multiple global optima exploiting differential evolution's niching capability. In *Proceedings of the 2011 IEEE Symposium on Differential Evolution*, pp. 80–87, 2011.

[44] S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 1, pp. 4–31, 2011.

[45] R. Thomsen. Multimodal optimization using crowding-based differential evolution. In *Proceedings of the Congress on Evolutionary Computation 2004*, pp. 1382–1389, 2004.

[46] J. Lampinen. An extended mutation concept for the local selection based differential evolution algorithm. In *Proceedings of the 2007 Conference on Genetic and Evolutionary Computation*, pp. 689–696. ACM, 2007.

[47] B. Qu and P. N. Suganthan. Novel multimodal problems and differential evolution with ensemble of restricted tournament selection. In *Proceedings of the Congress on Evolutionary Computation 2010*, pp. 1–7, 2010.

[48] X. Li. Efficient differential evolution using speciation for multimodal function optimization. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pp. 873–880. ACM, 2005.

[49] 柴坂美祐喜, 原章, 市村匠, 高濱徹行. 種分化を導入した Differential Evolution による複数解をもつ多峰性関数の最適化. 電子情報通信学会論文誌 D, Vol. J92–D, No. 7, pp. 1003–1014, 2009.

[50] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, Vol. 18, No. 9, pp. 590–517, 1975.

[51] P. Indyk, R. Motwani, P. Raghavan and S. Vempala. Locality-preserving hashing in multidimensional spaces. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pp. 618–625, 1997.

[52] 佐藤竜也, 有田隆也. 競合型共進化アルゴリズムにおける過度の特殊化問題に対する適応度共有法の適用. 情報処理学会論文誌, Vol. 49, No. 1, pp. 476–486, 2008.

[53] 廣安知之, 三木光範, 佐野正樹, 谷村勇輔, 濱崎雅弘. 2 個体分散遺伝的アルゴリズム. 計測自動制御学会論文集, Vol. 38, No. 11, pp. 990–995, 2002.

[54] 廣安知之, 三木光範, 下坂久司, 佐野正樹, 筒井茂義. 分散確率モデル遺伝的アルゴリズム. 情報処理学会論文誌, Vol. 45, No. SIG 2 (TOM 10), pp. 56–65, 2004.

[55] 池田心, 小林重信. 生得分離モデルを用いた GA と JSP への適用. 人工知能学会論文誌, Vol. 17, No. 5, pp. 530–538, 2002.

関連発表論文

査読付き原著論文

[1] T. Otani, R. Suzuki and T. Arita. DE/isolated/1: A new mutation operator for multimodal optimization with differential evolution. *International Journal of Machine Learning and Cybernetics*, DOI:10.1007/s13042-012-0075-y, 2012 (7 pages).

[2] T. Otani and T. Arita. Implementation of a probabilistic model-building co-evolutionary algorithm. *Artificial Life and Robotics*, Vol. 13, No. 5, pp. 373–377, 2011.

査読付き国際会議発表

[1] T. Otani, R. Suzuki and T. Arita. DE/isolated/1: A new mutation operator for multimodal optimization with differential evolution. In *Proceedings of the 24th Australasian Joint Conference on Artificial Intelligence*, LNAI 7106, pp. 321–330, Perth – AUSTRALIA, 2011.

[2] T. Otani and T. Arita. An implementation of probabilistic model-building co-evolutionary algorithm, In *Proceedings of the 16th International Symposium on Artificial Life and Robotics*, GS7–4, pp. 715–718, Beppu, Oita – JAPAN, 2011.

国内会議・研究会発表

[1] 大谷隆浩, 鈴木麗璽, 有田隆也. 孤立個体に基づく変異操作を導入した Differential Evolution による多峰性関数最適化. 第 32 回東海ファジィ研究会, 2012.