

平成 30 年度 電気通信大学大学院情報理工学研究科 修士論文

適応的個体間距離に基づく複数解探索型 Bat Algorithm

所属	情報学専攻メディア情報学プログラム
学籍番号	1730022
氏名	岩瀬拓哉
主任指導教員	高玉圭樹教授
指導教員	佐藤寛之准教授
提出日	平成 30 年 1 月 29 日

概要

本論文では, 大域探索と局所探索を調節可能な Bat Algorithm と個体間距離に基づく動的变化を考慮した Niche Radius を組み合わせることによる多峰性最適化問題における, 複数解探索アルゴリズムを構築し, その手法の有効性を検証する. 従来の多点探索アルゴリズムは一つの最適解に収束する傾向にあるが, 実問題への適用を考慮した時に複数の最適解及び局所解を探索する必要がある. 多点探索アルゴリズムの中でも大域と局所の探索バランスに優れた Bat Algorithm は探索する上で全個体の最良解を参照して移動するため, 最終的に最適解に収束することから複数解を同時に探索することは困難である. 本研究では各個体の探索領域を分割させる Niche Radius を用いることで最適解だけでなく, 局所解も同時に探索可能な Bat Algorithm の構築をする. 従来手法と提案手法の性能を比較するため, 最適解と局所解の数が異なるパターンの多峰性関数を用いてシミュレーション実験を行った結果, 全ての関数に対して従来手法は一つの最適解に収束していたが, 提案手法では全最適解及び局所解を探索することができ, 従来に対する変更点が有効であることを示した.

目次

1	はじめに	1
1.1	背景と目的	1
1.2	関連研究	2
1.3	論文構成	3
2	Metaheuristic Algorithms	4
2.1	遺伝的アルゴリズム (GA)	4
2.2	粒子群最適化 (PSO)	5
2.3	差分進化 (DE)	6
2.4	Bat Algorithm (BA)	8
2.5	Evolution Strategies with Covariance Matrix Adaptation (CMA-ES)	10
3	Niching Scheme	13
3.1	Crowding	13
3.2	Niche Radius	13
3.3	Fitness Sharing	13
3.4	Dynamic Niche Sharing	14
4	Niching Methods	15
4.1	Crowding DE (CDE)	15
4.2	A Dynamic Archive Niching Differential Evolution (dADE)	15
4.3	Niching the CMA-ES via Nearest-Better Clustering (NEA)	18
4.4	Parameterless-Niching-Assisted NSGAI (PNA-NSGAI)	19
5	多峰性最適化問題	20
5.1	最小化問題における評価関数	20
5.2	最大化問題における評価関数	28
6	Novelty Search-based Bat Algorithm (NSBA)	33
6.1	Novelty Search	33
6.2	メカニズム	33
6.3	アルゴリズム	34
6.4	実験	34
6.4.1	評価関数	34
6.4.2	評価基準	35
6.4.3	実験設定	35
6.4.4	実験結果	35
6.5	考察	35
6.5.1	個体数による影響	35
6.5.2	解の分布	36

7	Niche Radius-based Bat Algorithm (NRBA)	45
7.1	メカニズム	45
7.2	アルゴリズム	46
7.3	実験	47
7.3.1	評価関数	47
7.3.2	評価基準	47
7.3.3	実験設定	47
7.3.4	実験結果	47
7.4	考察	48
7.4.1	解の発見数	48
7.4.2	最終世代における解の分布	48
8	Dynamic Niche Radius-based Bat Algorithm (DNRBA)	54
8.1	メカニズム	54
8.2	アルゴリズム	54
8.3	実験	54
8.3.1	評価関数	55
8.3.2	評価基準	55
8.3.3	Peak Ratio	55
8.3.4	Peak Accuracy	56
8.4	実験設定	56
8.5	実験結果と考察	56
8.5.1	Peak Ratio	57
8.5.2	Peak Accuracy	57
8.6	まとめ	57
9	他の最先端手法との性能比較実験	60
9.1	複数解探索における多峰性最適化問題	60
9.2	評価関数	60
9.3	評価基準	60
9.4	結果	60
9.5	考察	63
10	おわりに	64
10.1	まとめ	64
10.2	今後の課題	64
	謝辞	65
	参考文献	66
	付録	68

1 はじめに

1.1 背景と目的

近年, 多点探索アルゴリズムは最適化問題において, 一般的なメタヒューリスティック手法として用いられるようになった. 多点探索アルゴリズムは特に非線形な問題に対しても適用することが可能であり, 魚や鳥の群れをモデルにした Particle Swarm Optimization(PSO)[6] や, ホタルの光強度により互いのホタルが引き寄せられる Firefly Algorithm(FA)[?] は高次元な最適化問題に対して有効であることを示している. 中でも Bat Algorithm(BA) は大域探索と局所探索の性能を自動で切り替えるという点で優れたアルゴリズムである [14]. しかし多峰性最適化問題における, 従来の多点探索アルゴリズムは全個体の中の最良解を参照して一点へ移動するため, 探索終了時に一つの最適解に収束する傾向にあるが, 実環境への適用を考慮した時に最適解だけでなく局所解を探索し, 保持しておくことは非常に重要な意味を持つ. 応用先の一例として, 災害時における被災者を解, 救助ロボットを個体と見立てた時に, 不特定多数の被災者を探索することは人間にとって困難であり, 複数の解を同時に探索しなければならない.

そこで本研究では, 探索範囲の自動調整可能な BA を用い, 各個体の探索領域を分割させることで個体の分散化を図る. 探索空間のスケールと解の数から算出される Niche Radius[?] を用いることで, 予め各個体の探索領域を決定し, その探索領域内の最良個体から遠ざかる方向へ移動することで, 個体同士が同じ解に留まらず, 散らばるように改良する. 従来手法の探索アルゴリズムに対して3つの変更をした. (i) 大域探索: 分割探索領域内の最良解を参照; (ii) 局所探索: 分割探索領域内の最良解付近に解候補を生成; (iii) ランダム探索: 選択した個体の分割領域内にランダムで解候補を生成. これらの変更により, 従来手法と提案手法の探索性能を比較するため, 発見した最適解及び局所解の数を評価尺度として, 複数の異なる峰を持つ多峰性関数を用いてシミュレーション上で実験を行う.

1.2 関連研究

1.3 論文構成

本論文の構成は次の通りである. 2 章で最適化問題において一般的に用いられる EAs について説明し, 3 章で実問題を模擬的に表した多峰性最適化問題における複数解探索手法の機構である Niching Scheme について説明する. 4 章では, EAs と Niching Scheme を組み合わせた複数解探索手法を説明し, 6 章から 8 章までは提案手法である NSBA, NRBA 及び DNRBA の説明をする.

2 Metaheuristic Algorithms

現実問題は非常に複雑であり，その複雑さを多峰性関数として表現した問題を最適化するために Evolutionary Algorithms (EAs) が用いられるようになった．EAs のメカニズムとして単一の最適解を探索することを目的として設計されており，生物の生殖や突然変異，交叉，適者生存といった過程をモデルとしている．その代表的なアルゴリズムを 2.1 節から 2.5 節で紹介する．

2.1 遺伝的アルゴリズム (GA)

遺伝的アルゴリズム (Genetic Algorithm:GA) [5] は生物の進化の過程を模擬したアルゴリズムであり，最適化問題において最も基本的なヒューリスティック手法である．生物は次の世代により良い遺伝子を持った個体を残すため，まずは親集合の中から個体を選択し，選択した個体同士を交叉させる．この時，ある一定の確率で突然変異させる．次に，交叉させて新たに生成された個体を子集合の解候補とし，親と子の個体を評価して，評価値の高い個体は次世代の個体として保存され，評価値の低い個体は淘汰される．各個体に対し，この「選択」，「交叉」，「突然変異」，「評価 (淘汰)」を繰り返すことで，環境に対する適合度が高くなっていく (評価値の高い遺伝子が残る)．アルゴリズムの疑似コードは，以下の Algorithm1 に記す．

- **STEP1: 個体の初期化**

N 個の個体 x_i を初期集合として生成し，ランダムな評価値を割り当てる．また交叉率と突然変異率を定義する．(1-2 行目)

- **STEP2: 選択と交叉**

選択した親同士 x_i を交叉率 P_c により交叉させ，子 (解候補) x_i^{new} を生成する．(5-7 行目)

- **STEP4: 突然変異**

突然変異率 P_m により，子 (解候補) の評価値を変化させる．(8-10 行目)

- **STEP5: 評価**

親個体 x_i と子個体 x_i^{new} の評価値を比較し，評価値の高い個体を次世代に残す．(11-13 行目)

Algorithm 1 Genetic Algorithm**Input:** Objective Function $F(x)$, $x = (x_1, x_2, \dots, x_d)$

```

1: Initialize Population  $x_i (i = 1, 2, \dots, N)$ 
2: Initialize  $P_m, P_c$ 
3: while  $t < \text{Max Generation}$  do
4:   for  $i=1$  to  $N$  do
5:     if  $\text{rand}(0, 1) < P_c$  then
6:       Generate an offspring  $x_i^{\text{new}}$ 
7:     end if
8:     if  $\text{rand}(0, 1) < P_m$  then
9:       Replace mutated offspring  $x_i^{\text{new}}$ 
10:    end if
11:    if  $F(x_i) < F(x_i^{\text{new}})$  then
12:      Replace  $x_i$  with  $x_i^{\text{new}}$ 
13:    end if
14:  end for
15:   $t=t+1$ 
16: end while

```

2.2 粒子群最適化 (PSO)

最適化手法の一つとして粒子群最適化 (Particle Swarm Optimization: PSO) がある。PSO は、ランダムに個体を初期化するという点において、遺伝的アルゴリズム (GA) に似た性質を持つ。魚や鳥の群れの動きをモデルにしたアルゴリズムであり、個体間のユークリッド距離を速度として新たに解候補を生成する。個体の速度及び生成式は以下の通りである。

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + c_1r_1 \cdot (\mathbf{x}_{pbest}^t - \mathbf{x}_i^t) + c_2r_2 \cdot (\mathbf{x}_{gbest}^t - \mathbf{x}_i^t) \quad (2.1)$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \quad (2.2)$$

\mathbf{x}_i , \mathbf{v}_i は時刻 t における各個体の現在位置と速度を表し, w, c_1, c_2 は係数, r_1, r_2 は一様乱数を表す。これらの係数が速度を制限するパラメータとなっており, w が時刻 t での速度を調整し, c_1, c_2 は値が 1 より小さいほど局所探索を行い, 1 より大きくなると最良個体を含む広い範囲を大域探索するようになる。PSO のアルゴリズムの疑似コードである Algorithm2 を以下に記す。

Algorithm 2 Particle Swarm Optimization

Input: Objective Function $F(x)$, $x = (x_1, x_2, \dots, x_d)$
 Initialize Population $x_i (i = 1, 2, \dots, N)$ and Velocity v_i
 2: $F(x_{gbest}) = \max F(x_i)$
 Initialize c_1, c_2, w
 4: **while** $t < \text{Max Iteration}$ **do**
 for $i=1$ to N **do**
 6: Generate a new solution x_i^{t+1} and update v_i [Eqs.(2.1),(2.2)]
 if $F(x_i^{t+1}) > F(x_{pbest})$ **then**
 8: Replace the individual x_i^{t+1} as x_{pbest}
 end if
 10: **if** $F(x_{gbest}) < F(x_{pbest})$ **then**
 $x_{gbest} = x_{pbest}$
 end if
 end for
 14: $t=t+1$
end while

2.3 差分進化 (DE)

差分進化 (Differential Evolution:DE) [10] は進化的計算手法の一つであり, 問題に応じて個体間同士の相対距離に基づいた探索戦略を用いることのできる, 他のアルゴリズムとは異なる特徴を持つ. DE は以下の手順により探索を繰り返す.

- **STEP1: 個体の初期化**

探索領域内にランダムで個体を生成する.

- **STEP2: 突然変異による解候補の生成**

DE は以下, いずれかの探索戦略を用いて解候補を生成する.

1. DE/rand/1

$$v_{i,j}^{t+1} = x_{r1,j}^t + F \cdot (x_{r2,j}^t - x_{r3,j}^t) \quad (2.3)$$

2. DE/best/1

$$v_{i,j}^{t+1} = x_{gbest,j}^t + F \cdot (x_{r1,j}^t - x_{r2,j}^t) \quad (2.4)$$

3. DE/current-to-best/1

$$v_{i,j}^{t+1} = x_{i,j}^t + F \cdot (x_{gbest,j}^t - x_{i,j}^t) + F \cdot (x_{r1,j}^t - x_{r2,j}^t) \quad (2.5)$$

4. DE/rand/2

$$v_{i,j}^{t+1} = x_{r1,j}^t + F \cdot (x_{r2,j}^t - x_{r3,j}^t) + F \cdot (x_{r4,j}^t - x_{r5,j}^t) \quad (2.6)$$

5. DE/best/2

$$v_{i,j}^{t+1} = x_{gbest,j}^t + F \cdot (x_{r1,j}^t - x_{r2,j}^t) + F \cdot (x_{r3,j}^t - x_{r4,j}^t) \quad (2.7)$$

この時, i, j は個体番号と次元数を表し, r_1, \dots, r_4 は 1 から N までの一様乱数の整数を表す. スケール因子 F は解候補を生成するためのベクトルを制御するための変数で, 0 から 1 までの乱数で表される. 「DE/rand/1」は, 個体の親集団の中から 3 つの個体をランダムに選択し, その相対距離を用いて新たに解候補 $v_{i,j}^{t+1}$ を生成する. 「DE/best/1」は, 個体の親集団の中の最良個体と個体を 2 つランダムに選択し, その最良個体付近に新しい解候補 $v_{i,j}^{t+1}$ を生成する. 「DE/current-to-best/1」は, ランダムに選択した 2 つの個体と, 最良個体と個体自身の相対距離と用い, 最良個体方向へ新たに解候補 $v_{i,j}^{t+1}$ を生成する. 「DE/rand/2」では, 親集合から 5 つの個体をランダムに選択し, 式 (2.3) よりも広い探索領域内に新しく解候補 $v_{i,j}^{t+1}$ を生成する. 「DE/best/2」も同様, 式 (2.4) よりも広い探索領域内に新しく解候補 $v_{i,j}^{t+1}$ を生成する.

- **STEP3: 交叉**

解に多様性を持たせるため, ここではSTEP2で生成した解候補 $v_{i,j}^{t+1}$ と個体 x_i を確率的に交叉させ, 新たに解 $u_{i,j}^{t+1}$ を生成する. 生成式は次式の通りである.

$$u_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1} & \text{if } (rand(0, 1) \leq C_r) \text{ or } j = D \\ x_{i,j}^{t+1} & \text{if } (rand(0, 1) > C_r) \text{ and } j \neq D \end{cases} \quad (2.8)$$

ここで C_r は $[0, 1]$ の範囲内での交叉係数を表す. この交叉係数 C_r が一様乱数以上であればSTEP2で生成した解候補 $v_{i,j}^{t+1}$ が採用され, 一様乱数未満であれば個体 $x_{i,j}$ が採用される.

- **STEP4: 評価**

最良個体と生成した解 $u_{i,j}^{t+1}$ を比較し, 評価値の高い方を次世代に引き継ぐ.

- **STEP5: 終了条件を満たすまでSTEP2へ戻る**

ここでは, 一般的に用いられる DE/rand/1 のアルゴリズムの疑似コードを以下の Algorithm3 に記す.

Algorithm 3 Differential Evolution (DE/rand/1)

Input: Objective Function $f(x)$, $x = (x_1, x_2, \dots, x_d)$
 Initialize Population $x_i (i = 1, 2, \dots, N)$
 Define crossover probability C_r , and scaling factor F

```

3: while t < Max Iteration do
    for i=1 to N do
        Select random integer  $r_1, r_2 \in \{1, 2, \dots, N | r_1 \neq r_2 \neq i\}$ 
6:        Generate a candidate  $v_{i,j}$  [Eq. (2.3)]
        for j = 1 to D do
            if  $\text{rand}(0, 1) \leq C_r$  or  $j = D$  then
9:                offspring  $u_{i,j} = v_{i,j}^{t+1}$ 
            else  $\{\text{rand}(0, 1) > C_r \text{ and } j \neq D\}$ 
                offspring  $u_{i,j} = x_{i,j}^t$ 
12:            end if
        end for
        if  $f(u_{i,j}^{t+1}) > f(x_{pbest})$  then
15:            Replace the offspring  $u_{i,j}^{t+1}$  as  $x_{pbest}$ 
        end if
    end for
18:    t=t+1
end while

```

2.4 Bat Algorithm (BA)

Bat Algorithm(BA) [14] は群知能アルゴリズムの一つで、対象物までの方向や距離を知るコウモリの特徴（エコロケーション）を利用して周囲の状況を認知し、大域探索と局所探索が進むにつれて探索速度を徐々に落とし、探索性能を自動調節することが可能なアルゴリズムである。各個体の周波数 f_i 、速度 v_i 、位置 x_i は以下の式で定義し、更新される。ラウドネス A は、コウモリが対象物に近づくとき値が減少し、移動距離も比例して短くなる。コウモリの行動は以下3つで構成される。

- 大域探索: 各コウモリは位置 x_i において、自身が発する周波数 f_i の反響によって対象物との距離を測り、対象物に向かって速度 v_i で移動する。
- 局所探索: 対象物近辺にコウモリを移動させる。
- ランダム探索: 探索領域内にコウモリをランダムで移動させる。

BA で扱う各個体の周波数 f_i 、速度 v_i 、位置 x_i は以下の式で定義される。

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (2.9)$$

$$v_i^{t+1} = v_i^t + (x_* - x_i^t) * f_i \quad (2.10)$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \quad (2.11)$$

個体番号を i とし, 各個体の周波数 f_i は個体の速度を制限するパラメータであり, $[0, 1]$ の区間で表される. ここでは $f_{min} = 0$, $f_{max} = 1$ として設定し, β は 0 から 1 の乱数が割り当てられる. 局所探索では, 全個体の最良解 (グローバルベスト) \mathbf{x}_* の周辺に新しい解候補 \mathbf{x}_{loc} を生成する. 生成式は次の通りである.

$$\mathbf{x}_{loc} = \mathbf{x}_* + \epsilon \mathbf{A}_i^t \quad (2.12)$$

パラメータ ϵ は $1 \times D$ 次元の配列で $[-1, 1]$ 区間のランダムな値が割り当てられる. ランダム探索では解探索空間にランダムで新たに解候補を生成する. 生成式は以下の通りである.

$$\mathbf{x}_{rnd} = \mathbf{x}_{lb} + (\mathbf{x}_{ub} - \mathbf{x}_{lb}) * rand(1, D) \quad (2.13)$$

解探索空間の上限と下限をそれぞれ $\mathbf{x}_{ub}, \mathbf{x}_{lb}$ とし, $rand$ は 0 から 1 までの乱数が入る. 以上より各個体の解候補 \mathbf{x}_i^{t+1} , \mathbf{x}_{loc} , あるいは \mathbf{x}_{rnd} の評価値が各個体の最良解 (パーソナルベスト) \mathbf{x}_{i*} より良ければ更新され, 同時にラウドネス A とその反射波であるパルスレート r も以下の式に基づいて更新される.

$$A_i^{t+1} = \alpha A_i^t \quad (2.14)$$

$$r_i^{t+1} = r_i^t [1 - \exp(-\gamma t)] \quad (2.15)$$

解を更新する度にラウドネス A_i は徐々に減少し, それに比例して評価頻度を下げていく. 対してパルスレート r_i は増加していき, 探索が進むにつれて局所探索頻度が減少する. 従来の BA の疑似コードは以下の Algorithm 4 に記す.

Algorithm 4 Bat Algorithm

Input: Objective Function $F(x)$

```

1: Initialize Population  $x_i (i = 1, 2, \dots, N)$  and  $v_i$ 
2: Define frequency  $f_i$  at location  $x_i$  [eq.(2.9)]
3: Initialize pulse rates  $r_i$ , and loudness  $A_i$ 
4: while ( $t < \text{Max number of iterations}$ ) do
5:   for  $i=1$  to  $N$  do
6:     Generate a new solution  $x_i$  and velocity  $v_i$  [Eqs.(2.10) to (2.11)]
7:     if ( $rand > r_i$ ) then
8:       Generate a new solution  $x_{loc}$  around a global best solution  $x_i$  [Eq.(2.12)]
9:     end if
10:    Generate a new solution  $x_{rnd}$  randomly [Eq.(2.13)]
11:    if ( $rand < A_i \& \min(F(x_i), F(x_{loc}), F(x_{rnd})) < F(x_{i*})$ ) then
12:      Accept the new solution, and update pulse rate  $r_i$ 
        & loudness  $A_i$  [Eqs. (2.14)(2.15)]
13:    end if
14:    Evaluate all bats and select a best solution  $x_*$  in the current solutions
15:  end for
16:   $t=t+1$ 
17: end while

```

2.5 Evolution Strategies with Covariance Matrix Adaptation (CMA-ES)

Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [3][4] は多変量正規分布を用いて解候補を生成し、それらの評価値を基に算出する共分散行列から探索範囲を決定する。CMA-ES の大きな特徴として、変数間の依存度を考慮している (パラメータチューニングを必要としない) という点と、単調に増減する線形的な問題に依存しないという点が挙げられる。CMA-ES には様々な手法がある中で、ここでは一般的な $(\mu|\mu_w, \lambda)$ -CMA-ES [4] を紹介する。具体的なアルゴリズムについては以下の Algorithm 5 に示し、終了条件を満たすまで以下の手順を繰り返す。

- **STEP1: 個体の生成**

まず、正規分布 $N(0, I)$ の範囲で解候補 $z_i (i = 1, 2, \dots, \lambda)$ を独立に生成し、それを基に設計変数 $x_i (i = 1, 2, \dots, \lambda)$ を生成する。生成式は以下の通りである。

$$y_i^t = \sqrt{C^t} z_i \quad (2.16)$$

$$x_i^{t+1} = m^t + \sigma^t y_i \quad (2.17)$$

設計変数 x_i は共分散行列 $N(0, (\sigma^t)^2 C^t)$ の範囲から生成される。生成後、 x_i を評価値で降順 (評価値の高い順) にソートして順位付けし、 y_i 及び z_i も同様に順位付けする。 C^t は、その共分散行列の範囲の広がり度合いを $d \times d$ 次元で表し、 σ^t はステップサイズを、 m^t は平均値ベクトル (探索範囲の中心) を表す。初期個体生成時、 $C^0 = I, \sigma^0 = 0$ とする。また、 $\lambda = 4 + [3 \ln(n)]$, $\mu = [\frac{\lambda}{2}]$ とする。

- **STEP2: μ 個体の荷重和を算出し、平均ベクトル m_i^t を更新**

評価値で降順ソートした設計変数 x_i のうち、上位 μ 個の荷重和を算出し、次式に従って平均値ベクトル m を更新する。

$$m^{t+1} = m^t + \sum_{i=1}^{\mu} w_i (x_i^t - m^t) \quad (2.18)$$

この時、重み w は次式で表される。

$$w_i = \ln\left(\frac{\lambda+1}{2}\right) - \ln(i) \quad (2.19)$$

重み w は $w_1 \geq w_2 \geq \dots \geq w_\mu > 0$ であり、 $\sum_{i=1}^{\mu} w_i = 1$ を満たす。

- **STEP3: ステップサイズ σ の更新**

次式に従って進化パス p_σ 及びステップサイズ σ を更新する。

$$p_\sigma^{t+1} = (1 - c_\sigma) p_\sigma^t + \sqrt{c_\sigma(2 - c_\sigma)} \sqrt{\mu_{eff}} \sum_{i=1}^{\mu} w_i z_i \quad (2.20)$$

$$\sigma^{t+1} = \sigma^t \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|p_\sigma^{t+1}\|}{\hat{\chi}_n} - 1\right)\right) \quad (2.21)$$

この時, c_σ は進化パス p_σ の前世代との重みを表し, μ_{eff} は設計変数の上位 μ 個の加重平均の補正值, d_σ はステップサイズの減衰係数, $\hat{\chi}_n$ は n 変量正規分布のノルムの期待値を表す. これらのパラメータは次式で表される.

$$c_\sigma = \frac{4}{n+4} \quad (2.22)$$

$$\mu_{eff} = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \quad (2.23)$$

$$d_\sigma = \frac{1}{c_\sigma} + 1 \quad (2.24)$$

$$\hat{\chi}_n = E[||N(0,1)||] \approx \sqrt{n}(1 - \frac{1}{4n} + \frac{1}{21n^2}) \quad (2.25)$$

ここで n は最適化する数を表す.

• **STEP4: 共分散行列 C の更新**

共分散行列 C 及びその進化パス p_c は次式で更新される.

$$p_c^{t+1} = (1 - c_c)p_c^t + \sqrt{c_c(2 - c_c)} \sum_{i=1}^{\mu} w_i \mathbf{y}_i^t \quad (2.26)$$

$$\mathbf{C}^{t+1} = \mathbf{C}^t + c_1[p_c^{t+1}(p_c^{t+1})^T + (1 - c_c(2 - c_c))\mathbf{C}^t] + c_\mu \sum_{i=1}^{\mu} w_i(\mathbf{y}_i(\mathbf{y}_i)^T - \mathbf{C}^t) \quad (2.27)$$

c_σ は共分散行列 C の進化パス p_c の前世代との重みであり, c_1 及び c_μ は共分散行列 C の更新に用いられる学習率を表す. これらのパラメータは次式で表される.

$$c_1 = \frac{2}{(n + \sqrt{2})^2} \quad (2.28)$$

$$c_\mu = 1 - c_1 \quad (2.29)$$

$$c_c = \frac{4}{n+4} \quad (2.30)$$

Algorithm 5 CMA-ES

Input: Objective Function $F(x), x = (x_1, x_2, \dots, x_d)$

Calculate parameters [Eqs. (2.22) to (2.25), (2.28) to (2.30)]

while $t < \text{Max Iteration}$ **do**

for $i=1$ to N **do**

 Generate solution $x_i (i = 1, 2, \dots, \lambda)$ within $N(0, \sigma^2 C)$ [Eqs. (2.16),(2.17)]

5: **end for**

 Evaluate and sort solution x_i

for $i=1$ to N **do**

 Update m_i by w_i [Eqs. (2.19),(2.18)]

 Update step size σ and covariance matrix C by the evolution path p_σ and p_c
 [Eqs. (2.20),(2.21),(2.26),(2.27)]

10: **end for**

$t=t+1$

end while

3 Niching Scheme

この章では、多峰性最適化問題において、EAs を拡張させて複数の最適解及び局所解探索を行うための機構である Niching Scheme について説明する。EAs は一つの最適解を探索することを目的とした設計であるため、複数の局所解を探索し、保持するには限界がある。この問題を解決するために導入した Niching Scheme について 3.1 節から 3.4 節で紹介する。

3.1 Crowding

Crowding [1] は評価値が類似する個体が同じ場所付近に位置しているとき、その最近傍個体同士を比較し、評価値の高い方を残す。この時、比較する個体は親集合と子集合を含めた全個体における最近傍個体同士が比較対象となる。この動作を毎世代繰り返すことによって同じ局所解に個体が陥らないことを目的とした機構である。

3.2 Niche Radius

Niche Radius は探索空間の大きさと局所解数 (あるいは個体数) に基づいて算出される距離であり、Fig. は理想的な個体の分布及び Niche radius を示している。次式で表される。

$$dist = \frac{1}{2} \sqrt{(x_{ub} - x_{lb})^2} \quad (3.1)$$

$$\sigma = \frac{dist}{\sqrt[q]{q}} \quad (3.2)$$

この時、 x_{ub}, x_{lb} は探索空間の上限と下限を表しており、 D は次元数を表す。 q は解の数 (あるいは個体数) が適用される。

3.3 Fitness Sharing

Fitness Sharing [2] は Crowding と同様、類似個体の評価値が低い方を淘汰させるための機構として用いられる。ここでは、その類似度を定義することで、同じ類似度を持つ個体同士が評価値を比較する。一般的に、類似度の算出方法は次式で表される。

$$sh(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma})^\alpha & (\text{if } d_{ij} < \sigma) \\ 0 & (\text{otherwise}) \end{cases} \quad (3.3)$$

ここで d_{ij} は個体 i, j の間の距離を表し、 α は係数で σ はある恣意的な閾値 (あるいは Niche radius) を表す。個体間距離が近いほど Sharing function $sh(d_{ij})$ の値は大きくなり、この数値を基に Niche count m_i を算出する。

$$m_i = \sum_{j=1}^N sh(d_{ij}) \quad (3.4)$$

Niche count m_i は i 番目の個体に対する全個体の密度を表している. この Niche count より, Shared fitness f'_i が次式のように算出される.

$$\phi_i = \frac{F_i}{m_i} \quad (3.5)$$

ここで, F_i は i 番目の個体の評価値を表す. Shared fitness ϕ_i は, その個体密度である Niche count m_i によって値が大きく変動する. ϕ_i の値が小さいほど i 番目の個体密度は高く, 値が大きければ個体密度は低いということになる.

3.4 Dynamic Niche Sharing

3.3 節で前述した Shared fitness を用い, Dynamic niche sharing [8] は次式で表される.

$$m_i^{dyn} = \begin{cases} n_j & (\text{if individual } i \text{ is within the dynamic niche } j) \\ m_i & (\text{otherwise}) \end{cases} \quad (3.6)$$

4 Niching Methods

4.1 Crowding DE (CDE)

2.3 節で説明した DE をベースとし, Crowding [1] と組み合わせた手法で, 生成された子集合の中から最近傍である子 $u_{i,j_{NN}}$ と親 x_{pbest_i} を選択し, 評価値の低い個体は淘汰され, 評価値の高い方は次世代に引き継がれる.

Algorithm 6 CDE/rand/1)

Input: Objective Function $f(x)$, $x = (x_1, x_2, \dots, x_d)$

Initialize Population $x_i (i = 1, 2, \dots, N)$

Define crossover probability C_r , and scaling factor F

```

3: while t < Max Iteration do
    for i=1 to N do
        Select random integer  $r_1, r_2 \in \{1, 2, \dots, N | r_1 \neq r_2 \neq i\}$ 
6:      Generate a candidate  $v_{i,j}$  [Eq. (2.3)]
        for j = 1 to D do
            if  $\text{rand}(0, 1) \leq C_r$  or  $j = D$  then
9:              offspring  $u_{i,j} = v_{i,j}^{t+1}$ 
            else  $\{\text{rand}(0, 1) > C_r \text{ and } j \neq D\}$ 
                offspring  $u_{i,j} = x_{i,j}^t$ 
12:          end if
        end for
        if  $f(u_{i,j_{NN}}^{t+1}) > f(x_{pbest_i})$  then
15:            Replace the offspring  $u_{i,j_{NN}}$  as  $x_{pbest_i}$ 
        end if
    end for
18:  t=t+1
end while

```

4.2 A Dynamic Archive Niching Differential Evolution (dADE)

Niching methods は, 最適解の数が多くなるほど, 個体数を増やすといったアルゴリズムのパラメータチューニングが必要となる. そこで, 本アルゴリズムは動的にパラメータを制御する 2 つのメカニズムの改良を行った. 1 つ目のメカニズムとして, 環境に応じてパラメータの値を自動的に変化させ, 2 つ目のメカニズムとして未探索領域に解候補を生成するよう変更を加えた.

- **STEP1:** 最近傍個体間距離 R の算出

DE は主に, 突然変異率 C_r とスケール因子 F の制御によって探索を決定しているが, 複数解探索を行う上では, 探索領域全体を網羅するよう個体を配置させることが重要となる. ここではまず, 個体の密度を算出する.

$$R = \min(r^1, r^2, \dots, r^t) \quad (4.1)$$

$$r^t = \frac{\sum_{i=1}^n dist_i}{n} \quad (4.2)$$

$$dist_i = \min \|x_i^t - x_j^t\| : \forall x_i^t, x_j^t \in P \quad (4.3)$$

この時、 r^t は各個体の最近傍個体とのユークリッド距離の平均値を示す。個体が同じ場所に密集しているほど r の値は 0 に近づくことを意味する。

- **STEP2: 突然変異率 C_r とスケール因子 F の更新**

各世代において、突然変異率 C_r とスケール因子 F は、2 つの確率分布 $CR_i \sim N(\mu_{CR}, 0.1)$, $F_i \sim Cauchy(\mu_F, 0.1)$ に従って生成される。

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot \text{mean}_L(S_F) \quad (4.4)$$

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot \text{mean}_A(S_{CR}) \quad (4.5)$$

ここで、 c は $c \in [0, 1]$ の係数 (既定値は 0.1) であり、 $\text{mean}_L(\cdot)$ は Lehmer mean [15] を指し、次式で表される。

$$\text{mean}_L = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \quad (4.6)$$

$\text{mean}_A(\cdot)$ は t 世代目の突然変異率ベクトル S_{CR} の平均値を示し、 μ_{CR} と μ_F は

- **STEP3: 交叉**

2.3 節と同様、突然変異率 μ_{CR} に基づいて解候補 $u_{i,j}$ を決定する。

- **STEP4: 評価と更新**

最良個体と生成した解 $u_{i,j}^{t+1}$ を比較し、評価値の高い方を次世代に引き継ぐ。

- **STEP5: Dynamic Archive による各個体の最良解 x_{pbest_i} の再生成**

Archive の疑似コードを Algorithm 8 に記す。各個体の最良解 x_{pbest_i} を p として集合 S に格納し、 δ よりも評価値が良ければ update に TRUE を返す (2-7 行目)。次に、update が TRUE、あるいは δ の評価値と p の評価値の差分が accuracy level: ε よりも小さければ (8 行目)、 p と s のユークリッド距離に着目し、その距離が R 以下であれば s は新しく p として置き換えられ、found に TRUE を返し (9-14 行目)、 R より大きければ found に TRUE を返す (15-18 行目)。最後に、found が FALSE ならば、集合 S に p を追加する (21-24 行目)。この手順を繰り返すことで、閾値 ε を満たす最近傍個体同士を比較し、評価値の低い個体は新たに生成される。

- **STEP6: STEP1 へ戻る**

終了条件を満たすまで STEP1 から STEP5 までの手順を繰り返す。

Algorithm 7 dADE/nrand/1

```

Initialize Population  $x_i (i = 1, 2, \dots, N)$ 
Define crossover probability  $C_r$ , and scaling factor  $F$ 
3: while  $t < \text{Max Iteration}$  do
    Calculate the  $R$  value [Eq. (4.2)]
    for  $i=1$  to  $N$  do
6:     Update parameters  $CR$  and  $F$  [Eqs. (4.4),(4.5)]
        Select random integer  $r_1, r_2 \in \{1, 2, \dots, N | r_1 \neq r_2 \neq i\}$ 
        Generate a candidate  $v_{i,j}$  [Eq. (2.3)]
9:     for  $j = 1$  to  $D$  do
        if  $\text{rand}(0, 1) \leq C_r$  or  $j = D$  then
            offspring  $u_{i,j} = v_{i,j}^{t+1}$ 
12:        else  $\{\text{rand}(0, 1) > C_r \text{ and } j \neq D\}$ 
            offspring  $u_{i,j} = x_{i,j}^t$ 
        end if
15:    end for
        if  $f(u_{i,j_{NN}}^{t+1}) > f(x_{pbest_i})$  then
            Replace the offspring  $u_{i,j_{NN}}$  as  $x_{pbest_i}$ 
18:        Insert  $x_{pbest_i}$  to Dynamic Archive [Algorithm 8]
            if  $x_{pbest_i}$  is already in Archive ( $\text{found} == \text{TRUE}$ ) then
                Re-initialize individual  $x_{pbest_i}$ 
21:        end if
    end if
    end for
24:     $t=t+1$ 
end while

```

Algorithm 8 The algorithmic scheme for building a dynamic archive

Input: $p \in \mathbb{R}^D$ a potential solution, $R \in \mathbb{R}$ the niche radius, and ε an accuracy level or acceptance threshold

Output: S - a list of found solutions in a dynamic archive

```

found  $\leftarrow$  FALSE; update  $\leftarrow$  FALSE;
if  $S = \emptyset$  then
3:    $S \leftarrow S \cup p$ ;  $\delta \leftarrow f(p)$ 
else
   if  $f(p) > \delta$  then
6:    $\delta \leftarrow f(p)$ ; update  $\leftarrow$  TRUE
   end if
   if update or  $|f(p) - \delta| < \varepsilon$  then
9:   for each  $s \in S$  do
       if  $\|p - s\| \leq R$  then
           if  $f(p) > f(s)$  then
12:             $s \leftarrow p$ ;
              found  $\leftarrow$  TRUE;
              break;
15:         else
              found  $\leftarrow$  TRUE;
              break;
18:         end if
           end if
       end for
21:   if not found then
        $S \leftarrow S \cup p$ 
       end if
24: end if
end if

```

4.3 Niching the CMA-ES via Nearest-Better Clustering (NEA)

Algorithm 9 Nearest-better clustering (NBC)

Calculate all search points mutual distances

Create an empty graph with num (search points) nodes

```

3: for the search points do
    Find nearest search point that is better, create edge to it
    end for
6: Delete edges of length  $> \emptyset \cdot \text{mean}(\text{length of all edges})$ 
   Find connected components

```

4.4 Parameterless-Niching-Assisted NSGAI (PNA-NSGAI)

5 多峰性最適化問題

最適化問題において、実問題の複雑さを多峰性で見立てた評価関数を用いることが一般的である。その代表的な多峰性関数を次節で説明する。

5.1 最小化問題における評価関数

F_1 : Griewank (2D)

Griewank 関数 [9] の概形を図 5.1 に示し、関数式は次式で表される。

$$F_1(x_i) = \sum_{i=1}^D \frac{x_i}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \quad (5.1)$$

解空間の探索領域は $x_i \in [-10, 10]$ である ($i = 1, 2$)。最適解の座標は $x_* = [0, 0]$ で、その評価値は $F(x_*) = 0$ である。局所解の座標は $\pm x \approx [6.2800, 8.8769], [3.1400, 4.4385], [0, 8.8769], [6.2800, 0], [9.4200, 4.4385]$ となる。

F_2 : Rastrigin (2D)

Rastrigin 関数 [9] の概形を図 5.2 に示し、関数式は次式で表される。

$$F_2(x_i) = 10D + \sum_{i=1}^D [x_i^2 - 10 \cos(2\pi x_i)]. \quad (5.2)$$

D は次元数であり、探索領域は $x_i \in [-5, 5]$ である。

F_3 : Six-Hump Camel (2D)

最適解と局所解が各 2 個存在する Six-Hump Camel Function [9] は以下の式で表される。

$$F_3(x_1, x_2) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \quad (5.3)$$

この関数における解空間の探索領域は $x_1 \in [-2, 2]$, $x_2 \in [-1, 1]$ である。最適解の座標は $x_* = [0.0898, -0.7126], [-0.0898, 0.7126]$ であり、その評価値は $F_3(x_*) = -1.0316$ である。また局所解は $\pm x \approx [1.704, -0.7965]$ に位置する。

F_4 : Michalewicz (2D)

Michalewicz Function [9] の数式を以下に示す。

$$F_4(x_i) = - \sum_{i=1}^D \sin(x_i) \sin^{2m}\left(\frac{ix_i^2}{\pi}\right) \quad (5.4)$$

最適解 $x_* = [2.20, 1.57]$ の評価値 $F_3(x_*) = -1.8013$ であり、局所解は $x \approx [2.203, 2.7115]$ である。探索領域は $x_i \in [0, 4]$ である ($i = 1, 2$)。

F_5 : Himmelblau (2D)

Himmelblau Function [?] の関数式は次の通りとなる.

$$F_5(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \quad (5.5)$$

評価関数に局所解は存在せず, 最適解のみ 4 個持つ関数である. 最適解の位置は各々 $x_* = [3, 2], [-2.805118, 3.283186], [-3.779310, -3.283186], [3.584458, -1.848126]$ にあり, その評価値は $F_4(x_*) = 0$ である. 探索領域は $x_i \in [-5, 5]$ となる ($i = 1, 2$).

F_6 : Shubert Function

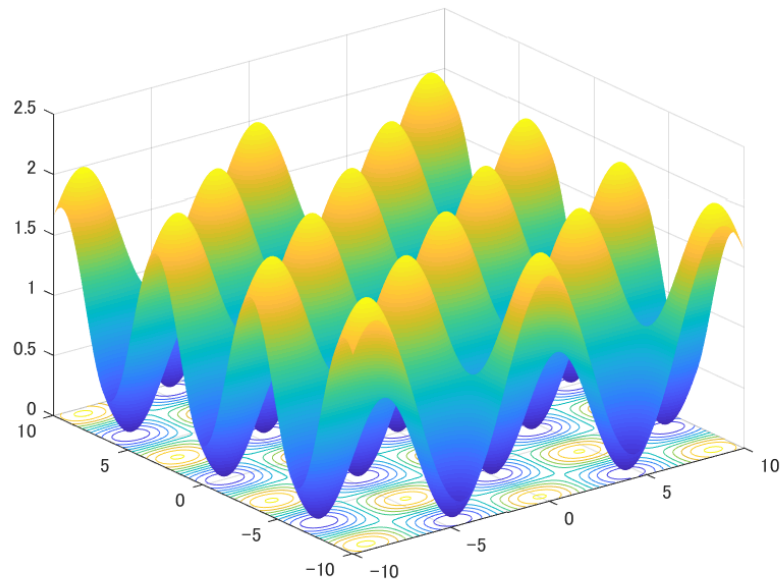
Shubert 関数の概形及び等高線を図 5.6 に示す.

$$F_6(x_i) = \prod_{i=1}^D \sum_{j=1}^5 j \cos[(j+1)x_i + j], \quad (5.6)$$

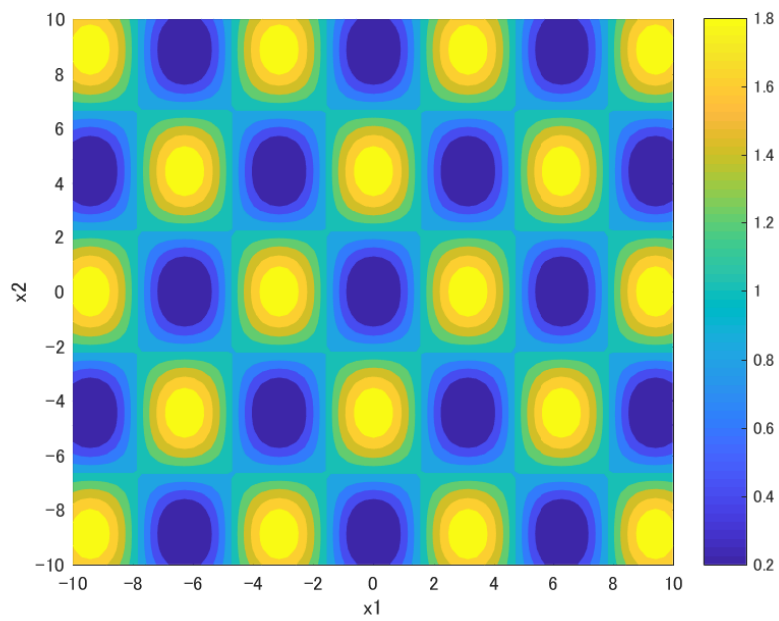
D は次元数を表し, 最適解の評価値は $F(x_*) = -187.731$ である. この関数では $D \cdot 3^D$ 個の最適解のみ存在し, 2 次元では 18 個の最適解を持つ. 探索範囲は $x_i \in [-10, 10]^D$ ($i = 1, 2, \dots, D$) である.

表 1: Benchmark Test Functions

Function	$F(x_*)$	Num of goptima	Num of loptima	D	Search Range
F_1	0	1	16	2	$x_i \in [-10, 10]$
F_2	0	1	120	2	$x_i \in [-5, 5]$
F_3	-1.0316	2	2	2	$x_1 \in [-2, 2]$ $x_2 \in [-1, 1]$
F_4	-1.8013	1	2	2	$x_i \in [0, 4]$
F_5	0	4	0	2	$x_i \in [-5, 5]$,
F_6	-187.731	18	many	2	$x_i \in [-10, 10]$,

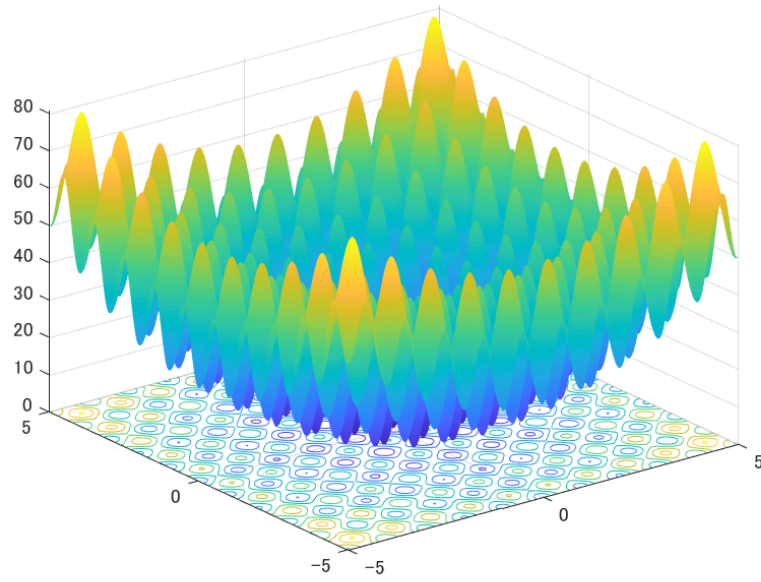


(a) Fitness Landscape

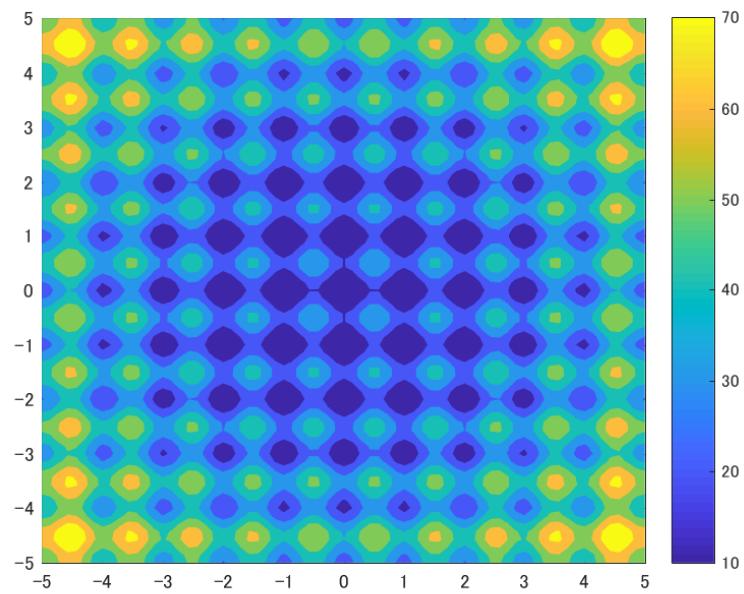


(b) Contour Plot

図 5.1: Griewank

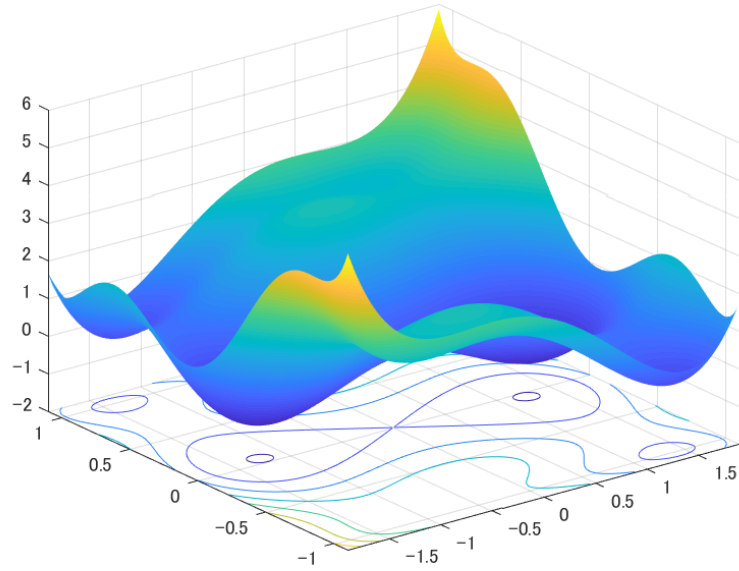


(a) Fitness Landscape

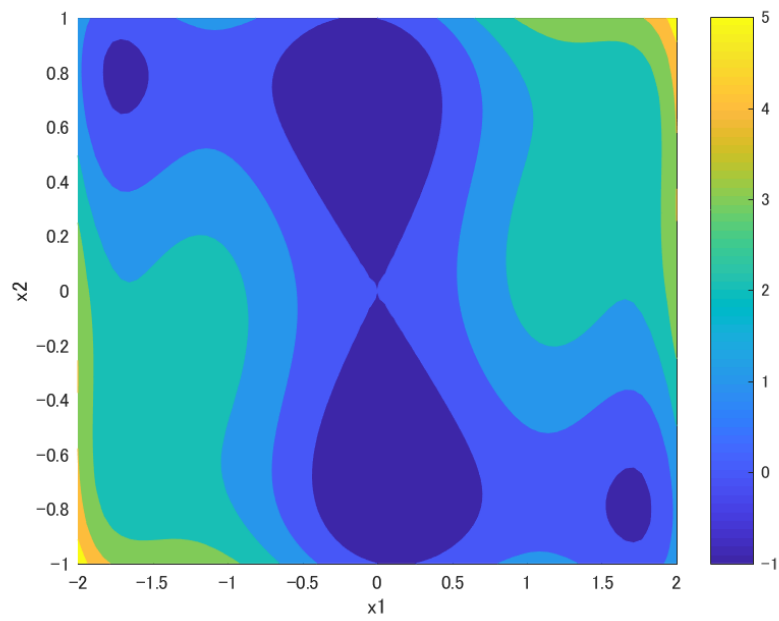


(b) Contour Plot

図 5.2: Rastrigin

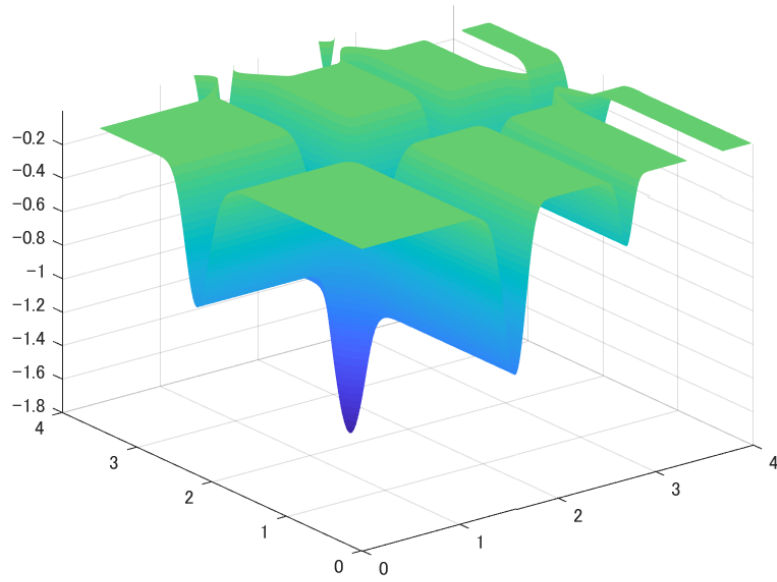


(a) Fitness Landscape

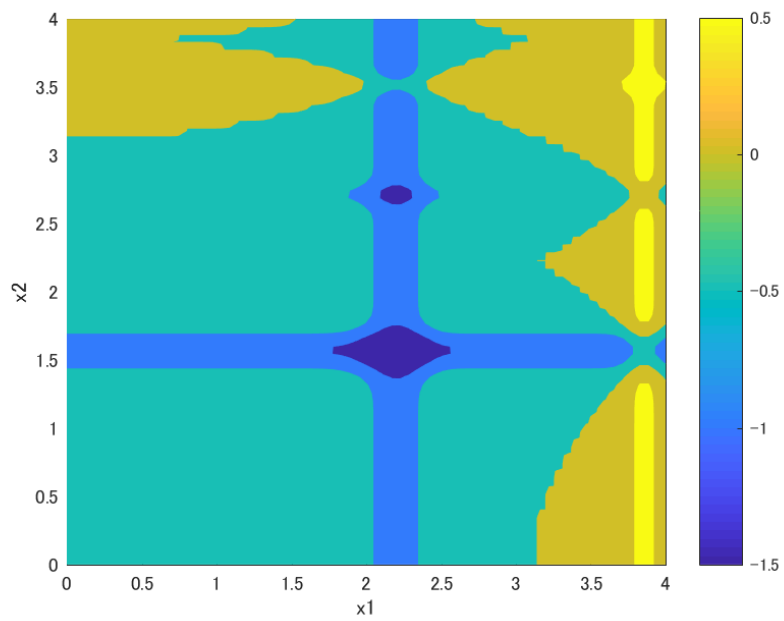


(b) Contour Plot

図 5.3: Six-Hump Camel

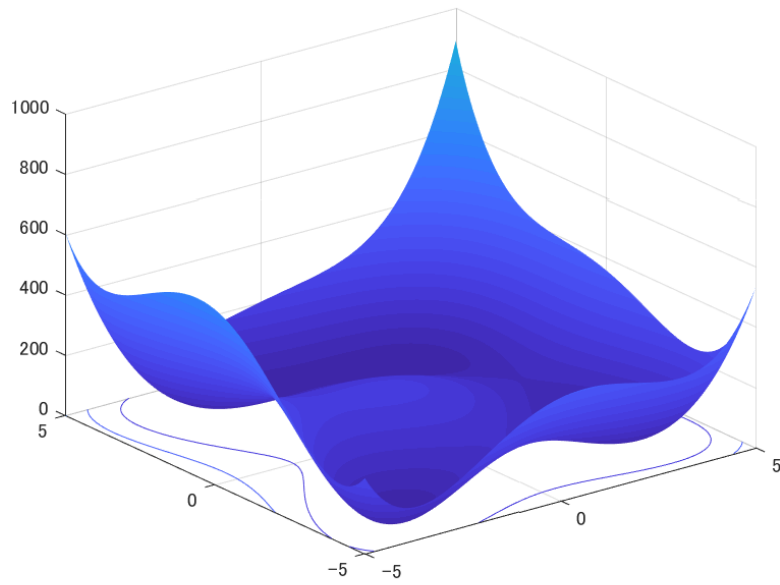


(a) Fitness Landscape

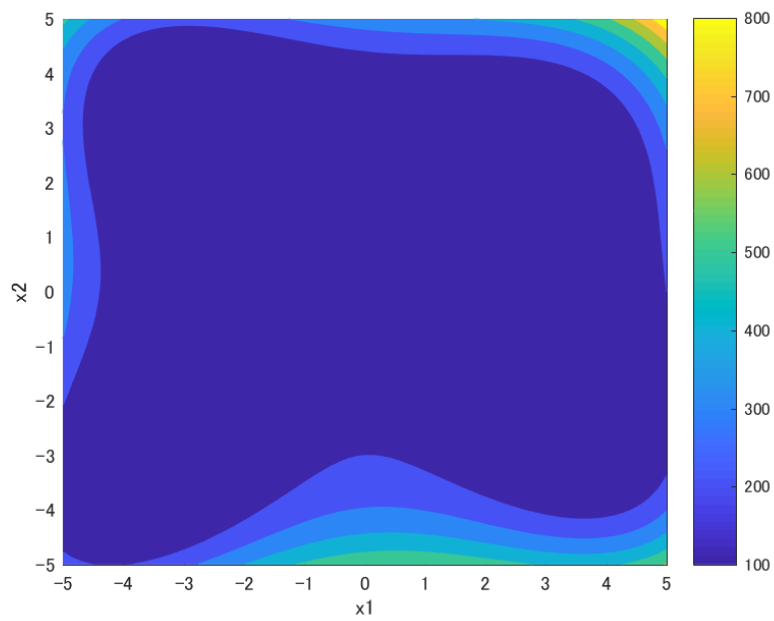


(b) Contour Plot

図 5.4: Michalewicz

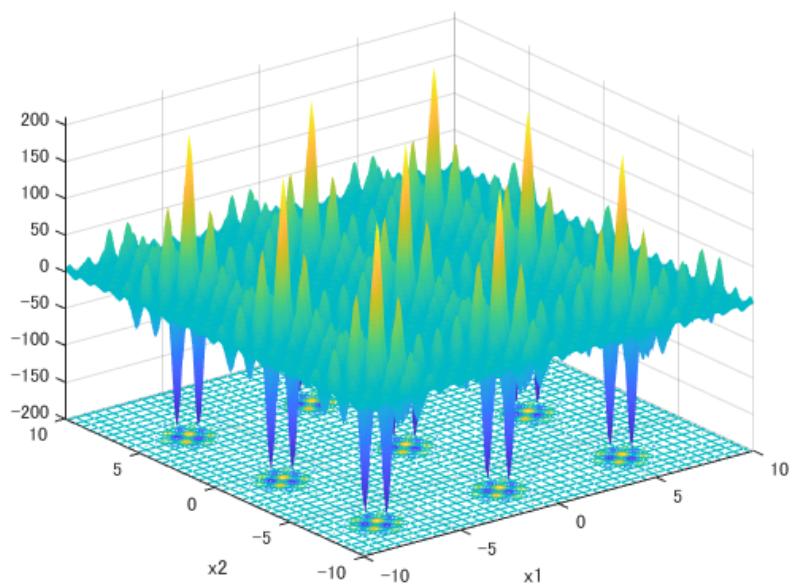


(a) Fitness Landscape

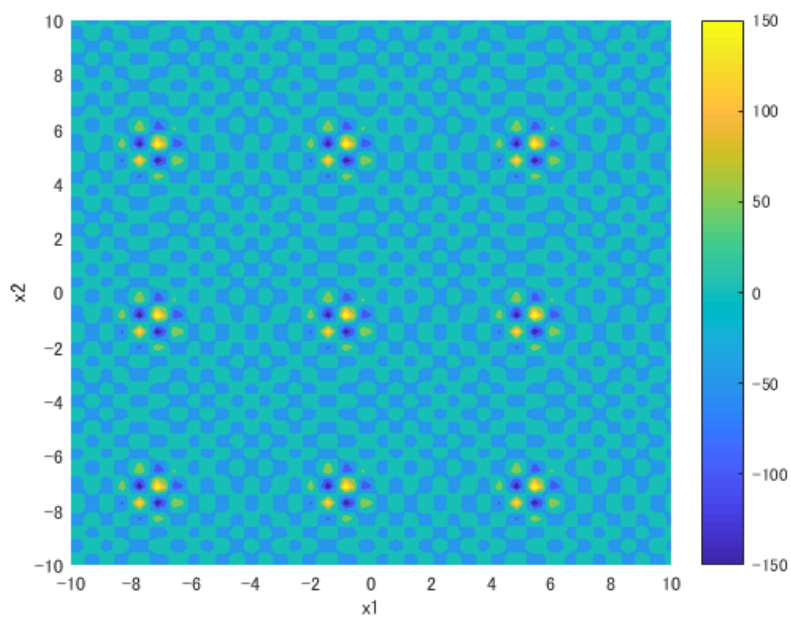


(b) Contour Plot

図 5.5: Himmelblau



(a) Fitness landscape



(b) Contour plot

図 5.6: F_6 : Shubert

5.2 最大化問題における評価関数

本実験では，複数最適解を探索する手法の性能を比較するため，CEC (*IEEE Congress on Evolutionary Computation*) 2013 Competition on Niching Methods for Multimodal Function Optimization [13] で扱われたベンチマーク関数を説明する．ベンチマーク関数の最適解の評価値 $F(x_*)$ と最適解数，探索領域を表 2 に示す．

F_1 : Five-Uneven-Peak Trap (1D)

$$F_1(x) = \begin{cases} 80(2.5 - x) & \text{for } 0 \leq x < 2.5, \\ 64(x - 2.5) & \text{for } 2.5 \leq x < 5.0, \\ 64(7.5 - x) & \text{for } 5.0 \leq x < 7.5, \\ 28(x - 7.5) & \text{for } 7.5 \leq x < 12.5, \\ 28(17.5 - x) & \text{for } 12.5 \leq x < 17.5, \\ 32(x - 17.5) & \text{for } 17.5 \leq x < 22.5, \\ 32(27.5 - x) & \text{for } 22.5 \leq x < 27.5, \\ 80(x - 27.5) & \text{for } 27.5 \leq x < 30. \end{cases} \quad (5.7)$$

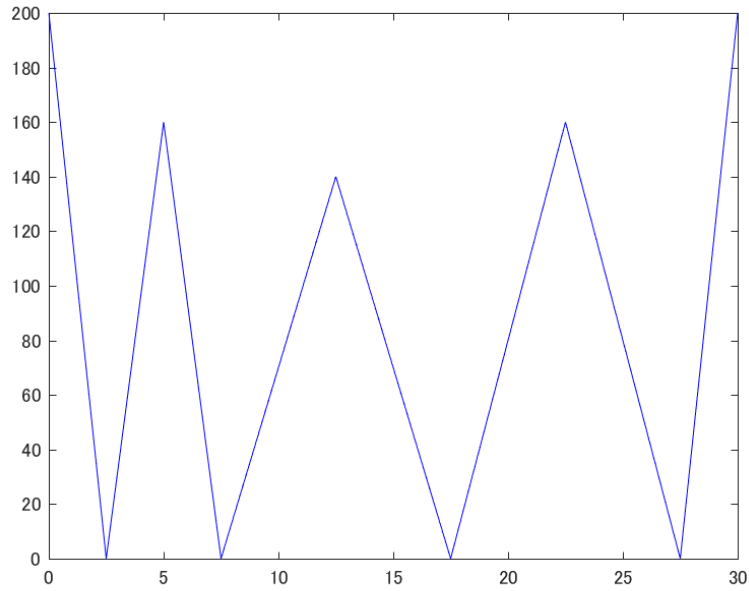


図 5.7: Five-Uneven-Peak Trap

F_2 : Equal Maxima (1D)

$$F_2(x) = \sin^6(5\pi x). \quad (5.8)$$

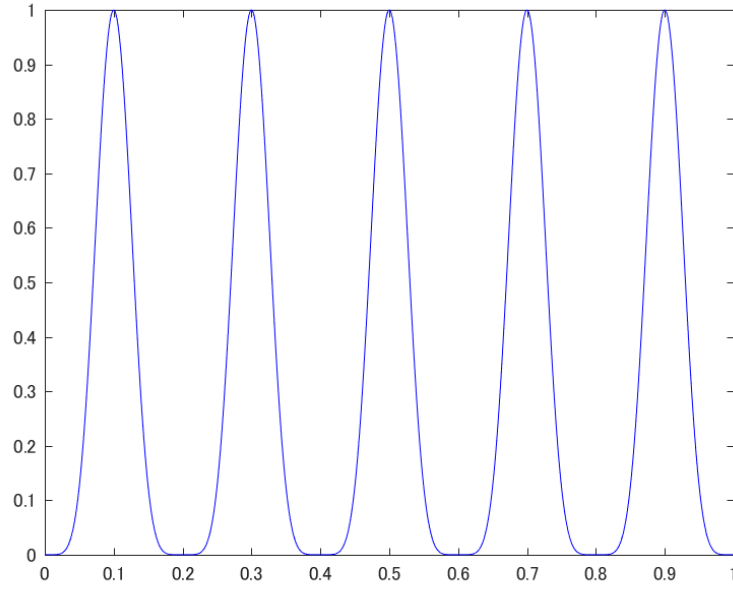


図 5.8: Equal Maxima

F_3 : Uneven Decreasing Maxima (1D)

$$F_3(x) = \exp(-2\log(2)(\frac{x-0.08}{0.854})^2) \sin^6(5\pi(x^{\frac{3}{4}} - 0.05)). \quad (5.9)$$

F_4 : Himmelblau (2D)

$$F_4(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2. \quad (5.10)$$

F_5 : Six-Hump Camel Back (2D)

$$F_5(x, y) = -4[(4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (4y^2 - 4)y^2]. \quad (5.11)$$

F_6 : Shubert (2D)

$$F_6(x) = -\prod_{i=1}^D \sum_{j=1}^5 j \cos[(j+1)x_i + j]. \quad (5.12)$$

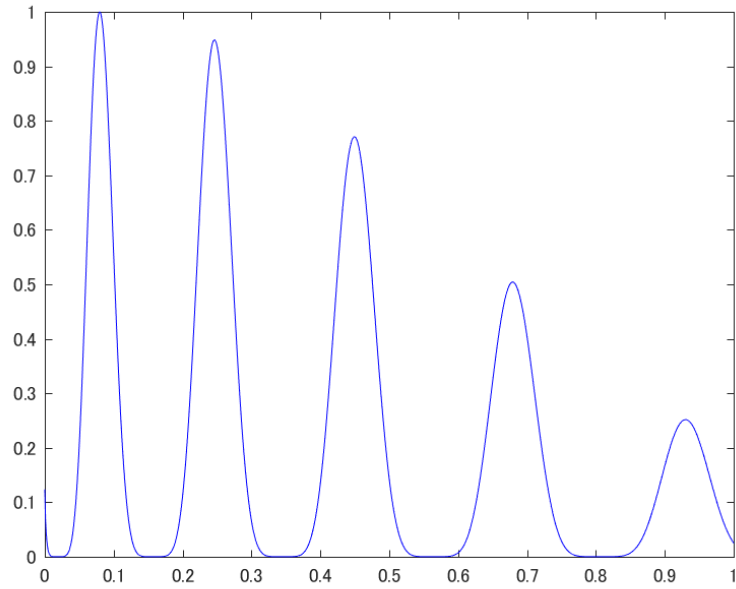


図 5.9: Uneven Decreasing Maxima

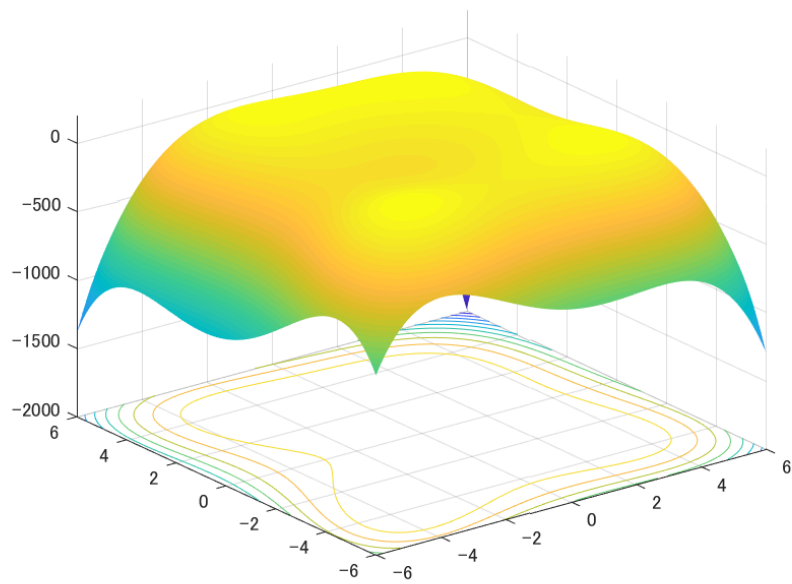


図 5.10: 2D Himmelblau

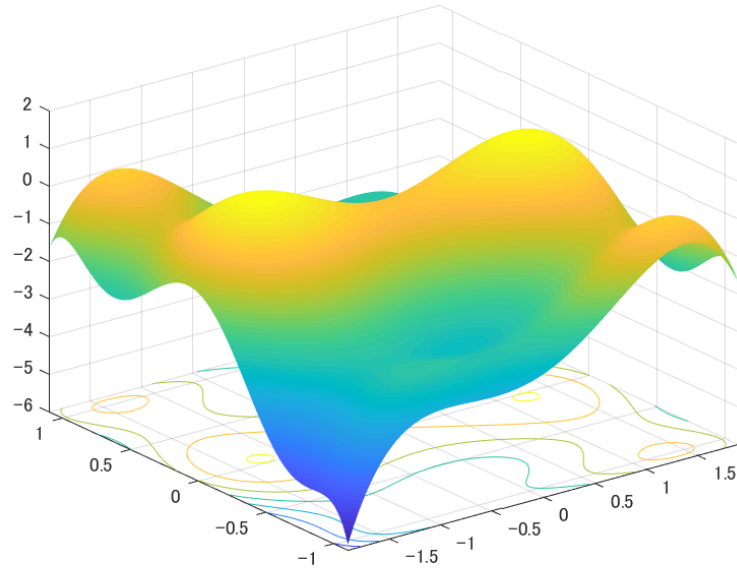


図 5.11: 2D Six-Hump Camel Back

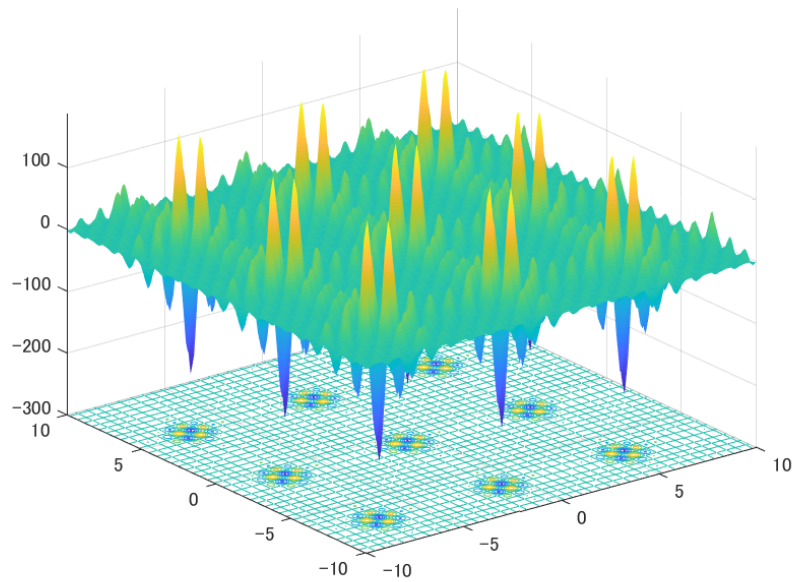


図 5.12: 2D Shubert

表 2: Benchmark Test Functions

Function	$F(x_*)$	Num of goptima	ρ	D	Search Range
F_1	200.0	2	0.01	1	$x \in [0, 30]$
F_2	1.0	5	0.01	1	$x \in [0, 1]$
F_3	1.0	1	0.01	1	$x \in [0, 1]$
F_4	200.0	4	0.01	2	$x, y \in [-6, 6]$
F_5	1.03163	2	0.5	2	$x \in [-1.9, 1.9],$ $y \in [-1.1, 1.1]$
F_6	186.731	18	0.5	2	$x_i \in [-10, 10]$

6 Novelty Search-based Bat Algorithm (NSBA)

本章では多峰性最適化に対し、大域探索と局所探索を自動で切り替えることに優れた Bat Algorithm (BA) と、未探索領域に解を生成する Novelty Search を組み合わせた NSBA [11] を説明する。

6.1 Novelty Search

Novelty Search [7] は未探索領域に新たに解を生成することを目的とした手法である。個体間距離の算出式は次式で表される。

$$\rho(x) = \frac{1}{k} \sum_{i=1}^k \text{dist}(x, \mu_i) \quad (6.1)$$

この時、 $\rho(x)$ は個体 x における密度を表しており、 k は個体 x の近傍数、 dist は個体 x と μ_i の距離を表す。図 6.1 は近傍数が 3 の時の解の生成を表す。

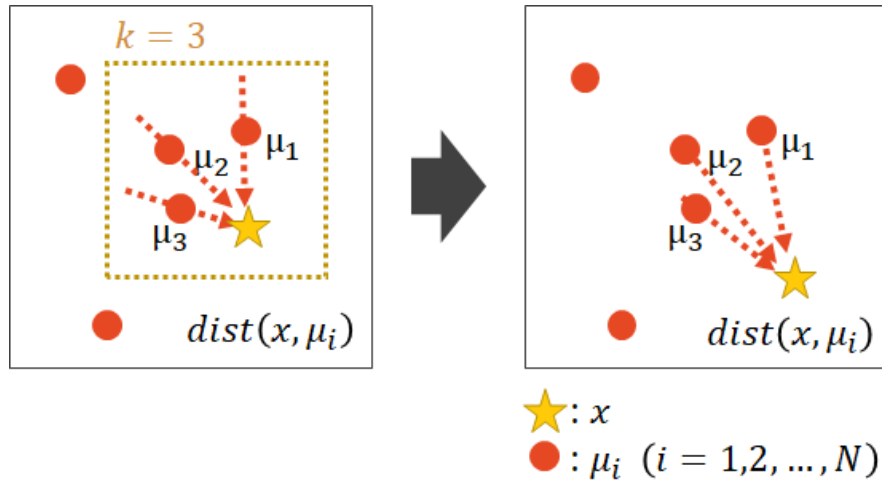


図 6.1: 解候補の生成

6.2 メカニズム

NSBA は BA に Novelty Search を組み込んだ複数解探索手法である。(6.1) 式を以下のベクトル式に変更することで、密集している個体が疎な方向へ新たに解候補を生成する。

$$d_i^t = \frac{1}{N} \sum_{j=1}^N \frac{(x_{pbest} - x_j^t)}{|x_{pbest} - x_j^t|^2} \quad (6.2)$$

$$v_i^{t+1} = v_i^t + d_i^t * f_i \quad (6.3)$$

この時、 x_{pbest} は i 番目の最良個体を表し、このベクトル式から速度 v_i を更新する。

局所探索では、最良個体 x_{pbest} 付近に新たに解候補を次式で生成される。

$$\mathbf{x}_{loc} = \mathbf{x}_{pbest} + \epsilon A^t, \quad (6.4)$$

ϵ は $[-1, 1]$ 区間における D 次元の一様乱数を表す。ランダム探索では、探索空間内に新たに解候補を生成する。

$$\mathbf{x}_{rnd} = x_{lb} + (x_{ub} - x_{lb}) * rand(1, D) \quad (6.5)$$

6.3 アルゴリズム

アルゴリズムの疑似コードを以下の Algorithm 10 に記す。

Algorithm 10 Novelty Search-based Bat Algorithm

Input: Objective Function $F(x)$

Initialize Population $x_i (i = 1, 2, \dots, N)$ and v_i

Define frequency f_i at location x_i [Eq.(2.9)]

Initialize pulse rates r_i , and loudness A_i

while ($t < \text{Max number of iterations}$) **do**

for $i=1$ to N **do**

6: Generate a new solution x_i and update velocity v_i [Eqs.(??)(6.2)(6.3)]

if ($rand > r_i$) **then**

 Generate a new solution x_{loc} around the solution x_i [Eq.(6.4)]

end if

 Generate a new solution x_{rnd} randomly (or without x_{rnd}) [Eq. (6.5)]

if ($rand < A_i$ & $\min(F(x_i), F(x_{new}), F(x_{rnd})) < F(x_{i*})$) **then**

12: Accept the new solution, and update pulse rate r_i

 & loudness A_i [Eqs. (2.14)(2.15)]

end if

end for

 Evaluate the all bats and select a best solution x_{i*} in the current solutions

end while

6.4 実験

本実験では従来の BA と NSBA の性能を比較するため、代表的な以下の多峰性関数を用いる。

6.4.1 評価関数

本実験では 5.1 節で説明した F_1 関数 (Griewank) と F_2 関数 (Rastrigin) を用いる。

6.4.2 評価基準

本実験では Peak Ratio (PR) [12] を採用し、最適解及び局所解の発見率を次式で求める。

$$PR = \frac{\sum_{run=1}^{MR} FPs}{TP * MR} \quad (6.6)$$

ここで、 TP は評価関数の全最適解と局所解数 (Total Peak) を表し、 MR は実験回数 (Max Run) を表す。 FPs は発見した解の数 (Found Peaks) を表す。 解発見の定義はピークとその最近傍個体とのユークリッド距離が 0.1 未満であった時、その解を発見したとする。

6.4.3 実験設定

本実験では、 F_1 関数では個体数 $N = 50, 100$, F_2 関数では $N = 100, 150$ とした。 2 つの関数において、 $f_{max} = 1, f_{min} = 0$, ラウドネス $A^0 = 1$, パルスレート $r^0 \in rand[0, 1]$ とし、 $\alpha = \gamma = 0.9$ と設定した。 世代数は 10000, 実験回数を 30 回とした。

6.4.4 実験結果

- PR と発見した解数

表 3 は BA と NSBA における、発見した解の数 (FPs) とその発見率 (PR) の 30 試行回数による平均値と標準偏差を示す。 表 3 より 2 つの関数に対し、NSBA は BA よりも FPs 及び PR 値が高いことから探索性能が良好である。 図 6.2, 6.3 は最終世代における解の分布を表し、白い丸が解の位置である。 図より BA は一つの最適解に収束する傾向にあるが、一方で NSBA は最適解とその周辺にある複数の局所解に留まっていることが分かった。

- 収束速度

図 6.4 は NSBA と BA の世代数による PR 値の推移を表しており、縦軸は PR 値、横軸は世代数を示す。 黒い実線は BA, 黒い点線は NSBA の推移を示している。 図 4(a) から 4(d) で BA は最終世代において、PR 値が 0% 付近で収束しているのに対し、NSBA は F_1 関数では 70% まで PR 値が上昇し、1000 世代以降は徐々に低下する傾向にある。 F_2 関数では NSBA の PR 値が 20% から 10% まで減少し、3000 世代以降は停滞する傾向が見られた。

6.5 考察

6.5.1 個体数による影響

個体数の変化による影響があるか調査するため、BA と NSBA にて個体数を変えて実験を行った。 表 3 より、BA の PR 値は F_1, F_2 関数のいずれにおいても個体数による変化がなかった。 一方、NSBA は個体数が増加すると 2 つの関数において PR 値が増加した。 NSBA は F_1 関数において、40% から 42.75% と探索率が上昇し、 F_2 関数においても 6.56%

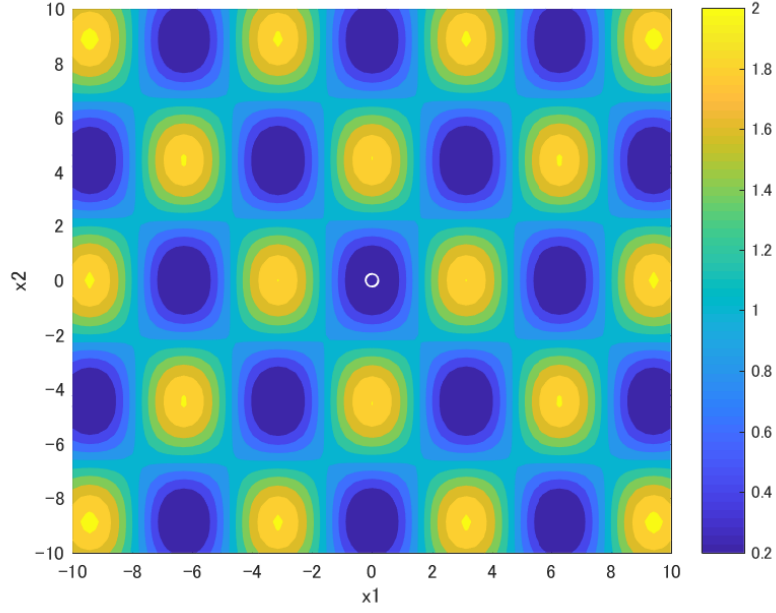
表 3: Found Peaks and Peak Ratio of BA and NSBA

Function	BA			NSBA		
	Mean	SD	PR	Mean	SD	PR
F_1 ($N = 50$)	1.0	0	5.89 %	6.8	0.7024	40.0 %
F_1 ($N = 100$)	1.0	0	5.89 %	7.267	0.5735	42.75 %
F_2 ($N = 100$)	1.0	0	0.87 %	7.9333	0.8929	6.56 %
F_2 ($N = 150$)	1.0	0	0.87 %	8.0667	0.7717	6.67 %

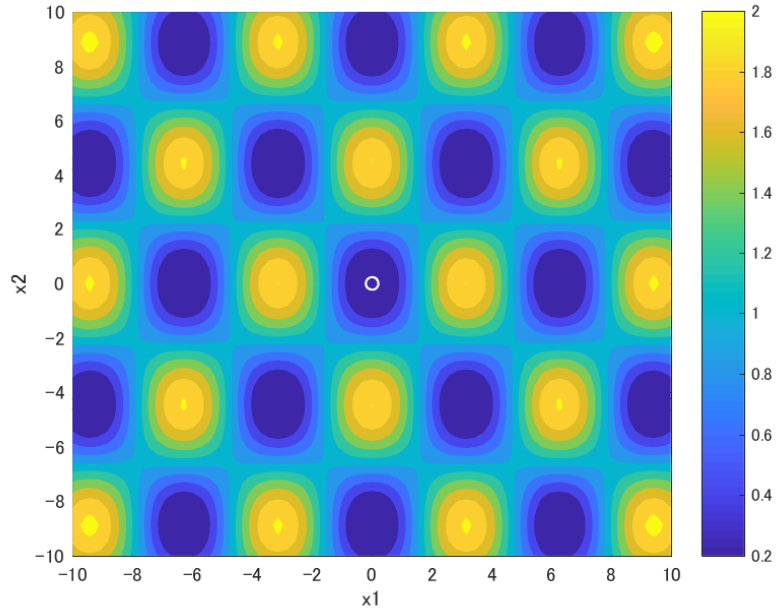
から 6.67%と僅かに上昇した．このことから個体数を増やすことで探索性能が向上したと考えられる．また，図 6.4 から個体数の増加により，世代数が増す中で PR 値を維持していることが分かる． F_1 関数で NSBA($N = 50$) の PR 値は 70%から 30%へと低下しているが，NSBA($N = 100$) では 6000 世代付近から 50%を維持していることが分かる．

6.5.2 解の分布

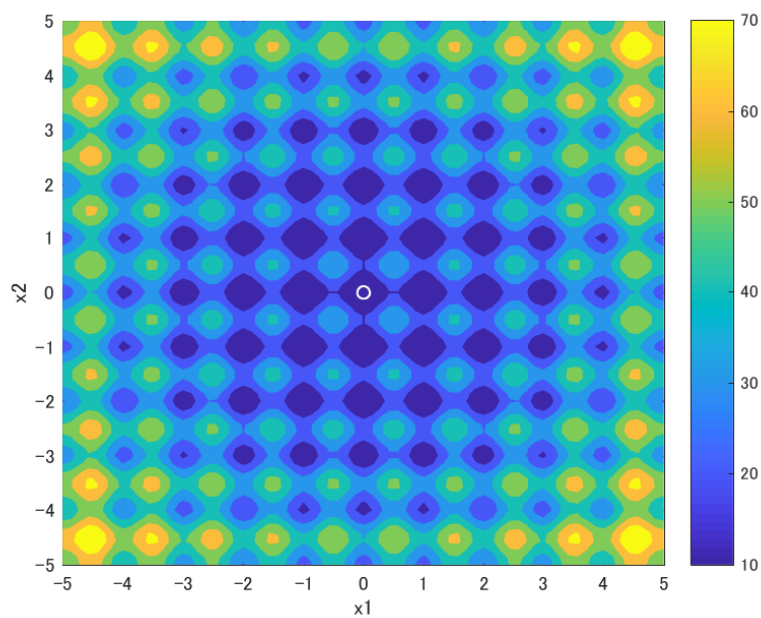
図 6.4 より PR 値を維持できない原因を分析するため，ここでは PR 値が最も高かった 1000 世代目の解分布に着目し，その時の分布図を図 6.5 に示す．



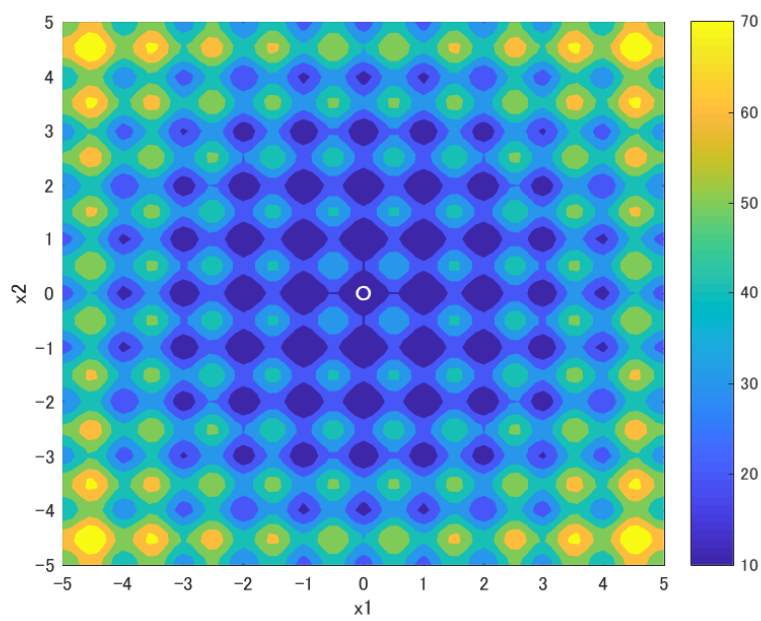
(a) $F_1 : (N = 50)$



(b) $F_1 : (N = 100)$

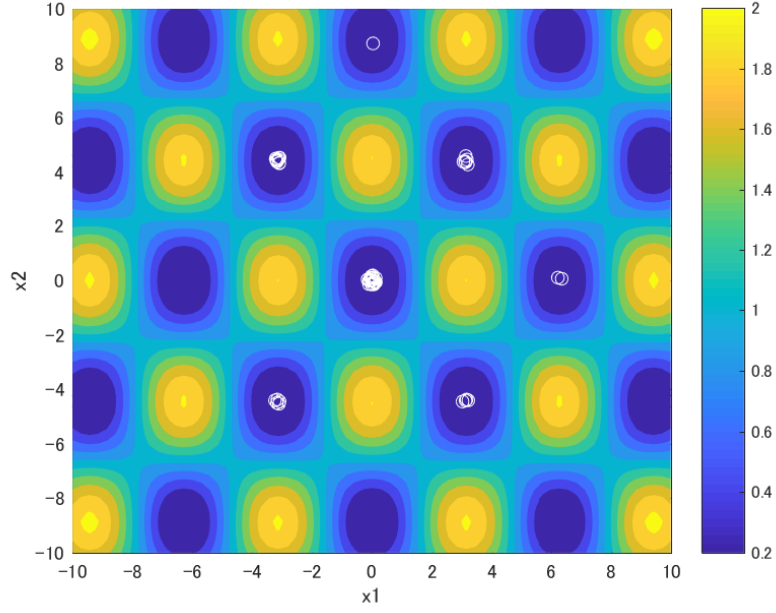


(c) $F_2 : (N = 100)$

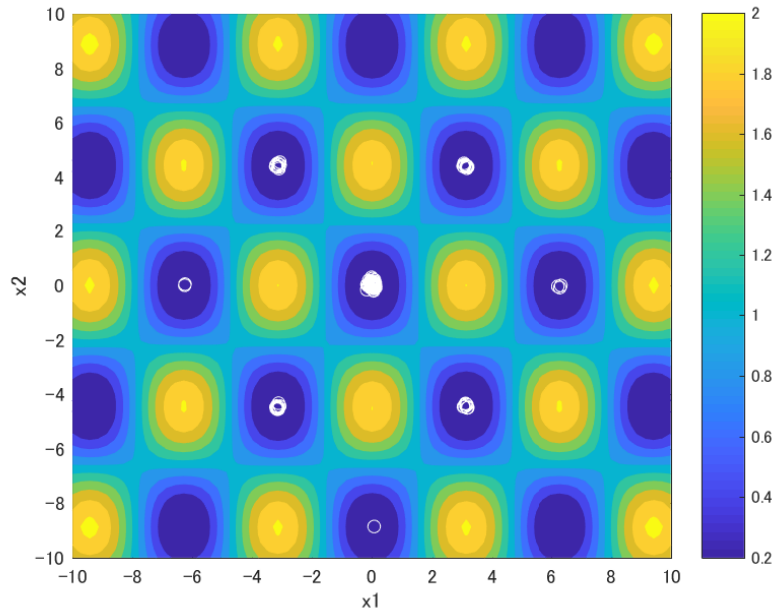


(d) $F_2 : (N = 150)$

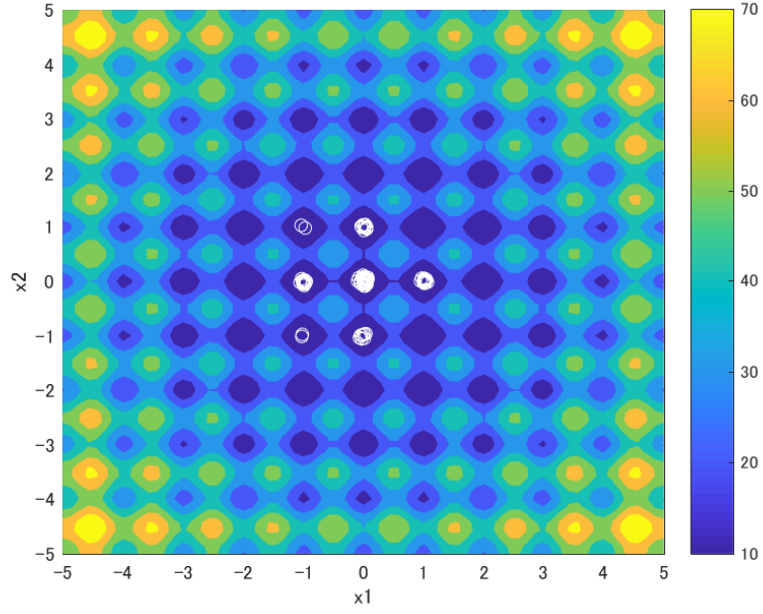
図 6.2: BA



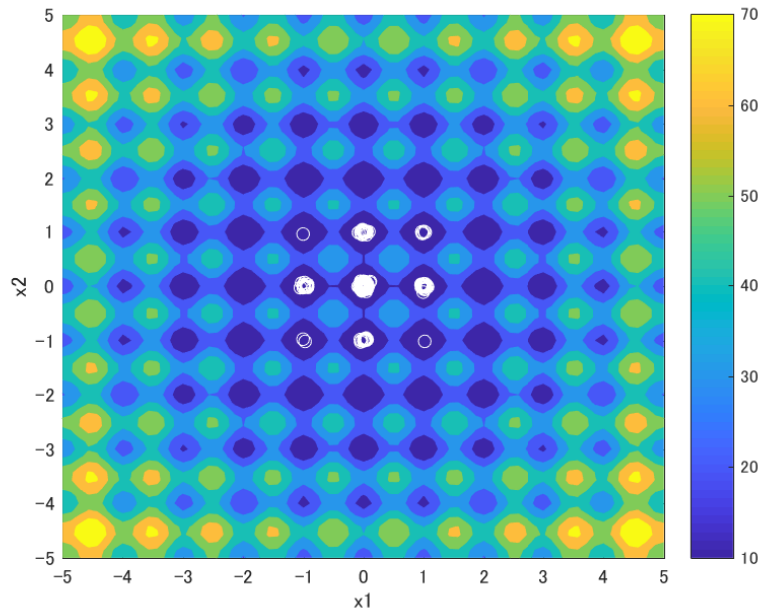
(a) $F_1 : (N = 50)$



(b) $F_1 : (N = 100)$

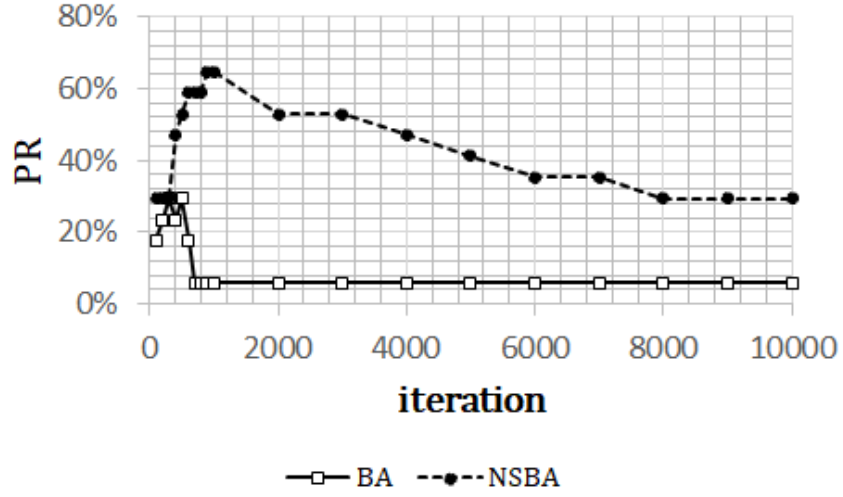


(c) $F_2 : (N = 100)$

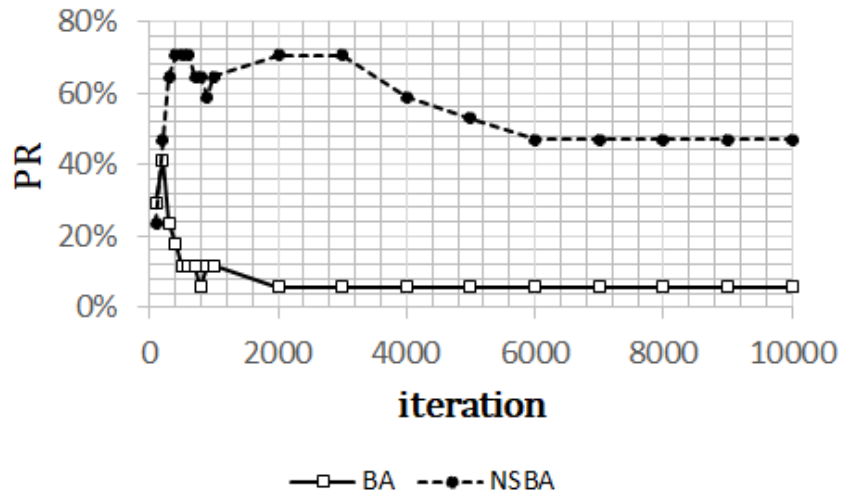


(d) $F_2 : (N = 150)$

図 6.3: NSBA



(a) $F_1 : (N = 50)$



(b) $F_1 : (N = 100)$

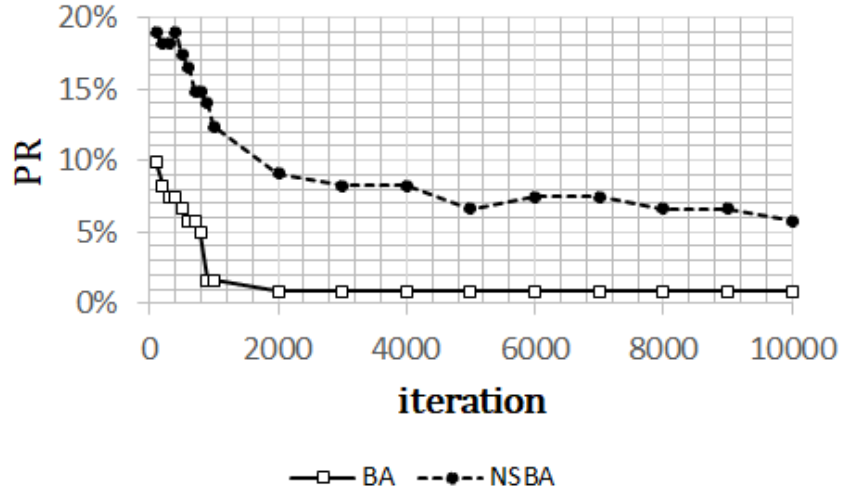
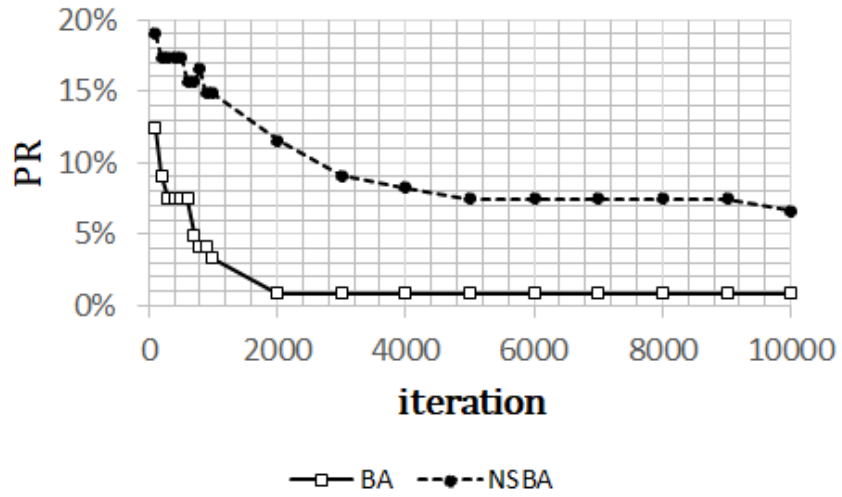
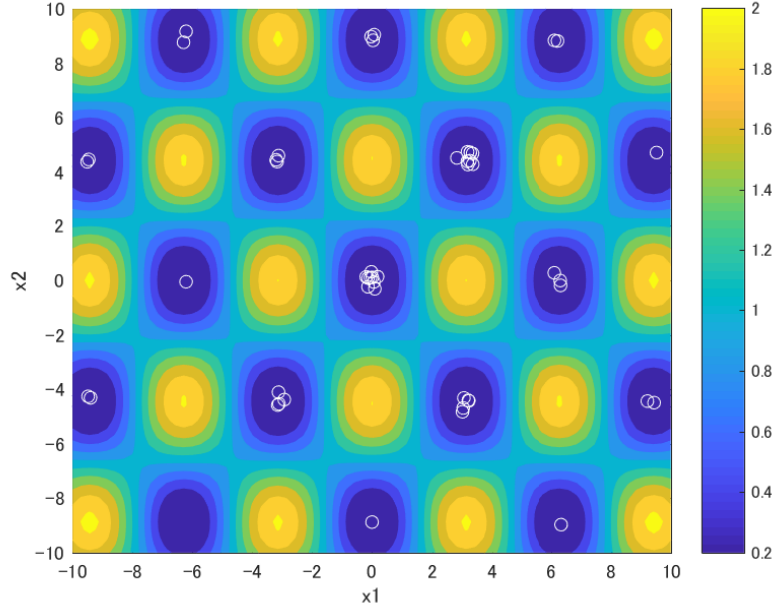
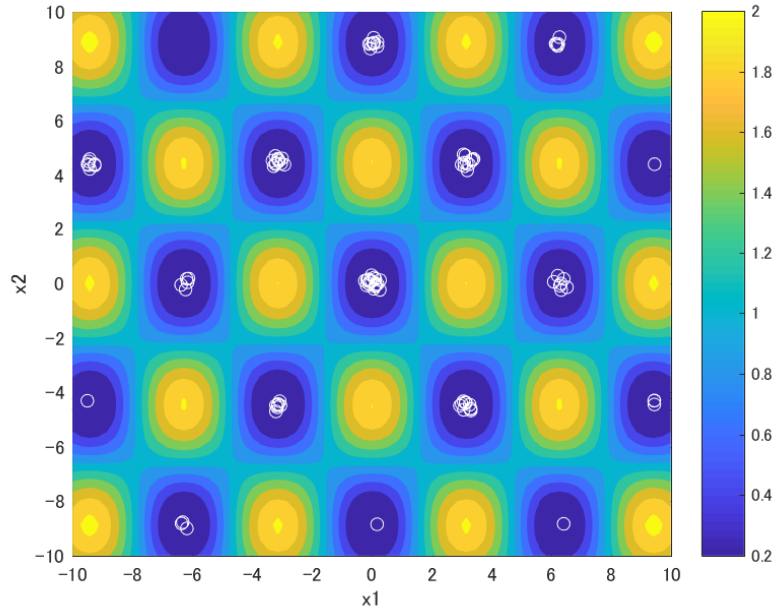
(c) $F_2 : (N = 100)$ (d) $F_2 : (N = 150)$

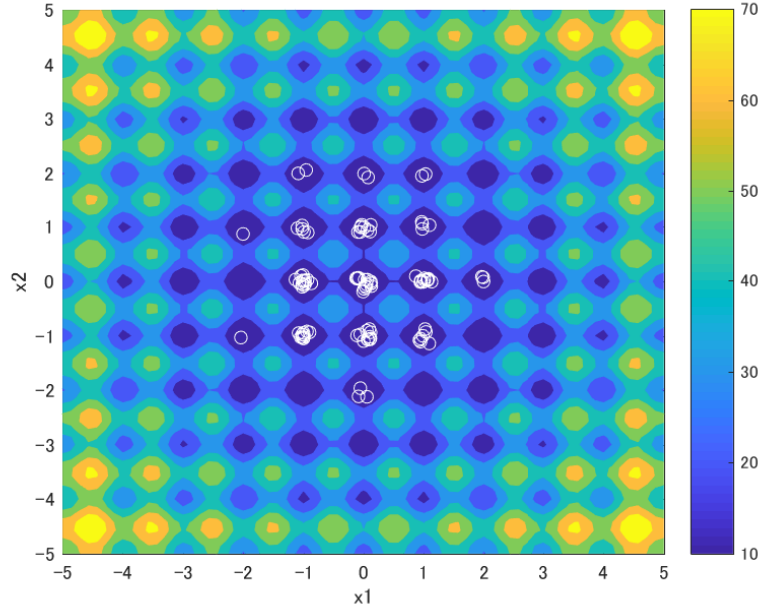
図 6.4: Convergence Speed of Peak Ratio implemented by BA and NSBA



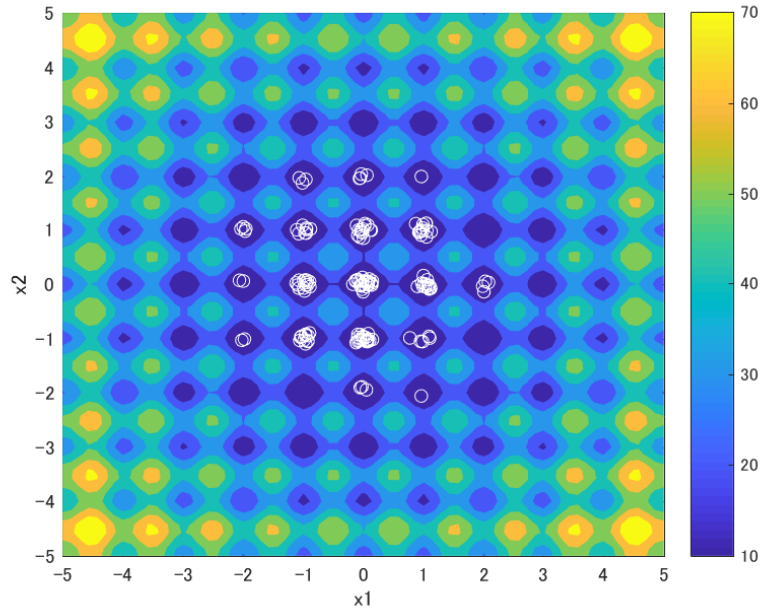
(a) $F_1 : (N = 50)$



(b) $F_1 : (N = 100)$



(c) $F_2 : (N = 100)$



(d) $F_2 : (N = 150)$

図 6.5: Distribution of Solutions (NSBA)

7 Niche Radius-based Bat Algorithm (NRBA)

探索空間の分割方法の一つとして Niche Radius が挙げられる．Niche Radius は探索空間のスケールと探索する解の数を元に個体の探索範囲を決定することのできる手法である．これにより，各個体が同じ解に留まることなく分散させ，従来の BA に以下 3 つの変更点を加えることで，最適解だけでなく局所解も同時に探索可能な NRBA [16] を説明する．

7.1 メカニズム

- BA からの変更点 1: 大域探索

ここでは Niche Radius を使用し，従来である BA の解候補の生成式 (??),(??) を次式のように変更を加えた．

$$v_i^{t+1} = v_i^t + (x_i^t - x_{NR*}) * f_i \quad (7.1)$$

$$x_i^{t+1} = \begin{cases} x_i^t + v_i^{t+1} & (\text{if } d_i^t < NR) \\ x_i^t & (\text{otherwise}) \end{cases} \quad (7.2)$$

個体移動時のイメージ図を図 7.1 に表す．各個体は NR を半径とした円の探索領域が決まっており，個体間距離 d_i が NR より小さい場合において，式 (7.1) にて NR 内の最良解 x_{NR*} を中心とした円から離れる方向へ個体 x_i^t が速度 v_i で移動する．また NR 内に他の個体が存在しない，あるいは最良解 x_{NR*} は移動をせず，その場所に留まる．この変更により，個体が同じ探索領域内に留まらず分散化をはかる．

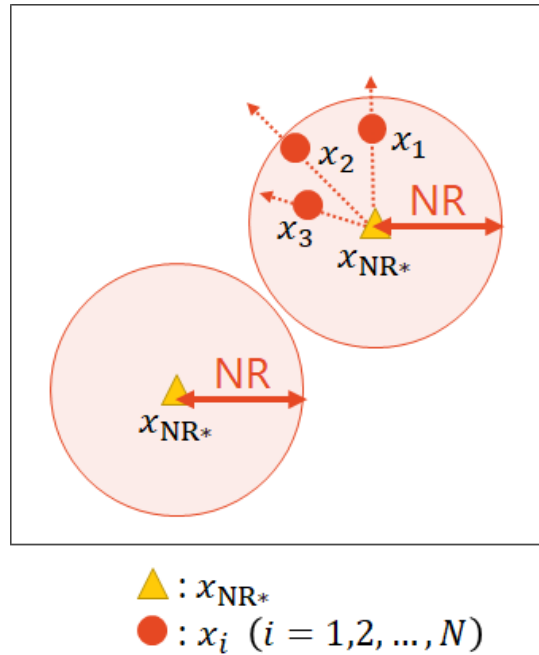


図 7.1: 解候補の生成

- BA からの変更点 2: 局所探索

次に局所探索性能を上げるため、各個体が持つ Niche Radius 内の最良解 x_{NR*} の周辺に新しい解候補 x_{loc} を生成するよう変更した。生成式は次の通りである。

$$x_{loc} = x_{NR*} + \epsilon A_i^t \quad (7.3)$$

ϵ は $1 \times D$ 次元の配列で $[-NR, NR]$ 区間のランダムな値が割り当てられる。この変更により、個体を局所解へ収束するよう促す。

- BA からの変更点 3: ランダム探索

ランダム探索では各個体の持つ NR 内にランダムで解候補を以下の式のように生成する。

$$x_{rnd} = x_i^t + rand(1, D, [-NR, NR]) \quad (7.4)$$

$[-NR, NR]$ 区間内の $1 \times D$ 次元の配列により現在位置 x_i^t 周辺に解候補を生成する。この変更では各個体を最適解あるいは局所解近辺へ移動させることで同じ場所に留まることを避ける。

7.2 アルゴリズム

提案手法の NRBA のアルゴリズムの疑似コードを Algorithm 11 に記す。

Algorithm 11 Niche Radius-based Bat Algorithm

Input: 評価関数 $F(x)$ の設定

各個体 $x_i (i = 1, 2, \dots, N)$ と速度 v_i の初期化

周波数 f_i の定義 [eq.(2.9)]

パルスレート r_i とラウドネス A_i の初期化

Niche Radius の算出 [eqs.(3.2), (3.1)]

while ($t < \text{Max Iteration}$) **do**

for $i=1$ to N **do**

7: **if** ($d_i < NR$) **then**

 大域探索: 新しい解候補 x_i^{t+1} の生成と速度 v_i の更新 [eqs.(7.1),(7.2)]

end if

if ($rand > r_i$) **then**

 局所探索: 生成した解候補 x_i^{t+1} 近辺に新しい解 x_{loc} を生成 [eq.(7.3)]

end if

 ランダム探索: パーソナルベストの持つ NR 内に解候補の生成 [eq.(7.4)]

14: **if** ($rand < A_i \ \& \ F(x_i), F(x_{loc}), F(x_{rnd}) < F(x_{i*})$) **then**

 新しい解の評価と更新

 パルスレート r_i の増加とラウドネス A_i の減少 [eqs.(2.14),(2.15)]

end if

end for

$t=t+1$

end while

7.3 実験

最適解と局所解の数が異なる評価関数において、各手法の探索性能にどのような影響があるか調査する。次の4つのパターンの評価関数を用意し、従来手法であるBA、前章で提案したNSBAと比較することで提案手法の探索性能の有効性を検証する。一つの最適解に対して複数の局所解を持つ; 複数の最適解に対して同じ数の局所解を持つ; 一つの最適解の数に対して一つの局所解を持つ; 最適解のみ複数持つ。これらのパターンに適した多峰性関数を用いて実験を行う。

7.3.1 評価関数

本実験で使用するベンチマーク関数は、1節で説明した F_1 : Griewank, F_3 : Six-Hump Camel, F_4 : Michalewicz, F_5 : Himmelblau の4つを用いる。

7.3.2 評価基準

本実験において、Congress on Evolutionary Computation (CEC2013) [13] のコンペティションで用いられた評価尺度である Peak Ratio (PR) [12] により評価する。評価式は以下のように設定した。

$$PR = \frac{\sum_{run=1}^{MR} FPs}{TP * MR} \quad (7.5)$$

Max Run (MR) は実験回数を表し、Found Peaks (FPs) は発見した解の数を、Total Peak (TP) は探索領域内の全最適解及び局所解数を表す。また最適解及び局所解の位置座標と最近傍個体とのユークリッド距離が0.1未満であれば、その解を発見したと定義する。

7.3.3 実験設定

個体数 $N = 50$ とし、各個体のパラメータ $A_i^0 = 1, r_i^0 \in [0, 1], f_{max} = 1, f_{min} = 0, \alpha = \gamma = 0.9$ と設定した。また Table 1 より、探索領域の上限 x_{ub} と下限 x_{lb} 、各評価関数の解の総数 q として使用した。また次元数 $D = 2$ 、世代数を 10000、実験回数 $MR = 30$ とした。

7.3.4 実験結果

各評価関数について、BA と NSBA、提案手法の NRBA における PR 値を表4に示す。表中の Mean (平均値) と SD (標準偏差) は実験回数 30 回での最終世代における発見した最適解及び局所解数を PR 値で表した結果である。各手法の最終世代での個体の分布を図 7.2, 7.3 で表す。またグラフ中の赤い丸は個体の分布を示す。従来手法である BA は全個体の最良解へ向かって進んでしまうため、全ての評価関数において、最適解あるいは評価値の高い局所解に収束した。しかし図 2(d) については局所解は存在しないが、最終世代では一つの最適解へ収束する結果となった。一方で提案した NRBA では図 7.3 から全評価関数において、全ての解に個体が到達しているように分布しているが、表4から実際

に最適解や局所解の位置まで到達していないケースが多く見られた。また最適解や局所解に到達していない個体については用いた評価関数によって分布に偏りがあった。

表 4: The value of PR (averaged over 30 runs)

	BA	NSBA	NRBA
Function	Mean \pm SD	Mean \pm SD	Mean \pm SD
F_1	0.0588 \pm 0	0.3760 \pm 0.0413	0.6922 \pm 0.0981
F_3	0.4917 \pm 0.0449	0.5 \pm 0	0.9917 \pm 0.0449
F_4	0.5 \pm 0	0.5 \pm 0	0.7 \pm 0.2450
F_5	0.2417 \pm 0.1367	1 \pm 0	0.8583 \pm 0.1239

7.4 考察

全ての評価関数において、従来の BA より提案の NRBA のほうが探索した解の数および発見率が高かったことから最適解への収束を防ぎ、複数の局所解へ分散させることができた。また提案手法の探索性能の有効性を確かめるため、評価尺度による解の発見数及び発見率の結果と、最終世代の解の分布という観点から分析を行った。

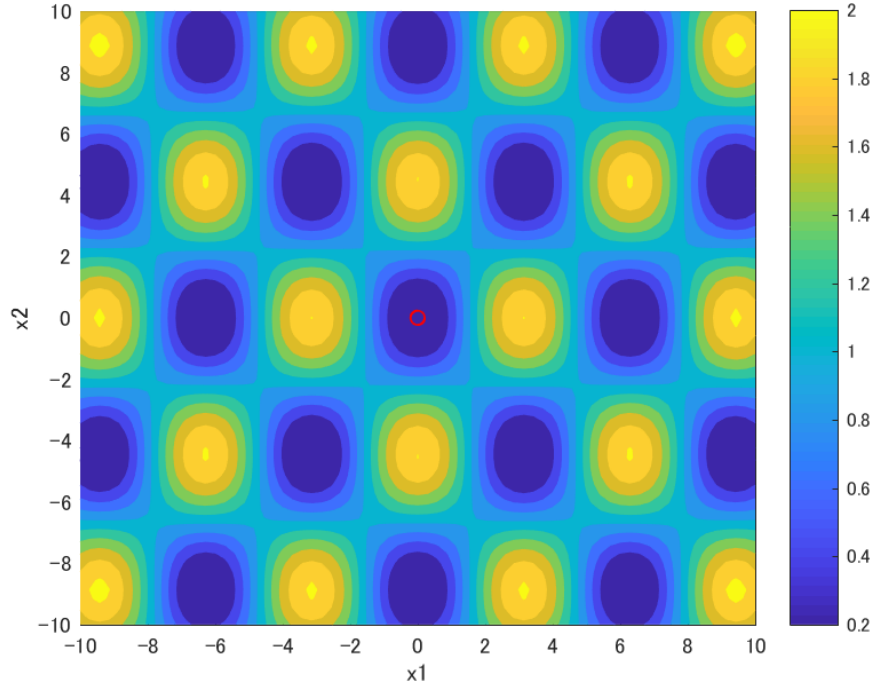
7.4.1 解の発見数

表 4 から従来手法の BA は全ての評価関数において一つの解へ収束する傾向が強かったが、 F_2 関数では 2 つの解を探索することができた。これは従来手法のアルゴリズムの解候補の生成において、全個体の最良解へ向かって探索をしていることが原因であると考えられる。提案手法の NRBA は全ての評価関数に対して、従来手法よりも発見した解の数が多く、最適解と複数の局所解を探索することができた。評価関数の中でも F_2 関数、次いで F_4 関数の PR の値が高いことから最適解のみ存在する場合と、最適解と局所解の評価値の差が小さい場合において提案手法の探索が有効に働いたと考えられる。また局所解を多く含む F_1 関数や局所解の範囲が非常に狭い F_3 関数では探索性能が落ちたことから今後の課題として手法を改良する必要がある。

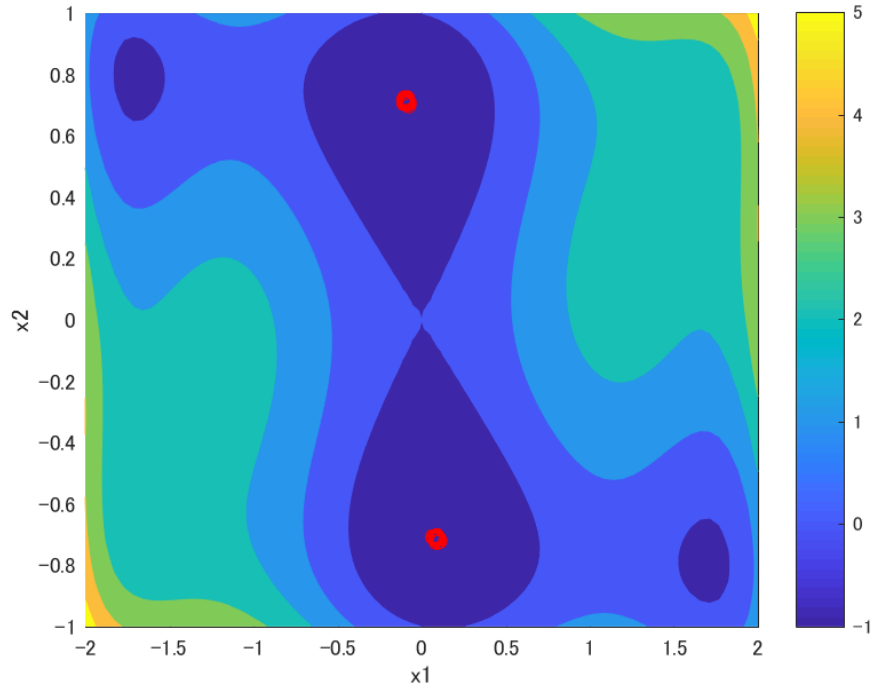
7.4.2 最終世代における解の分布

図 7.2 から従来の BA は、最適解への収束が非常に強かった。しかし、図 2(b) では 2 つの最適解に収束していたものの、図 2(d) では複数の最適解へ収束することなく一つの最適解へ収束した。これは、 F_4 関数のほうが探索領域が広く、評価値の数値の範囲も広いことから各個体の持つ評価値にバラつきが出やすく、一つの最適解へ収束したと考えられる。図 7.3 より提案手法の NRBA は、図 3(a) では色濃度の濃い領域に各個体を分散させることができたが、図 3(b), 3(c), 3(d) では最適解や局所解でない場所に分布する個体も多く見られた。このことから従来手法から変更した、Niche Radius 内の最良解より個体が遠ざ

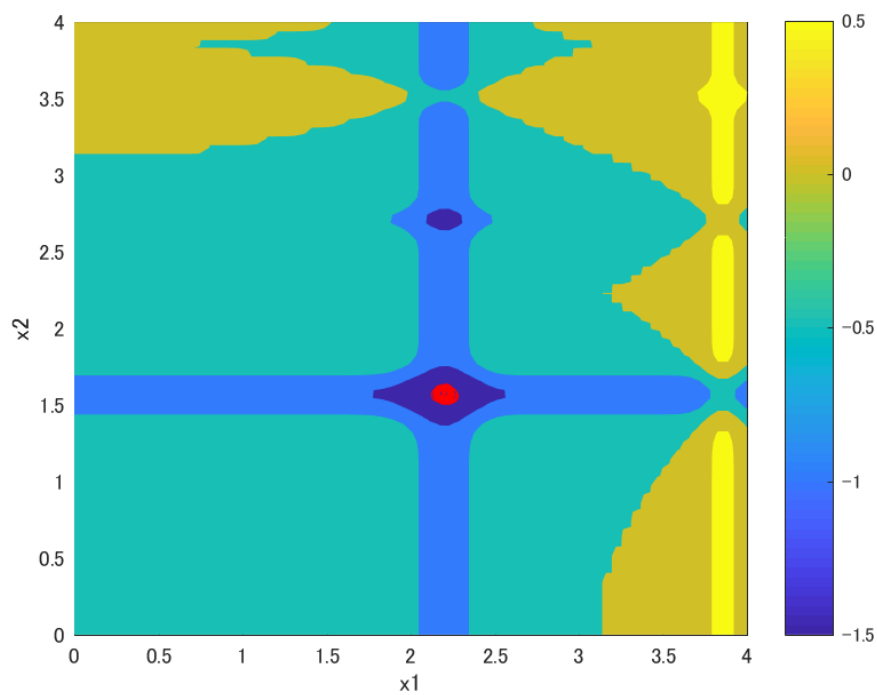
かる機構が働いたと考えられる。以上より探索領域を分割し、各個体の探索領域内での大域探索性能を保ちながらも徐々に局所解へ収束させることが、解の発見率の増加に繋がったと考えられる。



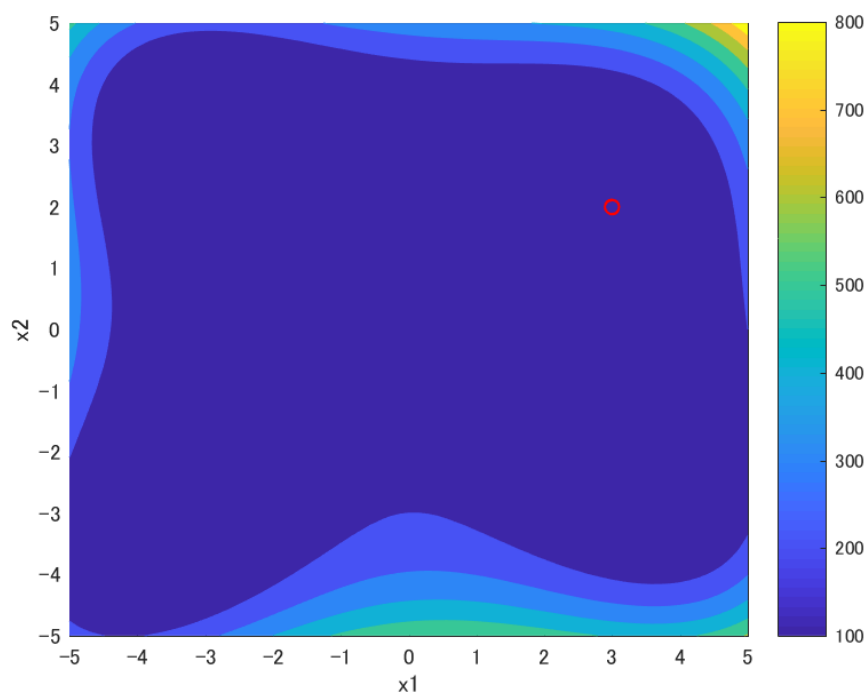
(a) F_1 : Griewank



(b) F_3 : Six-Hump Camel

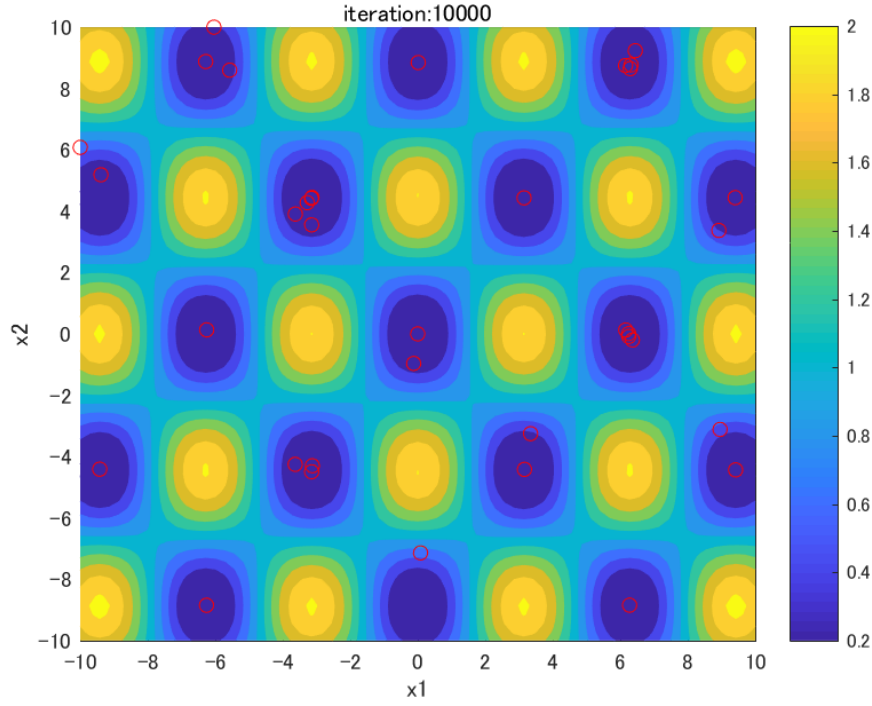


(c) F_4 : Michalewicz

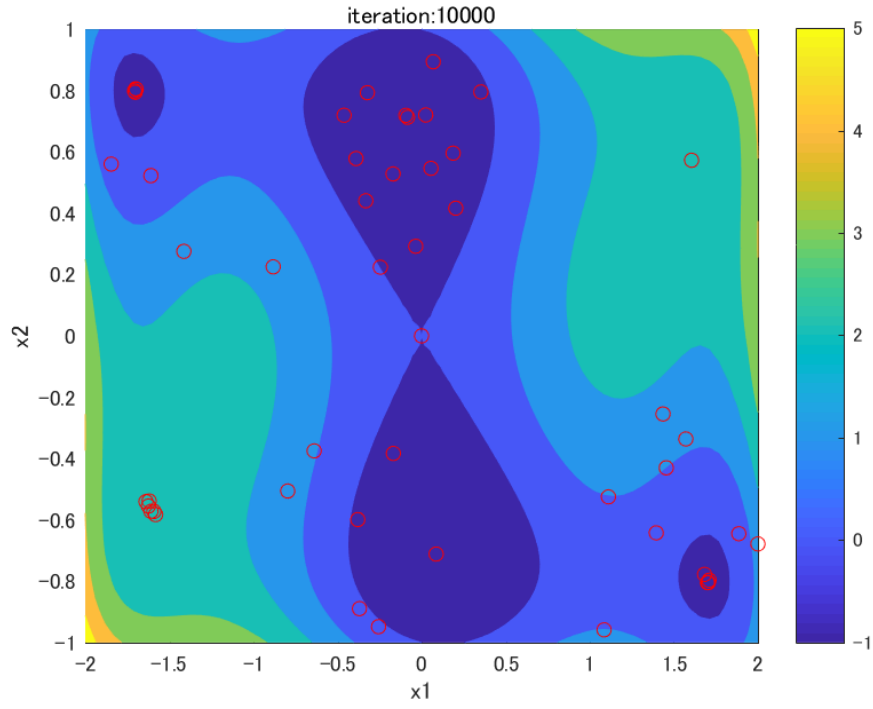


(d) F_5 : Himmelblau

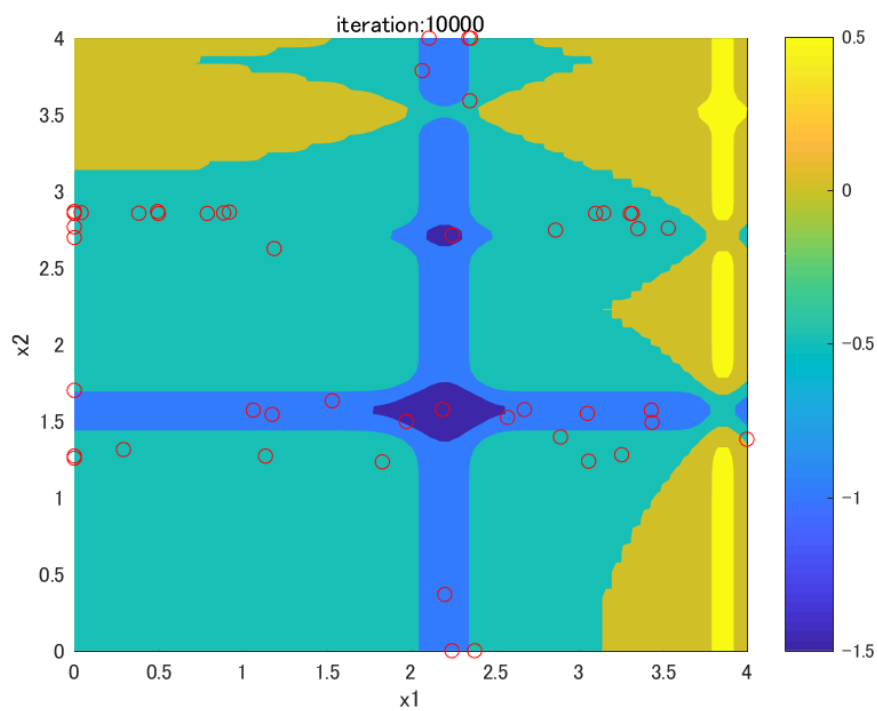
図 7.2: BA



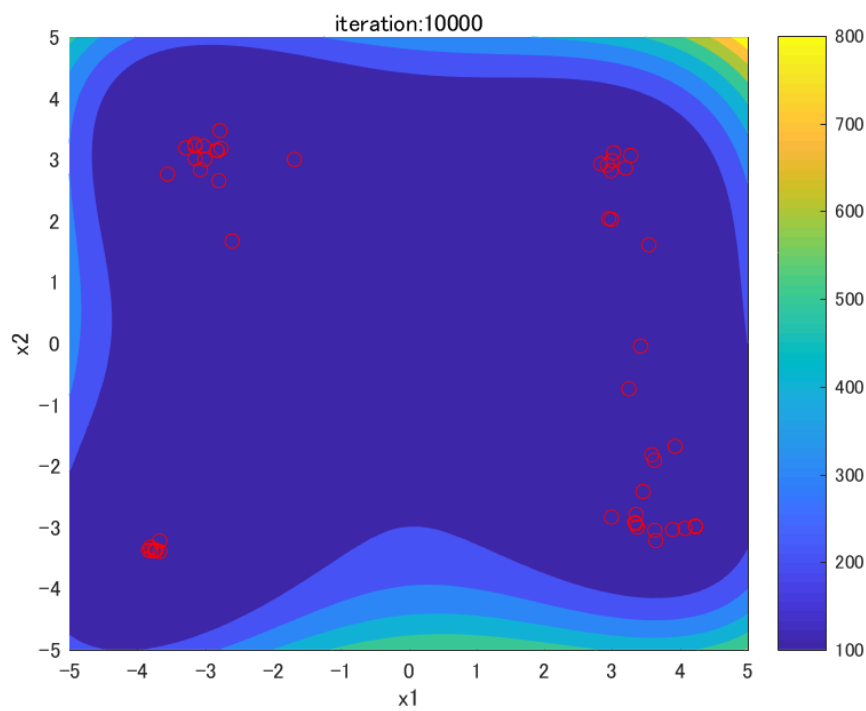
(a) F_1 : Griewank



(b) F_3 : Six-Hump Camel



(c) F_4 : Michalewicz



(d) F_5 : Himmelblau

図 7.3: NRBA

8 Dynamic Niche Radius-based Bat Algorithm (DNRBA)

本章では, NRBA で局所解に収束しなかった個体を最適解や局所解へ移動させることで, 収束性能を高めることを目的とした Dynamic Niche Radius を BA に適用させた DNRBA を提案する.

8.1 メカニズム

3.4 節で説明した Dynamic Niche Sharing [8] より, 探索領域に属する個体と同じ局所解に収束することを避けるため, 本研究では以下の変更を加えた.

$$m_i^{dyn} = \begin{cases} \sigma & (\text{if } m_i < \sigma) \\ m_i & (\text{otherwise}) \end{cases} \quad (8.1)$$

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + (\mathbf{x}_i^t - \mathbf{x}_{NR*}) * f_i \quad (8.2)$$

\mathbf{v}_i , \mathbf{x}_i は個体の速度と位置を表し, \mathbf{x}_{NR*} は \mathbf{x}_i が属する Niche Radius 内の最良個体を示す. (8.1) 式に基づいて, 個体の更新式は以下で表される.

$$\mathbf{x}_i^{t+1} = \begin{cases} \mathbf{x}_i^t + \mathbf{v}_i^{t+1} & (\text{if } m_i < \sigma) \\ \mathbf{x}_i^t & (\text{otherwise}) \end{cases} \quad (8.3)$$

ここでは個体の分布密度が高いほど, 個体 \mathbf{x}_i の持つ Niche Count m_i の範囲内にある最良個体 \mathbf{x}_{NR*} から遠ざかる方向へ新たに解候補を生成する.

局所探索では (8.1) 式で算出した最良個体 \mathbf{x}_{NR*} が持つ m_i の範囲内で個体 \mathbf{x}_i が新たに解候補 \mathbf{x}_{loc} を生成する.

$$\mathbf{x}_{loc} = \mathbf{x}_{NR*} + A_i^t * rand(1, D, [-m_i, m_i]) \quad (8.4)$$

ランダム探索では, 個体 \mathbf{x}_i の持つ m_i の範囲内で新たに解候補 \mathbf{x}_{rnd} を生成することで, 局所解への収束性能を高める. 生成式は次式で表される.

$$\mathbf{x}_{rnd} = \mathbf{x}_i^t + rand(1, D, [-m_i, m_i]) \quad (8.5)$$

8.2 アルゴリズム

DNRBA のアルゴリズムの疑似コードを以下の Algorithm 12, 13 に記す.

8.3 実験

DNRBA の収束性能を検証するため, ここでは BA と比較する. 本実験では最小化問題における, 最適解と局所解を持つ Griewank 関数と Shubert 関数を用いた.

Algorithm 12 Dynamic Niche Radius**Input:** Current Population $x_i (i = 1, 2, \dots, N)$ and v_i

```

for i=1 to N do
2:   for j=1 to N do
      Calculate  $d_{ij}$  between individuals  $i, j$ 
4:   if ( $d_{ij} < \sigma$ ) then
       $sh(d_{ij}) = (1 - \frac{d_{ij}}{\sigma})$  [Eq.(3.3)]
6:   else
       $sh(d_{ij}) = 0$  [Eq.(3.3)]
8:   end if
   end for
10:   $m_i = \sum_{j=1}^N sh(d_{ij})$  [Eq.(3.4)]
   end for
12: for i=1 to N do
      if ( $m_i < \sigma$ ) then
14:    $m_i^{dyn} = \sigma$  [Eq.(8.1)]
      else
16:    $m_i^{dyn} = m_i$  [Eq.(8.1)]
      end if
18: end for
return Dynamic Niche Radius  $m_i^{dyn}$ 

```

8.3.1 評価関数

実験で使用する評価関数は以下の通りである。

8.3.2 評価基準**8.3.3 Peak Ratio**

本実験では CEC2013 Competition で用いられた Peak Ratio (PR) [13] を用い、発見した解の数の割合を求める。PR の算出式は次式で表される。

$$PR = \frac{\sum_{run=1}^{MR} FPs}{TP * MR} \quad (8.6)$$

Max Run (MR) は実験回数を表し、Found Peaks (FPs) は発見した解の数を、Total Peak (TP) は探索領域内の全最適解及び局所解数を表す。また最適解及び局所解の位置座標と最近傍個体とのユークリッド距離が 0.1 未満であれば、その解を発見したと定義する。

Algorithm 13 Bat Algorithm with Dynamic Niche Radius (DNRBA)**Input:** Objective Function $F(x)$

```

Initialize Population  $x_i (i = 1, 2, \dots, N)$  and  $v_i$ 
Define frequency  $f_i$  at location  $x_i$  [Eq.(2.9)]
3: Initialize pulse rates  $r_i$ , and loudness  $A_i$ 
   while ( $t < \text{Max number of iterations}$ ) do
       Calculate Dynamic Niche Radius (Algorithm 2)
6:   for  $i=1$  to  $N$  do
       Generate a new solution  $x_i$  and velocity  $v_i$  [Eqs.(8.2) and (8.3)]
       if ( $\text{rand} > r_i$ ) then
9:         Generate a new solution  $x_{loc}$  around a global best solution  $x_i$  [Eq.(8.4)]
       end if
       Generate a new solution  $x_{rnd}$  randomly [Eq.(8.5)]
12:    if ( $\text{rand} < A_i \& \min(F(x_i), F(x_{loc}), F(x_{rnd})) < F(x_{i*})$ ) then
        Accept the new solution, and update pulse rate  $r_i$ 
        & loudness  $A_i$  [Eqs. (2.14) and (2.15)]
    end if
15:    Evaluate all bats and select a best solution  $x_*$  in the current solutions
   end for
end while

```

8.3.4 Peak Accuracy

探索する個体が最適解にどのくらい近づいているかを計測する尺度として Peak Accuracy (PA) [13] を用いる. PA の算出式は以下で表される.

$$PA = \sum_{j=1}^{TP} |F(s_j) - F(x_{NN_j})|, \quad (8.7)$$

この時, $F(s_j)$ は j 番目の最適解の評価値を表し, その最近傍個体 x_{NN_j} の評価値との差分和を求める. 則ち, PA 値が 0 に近づくほど, 個体が最適解に位置していることを示す.

8.4 実験設定

本実験では $f_{max} = 1$, $f_{min} = 0$, ラウドネス $A^0 = 1$, パルスレート $r^0 \in [0, 1]$ と設定した. また $\alpha = \gamma = 0.9$ とし, 個体数を 100, 世代数を 30000 としてランダムシードを変えた実験を 30 回行った.

8.5 実験結果と考察

本節では DNRBA の性能の効果を確かめるため, 各評価関数における PR と PA に着目する. 表 5, 6 は, BA と DNRBA のアルゴリズムにおける PR と PA の結果を表す. 各評

価関数の PR と PA の 30 試行の平均値及び標準偏差を示している．また図 8.1, 8.2 は最終世代の個体の分布を白い丸で表す．

表 5: PR and PA of BA and DNRBA (averaged over 30 runs)

$$\varepsilon = 1.0E - 1$$

	BA		DNRBA	
Function	PR (Mean and SD)	PA (Mean and SD)	PR (Mean and SD)	PA (Mean and SD)
F_1	0.2725 ± 0.0598	0.2416 ± 0.0149	0.9373 ± 0.1176	0.0094 ± 0.0155
F_2	0.4889 ± 0.1819	1.8160 ± 0.4990	0.4241 ± 0.388	2.4123 ± 0.6780

表 6: PR and PA of BA and DNRBA (averaged over 30 runs)

$$\varepsilon = 1.0E - 2$$

	BA		DNRBA	
Function	PR (Mean and SD)	PA (Mean and SD)	PR (Mean and SD)	PA (Mean and SD)
F_1	0.2725 ± 0.0608	0.2416 ± 0.0151	0.9373 ± 0.1176	0.0094 ± 0.0155
F_2	0.0556 ± 0.0619	1.8160 ± 0.4990	0.0426 ± 0.0477	2.4123 ± 0.6780

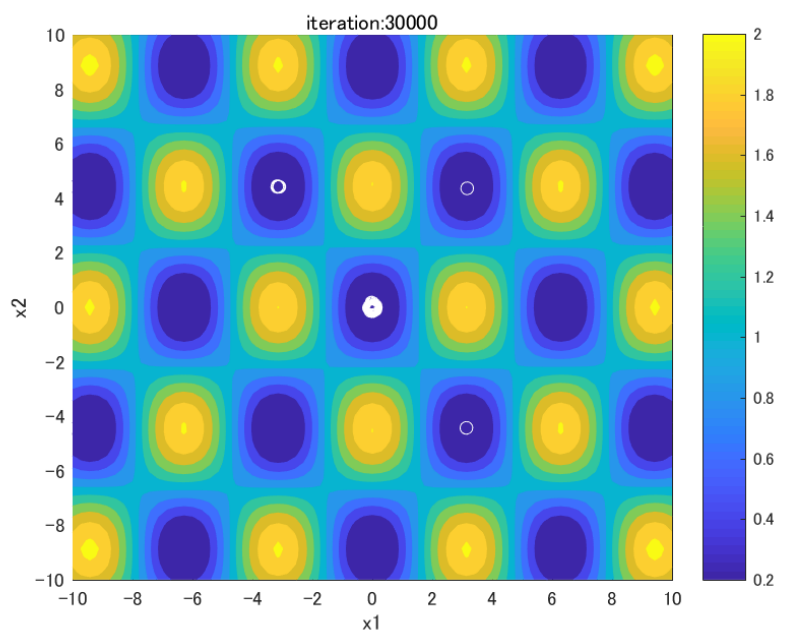
8.5.1 Peak Ratio

表 5 より $\varepsilon = 1.0E - 1$ の F_1 関数では，BA よりも DNRBA の PR 値の方が高く，図 ?? からほぼ全ての最適解と局所解を発見していることが分かる．対して F_2 関数では，図 1(b), 2(b) から個体が全最適解に位置しているように見て取れるが，DNRBA よりも BA の方が僅かに PR 値が高かった．このことから最適解しか存在しない関数については BA のほうが探索性能則ち収束性能が高いことが分かった． $\varepsilon = 1.0E - 2$ についても同様，PR 値は F_1 関数は DNRBA の方が高く， F_2 関数では BA の方が高かった．

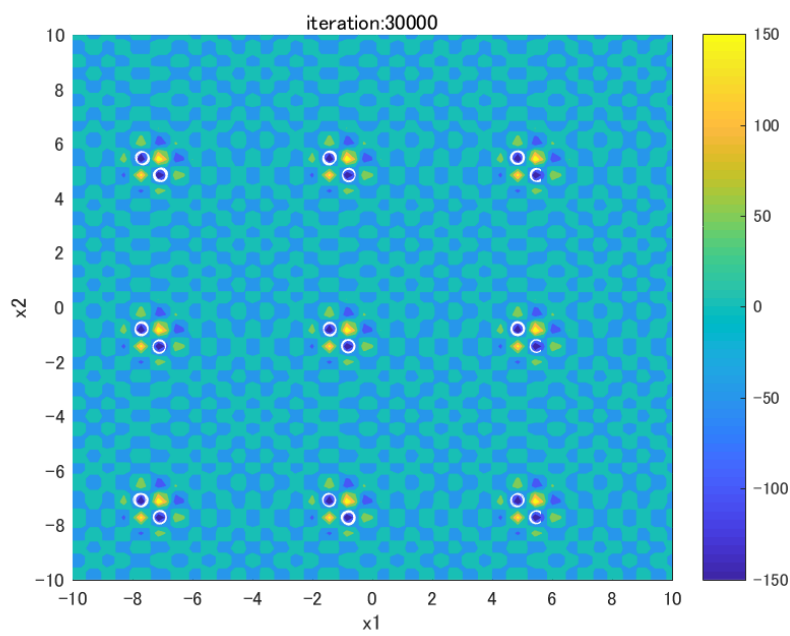
8.5.2 Peak Accuracy

表 5, 6 から F_1 関数では，BA より DNRBA の方が PA 値が高かったが， F_2 関数では BA の方が PA 値が高かった．

8.6 まとめ

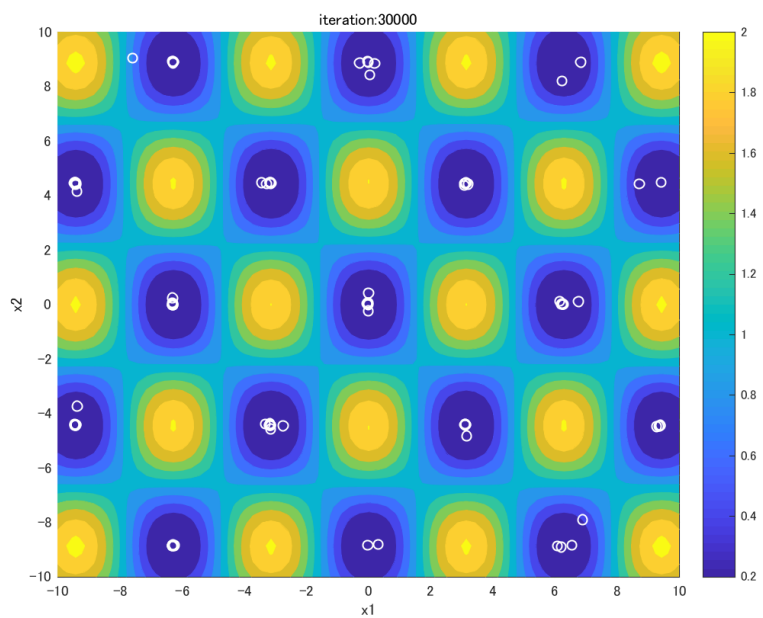


(a) F_1

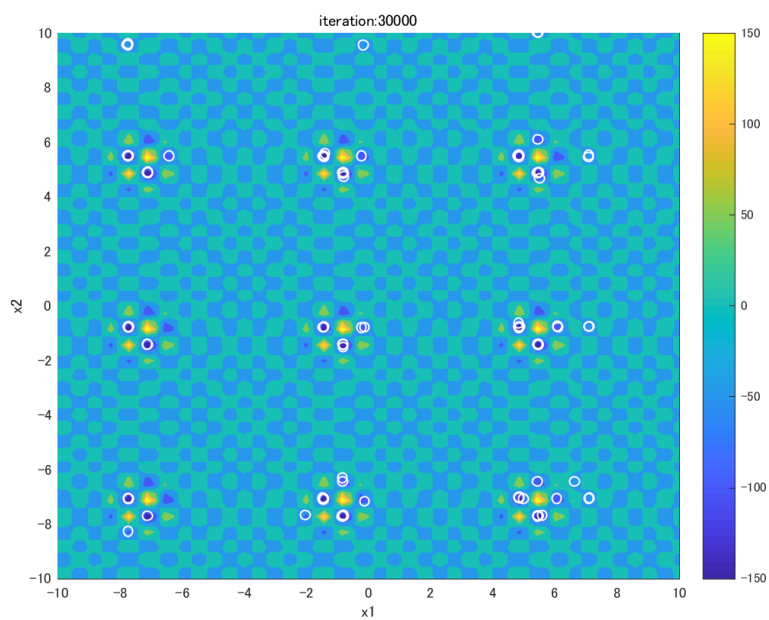


(b) F_2

図 8.1: BA



(a) F_1



(b) F_2

図 8.2: DNRBA

9 他の最先端手法との性能比較実験

9.1 複数解探索における多峰性最適化問題

9.2 評価関数

使用するベンチマーク関数とその概形は以下の通りである．

表 7: Benchmark Test Functions

Function	$F(x^*)$	Num of goptima	ρ	D	Search Range
F_1	200.0	2	0.01	1	$x \in [0, 30]$
F_2	1.0	5	0.01	1	$x \in [0, 1]$
F_3	1.0	1	0.01	1	$x \in [0, 1]$
F_4	200.0	4	0.01	2	$x, y \in [-6, 6]$
F_5	1.03163	2	0.5	2	$x \in [-1.9, 1.9],$ $y \in [-1.1, 1.1]$
F_6	186.731	18	0.5	2	$x_i \in [-10, 10]$

9.3 評価基準

本コンペティションでは PR [13] によりアルゴリズムの性能評価を行う．各評価関数で割り当てられた評価回数 (MaxFEs) と *accuracy level* ε を用いて PR は次式で表される．

$$PR = \frac{\sum_{run=1}^{NR} NPF_{run}}{NKP * NR} \quad (9.1)$$

NPF_{run} は、そのシードにおけるアルゴリズムが発見した最適解数を示し、NKP は評価関数を持つ全最適解数を示す．NR は実験の試行回数を示す．

9.4 結果

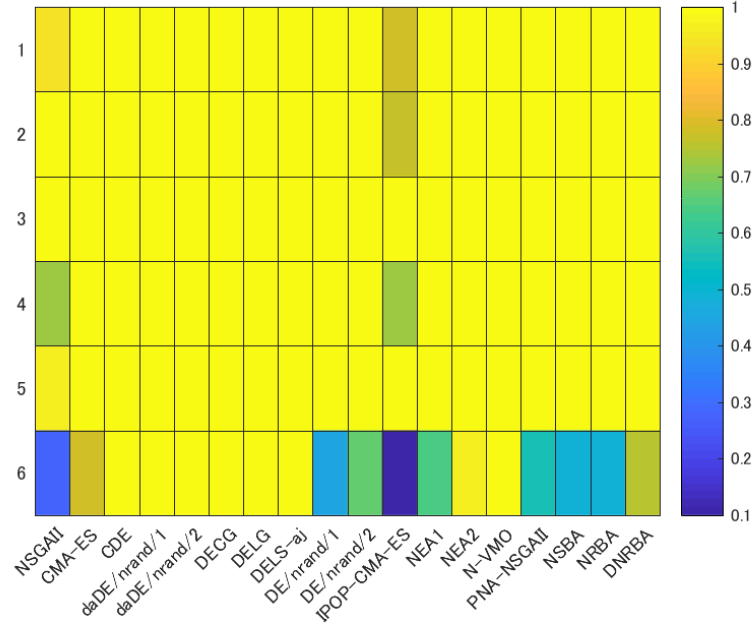


図 9.1: $\varepsilon = 1.0E - 1$

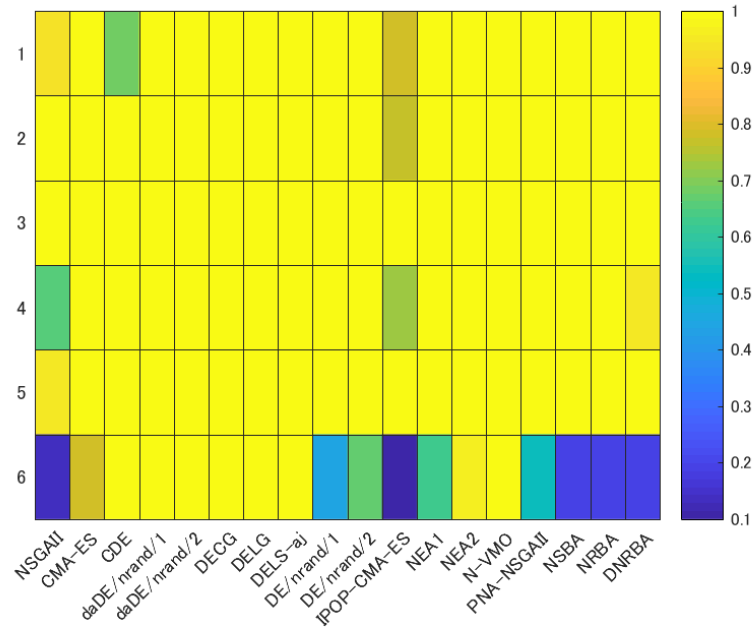


図 9.2: $\varepsilon = 1.0E - 2$

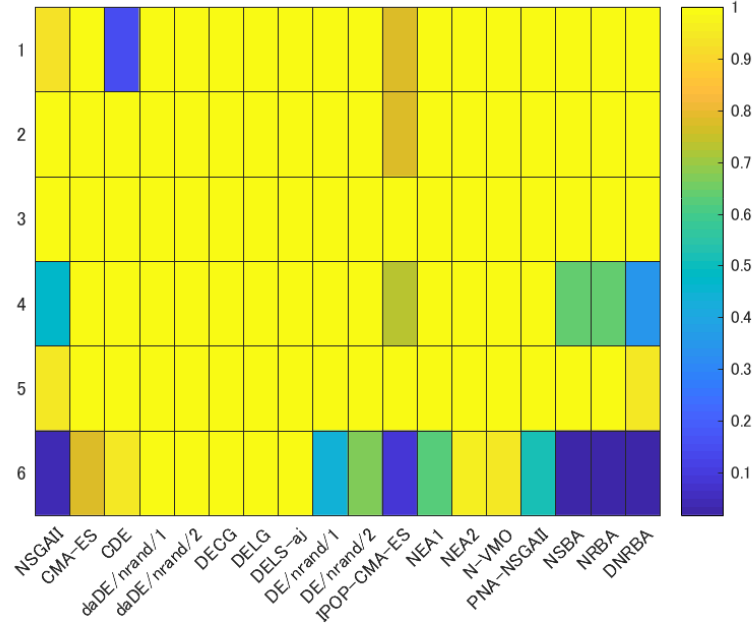


図 9.3: $\varepsilon = 1.0E - 3$

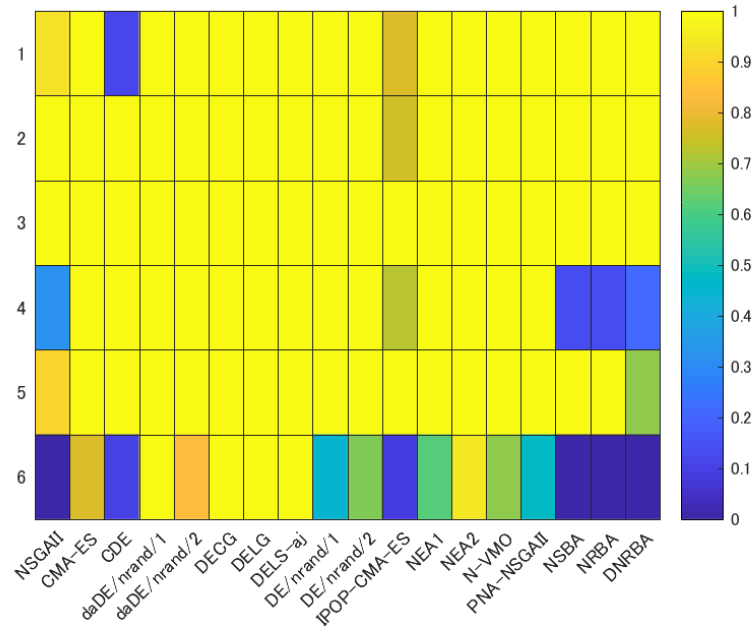


図 9.4: $\varepsilon = 1.0E - 4$

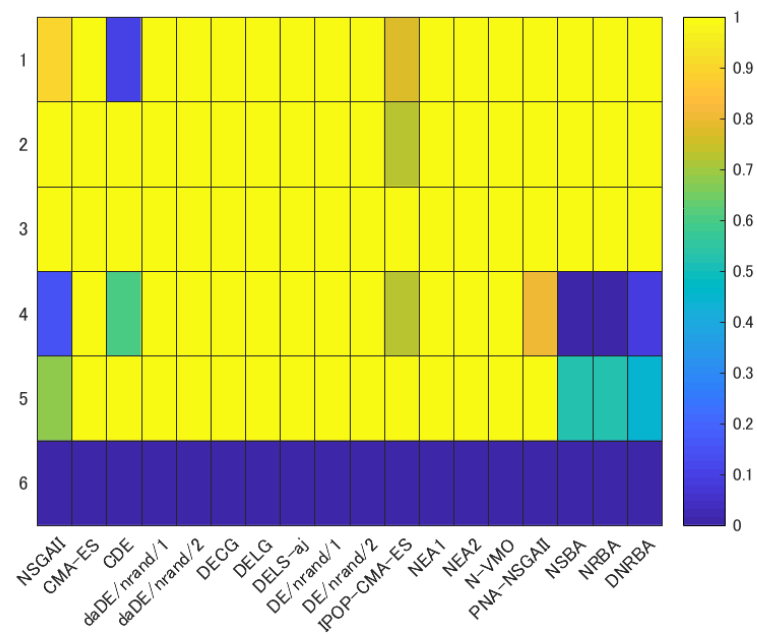


図 9.5: $\varepsilon = 1.0E - 5$

9.5 考察

10 おわりに

この章では本論文のまとめを 10.1 節で行い, 今後の課題を 10.2 節で行う.

10.1 まとめ

10.2 今後の課題

謝辞

本論文の執筆並びに研究を進める上で御指導頂いた高玉圭樹教授に感謝の意を表します。また、論文執筆において校正・校閲をして頂いた博士課程の高野諒さん、上野史人さん、日々の研究テーマに関して助言をして頂いた佐藤寛之准教授、並びに日々研究を共にしている研究室の皆様、研究を支えて頂いている皆様にこの場を借りて感謝致します。

参考文献

- [1] A. Kenneth De John. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Vol. 36. University of Michigan Ann Arbor, MI, USA, 1975.
- [2] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. in *Proc. of the Second International Conference on Genetic Algorithms*, pp. 41–49, 1987.
- [3] N. Hansen and A. Ostermeier. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_i, \lambda)$ -cma-es. *EUFIT'97, 5th Europ. Congr. on Intelligent Techniques and Soft Computing, Proceedings*, pp. 650–654, 1997.
- [4] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, Vol. 9(2), pp. 159–195, 2001.
- [5] J. H. Holland. Adaptation in natural and artificial systems. *University of Michigan Press, Ann Arbor, MI*, 1975.
- [6] J. Kennedy and R. Eberhart. A new optimizer using particle swarm theory. *Proc. Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, 1995.
- [7] J. Lehman and K. O. Stanley. Exploiting openendedness to solve problems through the search for novelty. In *ALIFE*, pp. 329–336, 2008.
- [8] B. Miller. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In: *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC' 96)*, 1996.
- [9] H. Pohlheim. Examples of objective functions. in *Documentation for Genetic and Evolutionary Algorithms for use with MATLAB:GEATbx version 3.7*, 2005.
- [10] K. Price R. Storn. Differential evolution a simple and efficient heuristic for global optimization over continuous spacesy. *International computer science institute*, Vol. 46, No. 2, 1995.
- [11] F. Uwano H. Sato T. Iwase, R. Takano and K. Takadama. Novelty search-based bat algorithm: Adjusting distance among solutions for multimodal optimization. *Proceedings of The 22nd Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES 2018)*, pp. 29–34, 2018.
- [12] R. Thomsen. Multimodal optimization using crowding-based differential evolution. In *the IEEE Congress on Evolutionary Computation*, Vol. 2, pp. 1382–1389, 2004.

- [13] A. Engelbrecht X. Li and M. G. Epitropakis. Benchmark functions for cec' 2013 special session and competition on niching methods for multimodal function optimization. *Evol. Comput.*, 2013.
- [14] X. S. Yang. A metaheuristic bat-inspired algorithm. *in: Nature Inspired Cooperative Strategies for Optimization*, Vol. 284, pp. 65–74, 2010.
- [15] J. Zhang and A. Sanderson. Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 5, pp. 945–958, 2009.
- [16] 上野史 佐藤寛之 高玉先生岩瀬拓哉. 複数解探索を考慮した分散型 bat algorithm. *SICE Symposium on Systems and Information 2018*, 2018.

付録