Differential Evolutionの性能向上と使用性向上への模索と評価

A Study of Modification and Evaluation of the Improvement of Performance and Usability for Differential Evolution

2016

岩井 亮

目 次

図 目 次

表 目 次

第1章

序

1.1　本研究の背景と本稿の概要

ある構造物やシステムを製作，運用するに当たって，形状や構造を設計する際，長さ，重さ，体積，係数，パラメータといった量的値の決定が必要となる．この量的値が明確な基準や，有力な目安を用いて決定される場合はよいが，そういった指標がない場合，しばしばこの値の決定において困難を生じることがある．このような問題に直面した場合，値を仮定することでその性能について評価できるならば，確率的最適化手法を用いることで値を決定することができる．確率的最適化手法は，解くべき問題を目的関数として書き下す際に，完全に静的な数式として書き下せない場合においても，入力と出力が一対一対応であるという関数の定義を満たしてさえいれば，最適化を試みることが可能だからである．しかし多くの確率的最適化手法においては，探索空間がより複雑な形状となるにつれて最適な値の組み合わせの発見は困難となり，一部の実問題では全く役に立たない，即ち製作物やシステムがその目的とする機能を発揮不可能な値を出力する場合も考えられる．

本研究では主に連続値最適化について着目することとし，第一に，現在の典型的な確率的最適化手法は探索空間に大域的単峰性を仮定している点に注目し，この大域的単峰性に乏しい探索空間においても最適化が可能となることを目指し，Differential Evolution の改良を行った．この報告を 2 章にて述べる．次にこの改良手法が実問題に対して有効であるかどうか確認するために，三つの古典的な問題と一つの新たに設計した実問題を用いて最適化実験を行った．この報告を 3 章にて述べる．それに続いて，最適化手法そのものを適用するために予め設定する必要のあるパラメータを決定することが困難である点に注目した．そしてこの困難さを取り除き有用性を向上させることを目指し，さらなる手法の改良を行った．この報告を 4 章にて述べる．

1.2　本研究で着目する問題範囲

確率的最適化手法は大きく分けて，離散値最適化を行うもの，連続値最適化を行うもの，組み合わせ最適化を行うものに分けられ，これらはそれぞれ手法の特性が異なる．したがって，全てについて同時に着目，比較することは困難であるため，本研究においては連続値最適化に着目することとし，その一例である Differential Evolution の性能向上についての取り組みを本稿にて述べる．特に Differential Evolution に着目した理由については 2.1.1 節にて述べる．

1.3　本研究の目標の重要性

1.1 節では非常に抽象的な性質のみを述べたが，確率的最適化手法，あるいは統計的最適化手法[1]と通称呼ばれ得る手法は，正しくは探索手法であり,「運が良ければ」最適解を発見し得るという点において最適化手法という表現がされ得るものと捉えることができる．したがって，確率的最適化手法は数学的に解を導出することが不可能あるいは困難である目的関数に対して,「一番最後に縋る手法」として主に用いるべき手法となる．これは，仮に問題が数学的に一意に解を導出可能であるならばそちらを優先するべきであることを示す．また，先述の通り確率的最適化手法はあくまで探索手法であり，人間に意味のある解の導出過程を出力せず，解の意味付けが大なり小なり必要な場合には利用不能あるいは困難であることも理由の一つである．このような手法でありながら，確率的最適化手法は場合により頼らざるを得ない手法であることも事実である．確率的最適化手法は多くの場合，最適化を試みるための条件として「入力に対して一意に出力が決まる」ことしか必要としていない．したがって，現状において人の手による試行錯誤でしか値を決めざるを得なかった問題に対して適用することが可能となる．幾つか具体的な問題例を挙げる．古典的な問題の例として，圧力器最適化問題が挙げられるだろう．問題の詳細は 3.3.1 節にて述べる．この問題は，提案された当初 [3] においても当時の工場の現状から改善された値が見つかっていたが，確率的最適化手法を適応することでより大きく改善され得ることが判明している [4, 5, 6]．圧力器最適化問題は辛うじて人間が試行錯誤し得る大きさかつ複雑度の問題であるが，当然より大きなあるいは複雑な問題においても用いられ得る．Hans らは水銀を酸化する際に，旧来の 50 以上の化学反応を用いる方法（Chemkin® [7] 内で用いられており，実用されることがある）の複雑性を問題として挙げ，5 つの反応のみを用いる方法を提案し，化学反応式内の係数と活性化エネルギー値の適切な値を MATLAB の確率的最適化を行う関数を用いて発見している [8]．昇平らは二足歩行機の制御方法としての Central Pattern Generator [9] を流用した歩行器 [10] に受動化制御を加えた方法を提案し，確率的最適化によって歩行器のパラメータを決定している [11]．星野は，ライントレースカーの制御をいわゆるガタガタにしないためにファジィ制御器を設計し，そのパラメータを自動調整するために確率的最適化手法を用いている [12]．これらように，確率的最適化手法に縋らざるを得ない場面

[1]英名は Stochastic Optimization である．

は多く考えられるため，その性能を向上させることは重要であると本研究において考えられた．即ち，より最適解の発見能力が高く，最適解を発見できなかった場合においてもより良い解を発見する手法が，実用を見据えるにおいてより重要であると本研究では捉えている．したがって，本研究では最適解を発見する能力として最適解発見回数割合を，より良い解を発見する能力として探索の継続能力を目安とし，これらの数値の高い手法を提案することを第一の目標とした．本研究で主に Differential Evolution に注目しているのは，これらの項目に対し比較的優秀な成績を示してくれるためである．

ところで，確率的最適化手法を適用せざるを得ない状況となった時に,「確率的最適化手法に設定するべきパラメータが分からない」という解決困難な事態に陥ることがある．これは即ち,「パラメータを設定するためにパラメータを設定しなければならないがそれが分からない」という本末転倒な事態が起こっていることを示す．本稿で主に注目する確率的最適化手法である Differential Evolution が持つこの問題に関わる詳細は 4.1.1 節にて述べている．最後に縋らざるを得ない手法として確率的最適化手法を選択している状況において，これ程に残酷な事態は存在しないだろう．4.1.1 節では問題点をより明確に述べるためにパラメータの総当たり実験を行っているが，問題の複雑さ，即ち目的関数の計算量が莫大なものであれば，総当たりによる導出は現実的ではなく，そして確率的最適化手法を用いる状況はしばしばそのような状況である．したがって，確率的最適化手法の使用性の高さもまた，重要な性質であると本研究において考えられた．それ故に，本研究において，確率的最適化手法の最適化性能の改善に次いで，使用性の向上も目標と設定された．具体的には，手法実装者，即ち確率的最適化手法を利用する人々によるパラメータ設定の必要性を廃絶，あるいは少なくともパラメータ設定の容易化を本研究における第二の目標とした．

まとめると，本研究においては最適解発見回数割合が高く探索継続力も高いような手法であり，かつパラメータの設定が不要であるような確率的最適化手法が有用であり重要な手法であると捉え，これを実現し得る手法の提案が目標とされた．そして 1.1 節にて述べた様に，その具体的方策として，第一に Differential Evolution の性能の改善，第二に使用性を向上させるための更なる改良が本研究にて行われた．

第2章

Differential Evolution on Scattered Parentsの提案とベンチマークによる性能評価

2.1　はじめに

2.1.1　背景

近年計算機の処理速度が飛躍的に向上したため，ある問題に対して，従来では膨大な時間が掛かり実質的に不可能であったような確率的探索による解法が実現可能となってきた．そこで，解析的に最適解の導出が困難な問題に対し，確率的アルゴリズムに基づいた数値的な解の探索を試みる最適化手法の研究が数多くなされるようになった [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]．これらの手法は探索空間の勾配情報が不要であり，目的関数が不明，または関数の形式で書き下せないような問題であっても最適化が可能である [24].

確率的最適化手法の中でも，Particle Swarm Optimization（PSO） [16] や，Genetic Algorithm（GA） [17]，Evolution Strategy（ES） [18] などの手法が注目されている．これらの手法は多点探索手法であり，探索空間内の複数箇所を同時に探索することから，Simulated Annealing [13] のような単点探索手法よりも多峰性探索空間において最適解を容易に発見可能であると考えられている．

多点探索手法の一つとして，Storn と Price によって提案された Differential Evolution（DE） [19] という手法がある．DE は，GA や ES の系統から派生した手法であり，親個体を用いた単純なベクトル演算により突然変異個体を生成し，この突然変異個体と親個体と交叉させることで子個体を生成する．そして，他の GA や ES の典型的な実装と比較して，予め与えなければならないパラメータが少なく，実装の容易な最適化手法でありながら，それらよりも最適解への収束が速く，さらに最適解発見前に探索が収束してしまう可能性も少なく，頑健であると考えられている [25, 26]．これら DE の性質から，

実システムへ適用を試みる研究も行われている．Bošković らは，チェスプログラム内の戦略を評価する関数を DE で最適化し，より強力なチェスプログラムへ調整することに成功した [27]．今川らは，通貨の売買のタイミングを計る指標と，通貨の上昇/下降傾向を判断する指標の計算式内の重みを DE により最適化することで，複数の局面において平均交叉 GA を用いて最適化を行った場合よりも高い利益を上げることに成功している [28]．

2.1.2　研究目的

前節に挙げられるような典型的な確率的最適化手法は，探索の際に，探索空間の大域的単峰性 [29, 30] を期待，あるいは前提としている．したがって，大域的単峰性の乏しい探索空間においては，これ以外の探索困難な特徴がなくともほとんどの試行で最適解を発見できず，局所解に収束したまま探索が停滞してしまう．

本研究では，大域的単峰性に乏しい探索空間においても，従来から存在する確率的最適化手法より安定して最適解を発見可能な確率的最適化手法として，親の散在性を利用し突然変異個体生成の多様性を向上させた Differential Evolution on Scattered Parents（DE-SP）を提案する．DE-SP では，DE における突然変異個体の生成の際，次元要素ごとに親を選び直しつつ計算し，突然変異個体をより多様とすることで，探索され得る点を多様化した．この多様化により最適解発見前に最適解以外の局所解へ収束することによる探索の停滞[1]が抑制される．

2.1.3　対象とする問題

本研究で扱う最適化問題は，原則として無制約であり定義域と値域が共に連続値となる目的関数 $f : \mathbb{R}^D \mapsto \mathbb{R}$ を最小化する問題である．ただし D は定義域の次元数である．即ち，

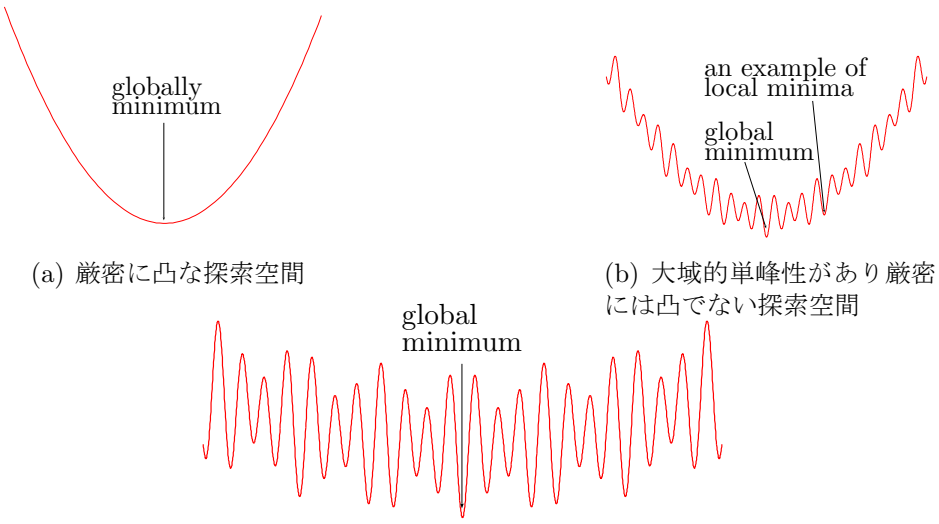
$$\boldsymbol{\chi}^* = \arg\min_{\boldsymbol{\chi}} f(\boldsymbol{\chi}) \tag{2.1}$$

となる $\boldsymbol{\chi}^*$ を求める問題を扱う．

2.2　大域的単峰性の仮定の妥当性

最適化において，しばしば図 2.1(a) の例に代表される厳密に単峰（凸）である探索空間が仮定される．しかし，実問題を最適化するにあたり，探索空間が厳密に単峰であるとする仮定は必ずしも妥当ではない．このため，厳密に単峰な探索空間より実問題に近い仮定として，図 2.1(b) の例に代表される，大域的には単峰性の性質を持つと考えられ

[1] この現象は分野により初期収束と呼ばれる．



(a) 厳密に凸な探索空間

(b) 大域的単峰性があり厳密には凸でない探索空間

(c) 大域的単峰性に乏しい探索空間

図 2.1: 探索空間の形状例

る性質が用いられる．この性質は大域的単峰性と呼ばれる．大域的単峰性が認められる問題の例として，Boese [30] は巡回セールスマン問題にこの性質が認められると報告している．

大域的単峰性のある探索空間においては，2.1.1 節にて例示した最適化手法などを用いた最適解の探索が十分に有効であると考えられている [29]．しかし二次割り当て問題やジョブショップスケジューリング問題などの実問題，実世界のシミュレーションなどの，わずかな差分が目的関数値に大きな影響を与え得る問題では，図 2.1(c) の例に代表される，大域的単峰性が全体的，あるいは部分的に仮定できない探索空間である可能性が存在する [31]．したがって，最適化手法は図 2.1(c) に代表される空間においても大域的単峰性が仮定できる探索空間同様の最適化性能を維持可能であることが望ましいと考えられる．

2.3　関連手法

2.3.1　Differential Evolution

Differential Evolution（DE）は，進化戦略に基づく最適化手法の一つである．DE の具体的実装は，同時に複数の形態が提案された [19] が，本研究では DE/rand/1/bin を用いる．DE の擬似コードをアルゴリズム 1 に示す．ここで，D を探索空間の次元数，N を個体数，F を突然変異個体生成時に用いるパラメータ，C_R を交叉時に行われる確率的な判定に用いる閾値とする．DE では，まず N 体の個体 $\boldsymbol{\chi}^i$ $(i = 1, 2, \ldots, N)$ を探索空間の定義域を対象範囲とした一様乱数にて初期化し，各個体の目的関数値を計算する．そして，終了条件を満たすまでの間，突然変異個体の生成，子個体候補の生成，次の摂動

アルゴリズム 1 Differential Evolution

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F \in [0, 2] \cdots$ スケーリングパラメータ
$C_R \in [0, 1] \cdots$ 交叉時に用いる閾値
$R_{\mathcal{U}}(\mathbb{R}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の実数一様乱数
$R_{\mathcal{U}}(\mathbb{N}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の自然数一様乱数

for $i = 1$ to N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
end for
while 終了条件が未成立 **do**
　for $i = 1$ to N **do**
　　サブ親 $\boldsymbol{\chi}^a$, $\boldsymbol{\chi}^b$, $\boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ をランダムに選択
　　突然変異個体を計算: $\boldsymbol{\chi}'^i \leftarrow \boldsymbol{\chi}^a + F(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c)$
　　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
　　for $k = 1$ to D **do**
　　　if $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < C_R$ **or** $k = l^i$ **then**
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k'^i$
　　　else
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k^i$
　　　end if
　　end for
　　目的関数 $E(\boldsymbol{p}^i)$ を計算
　end for
　for $i = 1$ to N **do**
　　if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
　　　親個体を子個体と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
　　else
　　　親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
　　end if
　end for
end while

で用いる個体群の生成，を繰り返す．突然変異個体は各親個体 $\boldsymbol{\chi}^i$ ごとに生成される．まず，$\boldsymbol{\chi}^i$ に重複せずかつ互いに重複しないサブ親 $\boldsymbol{\chi}^a$，$\boldsymbol{\chi}^b$，$\boldsymbol{\chi}^c$ が選択され，下式のベクトル演算にて突然変異個体 $\boldsymbol{\chi}'^i$ が生成される．

$$\boldsymbol{\chi}'^i = \boldsymbol{\chi}^a + F(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c) \tag{2.2}$$

子個体候補 $\boldsymbol{p}^i$ は，親個体 $\boldsymbol{\chi}^i$ と $\boldsymbol{\chi}^i$ から突然変異した個体 $\boldsymbol{\chi}'^i$ を交叉することで生成される．

$$\begin{cases} p_k^i = \chi_k'^i & (R_{\mathcal{U}}(\mathbb{R}, [0, 1]) \leq C_R \text{ or } k = l^i) \\ p_k^i = \chi_k^i & (R_{\mathcal{U}}(\mathbb{R}, [0, 1]) > C_R \text{ and } k \neq l^i) \end{cases} \qquad (k = 1, 2, \ldots, D) \tag{2.3}$$

ここで，$R_{\mathcal{U}}(\mathbb{R}, [0, 1])$ は範囲 $[0, 1]$ の実数値一様乱数，l^i は各親ごとに設定される定数であり，範囲 $[1, D]$ の自然数値一様乱数である．生成された子個体候補 $\boldsymbol{p}^i$ は，目的関数値

$E(\boldsymbol{p}^i)$ が親の目的関数値 $E(\boldsymbol{\chi}^i)$ よりも改善が認められた場合に親個体 $\boldsymbol{\chi}^i$ と置き換えられ，$E(\boldsymbol{p}^i)$ が $E(\boldsymbol{\chi}^i)$ よりも改悪された場合には置き換えられず，親個体 $\boldsymbol{\chi}^i$ は変更されない．

2.1.1 章にて DE は典型的な GA や ES の実装と比較して強力な手法であると述べたが，大域的単峰性に乏しい探索空間においては最適化が困難である．DE の突然変異個体は単純なベクトル演算によって生成されるため，一様乱数を用いた突然変異などと比較して，突然変異個体の多様性が低い．したがって，局所解への収束傾向が現れたとき，他の解の発見が困難となるため，この問題が現出すると考えられる．

2.3.2　Differential Evolution の既存の改良

DE の最適化性能をより向上させるために様々な DE 改良手法の研究が行われている．例えば，突然変異個体生成時に前節における DE の説明内の $\boldsymbol{\chi}^a$ にあたるサブ親をメイン親 $\boldsymbol{\chi}^i$ の最近傍個体とすることで探索の効率化を試みた DE/nrand/1 [32]，メイン親 $\boldsymbol{\chi}^i$ を除いた個体群での孤立個体をサブ親 $\boldsymbol{\chi}^a$ とし，$\boldsymbol{\chi}^b$，$\boldsymbol{\chi}^c$ を $\boldsymbol{\chi}^i$ の近傍個体から選択することで，突然変異個体を孤立個体周辺に生成される確率を上げ，探索領域の多様性の維持を試みた DE/isolated/1 [33] が挙げられる．他にも，PSO において各粒子が記憶する局所最適解を，DE を用いて遷移させることで PSO と DE を並列に動かし，最適化能力の向上を試みた Hybridizing Particle Swarm Optimization with Differential Evolution（PSO-DE） [34]，Wavelet Function を用いて DE のパラメータである F を動的に調整し，さらに子個体の位置の調整も行うことで，より早期に最適解を発見する事を目的とした Wavelet-Mutation-Wavelet-Crossover-Based Differential Evolution（WMWC-DE） [35] といった手法が挙げられる．これらの手法の具体的実装は 2.5.2 節にて述べる．

以上に取り上げた手法において，突然変異個体を単純なベクトル演算で生成している点は DE と同様であり，突然変異個体の多様性が低い．このため大域的単峰性が乏しい空間において他の解の発見が困難となる問題は残存すると考えられる．

本稿で提案する DE-SP では，突然変異個体生成演算を次元ごとに分け，各スカラー値ごとに親を選び直しながら演算を行うことで，従来の演算よりも突然変異個体の多様性が高くなる．このため，探索の続行が可能となり，最適解がより確実に発見できると考えられる．

2.4　Differential Evolution on Scattered Parents

前節で挙げた DE の問題点を改善するために，突然変異個体を生成するときに次元ごとにサブ親を選び直し，目的関数の値が最悪値から M 番目までにある親を目的関数の値に依存せず常に子個体の候補と交換する変更を加えた Differential Evolution on Scattered Parents（DE-SP）を提案する．DE-SP の擬似コードをアルゴリズム 2 に示す．ここで，DE から改変された操作をアルゴリズム 2 内にて斜体で示す．DE では各親個体

アルゴリズム 2 Differential Evolution on Scattered Parents

1: $D \cdots$ 探索空間の次元数
2: $N \cdots$ 個体数
3: $F \in [0, 2] \cdots$ スケーリングパラメータ
4: $C_R \in [0, 1] \cdots$ 交叉時に用いる閾値
5: $M \cdots$ 改悪を受理する個体数
6: $R_{\mathcal{U}}(\mathbb{R}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の実数一様乱数
7: $R_{\mathcal{U}}(\mathbb{N}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の自然数一様乱数
8:
9: **for** $i = 1$ to N **do**
10:　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
11:　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
12: **end for**
13: **while** 終了条件が未成立 **do**
14:　**for** $i = 1$ to N **do**
15:　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
16:　**for** $k = 1$ to D **do**
17:　サブ親 $\boldsymbol{\chi}^{a_k}$, $\boldsymbol{\chi}^{b_k}$, $\boldsymbol{\chi}^{c_k}$ $(i \neq a_k \neq b_k \neq c_k)$ をランダムに選択
18:　突然変異個体を計算: $\chi'^i_k \leftarrow \chi^{a_k}_k + F(\chi^{b_k}_k - \chi^{c_k}_k)$
19:　**if** $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < C_R$ **or** $k = l^i$ **then**
20:　子個体候補の要素を次のように設定: $p^i_k \leftarrow \chi'^i_k$
21:　**else**
22:　子個体候補の要素を次のように設定: $p^i_k \leftarrow \chi^i_k$
23:　**end if**
24:　**end for**
25:　目的関数 $E(\boldsymbol{p}^i)$ を計算
26:　**end for**
27:　**for** $i = 1$ to N **do**
28:　**if** $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **or** $E(\boldsymbol{\chi}^i)$ が最悪値から M 番目までの値である **then**
29:　親個体を子個体候補と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
30:　**else**
31:　親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
32:　**end if**
33:　**end for**
34: **end while**

$\boldsymbol{\chi}^i$ $(i = 1, 2, \ldots, N)$ に対してサブ親は一度しか選択されない（アルゴリズム 1，15 行目）が，DE-SP では次元ごとに選び直し（アルゴリズム 2，17 行目），これに応じて突然変異個体の演算も各次元ごとに行う．このため，突然変異個体を生成する式は式 (2.2) と異なり下式となる．

$$\chi'^i_k = \chi^{a_k}_k + F(\chi^{b_k}_k - \chi^{c_k}_k) \quad (k = 1, 2, \ldots, D) \tag{2.4}$$

ここで，k は個体の各次元を示す．この変更により，生成され得る突然変異個体が DE より多様となる．したがって，生成され得る子個体がより多様となり，DE よりも多様な探索領域が実現された．その一方で，各次元ごとに独立して演算が行われるため，探索領域の各次元間の相関は考慮されない．しかしながら，各次元間の相関を再度 DE-SP に取り入れると，探索初期に次元間の相関の影響を強く受けた場合に探索点の多様性が急速に減少し，探索が硬直する可能性がある．これは本研究の目的に反するため，各次

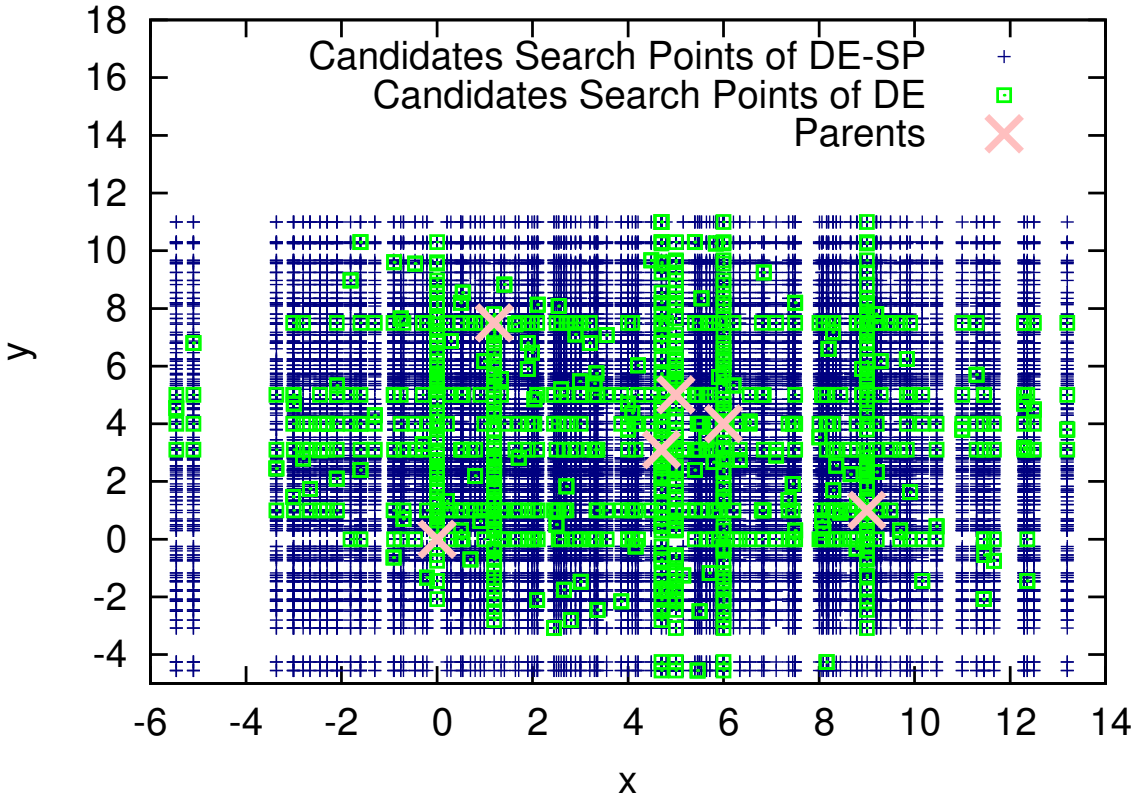


図 2.2: 2 次元探索空間における探索点候補の例

元間の相関を DE-SP に取り入れることは行わなかった．以後，生成され得る子個体のことを探索点候補とする．DE-SP における探索点候補の，DE との比較例を図 2.2 に示す．これは 2 次元上の探索空間において，$F = 0.7$，C_R は範囲 $(0, 1)$ のいずれか，個体数 $N = 6$ とした場合の DE-SP と DE それぞれの探索点候補を描画したものである．この図から，DE-SP では DE よりも子個体が広く多様な位置に生成され得ることが認められる．また，最悪値から M 番目までの個体であれば，無条件に親個体を子個体で置き換える（アルゴリズム 2 28 行目）ことで，一部の個体は探索が収束する方向とは逆方向への探索が可能となる．したがって，目的関数値が改善されたときのみ親個体が子個体と置き換えられる DE と比較して，DE-SP は局所解から離脱がしやすいと考えられる．以上より，DE-SP では最適解発見前に，比較的局所解へ収束しづらく，大域的単峰性が乏しい探索空間においても探索が継続可能と考えられる．

2.5　ベンチマークによる最適化実験

DE-SP の有効性を評価するため，ベンチマークを用いた最適化実験を行う．本実験においては五つのベンチマーク関数による評価と，二つの独自ベンチマーク関数による評価を行う．五つのベンチマークを用いた実験により DE-SP が通常仮定される形状の探索空間において有効であるか確認する．また，二つの独自ベンチマーク関数を用いた実

験により，本研究の目的である大域的単峰性に乏しい空間における最適化に有効であるか，大域的単峰性はあるが局所解の位置が不規則であるような探索空間における最適化に有効であるかを確認する．

2.5.1　使用するベンチマーク関数

最適化手法の性能を評価するためのベンチマーク関数が数多く提案されている．本実験では，次のベンチマーク関数を用いた．

- Rastrigin Function
- Rosenbrock Function
- Schwefel Function
- Griewank Function
- Ackley Function
- Noisy Function 1
- Noisy Function 2

次に，それぞれのベンチマーク関数についての概要を説明する．ただし以後の説明において，定義域の次元数を D，関数の入力値を $\boldsymbol{\chi}$ $(\in \mathbb{R}^D)$ とする．

Rastrigin Function

Rastrigin Function は多峰性関数であり，次のように表される．

$$f_{\mathrm{Ras}}(\boldsymbol{\chi}) = aD + \sum_{k=1}^{D}(\chi_k^2 - a\cos(2\pi\chi_k)) \tag{2.5}$$

ここで，a は予め設計する定数である．本実験において，定義域は $[-5.12, 5.12]^D$，$a = 10$ とする．最適解は，$f_{\mathrm{Ras}}(\mathbf{0}) = 0$ である．

Rosenbrock Function

Rosenbrock Function は変数間に依存性のある単峰性関数であり，次のように表される．

$$f_{\mathrm{Ros}}(\boldsymbol{\chi}) = \sum_{k=1}^{D-1}\left(100(\chi_{k+1} - \chi_k^2)^2 + (1 - \chi_k)^2\right) \tag{2.6}$$

本実験において，定義域は $[-2.048, 2.048]^D$ とする．最適解は，$f_{\mathrm{Ros}}(\mathbf{1}) = 0$ である．

Schwefel Function

Schwefel Function は探索領域の境界線寄りに最適解の存在する多峰性関数であり，次のように表される．

$$f_{\mathrm{Sch}}(\boldsymbol{\chi}) = \sum_{k=1}^{D} -\chi_k \sin(\sqrt{\|\chi_i\|}) \tag{2.7}$$

本実験において，定義域は $[-512, 512]^D$ とする．最適解は，
$f_{\mathrm{Sch}}(\mathbf{420.968750}) = -418.98288727D$ である．

Griewank Function

Griewank Function は変数間に依存性があり，典型的な大域的単峰性はあるが，大量の局所解が存在する多峰性関数であり，次のように表される．

$$f_{\mathrm{Gri}}(\boldsymbol{\chi}) = 1 + \sum_{k=1}^{D} \frac{\chi_k^2}{4000} - \prod_{k=1}^{D} \left(\cos\left(\frac{\chi_k}{\sqrt{k}} \right) \right) \tag{2.8}$$

本実験において，定義域は $[-512, 512]^D$ とする．最適解は，$f_{\mathrm{Gri}}(\mathbf{0}) = 0$ である．

Ackley Function

Ackley Function は中心に近づくにつれ急速に値が減少する傾向を示す多峰性関数であり，次のように表される．

$$f_{\mathrm{Ack}}(\boldsymbol{\chi}) = -a \exp\left(-b \sqrt{\frac{1}{D} \sum_{k=1}^{D} \chi_k^2} \right) - \exp\left(\frac{1}{D} \sum_{k=1}^{D} \cos(c\chi_k) \right) + a + e \tag{2.9}$$

ここで，a，b，c は予め設計する定数である．本実験では，定義域は $[-32.768, 32.768]^D$，$a = 20$，$b = 0.2$，$c = 2\pi$ とする．最適解は，$f_{\mathrm{Ack}}(\mathbf{0}) = 0$ である．

Noisy Function

前述の五つのベンチマーク関数は，基本的に大域的単峰性を持ち，格子状や同心円上に局所解が配置されているものが多い．また，入力値に対して不規則的な過敏性を持つ場合，例えばハンチングを起こしている様な状態の探索空間は考慮されていない．このような $\mathbb{R}^2$ 上の探索空間を想定した場合の最適化能力を評価するためのベンチマーク関数として，Noisy Function 1（NF1），Noisy Function 2（NF2）を生成し，本実験にて用いる．

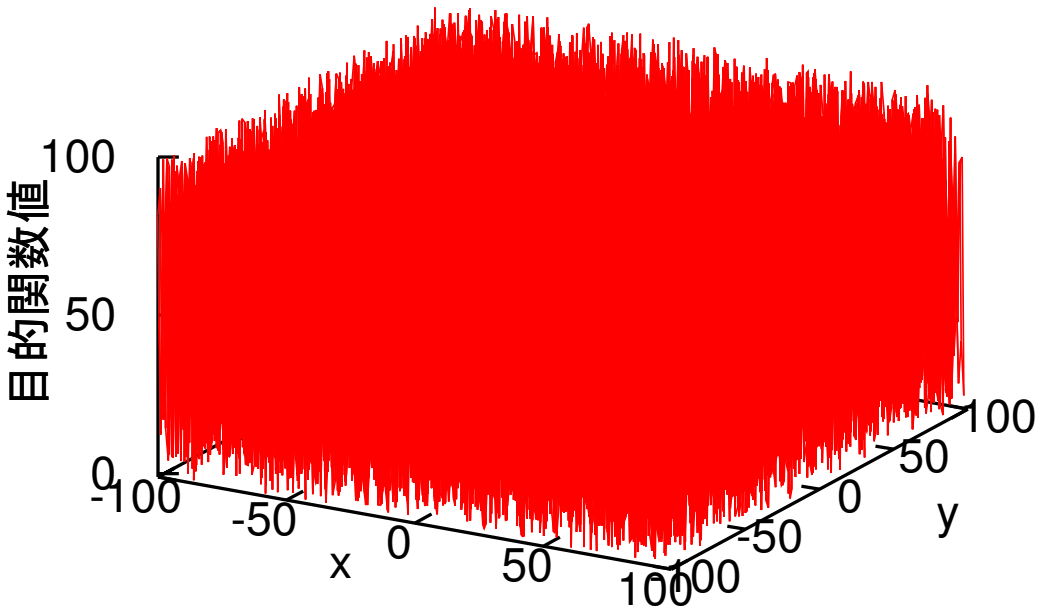


図 2.3: Noisy Function 1 の外観

Noisy Function 1　NF1 は，入力値に対して不規則な過敏性を持ち，かつ大域的単峰性に乏しい問題を想定したベンチマーク関数であり，この外観を図 2.3 に示す．具体的な生成方法は次の通りである．

- 定義域は $[-100, 100]^2$ とする．
- 整数格子点上の目的関数の値を，$R_{\mathcal{U}}(\mathbb{R}, [0, 100])$ を用いて配置する．ただし，$f_{\mathrm{NF1}}(0, 0) = -1$ とし，これを最小値とする．
- 整数格子点以外の座標 (x, y) の目的関数の値は，$(\lfloor x \rfloor, \lfloor y \rfloor)$，$(\lfloor x \rfloor + 1, \lfloor y \rfloor)$，$(\lfloor x \rfloor, \lfloor y \rfloor + 1)$，$(\lfloor x \rfloor + 1, \lfloor y \rfloor + 1)$ を用いた 4 点バイリニア法により決定する．4 点バイリニア法は付録 A.1 にて説明する．

この空間は大域的単峰性に乏しいという点以外は探索が困難となるような性質を持たないよう設計されている．したがって NF1 を用いることで，大域的単峰性が乏しいだけで最適化性能が大きく低下する手法との比較が容易となると考えられる．

Noisy Function 2　NF2 は，入力値に対して不規則な過敏性を持ち，かつ大域的単峰性のある問題を想定したベンチマーク関数であり，この外観を図 2.4 に示す．具体的な生成方法は次の通りである．

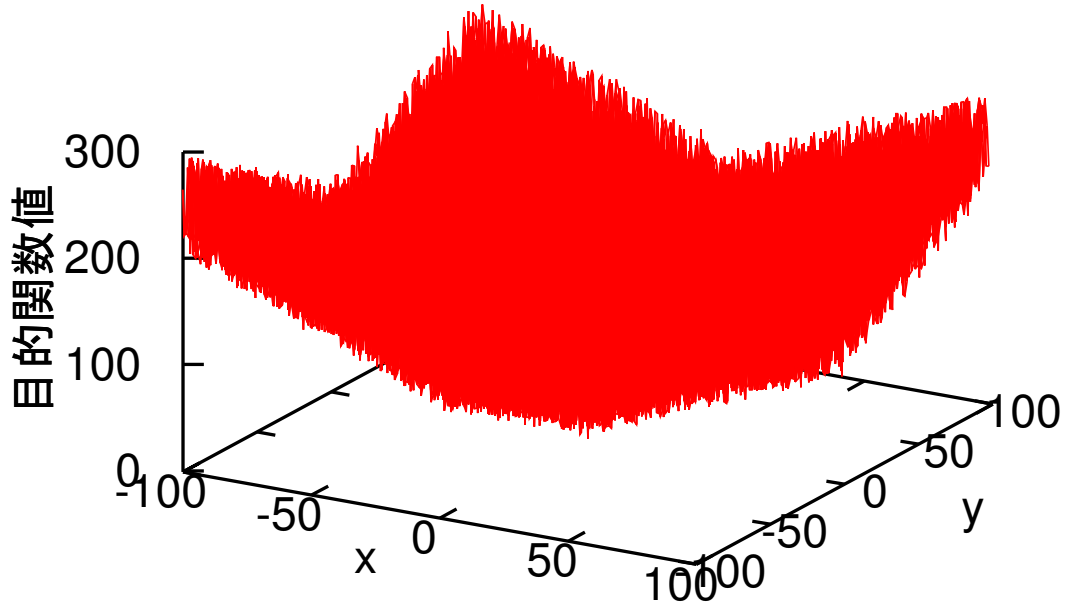


図 2.4: Noisy Function 2 の外観

- 定義域は $[-100, 100]^2$ とする．
- 整数格子点上の目的関数の値を，$x + y + R_{\mathcal{U}}(\mathbb{R}, [0, 100])$ を用いて配置する．ただし，$f_{\mathrm{NF2}}(0, 0) = -1$ とし，これを最小値とする．
- 整数格子点以外の座標 (x, y) の目的関数の値は，$(\lfloor x \rfloor, \lfloor y \rfloor)$，$(\lfloor x \rfloor + 1, \lfloor y \rfloor)$，$(\lfloor x \rfloor, \lfloor y \rfloor + 1)$，$(\lfloor x \rfloor + 1, \lfloor y \rfloor + 1)$ を用いた 4 点バイリニア法により決定する．

この空間を最適化した結果とベンチマーク関数，特に Rastrigin 関数の最適化結果を比較ことで，大域的単峰性がありかつ局所解が規則正しく並んでいる探索空間にのみ有効な手法を区別することができると考えられる．

2.5.2　比較手法

本実験では，比較手法として以下の手法を用いる．

- Simulated Annealing（SA） [13]
- Simulated Annealing with Advanced Adaptive Neighborhood（SA/AAN） [36]

- Nelder-Mead Simplex Method（NMS） [15]
- Harmony Search（HS） [37]
- 交叉方法に一様交叉，選択方式にエリート選択を用いた Genetic Algorithm（GA(Uni+Elite)）
- 交叉方法に単峰性正規分布交叉 [38]，選択方式に Minimal Generation Gap [39] を用いた Genetic Algorithm（GA(UNDX+MGG)）
- 交叉方法にブレンド交叉 [40]，選択方式に Minimal Generation Gap を用いた Genetic Algorithm (GA(BLX-α+MGG))
- $(1+1)$-Evolution Strategy（$(1+1)$-ES） [18]
- $(\mu+\lambda)$-Evolution Strategy（$(\mu+\lambda)$-ES） [18]
- Differential Evolution（DE） [19]
- DE-SP の突然変異個体生成部の改良のみ導入した DE (DE-SP($M=0$))
- DE-SP の改悪個体受理部のみ導入した DE (DE2)
- DE/nrand/1 [32]
- DE/isolated/1 [33]
- Hybridizing Particle Swarm Optimization with Differential Evolution (PSO-DE) [34]
- Wavelet-Mutation-Wavelet-Crossover-Based Differential Evolution (WMWC-DE) [35]
- Particle Swarm Optimization（PSO） [16]
- Krohling らによるパラメータを設定する必要のない Particle Swarm Optimization（PSO-KS） [41]
- Distributed Hierarchical Particle Swarm Optimization（DH-PSO） [42]
- Invasive Weed Optimization (IWO) [20]
- Group Search Optimizer (GSO) [21]
- Artificial Bee Colony Optimization (ABC) [22]
- Random Search（RS）
- Random Walk として Lévy Flight [43] を用いた手法 (LF)
- Cuckoo Search (CS) [23]

次に，それぞれの手法について概要を説明する．ただし，DE は 2.3.1 節にて説明済みであり，DE-SP($M=0$) と DE2 は上記通りの手法であるため本節での説明は省略する．

アルゴリズム 3 Simulated Annealing

$\boldsymbol{m}_p \cdots$ 近傍幅
$T \cdots$ 温度パラメータ
$\lambda \cdots$ 冷却パラメータ

解 $\boldsymbol{\chi}$ を乱数で初期化
while 終了条件が未成立 **do**
　摂動解 $\boldsymbol{\chi}'$ の生成: $\chi'_k = \chi_k + R_{\mathcal{U}}(\mathbb{R}, [-1, 1])m_{p_k}\ (k = 1, 2, \ldots D)$
　if $E(\boldsymbol{\chi}') < E(\boldsymbol{\chi})$ **or** $(E(\boldsymbol{\chi}') \geq E(\boldsymbol{\chi})$ **and** $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) \leq \exp\left(-\frac{E(\boldsymbol{\chi}')-E(\boldsymbol{\chi})}{T}\right))$ **then**
　　$\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}'$
　else
　　$\boldsymbol{\chi}$ を次の摂動に引き継ぎ
　end if
　冷却: $T \leftarrow \lambda T$
end while

Simulated Annealing

Simulated Annealing（SA）とは，鉄の焼き鈍し過程を参考に，Kirkpatrick ら [13] により考案された確率的最適化手法の一つある．SA は単点探索手法であり，1 つの点が，始めは大きく動きながら探索を行い，徐々に動きを小さくしていくことで，最適解が存在する領域の周辺を詳細に探索する．

具体的な手順は以下の通りである．また，擬似コードをアルゴリズム 3 に示す．

1. 摂動解生成 : 現在の解（暫定解）$\boldsymbol{\chi}$ から一定の範囲（近傍）内より確率的に次の解候補（摂動解）$\boldsymbol{\chi}'$ を生成する．本研究では，摂動解を近傍内から一様乱数によって選択する方式を採用する．

$$\chi'_k = \chi_k + R_{\mathcal{U}}(\mathbb{R}, [-1, 1])m_{p_k} \quad (k = 1, 2, \ldots, D) \tag{2.10}$$

　ここで，D は問題の次元数，$R_{\mathcal{U}}(\mathbb{A}, [a, b])$ は空間 $\mathbb{A}$ における範囲 $[a, b]$ の一様乱数，$\boldsymbol{m}_p$ は近傍幅である．

2. 受理判定 : $E(\boldsymbol{\chi}') < E(\boldsymbol{\chi})$ であれば $\boldsymbol{\chi}'$ を受理し，$\boldsymbol{\chi}$ と置き換える．ここで，E は目的関数である．そうでなければ，ある確率 P で $\boldsymbol{\chi}'$ を受理し，$\boldsymbol{\chi}$ と置き換える．本研究では P として，次式で表される Metropolis の基準 [44] を採用する．

$$P = \exp\left(-\frac{E(\boldsymbol{\chi}') - E(\boldsymbol{\chi})}{T}\right) \tag{2.11}$$

　ここで，T は温度パラメータである．

3. 冷却 : 受理判定での P の生成に用いる温度パラメータを減少させる．すなわち，$\boldsymbol{\chi}'$ が $\boldsymbol{\chi}$ よりも悪かった場合における $\boldsymbol{\chi}'$ の受理確率を低下させる．終了条件を満たしていない場合，手順 1. へ戻る．この過程において，真の最適解への漸近収束性を

アルゴリズム 4 Simulated Annealing with Advanced Adaptive Neighborhood

$\boldsymbol{m}_p \cdots$ 近傍幅
$T \cdots$ 温度パラメータ
$\lambda \cdots$ 冷却パラメータ
$\Delta \cdots$ 近傍調整係数を出力する関数
$h_0 \cdots$ 近傍拡大率
$\Delta' \cdots$ 近傍拡大率調整係数を出力する関数
$p_{\max}, p_{\min} \cdots$ 目標受理率の上下限
$S_{\boldsymbol{m}_p}, S_{h_0} \cdots$ 受理率を導出するために現在の摂動から遡る摂動数
$p \cdots$ 過去 $S_{\boldsymbol{m}_p}$ 摂動での近傍解受理率
$p' \cdots$ 過去 S_{h_0} 摂動での近傍解受理率

解 $\boldsymbol{\chi}$ を乱数で初期化
while 終了条件が未成立 **do**
　摂動解 $\boldsymbol{\chi}'$ の生成: $\chi'_k = \chi_k + R_{\mathcal{U}}(\mathbb{R}, [-1, 1]) m_{p_k}$ $(k = 1, 2, \ldots D)$
　if $E(\boldsymbol{\chi}') < E(\boldsymbol{\chi})$ **or** $(E(\boldsymbol{\chi}') \geq E(\boldsymbol{\chi})$ **and** $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) \leq \exp\left(-\frac{E(\boldsymbol{\chi}') - E(\boldsymbol{\chi})}{T}\right))$ **then**
　　$\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}'$
　else
　　$\boldsymbol{\chi}$ を次の摂動に引き継ぎ
　end if
　冷却: $T \leftarrow \lambda T$
　if 現在の摂動回数が $S_{\boldsymbol{m}_p}$ の倍数 **then**
　　近傍幅の更新: $\boldsymbol{m}_p \leftarrow \Delta(p)\boldsymbol{m}_p$
　end if
　if 現在の摂動回数が S_{h_0} の倍数 **then**
　　近傍拡大率の更新: $h_0 \leftarrow \Delta'(p')h_0$
　end if
end while

保障する場合，次式か，あるいはそれより遅い速度で冷却が行われる．これを対数アニーリングという．

$$T \leftarrow \frac{T}{\log n} \tag{2.12}$$

しかし，この対数アニーリングでは解の収束までに非常に時間が掛かる．そのため，本研究では次式で示す指数アニーリングを用いる．指数アニーリングでは得られた解が最適解である保障はされないものの，準最適な値を探索できる．

$$T \leftarrow \lambda T \quad (0 < \lambda < 1) \tag{2.13}$$

Simulated Annealing with Advanced Adaptive Neighborhood

Simulated Annealing with Advanced Adaptive Neighborhood とは，Miki ら [36] によって提案された，動的に近傍幅 m_p を調整することで局所解からの回避を試みた SA である．擬似コードをアルゴリズム 4 に示す．探索方法は SA と同一であるが，予め定めた $S_{\boldsymbol{m}_p}$ 摂動ごとに近傍幅 $\boldsymbol{m}_p$ の更新を行う．

$$\boldsymbol{m}_p \leftarrow \Delta(p)\boldsymbol{m}_p \tag{2.14}$$

ここで，Δ は近傍の調整を行う係数を決定する関数，p は解の受理率であり，それぞれ次のように表される．

$$\Delta(p) = \begin{cases} h_0 & (p > p_{\max}) \\ 0.5 & (p < p_{\min}) \\ 1 & (\text{otherwise}) \end{cases} \tag{2.15}$$

$$p = \frac{s}{S_{\boldsymbol{m}_p}} \tag{2.16}$$

h_0 は近傍の拡大率であり，$p_{\max}$，$p_{\min}$ はそれぞれ受理率目標値の上限と下限，s は過去 $S_{\boldsymbol{m}_p}$ 摂動で解が受理された回数である．また，h_0 は予め定められた S_{h_0} 摂動ごとに更新を行う．

$$h_0 \leftarrow \Delta'(p')h_0 \tag{2.17}$$

ここで，Δ' は h_0 の調整を行う係数を決定する関数，p' は解の受理率であり，それぞれ次のように表される．

$$\Delta'(p') = \begin{cases} 2 & (p' > p_{\max}) \\ 0.5 & (p' < p_{\min}) \\ 1 & (\text{otherwise}) \end{cases} \tag{2.18}$$

$$p' = \frac{s'}{S_{h_0}} \tag{2.19}$$

s' は過去 S_{h_0} 摂動で解が受理された回数である．

Nelder-Mead Simplex Method

Nelder-Mead Simplex Method（NMS）とは，Spendley らによって提案された Simplex 法 [14] が，Nelder らにより改良された [15] ものであり，確率的最適化手法の一つである．この手法は多点探索手法であり，まず，探索空間に複数の頂点を設けることで Simplex（幾何学的な図形）を形成する．そして Simplex の各頂点の中で,「目的関数の値が最悪になる頂点の，この頂点以外の頂点の幾何的重心に関する鏡映点では，目的関数の値が改善されるだろう」，という仮定に基づき，反射・拡大・縮小・収縮と呼ばれる動作を適時行い頂点を移動させることで探索を行い，最適解を発見する手法である．

これらの反射・拡大・縮小・収縮という四動作が NMS における基本動作となっている．次にこれらの基本動作内容を説明する．ただし，以後の説明において $\boldsymbol{\chi}^h$ は目的関数 E を最大とする頂点（最悪な頂点），$\boldsymbol{\chi}^s$ は E を 2 番目に大きい値とする頂点，$\boldsymbol{\chi}^l$ は E を最小とする頂点（最良な頂点），$\boldsymbol{\chi}^g$ は $i \neq h$ である全ての頂点 $\boldsymbol{\chi}^i$ の幾何的重心 $\left(\frac{1}{n-1}\sum_{i \neq h}\boldsymbol{\chi}^i\right)$ とする．また，n は頂点の数である．

反射 $\boldsymbol{\chi}^g$ を中心に，$\boldsymbol{\chi}^h$ と対称の位置にある $\boldsymbol{\chi}^r$ を生成する．

$$\boldsymbol{\chi}^r = (1+\alpha)\boldsymbol{\chi}^g - \alpha\boldsymbol{\chi}^h \tag{2.20}$$

アルゴリズム 5 Nelder-Mead Simplex Method

```
D ··· 次元数
N ··· 頂点数

for i = 1 から N do
    頂点 χ^i を乱数で初期化
end for
while 終了条件が未成立 do
    反射
    if E(χ^r) < E(χ^l) then
        拡大
        if E(χ^e) < E(χ^r) then
            χ^h ← χ^e
        else
            χ^h ← χ^r
        end if
    else
        if E(χ^s) < E(χ^r) < E(χ^h) or E(χ^h) ≤ E(χ^r) then
            if E(χ^s) < E(χ^r) < E(χ^h) then
                χ^h ← χ^r
            end if
            縮小
            if E(χ^c) < E(χ^h) then
                χ^h ← χ^c
            else
                収縮
            end if
        else
            χ^h ← χ^r
        end if
    end if
end while
```

ここで，α は反射係数である．

拡大 $\boldsymbol{\chi}^g$ から $\boldsymbol{\chi}^r$ の方向へ $\boldsymbol{\chi}^r$ を延長し，$\boldsymbol{\chi}^e$ を生成する．

$$\boldsymbol{\chi}^e = \beta\boldsymbol{\chi}^r + (1-\beta)\boldsymbol{\chi}^g \tag{2.21}$$

ここで，β は拡大係数である．

縮小 $\boldsymbol{\chi}^g$ から $\boldsymbol{\chi}^h$ の方向へ進んだ点 $\boldsymbol{\chi}^c$ を生成する．

$$\boldsymbol{\chi}^c = \gamma\boldsymbol{\chi}^h + (1-\gamma)\boldsymbol{\chi}^g \tag{2.22}$$

ここで，γ は縮小係数である．

収縮 Simplex 全体を $\boldsymbol{\chi}^l$ の方向へ半分に収縮する．

$$\boldsymbol{\chi}^l = \frac{\boldsymbol{\chi}^l + \boldsymbol{\chi}^i}{2} \quad (i = 1, 2, \ldots, n) \tag{2.23}$$

これらのパラメータは，実験的に，問題に依存せず $\alpha = 1$，$\beta = 2$，$\gamma = \frac{1}{2}$ が有効であると確認されている [15, 45].

次に，NMS の具体的な手順は次の通りである．また，擬似コードをアルゴリズム 5 に示す.

1. 初期の頂点群を作成し，各頂点で目的関数 E の評価を行う．ここで，E の次元が k である場合，頂点数 n は k より大きくなければならない.
2. 各 $E(\boldsymbol{\chi})$ の大小関係から $\boldsymbol{\chi}^h$，$\boldsymbol{\chi}^s$，$\boldsymbol{\chi}^l$ を探し，終了判定を行う．終了しないならば，$\boldsymbol{\chi}^h$ を除いた全頂点の重心 $\boldsymbol{\chi}^g$ を求める.
3. **反射**により $\boldsymbol{\chi}^r$ を求め，$E(\boldsymbol{\chi}^r)$ を評価する.
4. $E(\boldsymbol{\chi}^r) < E(\boldsymbol{\chi}^l)$ ならば，**拡大**により $\boldsymbol{\chi}^e$ を求め，$E(\boldsymbol{\chi}^e)$ の評価を行う.
   - $E(\boldsymbol{\chi}^e) < E(\boldsymbol{\chi}^r)$ ならば，$\boldsymbol{\chi}^h$ を $\boldsymbol{\chi}^e$ に置き換えて手順 2. へ戻る.
   - $E(\boldsymbol{\chi}^e) \geq E(\boldsymbol{\chi}^r)$ ならば，$\boldsymbol{\chi}^h$ を $\boldsymbol{\chi}^r$ に置き換えて手順 2. へ戻る.
5. $E(\boldsymbol{\chi}^s) < E(\boldsymbol{\chi}^r) < E(\boldsymbol{\chi}^h)$ ならば，$\boldsymbol{\chi}^h$ を $\boldsymbol{\chi}^r$ に置き換え，**縮小**により $\boldsymbol{\chi}^c$ を求め，$E(\boldsymbol{\chi}^c)$ の評価を行う.
   - $E(\boldsymbol{\chi}^c) < E(\boldsymbol{\chi}^h)$ ならば，$\boldsymbol{\chi}^h$ を $\boldsymbol{\chi}^c$ に置き換えて手順 2. へ戻る.
   - そうでなければ，**収縮**を行い，新たに生成された全ての頂点に対して $E(\boldsymbol{\chi})$ を評価し，手順 2. へ戻る.
6. $E(\boldsymbol{\chi}^h) \leq E(\boldsymbol{\chi}^r)$ ならば，**縮小**により $\boldsymbol{\chi}^c$ を求め，$E(\boldsymbol{\chi}^c)$ の評価を行う.
   - $E(\boldsymbol{\chi}^c) < E(\boldsymbol{\chi}^h)$ ならば，$\boldsymbol{\chi}^h$ を $\boldsymbol{\chi}^c$ に置き換えて手順 2. へ戻る.
   - そうでなければ，**収縮**を行い，新たに生成された全ての頂点に対して $E(\boldsymbol{\chi})$ を評価し，手順 2. へ戻る.
7. 手順 4.，5.，6. の全てに当てはまらないならば，$\boldsymbol{\chi}^h$ を $\boldsymbol{\chi}^r$ に置き換えて手順 2. へ戻る.

本研究において，NMS の終了条件は次式を用いる [46].

$$\frac{2|E(\boldsymbol{\chi}^h) - E(\boldsymbol{\chi}^l)|}{|E(\boldsymbol{\chi}^h)| + |E(\boldsymbol{\chi}^l)|} < \delta \tag{2.24}$$

ここで，δ は終了条件のための閾値である.

アルゴリズム 6 Harmony Search

$D \cdots$ 次元数
$N \cdots$ 個体数
$\delta_{\mathrm{HS}} \cdots$ 近傍幅
$P_{\mathrm{hmcr}}, P_{\mathrm{par}} \cdots$ 確率パラメータ

for $i = 1$ から N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
end for
while 終了条件が未成立 **do**
　子個体候補 $\boldsymbol{\chi}'$ を生成: $\chi'_k = \begin{cases} \chi_k^{R_{\mathcal{U}}(\mathbb{Z},[1,N])} & (P_{\mathrm{hmcr}}\text{の確率}) \\ \chi_k^{R_{\mathcal{U}}(\mathbb{R},\text{ 定義域})} & (\text{それ以外}) \end{cases} (k = 1, 2, \ldots D)$
　子個体候補の修正: $\chi'_k \leftarrow \begin{cases} \chi'_k + \delta_{\mathrm{HS}} R_{\mathcal{U}}(\mathbb{R}, [-1, 1]) & (P_{\mathrm{par}}\text{の確率}) \\ \chi'_k & (\text{それ以外}) \end{cases} (k = 1, 2, \ldots D)$
　if $\boldsymbol{\chi}'$ が HM 内の最悪解 $\boldsymbol{\chi}^{\mathrm{worst}}$ より悪い **then**
　　$\boldsymbol{\chi}^{\mathrm{worst}} \leftarrow \boldsymbol{\chi}'$
　end if
end while

Harmony Search

Harmony Search（HS）は，音楽演奏者が即興で旋律を創る様を参考に，Geem ら [37] により提案された確率的最適化手法である．HS は多点探索手法であり，各点は Harmony Memory（HM）に収められ，この HM 全体を用いることで新たな点を生成し，探索を行う手法である．HS は離散値の探索空間と連続値の探索空間のどちらも最適化ができるよう設計されているが，本研究では実数値の探索空間を前提とする．

HM のサイズを N，探索空間の次元を D とすると，HS の具体的手順は次の通りである．また，擬似コードをアルゴリズム 6 に示す．

1. HM 内の各点 $\boldsymbol{\chi}^i$ $(i = 1, 2, \ldots, N)$ を一様乱数で初期化する．
2. 終了条件を満たしていたら探索を終了する．そうでなければ，新たな探索点 $\boldsymbol{\chi}'$ の各要素 χ'_k $(k = 1, 2, \ldots D)$ をそれぞれ次の様に生成する．
   (a) ステップ 1
   - P_{hmcr} の確率で，$\chi'_k = \chi_k^{R_{\mathcal{U}}(\mathbb{Z},[1,N])}$ とする．ここで，P_{hmcr} は Harmony Memory Considering Rate であり，予め与える必要のある定数である．
   - それ以外の確率（$1 - P_{\mathrm{hmcr}}$ の確率）で，$\chi'_k = R_{\mathcal{U}}(\mathbb{R}, [\chi_{k_{\min}}, \chi_{k_{\max}}])$ ここで，$\chi_{k_{\min}}$ は χ_k のとり得る最小値，$\chi_{k_{\max}}$ は χ_k のとり得る最大値である．

   (b) ステップ 2
   - P_{par} の確率で，$\chi'_k \leftarrow \chi'_k + \delta_{\mathrm{HS}} R_{\mathcal{U}}(\mathbb{R}, [-1, 1])$ と χ'_k を修正する．ここで，P_{par} は，A Pitch Adjusting Rate であり，予め与える必要のある定数である．ここで，δ_{HS} は探索の近傍幅である．

アルゴリズム 7 Genetic Algorithm **or** Evolution Strategy

```
1: N ··· 個体数
2:
3: for i = 1 から N do
4:     個体 χ^i を乱数で初期化
5: end for
6: while 終了条件が未成立 do
7:     突然変異
8:     交叉
9:     生存個体選択
10: end while
```

- それ以外の確率で，χ'_k に修正は行わない．

3. 生成した $\boldsymbol{\chi}'$ が HM 内の最悪解 $\boldsymbol{\chi}^{\mathrm{worst}}$ よりよい解であれば，$\boldsymbol{\chi}^{\mathrm{worst}}$ を $\boldsymbol{\chi}'$ で置き換える．

4. 手順 2. へ戻る．

Genetic Algorithm

Genetic Algorithm（GA）とは，遺伝子が突然変異と交叉を繰り返し行い，その中でより環境に適応した遺伝子が多く残る様子から，Holland により発想 [17] され，発展されてきた進化計算の一体系であり，具体的実装として様々な形式が存在する．ほとんどの手法は多点探索であり，突然変異，交叉，生存選択により探索を行う．GA ではこれら三要素の具体的実装について，様々な模索，検討がなされている．GA の擬似抽象コードをアルゴリズム 7 に示す．これは，ES の擬似抽象コードと同一になる．

本研究では，交叉方法に一様交叉，生存選択にエリート選択を採用した GA と，交叉方法に単峰性正規分布交叉（UNDX），生存選択に Minimal Generation Gap（MGG）を採用した GA を用いる．突然変異の方法は共に一様乱数とする．

一様交叉　一様交叉とは，GA における交叉の具体的実装の一つであり，その概念図を図 2.5 に示す．原則として 2 体の親から 1 体または 2 体の子を生成する手法であるが，親の個体数は任意に拡張可能である．子のそれぞれの遺伝子要素は，どちらかの親の要素が引き継がれる．

単峰性正規分布交叉　単峰性正規分布交叉（UNDX）とは，小野ら [38] によって提案された，実数値 GA における交叉の具体的実装の一つである．この手法は，ランダムに選ばれた 3 体の親個体を用いて単峰性正規分布を生成し，この分布と分布に基づく乱数から 2 体の子個体を生成する手法である．生成される分布のイメージを図 2.6 に示す．ま

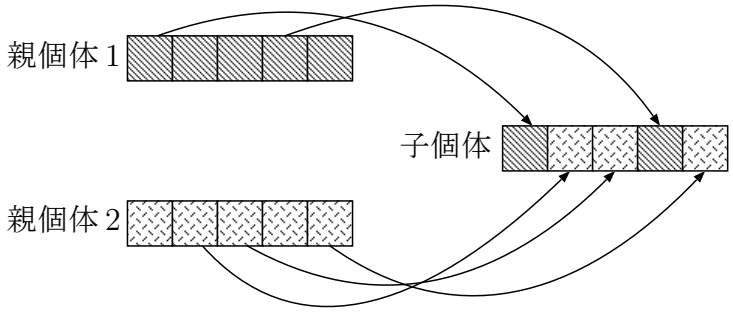


図 2.5: 一様交叉の概念図

た，子個体 $\boldsymbol{c}_1$，$\boldsymbol{c}_2$ 生成式は以下のように表される．

$$\boldsymbol{c}_1 = \frac{\boldsymbol{p}_1+\boldsymbol{p}_2}{2} + R_{\mathcal{N}}(\mathbb{R},0,\sigma_1^2)\boldsymbol{e}_1 + \sum_{k=2}^{D} R_{\mathcal{N}}(\mathbb{R},0,\sigma_2^2)\boldsymbol{e}_k \tag{2.25}$$

$$\boldsymbol{c}_2 = \frac{\boldsymbol{p}_1+\boldsymbol{p}_2}{2} + R_{\mathcal{N}}(\mathbb{R},0,\sigma_1^2)\boldsymbol{e}_1 - \sum_{k=2}^{D} R_{\mathcal{N}}(\mathbb{R},0,\sigma_2^2)\boldsymbol{e}_k \tag{2.26}$$

ここで，D は問題の次元数，$R_{\mathcal{N}}(\mathbb{A},\mu,\sigma^2)$ は空間 $\mathbb{A}$ での平均 μ かつ分散 σ^2 を用いた正規乱数，$\boldsymbol{e}_k\,(k=1,2,\ldots,D)$ は $\boldsymbol{e}_1$ を $\boldsymbol{p}_1-\boldsymbol{p}_2$ の方向に合わせた正規直行基底である．

$$\boldsymbol{e}_1 = \frac{\boldsymbol{p}_2-\boldsymbol{p}_1}{\|\boldsymbol{p}_2-\boldsymbol{p}_1\|} \tag{2.27}$$
$$\forall k \forall l [\boldsymbol{e}_k \perp \boldsymbol{e}_l \mid k \neq l] \quad (k,l=1,2,\ldots D)$$

σ_1，σ_2 は，正規乱数を生成する際に必要な分散情報である．

$$\sigma_1 = \alpha d_1 \tag{2.28}$$

$$\sigma_2 = \frac{\beta d_2}{\sqrt{D}} \tag{2.29}$$

ここで，α，β は予め与える必要のある定数である．また，d_1 は親 $\boldsymbol{p}_1$ と親 $\boldsymbol{p}_2$ との距離，d_2 は親 $\boldsymbol{p}_3$ と直線 $\overline{\boldsymbol{p}_1\boldsymbol{p}_2}$ との距離である．

$$d_1 = \|\boldsymbol{p}_1-\boldsymbol{p}_2\| \tag{2.30}$$

$$d_2 = \sqrt{\|\boldsymbol{p}_1-\boldsymbol{p}_3\|^2 - \left(\frac{\|\boldsymbol{p}_1-\boldsymbol{p}_3\|^2+\|\boldsymbol{p}_1-\boldsymbol{p}_2\|^2-\|\boldsymbol{p}_2-\boldsymbol{p}_3\|^2}{2\|\boldsymbol{p}_1-\boldsymbol{p}_2\|}\right)^2} \tag{2.31}$$

ブレンド交叉　ブレンド交叉（BLX-α）とは，Eshelman [40] によって提案された，実数値 GA における交叉の具体的実装の一つである．この手法では，ランダムに選ばれた 2 体の親個体を対角とする超平面より「一回り大きい」範囲からランダムに 2 体の子個体を生成する．「一回り大きい」の程度は設計変数 α によって定められ，超平面の各辺が $1+2\alpha$ 倍となる．2 次元空間でのこのイメージを図 2.7 に示す．

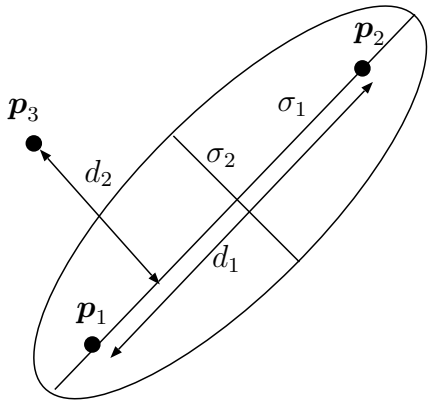


図 2.6: UNDX で生成される分布のイメージ

エリート選択　エリート選択とは，GA における生存戦略の具体的実装の一つである．名前の通り，次世代に残る個体の候補の内，目的関数の値が良いものから順に生存個体が選択される．

Minimal Generation Gap　Minimal Generation Gap（MGG）とは，佐藤ら [39] によって提案された，GA における生存戦略の具体的実装の一つである．この手法は，個体群から 2 個体を親個体として抜き出し，これらから子個体の候補を生成した場合を前提とする．以下の手順に基づき，次世代に残る個体を生成する．

1. 抜き出された親個体から，子個体の候補を生成する．生成数は任意とする．
2. 抜き出された親個体と子個体の候補から，エリート戦略で 1 体選択する．
3. 残った個体の中からルーレット選択によりさらに 1 体選択する．
4. 選択された 2 体を個体群に戻し，次世代の個体群とする．この際，抜き出されなかった個体は次世代に引き継がれる．

この操作は，探索序盤における選択圧をなるべく下げることで初期収束を回避し，ある程度探索が進んだ段階でも個体の多様性を維持し，探索の停滞を抑えることを目的としている．

一摂動当たりの子個体生成数が 2 体で不足である場合は，一摂動内で複数回交叉と MGG による選択を行う場合がある．

Evolution Strategy

Evolution Strategy（ES） [18] とは，動植物が進化によって環境に適応していく様子から，Rechenberg らによって発想，発展された進化計算の一体系であり，具体的実装と

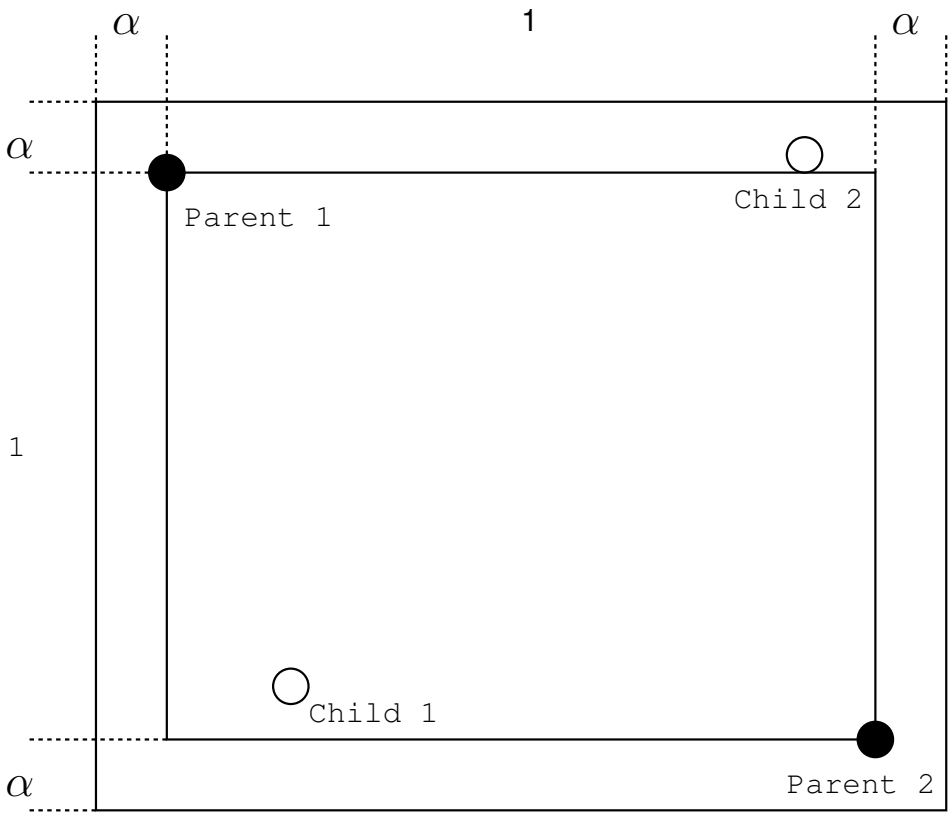


図 2.7: 2 次元上での BLX-α のイメージ

して様々な形式が存在する．ほとんどの手法は多点探索であり，突然変異，交叉，生存選択により探索を行う．ES の擬似抽象コードをアルゴリズム 7 に示す．これは GA の擬似抽象コードと同一になる．

ES の具体的実装の大半では，主に突然変異の方法を工夫し，より効率的な探索を行おうと試みられている．本研究ではその内，基本的な実装の例である (1+1)-ES，(μ+λ)-ES を用いる．

(1 + 1)-ES　(1 + 1)-ES は，ES の中で最も基本的で初歩的な探索手法である．ほとんどの ES の実装例は多点探索であるが，(1 + 1)-ES は単点探索であり，近傍探索と近い挙動を示す．探索は以下の手順によって行われる．

1. 個体 $\boldsymbol{\chi}$ 一様乱数で初期化し，標準偏差 σ の初期値を設定する．
2. 終了条件を満たしていたら探索を終了する．
3. 現在の個体 $\boldsymbol{\chi}$ の各要素 $\chi_k\,(k = 1, 2, \ldots, D)$ に次の操作を行い，子個体の候補 $\boldsymbol{\chi}'$ を生成する．

$$\chi'_k = \chi_k + R_{\mathcal{N}}(\mathbb{R}, 0, \sigma^2) \tag{2.32}$$

ここで，σ は $\dfrac{1}{5}$ ルールで決定される標準偏差である．

4. $E(\boldsymbol{\chi}') < E(\boldsymbol{\chi})$ ならば $\boldsymbol{\chi}'$ を，そうでなければ $\boldsymbol{\chi}$ を受理し，もう一方を破棄する．
5. 探索回数が n の倍数となったら，$\frac{1}{5}$ ルールに基づいて σ を更新する．n は予め与える必要のある定数である．
6. 手順 2. に戻る．

$\frac{1}{5}$ ルール　$(1+1)$-ES において，σ の値は探索性能に大きな影響を与えるため，これを適切に定めることが必要となる．Schwefel はこの問題に対し，次の様な規則に基づいて σ を更新する手法を提案した [18].

n 回探索が進むごとに，過去 $10n$ 回の探索における遷移の成功確率，すなわち $\boldsymbol{\chi}'$ が残された確率を確認し，成功確率が $\frac{1}{5}$ 未満ならば $0 < c_{\frac{1}{5}} < 1$ である定数 $c_{\frac{1}{5}}$ を σ にかける．逆に，成功確率が $\frac{1}{5}$ 以上ならば，$\frac{1}{c_{\frac{1}{5}}}$ を σ にかける．これは，最適な σ の値を設定すると，多くの問題において成功確率が 0.2 となる経験的事実に基づいている．

$(\mu+\lambda)$-ES　基本的な ES の一実装である．この手法は，μ 体の個体から λ 体の個体を生成し，両個体集団からエリート戦略で μ 個体を選び，次の摂動の個体群とする手法である．探索は以下の手順に従って行われる．

1. 各個体 $\boldsymbol{\chi}^i$ $(i = 1, 2, \ldots, \mu)$ を一様乱数で初期化する．また，各個体は正規分布の分散共分散情報を保持し，それぞれ分散情報 σ_k^i $(k = 1, 2, \ldots, D)$，共分散情報 α_{kl}^i $(k = 1, 2, \ldots, D-1,\ l = k, k+1, \ldots, D)$ とする．ここで，D は問題の次元数である．
2. 終了条件を満たしていたら探索を終了する．
3. μ 体の個体からランダムに 1 体選び，次の様に子個体を生成する．この操作を λ 回繰り返す．

$$\sigma_k' = \sigma_k e^{\tau'\xi + \tau\xi_k}\ (k = 1, 2, \ldots, D) \tag{2.33}$$
$$\alpha_{kl}' = \alpha_{kl} + \beta\xi_{kl}\ (k = 1, 2, \ldots, D-1,\ l = k, k+1, \ldots, D) \tag{2.34}$$
$$\boldsymbol{\chi}' = \boldsymbol{\chi} + R_{\mathcal{N}}(\mathbb{R}^D, 0, C) \tag{2.35}$$

ここで，ξ，ξ_k $(k = 1, 2, \ldots, D)$，ξ_{kl} $(k = 1, 2, \ldots, D-1,\ l = k, k+1, \ldots, D)$ はそれぞれ独立して生成された正規乱数である．

$$\xi = R_{\mathcal{N}}(\mathbb{R}, 0, 1) \tag{2.36}$$
$$\xi_k = R_{\mathcal{N}}(\mathbb{R}, 0, 1) \tag{2.37}$$
$$\xi_{kl} = R_{\mathcal{N}}(\mathbb{R}, 0, 1) \tag{2.38}$$

また C は，σ_k'，α_{kl}' を用いて作られる分散共分散行列，β，τ，τ' は予め与える必要のある定数である．

4. 元あった μ 個体と，生成された λ 個体からエリート戦略を用いて生存する μ 個体を選択する．

5. 手順 2. に戻る．

なお，$(\mu + \lambda)$-ES に必要な定数 β，τ，τ' は Schwefel による以下の推奨値が報告されている [18].

$$\beta = 0.0873 \tag{2.39}$$

$$\tau = \frac{1}{\sqrt{2\sqrt{D}}} \tag{2.40}$$

$$\tau' = \frac{1}{\sqrt{2D}} \tag{2.41}$$

DE/nrand/1

DE/nrand/1 とは，Epitropakis [32] らによって考案された DE の一改良手法である．DE からの改良点はサブ親の選択部分（アルゴリズム 8，12 行目）であり，サブ親 $\boldsymbol{\chi}^a$ をランダムで選択せず，現在のメイン親 $\boldsymbol{\chi}$ の最近傍個体としている．DE/nrand/1 の擬似コードをアルゴリズム 8 に示す．結果として，突然変異個体 $\boldsymbol{\chi}'$ の生成候補は少なく・狭くなり，近傍探索能力が高くなることが期待される．これにより，最適解の近傍が発見された後の収束能力が高いと考えられる．

アルゴリズム 8 DE/nrand/1

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F \in [0, 2] \cdots$ スケーリングパラメータ
$C_R \in [0, 1] \cdots$ 交叉時に用いる閾値

for $i = 1$ to N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
end for
while 終了条件が未成立 **do**
　for $i = 1$ to N **do**
　　サブ親 $\boldsymbol{\chi}^a$ を $\boldsymbol{\chi}^i$ の最近傍個体とする
　　サブ親 $\boldsymbol{\chi}^b$, $\boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ をランダムに選択
　　突然変異個体を計算: $\boldsymbol{\chi}'^i \leftarrow \boldsymbol{\chi}^a + F(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c)$
　　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
　　for $k = 1$ to D **do**
　　　if $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < C_R$ **or** $k = l^i$ **then**
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k'^i$
　　　else
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k^i$
　　　end if
　　end for
　　目的関数 $E(\boldsymbol{p}^i)$ を計算
　end for
　for $i = 1$ to N **do**
　　if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
　　　親個体を子個体と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
　　else
　　　親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
　　end if
　end for
end while

アルゴリズム 9 DE/isolated/1

1: $D \cdots$ 探索空間の次元数
2: $N \cdots$ 個体数
3: $F \in [0, 2] \cdots$ スケーリングパラメータ
4: $C_R \in [0, 1] \cdots$ 交叉時に用いる閾値
5: $n_w \cdots$ 子個体を受理できなかった回数
6: $N_w \cdots$ 子個体を受理できなかった回数を保持する変数の閾値
7: $N_d \cdots$ サブ親選択に用いるメイン親の近傍個体数
8:
9: **for** $i = 1$ to N **do**
10:　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
11:　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
12: **end for**
13: **while** 終了条件が未成立 **do**
14:　**for** $i = 1$ to N **do**
15:　　**if** $n_w < N_w$ **then**
16:　　サブ親 $\boldsymbol{\chi}^a$ $(i \neq a)$ を最近傍個体との距離が最も大きい個体とする
17:　　サブ親 $\boldsymbol{\chi}^b$, $\boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ を $\boldsymbol{\chi}^i$ の N_d 近傍個体からランダムに選択
18:　　**else**
19:　　サブ親 $\boldsymbol{\chi}^a$ $(i \neq a)$ をランダムに選択
20:　　サブ親 $\boldsymbol{\chi}^b$, $\boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ をランダムに選択
21:　　$n_w \leftarrow 0$
22:　　**end if**
23:　　突然変異個体を計算: $\boldsymbol{\chi}'^i \leftarrow \boldsymbol{\chi}^a + F(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c)$
24:　　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
25:　　**for** $k = 1$ to D **do**
26:　　**if** $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < C_R$ **or** $k = l^i$ **then**
27:　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k'^i$
28:　　**else**
29:　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k^i$
30:　　**end if**
31:　　**end for**
32:　　目的関数 $E(\boldsymbol{p}^i)$ を計算
33:　　**if** $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
34:　　親個体を子個体と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
35:　　**else**
36:　　親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
37:　　$n_w \leftarrow n_w + 1$
38:　　**end if**
39:　**end for**
40: **end while**

DE/isolated/1

DE/isolated/1 とは，Otani ら [33] によって提案された DE の一改良手法である．この手法では，DE と比較してサブ親の選択（アルゴリズム 9，15 行目から 22）と，子個体を更新するタイミング（アルゴリズム 9，33 行目から 38 行目）に主な改良点がある．一

つ目は，サブ親 $\boldsymbol{\chi}^a$ を最近傍個体との距離が最も大きな個体，即ち「はぐれ個体」とすることである．これにより，個体の多様性が維持されることが期待される．そして，サブ親 $\boldsymbol{\chi}^b, \boldsymbol{\chi}^c$ を全体からではなく $\boldsymbol{\chi}^i$ の N_d 近傍個体からランダムに選択する．これは，一つ目の改良点で求めている「はぐれ個体」周辺に突然変異個体 $\boldsymbol{\chi}'$ が生成されやすくするための工夫である．この工夫は一つ目の改良点による多様性維持を補助する目的で設計されている．ただし，このような操作を続けていると最適解への収束性能が悪化する場合が存在するため，子個体が受理できなかった回数 n_w が閾値 N_w を越えた時のみ通常の DE と同様に突然変異個体を生成する．この時 n_w は 0 に初期化される．二つ目は，子個体を生成した際にすぐさま親個体へ更新を行うことで常に探索結果の知見を反映し，探索が初期収束に陥ることの回避を目的としている．

アルゴリズム 10 Hybridizing Particle Swarm Optimization with Differential Evolution

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$G \cdots$ 制約条件を表す関数
$F \in [0, 2] \cdots$ スケーリングパラメータ
$C_R \in [0, 1] \cdots$ 交叉時に用いる閾値
$\chi_{k_{\max}}, \chi_{k_{\min}}\ (k = 1, 2, \ldots, D) \cdots$ 定義域の上下限
$\mathcal{D}(\boldsymbol{a}, <, \boldsymbol{b}) \cdots$ Deb's Feasibility Rules により $\boldsymbol{a} < \boldsymbol{b}$ が正しいか判定する関数

for $i = 1$ から N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　速度 $\boldsymbol{v}^i$ を乱数で初期化
end for
while 終了条件が未成立 **do**
　個体を $G(\boldsymbol{\chi})$ の降順でソート
　for $i = 1$ から $\left\lfloor \dfrac{N}{2} \right\rfloor$ **do**
　　個体位置更新: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{\chi}^i + \boldsymbol{v}^i$
　　個体速度更新: $\boldsymbol{v}^i \leftarrow |R_{\mathcal{N}}(\mathbb{R}, 0, 1)|(\hat{\boldsymbol{\chi}}^i - \boldsymbol{\chi}^i) + |R_{\mathcal{N}}(\mathbb{R}, 0, 1)|(\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} - \boldsymbol{\chi}^i)$
　　if $\boldsymbol{\chi}^i$ が定義域を越えている **then**
　　　柔らかに押し戻す: $\chi_k^i \leftarrow \begin{cases} \frac{1}{2}(\chi_{k_{\min}} + \chi_k^i) & (\chi_k^i < \chi_{k_{\min}}) \\ \frac{1}{2}(\chi_{k_{\max}} + \chi_k^i) & (\chi_k^i > \chi_{k_{\max}}) \\ \chi_k^i & (\text{それ以外}) \end{cases}$
　　end if
　　if $\mathcal{D}(\boldsymbol{\chi}^i, <, \hat{\boldsymbol{\chi}}^i)$ **then**
　　　個体最良解更新: $\hat{\boldsymbol{\chi}}^i \leftarrow \boldsymbol{\chi}$
　　end if
　　if $\mathcal{D}(\boldsymbol{\chi}^i, <, \hat{\boldsymbol{\chi}}^{\mathrm{GBEST}})$ **then**
　　　全体最良解更新: $\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} \leftarrow \boldsymbol{\chi}$
　　end if
　end for
　for $i = 1$ から N **do**
　　$\hat{\boldsymbol{\chi}}^i$ をメイン親とし，個体最良解群を用いて DE/rand/1，DE/best/1，DE/rand/2 の規則に基づいて 3 体の突然変異個体を生成:
　　$\boldsymbol{\beta}^1 = \hat{\boldsymbol{\chi}}^a + F(\hat{\boldsymbol{\chi}}^b - \hat{\boldsymbol{\chi}}^c)\ (i \neq a \neq b \neq c)$
　　$\boldsymbol{\beta}^2 = \hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} + F(\hat{\boldsymbol{\chi}}^b - \hat{\boldsymbol{\chi}}^c)\ (i \neq b \neq c)$
　　$\boldsymbol{\beta}^3 = \hat{\boldsymbol{\chi}}^a + F(\hat{\boldsymbol{\chi}}^b - \hat{\boldsymbol{\chi}}^c) + F(\hat{\boldsymbol{\chi}}^d - \hat{\boldsymbol{\chi}}^e)\ (i \neq a \neq b \neq c \neq d \neq e)$
　　交叉により子個体候補の生成: $\beta_k^j \leftarrow \begin{cases} \beta_k^j & (C_R\text{の確率}) \\ \hat{\chi}_k^{\mathrm{GBEST}} & (\text{それ以外}) \end{cases}\ (k = 1, 2, \ldots, D\ j = 1, 2, 3)$
　　for $j = 1$ から 3 **do**
　　　if $\boldsymbol{\beta}^j$ が定義域を越えている **then**
　　　　跳ね返す: $\beta_k^j \leftarrow \begin{cases} 2\chi_{k_{\min}} - \beta_k^j & (\beta_k^j < \chi_{k_{\min}}) \\ 2\chi_{k_{\max}} - \beta_k^j & (\beta_k^j > \chi_{k_{\max}}) \\ \chi_k^j & (\text{それ以外}) \end{cases}$
　　　end if
　　end for
　　$\hat{\boldsymbol{\chi}}^i, \boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \boldsymbol{\beta}^3$ を $\mathcal{D}$ を用いて比較し最も良いもので $\hat{\boldsymbol{\chi}}^i$ を置き換え
　　if $\mathcal{D}(\hat{\boldsymbol{\chi}}^i, <, \hat{\boldsymbol{\chi}}^{\mathrm{GBEST}})$ **then**
　　　全体最良解更新: $\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} \leftarrow \hat{\boldsymbol{\chi}}^i$
　　end if
　end for
end while

Hybridizing Particle Swarm Optimization with Differential Evolution

Hybridizing Particle Swarm Optimization with Differential Evolution（PSO-DE）とは，Liu ら [34] によって提案された，DE と PSO-KS の混合手法である．擬似コードをアルゴリズム 10 に示す．この手法は制約付き連続値最適化問題を前提としており，個体の良し悪しは原則として Deb's Feasibility Rules [47] によって比較される．具体的には次の手順を終了条件を満たすまで繰り返す.

1. 制約条件関数値の降順で個体を並べ，頭半分を PSO-KS に基づいて移動させる．この時，探索空間の定義域を越えた個体は以下の式に基づいて修正される [48].

$$\chi_k^i \leftarrow \begin{cases} \frac{1}{2}(\chi_{k_{\min}} + \chi_k^i) & (\chi_k^i < \chi_{k_{\min}}) \\ \frac{1}{2}(\chi_{k_{\max}} + \chi_k^i) & (\chi_k^i > \chi_{k_{\max}}) \\ \chi_k^i & (\text{それ以外}) \end{cases} \quad (k = 1, 2, \ldots, D) \tag{2.42}$$

   ここで，$\chi_{k_{\max}}$，$\chi_{k_{\min}}$ は定義域の上下限である．

2. 個体最良解を DE に基づいて移動させる．この時使用する DE の種類は，DE/rand/1，DE/best/1，DE/rand/2 とし，これら三つを同時に用いて最も結果の良いものを採用する．この時，探索空間の定義域を越えた個体は以下の式に基づいて修正される [49].

$$\beta_k^j \leftarrow \begin{cases} 2\chi_{k_{\min}} - \beta_k^j & (\beta_k^j < \chi_{k_{\min}}) \\ 2\chi_{k_{\max}} - \beta_k^j & (\beta_k^j > \chi_{k_{\max}}) \\ \chi_k^j & (\text{それ以外}) \end{cases} \quad (k = 1, 2, \ldots, D \;\; j = 1, 2, 3) \tag{2.43}$$

   ここで，$\boldsymbol{\beta}^1$，$\boldsymbol{\beta}^2$，$\boldsymbol{\beta}^3$ はそれぞれ DE/rand/1，DE/best/1，DE/rand/2 によって生成された子個体候補である．

アルゴリズム 11 Wavelet-Mutation-Wavelet-Crossover-Based Differential Evolution

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F \in [0, 2] \cdots$ スケーリングパラメータ
$C_R \in [0, 1] \cdots$ 交叉時に用いる閾値
$\psi \cdots$ Wavelet Function
$a, \sigma \cdots$ 調整パラメータ
$g \cdots a$ の上限

for $i = 1$ to N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
end for
while 終了条件が未成立 **do**
　for $i = 1$ to N **do**
　　サブ親 $\boldsymbol{\chi}^a, \boldsymbol{\chi}^b, \boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ をランダムに選択
　　F を生成: $F = \dfrac{1}{\sqrt{a}} \psi \left(\dfrac{\varphi}{a} \right)$
　　突然変異個体を計算: $\boldsymbol{\chi}'^i \leftarrow \boldsymbol{\chi}^a + F(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c)$
　　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
　　for $k = 1$ to D **do**
　　　if $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < C_R$ **or** $k = l^i$ **then**
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k'^i$
　　　else
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k^i$
　　　end if
　　　子個体候補の調整: $p_k^i = \begin{cases} p_k^i + \sigma(\chi_{k_{\max}} - p_k^i) & (\sigma > 0) \\ p_k^i + \sigma(p_k^i - \chi_{k_{\min}}) & (\sigma \leq 0) \end{cases}$
　　end for
　　目的関数 $E(\boldsymbol{p}^i)$ を計算
　end for
　for $i = 1$ to N **do**
　　if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
　　　親個体を子個体と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
　　else
　　　親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
　　end if
　end for
end while

Wavelet-Mutation-Wavelet-Crossover-Based Differential Evolution

Wavelet-Mutation-Wavelet-Crossover-Based Differential Evolution（WMWC-DE）とは，Lai ら [35] によって提案された DE の一改良手法である．この手法では DE のパラメータである F を，各摂動ごとに乱数を入力とした Wavelet Function（付録 A.2 参照）の出力値として設定している．また，交叉を行った後の結果に対しても Wavelet Function

に基づいた修正を行う．

具体的な手順は以下の通りである．また，擬似コードをアルゴリズム 11 に示す．

1. 初期化：各個体を乱数で初期化する．
2. 終了条件判定：終了条件を満たしていればアルゴリズムを終了する．
3. F の生成：この摂動で用いる F を以下の式に基づいて生成する．

$$F = \frac{1}{\sqrt{a}}\psi\left(\frac{\varphi}{a}\right) \tag{2.44}$$

ここで，ψ は Wavelet Function であり，以下の式により定義される．

$$\psi(x) = e^{-\frac{x^2}{2}}\cos(5x) \tag{2.45}$$

φ は入力変数であり，WMWC-DE では $R_{\mathcal{U}}(\mathbb{R}, [-2.5, 2.5])$ を用いて与えられる．a はスケール調整パラメータであり，以下の式に基づいて計算される．

$$a = e^{-\log_e(g)\left(1-\frac{t}{T}\right)^{\zeta_{wm}}+\log_e(g)} \tag{2.46}$$

ここで，t は現在の摂動回数，T は最大摂動回数，g は a の上限である．この a により $|F|$ が摂動回数を経るごとに小さくなっていくため，探索が進むにつれて近傍探索能力が向上することが期待される．

4. 突然変異：DE と同様に突然変異を行い，突然変異個体 $\boldsymbol{\chi}'^i$ $(i = 1, 2, \ldots, D)$ を生成する．
5. 交叉：DE と同様に交叉を行い，子個体候補 $\boldsymbol{p}^i$ $(i = 1, 2, \ldots, D)$ を生成する．
6. 子個体候補の修正：以下の式により子個体候補 $\boldsymbol{p}$ を修正する．

$$p_k^i = \begin{cases} p_k^i + \sigma(\chi_{k_{\max}} - p_k^i) & (\sigma > 0) \\ p_k^i + \sigma(p_k^i - \chi_{k_{\min}}) & (\sigma \leq 0) \end{cases} \tag{2.47}$$

ここで，$\chi_{k_{\max}}$，$\chi_{k_{\min}}$　$(k = 1, 2, \ldots, D)$ は探索空間の定義域，σ は調整パラメータであり，以下の式に基づいて計算される．

$$\sigma = \frac{1}{\sqrt{a}}\psi\left(\frac{\varphi}{a}\right) \tag{2.48}$$

7. 生存個体選択：$E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ ならば子個体候補 $\boldsymbol{p}^i$ が親個体 $\boldsymbol{\chi}^i$ に上書きされ，$E(\boldsymbol{p}^i) \geq E(\boldsymbol{\chi}^i)$ ならば親個体がそのまま次の摂動に持ち越される．
8. 手順 2 に戻る．

アルゴリズム 12 Particle Swarm Optimization

$N \cdots$ 粒子数
$w, c_1, c_2 \cdots$ 非負の重み定数

for $i = 1$ から N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　速度 $\boldsymbol{v}^i$ を乱数で初期化
end for
while 終了条件が未成立 **do**
　for $i = 1$ から N **do**
　　if $E(\boldsymbol{\chi}^i) < E(\hat{\boldsymbol{\chi}}^i)$ **then**
　　　粒子最良解の更新: $\hat{\boldsymbol{\chi}}^i \leftarrow \boldsymbol{\chi}^i$
　　end if
　　if $E(\boldsymbol{\chi}^i) < E(\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}})$ **then**
　　　全体最良解の更新: $\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} \leftarrow \boldsymbol{\chi}^i$
　　end if
　　粒子位置更新: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{\chi}^i + \boldsymbol{v}^i$
　　粒子の速度更新: $\boldsymbol{v}^i \leftarrow w\boldsymbol{v}^i c_1 R_{\mathcal{U}}(\mathbb{R}, [0,1])(\hat{\boldsymbol{\chi}}^i - \boldsymbol{\chi}^i) + c_2 R_{\mathcal{U}}(\mathbb{R}, [0,1])(\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} - \boldsymbol{\chi}^i)$
　end for
end while

Particle Swarm Optimization

Particle Swarm Optimization（PSO）とは，昆虫や魚などの群れにおいて，一匹が良さそうな経路（食料のある経路や外敵のいない経路）を発見すると，群れの残りはどこにいても素早くそれに倣う様を参考に，Eberhart ら [16] によって考案された，群知能に基づいた確率的最適化手法の一つであり，多点探索手法である．本研究では，基本的実装とされている GBEST モデル [16] を採用する．これは，群れの一匹一匹を粒子とみなし，各粒子の知っている粒子最良解と，全粒子の知識を統合して得られる全体最良解を用いて，各粒子の位置と速度を更新することで最適解を発見する手法である．

具体的な手順は以下の通りである．また，擬似コードをアルゴリズム 12 に示す．

1. 各粒子について位置 $\boldsymbol{\chi}$ と速度 $\boldsymbol{v}$ を初期化する．
2. 各粒子の粒子最良解 $\hat{\boldsymbol{\chi}}$ について $E(\boldsymbol{\chi}) < E(\hat{\boldsymbol{\chi}})$ ならば $\hat{\boldsymbol{\chi}}$ を $\boldsymbol{\chi}$ で更新し，全体最良解 $\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}}$ について $E(\boldsymbol{\chi}) < E(\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}})$ ならば $\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}}$ を $\boldsymbol{\chi}$ で更新する．このとき，終了条件を満たしていたら終了する．
3. 各粒子の位置 $\boldsymbol{\chi}$ と速度 $\boldsymbol{v}$ を次のように更新し，手順 2. へ戻る．

$$\boldsymbol{\chi} \leftarrow \boldsymbol{\chi} + \boldsymbol{v} \tag{2.49}$$
$$\boldsymbol{v} \leftarrow w\boldsymbol{v} + c_1 R_{\mathcal{U}}(\mathbb{R}, [0,1])(\hat{\boldsymbol{\chi}} - \boldsymbol{\chi}) + c_2 R_{\mathcal{U}}(\mathbb{R}, [0,1])(\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} - \boldsymbol{\chi}) \tag{2.50}$$

ここで，w，c_1，c_2 は非負の重み係数であり，手動にて設定が必要なパラメータである．

Krohling らによる Particle Swarm Optimization の改良

PSO を用いるためには w，c_1，c_2 の三つのパラメータを設定する必要があるが，これらのパラメータを前知識なしで設定することは容易ではない．そこで Krohling らは統計的に有効である可能性の高いパラメータを推測し，粒子速度の更新式を以下のように改変した．

$$\boldsymbol{v}^i = |R_{\mathcal{N}}(\mathbb{R}, 0, 1)|(\hat{\boldsymbol{\chi}}^i - \boldsymbol{\chi}^i) + |R_{\mathcal{N}}(\mathbb{R}, 0, 1)|(\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} - \boldsymbol{\chi}^i) \tag{2.51}$$

アルゴリズム 12 と比較して，この更新以外の改変はないため，擬似コードは省略する．

アルゴリズム 13 Distributed Hierarchical Particle Swarm Optimization

$H \cdots$ 階層数
$d \cdots$ 分岐数
$N_i \cdots$ 階層 i にいる群れの個体数

for $i = 1$ から H **do**
  for $j = 1$ から d **do**
    for $k = 1$ から N_i **do**
      粒子 ${}^{A^i_j}\boldsymbol{\chi}^k$ を乱数で初期化
      速度 ${}^{A^i_j}\boldsymbol{v}^k$ を乱数で初期化
    end for
  end for
end for
while 終了条件が未成立 **do**
  for $i = H$ から 1 **do**
    for $j = 1$ から d **do**
      for $k = 1$ から N_i **do**
        if $E({}^{A^i_j}\boldsymbol{\chi}^k) < E({}^{A^i_j}\hat{\boldsymbol{\chi}}^k)$ **then**
          粒子最良解の更新: ${}^{A^i_j}\hat{\boldsymbol{\chi}}^k \leftarrow {}^{A^i_j}\boldsymbol{\chi}^k$
        end if
        if $E({}^{A^i_j}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}}) < E({}^{A^i_j}\boldsymbol{\chi}^k)$ **then**
          群れ最良解の更新: ${}^{A^i_j}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} \leftarrow {}^{A^i_j}\boldsymbol{\chi}^k$
        end if
        if $\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} < {}^{A^i_j}\boldsymbol{\chi}^k$ **then**
          全体最良解の更新: $\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} \leftarrow {}^{A^i_j}\boldsymbol{\chi}^k$
        end if
      end for
      if 摂動回数が $S_{\mathrm{migration}}$ の倍数 **then**
        if $E({}^{A^{i+1}_j}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}}) > E({}^{A^{i-1}_{j+1}}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}})$ **and** $i \neq 1$ **then**
          移住: ${}^{A^{i-1}_j}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} \leftarrow {}^{A^{i-1}_{j+1}}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}}$
        else
          移住: ${}^{A^{i-1}_{j+1}}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} \leftarrow {}^{A^{i-1}_j}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}}$
        end if
      end if
      if 摂動回数が S_{exchange} の倍数 **then**
        if $E({}^{A^i_j}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}}) < E({}^{A^{i-1}_{\mathrm{worst}}}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}})$ **or** $i \neq 1$ **then**
          階層を越えた群れ最良解書き換え: ${}^{A^{i-1}_{\mathrm{worst}}}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}} \leftarrow {}^{A^i_j}\hat{\boldsymbol{\chi}}^{\mathrm{GBEST}}$
        end if
      end if
    end for
  end for
  if 摂動回数が S_{reset} の倍数 **then**
    for $i = 1$ から N_H **do**
      個体 ${}^{A^H_1}\boldsymbol{\chi}^i$ を乱数で初期化
    end for
  end if
end while

Distributed Hierarchical Particle Swarm Optimization

Distributed Hierarchical Particle Swarm Optimization（DH-PSO）とは林ら [42] によって提案された PSO の一改良である．この手法は，PSO の個体を複数の群れに分割し，各群れごとに独立して PSO を実行する分割型 PSO（D-PSO） [50] に，個体を階層に所属させ，各階層ごとに PSO を実行し，発見された解の優先度を定義することで，局所解への収束を回避する階層型 PSO（H-PSO） [51] の性質を加えることで，より局所解に捕われず，最適解発見能力を向上させた手法である．この擬似コードをアルゴリズム 13 に示す.

DH-PSO において，階層数を H，各階層を $h_i\ (i = 1, 2, \ldots, H)$ とする．階層構造は，最上位階層 h_H を根とした木として管理され，分岐数を d とする．例えば，$H = 3$，$d = 2$ とすると，第 1 階層に四つの群れ，第 2 階層に二つの群れ，第 3 階層に一つの群れが存在することになる．最上位層である h_H の群れはただ一つであり，複数存在することはないものとする．この手法では，各階層，群れごとに独立して PSO と同様の探索を行う点は D-PSO と同様であるが，ある一定ステップごとに以下のような操作を行う．

- 移住判定：$S_{\text{migration}}$ ステップごとに，$h_i\ (i = 2, 3, \ldots, H)$ に属しているそれぞれの群れ A^i に属す，一つ下の階層の群れ $a_j\ (j = 1, 2, \ldots, d)$ を以下の規則を用いて移住させる．
  - $E({}^{a_j}\hat{\boldsymbol{\chi}}^{\text{GBEST}}) > E({}^{a_{j+1}}\hat{\boldsymbol{\chi}}^{\text{GBEST}})$ ならば，${}^{a_j}\hat{\boldsymbol{\chi}}^{\text{GBEST}} \leftarrow {}^{a_{j+1}}\hat{\boldsymbol{\chi}}^{\text{GBEST}}$ と更新する．
  - $E({}^{a_j}\hat{\boldsymbol{\chi}}^{\text{GBEST}}) < E({}^{a_{j+1}}\hat{\boldsymbol{\chi}}^{\text{GBEST}})$ ならば，${}^{a_{j+1}}\hat{\boldsymbol{\chi}}^{\text{GBEST}} \leftarrow {}^{a_j}\hat{\boldsymbol{\chi}}^{\text{GBEST}}$ と更新する．

  ただし，$j + 1 = d + 1$ となるときは，$j + 1 = 1$ とする．
- 階層を越えた群れ最良解書き換え:S_{exchange} ステップごとに，$E({}^{A^i}\hat{\boldsymbol{\chi}}^{\text{GBEST}}) < E({}^{a_{\text{worst}}}\hat{\boldsymbol{\chi}}^{\text{GBEST}})$ ならば，${}^{a_{\text{worst}}}\hat{\boldsymbol{\chi}}^{\text{GBEST}} \leftarrow {}^{A^i}\hat{\boldsymbol{\chi}}^{\text{GBEST}}$ と更新する．ここで ${}^{a_{\text{worst}}}\boldsymbol{\chi}^{\text{GBEST}}$ は，a_j の中で ${}^{a_j}\hat{\boldsymbol{\chi}}^{\text{GBEST}}$ が最悪となる群れの最適解である．
- 最上位層個体の初期化：S_{reset} ステップごとに，最上位層 h_H の群れの探索を初期化する．この操作は，個体の多様性を維持することで，移住や最適解書き換えの操作によって発生しやすい初期収束を抑え，局所解から離脱するために行われる．

Invasive Weed Optimization

Invasive Weed Optimization（IWO）は，Mehrabian ら [20] によって提案された，雑草地の拡散戦略を発想元とした r-K 戦略 [52]（付録 A.3 参照）に基づく Swarm Intelligence を用いた最適化手法である．この手法では，探索序盤は r 戦略のような挙動を示し，探索終盤では K 戦略のような挙動を示すことが期待され，よって序盤では個体の多様性が維持され，終盤では近傍探索能力が向上することが期待される．

具体的な手順は以下の通りである．また，擬似コードをアルゴリズム 14 に示す.

アルゴリズム 14 Invasive Weed Optimization

N_{first}... 雑草数初期数
$N \cdots$ 現在の雑草数
$N_{\max} \cdots$ 雑草数上限
$\sigma \cdots$ 種子ばら撒き時に用いる正規分布の標準偏差
$\sigma_{\text{initial}}, \sigma_{\text{final}} \cdots$ σ の初期値と最終値
$n \cdots$ σ 用の設計パラメータ
$f_{\max}, f_{\min} \cdots$ 雑草群の目的関数値の上下限

for $i = 1$ から N_{first} **do**
　雑草 $\boldsymbol{\chi}^i$ を乱数を用いて初期化
end for
while 終了条件が未成立 **do**
　雑草群を目的関数値順にソート
　各雑草が生成する種子数を決定するための直線を計算:

$$y(x) = \frac{s_{\max} - s_{\min}}{f_{\max} - f_{\min}} x + \frac{(f_{\max} - f_{\min}) s_{\min}}{(s_{\max} - s_{\min}) f_{\min}}$$

　σ の計算: $\sigma = \left(\dfrac{T-t}{T}\right)^n (\sigma_{\text{initial}} - \sigma_{\text{final}}) + \sigma_{\text{final}}$
　for $i = 1$ から N **do**
　　雑草 $\boldsymbol{\chi}^i$ の生成する種子の数を計算: $\lfloor y(E(\boldsymbol{\chi})) \rfloor$
　　雑草 $\boldsymbol{\chi}^i$ から標準偏差 σ の正規分布基づいた種子のばら撒き
　end for
　ばら撒かれた種子数を N に加算
　if $N > N_{\max}$ **then**
　　現在生きている雑草とばら撒かれた種子の和集合から $N_{\max}$ 体分エリート選択
　end if
end while

1. 初期化：初期に存在する雑草を乱数を用いて初期化する．
2. 雑草のソート：目的関数の値に基づいて雑草をソートする．
3. 種子生成：次の手順に基づいて各個体が生成する種子の数を決定する．
   (a) 以下で定義される直線を求める．

$$y(x) = \frac{s_{\max} - s_{\min}}{f_{\max} - f_{\min}} x + \frac{(f_{\max} - f_{\min}) s_{\min}}{(s_{\max} - s_{\min}) f_{\min}} \tag{2.52}$$

   ここで，$s_{\max}$，$s_{\min}$ は各種子が生成可能な種子数の上下限，$f_{\max}$，$f_{\min}$ はソートした結果確認される雑草群の目的関数値の上下限である．
   (b) 各雑草の生成する種子の数を $\lfloor y(E(\boldsymbol{\chi})) \rfloor$ とする．
4. 種子拡散：各雑草を中心として，標準偏差 σ の正規分布に基づいて種をばら撒く．ただし，σ は各摂動ごとに次の式により計算される．

$$\sigma = \left(\frac{T-t}{T}\right)^n (\sigma_{\text{initial}} - \sigma_{\text{final}}) + \sigma_{\text{final}} \tag{2.53}$$

ここで，t は現在の摂動回数，T は摂動回数上限，n は調整パラメータ，σ_{initial} は σ の初期値，σ_{final} は σ の最終値である．

5. 種子選択 : 種子をばら撒いた結果，雑草数上限を上回った場合，現在生きている雑草とばら撒かれた種子をひとまとめに捉え，上位雑草数上限個体を抜き出し次の雑草群とする．

6. 手順 2 に戻る．

アルゴリズム 15 Group Search Optimizer

$D \cdots$ 次元数
$N \cdots$ 個体数
$N_s \cdots$ 最良個体についていく個体（たかり屋）の数
$l_{\max} \cdots$ 個体視界の距離限界
$\theta_{\max} \cdots$ 個体視界の視野角限界
$\alpha_{max} \cdots$ 個体の旋回角度現界
$a \cdots$ 最良個体の方向を戻すことを待つ摂動回数
$\circ \cdots$ Hadamard の掛け算
$\boldsymbol{D} \cdots$ 角度ベクトル $\boldsymbol{\varphi}$ をその角度を向いた単位ベクトルに変換する関数

for $i = 1$ から N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　個体の方向 $\boldsymbol{\varphi}^i$ を乱数で初期化
end for
while 終了条件が未成立 **do**
　最良個体 $\boldsymbol{\chi}^p$ が前方 3 点 $\boldsymbol{\chi}^z$，$\boldsymbol{\chi}^r$，$\boldsymbol{\chi}^l$ を確認:

$$
\begin{aligned}
\boldsymbol{\chi}^z &= \boldsymbol{\chi}^p + R_{\mathcal{N}}(\mathbb{R}, 0, 1) l_{\max} \boldsymbol{D}_p(\boldsymbol{\varphi}^p) \\
\boldsymbol{\chi}^r &= \boldsymbol{\chi}^p + R_{\mathcal{N}}(\mathbb{R}, 0, 1) l_{\max} \boldsymbol{D}_p(\boldsymbol{\varphi}^p + \frac{R_{\mathcal{U}}(\mathbb{R}^{D-1}, [0,1]^{D-1}) \theta_{\max}}{2}) \\
\boldsymbol{\chi}^l &= \boldsymbol{\chi}^p + R_{\mathcal{N}}(\mathbb{R}, 0, 1) l_{\max} \boldsymbol{D}_p(\boldsymbol{\varphi}^p - \frac{R_{\mathcal{U}}(\mathbb{R}^{D-1}, [0,1]^{D-1}) \theta_{\max}}{2})
\end{aligned}
$$

　if $E(\boldsymbol{\chi}^z)$，$E(\boldsymbol{\chi}^r)$，$E(\boldsymbol{\chi}^l)$ のどれかが $E(\boldsymbol{\chi}^p)$ より良い **then**
　　$\boldsymbol{\chi}^p \leftarrow \underset{\boldsymbol{\chi}}{\operatorname{argmin}}[E(\boldsymbol{\chi}^z), E(\boldsymbol{\chi}^r), E(\boldsymbol{\chi}^l)]$
　else
　　旋回: $\boldsymbol{\varphi}^p \leftarrow \boldsymbol{\varphi}^p + R_{\mathcal{U}}(\mathbb{R}^{D-1}, [0,1]^{D-1}) \alpha_{\max}$
　end if
　if 過去 a 摂動の間 $\boldsymbol{\chi}^p$ が更新されていない **then**
　　$\boldsymbol{\varphi}^p$ を a 摂動前のものに復元
　end if
　$\boldsymbol{\chi}^p$ 以外からたかり屋を N_s 体ランダムに選択
　for $i = 1$ から N $(i \neq p)$ **do**
　　if $\boldsymbol{\chi}^i$ がたかり屋である **then**
　　　位置を更新する: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{\chi}^i + R_{\mathcal{U}}(\mathbb{R}^D, [0,1]^D) \circ (\boldsymbol{\chi}^p - \boldsymbol{\chi}^i)$
　　else
　　　位置を更新する: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{\chi}^i + R_{\mathcal{N}}(\mathbb{R}, 0, 1) \alpha_{\max} l_{\max} \boldsymbol{D}^i(\boldsymbol{\varphi}^i)$
　　end if
　end for
end while

Group Search Optimizer

Group Search Optimizer（GSO）とは，He ら [21] によって提案された，動物群における各個体の動作を発想元とした Swarm Intelligence による最適化手法である．

具体的な手順を以下に示す．また，擬似コードをアルゴリズム 15 に示す．

1. 初期化：各個体 $\boldsymbol{\chi}^i$ を乱数で初期化する．
2. 終了条件判定：条件を満たしていれば探索を終了する．
3. 生産者の探索：最も良い解を保持する個体 $\boldsymbol{\chi}^p$ を生産者とする．この個体に前方三点を次のように生成，探索させ，解が改善された場合はその場に移動する．

$$\boldsymbol{\chi}^z = \boldsymbol{\chi}^p + R_{\mathcal{N}}(\mathbb{R},0,1) l_{\max} \boldsymbol{D}^p(\boldsymbol{\varphi}^p) \tag{2.54}$$

$$\boldsymbol{\chi}^r = \boldsymbol{\chi}^p + R_{\mathcal{N}}(\mathbb{R},0,1) l_{\max} \boldsymbol{D}^p(\boldsymbol{\varphi}^p + \frac{R_{\mathcal{U}}(\mathbb{R}^{D-1},[0,1]^{D-1})\theta_{\max}}{2}) \tag{2.55}$$

$$\boldsymbol{\chi}^l = \boldsymbol{\chi}^p + R_{\mathcal{N}}(\mathbb{R},0,1) l_{\max} \boldsymbol{D}^p(\boldsymbol{\varphi}^p - \frac{R_{\mathcal{U}}(\mathbb{R}^{D-1},[0,1]^{D-1})\theta_{\max}}{2}) \tag{2.56}$$

ここで，$l_{\max}$ は個体視界の距離限界であり，$\theta_{\max}$ は個体視界の視野角限界である．もし改善されなかった場合は，次の式に基づいて旋回する．

$$\boldsymbol{\varphi}^p \leftarrow \boldsymbol{\varphi}^p + R_{\mathcal{U}}(\mathbb{R}^{D-1},[0,1]^{D-1})\alpha_{\max} \tag{2.57}$$

ここで，$\alpha_{\max}$ は個体の旋回可能な角度の限界である．

4. たかり屋の移動：生産者以外の $N_s(< N)$ 体をランダムにたかり屋として選定し，次の式に基づいて生産者に付いて行くような移動を行う．

$$\boldsymbol{\chi}^i \leftarrow \boldsymbol{\chi}^i + R_{\mathcal{U}}(\mathbb{R}^D,[0,1]^D) \circ (\boldsymbol{\chi}^p - \boldsymbol{\chi}^i) \tag{2.58}$$

ここで，$\circ$ は Hadamard の掛け算を表す．

5. 非従属個体の移動：生産者にたからず独自に動く個体は次の式に基づいて移動を行う．

$$\boldsymbol{\chi}^i \leftarrow \boldsymbol{\chi}^i + R_{\mathcal{N}}(\mathbb{R},0,1)\alpha_{\max} l_{\max} \boldsymbol{D}^i(\boldsymbol{\varphi}^i) \tag{2.59}$$

6. 手順 2 に戻る．

Artificial Bee Colony Optimization

Artificial Bee Colony Optimization（ABC）とは，Karaboga ら [22] によって提案された，蜂の給餌行動を参考にした Swarm Intelligence による最適化手法の一つである．具体的な手順を以下に示す．また，擬似コードをアルゴリズム 16 に示す．

1. 初期化：各蜂 $\boldsymbol{\chi}^i$ を乱数で初期化する．
2. 終了条件判定：条件が満たされていれば探索を終了する．
3. 蜂自身の移動：次の式に基づいて $\boldsymbol{v}^i$ を生成し，目的関数値の良い方で $\boldsymbol{\chi}^i$ を更新する．

$$\boldsymbol{v}^i = \boldsymbol{\chi}^i + R_{\mathcal{U}}(\mathbb{R}^D,[-1,1]^D)(\boldsymbol{\chi}^i - \boldsymbol{\chi}^k) \tag{2.60}$$

アルゴリズム 16 Artificial Bee Colony Optimization

$D \cdots$ 次元数
$N \cdots$ 個体数
$N_s \cdots$ scout 行動する蜂の数
$p_i\ (i = 1, 2, \ldots N) \cdots$ onlooker の ID
$F \cdots$ 適応度関数 $F(\boldsymbol{\chi}) = -E(\boldsymbol{\chi}) + E$ の最大値
$\boldsymbol{\chi}^{\max}, \boldsymbol{\chi}^{\min} \cdots$ 定義域の上下限

for $i = 1$ から N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
end for
while 終了条件が未成立 **do**
　for $i = 1$ から N **do**
　　サブ親 $\boldsymbol{\chi}^k\ (k \neq i)$ をランダムに選択
　　移動後個体生成: $\boldsymbol{v}^i = \boldsymbol{\chi}^i + R_{\mathcal{U}}(\mathbb{R}^D, [-1, 1]^D)(\boldsymbol{\chi}^i - \boldsymbol{\chi}^k)$
　　if $E(\boldsymbol{v}^i) < E(\boldsymbol{\chi}^i)$ **then**
　　$\boldsymbol{\chi}^i \leftarrow \boldsymbol{v}^i$
　　end if
　　$\dfrac{F(\boldsymbol{\chi}^m)}{\sum_{l=1}^{N} F(\boldsymbol{\chi}^l)}$ の確率で $p_i = m$ となる
　　サブ親 $\boldsymbol{\chi}^k\ (k \neq p_i)$ をランダムに選択
　　移動後個体生成: $\boldsymbol{v}^i = \boldsymbol{\chi}^{p_i} + R_{\mathcal{U}}(\mathbb{R}^D, [-1, 1]^D)(\boldsymbol{\chi}^{p_i} - \boldsymbol{\chi}^k)$
　　if $E(\boldsymbol{v}^i) < E(\boldsymbol{\chi}^{p_i})$ **then**
　　$\boldsymbol{\chi}^i \leftarrow \boldsymbol{v}^i$
　　end if
　end for
　for $i = 1$ から N_s **do**
　　ランダム移動: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{\chi}^{\min} + R_{\mathcal{U}}(\mathbb{R}^D, [0, 1]^D)(\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min})$
　end for
end while

4. onlooker 蜂の同定：蜂 $\boldsymbol{\chi}^i$ が見ている蜂を，$\dfrac{F(\boldsymbol{\chi}^m)}{\sum_{l=1}^{N} F(\boldsymbol{\chi}^l)}$ の確率で $p_i = m$ となる $\boldsymbol{\chi}^{p_i}$ として設定する．ここで，F は適応度関数であり，目的関数 E を値が大きい方が良くかつ値域が 0 以上に修正した関数である．

5. 蜂自身の移動：次の式に基づいて $\boldsymbol{v}^i$ を生成し，目的関数値の良い方で $\boldsymbol{\chi}^i$ を更新する．

$$\boldsymbol{v}^i = \boldsymbol{\chi}^{p_i} + R_{\mathcal{U}}(\mathbb{R}^D, [-1, 1]^D)(\boldsymbol{\chi}^{p_i} - \boldsymbol{\chi}^k) \tag{2.61}$$

6. scout 蜂の探索：$i \in [0, N_s]$ である蜂 $\boldsymbol{\chi}^i$ は scout 蜂としてランダム探索を行う．

$$\boldsymbol{\chi}^i \leftarrow \boldsymbol{\chi}^{\min} + R_{\mathcal{U}}(\mathbb{R}^D, [0, 1]^D)(\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}) \tag{2.62}$$

ここで，$\boldsymbol{\chi}^{\max}$，$\boldsymbol{\chi}^{\min}$ は定義域の上下限である．

7. 手順 2 に戻る．

Random Search

Random Search（RS）は，探索空間全域を対象とした乱数を生成することで，擬似的に解の全ての可能性を考慮する最も単純な確率的最適化手法である．探索空間に一切の仮定を置かないという強力な利点があるが，探索空間が大きい場合は解の発見までに大きさに応じた時間が掛かり，探索空間が連続であると厳密な最適解の導出確率は確率的に 0 であり，ただ単純に用いるのみの場合，最適解の導出は不可能である．

Random Walk として Lévy Flight を用いた手法

Random Walk とは，Random Search の実装の一つであり，複数の具体的な実装が考えられている．本研究では 1 摂動ごとに探索空間全域の乱数を直前の摂動時の解に加えて次の解を生成し，結果が定義域を越えていた場合は探索空間がトーラス状であると仮定して位置を決める実装を用いる．

Random Walk としての Lévy Flight (LF) [43] とは，Lévy 分布に基づいた Random Walk のことである．LF は通常の Random Walk よりも大きな移動と小さな移動の差が明確であり，大域的な探索と近傍探索のバランスが一様分布に基づいた Random Walk よりもよくなることが期待される．

Cuckoo Search

Cuckoo Search（CS）とは，Yang ら [23] によって提案された Random Search を用いた多点探索手法である．この手法では Lévy 分布に基づいた乱数を用いて探索を行うため，LF の一改良と捉えることも可能である．具体的な実装を以下に示す．また，擬似コードをアルゴリズム 17 に示す．

1. 初期化：各個体 $\boldsymbol{\chi}^i$ を乱数で初期化する．
2. 終了条件判定：条件を満たしていれば終了する．
3. Lévy 分布による突然変異個体生成：次の式に基づいて突然変異個体 $\boldsymbol{\chi}'^i$ を生成

$$\boldsymbol{\chi}'^i = \boldsymbol{\chi}^i + \alpha \circ R_{\mathcal{L}}(\mathbb{R}^D, \lambda) \tag{2.63}$$

   ここで，α はステップサイズを表し，$\circ$ は Hadamard の掛け算を表す．
4. 生存選択：$E(\boldsymbol{\chi}^i)$，$E(\boldsymbol{\chi}'^i)$ を比較し，より良い個体を新たな $\boldsymbol{\chi}^i$ とする．
5. ランダムな移動：各個体は P_a の確率でランダムな移動を行う．

$$\boldsymbol{\chi}^i \leftarrow R_{\mathcal{U}}(\mathbb{R}^D, [\boldsymbol{\chi}^{\min}, \boldsymbol{\chi}^{\max}]) \tag{2.64}$$

6. 手順 2 に戻る．

アルゴリズム 17 Cuckoo Search

$D \cdots$ 次元数
$N \cdots$ 個体数
$\alpha \cdots$ ステップサイズ
$R_{\mathcal{L}}(\mathbb{A}, a) \cdots$ 領域 $\mathbb{A}$ にてパラメータ a を用いた Lévy 分布に基づく乱数
$P_a \cdots$ 個体が通常の Random Search を行う確率
$\circ \cdots$ Hadamard の掛け算
$\lambda \cdots$ ステップのために用いる Lévy 分布のパラメータ
$\boldsymbol{\chi}^{\max}, \boldsymbol{\chi}^{\min} \cdots$ 定義域の上下限

for $i = 1$ から N **do**
　個体 $\boldsymbol{\chi}^i$ を初期化
end for
while 終了条件が未成立 **do**
　for $i = 1$ から N **do**
　　突然変異個体生成: $\boldsymbol{\chi}'^i = \boldsymbol{\chi}^i + \alpha \circ R_{\mathcal{L}}(\mathbb{R}^D, \lambda)$
　　if $E(\boldsymbol{\chi}'^i) < E(\boldsymbol{\chi}^i)$ **then**
　　　個体の更新: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{\chi}'^i$
　　end if
　　if P_a の確率判定に成功 **then**
　　　個体のランダム移動: $\boldsymbol{\chi}^i \leftarrow R_{\mathcal{U}}(\mathbb{R}^D, [\boldsymbol{\chi}^{\min}, \boldsymbol{\chi}^{\max}])$
　　end if
　end for
end while

2.5.3　実験設定

2.5.1 節にて列挙したベンチマーク関数を，DE-SP と 2.5.2 節で述べた手法を用いて最適化を行い，結果を比較する．実験設定として，探索空間の次元数を $D = 2$，摂動回数上限を目的関数評価回数が 1000000 回となるように設定した．各手法に用いたパラメータは付録 A.4 に示す．これらのパラメータのほとんどは各手法の提案されている文献の推奨値を利用した．推奨値のないパラメータ（DE 類縁手法の F，C_R 等）は事前に小規模の総当たり実験を行い，最も成績のよいパラメータを使用した．

実験において，評価基準は最適解発見回数割合を用いるものとし，試行回数は 1000 回とする．

表 2.1: 各探索空間における各手法の最適感発見回数割合

	DE-SP	DE	DE-SP($M=0$)	DE2	DE/nrand/1	DE/isolated/1	PSO-DE
f_{Ras}	100	100	100	100	33.6	100	100
f_{Ros}	100	100	99.8	100	31.3	100	100
f_{Sch}	100	100	99.9	100	56	100	100
f_{Gri}	100	100	100	100	25	100	44.8
f_{Ack}	100	100	100	100	84.4	100	100
f_{NF1}	76.3	24.3	71.9	24.8	0.3	15	13.1
f_{NF2}	100	100	100	100	17.2	94.4	42.3

	WMWC-DE	SA	SA/AAN	NMS	HS	GA(Uni+Elite)	GA(UNDX+MGG)
f_{Ras}	0	0	8.2	27.3	0	0	0
f_{Ros}	0	0	39.3	4.9	0	0	0
f_{Sch}	0	0	25.7	34.3	0	0	0
f_{Gri}	0	0	0.1	6.9	0	0	0
f_{Ack}	0	3.3	99.9	0	0	0	0
f_{NF1}	0	0.3	0.1	0	0	0	0
f_{NF2}	0	2	6	0	0	0	0

	GA(BLX-α+MGG)	$(1+1)$-ES	$(\mu+\lambda)$-ES	PSO	PSO-KS	DH-PSO	IWO
f_{Ras}	0	0	56.8	99	93.9	83.4	0
f_{Ros}	0	0	99.9	0.5	99.6	0	0
f_{Sch}	0	0	42.4	26.3	31.7	14.7	7.3
f_{Gri}	0	0	8.7	94.8	33.5	53.6	0
f_{Ack}	0	0	100	0.2	99.7	0	0
f_{NF1}	0	0	0	0	27.8	0	0
f_{NF2}	0	0	3.8	0	42.3	0	0

	GSO	ABC	RS	LF	CS		
f_{Ras}	0	100	0	0	0		
f_{Ros}	0	100	0	0	0		
f_{Sch}	0	100	0	0	0		
f_{Gri}	0	99.4	0	0	0		
f_{Ack}	0	100	0	0	0		
f_{NF1}	0	18.1	0	0	0		
f_{NF2}	0	83.4	0	0	0		

2.5.4　結果と考察

結果を表 2.1 に示す．また，各ベンチマーク関数を最適化している間の最良解平均を図 2.8，図 2.9，図 2.10，図 2.11，図 2.12，図 2.13，図 2.14 に示す．

結果全体より示唆されること

表 2.1 から，通常用いられるベンチマーク関数五つについて，DE-SP，DE，DE2，DE/isolated/1 が全て 100%であり，最も良い性能を示していることが確認できる．次

点としてこれら四つの手法に，0.1，0.2%程度成績が悪いものとして，DE-SP($M = 0$)，Griewank Function 以外の成績が 100%であるものとして ABC，PSO-DE が挙げられる．DE/nrand/1 や WMWC-DE は，DE 類縁手法でありながら従来の DE よりも悪い結果となり，WMWC-DE においては厳密に最適解を発見できなかった．また，今回比較手法として挙げた構成の三つの GA も最適解を発見できなかった．ES においては $(1+1)$-ES の成績は GA と同様であるが，$(\mu+\lambda)$-ES は最適解の発見に成功しており，Rosenbrock Function や Ackley Function においては小数点第一位を四捨五入すると 100%となる程の良い成績を示した．Swarm Intelligence の手法の中では PSO，PSO-KS，DH-PSO（Rastrigin，Schwefel，Griewank Function のみ），ABC は最適解の発見に成功しているが，IWO（Schwefel Function 以外），GSO は最適解を発見できなかった．SA，SA/AAN（Ackley Function 以外），NMS，HS については，40%未満の成績であり，HS についてはそもそも最適解の発見に成功しなかった．ここで，図 2.8，図 2.9，図 2.10，図 2.11，図 2.12 から DE 類縁手法や二つの ES，NMS は比較的早期に収束し，そののちに探索が停滞していることが確認される．これらにやや遅れて二つの SA が収束していることが確認される．Swarm Intelligence 系統の手法，三つの GA，三つの Random Search は，比較的全探索時間に渡って探索が進行していることが確認される．特に IWO は Schwefel Function 以外の探索空間において最終摂動時に最適解に限りなく近い値に辿りついている．しかしながら，表 2.1 では最適解発見回数割合が 0%であったことから，IWO は探索最終期においての解への収束性が悪いと考えられる．

NF1 最適化結果より示唆されること

表 2.1 の NF1 の最適化結果に注目する．各手法における NF1 の最適化結果は，NF2 を含む他のベンチマーク関数と比較して最適解発見率が最も低いことが確認される．これは NF1 の従来より仮定されている大域的単峰性が存在しないという性質が，実際に探索困難性として現れたものと考えられる．その中で，DE-SP は全手法の中で最も良い性能を示すことが確認される．そして，DE-SP($M = 0$) と，DE2 の結果を比較すると，DE-SP($M = 0$) の結果の方が良いことから，要素ごとにサブ親を選択し直す点がより大きく NF1 における最適化性能の向上に寄与したのではないかと考えられる．DE 類縁手法の中では DE/nrand/1，DE/isolated/1，PSO-DE，WMWC-DE が NF1 において従来の DE よりも悪い結果となり，WMWC-DE においてはベンチマーク関数時と同様厳密に最適解を発見できなかった．図 2.13 に注目すると，DE-SP($M = 0$) は DE-SP よりも速やかに収束している．これは，DE-SP の M を増やすと直接に探索に関わる個体数が減少するためであると考えられる．NF1 においては最適解発見率に大きな差異は見られなかったが，M の値により最適解発見率が大きく変わってくるような場合は，最適解発見率と収束速度のトレードオフを図ることになると考えられる．また，RS，LF，CS といったランダムサーチ系統は探索が継続されていることが確認できる．また，DE，DE2，DE/nrand/1，DE/isolated/1，は探索が早期に停滞しているが，PSO-DE や WMWC-DE は探索が継続されていることが確認される．しかし，RS，LF，CS，PSO-DE，WMWC-DE の収束速度は DE-SP のものと比較して遅いものであることから，表 2.1 の結果は他の手法より悪いものとなったと考えられる．また，IWO は探索終盤に探索の進行速度が上昇している

ことが確認される．IWO は探索終盤になると近傍探索能力が大きく上昇するため，この性質が現れたものではないかと考えられる．

NF2 最適化結果より示唆されること

表 2.1 の NF2 の最適化結果に注目する．NF2 には大域的単峰性が存在するが，他のベンチマーク関数ほど局所解が規則的に配置されていない．そのため，従来法には最適解発見回数割合が下がっているものがあることが確認される．また，NF2 は局所解の数も多いため，同じく局所解の数が多い Griewank Function での結果が比較的悪い手法は NF2 での結果も比較的悪いことが確認される．その中でも，従来の DE は 100%の最適解発見回数割合を維持しており，DE-SP でもそれが維持されていることが確認される．図 2.14 に注目すると，全体傾向としては NF1 よりも収束が速くなったような図となっているが，WMWC-DE の探索が途中から停滞していることが確認される．WMWC-DE では ζ_{wm} に基づいて徐々にパラメータ F の最大値を減らしているが，これにより各個体の移動量が必要以上に削がれている可能性がある．これが，探索空間によっては局所解から出ることができなくなり，表 2.1 の結果を悪くし，探索を停滞させる原因となっている可能性がある．

Rastrigin Function 最適化結果より示唆されること

図 2.8 に注目する．まず，ES-(1 + 1)，SA，SA/AAN といった単点探索手法が 5000 摂動に到達する前に摂動が停滞していており，かつ最良解平均が他の手法と比較して高い位置を推移していることが確認される．Rastrigin Function は局所解が格子状でありかつ厳密に凸な関数を大きく変形せず拡張した目的関数であるため，多峰性関数の中では単純なものであると捉えることができる．しかしながらこれらの手法は有効な探索が可能とは評価し難い結果を示していることから，目的関数の構造が単純であっても，多峰性である限り単点探索手法は有効ではないことが示唆される．ただし，NMS，$(\mu + \lambda)$-ES，GA(UNDX+MGG)，PSO-KS といった手法も，同時期に探索が停滞していることから，任意の多点探索手法が局所解から離脱する能力に長けていないことも確認される．探索の継続に成功していると思われる手法は，IWO，HS，CS，GA(Uni+Elite)，GSO，PSO，RS，LF，DH-PSO，ABC，DE に分類される全手法である．ただし，GA(Uni+Elite) ののグラフの右下がりの程度は他に列挙した手法と比べて非常に軽微であり，即ち探索継続能力はこれらの中で最も低く，IWO のグラフの右下がり程度は終盤において急激に増加するため，これは序盤に収束させない代わりに終盤の解収束時に間違いなく最適解へ至ろうとする IWO 独自の工夫が手法設計者の意図通りに機能した結果と捉えるべきであろう．

探索の継続に成功している手法について表 2.1 も合わせて確認すると，実際に最適解の発見に成功しているのは WMWC-DE を除いた DE に分類される手法，PSO，DH-PSO，ABC であり，それ以外は全試行にて最適解の発見に失敗している．RS，LF，CS はラン

ダムサーチの手法であるため，解へ個体を収束させる機能そのものが存在せず，この結果は十分にものである．しかし，終盤において解への収束速度が上がるよう調整されている IWO においても最適解発見回数は 0 であり，この特性をもってしてもなお解への収束精度に難があることが示唆される．HS，GA(Uni+Elite)，GSO，WMWC-DE も最適解発見回数は 0 であるため，これらの手法も収束能力は低いことが確認される．また，探索が途中で停滞している手法について確認すると，SA/AAN，NMS，PSO-KS は最適解を発見している試行が存在しており，探索継続性能，即ち局所解から離脱する能力は低くとも，現在の探索において最適解の存在する極値周辺を発見する能力が存在することが示唆される．また，この中で SA/AAN は単点探索手法であることから，SA/AAN は単点探索手法の中でも比較的高い探索性能を持つことも示唆される．

Rosenbrck Function 最適化結果より示唆されること

図 2.9 に注目する．今回は (1 + 1)-ES を除いて多くの手法が探索継続に成功していることが確認される．この結果に加え，Rosenbrock Function は単峰性関数であることから，この関数は探索の継続それ自体は容易な探索空間であり得るという性質が表面化したことが示唆される．一方，表 2.1 を確認すると，WMWC-De，HS，GA3 種，(1 + 1)-ES，DH-PSO，IWO，GSO，RS，LF，CS が最適解発見に失敗している．特に DH-PSO は，Rastrigin Function においては最適解発見に成功していることから，Rosenbrock Function は探索継続自体は容易でも，最適解への収束は困難であることが示唆される．これは，図 2.9(b) と図 2.8(b) を比較してることでも伺うことができる．図 2.8(b) において縦軸に沿っているかのように見えるほどに拘束に収束している手法でも，図 2.9(b) では縦軸からは僅かに離れており，即ち最適解への収束により多くの摂動を必要としたことが確認される．

Schwefel Function 最適化結果より示唆されること

図 2.10 に注目する．この探索空間では多くの手法が探索停滞を起こしており，(1 + 1)-ES，PSO-KS，NMS，$(\mu + \lambda)$-ES，DE/nrand/1，IWO，SA/AAN，HS，SA の探索は停滞していることが窺える．IWO について，Schwefel Function 以外では探索の停滞を起こしていないにも関わらずこの探索空間では停滞してしまっていることから，IWO は最適解が中央付近に存在せずかつ有力な局所解（値が十分に小さな局所解）が斑に複数存在すると，探索が進まなくなるであろうことが示唆される．ただし表 2.1 より，IWO は最適解の発見には成功しており，このことから Schwefel Function は最適解周辺の発見に成功しさえすれば解への収束は容易であることが示唆される．

Griwank Function 最適化結果より示唆されること

図 2.11 に注目する．この探索空間では (1+1)-ES，(μ+λ)-ES，SA，GA(UNDX+MGG)，NMS，SA/AAN，PSO-KS，GA(BLX-α+MGG)，GA(Uni+Elite)，PSO-DE，DE/nrand/1 の探索が停滞していることが確認される．ただし，$(1+1)$-ES 以外の手法は最良解平均が 0.2 以下に至ることに成功しているため，多くの手法はある程度の探索が可能であることが確認される．Griwank Function には大域的単峰性はあるものの，探索空間の広さに対して局所解が多く存在する探索空間であるため，この探索空間の性質として，最適解付近までは個体が至れたとしてもその状態から最適解を発見するに至る段階にて探索が進まず停滞しやすいことが予見される．この性質が典型的に図 2.11 に表れたと示唆することができる．解の発見に成功している手法は WMWC-DE 以外の DE 類縁手法，SA/AAN，NMS，$(\mu+\lambda)$-ES，PSO，PSO-KS，DH-PSO，ABC である．特に DE 類縁手法は 100%を示す手法が多く，典型的な大域的単峰性があれば，Griwank Function のような探索空間においても有効に活用できることが示唆される．

Ackley Function 最適化結果より示唆されること

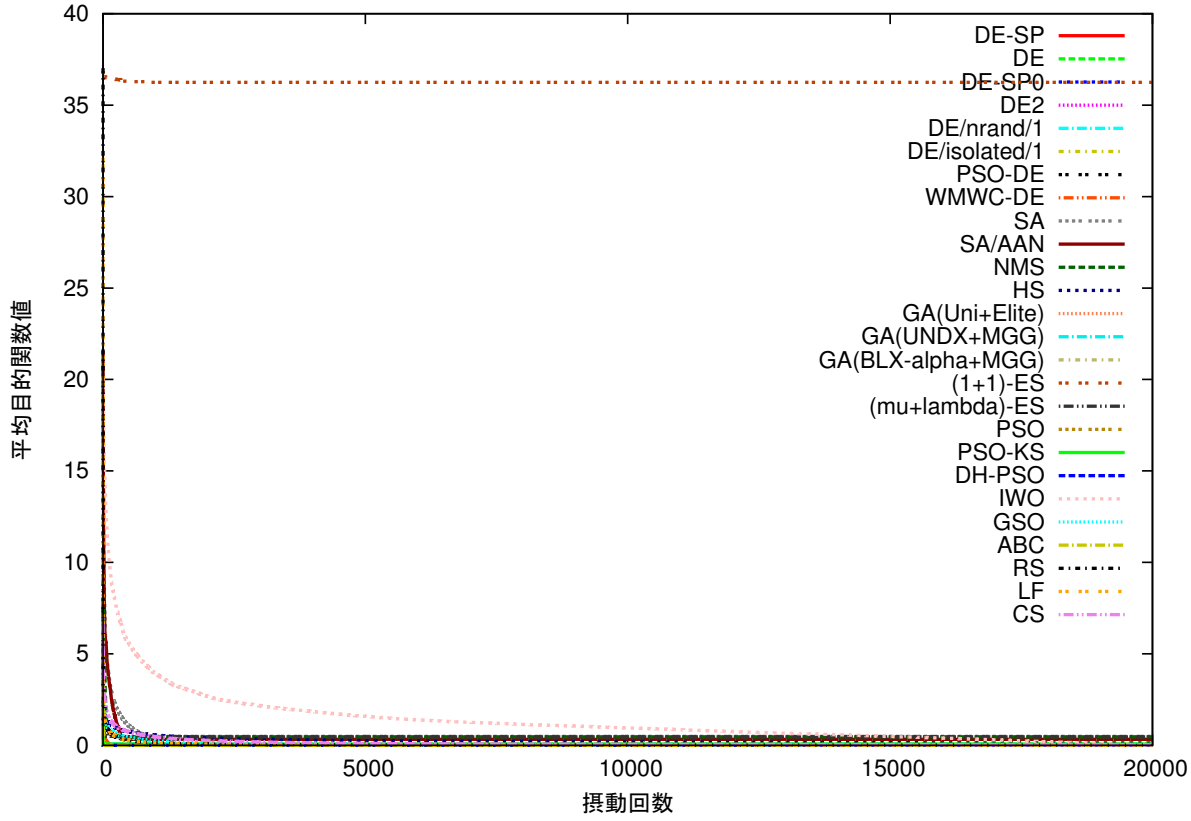
図 2.12 に注目する．Griwank Function ある程度似た傾向があり，$(1+1)$-ES 以外の手法の最良解平均が 1.4 以下に至ることに成功している．ただし，探索の停滞が確認されるのは，$(1+1)$-ES，SA，GA(UNDX+MGG)，PSO-KS であり，Griwank Function よりは探索の継続は容易な探索空間であることが示唆される．解の発見に成功している手法は WMWC-DE 以外の DE 類縁手法，SA/AAN，$(\mu+\lambda)$-ES，PSO，PSO-KS，ABC である．ここで，SA/AAN と NMS の結果を比較する．SA/AAN は単点探索手法でありながら 99.9%の発見率であるが，多点探索手法である NMS は一度も最適解の発見に成功していない．手法を振り返ると，SA/AAN では探索時に解の受理率が悪ければ近傍幅が大きくなり続けるが，NMS においては反射時に得られる一時個体 $E(\boldsymbol{\chi}^r)$ が最良解 $E(\boldsymbol{\chi}^l)$ より良い場合にしか探索範囲拡張処理が発生せず，探索停滞時からの離脱能力は SA/AAN と比べて低いことが予見される．このことから，Ackley Function は探索範囲の再伸長能力が低い手法では探索が容易ではない可能性が示唆される．

2.6　まとめ

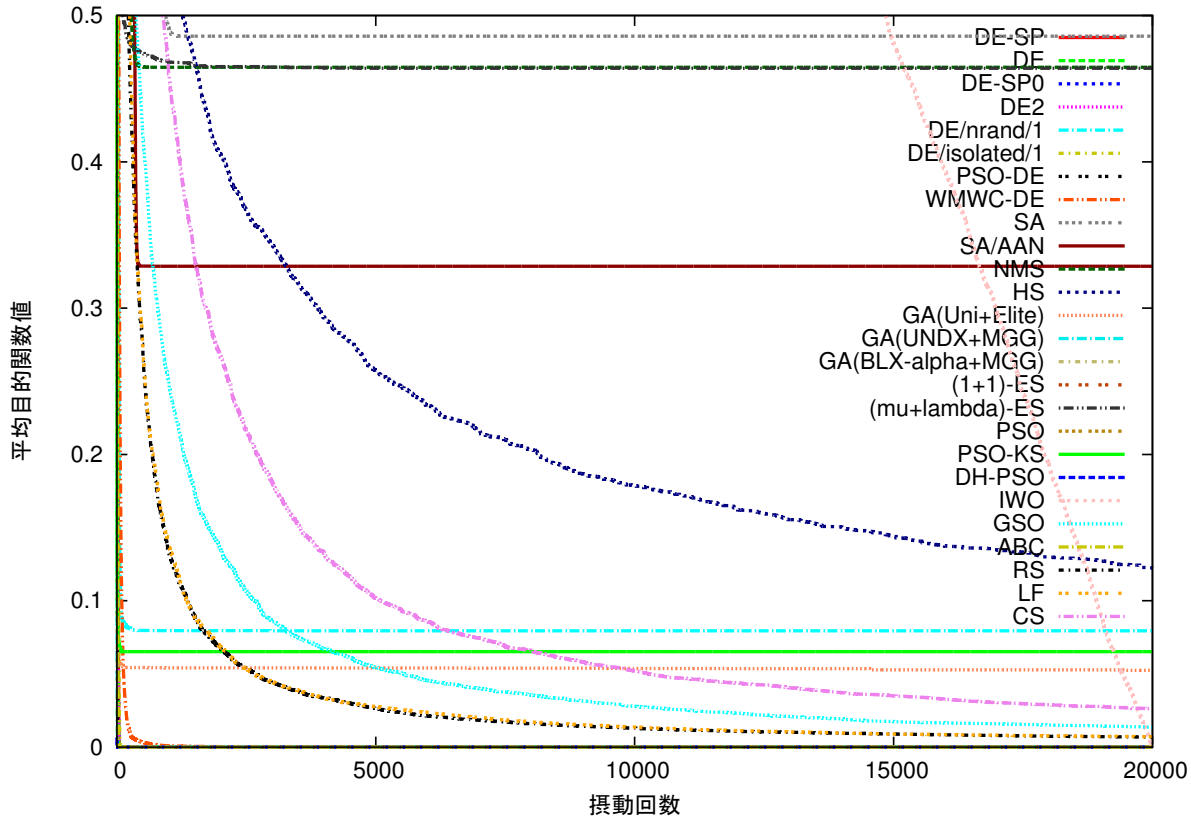
本章では大域的単峰性に乏しい空間においても探索が継続可能な Differential Evolution on Scattered Parents（DE-SP）を提案した．そして，ベンチマーク関数による性能評価を行い，DE-SP の性能が最も良く，特に大域的単峰性の乏しい NF1 において性能差が顕著であることを確認した．

本章では 2 次元の，大域的単峰性が乏しいものがある以外は慣例的に仮定される性質しか持たないベンチマーク関数を用いた実験のみ行った．DE-SP の実問題・擬実問題へ

の適用は次章にて述べる.

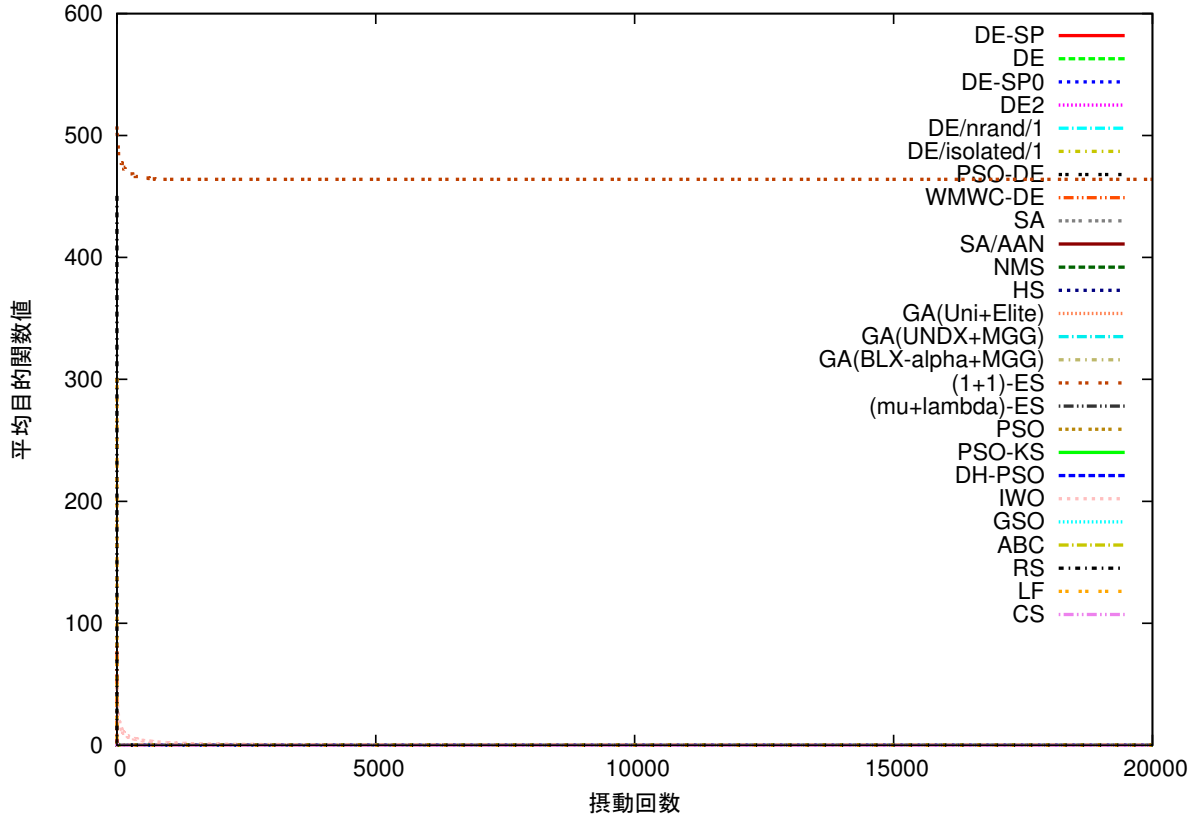


(a) 全体図

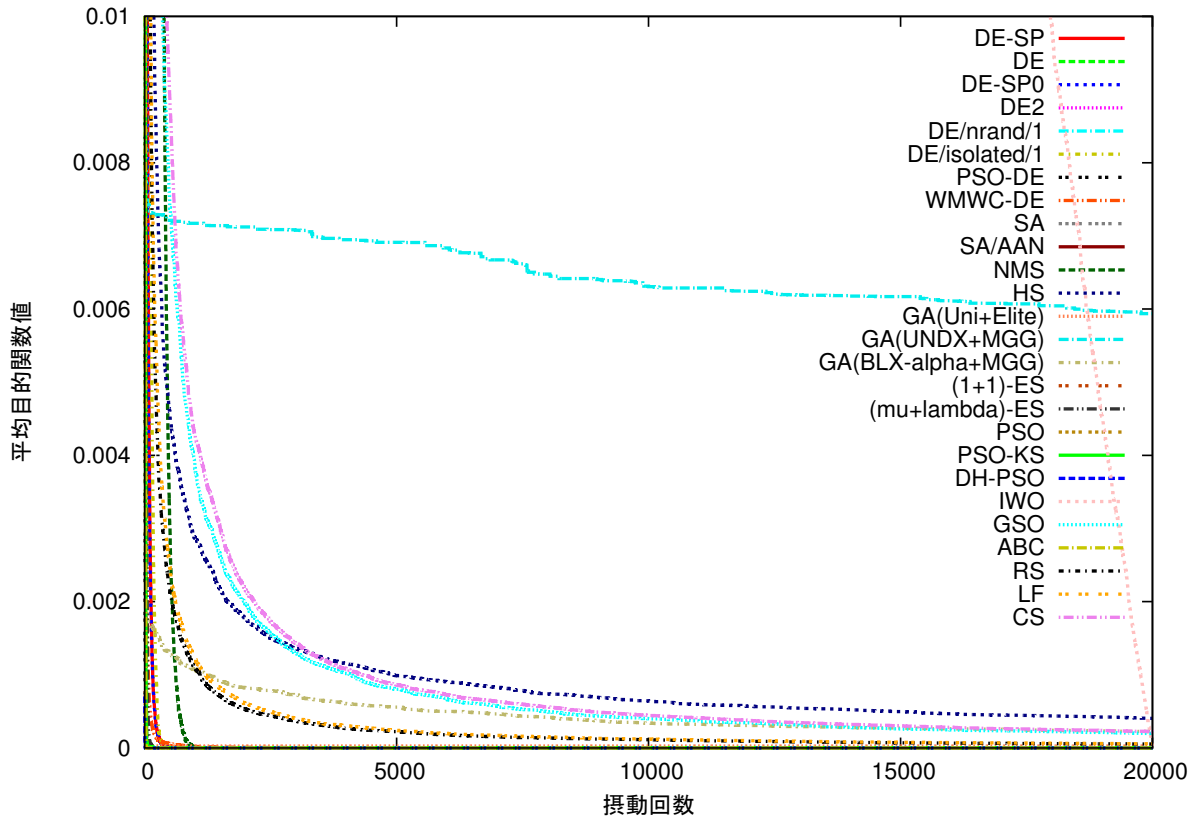


(b) 密集域拡大図

図 2.8: Rastrigin Function における各手法の最良解平均

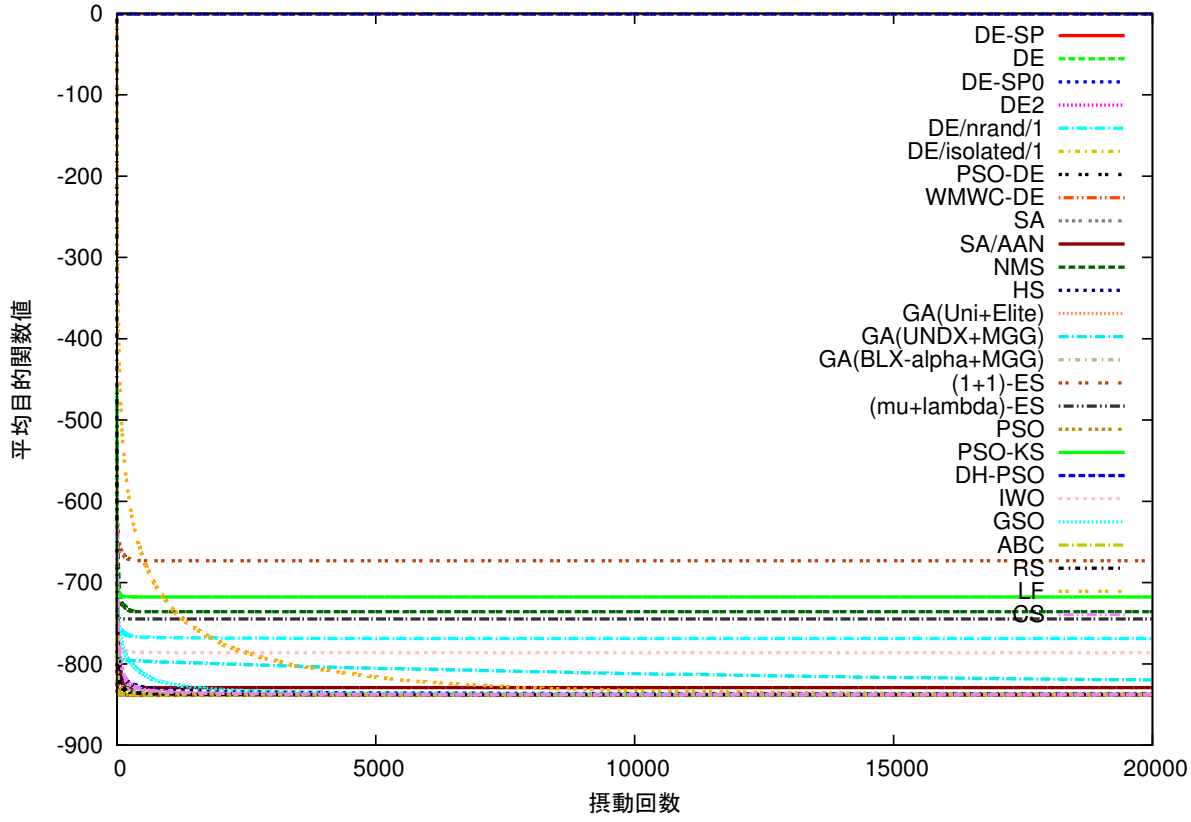


(a) 全体図

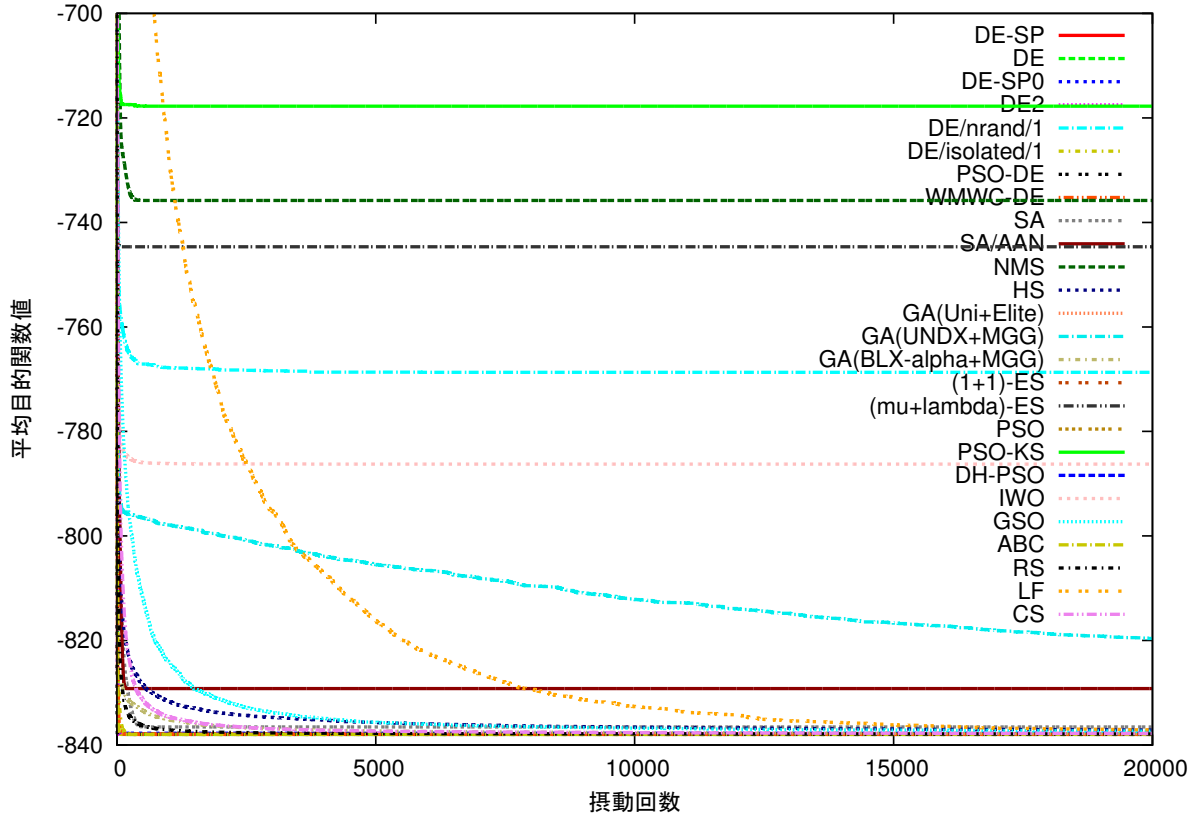


(b) 密集域拡大図

図 2.9: Rosenbrock Function における各手法の最良解平均

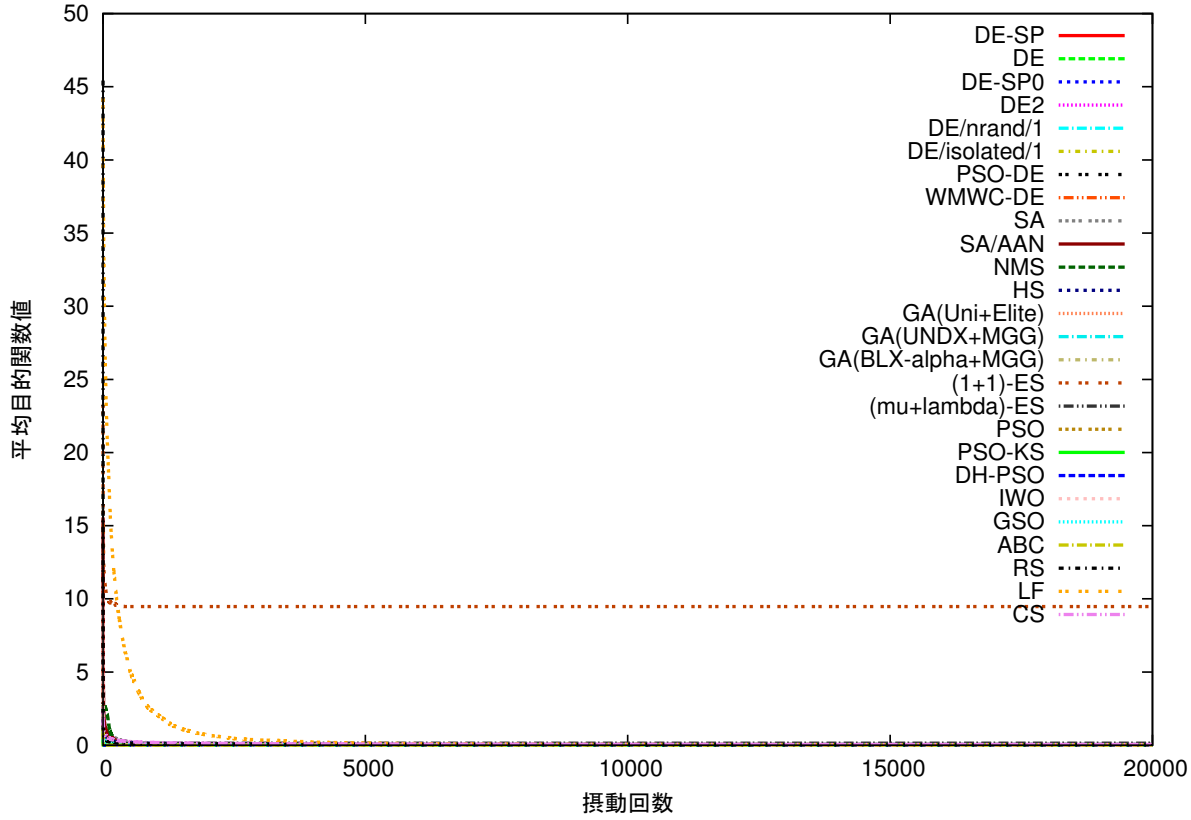


(a) 全体図

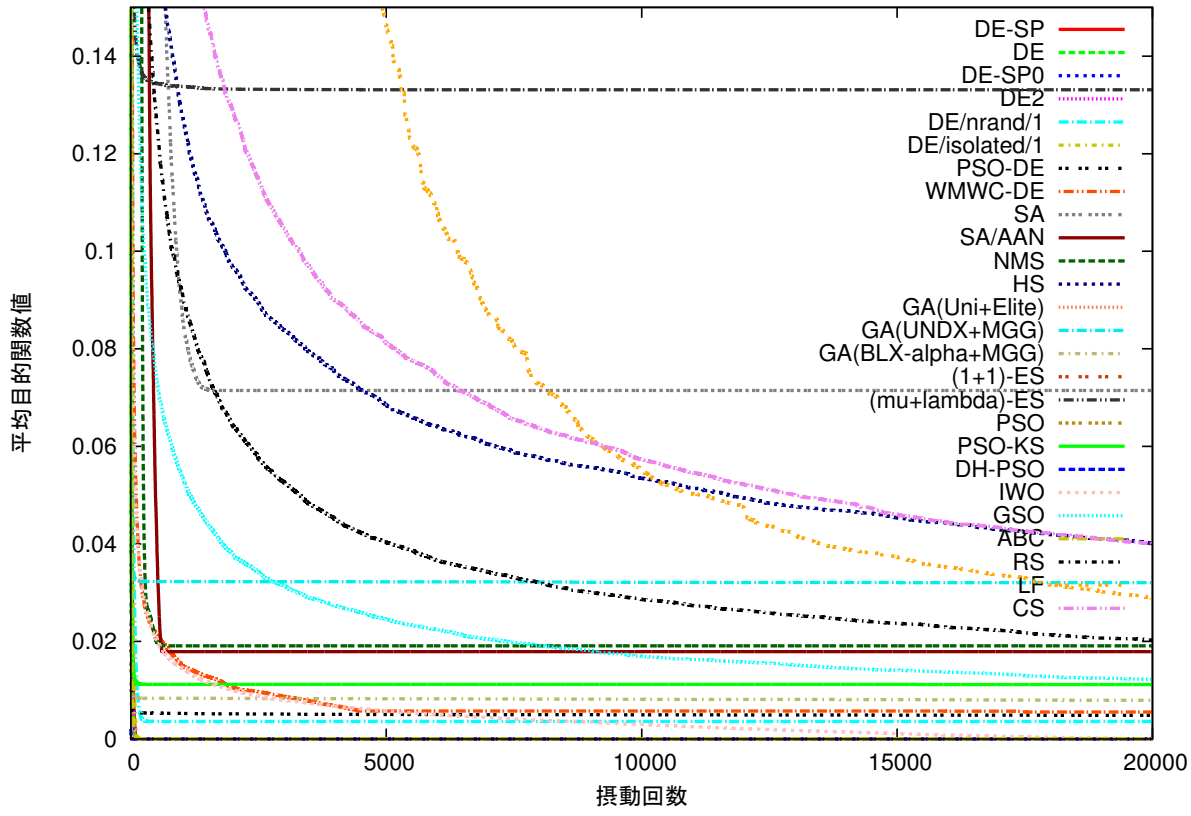


(b) 密集域拡大図

図 2.10: Schwefel Function における各手法の最良解平均

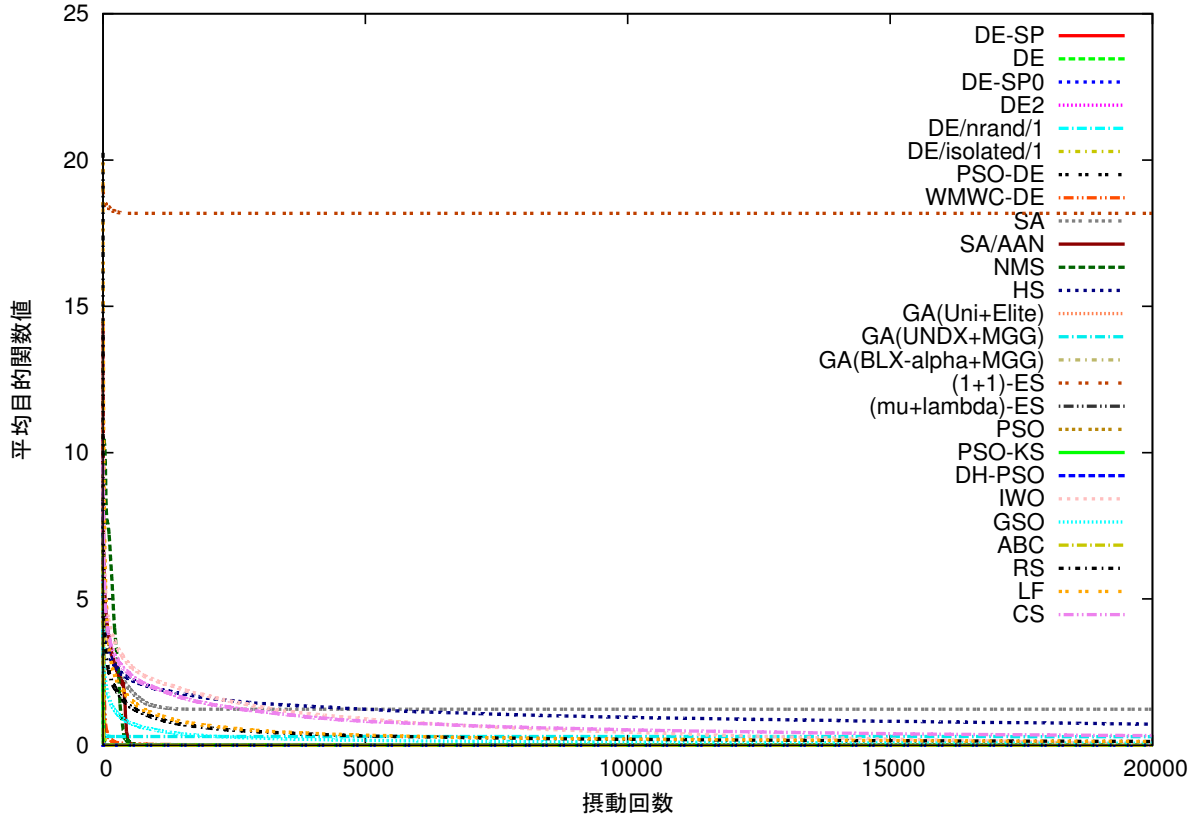


(a) 全体図

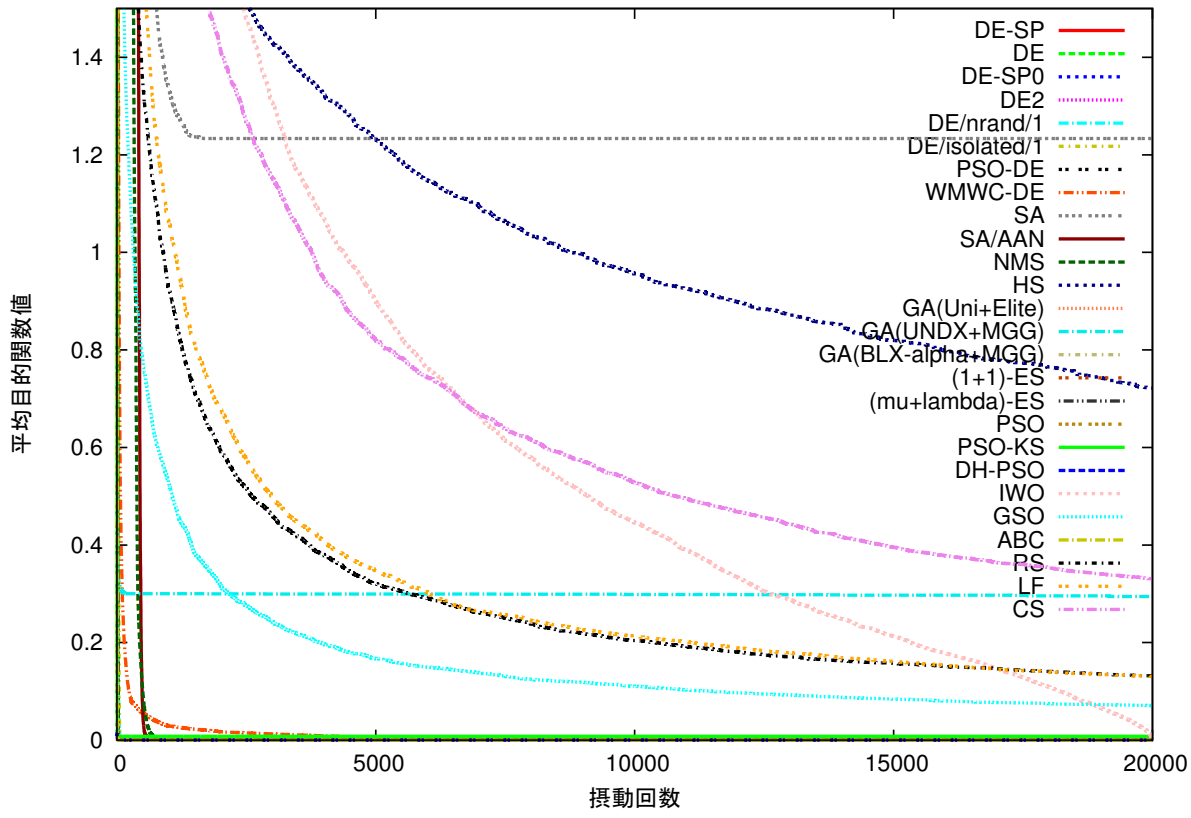


(b) 密集域拡大図

図 2.11: Griewank Function における各手法の最良解平均

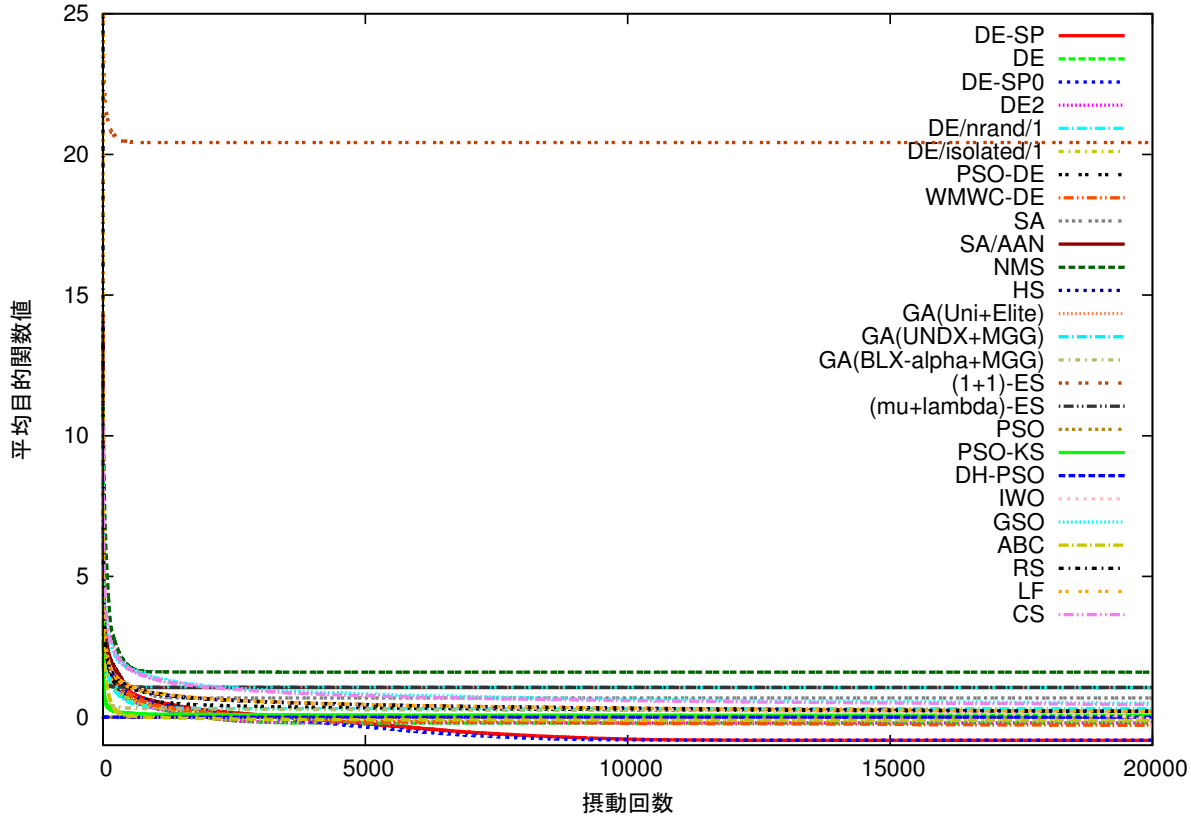


(a) 全体図

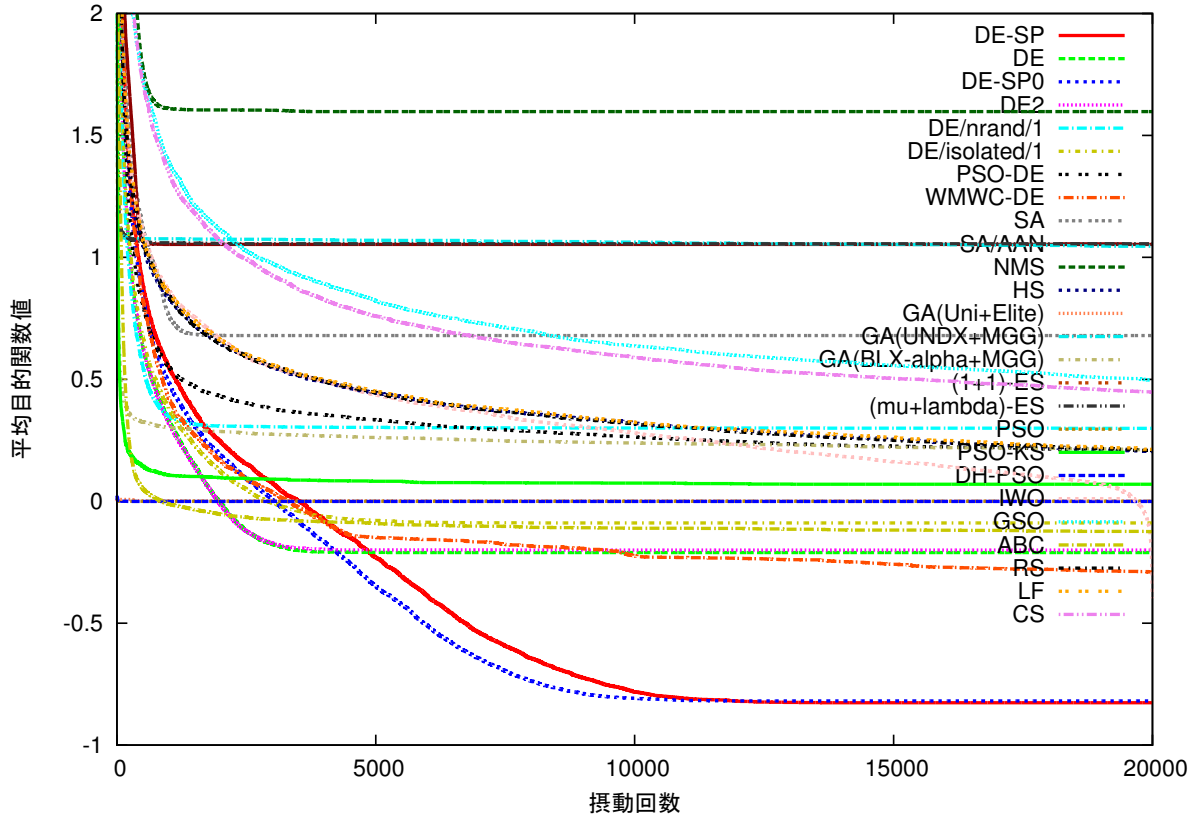


(b) 密集域拡大図

図 2.12: Ackley Function における各手法の最良解平均

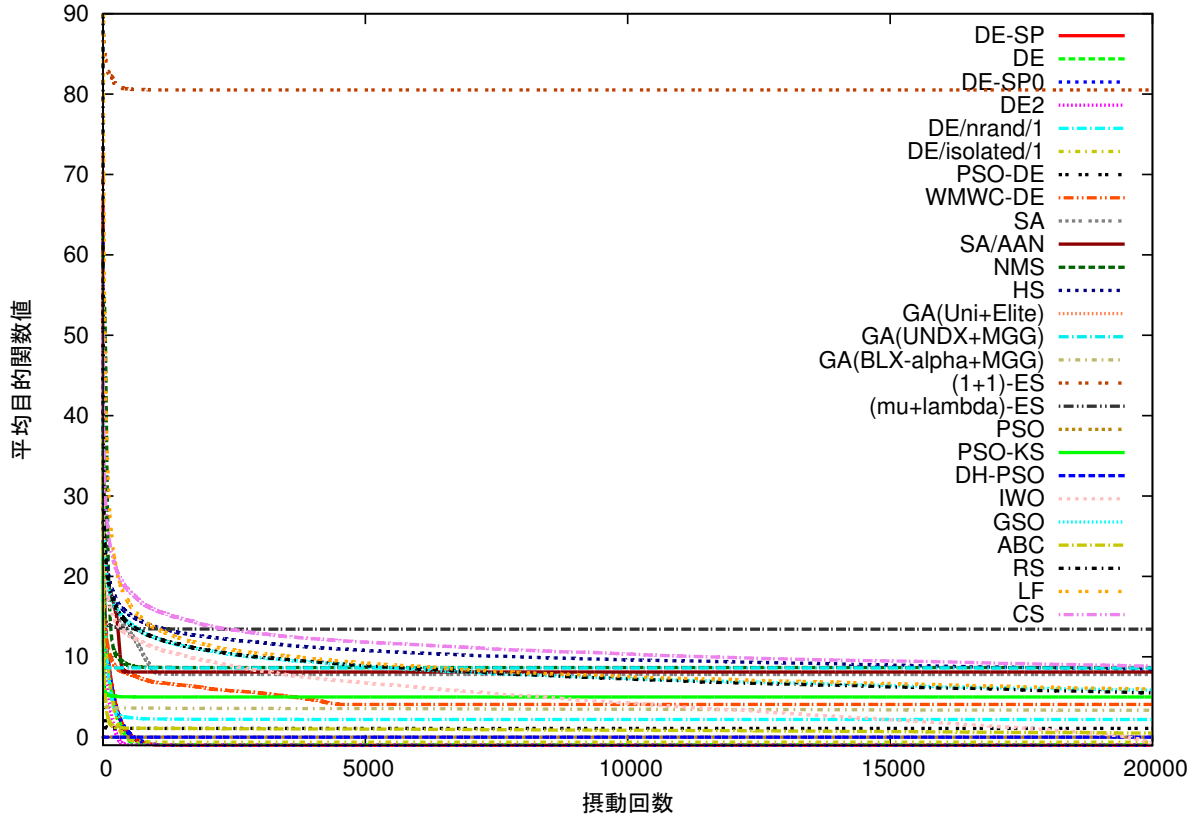


(a) 全体図

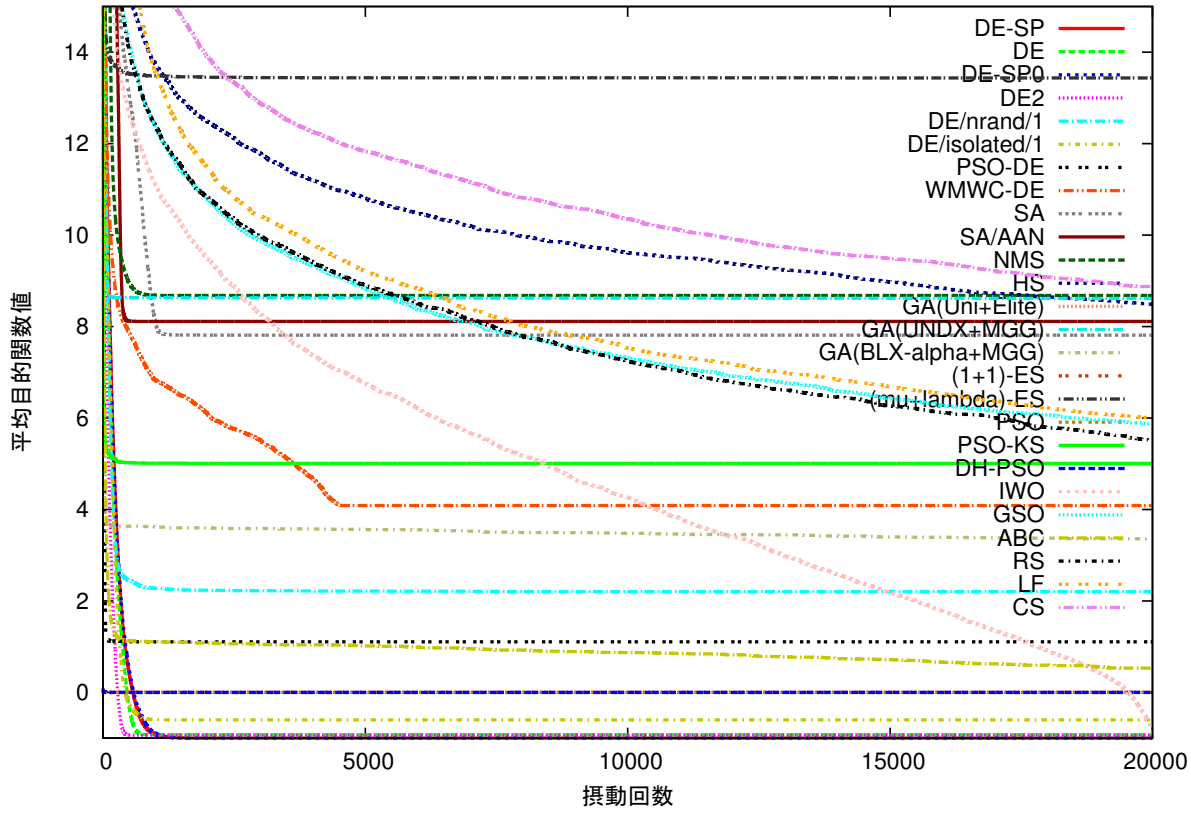


(b) 密集域拡大図

図 2.13: Noisy Function 1 における各手法の最良解平均



(a) 全体図



(b) 密集域拡大図

図 2.14: Noisy Function 2 における各手法の最良解平均

第3章

Differential Evolution on Scattered Parentsの実問題に対する有効性の評価

3.1　はじめに

前章においてDE-SP を提案し，ベンチマーク関数において有効性を確認した．しかしこの結果は必ずしも実問題において有効であることを示唆しないため，本章ではEngineering Optimization から取り上げた三つの擬実問題と，報告 [53] にて設計された二足ロボットの立位保持制御問題を用いた最適化実験を行う．これにより DE-SP の実問題での有効性を確認する．

3.2　無制約な最適化手法の制約付き連続値最適化問題への適用

DE-SP に限らず DE 類縁手法は原則として無制約の連続値最適化を行う手法であり，そのままでは本章で述べる実験にて取り上げる最適化問題に含まれる制約条件を扱うことができない．したがって，本研究ではDeb's Feasibility Rules [47] により大小比較の拡張を行う．これにより各最適化問題を見かけ上の無制約の連続値最適化問題へと置き換えることで，DE 類縁手法による最適化が可能となるようにする．また，Deb's Feasibility Rules は一つの制約条件のみ扱い，複数の制約条件には対応できないため，制約条件を取りまとめた拡大制約条件を設計する．さらに，一部の最適化問題は離散値を要素としている．この点も DE 類縁手法をそのまま適用できないため，ペナルティ関数を設計し要求される離散値に解を誘導することで，擬似的に離散値要素の最適化を実現する．このペナルティ関数も拡大制約条件に組み込まれる．

3.2.1　Deb's Feasibility Rules

Deb's Feasibility Rules [47] は，目的関数 $E(\boldsymbol{\chi})$ の大小比較を制約条件 $G(\boldsymbol{\chi})$ が $E(\boldsymbol{\chi})$ と独立して存在する場合でも適用可能とするために拡張した手法であり，以下の 3 つのルールにて表される．ただし，$E(\boldsymbol{\chi})$ は小さいほど良い値であり，$G(\boldsymbol{\chi})$ は $G(\boldsymbol{\chi}) \leq 0$ であれば制約条件が満たされていることを表すものとする．

- 任意の $G(\boldsymbol{\chi}^i) \leq 0$ な個体 $\boldsymbol{\chi}^i$ は任意の $G(\boldsymbol{\chi}^j) > 0$ である個体 $\boldsymbol{\chi}^j$ より良い個体である．
- 個体 $\boldsymbol{\chi}^i$，$\boldsymbol{\chi}^j$ において，$G(\boldsymbol{\chi}^i) \leq 0$ と $G(\boldsymbol{\chi}^j) \leq 0$ が共に満たされている時，$E(\boldsymbol{\chi})$ の値が小さい個体の方が良い個体である．
- 個体 $\boldsymbol{\chi}^i$，$\boldsymbol{\chi}^j$ において，$G(\boldsymbol{\chi}^i) > 0$ と $G(\boldsymbol{\chi}^j) > 0$ が共に満たされている時，$G(\boldsymbol{\chi})$ の値が小さい個体の方が良い個体である．

これを用いて目的関数の比較を目的関数と制約条件の比較に拡張し，制約条件を DE 類縁手法においても取り扱えるようにする．

3.2.2　拡大制約条件

本研究にて用いる拡大制約条件 G は，全制約条件値の加算値とする．ただし，制約を充足している（0 未満の値を取る）制約条件値は加算しない．

$$G(\boldsymbol{\chi}) = \sum_{j=1}^{N_G} R(g_j(\boldsymbol{\chi})) \tag{3.1}$$

ここで，N_G は制約条件の個数，R はランプ関数，$\boldsymbol{\chi}$ は注目している解である．ただし，各制約条件 g_j $(j = 1, 2, \ldots, N_G)$ は，$g_j(\boldsymbol{\chi}) \leq 0$ の形式であらわされるものとする．

3.2.3　離散値を連続値上で扱うためのペナルティ関数

本研究では離散値を連続値上で扱うためのペナルティ関数 g_p として次のものを用いる [54]．

$$g_p(\boldsymbol{\chi}) = \sum_{h \in \text{離散要素全体}} \frac{1}{2} \left[\sin \frac{2\pi \left\{ \chi_h - \frac{1}{4}(d_{h_+} - 3d_{h_-}) \right\}}{d_{h_+} - d_{h_-}} + 1 \right] \tag{3.2}$$

ここで，$\boldsymbol{\chi}$ は現在注目している解，d_{h_-}，d_{h_+} は，それぞれ現在の要素値以下の最大の離散値と，現在の要素値よりも大きな最小の離散値である[1]．

[1]実数を整数にする際の切り捨て値と切り上げ値に相当するもの

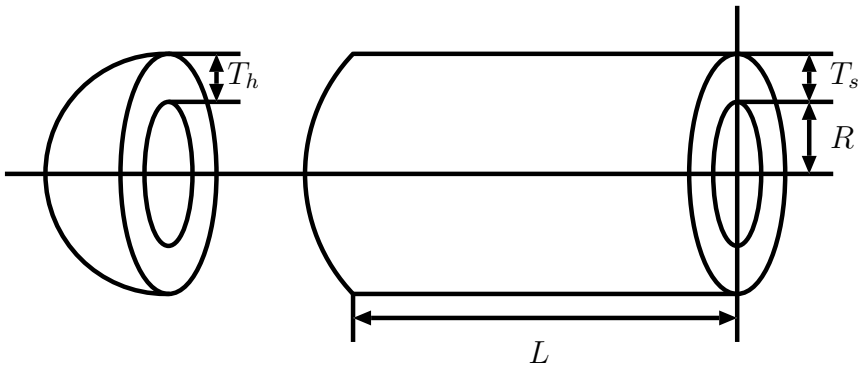


図 3.1: 圧力器の概念図

このペナルティ関数も拡大制約条件に組み込まれる．

3.3　最適化実験

3.3.1　Engineering Optimization

圧力器の最適設計問題

圧力器の最適設計問題は，機械設計における最適化問題の一つであり，Sandgren [3] により呈された問題である．本問題の目的は，図 3.1 に示される圧力容器の末端部位（以下，圧力器とする）の製作コストを最小化することである．図において，R は圧力器内の円筒状空間の直径，L は圧力器胴体部分の長さ，T_s は胴体部分の厚さ，T_h は蓋部分の厚さである．圧力器としての機能要件を満たし，また ASME の規格に準拠するよう T_s，T_h は 0.0625 の倍数しか取れないため，この問題は離散値と連続値の混在した制約付き最適化問題となる．

この圧力器の製作コストは次のように表される．

$$\mathrm{C_{ost}} = 2\pi D(C_s R T_s L + C_h R^2 T_h) + V_l D C_w + V_s D C_w \tag{3.3}$$

ここで，D は炭素鋼の密度，C_w は素材コスト，C_s は金属板を筒状に丸めるコスト，C_h は金属を鍛造するコストである．また，V_l，V_s はそれぞれ溶接コストに関わる値であり，次のように表される．

$$V_l = \pi \left(\frac{T_s}{\cos 30^\circ} \right)^2 \cdot \frac{60}{360} \cdot 2L \tag{3.4}$$

$$V_s = \pi \left(\frac{T_s}{\cos 30^\circ} \right)^2 \cdot \frac{60}{360} \cdot 4\pi R \tag{3.5}$$

D，C_w，C_s，C_h は変化し得る値だが，設計者が意図的に調整可能な値ではないため，本研究ではこれらを問題提案時に用いられた値に固定し，次の目的関数を用いることと

する．

$$E(R, L, T_s, T_h) = 0.6224RLT_s + 1.7781R^2T_h + 3.1661LT_s^2 + 19.84RT_s^2 \tag{3.6}$$

今後，各値を一まとめに扱うため，

$$\boldsymbol{\chi} = \begin{pmatrix} R \\ L \\ T_s \\ T_h \end{pmatrix} \tag{3.7}$$

$$E(\boldsymbol{\chi}) = 0.6224\chi_1\chi_2\chi_3^2 + 1.7781\chi_1^2\chi_4 + 3.1661\chi_2\chi_3^2 + 19.84\chi_1\chi_3^2 \tag{3.8}$$

とする．次に，定義域は次の通りである．

$$25 \leq \chi_1 \leq 150$$
$$25 \leq \chi_2 \leq 240$$
$$0.0625 \leq \chi_3, \chi_4\ (\in \mathbb{A}) \leq 1.25$$

ただし，$\mathbb{A}$ は実数値上に取られた離散値の集合であり，次のように表される．

$$\mathbb{A} = \{0.0625n \mid n \in \mathbb{N}\} \tag{3.9}$$

また，制約条件は次の通りである．

$$g_1(\boldsymbol{\chi}) = \frac{0.0193\chi_1}{\chi_3} - 1 \leq 0 \tag{3.10}$$

$$g_2(\boldsymbol{\chi}) = \frac{0.00954\chi_1}{\chi_4} - 1 \leq 0 \tag{3.11}$$

$$g_3(\boldsymbol{\chi}) = \frac{\chi_2}{240} - 1 \leq 0 \tag{3.12}$$

$$g_4(\boldsymbol{\chi}) = \frac{1296000 - \frac{4}{3}\pi\chi_1^3}{\pi\chi_1^2\chi_2} - 1 \leq 0 \tag{3.13}$$

減速機の最適設計問題

減速器の最適設計問題は，圧力器の最適設計問題と同様，機械設計における最適化問題の一つであり，Golinski [55] により呈された問題である．本問題の目的は，図 3.2 に示される減速器の総重量を最小化することである．図において，l_1，l_2 は各シャフトの長さ，d_1，d_2 はシャフトの直径，z は小歯車の歯の数とし，図中に記載されていないパラメータとして，b を有効歯幅，m を歯車の歯幅とする．減速器としての機能要件を満たすため，この問題は離散値と連続値の混在した制約付き最適化問題となる．まず，目的関数は次のように表される．

$$\begin{aligned} E(b, m, z, l_1, l_2, d_1, d_2) = {} & 0.7854bm^2(3.3333z^2 + 14.9334z - 43.0934) \\ & -1.508b(d_1^2 + d_2^2) + 7.477(d_1^3 + d_2^3) + 0.7854(l_1d_1^2 + l_2d_2^2) \end{aligned} \tag{3.14}$$

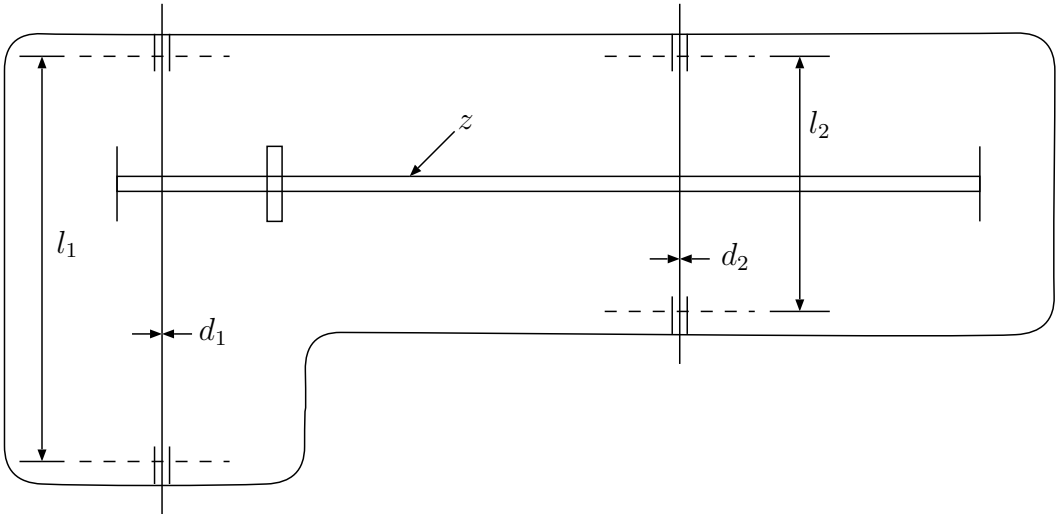


図 3.2: 減速器の概念図

今後，各値を一まとめに扱うため，

$$\boldsymbol{\chi} = \begin{pmatrix} b \\ m \\ z \\ l_1 \\ l_2 \\ d_1 \\ d_2 \end{pmatrix} \tag{3.15}$$

$$\begin{aligned} E(\boldsymbol{\chi}) &= 0.7854\chi_1\chi_2^2(3.3333\chi_3^2 + 14.9334\chi_3 - 43.0934) \\ &\quad -1.508\chi_1(\chi_6^2 + \chi_7^2) + 7.477(\chi_6^3 + \chi_7^3) + 0.7854(\chi_4\chi_6^2 + \chi_5\chi_7^2) \end{aligned} \tag{3.16}$$

とする．次に，定義域は次の通りである．

$$\begin{gathered} 2.6 \leq \chi_1 \leq 3.6 \\ 0.7 \leq \chi_2 \leq 0.8 \\ 17 \leq \chi_3\,(\in \mathbb{N}) \leq 28 \\ 7.3 \leq \chi_4 \leq 8.3 \\ 7.3 \leq \chi_5 \leq 8.3 \\ 2.9 \leq \chi_6 \leq 3.9 \\ 5.0 \leq \chi_7 \leq 5.5 \end{gathered}$$

また，制約条件は次の通りである．

$$g_1(\boldsymbol{\chi}) = \frac{27}{\chi_1\chi_2^2\chi_3} - 1 \leq 0 \tag{3.17}$$

$$g_2(\boldsymbol{\chi}) = \frac{397.5}{\chi_2\chi_2^2\chi_3^2} - 1 \leq 0 \tag{3.18}$$

$$g_3(\boldsymbol{\chi}) = \frac{1.93\chi_4^3}{\chi_2\chi_3\chi_6^4} - 1 \leq 0 \tag{3.19}$$

$$g_4(\boldsymbol{\chi}) = \frac{1.93\chi_5^3}{\chi_2\chi_3\chi_7^4} - 1 \leq 0 \tag{3.20}$$

$$g_5(\boldsymbol{\chi}) = \frac{1}{110\chi_6^3}\sqrt{\left(\frac{745.0\chi_4}{\chi_2\chi_3}\right)^2 + 1.69 \cdot 10^7} - 1 \leq 0 \tag{3.21}$$

$$g_6(\boldsymbol{\chi}) = \frac{1}{85\chi_7^3}\sqrt{\left(\frac{745.0\chi_5}{\chi_2\chi_3}\right)^2 + 1.575 \cdot 10^8} - 1 \leq 0 \tag{3.22}$$

$$g_7(\boldsymbol{\chi}) = \frac{\chi_2\chi_3}{40} - 1 \leq 0 \tag{3.23}$$

$$g_8(\boldsymbol{\chi}) = \frac{5\chi_2}{\chi_1} - 1 \leq 0 \tag{3.24}$$

$$g_9(\boldsymbol{\chi}) = \frac{\chi_1}{12\chi_2} - 1 \leq 0 \tag{3.25}$$

$$g_{10}(\boldsymbol{\chi}) = \frac{1.5\chi_6 + 1.9}{\chi_4} - 1 \leq 0 \tag{3.26}$$

$$g_{11}(\boldsymbol{\chi}) = \frac{1.1\chi_7 + 1.9}{\chi_5} - 1 \leq 0 \tag{3.27}$$

熔接梁の最適設計問題

熔接梁の最適設計問題は，前述の二つの問題と同様，機械設計における最適化問題の一つであり，Ragsdell ら [1] が GA を用いて解こうとした問題である．本問題の目的は図 3.3 に示される梁の熔接コストの最小化である．図において，x_1，x_2 は熔接代の高さと幅，x_3，x_4 はそれぞれ梁の高さと幅である．まず，目的関数は次のように表される．

$$E(x_1, x_2, x_3, x_4) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{3.28}$$

今後，各値を一まとめに扱うため，

$$\boldsymbol{\chi} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \tag{3.29}$$

$$E(\boldsymbol{\chi}) = 1.10471\chi_1^2\chi_2 + 0.04811\chi_3\chi_4(14.0 + \chi_2) \tag{3.30}$$

とする．次に定義域は次の通りである．

$$0.1 \leq \chi_1, \chi_4 \leq 2.0 \tag{3.31}$$

$$0.1 \leq \chi_2, \chi_3 \leq 10.0 \tag{3.32}$$

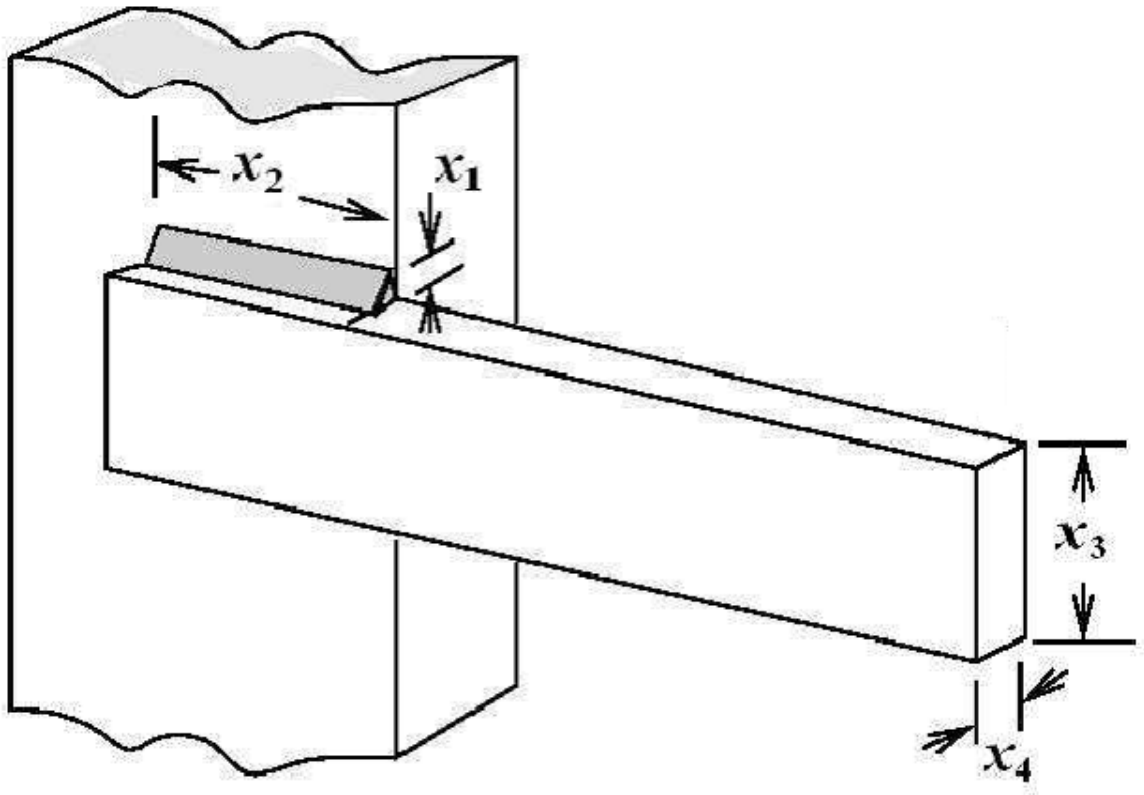


図 3.3: 熔接梁の概念図（文献 [1] より引用）

また，制約条件は次の通りである．

$$g_1(\boldsymbol{\chi}) = \tau(\boldsymbol{\chi}) - 13600 \leq 0 \tag{3.33}$$
$$g_2(\boldsymbol{\chi}) = \sigma(\boldsymbol{\chi}) - 30000 \leq 0 \tag{3.34}$$
$$g_3(\boldsymbol{\chi}) = \chi_1 - \chi_4 \leq 0 \tag{3.35}$$
$$g_4(\boldsymbol{\chi}) = 0.10471\chi_1^2 + 0.04811\chi_3\chi_4(14 + \chi_2) - 5.0 \leq 0 \tag{3.36}$$
$$g_5(\boldsymbol{\chi}) = 0.125 - \chi_1 \leq 0 \tag{3.37}$$
$$g_6(\boldsymbol{\chi}) = \delta(\boldsymbol{\chi}) - 0.25 \leq 0 \tag{3.38}$$
$$g_7(\boldsymbol{\chi}) = 6000 - P_c(\boldsymbol{\chi}) \leq 0 \tag{3.39}$$

ここで，τ は剪断応力であり，次のように表される．

$$\tau(\boldsymbol{\chi}) = \sqrt{(\tau'(\boldsymbol{\chi}))^2 + (2\tau'(\boldsymbol{\chi})\tau''(\boldsymbol{\chi}))\frac{\chi_2}{2R} + (\tau''(\boldsymbol{\chi}))^2} \tag{3.40}$$

$$\tau'(\boldsymbol{\chi}) = \frac{6000}{\sqrt{2}\chi_1\chi_2} \tag{3.41}$$

$$\tau''(\boldsymbol{\chi}) = \frac{M(\boldsymbol{\chi})R(\boldsymbol{\chi})}{J(\boldsymbol{\chi})} \tag{3.42}$$

$$M(\boldsymbol{\chi}) = 6000\left(14 + \frac{\chi_2}{2}\right) \tag{3.43}$$

$$R(\boldsymbol{\chi}) = \sqrt{\frac{\chi_2^2}{4} + \left(\frac{\chi_1 + \chi_3}{2}\right)^2} \tag{3.44}$$

$$J(\boldsymbol{\chi}) = 2\left[\chi_1\chi_2 \cdot \sqrt{2}\left\{\frac{\chi_2^2}{12} + \left(\frac{\chi_1 + \chi_3}{2}\right)^2\right\}\right] \tag{3.45}$$

$$\tag{3.46}$$

σ は湾曲負荷であり，次のように表される．

$$\sigma(\boldsymbol{\chi}) = \frac{504000}{\chi_4\chi_3^2} \tag{3.47}$$

δ は梁の撓みであり，次のように表される．

$$\delta(\boldsymbol{\chi}) = \frac{65856000}{3 \cdot 10^7\chi_4\chi_3^3} \tag{3.48}$$

P_c は座屈荷重であり，次のように表される．

$$P_c(\boldsymbol{\chi}) = \frac{4.013(3 \cdot 10^7)\sqrt{\frac{\chi_3^2\chi_4^6}{36}}}{196}\left(1 - \frac{\chi_3\sqrt{\frac{3 \cdot 10^7}{48 \cdot 10^6}}}{28}\right) \tag{3.49}$$

3.3.2　二足ロボットの立位保持制御

実問題の一つとして，二足ロボットの立位保持制御問題を扱う．本実験では，脚関節部モータのトルクが測定値微分先行型 PID 制御（付録 B.1 を参照）によって制御される二足ロボットの立位を維持することを目的とする．このため，本実験では立位の保持が可能となるよう各制御器の PID ゲインの最適化を行う．ロボットのリンク構造は，図 3.4 に示す通り，股関節 2 自由度，膝 1 自由度，足首 2 自由度の計 10 自由度（片足 5 自由度）とし，各脚の部位が共通しているモータのゲインは共有するものとする．このため，本ロボットモデルにおいて最適化対象とする PID ゲインは計 15 個となる．また，各関節は直立状態を初期状態とし，これらの大きさと重量情報は表 3.1 に，可動域は表 3.2 に示す．ロボットの動作確認は動力学シミュレータ Open Dynamics Engine（Ver. 0.11.1）[56] を用いて行った．

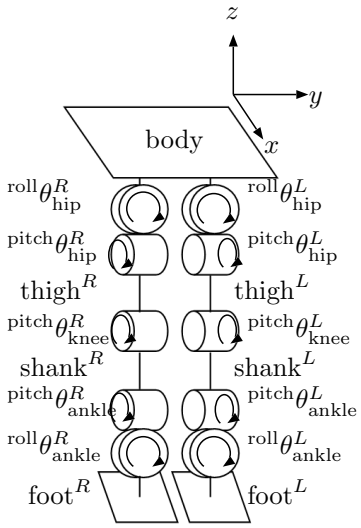


図 3.4: 二足ロボットのリンク構造図

表 3.1: 二足ロボットの各リンクの大きさと重量情報

	(x, y, z)	重量
body	$(2.0, 2.0, 2.0)$	10.0
thigh^{R,L}	$(0.2, 0.2, 0.8)$	2.5
shank^{R,L}	$(0.2, 0.2, 0.8)$	2.5
foot^{R,L}	$(1.0, 0.5, 0.05)$	1.0

表 3.2: 二足ロボットの各関節可動域

	roll $\theta^{R,L}_{\text{hip}}$	pitch $\theta^{R,L}_{\text{hip}}$	pitch $\theta^{R,L}_{\text{knee}}$	pitch $\theta^{R,L}_{\text{ankle}}$	roll $\theta^{R,L}_{\text{ankle}}$
Min[rad]	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{2\pi}{3}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
Max[rad]	$\frac{\pi}{2}$	$\frac{\pi}{2}$	0.02	$\frac{\pi}{2}$	$\frac{\pi}{2}$

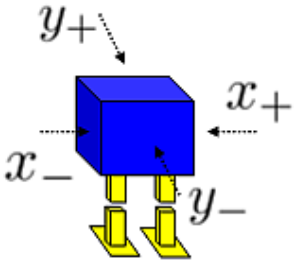


図 3.5: 各 $\mathcal{D}$ の要素の図示

本実験では，ロボットがより安定した立位を保持する事を目的とし，立位が安定であるほど，そしてより速やかに安定となるほど小さな値を示す目的関数 E を下式の通り設定する．

$$E(\boldsymbol{\chi}) = \sum_{d \in \mathcal{D}} e^{\beta_1^d} \sum_{t=0}^{U-1} |S(\boldsymbol{\chi}, d, t)|^2 \tag{3.50}$$

ここで，$\boldsymbol{\chi}$ はロボットの全 PID ゲインを格納した 15 次元ベクトル，d はロボット上体に加える外力の方向とし，モータを制御しないという局所解に陥ることを防止するために用いる．β_1^d は $|S(\boldsymbol{\chi}, d, t)|^2$ の時系列データに対して単回帰直線を引くときに求められる傾きであり，$|S(\boldsymbol{\chi}, d, t)|^2$ の時系列データのおおよその傾きとして用いる．$\mathcal{D}$ は d の全候補 $\{x_+, x_-, y_+, y_-\}$ であり，具体的な方向は図 3.5 に示す．U はシミュレーション最大時間，$S(\boldsymbol{\chi}, d, t)$ は個体 $\boldsymbol{\chi}$ に対して方向 d に力を加えた場合の時刻 t における安定性指標の値である．本実験では，この安定性指標として，下式に従いロボット重心の位置と速度から計算される値 [57] を用いる．

$$S(\boldsymbol{\chi}, d, t) = \boldsymbol{q}_{\boldsymbol{\chi},d}(t) + \sqrt{\frac{h}{g}} \left(\frac{d}{dt} \boldsymbol{q}_{\boldsymbol{\chi},d}(t) \right) \tag{3.51}$$

ここで，$\boldsymbol{q}_{\boldsymbol{\chi},d}(t)$ は，個体 $\boldsymbol{\chi}$ のゲインを適応したロボットに対して方向 d に力を加えた場合の時刻 t における重心を床面に正射影した点，h はロボットの高さ，g は重力定数であ

る．この安定性指標 $S(\boldsymbol{\chi}, d, t)$ は，支持多角形の幾何的中心を原点とするとき，小さなノルムのベクトルであるほど立位が安定であり，大きな値であるほど立位が不安定であると判断する目安となる．また，この目的関数 $E(\boldsymbol{\chi})$ は安定した立位の実現以外に，ロボットが速やかに転倒し静止することでも小さな値を取る場合がある．この局所解に陥ることを防止するため，本実験では次の制約条件を用いて，ロボットが転倒する解の受理を防ぐ．

$$G(\boldsymbol{\chi}) = \sum_{d \in \mathcal{D}} \left(U - T^{\mathrm{tamble}}_{\boldsymbol{\chi},d} \right) = 0 \tag{3.52}$$

ここで，$T^{\mathrm{tamble}}_{\boldsymbol{\chi},d}$ は方向 d へ押した場合にロボットが転倒した時刻であり，シミュレーション中（時間が U 経過する間）に転倒しなければ $T^{\mathrm{tamble}}_{\boldsymbol{\chi},d} = U$ となる．全ての方向 d からの外力に対しロボットが転倒しなければ，$G(\boldsymbol{\chi}) = 0$ となる．

本問題の目的関数の最適化は，先行研究 [53] で行った調査で得られた知見から，立位の維持が可能な解は Nelder-Mead Simplex Method [15] のような大域的単峰性を仮定した最適化手法によっても得られると考えられる．しかしながら，更に安定した立位保持が可能な解の探索は，わずかなパラメータ変化により安定状況が大きく変化する場合があるため，困難であると考えられる．

3.3.3　実験設定

圧力器・減速器・熔接梁の最適設計問題

実験において，個体数 $N = 50$，摂動回数 20000 回とし，比較手法は前章で取り上げた DE 類縁手法の中から WMWC-DE と改良点の単一実装手法を取り除いた，DE-SP，DE，DE/nrand/1，DE/isolated/1，PSO-DE の 5 手法とする．F，C_R は各手法に対して小規模の総当たり実験を行い，その他のパラメータは各手法が解説されている文献を参考に設定した．これら実験時に設定したパラメータは付録 B.2.1，B.2.2，B.2.3 に示す．

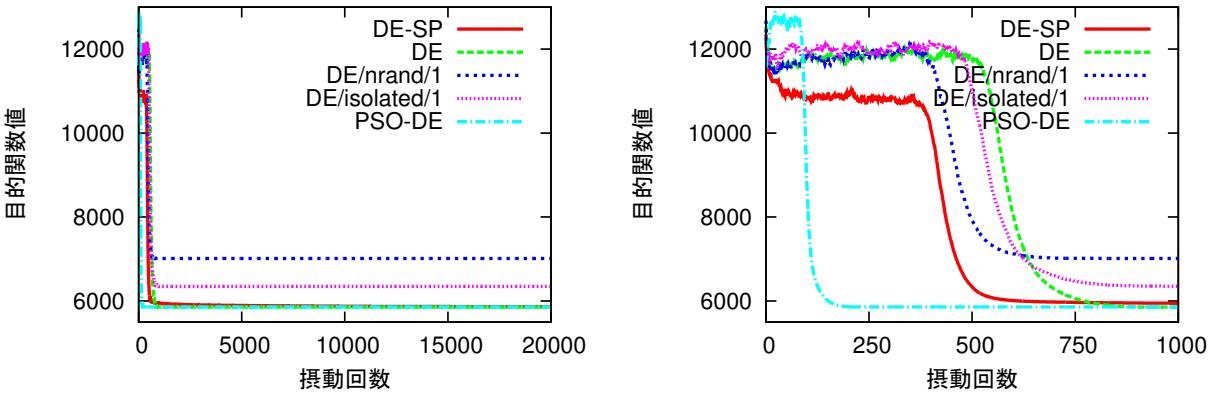
実験は 1000 試行行い，発見された最良解と解の 1000 試行平均を性能の指標として比較する．

二足ロボットの立位保持制御問題

実験において，個体数 $N = 50$，摂動回数 5000 回とし，比較手法は前節と同一の 6 手法とする．各手法に設定するパラメータは F，C_R は各手法に対して，$F \in \{0.3, 0.5, 0.7\}$，$C_R \in \{0.1, 0.5, 0.9\}$ の中で総当たりを行い，探索されたものの中から最良のものを用いる．その他のパラメータは各手法が解説されている文献を参考に設定した．これら実験時に設定したパラメータは付録 B.2.4 に示す．また動力学シミュレータの設定は，$U = 12000$，方向 d に力を加える時刻は $t = 800$ とし，t が一経過することと現実世界で 2.5[msec] 経過することが対応するものとする．

表 3.3: 圧力器の最適化問題における最良の $E(\boldsymbol{\chi})$ と $G(\boldsymbol{\chi})$.

	DE-SP	DE	DE/nrand/1	DE/isolated/1	PSO-DE
$E(\boldsymbol{\chi})$ （最良値）	**5850.38**	**5850.38**	**5850.38**	**5850.38**	**5850.38**
$E(\boldsymbol{\chi})$ （平均値）	5879.52	**5852.06**	7104.45	5893.93	5854.90
$G(\boldsymbol{\chi})$ （最良値）	0	0	0	0	0
$G(\boldsymbol{\chi})$ （平均値）	0	0	0	0	0



(a) 全体図　(b) 大きな変動が見られる部分の拡大図

図 3.6: 圧力器の最適化実験により得られた最良の $E(\boldsymbol{\chi})$ の平均

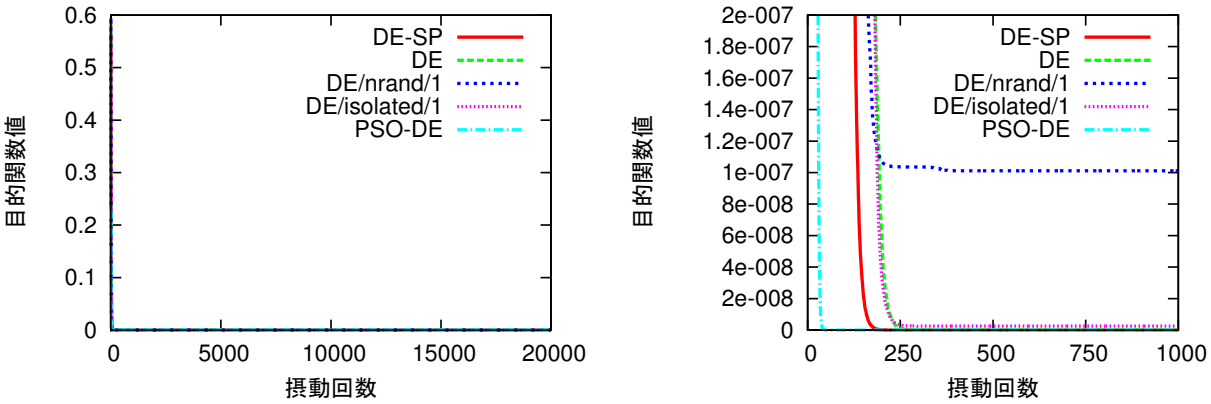
実験は 10 試行行い，発見された最良解を性能の指標として比較する．

3.3.4　実験結果と考察

圧力器の最適設計問題

実験結果として，各手法を用いて最適化された目的関数値 $E(\boldsymbol{\chi})$ の最良値と平均値，制約条件 $G(\boldsymbol{\chi})$ の最良値と平均値の値を表 3.3 に示す．この結果から，最良値は全手法において同一であるが，平均値は僅差で DE が最も良いことが確認される．また，制約条件値はどの手法においても違反していないため，これらの結果の信頼性は保証される．DE がこの問題において最も良い性能を残した理由として，格子状に局所解が配置されている問題は DE に有利な問題である，という仮説を立てている．すなわち，圧力器の最適化問題は χ_3，χ_4 に適用されるペナルティ関数 g_p （式 (3.2) 参照）によって格子状に配置される局所解が存在するため，DE に有利なのではないかという仮説である．これは，2.4 節にて図 2.2 を用いて行った説明と対になるものであり，DE の格子状に探索点候補が固まりやすい性質により，格子状に局所解が配置されている問題において有利に探索可能なのではないかという予想である．この予想はこの問題だけでなく，付録 B.3 の実験結果も参考としている[2]．

[2]この問題は計算に莫大な時間を必要とするために収集された実験結果が，最適化結果として考察するには十分な量が揃わなかったため，付録に掲載するに留める．



(a) 全体図　(b) 大きな変動が見られる部分の拡大図

図 3.7: 圧力器の最適化実験により得られた最良の $G(\boldsymbol{\chi})$ の平均

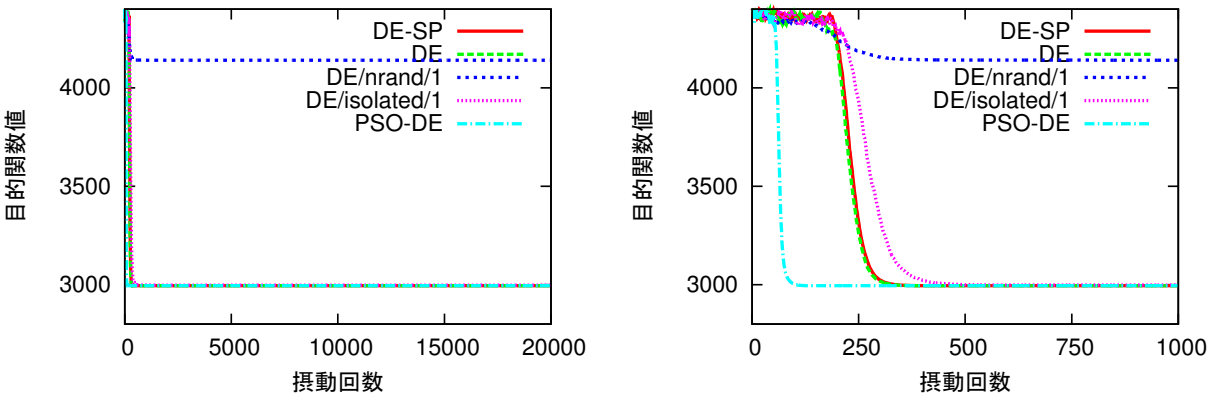
表 3.4: 減速器の最適化問題における最良の $E(\boldsymbol{\chi})$ と $G(\boldsymbol{\chi})$.

	DE-SP	DE	DE/nrand/1	DE/isolated/1	PSO-DE
$E(\boldsymbol{\chi})$（最良値）	**2994.67**	**2994.67**	**2994.67**	**2994.67**	**2994.67**
$E(\boldsymbol{\chi})$（平均値）	**2994.67**	**2994.67**	4140.48	**2994.67**	**2994.67**
$G(\boldsymbol{\chi})$（最良値）	0	0	0	0	0
$G(\boldsymbol{\chi})$（平均値）	0	0	0	0	0

実験を 1000 試行行うことによって得られた最良の $E(\boldsymbol{\chi})$ の平均を図 3.6 に，最良の $G(\boldsymbol{\chi})$ の平均を図 3.7 に示す．本実験では Deb's Feasibility Rules を用いたため，探索の序盤では主に $G(\boldsymbol{\chi})$ の最適化が行われる．この影響が最良の $E(\boldsymbol{\chi})$ の平均のがたつきとして表れていることが確認される．これを踏まえた上で，最良の $G(\boldsymbol{\chi})$ の平均が最も早く 0 へ収束し，結果として最良の $E(\boldsymbol{\chi})$ の平均が速やかに最良解へ収束していることから，最良解を発見する速さの面では PSO-DE が最も優れていることが確認できる．これに続いて DE-SP，DE/nrand/1，DE/isolated/1，DE の順で序盤の探索が速やかに進んでいることが確認される．ただし，探索がある程度進んだ後は DE が DE/nrand/1 と DE/isolated/1 を追い越して最良解へ収束している．また，DE/nrand/1 の最良の $G(\boldsymbol{\chi})$ の平均は最終値こそ 0 へ収束しているものの，その速度は他の手法と比較して最も遅いことが確認される．

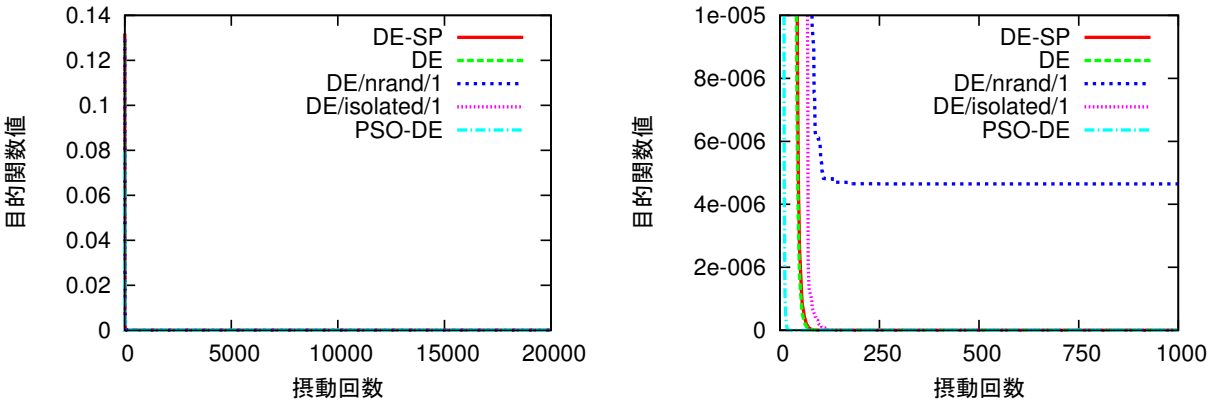
減速器の最適設計問題

実験結果として，各手法を用いて最適化された目的関数値 $E(\boldsymbol{\chi})$ の最良値と平均値，制約条件 $G(\boldsymbol{\chi})$ の最良値と平均値の値を表 3.4 に示す．この結果から，DE/nrand/1 の平均値が比較的悪い点除き，全手法において同等の性能を示していることが確認される．また，$E(\boldsymbol{\chi})$ の最良値と平均値が一致していることから，安定して最良値を発見していることが確認される．制約条件値はどの手法においても違反していないため，これらの結果の信頼性は保証される．



(a) 全体図　(b) 大きな変動が見られる部分の拡大図

図 3.8: 減速器の最適化実験により得られた最良の $E(\boldsymbol{\chi})$ の平均



(a) 全体図　(b) 大きな変動が見られる部分の拡大図

図 3.9: 減速器の最適化実験により得られた最良の $G(\boldsymbol{\chi})$ の平均

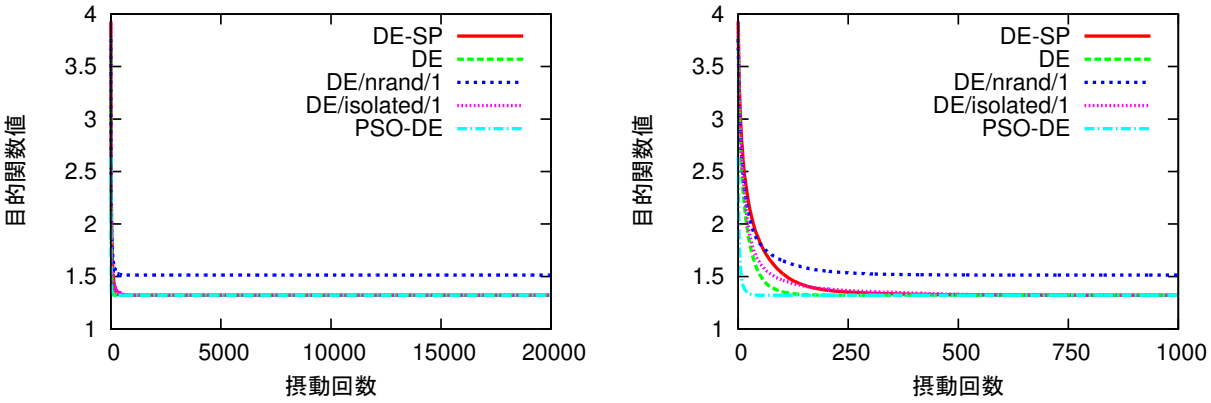
実験を 1000 試行行うことによって得られた最良の $E(\boldsymbol{\chi})$ の平均を図 3.8 に，最良の $G(\boldsymbol{\chi})$ の平均を図 3.9 に示す．圧力器の最適化問題の時と同様に PSO-DE が最も速やかに最良解に収束しており，DE-SP と DE はほぼ同等の速度で探索が行われていることも確認される．一方で，DE/nrand/1 の最良の $E(\boldsymbol{\chi})$ の平均は，他の手法と比較して初期解からの改善が小さく，最良の $G(\boldsymbol{\chi})$ の平均も序盤では 0 へ収束し切れていないことが確認される．

熔接梁の最適設計問題

実験結果として，各手法を用いて最適化された目的関数値 $E(\boldsymbol{\chi})$ の最良値と平均値，制約条件 $G(\boldsymbol{\chi})$ の最良値と平均値の値を表 3.5 に示す．この結果から，DE/nrand/1 の平均値が比較的悪い点除き，全手法において同等の性能を示していることが確認される．また，$E(\boldsymbol{\chi})$ の最良値と平均値が一致していることから，安定して最良値を発見していることが確認される．制約条件値はどの手法においても違反していないため，これらの結果の信頼性は保証される．

表 3.5: 熔接梁の最適化問題における最良の $E(\boldsymbol{\chi})$ と $G(\boldsymbol{\chi})$.

	DE-SP	DE	DE/nrand/1	DE/isolated/1	PSO-DE
$E(\boldsymbol{\chi})$（最良値）	**1.3210**	**1.3210**	**1.3210**	**1.3210**	**1.3210**
$E(\boldsymbol{\chi})$（平均値）	**1.3210**	**1.3210**	1.5132	**1.3210**	**1.3210**
$G(\boldsymbol{\chi})$（最良値）	0	0	0	0	0
$G(\boldsymbol{\chi})$（平均値）	0	0	0	0	0



(a) 全体図　(b) 大きな変動が見られる部分の拡大図

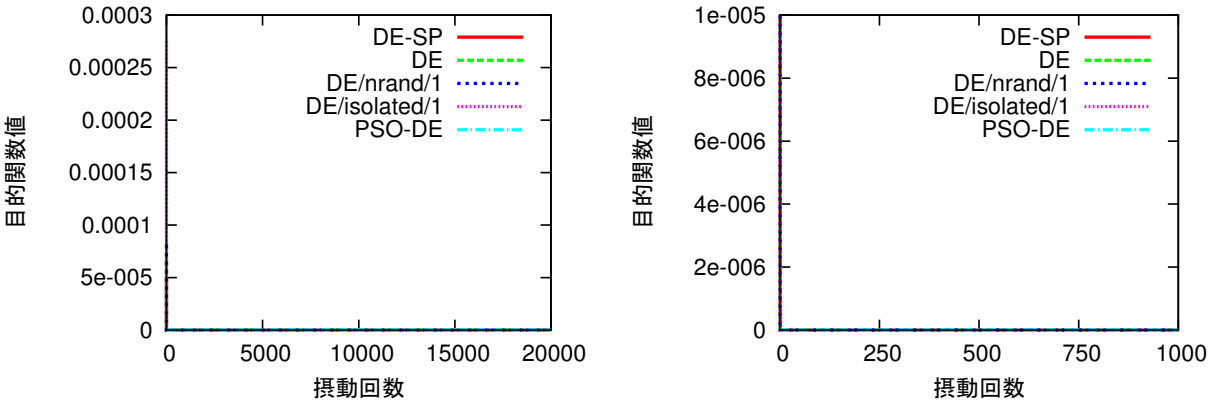
図 3.10: 熔接梁の最適化実験により得られた最良の $E(\boldsymbol{\chi})$ の平均

実験を 1000 試行行うことによって得られた最良の $E(\boldsymbol{\chi})$ の平均を図 3.8 に，最良の $G(\boldsymbol{\chi})$ の平均を図 3.9 に示す．

結果の傾向は前述の二つの実験結果と同等であるが，図 3.10(b) より，DE/isolated/1 が DE-SP よりも速く最良解へ収束していることが確認される．また，図 3.11(b) より，全手法において最良の $G(\boldsymbol{\chi})$ の平均が前述の二つの問題と比較して速やかに 0 へ収束しているため，この問題の拡大制約条件は比較的達成されやすいものであると推察することができる．

二足ロボットの立位保持制御問題

実験結果として，各手法における最適化された目的関数値 $E(\boldsymbol{\chi})$ と制約条件 $G(\boldsymbol{\chi})$ の値を表 3.6 に示す．この結果から，DE-SP と全比較手法において $G(\boldsymbol{\chi}) = 0$ であり，全ての手法で二足ロボットの立位姿勢保持に成功していることが認められる．全手法の中では，DE-SP が最も小さな $E(\boldsymbol{\chi})$ の発見に成功しており，実問題を最適化する手法として有効であることが示唆される．また，DE-SP($M = 0$) の $E(\boldsymbol{\chi})$ はほぼ DE と差がなく，DE2 の $E(\boldsymbol{\chi})$ は DE，DE/nrand/1，DE/isolated/1，WMWC-DE よりも悪い値となっている．したがって，DE-SP の両改良点（アルゴリズム 2，18 と 28 行目）が共に作用することで，本問題における DE-SP の探索性能が向上したと考えられる．一方で，PSO-DE は他の比較手法と比べて $E(\boldsymbol{\chi})$ の値が 3 倍以上となっており，最適化能力が極めて低いことが認められる．PSO-DE の最適化の過程を確認すると，$E(\boldsymbol{\chi})$ が改悪される方向へ



(a) 全体図　(b) 大きな変動が見られる部分の拡大図

図 3.11: 熔接梁の最適化実験により得られた最良の $G(\boldsymbol{\chi})$ の平均

表 3.6: 二足ロボットの立位保持制御問題における最良の $E(\boldsymbol{\chi})$ と $G(\boldsymbol{\chi})$ $[\times 10^{-4}]$.

	DE-SP	DE	DE/nrand/1	DE/isolated/1	PSO-DE
$E(\boldsymbol{\chi})$	**6.91**	7.12	7.83	7.11	27.1
$G(\boldsymbol{\chi})$	0	0	0	0	0

の移動が多く，目的関数値の低い個体の周辺にほとんど個体が存在しない状況が多く認められた．このため PSO-DE では，本問題における探索の効率的な進行が困難であったと考えられる．

ここで，各手法で最適化されたゲインを入力として実験と同値の条件で動力学シミュレータを稼働した場合の，ロボットの重心位置の軌跡を図 3.12，図 3.13，図 3.14，図 3.15 に示す．ただし，これらの図においてロボットの正面は右方向（x 軸の + 方向）である．また，図が平らとなって視認性が顕著に悪くなるため，x 軸と y 軸の縮尺は統一していない．この結果を見ると，左右方向からロボットを押した場合は，他の手法と比較して PSO-DE が中央周辺に重心を留めることができているが，前後方向から押した場合は中央周辺を周回するような軌道を取り，中央に重心が戻っていないことが確認できる．このため，PSO-DE は他の手法よりも悪い局所解の段階で探索が停滞したと考えられる．DE/nrand/1 においては，目的関数値は PSO-DE よりも良いが，四方向全ての方向の場合において他の手法よりも大きく重心を崩されている．DE，DE-SP，DE/isolated/1 においては，左右方向から押された場合はやや DE/isolated/1 が最も重心を崩されておらず，中央に戻りつつあるが，小刻みに旋回しながらであることが確認でき，DE と DE-SP は同等の軌跡を描いていることが確認できる．前後方向から押した場合は押された直後の重心の崩れは DE/isolated/1 が最も良いが，戻る際に大きく左右に振れており，また重心を中央へ戻す際に崩れた方向とは逆に行き過ぎてしまっている．一方 DE は DE-SP や DE/isolated/1 よりも前後に揺れていることが確認できる．DE-SP は DE と DE/isolated/1 の中間を取るような軌跡を描いているが，重心が中央に戻る際の行き過ぎ量は最も少ないことが確認できる．

DE-SP の軌跡が DE と DE/isolated/1 の中間のような軌跡と捉えられるようなもので

あることと，DE-SP の目的関数値が最も低いことから，DE-SP の発見したゲインによるこの軌跡は四方全ての外乱に対してロボットを安定させる良いトレードオフとなっている可能性がある．

3.4　おわりに

本調査では，DE 類縁手法と前章での提案手法 DE-SP を用い，圧力器，減速器，熔接梁，二足ロボットの立位保持制御問題における最適化実験を行った．この結果，圧力器の問題では DE が僅差で優れており，二足ロボットの立位保持制御問題では DE-SP が最良であることが確認された．したがって，問題の複雑さ次第により DE と DE-SP は使い分けられるべきであることが示唆される．ただし，三つの Engineering Optimization において DE-SP の最良解は DE と同一であり，最良の $E(\boldsymbol{\chi})$，$G(\boldsymbol{\chi})$ の平均の推移から探索の継続能力は DE と同程度であることが示唆されるため，複雑さが比較的低いような問題においても DE-SP は十分に実用可能であることは予想されるだろう．また，最良の $E(\boldsymbol{\chi})$，$G(\boldsymbol{\chi})$ の平均の結果から，本調査で扱った三つの Engineering Optimization 規模の問題では PSO-DE の収束性能が最も高いが，この手法は二足ロボットの立位保持制御問題規模の問題では有効に機能しなかった．このため，PSO-DE は複雑度の比較的高い問題においては適用困難であることが示唆された．

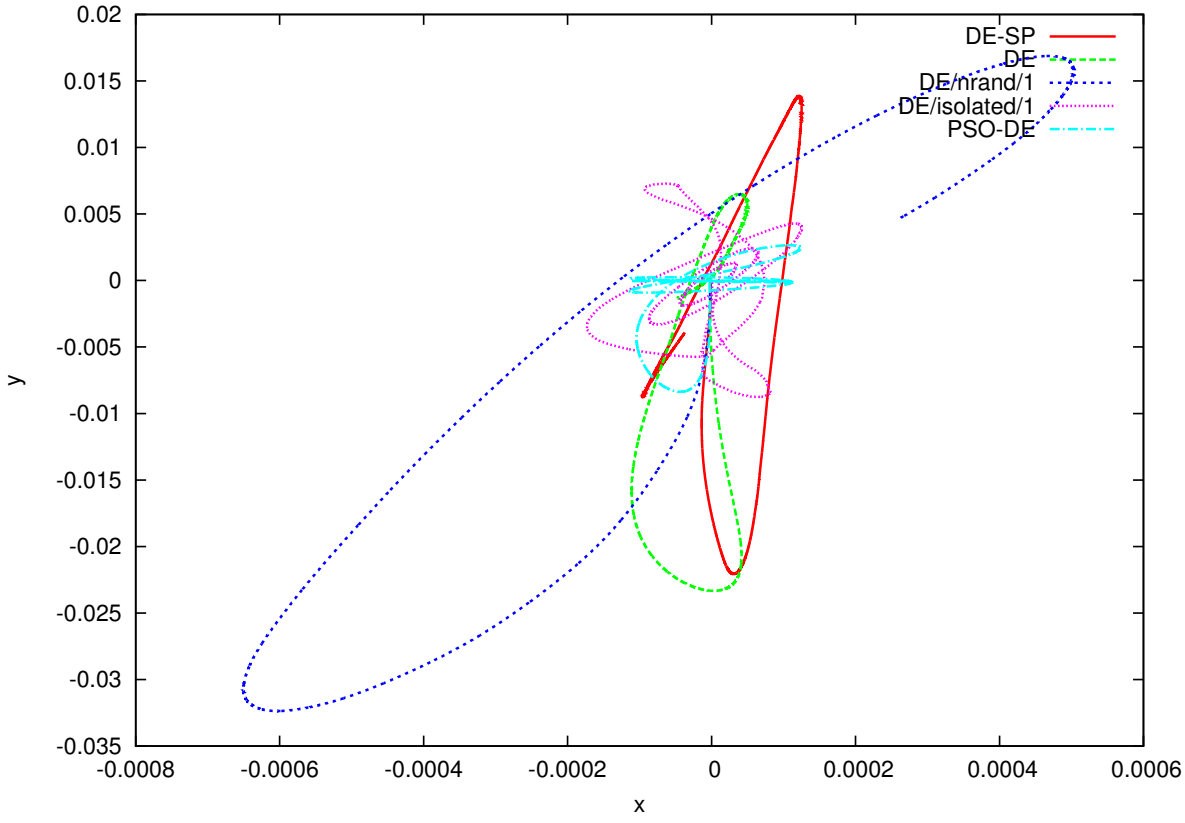


図 3.12: ロボットを右側（図中上）から押した場合の重心軌跡

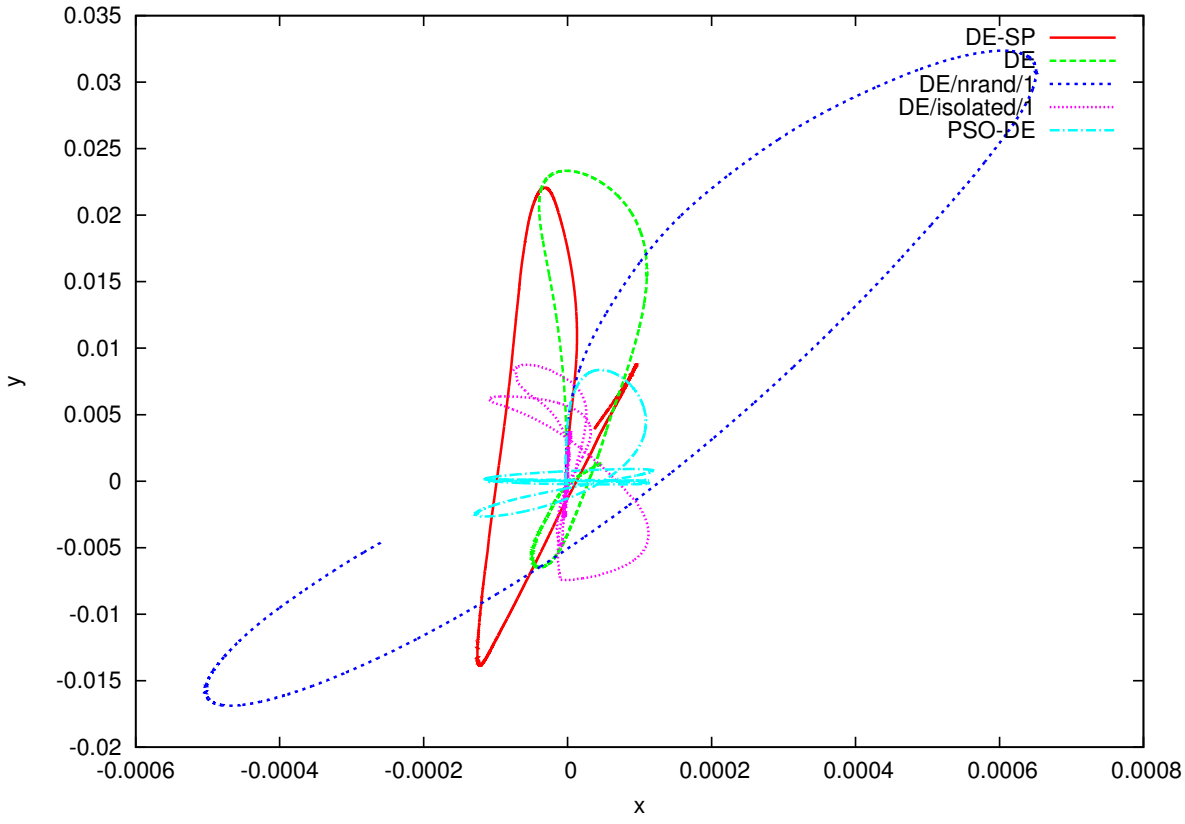


図 3.13: ロボットを左側（図中下）から押した場合の重心軌跡

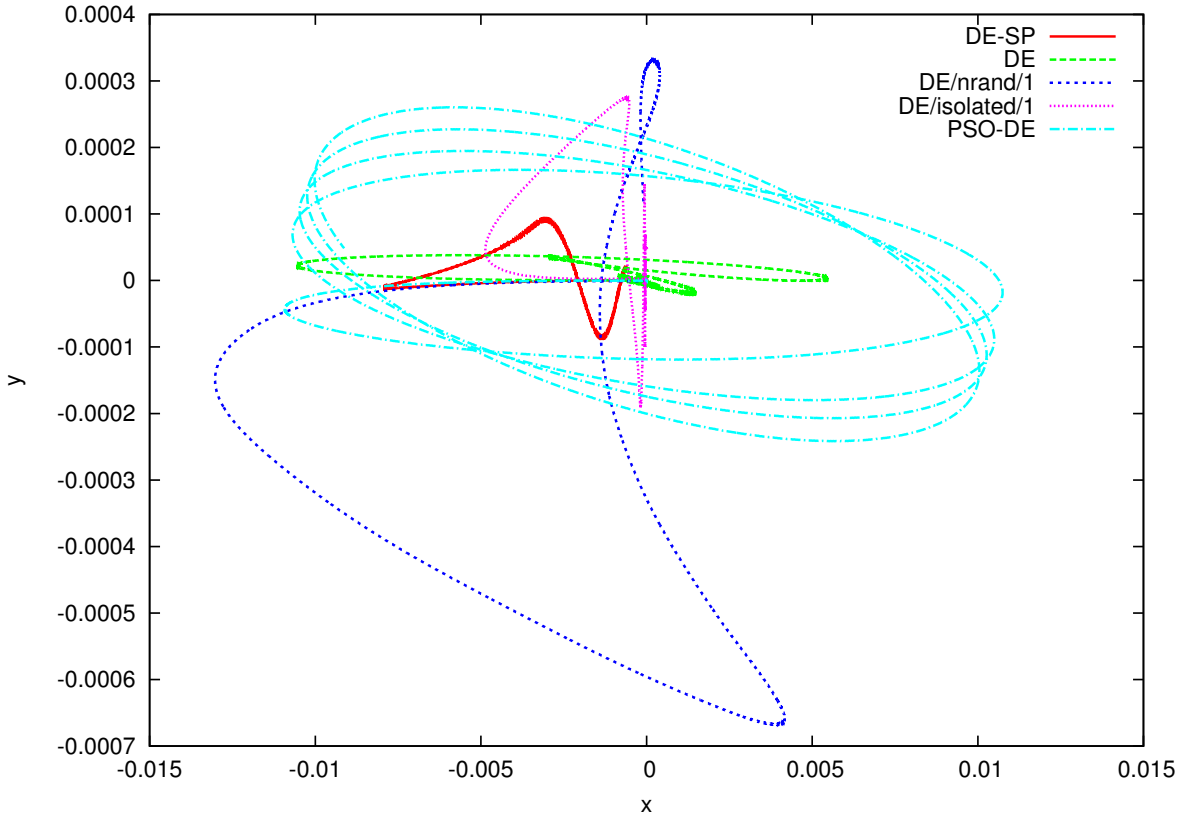


図 3.14: ロボットを正面側（図中右）から押した場合の重心軌跡

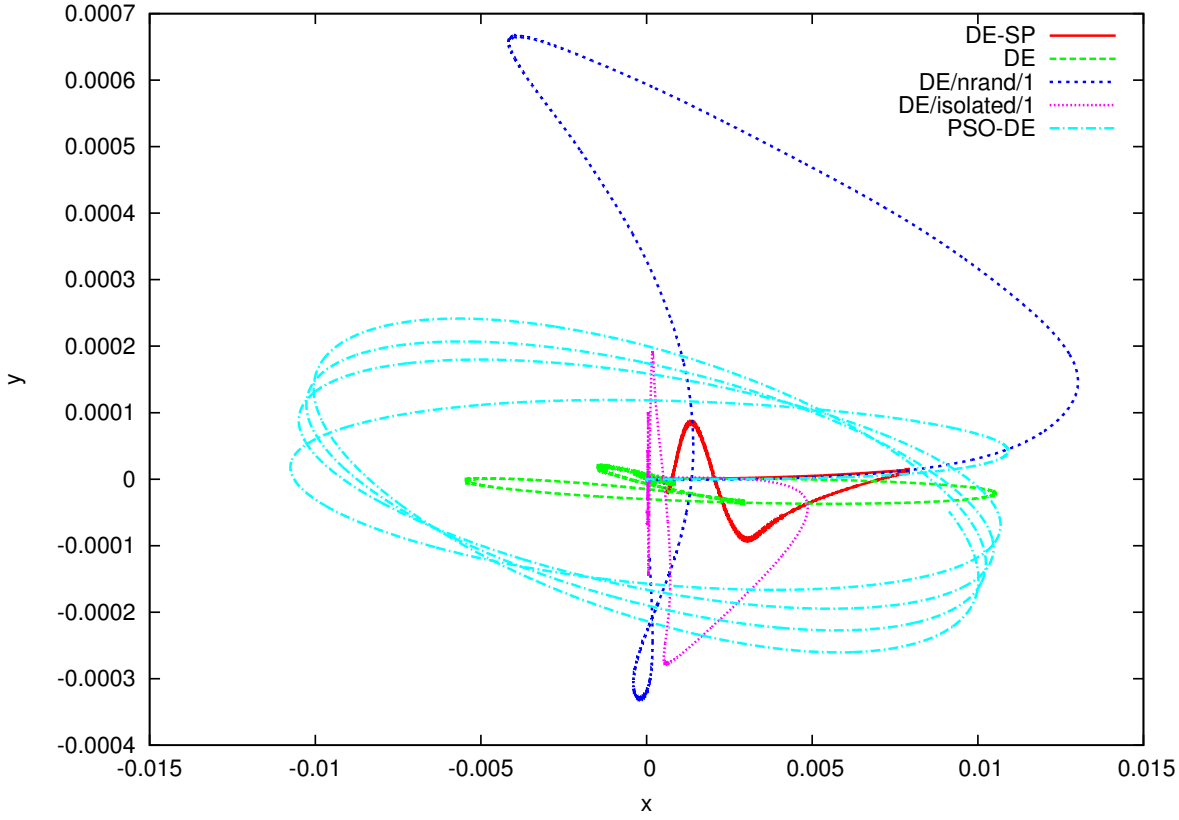


図 3.15: ロボットを背面側（図中左）から押した場合の重心軌跡

第4章

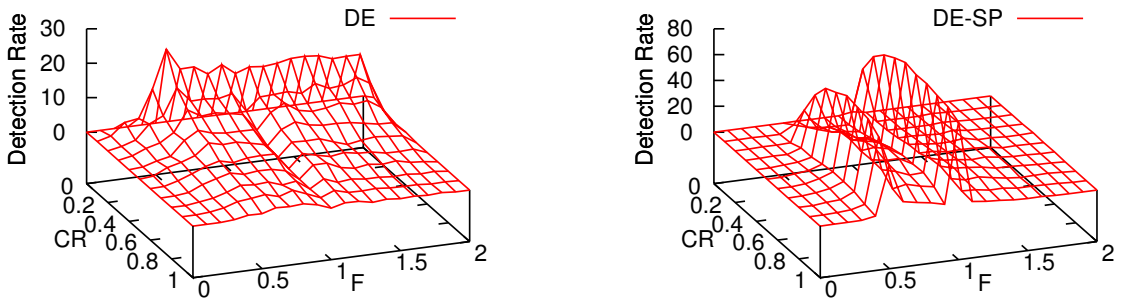
Self-Adaptive Differential Evolution on Scattered Parents with Dynamic Restartの提案と性能評価

4.1　はじめに

4.1.1　背景

ここまで，DE の性能の向上を目的とし改良を施した DE-SP を提案し，ベンチマーク問題と実問題の最適化問題をいくつか取り上げて実験を行うことにより有効性を確認してきた．しかし，1 章にて簡易に触れたように，DE にはパラメータ F，C_R を設定することが困難であるという問題が指摘されており [58, 59]，DE-SP においてもこの問題は解決されていない．どのような困難性を持つのか確認するために，2.5.3 にて述べた DE と DE-SP を用いて NF1 を最適化する場合における F，C_R の総当たり実験の結果を例示として図 4.1 に示す．この図は横軸に F，縦軸に C_R，鉛直軸に実験結果である最適解発見率をとったグラフである．これら図から，DE と DE-SP の性能は F または C_R に対して過敏に上下し，外れパラメータにおいては最適化手法としての用をなさない場合が多いことが確認される．この事態を根本的に回避するためには問題ごとに総当たり実験を行うこととなるが，F，C_R は連続値パラメータであるため，どの程度の細かさで総当たりをするべきなのかは問題の規模に応じて変化し得る．また，実験一試行に時間がかかる場合は総当たり自体が実用上困難である場合も考え得る．端的に問題を述べ直すのであれば，このままでは手法実装者による負担が大きいため，実用にはとても使えたものではない手法となってしまっている．そこで本研究では，探索中に有効なパラメータ自体も探索させることでこの問題を解決することを改めて目標とした．

本研究では当初，先行研究による DE-SP に対し，探索の停滞傾向に応じてパラメータを再設定することを目的とした機構を付け加えた Simple Self-Adaptive Differential Evolution on Scattered Parents（SSDE-SP）を提案した [60, 2]．しかし，この手法ではパラメータの再設定が適切に行われないことが確認されたため，改めて改良を施した



(a) DE における総当たり結果　　(b) DE-SP における総当たり結果

図 4.1: DE と DE-SP を用いて NF1 を最適化する場合における F，C_R の総当たり実験の結果

Self-Adaptive Differential Evolution on Scattered Parents with Random Jump（SDE-SP-RJ）を続いて提案した [61]．SDE-SP-RJ においては SSDE-SP において実現に失敗していたパラメータの再設定，それによる探索継続が行われることが確認され，なおかつ SDE-SP-RJ において置き換えられたパラメータである $T_{S_{\max}}$ は，DE や DE-SP の F，C_R よりも寛容であることも確認された．そのため，パラメータを完全に廃すという点に対して妥協をするのであれば，SDE-SP-RJ は SSDE-SP よりも妥当な改良であることが示された [62]．

当然ながら利便性を重視するのであれば，理想的には手法実装者により設定される必要のあるパラメータは完全に廃されているべきである．本研究では，この実現可能性について再び向かい合い，改良方法を模索することとした．その上で，本稿ではこの理想を実現する可能性がある手法として，Differential Evolution on Scattered Parents with Dynamic Restart（SDE-SP-DR）を提案する．SDE-SP-DR では手法実装者により設定する必要のあるパラメータは完全に廃されており，SDE-SP-DR は本研究の目的をひとまずは妥協なしに実現したものと捉えることができる．

本章では当初の改良である SSDE-SP とその問題点を 4.2 節にて，ひとまずの解決を見た SDE-SP-RJ と有効性確認実験についてを 4.3 節にて，そして，本稿の最終的な提案手法として SDE-SP-DR と有効性確認実験についてを 4.5.1 節にて，順を追って述べる．

4.1.2　Differential Evolution の既存の改良

本研究と同様の問題意識を持ち，DE の改良を試みる研究は他にも行われている．例えば Liu ら [63] は，F と C_R を直接設定するのではなく，探索の状況に応じて F と C_R を出力する Mamdani の Fuzzy 推論器を設計する問題に置き換えることを試みている．Omran ら [25] は F と C_R を毎目的関数評価ごとに乱数を用いて決定することでパラメータ設定自体が不要となるよう改良した．Noman ら [64]，Soliman ら [65]，Brest ら [66]

は DE の各個体に F と C_R を持たせ，突然変異と淘汰の影響を F と C_R にも与える[1]ことで，適切なパラメータが選別されることを期待した改良を施している．山口 [67] は，DE/rand/1/exp を基本の手法として注目し，現在発見されている最も良い解（最良解）と一定摂動回数前の最良解との差分を探索の進行程度の指標とし，F と C_R を調整している．本節ではこれらの手法の具体的実装について簡易的に述べる．

Fuzzy Adaptive Differential Evolution

Fuzzy Adaptive Differential Evolution（FADE）とは，Liu ら [63] によって提案された，DE のパラメータ F，C_R を毎摂動ごとに Mamudani の Fuzzy 推論器を用いて更新することで，探索性能の向上を図った手法である．疑似コードをアルゴリズム 18 に示す．ただし，Fuzzy 推論器の設計は煩雑となるので疑似コード内では略記している．この Fuzzy 推論器の設計は以下に述べる．また，Liu らによるこの推論器の設計は汎用的に利用可能なものとなることを期待し，行われたものである．

FADE で用いる Fuzzy 推論器は F を推論するものと C_R を推論するものの二つが独立して存在する．F を推論するものは d_{11}，d_{12} を，C_R を推論するものは d_{21}，d_{22} を入力に取る．ここで四つの d は次のように計算される．

$$d_{11} = \frac{1-(1+P_C)}{e^{P_C}} \tag{4.1}$$

$$d_{12} = \frac{1-(1+F_C)}{e^{F_C}} \tag{4.2}$$

$$d_{21} = 2d_{11} \tag{4.3}$$

$$d_{22} = 2d_{12} \tag{4.4}$$

この計算で用いられている P_C，F_C はそれぞれ一摂動前の個体群と現個体群との各々の差の二乗和の平方と一摂動前の個体群と現個体群との各々の目的関数の差の二乗和の平方であり，次のように計算される．

$$P_C = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D}(\chi_j^i - {}^{\mathrm{old}}\chi_j^i)^2} \tag{4.5}$$

$$F_C = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(E(\boldsymbol{\chi}_j) - E({}^{\mathrm{old}}\boldsymbol{\chi}_j)\right)^2} \tag{4.6}$$

推論器は各入力に対して三つのファジィ部分集合からなる一つのファジィ集合を定義する．この三つのファジィ部分集合は,「小さい，中くらい，大きい」を指す．以後の説明において，入力 A に対するファジィ集合を $\mathcal{S}^A$ と表記し，この各ファジィ集合を，

$$\mathcal{S}^A = \{\mathcal{S}_S^A, \mathcal{S}_M^A, \mathcal{S}_B^A\} \tag{4.7}$$

[1] これら三つの研究においては F と C_R の交叉は行っていない

表 4.1: 各ファジィ部分集合 S_W^A に割り当てられるメンバシップ関数 $\mu(\mathcal{S}_W^A, A)$ の表

	小さい	中くらい	大きい
$\mathcal{S}^{d_{11}}$	$f_{\mathcal{G}}(d_{11}, 0.05, 0.25)$	$f_{\mathcal{G}}(d_{11}, 0.5, 0.25)$	$f_{\mathcal{G}}(d_{11}, 0.9, 0.25)$
$\mathcal{S}^{d_{12}}$	$f_{\mathcal{G}}(d_{12}, 0.01, 0.35)$	$f_{\mathcal{G}}(d_{12}, 0.5, 0.35)$	$f_{\mathcal{G}}(d_{12}, 0.9, 0.35)$
$\mathcal{S}^{F}$	$f_{\mathcal{G}}(F, 0.3, 0.5)$	$f_{\mathcal{G}}(F, 0.6, 0.5)$	$f_{\mathcal{G}}(F, 0.9, 0.5)$
$\mathcal{S}^{d_{21}}$	$f_{\mathcal{G}}(d_{21}, 0.1, 0.5)$	$f_{\mathcal{G}}(d_{21}, 0.8, 0.5)$	$f_{\mathcal{G}}(d_{21}, 1.5, 0.5)$
$\mathcal{S}^{d_{22}}$	$f_{\mathcal{G}}(d_{22}, 0.1, 0.5)$	$f_{\mathcal{G}}(d_{22}, 0.8, 0.5)$	$f_{\mathcal{G}}(d_{22}, 1.5, 0.5)$
$\mathcal{S}_{C_R}$	$f_{\mathcal{G}}(C_R, 0.4, 0.35)$	$f_{\mathcal{G}}(C_R, 0.7, 0.35)$	$f_{\mathcal{G}}(C_R, 1.0, 0.35)$

表 4.2: Fuzzy 推論器の IF-THEN ルール

	前件部		後件部
	d_{11}，d_{21} に対する W_1	d_{21}，d_{22} に対する W_2	F，C_R に対する W_3
1	S	S	S
2	S	M	M
3	S	B	B
4	M	S	M
5	M	M	M
6	M	B	B
7	B	S	B
8	B	M	B
9	B	B	B

と表記する．推論器の入力に対応する各ファジィ部分集合 $\mathcal{S}_W^A$（W は S，M，B のいずれか）に割り当てられるメンバシップ関数 $\mu(\mathcal{S}_W^A, A)$ はガウス関数に基づいたものを用いている．ガウス関数を，

$$f_{\mathcal{G}}(x, \mu, \sigma) = e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.8}$$

と表現する時の，各ファジィ部分集合に割り当てられるメンバシップ関数を表にまとめたものを表 4.1 に示す．次に，各推論器における IF-THEN ルールは，F を推論するものは「d_{11} が $\mathcal{S}_{W_1}^{d_{11}}$ かつ d_{12} が $\mathcal{S}_{W_2}^{d_{12}}$ ならば F は $\mathcal{S}_{W_3}^{F}$ である」（W_1，W_2，W_3 はそれぞれ S，M，B のいずれか）となっており，C_R を推論するものは「d_{21} が $\mathcal{S}_{W_1}^{d_{21}}$ かつ d_{12} が $\mathcal{S}_{W_2}^{d_{22}}$ ならば C_R は $\mathcal{S}_{W_3}^{C_R}$ である」となっている．このルールの取り得る組み合わせは 27 通りあるが，FADE では利用するルールは 9 通りに選別されている．これらの選別されているルール群は F を推論するものと C_R を推論するものとで同一のものである．これらの選別されたルールをまとめた表を表 4.2 に示す．続いて，これらのルールを用いて推論結果集合を決定する．F を推論する Fuzzy 推論器において，推論結果集合を B_F，表 4.2 のルール番号を k とし，各ルールに対する推論結果部分集合を B_F^k とすると，B_F は，

$$B_F = \bigcup_k B_F^k \tag{4.9}$$

であり，対応する適合度メンバーシップ値はメンバシップ関数，

$$\mu(B_F, F) = \bigvee_k \left(\mu(S^{d_{11}}_{W^k_1}, d_{11}) \wedge \mu(S^{d_{12}}_{W^k_2}, d_{12}) \wedge \mu(S^{F}_{W^k_3}, F) \right) \tag{4.10}$$

によって列挙される．同様に C_R を推論する Fuzzy 推論器において，推論結果集合を B_{C_R}，各ルールに対する推論結果部分集合を $B^k_{C_R}$ とすると，B_{C_R} は，

$$B_{C_R} = \bigcup_k B^k_{C_R} \tag{4.11}$$

であり，対応する適合度メンバーシップ値はメンバシップ関数，

$$\mu(B_{C_R}, C_R) = \bigvee_k \left(\mu(S^{d_{21}}_{W^k_1}, d_{21}) \wedge \mu(S^{d_{22}}_{W^k_2}, d_{22}) \wedge \mu(S^{C_R}_{W^k_3}, C_R) \right) \tag{4.12}$$

によって列挙される．最後に，これらにより列挙されたメンバーシップ値を重心を用いた非ファジィ化 [68] を適用することで F，C_R の値を推論する．即ち，F の値は，

$$\begin{aligned} F &= \frac{\int u \cdot \mathcal{S}^F_u \, \mathrm{d}u}{\int \mathcal{S}^F_u \, \mathrm{d}u} \\ &= \frac{\mu(B_F, w(\mu(\mathcal{S}^F_S))) \cdot w(\mu(\mathcal{S}^F_S)) + \mu(B_F, w(\mu(\mathcal{S}^F_M))) \cdot w(\mu(\mathcal{S}^F_M)) + \mu(B_F, w(\mu(\mathcal{S}^F_B))) \cdot w(\mu(\mathcal{S}^F_B))}{\mu(B_F, w(\mu(\mathcal{S}^F_S))) + \mu(B_F, w(\mu(\mathcal{S}^F_M))) + \mu(B_F, w(\mu(\mathcal{S}^F_B)))} \end{aligned} \tag{4.13}$$

となる．ここで，w は引数の幾何重心を求める関数である．今回のメンバシップ関数は全てガウス分布によって与えられているため，実質的に各分布の平均値となっている．

アルゴリズム 18 Fuzzy Adaptive Differential Evolution

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F \in [0, 2] \cdots$ スケーリングパラメータ
$C_R \in [0, 1] \cdots$ 交叉時に用いる閾値
${}^{\mathrm{old}}\boldsymbol{\chi} \cdots$ 一摂動前の個体
$P_C \cdots$ 一摂動前の個体群と現個体群との各々の差の二乗和の平方
$F_C \cdots$ 一摂動前の個体群と現個体群との各々の目的関数の差の二乗和の平方
$d_{11}, d_{12} \cdots$ F を調整する Fuzzy 推論器への入力
$d_{21}, d_{22} \cdots$ C_R を調整する Fuzzy 推論器への入力
$R_{\mathcal{U}}(\mathbb{R}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の実数一様乱数
$R_{\mathcal{U}}(\mathbb{N}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の自然数一様乱数

for $i = 1$ to N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
end for
while 終了条件が未成立 **do**
　P_C を計算: $P_C \leftarrow \sqrt{\dfrac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{D} (\chi_j^i - {}^{\mathrm{old}}\chi_j^i)^2}$
　F_C を計算: $F_C \leftarrow \sqrt{\dfrac{1}{N} \sum_{i=1}^{N} \left(E(\boldsymbol{\chi}^i) - E({}^{\mathrm{old}}\boldsymbol{\chi}^i)\right)^2}$
　各 d を計算: $d_{11} \leftarrow \dfrac{1 - (1 + P_C)}{e^{P_C}}$，$d_{12} \leftarrow \dfrac{1 - (1 + F_C)}{e^{F_C}}$，$d_{21} \leftarrow 2d_{11}$，$d_{22} \leftarrow 2d_{12}$
　各 d，F，C_R を入力とした Fuzzy 推論器を用いて F，C_R を更新
　for $i = 1$ to N **do**
　　サブ親 $\boldsymbol{\chi}^a$, $\boldsymbol{\chi}^b$, $\boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ をランダムに選択
　　突然変異個体を計算: $\boldsymbol{\chi}'^i \leftarrow \boldsymbol{\chi}^a + F(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c)$
　　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
　　for $k = 1$ to D **do**
　　　if $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < C_R$ **or** $k = l^i$ **then**
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k'^i$
　　　else
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k^i$
　　　end if
　　end for
　　目的関数 $E(\boldsymbol{p}^i)$ を計算
　end for
　for $i = 1$ to N **do**
　　if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
　　　親個体を子個体と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
　　else
　　　親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
　　end if
　end for
end while

Self-adaptive Differential Evolution

Self-adaptive Differential Evolution（SDE）とは，Omran ら [25] によって提案された，DE のパラメータ F，C_R を各摂動時に各個体の各要素ごとに乱数を用いて調整し，探索性能の向上を図った手法である．疑似コードをアルゴリズム 19 に示す.

この手法の計算手順は DE に準拠するが，各摂動時の各個体の各要素ごとに乱数を振り直すことで常に F，C_R の値を変動させている．具体的な変動方法は次の通りである．まず F の決定の方法について述べる．F は各摂動の開始時に，一様乱数を用いて N 個の種を用意する（アルゴリズム 19，14 行目）

$$F^i = R_{\mathcal{U}}(\mathbb{R}, [0, 2]) \quad (i = 1, 2, \ldots N) \tag{4.14}$$

そしてこれらの種を用いて，突然変異個体の演算時に，**各要素ごと**に以下の式に基づいて F を決定する（アルゴリズム 19，20 行目）

$$F = F^{R_{\mathcal{U}}(\mathbb{N},[1,N])} + R_{\mathcal{N}}(\mathbb{R}, 0, 0.5)(F^{R_{\mathcal{U}}(\mathbb{N},[1,N])} - F^{R_{\mathcal{U}}(\mathbb{N},[1,N])}) \tag{4.15}$$

続いて，C_R の決定の方法について述べる．C_R の決定は要素を子個体に引き継ぐか否かの確率判定を行う度に，毎回正規乱数を用いて決定される（アルゴリズム 19，22 行目）

$$C_R = R_{\mathcal{N}}(\mathbb{R}, 0.5, 0.15) \tag{4.16}$$

Omran らは，これらの改変によりパラメータが固定化されることによる探索の停滞が緩和されることを期待している.

アルゴリズム 19 Self-adaptive Differential Evolution

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F \in [0, 2] \cdots$ スケーリングパラメータ
$R_{\mathcal{U}}(\mathbb{R}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の実数一様乱数
$R_{\mathcal{U}}(\mathbb{N}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の自然数一様乱数
$R_{\mathcal{N}}(\mathbb{R}, \mu, \sigma^2) \cdots$ 平均 μ 分散 σ^2 の正規乱数

for $i = 1$ to N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
end for
while 終了条件が未成立 **do**
　for $i = 1$ to N **do**
　　F^i を乱数で初期化: $F^i \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0, 2])$
　end for
　for $i = 1$ to N **do**
　　サブ親 $\boldsymbol{\chi}^a$, $\boldsymbol{\chi}^b$, $\boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ をランダムに選択
　　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
　　for $k = 1$ to D **do**
　　　次の計算で用いる F を決定: $F \leftarrow F^{R_{\mathcal{U}}(\mathbb{N},[1,N])} + R_{\mathcal{N}}(\mathbb{R}, 0, 0.5)(F^{R_{\mathcal{U}}(\mathbb{N},[1,N])} - F^{R_{\mathcal{U}}(\mathbb{N},[1,N])})$
　　　突然変異個体の要素を計算: $\boldsymbol{\chi}'^i \leftarrow \boldsymbol{\chi}^a + F(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c)$
　　　if $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < R_{\mathcal{N}}(\mathbb{R}, 0.5, 0.15)$ **or** $k = l^i$ **then**
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k'^i$
　　　else
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k^i$
　　　end if
　　end for
　　目的関数 $E(\boldsymbol{p}^i)$ を計算
　end for
　for $i = 1$ to N **do**
　　if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
　　　親個体を子個体と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
　　else
　　　親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
　　end if
　end for
end while

Noman らによる Adaptive Differential Evolution

Noman らは文献 [64] にて，F と C_R を各個体に所持させ，比較的悪い個体の F，C_R に対して突然変異を施す Adaptive Differential Evolution の一形態を提案している．疑似コードをアルゴリズム 20 に示す.

この手法の計算手順は DE に準拠するが，F と C_R が各個体ごとに用意されている点（アルゴリズム 20，3，4 行目），子個体の目的関数値 $E(\boldsymbol{\chi}^i)$ が親個体の目的関数値の平均 $\bar{E}$ よりも悪い場合に F と C_R に突然変異を施す点（アルゴリズム 20，29 行目）が異なる．また，この手法では F の値域を $[0.1, 1]$ としている点も通常の DE と異なる．

$$F^i = \begin{cases} F^{i-1} & \left(E(\boldsymbol{p}^i) < \bar{E}\right) \\ R_{\mathcal{U}}(\mathbb{R}, [0.1, 1]) & \left(E(\boldsymbol{p}^i) \geq \bar{E}\right) \end{cases} \tag{4.17}$$

$$C_R^i = \begin{cases} C_R^{i-1} & \left(E(\boldsymbol{p}^i) < \bar{E}\right) \\ R_{\mathcal{U}}(\mathbb{R}, [0, 1]) & \left(E(\boldsymbol{p}^i) \geq \bar{E}\right) \end{cases} \tag{4.18}$$

Soliman らによる Self-Adaptive Differential Evolution

Soliman らは文献 [65] にて F と C_R を各個体ごとに用意し，コーシー分布に基づく乱数（以下コーシー乱数）を用いて突然変異をさせる改良を施した Differential Evolution を提案している．疑似コードをアルゴリズム 21 に示す.

この手法の計算手順は DE に準拠するが，F と C_R が各個体ごとに用意されている点（アルゴリズム 21，3，4 行目），新たなパラメータ δ とコーシー乱数を用いてこれら F と C_R を突然変異させている点が異なる．

$$f^i = \begin{cases} R_{\mathcal{C}}(\mathbb{R}, [\mu, \delta^i]) & (R_{\mathcal{U}}(\mathbb{R}, [0, 1]) \leq \pi_1) \\ R_{\mathcal{C}}(\mathbb{R}, [\mu, \delta^i]) & (R_{\mathcal{U}}(\mathbb{R}, [0, 1]) > \pi_1) \end{cases} \tag{4.19}$$

$$c_r^i = \begin{cases} R_{\mathcal{U}}(\mathbb{R}, [0, 1]) & (R_{\mathcal{U}}(\mathbb{R}, [0, 1]) \leq \pi_2) \\ C_R^i & (R_{\mathcal{U}}(\mathbb{R}, [0, 1]) > \pi_1) \end{cases} \tag{4.20}$$

ここで，μ，π_i，π_2 はパラメータであるかのように設けられてはいるものの，Soliman らによってそれぞれ 0.8，0.1，0.1 が適当であると考えられており，本稿においてもこれに従う．また，f，c_r は対応する子個体候補を示し，子個体が受理されなかった場合，これらのパラメータも受理されず，現摂動のものが次摂動へと引き継がれることとなる．

Brest らによる Self-Adaptive Differential Evolution

Brest らは文献 [66] にて F，C_R を各個体ごとに用意し，また突然変異により変異させる Differential Evolution を提案している．疑似コードをアルゴリズム 22 に示す．この着

想そのものは Soliman のものと類似するが，F，C_R の突然変異は多様性を維持するために無条件に受理される点，F が 0 付近の値である場合に個体の突然変異が発生しない事態を防ぐため，値域を $[0.1, 1]$ としている点，F，C_R の突然変異は一様乱数を用いたものである点が異なる．

$$F_i = \begin{cases} F_l + R_{\mathcal{U}}(\mathbb{R}, [0,1]) \times F_u & (R_{\mathcal{U}}(\mathbb{R}, [0,1]) < \tau_1) \\ F_{i-1} & (R_{\mathcal{U}}(\mathbb{R}, [0,1]) \geq \tau_1) \end{cases} \tag{4.21}$$

$$C_R^i = \begin{cases} R_{\mathcal{U}}(\mathbb{R}, [0,1]) & (R_{\mathcal{U}}(\mathbb{R}, [0,1]) < \tau_2) \\ C_R^{i-1} & \end{cases} \tag{4.22}$$

ここで，F_l，F_u，τ_1，τ_2 は Brest らが設けるパラメータであるが，それぞれ 0.1，0.9，0.1，0.1 とされている．F_l と F_u の値は F の値域を $[0.1, 1]$ とするためのものであり，τ_1，τ_2 の値は，Brest らの実験により，この値を調整する意義は実用上はかなり薄いと示唆され，0.1 を設定すればよいと示されたため，本稿においてもこれに従う．

山口による Self-Adaptive Differential Evolution

山口らは文献 [67] にて DE/rand/1/exp を基本とし，一定間隔 T 摂動置きに最良値の変化を観察し，その変化がより大きくなっている（探索が進んでいる）かより小さくなっている（探索が停滞しつつある）かに応じて F を，子個体候補の受理率に応じて C_R を変化させる Self-Adaptive Differential Evolution を提案している．疑似コードをアルゴリズム 23 に示す．この手法において，F の調整は最良解の差分が大きくなったか（$\Delta E - \Delta E_p > 0$）あるいは小さくなったか（$\Delta E - \Delta E_p < 0$）を基準とし，大きくなった場合は「方針が適切であった」と判断して F の調整方針を継続し，一方小さくなった場合は,「方針が不適切であった」と判断して F の調整方針を反転する．

$$F = \begin{cases} F \times \xi & (F_R = \text{減少}) \\ \dfrac{F}{\xi} & (F_R = \text{増加}) \\ F & (\Delta E - \Delta E_p = 0) \end{cases} \tag{4.23}$$

ここで，F_R は { 増加, 減少 } のどちらかを示す「方針」であり，$\xi \in (0, 1)$ は F を調整するためのパラメータである．また，C_R の調整は，$l_l \in \{0, 1, \ldots, D\}$ ごとの解の受理率に基づいて計算される．まず，閾値として $\iota \in (0, 1)$ を設定しておく．そして，探索の状況を集計して判明する l_l ごとの解の受理率を計算する．

$$q_k = \frac{\dfrac{M_s(k)}{MP(k)}}{\displaystyle\sum_{h=1}^{D} \frac{M_s(h)}{MP(h)}} \tag{4.24}$$

ここで，$M_s(k)$　$(k = 1, 2, \ldots, D)$ は直近 T 摂動において，長さ k の交叉があった場合に子個体候補が受理された回数であり，$M = TN$ である．また，$P(x)$ は l_l の確率分布

であり，次のように表される．

$$P(x) = \begin{cases} (1 - C_R)C_R^{x-1} & (1 \leq x < D) \\ C_R^{D-1} & (x = D) \end{cases} \tag{4.25}$$

各 q が導出されたならば，L を $\sum_{h=1}^{L} q_h$ が初めて ι 以上となる最小の整数とする．次に，L を集計情報からではなく，式 (4.25) から導出したものを L' とする．

$$L' = \left\lceil \frac{\log(1 - \iota)}{\log C_R} \right\rceil \tag{4.26}$$

そして，$L < L'$ であれば，C_R が大きすぎであり，$L > L'$ であれば小さすぎであると判断する．

$$C_R = \begin{cases} C_R \times \eta & (L < L') \\ \dfrac{C_R}{\eta} & (L > L') \\ C_R & (L = L') \end{cases} \tag{4.27}$$

手法実装者が設定すべきパラメータは，ξ，η，ι であるが，これらは目安となり得るものが文献 [67] にて既に述べられている．本稿でもこれに従い，$\xi = 0.85$，$\eta = 0.85$，$\iota = 0.8$ とした．

アルゴリズム 20 Noman らによる Adaptive Differential Evolution

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F^{1,2,\ldots,N} \in [0.1, 1] \cdots$ スケーリングパラメータ
$C_R^{1,2,\ldots,N} \in [0, 1] \cdots$ 交叉時に用いる閾値
$R_{\mathcal{U}}(\mathbb{R}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の実数一様乱数
$R_{\mathcal{U}}(\mathbb{N}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の自然数一様乱数

for $i = 1$ to N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
　F^i を初期化: $F^i \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0.1, 1])$
　C_R^i を初期化: $C_R^i \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0, 1])$
end for
while 終了条件が未成立 **do**
　親個体の目的関数値の平均を計算: $\bar{E} \leftarrow \dfrac{1}{N} \displaystyle\sum_{i=1}^{N} E(\boldsymbol{\chi}^i)$
　for $i = 1$ to N **do**
　　サブ親 $\boldsymbol{\chi}^a$, $\boldsymbol{\chi}^b$, $\boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ をランダムに選択
　　突然変異個体を計算: $\boldsymbol{\chi}'^i \leftarrow \boldsymbol{\chi}^a + F^i(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c)$
　　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
　　for $k = 1$ to D **do**
　　　if $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < C_R^i$ **or** $k = l^i$ **then**
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k'^i$
　　　else
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k^i$
　　　end if
　　end for
　　目的関数 $E(\boldsymbol{p}^i)$ を計算
　　if $E(\boldsymbol{p}^i) > \bar{E}$ **then**
　　　F^i, C_R^i を突然変異:

$$\begin{aligned} F^i &\leftarrow R_{\mathcal{U}}(\mathbb{R}, [0.1, 1]) \\ C_R^i &\leftarrow R_{\mathcal{U}}(\mathbb{R}, [0, 1]) \end{aligned}$$

　　else
　　　F^i, C_R^i をそのまま引き継ぐ
　　end if
　end for
　for $i = 1$ to N **do**
　　if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
　　　親個体を子個体と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
　　else
　　　親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
　　end if
　end for
end while

アルゴリズム 21 Soliman らによる改良を施された Differential Evolution

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F^{1,2,\ldots,N}, f^{1,2,\ldots,N} \in [0,1] \cdots$ スケーリングパラメータ（小文字は子個体候補）
$C_R^{1,2,\ldots,N}, c_r^{1,2,\ldots,N} \in [0,1] \cdots$ 交叉時に用いる閾値（小文字は子個体候補）
$\delta^{1,2,\ldots,N} \in [\delta_l, \delta_u] \cdots$ F，C_R の突然変異に用いる分散パラメータ
$\delta_l = 0.1$，$\delta_u = 0.9 \cdots$ パラメータ δ の上下限
$\pi_1 = 0.1$，$\pi_2 = 0.1 \cdots$ それぞれ F，C_R の突然変異率
$\mu = 0.8$ コーシー乱数に用いられる平均
$R_{\mathcal{U}}(\mathbb{R}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の実数一様乱数
$R_{\mathcal{U}}(\mathbb{N}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の自然数一様乱数
$R_{\mathcal{C}}(\mathbb{R}, [\mu, \gamma]) \cdots$ 平均 μ 分散 γ に基づく実数コーシー乱数

for $i = 1$ to N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
　F^i を初期化: $F^i \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0,1])$
　C_R^i を初期化: $C_R^i \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0,1])$
end for
while 終了条件が未成立 **do**
　for $i = 1$ to N **do**
　　if $R_{\mathcal{U}}(\mathbb{R}, [0,1]) \leq \pi_1$ **then**
　　　δ^i を更新: $\delta^i \leftarrow \delta_l + \delta_u \times R_{\mathcal{U}}(\mathbb{R}, [0,1])$
　　　f^i を計算: $f_i \leftarrow R_{\mathcal{C}}(\mathbb{R}, [\mu, \delta^i])$
　　else
　　　f^i を計算: $f_i \leftarrow R_{\mathcal{C}}(\mathbb{R}, [\mu, \delta^i])$
　　　δ^i を更新: $\delta_i \leftarrow \delta_l + \delta_u \times R_{\mathcal{U}}(\mathbb{R}, [0,1])$
　　end if
　　if $R_{\mathcal{U}}(\mathbb{R}, [0,1]) \leq \pi_2$ **then**
　　　c_r^i を計算: $c_r^i \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0,1])$
　　else
　　　c_r^i に現在の値を引き継ぐ: $c_r^i \leftarrow C_R^i$
　　end if
　　サブ親 $\boldsymbol{\chi}^a$, $\boldsymbol{\chi}^b$, $\boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ をランダムに選択
　　突然変異個体を計算: $\boldsymbol{\chi}'^i \leftarrow \boldsymbol{\chi}^a + f^i(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c)$
　　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
　　for $k = 1$ to D **do**
　　　if $R_{\mathcal{U}}(\mathbb{R}, [0,1]) < c_r^i$ **or** $k = l^i$ **then**
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k'^i$
　　　else
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k^i$
　　　end if
　　end for
　　目的関数 $E(\boldsymbol{p}^i)$ を計算
　end for
　for $i = 1$ to N **do**
　　if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
　　　F，C_R を含め，親個体を子個体と置換:

$$\begin{aligned} \boldsymbol{\chi}^i &\leftarrow \boldsymbol{p}^i \\ F^i &\leftarrow f^i \\ C_R^i &\leftarrow c_r^i \end{aligned}$$

　　else
　　　F，C_R を含め，親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
　　end if
　end for
end while

アルゴリズム 22 Brest らによる改良を施された Differential Evolution

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F^{1,2,\ldots,N} \in [0,1] \cdots$ スケーリングパラメータ
$C_R^{1,2,\ldots,N} \in [0,1] \cdots$ 交叉時に用いる閾値
$F_l = 0.1,\ F_u = 0.9 \cdots$ 各 F^i 生成時に用いるパラメータ
$\tau_1 = 0.1,\ \tau_2 = 0.1 \cdots$ それぞれ各 $F^i,\ C_R^i$ 突然変異時に用いる確率閾値
$R_{\mathcal{U}}(\mathbb{R}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の実数一様乱数
$R_{\mathcal{U}}(\mathbb{N}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の自然数一様乱数

for $i = 1$ to N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
　F^i を初期化: $F^i \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0.1, 1])$
　C_R^i を初期化: $C_R^i \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0, 1])$
end for
while 終了条件が未成立 **do**
　for $i = 1$ to N **do**
　　if $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < \tau_1$ **then**
　　　F^i を計算: $F_i \leftarrow F_l + R_{\mathcal{U}}(\mathbb{R}, [0, 1]) \times F_u$
　　else
　　　F^i をそのまま保持
　　end if
　　if $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) \leq \tau_2$ **then**
　　　C_R^i を計算: $C_R^i \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0, 1])$
　　else
　　　C_R^i をそのまま保持
　　end if
　　サブ親 $\boldsymbol{\chi}^a, \boldsymbol{\chi}^b, \boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ をランダムに選択
　　突然変異個体を計算: $\boldsymbol{\chi}'^i \leftarrow \boldsymbol{\chi}^a + F^i(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c)$
　　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
　　for $k = 1$ to D **do**
　　　if $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < C_R^i$ **or** $k = l^i$ **then**
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k'^i$
　　　else
　　　　子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k^i$
　　　end if
　　end for
　　目的関数 $E(\boldsymbol{p}^i)$ を計算
　end for
　for $i = 1$ to N **do**
　　if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
　　　親個体を子個体と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
　　else
　　　親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
　　end if
　end for
end while

アルゴリズム 23 山口による Self-Adaptive Differential Evolution

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F \in [0, 2] \cdots$ スケーリングパラメータ
$\xi \in (0, 1) \cdots$ F 調整時に用いるパラメータ
$C_R \in [0, 1] \cdots$ 交叉時に用いる閾値
$\eta, \iota \in (0, 1) \cdots$ C_R 調整時に用いるパラメータ
$l,\ l_l \cdots$ 交叉開始点と交叉長
$T \cdots$ $F,\ C_R$ を調整する間隔
$F_R \in \{$ 増加, 減少 $\} \cdots$ 現在の F 調整方法の方針
$M_s(l_l) \cdots$ 直前の T 摂動間で交叉長が l_l であった時に子個体候補が受理された回数
$E_b,\ E_{b_p} \cdots$ 現在の最良解と T 摂動前の最良解
$\Delta E_b,\ \Delta E_{b_p} \cdots$ 現在の最良解と T 摂動前の最良解との差分，T 摂動前と $2T$ 摂動前の最良解との差分
$R_{\mathcal{U}}(\mathbb{R}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の実数一様乱数
$R_{\mathcal{U}}(\mathbb{N}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の自然数一様乱数

for $i = 1$ to N **do**
    個体 $\boldsymbol{\chi}^i$ を乱数で初期化
    目的関数 $E(\boldsymbol{\chi}^i)$ を計算
end for
タイマーを初期化: $t \leftarrow 0$
E_{b_p} を初期化: $E_{b_p} \leftarrow E_b$
ΔE_p を初期化: $\Delta E_b \leftarrow \infty$
F_R を初期化: $F_R \leftarrow$ 増加
$M_s(l_l) \quad (l_l = 1, 2, \ldots, D)$ を 0 で初期化
while 終了条件が未成立 **do**
    for $i = 1$ to N **do**
        サブ親 $\boldsymbol{\chi}^a$, $\boldsymbol{\chi}^b$, $\boldsymbol{\chi}^c$ $(i \neq a \neq b \neq c)$ をランダムに選択
        突然変異個体を計算: $\boldsymbol{\chi}'^i \leftarrow \boldsymbol{\chi}^a + F(\boldsymbol{\chi}^b - \boldsymbol{\chi}^c)$
        交叉開始点を設定: $l \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
        交叉長を初期化: $l_l \leftarrow 0$
        while $C_R^{l_l+1} > R_{\mathcal{U}}(\mathbb{R}, [0, 1])$ **and** $l_l < D$ **do**
            交叉長を延長: $l_l \leftarrow l_l + 1$
        end while
        for $k = 1$ to D **do**
            if $(l + l_l \leq D$ **and** $k \geq l$ **and** $k \leq l + l_l)$ **or** $(l + l_l > D$ **and** $(k \geq l$ **or** $k \leq l + l_l (\mod D)))$ **then**
                子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k'^i$
            else
                子個体候補の要素を次のように設定: $p_k^i \leftarrow \chi_k^i$
            end if
        end for
        目的関数 $E(\boldsymbol{p}^i)$ を計算
    end for
    for $i = 1$ to N **do**
        if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
            親個体を子個体と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
            $M_s(l_l)$ を更新: $M_s(l_l) \leftarrow M_s(l_l) + 1$
        else
            親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
        end if
    end for
    タイマーを 1 増やす: $t \leftarrow t + 1$
    if $t = T$ **then**
        ΔE を計算: $\Delta E \leftarrow |E_{b_p} - E_b|$
        F_R を更新: $F_R \leftarrow \begin{cases} F_R & (\Delta E - \Delta E_p \geq 0) \\ \neg F_R & (\Delta E - \Delta E_p < 0) \end{cases}$
        F を更新: $F \leftarrow \begin{cases} F \times \xi & (F_R = \text{減少}) \\ \dfrac{F}{\xi} & (F_R = \text{増加}) \\ F & (\Delta E - \Delta E_p = 0) \end{cases}$
        E_{b_p}, ΔE_p の更新: $E_{b_p} \leftarrow E_b$, $\Delta E_p = \Delta E$
        for $k = 1$ to D **do**
            閾値 L を導出するための中間変数 q_k を設定: $q_k \leftarrow \dfrac{\dfrac{M_s(k)}{MP(k)}}{\displaystyle\sum_{h=1}^{D} \dfrac{M_s(h)}{MP(h)}}$
            但し, $P(x) = \begin{cases} (1 - C_R) C_R^{x-1} & (1 \leq x < D) \\ C_R^{D-1} & (x = D) \end{cases}$
        end for
        $\displaystyle\sum_{h=1}^{L} q_h$ が初めて ι 以上となる L を導出
        閾値 L と比較する理論値 L' を計算: $L' \leftarrow \left\lceil \dfrac{\log(1 - \iota)}{\log C_R} \right\rceil$
        C_R を更新: $C_R \leftarrow \begin{cases} C_R \times \eta & (L < L') \\ \dfrac{C_R}{\eta} & (L > L') \\ C_R & (L = L') \end{cases}$
        タイマーを再初期化: $t \leftarrow 0$
        $M_s(l_l) \quad (l_l = 1, 2, \ldots, D)$ を再初期化: $M_s(l_l) \leftarrow 0 \quad (l_l = 1, 2, \ldots, D)$
    end if
end while

4.2 Simple Self-Adaptive Differential Evolution on Scattered Parents とその問題点

4.2.1 Simple Self-Adaptive Differential Evolution on Scattered Parents

まず当初提案した Simple Self-Adaptive Differential Evolution on Scattered Parents（SSDE-SP）の擬似コードをアルゴリズム 24 に示す．SSDE-SP では，DE-SP を用いるにあたって予め設定する必要のあった F，C_R は乱数で初期化され，M は 0 で初期化されるように改良されている（アルゴリズム 24，行番号 11）．SSDE-SP は原則として DE-SP のアルゴリズムに従って探索を行うが，探索の進行に合わせて連続失敗回数 T_F を記録する（アルゴリズム 24，行番号 33，38）．ここで，失敗とは子個体候補が親個体よりも悪い目的関数値を持つために置き換えが発生せず，親個体がそのまま次の摂動に引き継がれることを示す．したがって，T_F は連続して親個体が引き継がれている回数を示す．この値が増加するということは探索が停滞しているか収束していることを示す．SSDE-SP では T_F が予め設定された $T_{F_{\max}}$ を上回った時，F，C_R，M が適切でないために探索が停滞していると判断する．この時，F と C_R は乱数によって再設定され，M は 1 が加算される（アルゴリズム 24，行番号 40）．ただし M が個体数 N 以上となった場合，探索の挙動はランダムウォークに近くなり，個体は発散するように動くと考えられるため，再度 $M = 0$ として収束を試みる．

SSDE-SP では F，C_R，M を設定する代わりに $T_{F_{\max}}$ を設定する必要がある．この値はパラメータの再設定間隔のみに関わるものであり，変更に慎重であることを望む場合には大きく，積極的であることを望む場合には小さく設定すればよい．すなわち，このパラメータは対象とする最適化問題が難しい部類であると予想されるならば探索が停滞しやすいため小さく設定し，そうでないならばパラメータをそれほど調整しなくとも探索が進むと予想されるため比較的大きく設定すれば良いと予想されるものである．

4.2.2 問題点

この SSDE-SP が有効に機能しないと判断される根拠となる図を文献 [2] から引用し，図 4.2 に示す．この図は 2.5.3 で述べた最良パラメータの DE-SP，$T_{F_{\max}} = 12500$ と設定した SSDE-SP，比較手法である Self-adaptive Differential Evolution（SDE）[25]，Fuzzy Adaptive Differential Evolution（FADE）[63] を用いて，NF1 を最適化する実験を 1000 試行行った際に得られる最良解平均の推移である．この図の SSDE-SP の曲線を確認すると，ある程度探索が進んだ段階から横ばいとなっており，全試行において探索が停滞していることが読み取れる．もし SSDE-SP が適切にパラメータ調節を行っていたと仮定するならば，探索が停滞した後にパラメータの再設定が何度か行われ，現状に適切なパラメータに当たった段階で探索が再度進行することから，この曲線は緩やかにはなれど

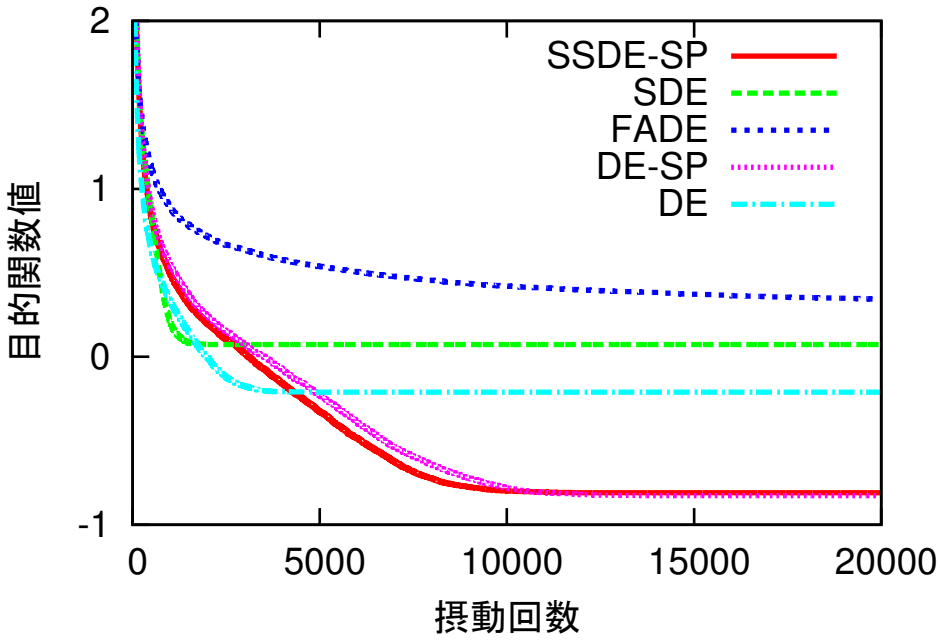


図 4.2: NF1 最適化時の最良解平均（文献 [2] より引用）

も右下がりであり続けることになる．しかしながらこの図では仮定の通りにはなっておらず，実際にはパラメータの再設定が適切に行われていないと考えられる．さらに，実験時に最適解の探索に失敗した試行の F，C_R の変遷を確認したところ図 4.3 の事例のように，探索初期にはパラメータの再設定がなされていても，探索中期以後に探索が停滞しているにも関わらずパラメータの再設定が行われない挙動が確認された．以上のことから，SSDE-SP は探索停滞時にパラメータの再設定を適切に行うことが出来ないため，本研究の目的を達成し得る DE-SP の改良として妥当ではないと考えられる．

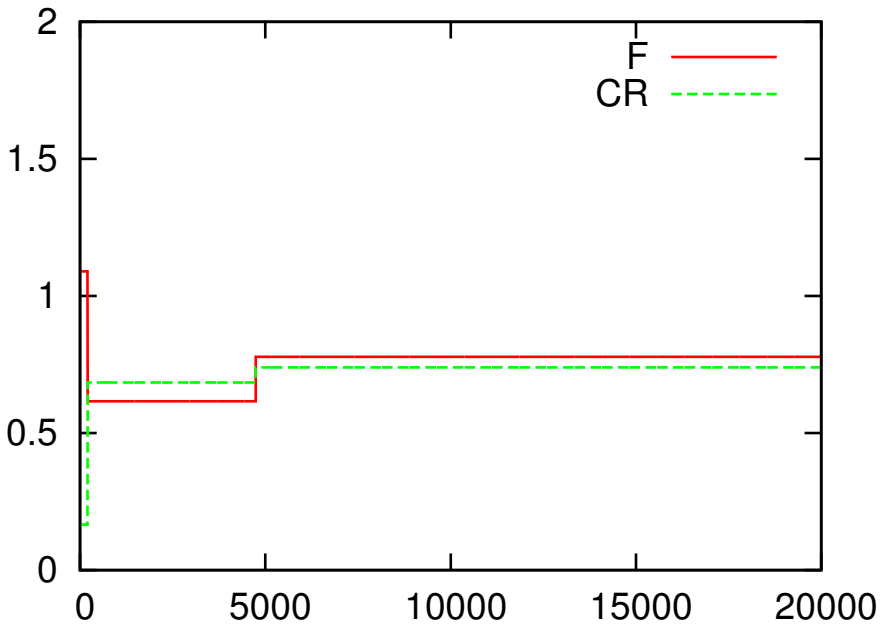


図 4.3: SSDE-SP を用いて NF1 の探索に失敗した場合の F，C_R の変遷例

アルゴリズム 24 Simple Self-Adaptive Differential Evolution on Scattered Parents

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F \in [0,2] \cdots$ スケーリングパラメータ
$C_R \in [0,1] \cdots$ 交叉時に用いる閾値
$M \cdots$ 改悪を受理する個体数
$T_F \cdots$ 連続失敗回数
$T_{F_{\max}}$ 連続失敗回数上限
$R_{\mathcal{U}}(\mathbb{R}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の実数一様乱数
$R_{\mathcal{U}}(\mathbb{N}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の自然数一様乱数

$F \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0,2])$，$C_R \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0,1])$，$M \leftarrow 0$，$T_F \leftarrow 0$ とパラメータを初期化
for $i = 1$ to N **do**
　個体 $\boldsymbol{\chi}^i$ を乱数で初期化
　目的関数 $E(\boldsymbol{\chi}^i)$ を計算
end for
while 終了条件が未成立 **do**
　for $i = 1$ to N **do**
　　$l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
　　for $k = 1$ to D **do**
　　　サブ親 $\boldsymbol{\chi}^{a_k}$, $\boldsymbol{\chi}^{b_k}$, $\boldsymbol{\chi}^{c_k}$ $(i \neq a_k \neq b_k \neq c_k)$ をランダムに選択
　　　突然変異個体を計算: $\chi'^i_k \leftarrow \chi^{a_k}_k + F(\chi^{b_k}_k - \chi^{c_k}_k)$
　　　if $R_{\mathcal{U}}(\mathbb{R}, [0,1]) < C_R$ **or** $k = l^i$ **then**
　　　　子個体候補の要素を次のように設定: $p^i_k \leftarrow \chi'^i_k$
　　　else
　　　　子個体候補の要素を次のように設定: $p^i_k \leftarrow \chi^i_k$
　　　end if
　　end for
　　目的関数 $E(\boldsymbol{p}^i)$ を計算
　end for
　for $i = 1$ to N **do**
　　if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **then**
　　　親個体を子個体候補と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
　　　$T_F \leftarrow 0$
　　else if $E(\boldsymbol{\chi}^i)$ が最悪値から M 番目までの値である **then**
　　　親個体を子個体候補と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
　　else
　　　親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
　　　$T_F \leftarrow T_F + 1$
　　　if $T_F > T_{F_{\max}}$ **then**
　　　　$F \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0,2])$，$C_R \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0,1])$ と再設定
　　　　$M \leftarrow M + 1$
　　　　if $M \geq N$ **then**
　　　　　$M \leftarrow 0$
　　　　end if
　　　end if
　　end if
　end for
end while

4.3 Self-Adaptive Differential Evolution on Scattered Parents with Random Jump

4.3.1 Self-Adaptive Differential Evolution on Scattered Parents with Random Jump のアルゴリズム

SSDE-SP の後，改良すべき点を再考した手法である Self-Adaptive Differential Evolution on Scattered Parents with Random Jump（SDE-SP-RJ）の疑似コードをアルゴリズム 25 に示す．SDE-SP-RJ では始めに F，C_R，M が乱数で初期化され，その後，原則として DE-SP と同様の手順で探索が進められる．DE-SP からの差異は，目的関数が最悪値から M 番目である個体を各摂動における突然変異計算の開始時に再度乱数を用いて初期化する点（アルゴリズム 25，24 行目）と，最後に最良解が更新されてからの摂動回数 T_S を記録（アルゴリズム 25，45 から 49 行目）し，T_S が閾値 $T_{S_{\max}}$ を超えた場合は摂動の開始時に F，C_R，M を再度乱数を用いて振り直す点（アルゴリズム 25，18 から 21）である．すなわち，この T_S は SDE-SP-RJ において探索の停滞を検出する指標として用い，$T_S > T_{S_{\max}}$ となった時に探索が停滞していると判断してパラメータを再設定し，探索の継続が可能なパラメータを模索する設計となっている．

また，当初提案していた SSDE-SP では探索停滞を検出する指標として，連続失敗回数 T_F を用いていたが，この指標は局所解に収束する過程でも容易に 0 へとリセットされてしまう指標[2]であり，探索停滞を検出する指標として適切ではない可能性があるため，SDE-SP-RJ では採用しなかった．

4.3.2 ベンチマーク最適化実験による性能比較と評価

SDE-SP-RJ の有効性を評価するため，ベンチマークを用いた最適化実験を行う．本実験においては 2.5 にて用いた Noisy Function 1（NF1），Noisy Function 2（NF2）を最適化することによって評価を行う．

4.3.3 比較手法

本実験では，比較手法として以下の手法を用いる．

- NF1，NF2 それぞれに対して A.4 節のパラメータを設定した Differential Evolution

[2]例えばある最良解へ他の最良解でない個体が移動しようとする過程でも 0 となってしまう．

表 4.3: 各空間における最適解発見回数割合 [%]

	DE	DE-SP	SDE-SP-RJ	SDE	FADE
NF1	24.3	**76.3**	65.6	5.7	0
NF2	**100**	**100**	95.7	92.9	0

- NF1，NF2 それぞれに対して A.4 節のパラメータを設定した Differential Evolution on Scattered Parents
- Self-adaptive Differential Evolution（SDE） [25]
- Fazzy Adaptive Differential Evolution（FADE） [63]

SDE と FADE の詳細は 4.1.2 節にて述べた通りである．

4.3.4　実験設定

NF1 と NF2 を，SDE-SP-RJ と 4.3.3 にて述べた手法を用いて最適化を行い，結果を比較する．実験設定として，試行回数を 1000 回，各試行の摂動回数上限を 300000 回とし，SDE-SP-RJ において $T_{S_{\max}} = 3000$（摂動回数の 1%）とした．結果の評価指標は 2.5 と同様に最適解発見回数割合とする．

4.3.5　実験結果と考察

実験結果を表 4.3 に，各探索空間を最適化した時の最良解平均の推移を図 4.4 に示す．表 4.3 から，最適に調整された DE-SP には劣るものの，SDE や FADE と比較した場合には性能が高いことが確認される．また，NF1 においては最適に調整された DE よりも結果が良いため，SDE-SP-RJ は DE 類縁手法を適用する場合において選択肢として挙げられ得る性能であることが示唆される．また，FADE において図 4.4 を確認すると，両探索空間においてグラフが右下がりであり探索が継続されていることが確認されるため，探索は継続されるが最適解への収束能力が低いために，両探索空間における最適解発見回数割合が 0%であることが示唆される．SDE において，NF2 では SDE-SP-RJ と同等の性能を示すものの，この NF2 での結果と比較して NF1 の最適解発見回数割合が大きく低下している．さらに，図 4.4(a) において SDE のグラフは横這いであるため，NF1 程度の複雑性を持つ探索空間では SDE では探索の継続が不可能であることが示唆される．また，図 4.4(b) においても SDE のグラフは横這いであるので，本研究で取り上げた動的にパラメータを調整する DE 類縁手法の中では探索継続能力は最も低いことが示唆される．

表 4.4: 各探索空間における各 $T_{S_{\max}}$ の値ごとの最適解発見回数割合 [%]

$T_{S_{\max}}$	750	1500	2250	3000	6000	9000	12000	15000
NF1	63.1	63.8	62.9	**65.6**	59.4	55.3	49.6	46.7
NF2	94.5	94.6	94.8	95.1	95.4	95.3	**95.7**	95.2

4.4　パラメータ変化による性能変化調査

SDE-SP-RJ は手法実装者により設定が必要なパラメータとして $T_{S_{\max}}$ が残っている．SDE-SP-RJ が DE-SP のパラメータ設定の困難性を少なくとも緩和するものであるためには，この $T_{S_{\max}}$ が DE-SP の F，C_R よりも寛容であると示唆される必要がある．$T_{S_{\max}}$ が実際にこの要件を満たし得るか確認するため，$T_{S_{\max}}$ の変化に対する SDE-SP-RJ の探索能力の変化を確認する．

4.4.1　実験設定

探索空間は前実験と同様に NF1，NF2 とする．試行回数は 1000 回とし，各試行の摂動回数上限を 300000 回，結果の評価指標は最適解発見回数割合とする．この条件の実験を，$T_{S_{\max}}$ =750，1500，2250，3000，6000，9000，12000，15000 とした場合において行い，結果を比較する．

4.4.2　実験結果と考察

実験結果を表 4.4 に，各探索空間を最適化した時の最良解平均の推移を図 4.5 に示す．表 4.4 から，NF1 においては $T_{S_{\max}} = 750$ から $T_{S_{\max}} = 3000$ までは最適解発見回数割合が 60%前半辺りで上下し，それより $T_{S_{\max}}$ が大きい領域では，$T_{S_{\max}}$ が大きいほど下降していくことが確認される．また NF2 における結果から，本実験で確認を行った $T_{S_{\max}}$ の範囲では性能は大きく変化しないことが確認される．直接探索法を適応する場合において，NF1 と NF2 とでは NF2 の方が比較的最適化が容易な探索空間であり，すなわち NF1 はより比較的最適化が困難な探索空間である．したがって，$T_{S_{\max}}$ が小さいと SDE-SP-RJ は有効に機能し，$T_{S_{\max}}$ が有効な領域は問題が複雑な方が狭くなることが予想される．

図 4.5(a) を確認すると，$T_{S_{\max}}$ の値によって傾きは変わるものの，全ての場合においてグラフが右下がりとなっていることから，探索が継続されていることが確認できる．また，表 4.4 での結果は $T_{S_{\max}} = 3000$（1%）が最も良かったが，図 4.5(a) では $T_{S_{\max}} = 750$（0.25%）のグラフが最も下に来ている．そして $T_{S_{\max}} = 2250$（0.75%）と $T_{S_{\max}} = 3000$（1%）が入れ替わっている点を除き，$T_{S_{\max}}$ が小さい順により下をグラフが遷移していることが確認できる．このため，必ずとは言い切れないが，傾向として $T_{S_{\max}}$ が小さいほど SDE-SP-RJ の探索継続能力は高くなると考えられる．一方で $T_{S_{\max}} = 750$（0.25%）

の最適解発見回数割合が最大でない理由は，$T_{S_{\max}}$ が低すぎるため，最適解周辺にある程度個体が集まり，最適解へ収束する過程での探索失敗が原因でパラメータが再設定されてしまい，収束を阻害してしていることが一要因として考えられる．

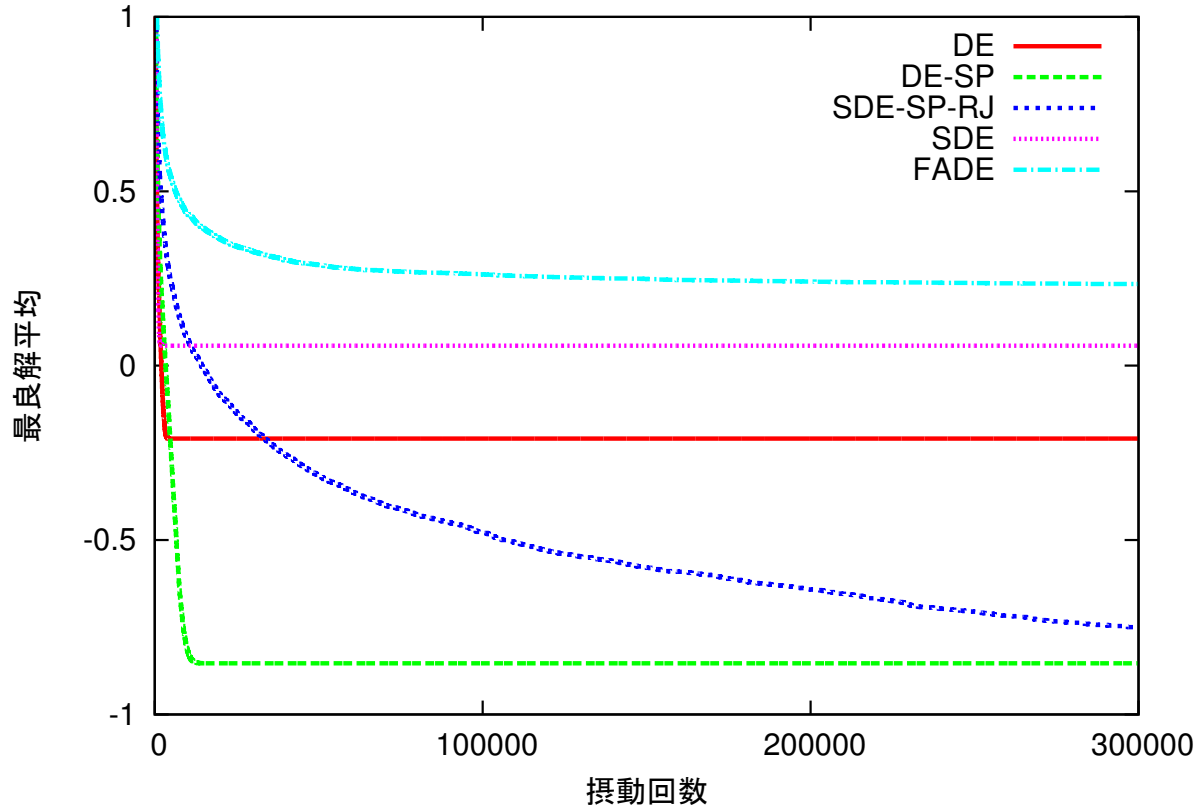
図 4.5(b) を確認すると，僅かずつグラフが下降してはいるものの，NF1 の時とは逆に $T_{S_{\max}} = 750$（0.25%）や $T_{S_{\max}} = 1500$（0.50%）は探索が途中でほぼ停滞してしまっている．一方で $T_{S_{\max}} = 15000$（5%）は，非常に緩やかではあるが僅かずつグラフが下降していることが確認できる．これは先述した，最適解への収束を阻害している要因が影響した結果であると考えられる．また探索序盤に，$T_{S_{\max}}$ が大きいほど大きな段差部分が見られるが，これは NF2 が NF1 よりも探索が容易に行えることが原因で発生しているものと考えられる．発生機序は次の通りである．まず，探索最序盤は多少悪いパラメータでも探索が進む，そのため図 4.5(b) の最良解平均は全ての場合において急落する．次に，最序盤で用いたパラメータでは探索が停滞する段階に突入する．ここでパラメータの振り直しが何度か行われ，この間最良解は更新されない．続いて，探索の継続可能なパラメータが発見された場合，探索が再開され，最良解は再び更新される．これらの「最序盤の探索 → 探索の停滞 → 探索の再開」の各場面が各々試行において近い摂動回数で発生するため，最良解平均もその影響を受け図 4.5(b) のような段差が発生する．ここで，「探索の停滞」の場面の長さは $T_{S_{\max}}$ が長いほど，パラメータを振り直す間隔が長くなるために長くなる．したがって，$T_{S_{\max}}$ の大きなものほど段差部分は大きくなる．このような機序による探索の進行が発生している例として，$T_{S_{\max}} = 15000$（5%）における最良解の変動を図 4.6 に五例示す．これらの例で発生しているような最良解推移が，最良解平均の推移に影響していると考えられる．

詳細な考察としては以上の通りである．ここで，改めて表 4.4 を確認する．大雑把に見るならば，NF1 においては $T_{S_{\max}} \leq 3000$（1%）までは最適解発見回数割合 60%代の性能を維持しており，NF2 に至っては本実験の範囲において性能の急落は見られない．このことから本実験において，SDE-SP-RJ の $T_{S_{\max}}$ は DE-SP の F，C_R と比較して寛容なパラメータとなっていることが示唆され，本研究ではこの結果を文献 [62] へとまとめ，一度の区切りとしていた．しかしながら，4.1.1 節にて述べたように，パラメータ設定の負担を軽減することを目的とするならば，その理想とすべきところはパラメータ設定の必要性の完全な撤廃である．そこで本研究では，手法の改良方法を改めて考察し直し，パラメータ設定の必要性を完全に撤廃した SDE-SP-DR を改めて設計し，これを本稿における最終的な提案手法とする．SDE-SP-DR の具体的手順，有効性を確認する実験とその考察については次節である 4.5.1 にて述べる．

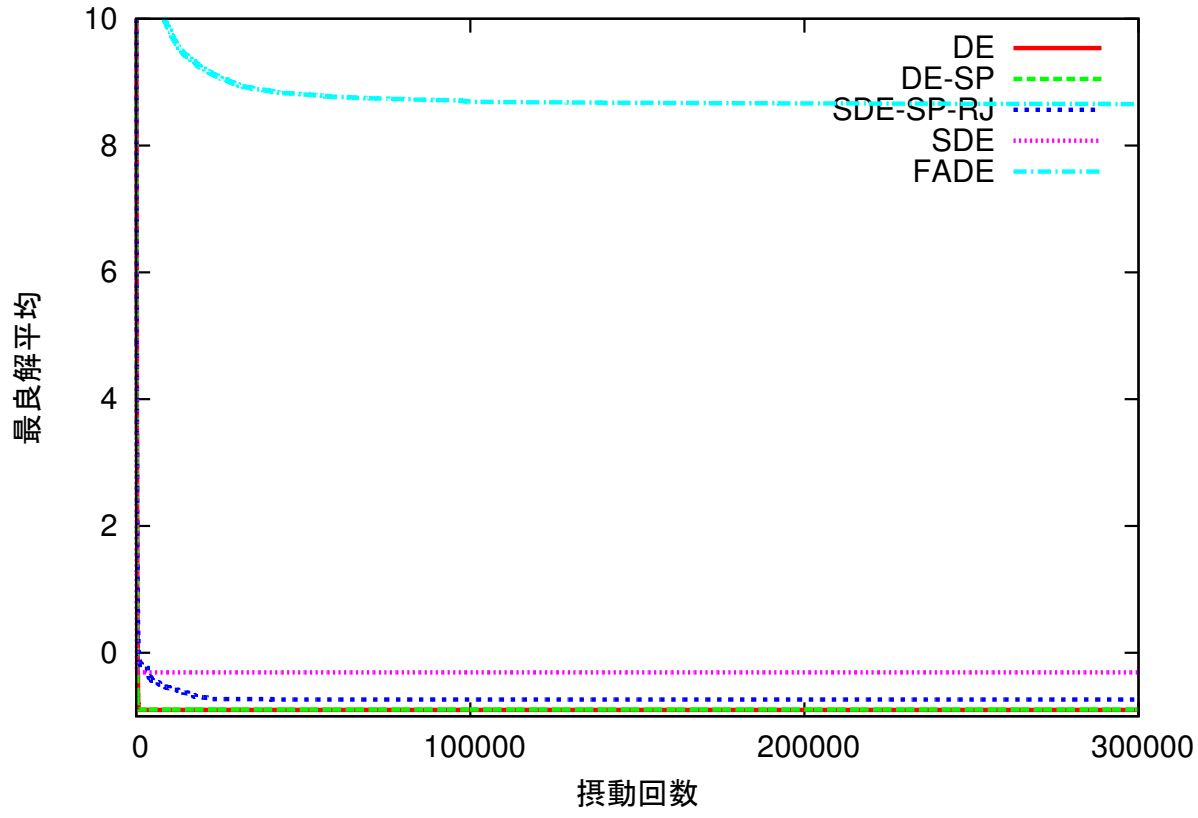
アルゴリズム 25 Self-Adaptive Differential Evolution on Scattered Parents with Random Jump

$D \cdots$ 探索空間の次元数
$N \cdots$ 個体数
$F \in [0, 2] \cdots$ スケーリングパラメータ
$C_R \in [0, 1] \cdots$ 交叉時に用いる閾値
$M \cdots$ 改悪を受理する個体数
T_S 最良解更新からの摂動回数
$T_{S_{\max}}$ 最良解更新からパラメータ F，C_R，M の振り直しを待つ最大摂動回数
$R_{\mathcal{U}}(\mathbb{R}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の実数一様乱数
$R_{\mathcal{U}}(\mathbb{N}, [\alpha, \beta]) \cdots$ 範囲 $[\alpha, \beta]$ の自然数一様乱数

F，C_R，M を乱数で初期化: $F \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0, 2])$，$C_R \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0, 1])$，$M \leftarrow R_{\mathcal{U}}(\mathbb{N}, [0, N])$
T_S を初期化: $T_S \leftarrow 0$
for $i = 1$ to N **do**
    個体 $\boldsymbol{\chi}^i$ を乱数で初期化
    目的関数 $E(\boldsymbol{\chi}^i)$ を計算
end for
while 終了条件が未成立 **do**
    if $T_S \geq T_{S_{\max}}$ **then**
        F，C_R，M を乱数を用いて振り直す: $F \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0, 2])$，$C_R \leftarrow R_{\mathcal{U}}(\mathbb{R}, [0, 1])$，$M \leftarrow R_{\mathcal{U}}(\mathbb{N}, [0, N])$
        T_S を初期化: $T_S \leftarrow 0$
    end if
    for $i = 1$ to N **do**
        if $E(\boldsymbol{\chi}^i)$ が最悪値から M 番目までの値である **then**
            個体 $\boldsymbol{\chi}^i$ を乱数で初期化
        end if
        $l^i \leftarrow R_{\mathcal{U}}(\mathbb{N}, [1, D])$
        for $k = 1$ to D **do**
            サブ親 $\boldsymbol{\chi}^{a_k}, \boldsymbol{\chi}^{b_k}, \boldsymbol{\chi}^{c_k}$ $(i \neq a_k \neq b_k \neq c_k)$ をランダムに選択
            突然変異個体を計算: $\chi'^i_k \leftarrow \chi^{a_k}_k + F(\chi^{b_k}_k - \chi^{c_k}_k)$
            if $R_{\mathcal{U}}(\mathbb{R}, [0, 1]) < C_R$ **or** $k = l^i$ **then**
                子個体候補の要素を次のように設定: $p^i_k \leftarrow \chi'^i_k$
            else
                子個体候補の要素を次のように設定: $p^i_k \leftarrow \chi^i_k$
            end if
        end for
        目的関数 $E(\boldsymbol{p}^i)$ を計算
    end for
    for $i = 1$ to N **do**
        if $E(\boldsymbol{p}^i) < E(\boldsymbol{\chi}^i)$ **or** $E(\boldsymbol{\chi}^i)$ が最悪値から M 番目までの値である **then**
            親個体を子個体候補と置換: $\boldsymbol{\chi}^i \leftarrow \boldsymbol{p}^i$
        else
            親個体 $\boldsymbol{\chi}^i$ がそのまま子個体となり次の摂動に引き継ぎ
        end if
    end for
    if 最良解が更新 **then**
        T_S を初期化: $T_S \leftarrow 0$
    else
        T_S をインクリメント: $T_S \leftarrow T_S + 1$
    end if
end while

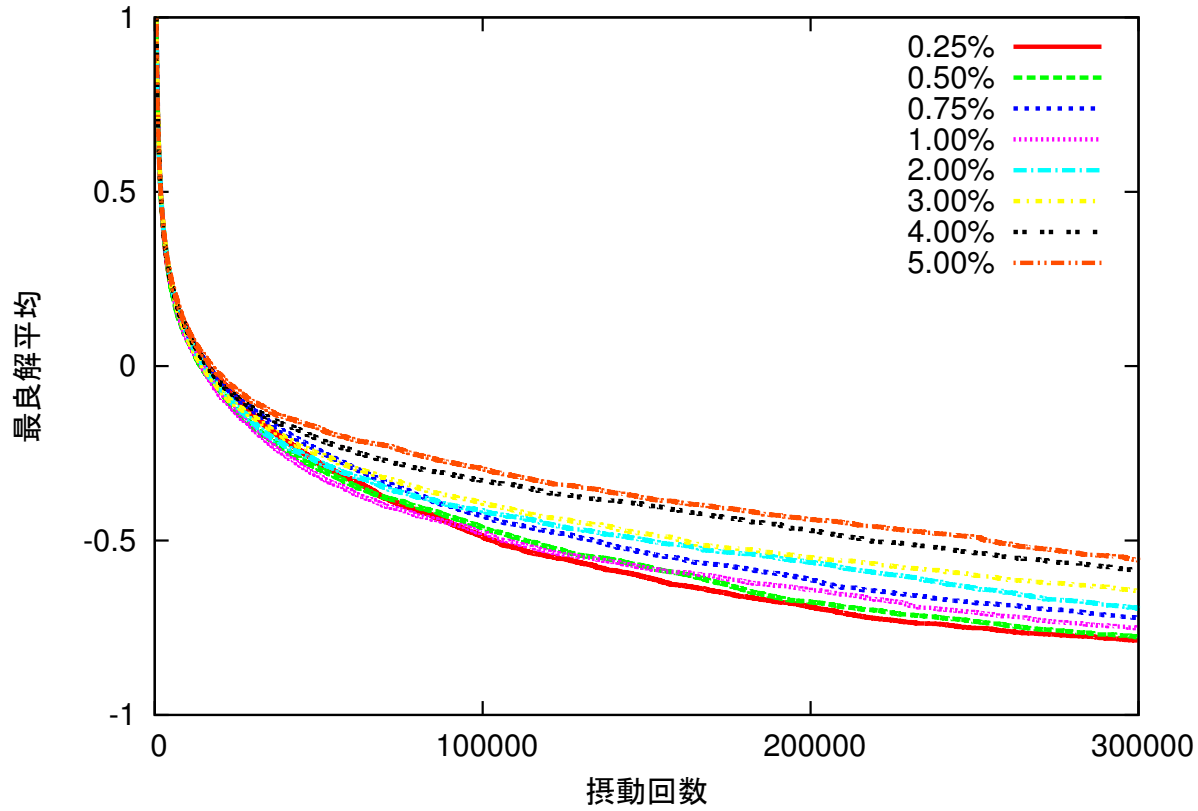


(a) Noisy Function 1

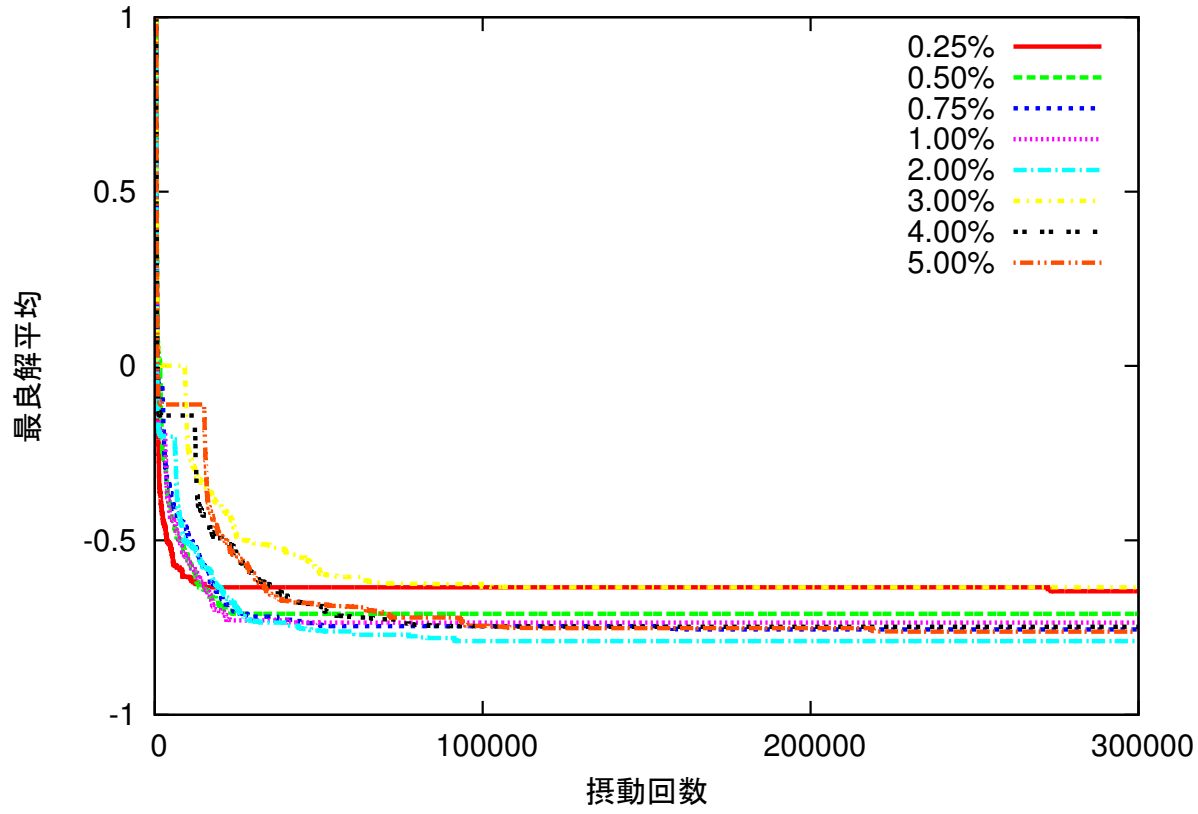


(b) Noisy Function 2

図 4.4: 各探索空間における各手法の最良解平均の推移



(a) Noisy Function 1



(b) Noisy Function 2

図 4.5: 各探索空間における各 $T_{S_{\max}}$ 毎の SDE-SP-RJ の最良解平均の推移

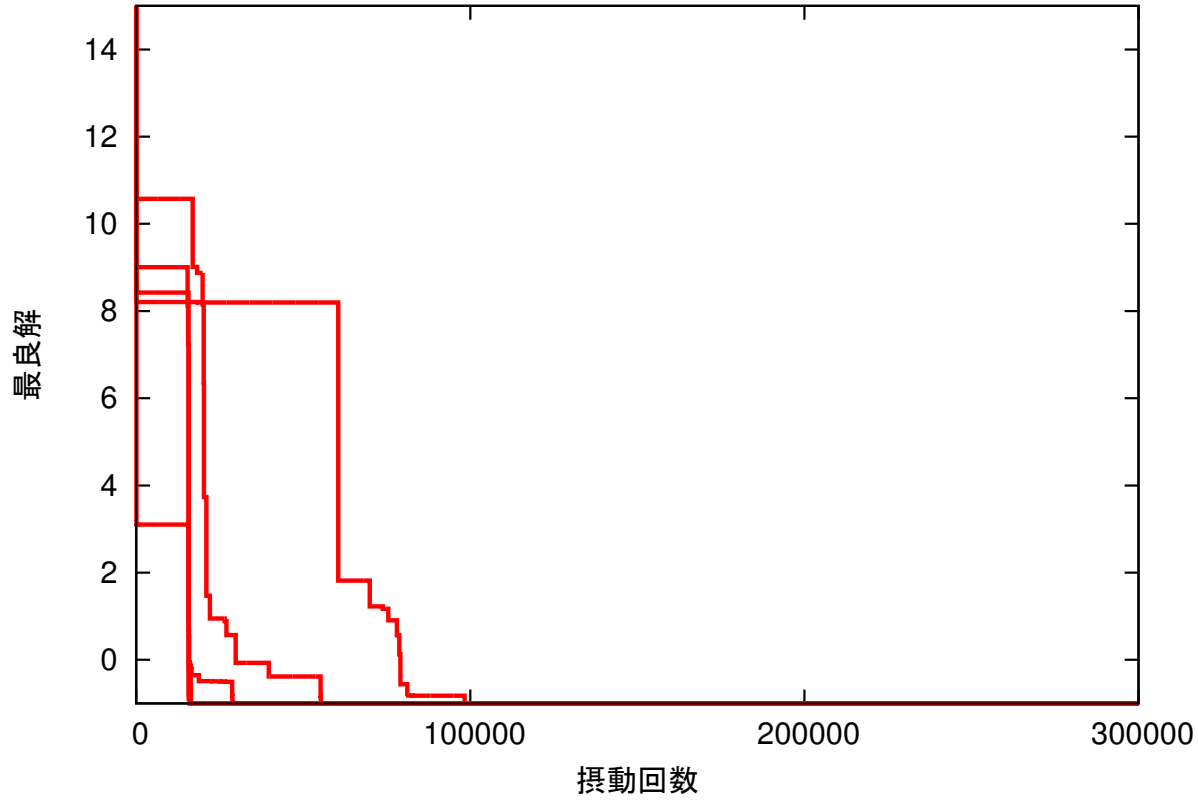


図 4.6: $T_{S_{\max}} = 15000$（最大摂動回数の 5%）時の階段状に遷移する最良解変動例

4.5 Self-adaptive Differential Evolution on Scattered Paretns with Dynamic Restart

4.5.1 手法概説

本研究の現状においては最終的な到達点であり，本稿の提案手法である SDE-SP-DR の疑似コードをアルゴリズム 26 に示す．SDE-SP-DR に施された改変について詳細を述べる前に，DE-SP で存在していた M が SDE-SP-DR では廃止されていることを予め明記する．この意図は個体群の再初期化に関わる説明部にて述べる．ここから，アルゴリズム 26 の順序に従って，DE-SP，そして SDE-SP-RJ からの改変点を述べていく．まず，F と C_R が乱数によって設定される点は SDE-SP-RJ と同等の発想による DE-SP からの改変である（アルゴリズム 26, 15 行目）．次に，乱数によって設定された F と C_R は現在の摂動を終える時,「全親個体が子個体候補と置換されなかった場合」に乱数によって再設定される（アルゴリズム 26, 37 行目，38 行目）．この改変の意図は，探索が停滞した場合にパラメータの再設定が発生することを期待したものである．しかし，この「全親個体が子個体候補と置換されなかった場合」は基準としては不確実なものであり，必ずしも探索が停滞していないにも関わらずパラメータの再設定が頻繁に発生することが予測される．このような状況が予想されながら本研究においてこの基準を採用した理由は，報告 [25] にて提案されている Self-adaptive Differential Evolution の性質にある．Self-adaptive Differential Evolution では，F と C_R の二つのパラメータがただの一様乱数と同等な程度に高い頻度で再設定されているにも関わらず，高い性能を示し得る [61] ことが確認された．従って，F，C_R の高頻度での振り直しは必ずしも探索性能の低下に直結しないと予想し,「探索が停滞していても振り直しが発生しない可能性がある基準」よりは,「探索が停滞していなくとも振り直しは発生するが，停滞していた場合には速やかに振り直しが発生する基準」の方が，改変の試みとして妥当であると，手法設計者は判断したのである．続いて，F と C_R の再設定判定が終わった後,「最良解の目的関数値と最悪解の目的関数値が等しい場合」は個体群が再初期化され，探索が再開される（アルゴリズム 26，40 行目，41 行目）．この改変の意図は，個体群がある局所解に収束した場合に探索が自動的にやり直されるようにすることである．この基準を探索の停滞（即ち，F と C_R の再設定判定）と分けた理由は，個体群が局所解に収束するに当たり，探索が停滞しているように捉えられる状況に何度も陥ることがあるためである．このため，探索が停滞した場合に即座に探索をやり直してしまうと，個体群の収束が不十分なまま探索が初期化され得る．この結果として最適解の発見がいつまで経っても為されないという事態が容易に予想される．従って，SDE-SP-DR においては，探索の停滞の判定よりは厳しい基準を持って個体群の再初期化を行うよう手法設計者は改変した．ただし，こちらの基準は恣意的に設定されたものであることは明記しておく．またこの改変により全個体が探索領域に撒かれ直すこととなるため，DE-SP では局所解からの離脱を補助するためのパラメータであった M は SDE-SP-DR において非常に効果の薄いものとなった．このため，SDE-SP-DR では M を廃した．M を廃すことにより，SDE-SP-DR の個体群はその全てが探索に寄与するため，短期的な探索能力の低下を抑える効能も期待で

きる．

以上が本提案手法 SDE-SP-DR の概要である．SDE-SP-DR では手法設計者によるパラメータは完全に廃された．このため本研究の目的と経緯に沿うならば，本手法の性能が DE-SP より急落しておらず SDE-SP-RJ と性能が同等以上であれば，手法設計者のパラメータ設定の困難を解決する手法として SDE-SP-DR は SDE-SP-RJ よりも適切な手法であると考えることができるだろう．このことを確認するため，最適化実験による性能比較を行った．次節にて，実験の詳説と考察を行う．

4.5.2　最適化実験による性能比較

対象とする問題

本研究において，SDE-SP-DR の性能を確認するために用いた探索空間は以下通りである．

- NF1，2
- 2，20，50 次元の Rastrigin 関数，Rosenbrock 関数，Schwefel 関数，Griwank 関数，Ackley 関数
- Lennard-Jones Clusters Optimization Problem（LJ-Problem）

NF1，2 と 5 種のベンチマーク関数の詳細は 2.5.1 節にて述べられているため，ここでは LJ-Problem について述べる．

Lennard-Jones Clusters Optimization Problem（LJ-Problem）は，Lennard-Jones Potential [69] の総和が最小となる粒子の配置を求める問題である．Lennard-Jones Potential は 2 粒子間の相互作用ポテンシャルエネルギーのフィッティングによる近似曲線と解釈され，次のように表される．

$$V_{\mathrm{LJ}}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{p} - \left(\frac{\sigma}{r}\right)^{q} \right] \tag{4.28}$$

ここで，r は粒子間距離であり，σ は $V_{\mathrm{LJ}} = 0$ となる $r = \infty$ 以外の（即ち自明ではない）r，ϵ はポテンシャル井戸の深さ，p は斥力項を調整するパラメータ，q は引力項を調整するパラメータである．本実験では，$p = 12$，$q = 6$，$\epsilon = \sigma = 1$ という最も単純とされるモデルを用いる．

$$V_{\mathrm{LJ}}(r) = 4\left(\frac{1}{r^{12}} - \frac{1}{r^{6}}\right) \tag{4.29}$$

ここで，粒子の数を N とした場合，目的関数は次のように表される．

$$E(\boldsymbol{\chi}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} V_{\mathrm{LJ}}(||\boldsymbol{\chi}_i - \boldsymbol{\chi}_j||) \tag{4.30}$$

LJ-Problem における個体 $\boldsymbol{\chi}$ は，粒子の三次元座標が N 個格納された行列とする．

$$\boldsymbol{\chi} = \left(\begin{array}{c|c|c|c} \boldsymbol{\chi}_1 & \boldsymbol{\chi}_2 & \cdots & \boldsymbol{\chi}_N \end{array} \right) \tag{4.31}$$

したがって，N 粒子の問題を解く場合，最適化手法から確認される目的関数の次元は $3N$ 次元となる．

全ての実験とその結果について纏めて記述すると煩雑となるため，本節では全最適化実験を，NF1，2 の最適化実験，他 5 種類のベンチマーク関数最適化実験，LJ-Problem の三つに分け，それぞれに対して実験設定，結果，考察を述べる．

Noisy Function の最適化実験

本節では 2 章の最適化実験にて用いられた 2 種の Noisy Function を用いた最適化実験による結果比較を行い，SDE-SP-DR の性能を確認する．まず，最適化の環境は以下の通りである．

次元 2 次元

摂動回数上限 300000 回

個体数 50

試行回数 1000 回

性能指標 最適解発見回数割合

最適解発見成否判断基準 個体 $\boldsymbol{\chi}$ に対して目的関数 $E(\boldsymbol{\chi})$ を計算した場合に倍精度（64bit）符号小数点演算において丸め誤差が発生することで，最適値 $E(\boldsymbol{\chi}^*)$ と等号比較により一致するか否かを確認し，一致した場合に成功したとみなす.[3]

また，比較手法は以下の通りである．

- Liu らによる Fuzzy Adaptive Differential Evolution（FADE） [63]
- Orman らによる Self-adaptive Differential Evolution（SDE） [25]
- Noman らによる DE の改良（Noman） [64]

[3]すなわち，最適化手法が倍精度符号小数点演算を用いた環境においてこれ以上目的関数を小さくする手がかりが手に入れなくなった場合にのみ，最適解の発見に成功したとみなす．NF1 を例とすると，2 次元でかつ 4 点バイリニア法により計算されるため，この値は 2 次の項の和となっており，最適解，最適値は $E(\mathbf{0}) = -\mathbf{1}$ である．したがって，NF1 の定義域において倍精度符号小数点演算の上では，最適解から各軸方向それぞれに対して約 10^{-17} のずれが許容される．

- Soliman らによる DE の改良（Soliman） [65]
- Brest らによる DE の改良（Brest） [66]
- 山口による DE/rand/1/exp の改良（山口） [67]
- パラメータを各目的関数に対して調整を施した[6]DE [19]
- パラメータを各目的関数に対して調整を施した[6]DE-SP [70]
- $T_{S_{\max}} = 3000$（最大摂動回数の 1%）とした SDE-SP-RJ [61]
- DE に本稿での提案手続きを適用した手法（SDE-DR）
- DE/nrand/1 [32] に本稿での提案手続きを適用した手法（SDEnrand-DR）
- DE/isolated/1 [33] に本稿での提案手続きを適用した手法（DEiso-DR）

実験結果を表 4.5 に，NF1，NF2 最適化実験時の最良解平均をそれぞれ図 4.7，図 4.8 に示す．まず NF1 において，SDE-SP-DR は従来の改良である DE-SP や SDE-SP-RJ よりも最適解発見回数割合が高く，パラメータ再設定や探索の再初期化が有効に機能している事が示唆される．特に，SDE-SP-RJ のパラメータ $T_{S_{\max}}$ を最大摂動回数の 1%と設定したものは，文献 [61] の実験にて確認されたパラメータの中では最も高い性能を示すもの（表 4.3 あるいは表 4.4 を参照）であり，しかしながら本稿で提案している SDE-SP-DR はその最適解発見回数割合よりも高いものとなっている．また図 4.7 を確認すると SDE-SP-DR のグラフは SDE-SP-RJ のものを下回るようにかつ右下がりであり続けるため，探索の継続能力も SDE-SP-RJ より高いことが示唆される．したがって，SDE-SP-DR は，パラメータ設定が不要でありかつ性能がより高いという点において，SDE-SP-RJ の上位互換となり得ることが示唆される．提案手続きを適用した他手法については，SDE-DR が 77%の最適解発見回数割合となったことと比較し，SDEnrand-DR は 12%，SDEiso-DR は 0%と低い値となっている．よって本提案手続きは，これを適用可能な DE 類縁手法全てに有効とは言えないことが示唆され，本稿で比較に用いていない手法に適用する場合にはこのことに留意が必要であると考えられる．関連手法の最適回発見回数割合は SDE-SP-DR や SDE-DR と比べて低く，図 4.7 を確認すると山口による改良以外のグラフは横這いとなっており，探索が停滞していることが確認できる．また，山口による改良は探索の継続が可能であることは確認されるものの，他の手法と比べ高い位置で最良解平均が推移しており，収束速度性能が低いことが示唆される．したがって，これらの手法は大域的単峰性に乏しい探索空間においては，次元の小さな空間においても探索が困難となる可能性が示唆される．次に NF2 において，DE や従来の改良である DE-SP は 100%の最適解発見回数割合を示すのに対し，SDE-SP-DR は 99.5%と下がっている．この値は SDE-SP-RJ や他の関連手法と比較すると高い値であり，適応的なパラメータ調整を試みている手法の中では最も高い性能である．このようにこの結果を解釈され得る一方，DE から DE-SP へ改良するにあたっては 100%の最適解発見回数割合の維持に成

[6]各目的関数に対してパラメータ F，C_R をそれぞれ 0.1 刻みで総当たりを行い，最も性能が高かったものを本来の性能とする．

表 4.5: 各 Noisy Function 最適化実験での最適解発見回数割合 [%]

	SDE-SP-DR	SDE-DR	SDEnrand-DR	SDEiso-DR	FADE	SDE	Noman
NF1	88.7	77	12	0	0	5.7	0
NF2	99.5	100	87	0	0	92.7	0
	Soliman	Brest	山口	DE	DE-SP	SDE-SP-RJ	
NF1	2.5	4.9	0	24.2	77.9	65.6	
NF2	70.3	90.0	0	100	100	95.1	

功しているのに対し，DE-SP から SDE-SP-DR へ改良するにあたってはこの維持に失敗していると解釈することも可能である．したがって，SDE-SP-DR の収束性能は DE-SP と比較して顕著に低く，使用するにあたっては十分大きな摂動回数上限を設定する必要があることが示唆される．

アルゴリズム 26 Self-adaptive Differential Evolution on Scattered Paretns with Dynamic Restart の疑似コード

```
D ··· 目的関数の次元数
N ··· 個体数
F ∈ [0, 2] ··· スケーリングパラメータ
C_R ∈ [0, 1] ··· 交叉時に用いる閾値
M ··· 改悪を受理する個体数
T_S 最良解更新からの摂動回数
T_{S_max} 最良解更新からパラメータ F，C_R，M の振り直しを待つ最大摂動回数
R_U(ℝ, [α, β]) ··· 範囲 [α, β] の実数一様乱数
R_U(ℕ, [α, β]) ··· 範囲 [α, β] の自然数一様乱数

for i = 1 to N do
    個体 χ^i を乱数で初期化
    目的関数 E(χ^i) を計算
end for
F，C_R を乱数で初期化: F ← R_U(ℝ, [0, 2]), C_R ← R_U(ℝ, [0, 1])
while 終了条件が未成立 do
    for i = 1 to N do
        l^i ← R_U(ℕ, [1, D])
        for k = 1 to D do
            サブ親 χ^{a_k}, χ^{b_k}, χ^{c_k} (i ≠ a_k ≠ b_k ≠ c_k) をランダムに選択
            突然変異個体を計算: χ'^i_k ← χ^{a_k}_k + F(χ^{b_k}_k − χ^{c_k}_k)
            if R_U(ℝ, [0, 1]) < C_R or k = l^i then
                子個体候補の要素を次のように設定: p^i_k ← χ'^i_k
            else
                子個体候補の要素を次のように設定: p^i_k ← χ^i_k
            end if
        end for
        目的関数 E(p^i) を計算
    end for
    for i = 1 to N do
        if E(p^i) < E(χ^i) then
            親個体を子個体候補と置換: χ^i ← p^i
        else
            親個体 χ^i がそのまま子個体となり次の摂動に引き継ぎ
        end if
    end for
    if 全親個体が子個体候補と置換されなかった then
        F，C_R を乱数を用いて再設定: F ← R_U(ℝ, [0, 2]), C_R ← R_U(ℝ, [0, 1])
    end if
    if min E(χ) = max E(χ) then
        個体群の再初期化
    end if
end while
```

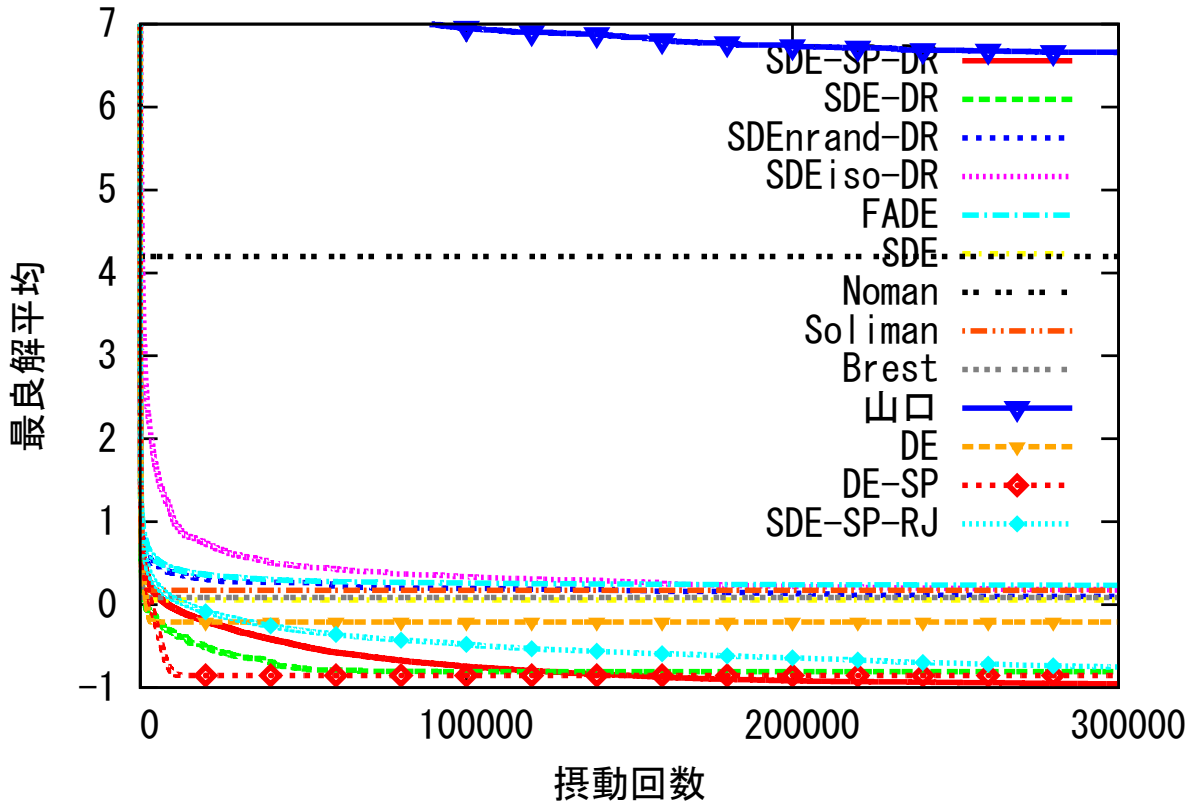


図 4.7: NF1 最適化時の各手法の最良解平均

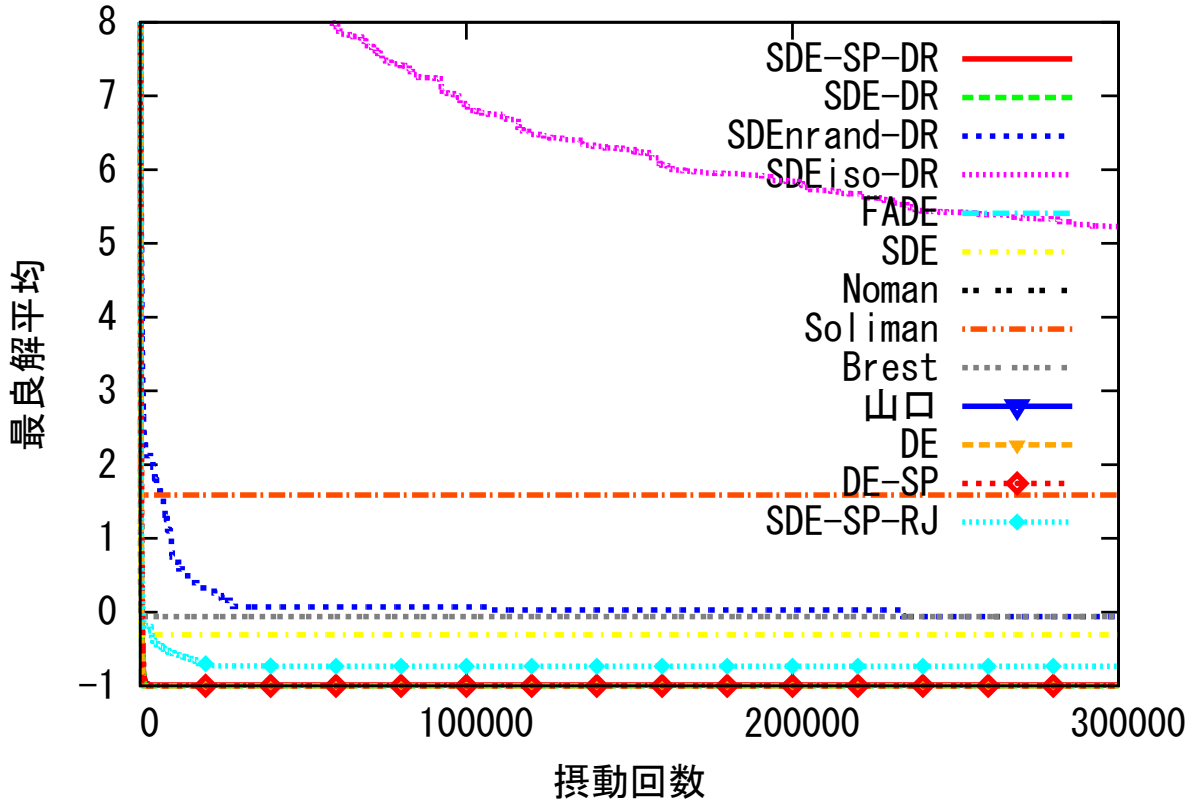


図 4.8: NF2 最適化時の各手法の最良解平均

ベンチマーク関数最適化実験

本節では 2.5.1 節にて述べられている Rastrigin 関数，Rosenbrock 関数，Schwefel 関数，Griwank 関数，Ackley 関数を用いた最適化実験による結果比較を行い，SDE-SP-DR の性能を確認する．まず，最適化の環境は以下の通りである．

次元 2 次元，20 次元，50 次元

摂動回数上限 それぞれの次元に対し 30000 回，90000 回，150000 回

個体数 次元数 ×10

試行回数 2 次元と 20 次元は 100 回，50 次元は 20 回

性能指標 4.5.2 節と同様

最適解発見成否判断基準 4.5.2 節と同様

次に，比較手法は以下の通りである．

- Liu らによる Fuzzy Adaptive Differential Evolution（FADE） [63]
- Orman らによる Self-adaptive Differential Evolution（SDE） [25]
- Noman らによる DE の改良（Noman） [64]
- Soliman らによる DE の改良（Soliman） [65]
- Brest らによる DE の改良（Brest） [66]
- 山口による DE/rand/1/exp の改良（山口） [67]
- DE に本稿での提案手続きを適用した手法（SDE-DR）
- DE/nrand/1 [32] に本稿での提案手続きを適用した手法（SDEnrand-DR）
- DE/isolated/1 [33] に本稿での提案手続きを適用した手法（DEiso-DR）
- $T_{S_{\max}} = 3000$（最大摂動回数の 1%）とした SDE-SP-RJ [61]

これらの条件で最適化実験を行った．

実験結果を表 4.6 から表 4.8 に，摂動回数上限 150000 の最良解平均を図 4.9 から図 4.23 に示す．続いて，各関数毎に結果と考察を述べ，その後に全体の考察を述べる．

Rastrigin 関数最適化結果　まず，表 4.6，表 4.7，表 4.8 から，SDE-SP-DR，Soliman，Brest，SDE-DR が全体的に 90%以上の最適解発見回数割合である．また比較的低次元においては，SDE，SDEnrand-DR，SDE-SP-RJ も比較的良い最適解発見回数割合を示している．ただし，SDE においては 20 次元の 30000 摂動時の結果が 0%であるため，ここから SDE は摂動回数条件の不足による性能低下の程度が大きいことが示唆される．図 4.9，図 4.10，図 4.11 を確認すると，山口，FADE，Noman による探索は停滞しがちであり，他の手法は探索の継続を試みることはできていることが確認される．ただし，表 4.8 より，SDE，SDEiso-DR，SDE-SP-RJ は比較的良い最適解発見率である 4 手法と比較して性能は良くても半分程度で，悪いものでは 0%である．したがって，これら 3 手法を用いて 50 次元の Rastrigin 関数に類似した探索空間を探索する場合，非常に長い摂動回数を設定する必要があると示唆される．

Rosenbrock 関数最適化結果　まず，表 4.6，表 4.7，表 4.8 から，Brest と SDE-DR が比較的良い最適解発見割合を示していおり，低次元では SDE，SDE-DR，SDEnrand-DR，SDE-SP-DR もよい最適解発見回数割合を示す．しかしながら，50 次元の探索空間においてはこの 2 手法においても 15%以下の最適解発見回数割合であり，本実験における手法群にとっては，Rosenbrock 関数の探索は比較的困難であることが示唆される．図 4.12，図 4.13，図 4.14 を確認すると，Brest が他と比較して非常に速やかに探索を行えていることが確認される．ただし，表 4.8 の結果は悪い事から，探索が停滞した状態から抜け出す能力には乏しいことが確認される．他の手法において，Soliman と Noman は若干停滞しがちではあるものの，山口以外の全手法が探索の継続に成功していることが確認される．以上より，これらの手法群に Rosenbrock 関数に類似した探索空間，即ち依存関係の存在しない変数が存在しないような探索空間の探索をさせることは比較的困難であり，他の手法を用いるか，摂動回数上限を十分に取るべきであることが示唆される．

Schwefel 関数最適化結果　まず，表 4.6，表 4.7，表 4.8 から，SDE-SP-DR，Soliman，SDE-DR，SDE-SP-RJ が比較的良い最適解発見回数割合を示すことが確認される．また低次元においては，SDE，SDEnrand-DR も良い最適解発見回数割合を示しており，Soliman はより高次元の時と比較して 2 次元の最適解発見回数割合が若干悪い事が確認される．したがって，Soliman は Schwefel 関数に類似した低次元の探索空間においては，次元数 ×10 より多く個体数を取った方が良い可能性が示唆される．また，後述の Griwank 関数の最適化においてはこの結果をさらに顕著にしたような性能低下が確認されるため，局所解の数がある程度多い問題においては，Soliman の個体数は次元に因らず大きいものにした方が良い可能性も示唆される．図 4.15，図 4.16，図 4.17 を確認すると，山口，Noman 以外の手法は探索の継続に成功していることが確認される．ただし，図 4.15 より Soliman は低次元においては探索が停滞している傾向が見られ，ここからも Soliman は Schwefel 関数に類似した低次元の探索空間においては，次元数 ×10 より多く個体数を取った方が良い可能性が示唆される．また，Noman と Brest は，最適解周辺までは速やかに収束しているが，そこから最適解へ至ることが出来ていないことも，表 4.8 と図 4.17 から確認される．

Griwank 関数最適化結果　まず，表 4.6，表 4.7，表 4.8 から，Soliman と Brest が比較的良い最適解発見回数割合を示すことが確認される．また低次元においては，SDE-SP-DR，SDE，SDEnrand-DR，SDE-SP-RJ も良い最適解発見回数割合を示し，Soliman の 2 次元における最適解発見回数割合が低いことが確認される．この Soliman の性能低下の程度は Schwefel 関数の時のものよりも大きく，Griwank 関数に類似する局所解の多い探索空間においては，Soliman の個体数は次元に関わらず大きいものとした方が良い可能性が示唆される．図 4.18，図 4.19，図 4.20 を確認すると，山口と Noman 以外の手法は探索の継続に成功していることが確認される．Noman においては最適解周辺までは速やかに収束しているものの，そこから最適解への収束に失敗していることが確認される．

Ackley 関数最適化結果　まず，表 4.6，表 4.7，表 4.8 から，Soliman が比較的良い最適解発見回数割合を示すことが確認されるが，50 次元においては全手法において最適解発見回数割合が 0%であり，Ackley 関数は本実験の手法群では比較的探索困難な探索空間であると示唆される．2 次元においては，SDE-SP-DR，SDE，SDE-DR，SDEnrand-DR，SDE-SP-RJ も良い最適解発見回数割合を示すことが確認される．図 4.21，図 4.22，図 4.23 を確認すると，山口，Noman，SDE-DR は探索が停滞していることが確認され，FADE と SDEnrand-DR も探索の継続は行われているものの進行速度は比較的遅いことが確認される．それ以外の手法においても，最適解周辺までの探索には成功しているようであっても，表 4.8 から確認されるように，そこからは探索が停滞しているであろうことが示唆される．

ベンチマーク最適化実験のまとめ　まず最適解発見回数割合の結果から，2 次元の Rosenbrock 関数を除いて，提案手法 SDE-SP-DR は従来の SDE-SP-RJ の上位互換となっていることが確認される．また最良解平均推移を確認しても，SDE-SP-DR は SDE-SP-RJ を下回るように推移しており，この点からも SDE-SP-DR が SDE-SP-RJ よりも有効な手法であることがより強く示唆される．手法全体を俯瞰すると，Soliman，Brest が高い性能を示し，次点として SDE-SP-DR，SDE-DR，SDE の性能が高いと捉えられるだろう．ただし，Soliman，SDE については探索の継続力が比較的劣っている場合もあり，何の考慮もなく運用することは容易ではないことが示唆される．

最良解平均推移の結果から SDE-SP-DR の様子を観察すると，最適解を発見できなかった場合でも，グラフは基本的に右下がりであり，探索の継続は行われていることが示唆される．従って，SDE-SP-DR を有効に運用するためには，非常に多くの摂動回数が必要であることが示唆されることに，実運用上は留意する必要があるだろう．

表 4.6: 2 次元の各ベンチマーク関数最適化実験での最適解発見回数割合 [%]

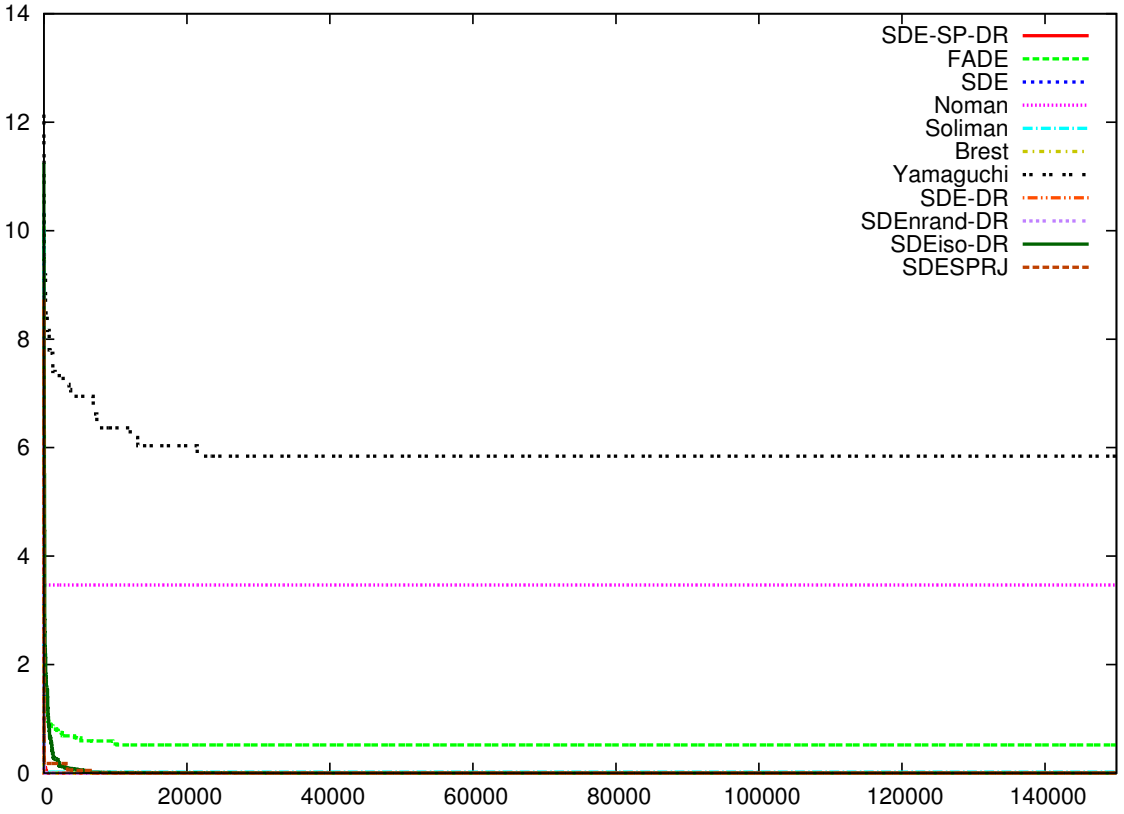
ベンチマーク関数	Rastrigin Function			Rosenbrock Function			Schwefel Function			Griwank Function			Ackley Function		
摂動回数	30000	90000	150000	30000	90000	150000	30000	90000	150000	30000	90000	150000	30000	90000	150000
SDE-SP-RJ	100	100	100	42	61	70	100	100	100	88	89	90	100	100	100
FADE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SDE	99	100	100	96	95	85	99	100	100	76	76	85	100	100	100
Noman	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Soliman	80	75	85	10	17	5	80	80	85	28	30	35	83	78	85
Brest	98	99	100	98	100	100	0	0	0	82	81	90	0	0	0
山口	1	1	5	0	1	0	0	0	0	1	1	5	1	1	5
SDE-DR	100	100	100	100	99	100	100	100	100	92	89	95	99	100	100
SDEnrand-DR	100	100	100	98	100	100	100	100	100	88	87	90	100	100	100
SDEiso-DR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SDE-SP-RJ	100	100	100	65	70	85	100	100	100	88	88	90	100	100	100

表 4.7: 20 次元の各ベンチマーク関数最適化実験での最適解発見回数割合 [%]

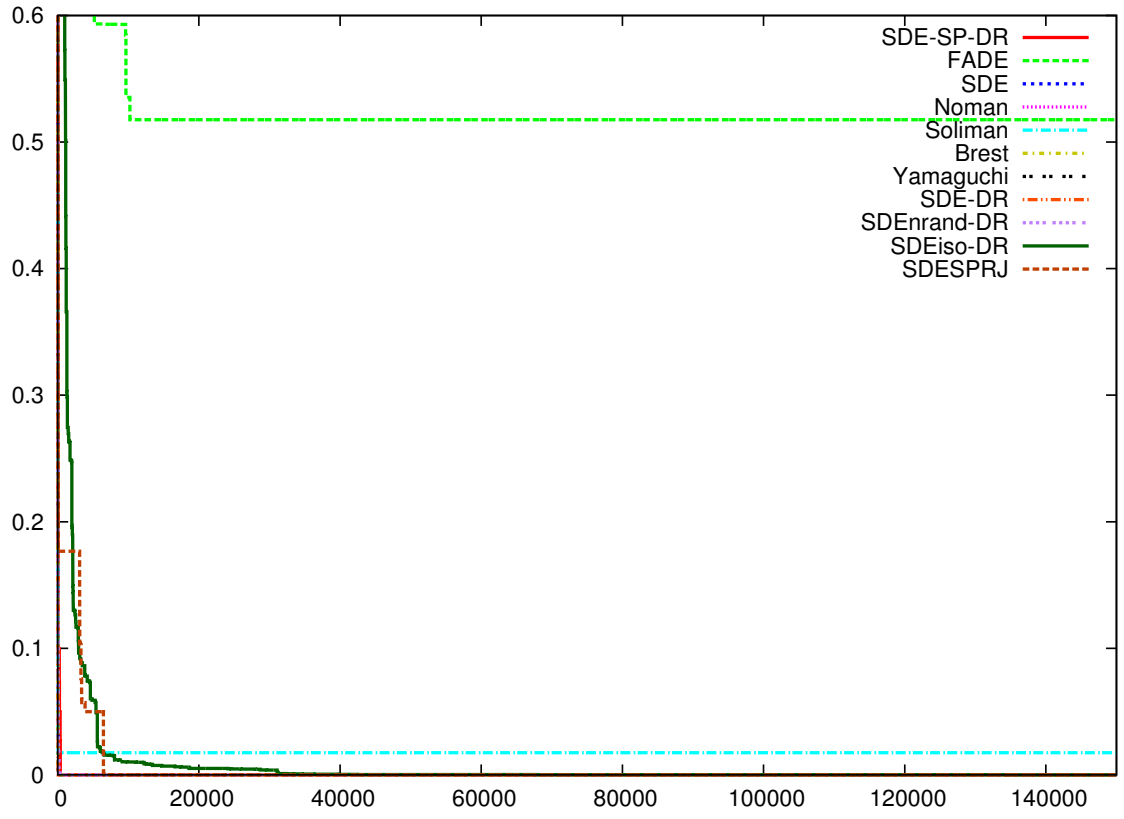
ベンチマーク関数	Rastrigin Function			Rosenbrock Function			Schwefel Function			Griwank Function			Ackley Function		
摂動回数	30000	90000	150000	30000	90000	150000	30000	90000	150000	30000	90000	150000	30000	90000	150000
SDE-SP-RJ	100	100	100	0	0	0	100	100	100	99	99	100	1	1	5
FADE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SDE	0	100	100	0	0	0	100	100	100	100	100	100	0	3	5
Noman	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Soliman	100	100	100	0	0	0	100	100	100	100	100	100	36	60	80
Brest	100	100	100	94	97	95	0	0	0	100	100	100	0	0	0
山口	1	1	5	0	0	0	0	0	0	0	0	0	0	0	0
SDE-DR	100	100	95	31	45	50	99	100	100	95	97	90	2	0	0
SDEnrand-DR	34	62	65	0	6	0	20	47	30	27	39	30	0	0	0
SDEiso-DR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SDE-SP-RJ	80	95	95	0	0	0	100	100	100	61	80	90	0	0	0

表 4.8: 50 次元の各ベンチマーク関数最適化実験での最適解発見回数割合 [%]

ベンチマーク関数	Rastrigin Function			Rosenbrock Function			Schwefel Function			Griwank Function			Ackley Function		
摂動回数	30000	90000	150000	30000	90000	150000	30000	90000	150000	30000	90000	150000	30000	90000	150000
SDE-SP-RJ	97	97	95	0	0	0	100	97	100	11	21	25	0	0	0
FADE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SDE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Noman	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Soliman	100	100	100	0	0	0	100	100	100	94	100	100	0	0	0
Brest	100	100	100	3	2	5	0	0	0	99	100	100	0	0	0
山口	1	1	5	0	0	0	0	0	0	0	0	0	0	0	0
SDE-DR	99	100	100	4	14	10	99	99	95	8	19	35	0	0	0
SDEnrand-DR	10	34	35	0	0	0	3	10	20	0	0	0	0	0	0
SDEiso-DR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SDE-SP-RJ	15	46	50	0	0	0	75	75	80	0	0	10	0	0	0

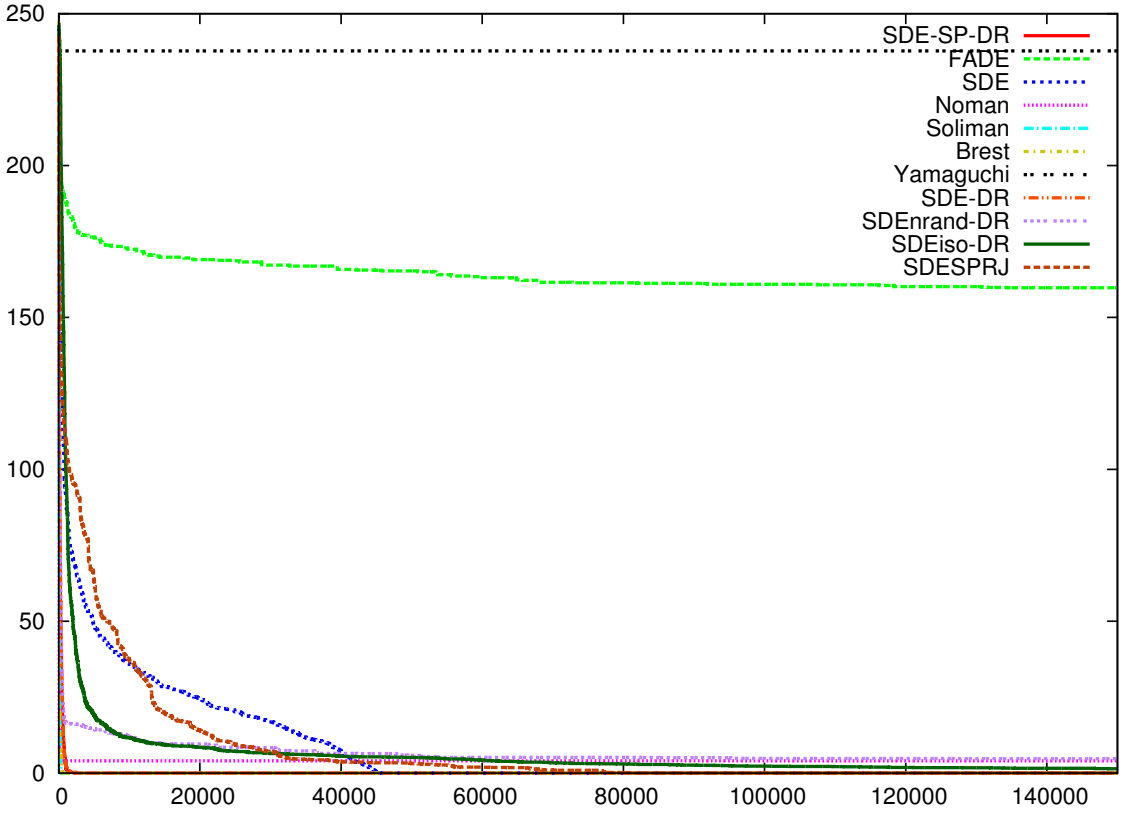


(a) 全体図

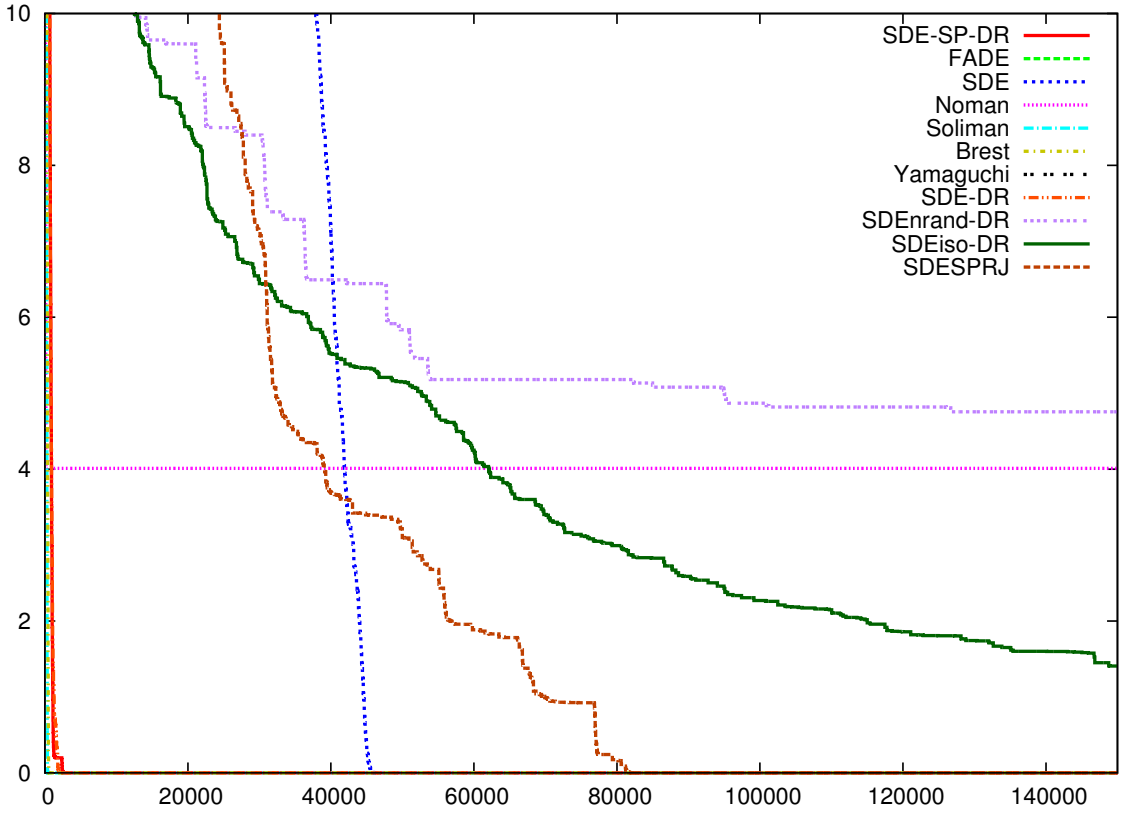


(b) 密集域拡大図

図 4.9: 2 次元の Rastrigin 関数における各手法の最良解平均

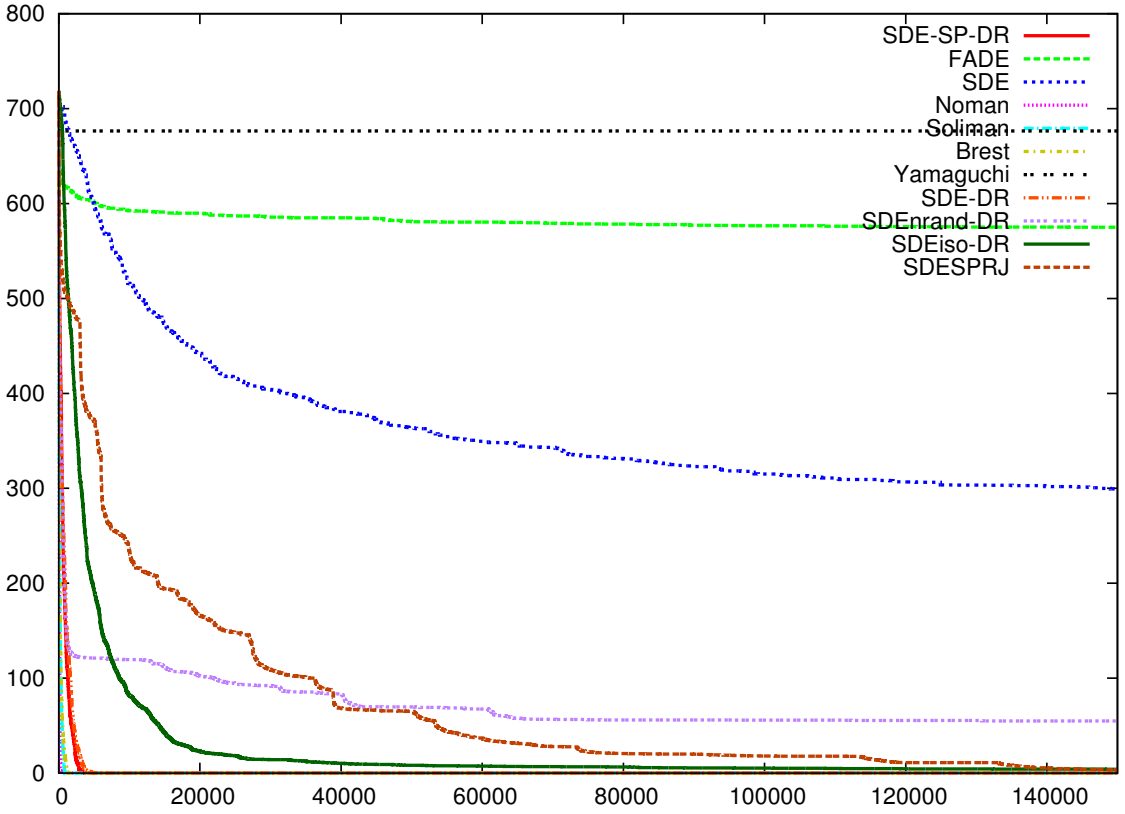


(a) 全体図

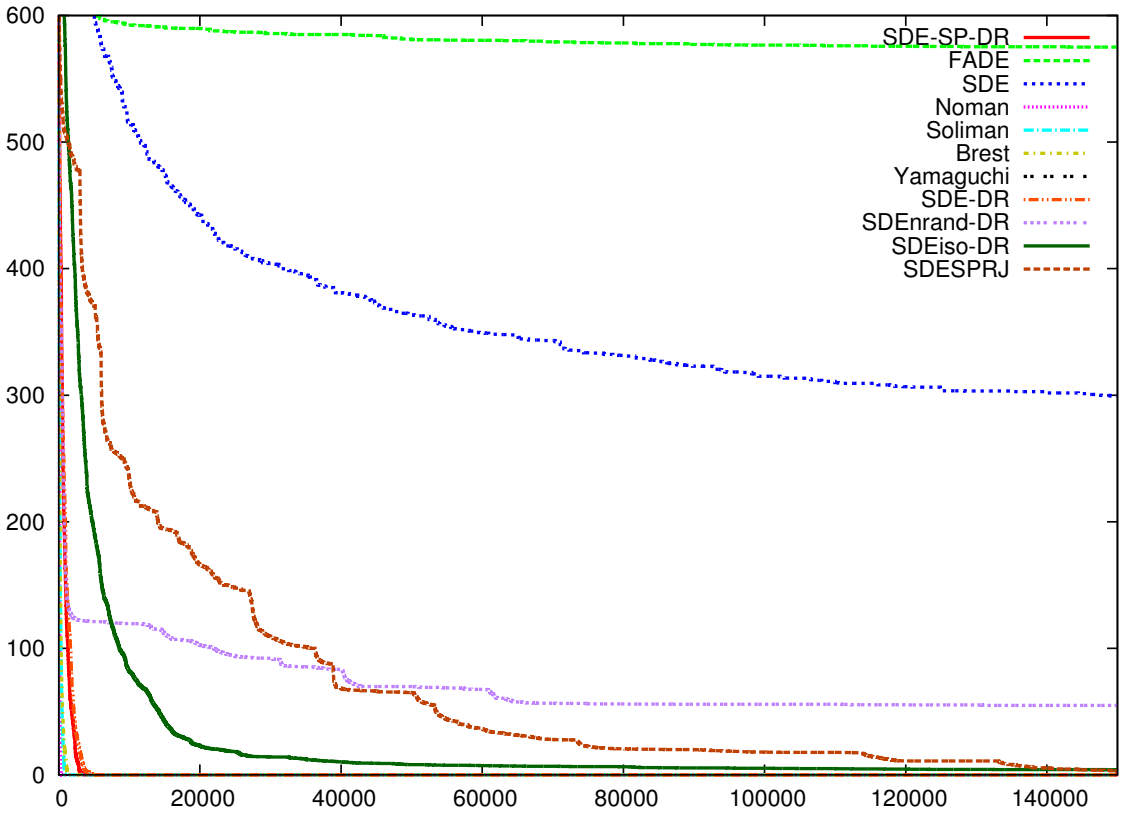


(b) 密集域拡大図

図 4.10: 20 次元の Rastrigin 関数における各手法の最良解平均

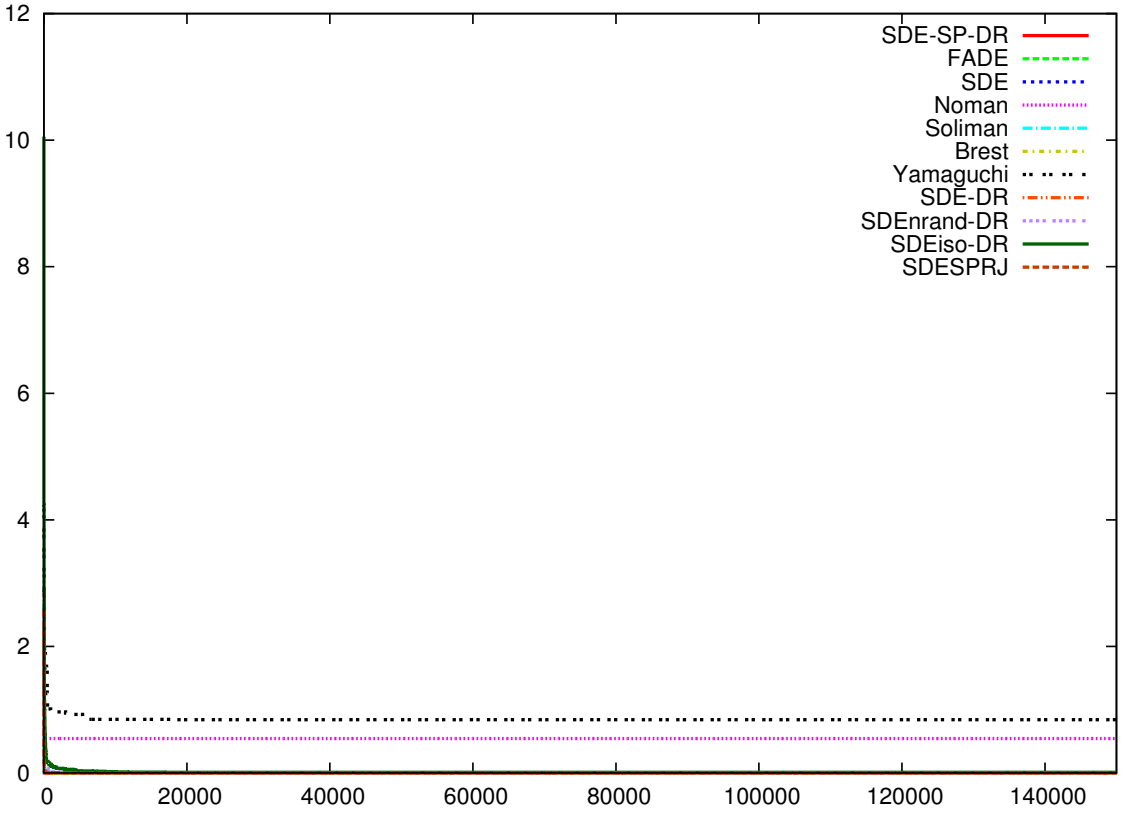


(a) 全体図

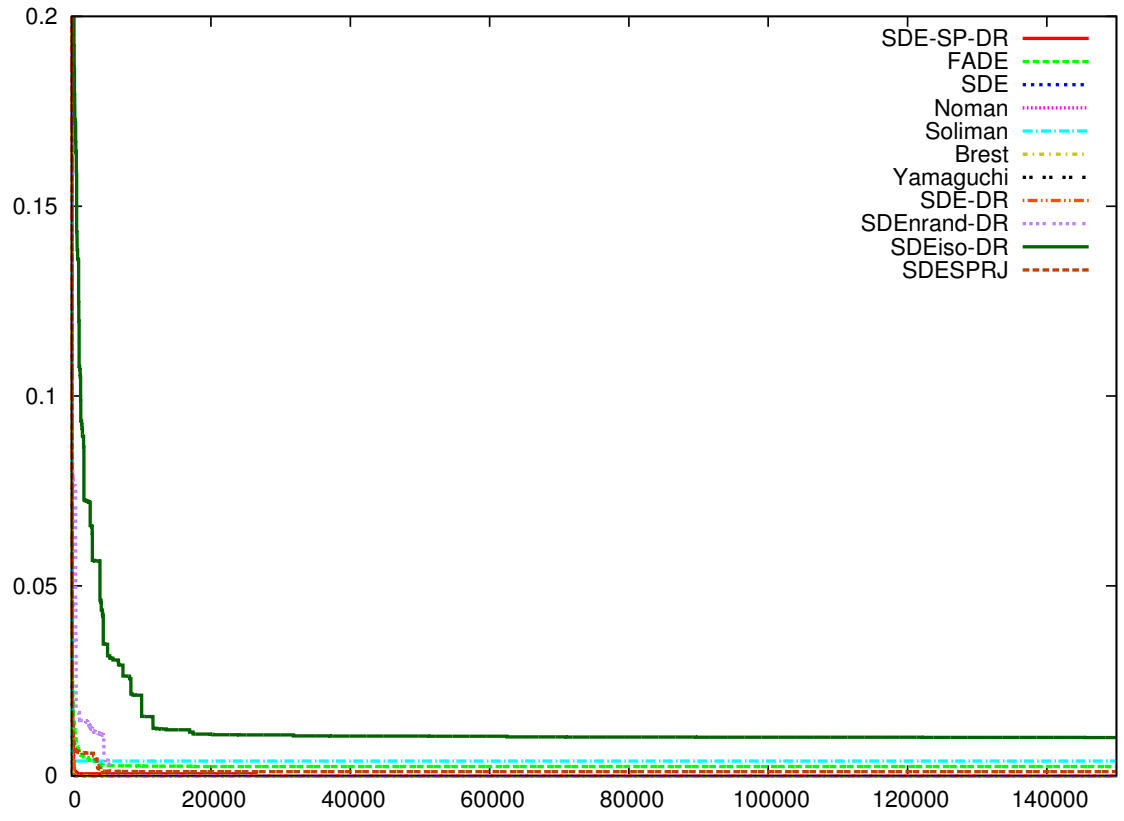


(b) 密集域拡大図

図 4.11: 50 次元の Rastrigin 関数における各手法の最良解平均

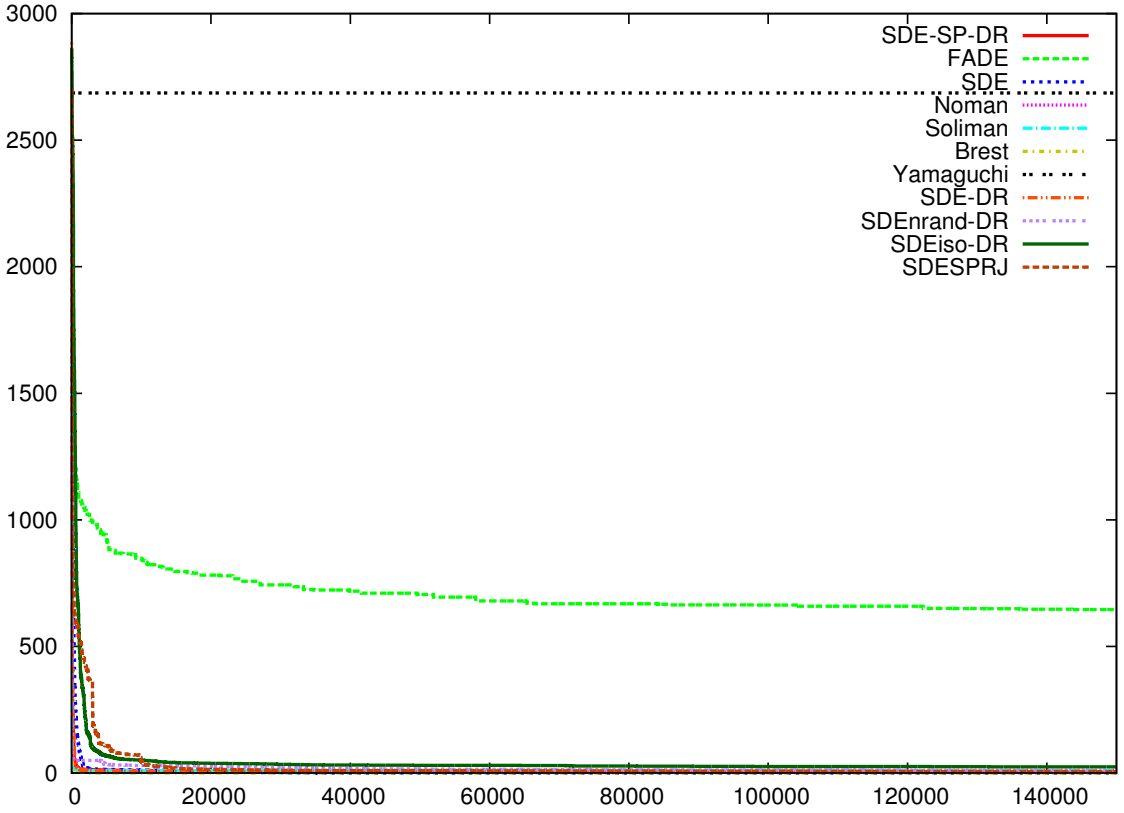


(a) 全体図

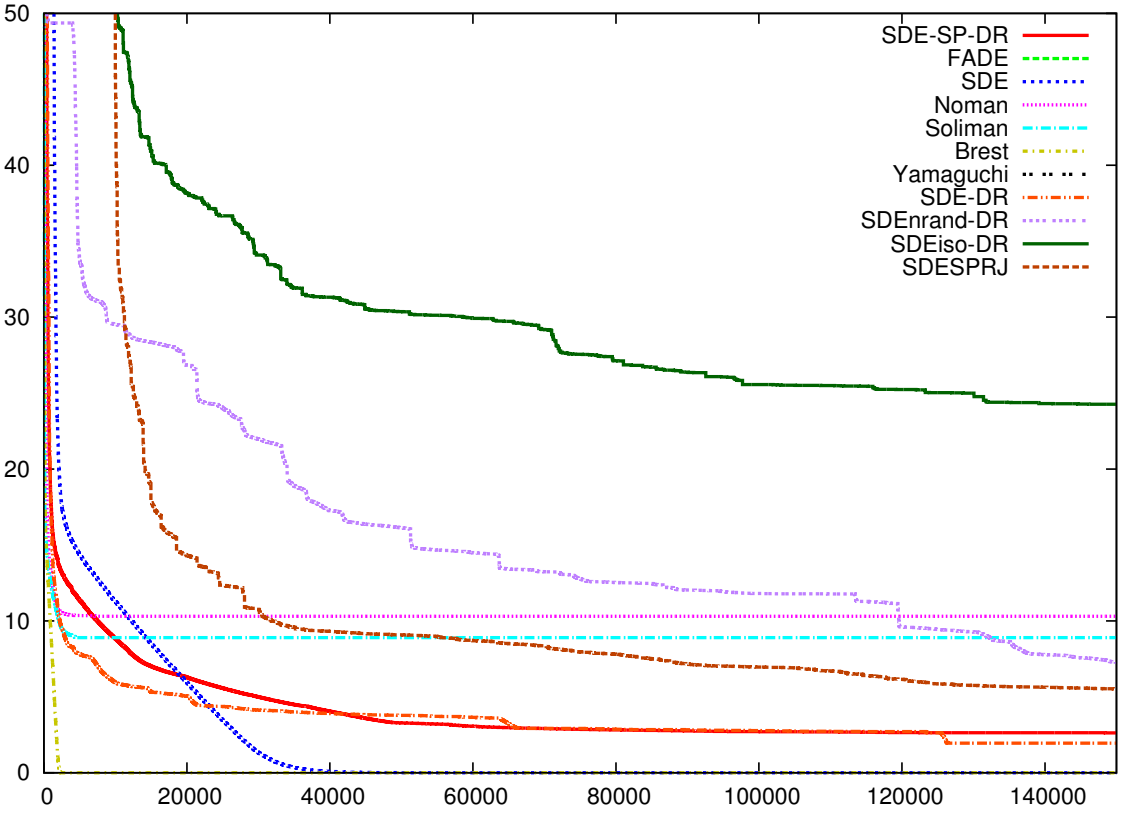


(b) 密集域拡大図

図 4.12: 2 次元の Rosenbrock 関数における各手法の最良解平均

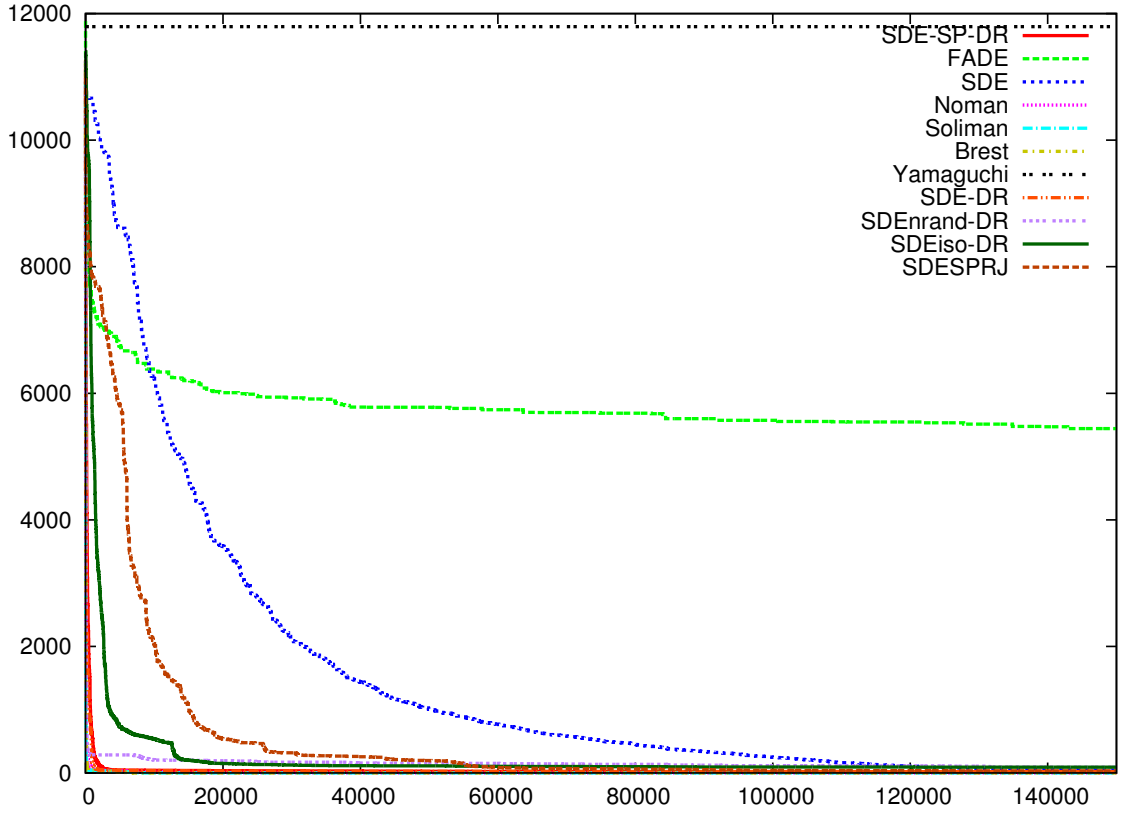


(a) 全体図

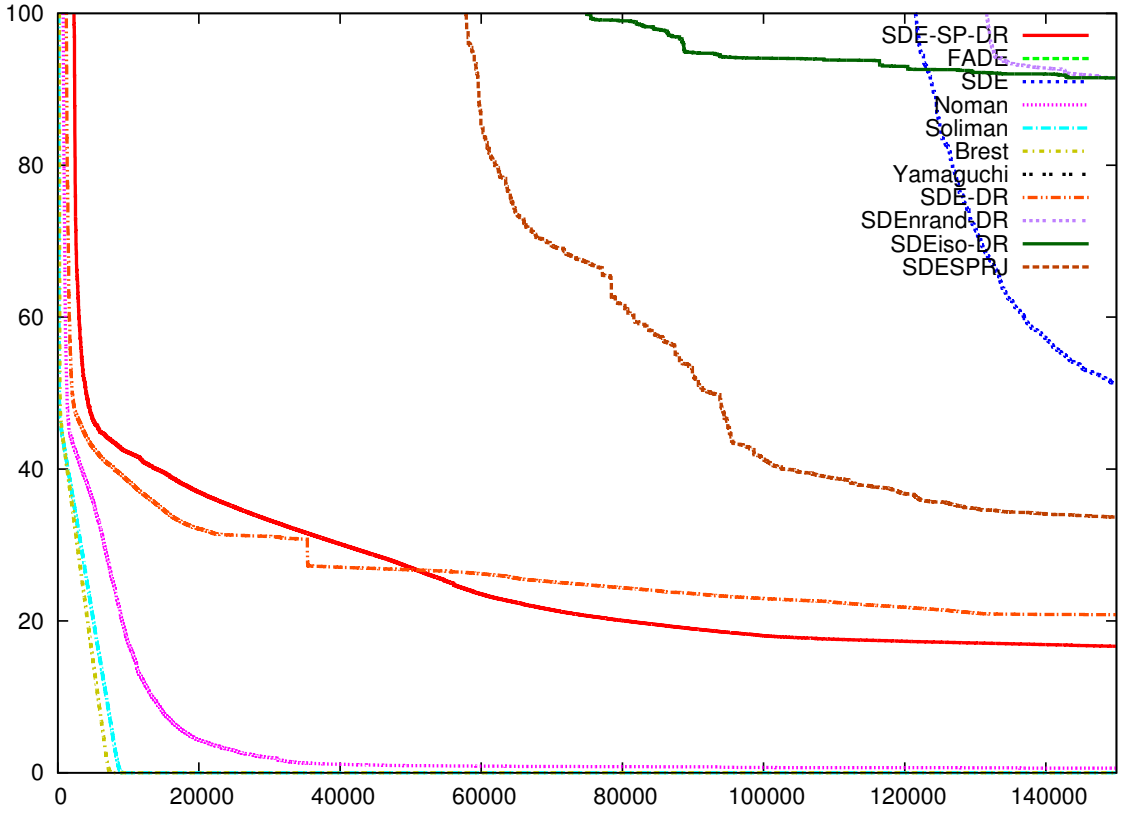


(b) 密集域拡大図

図 4.13: 20 次元の Rosenbrock 関数における各手法の最良解平均

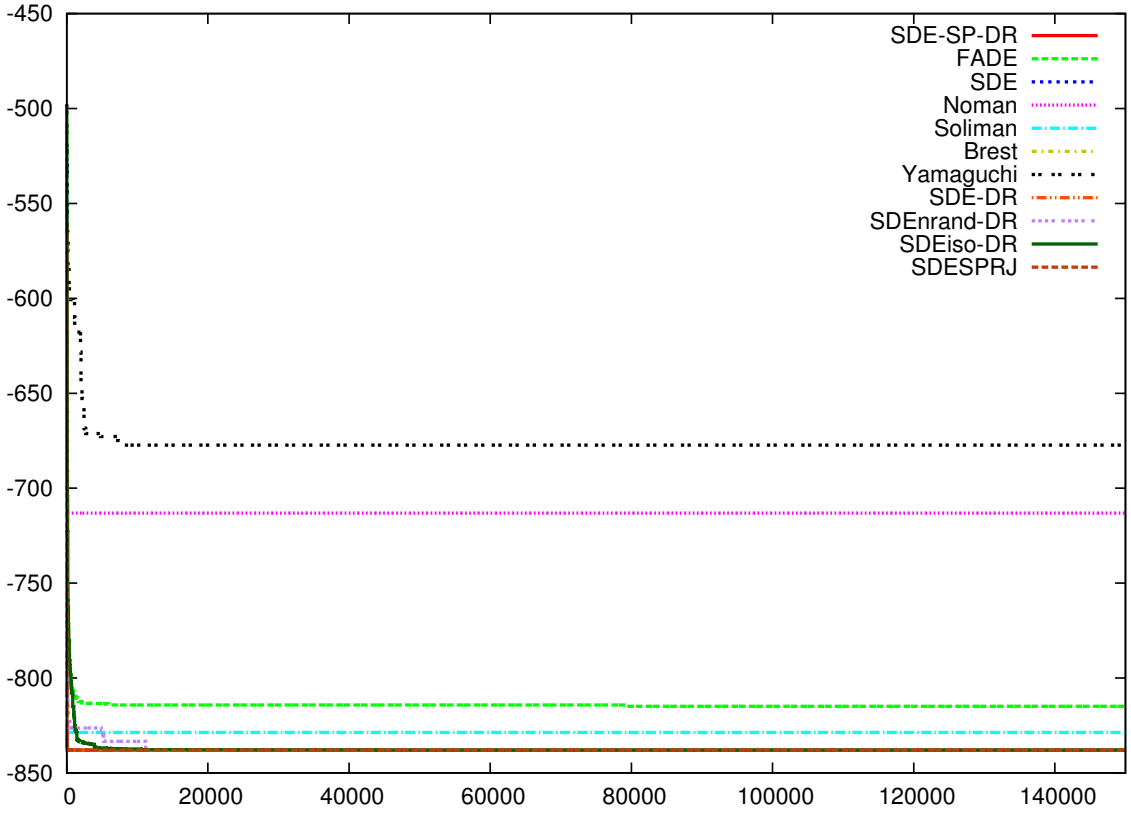


(a) 全体図

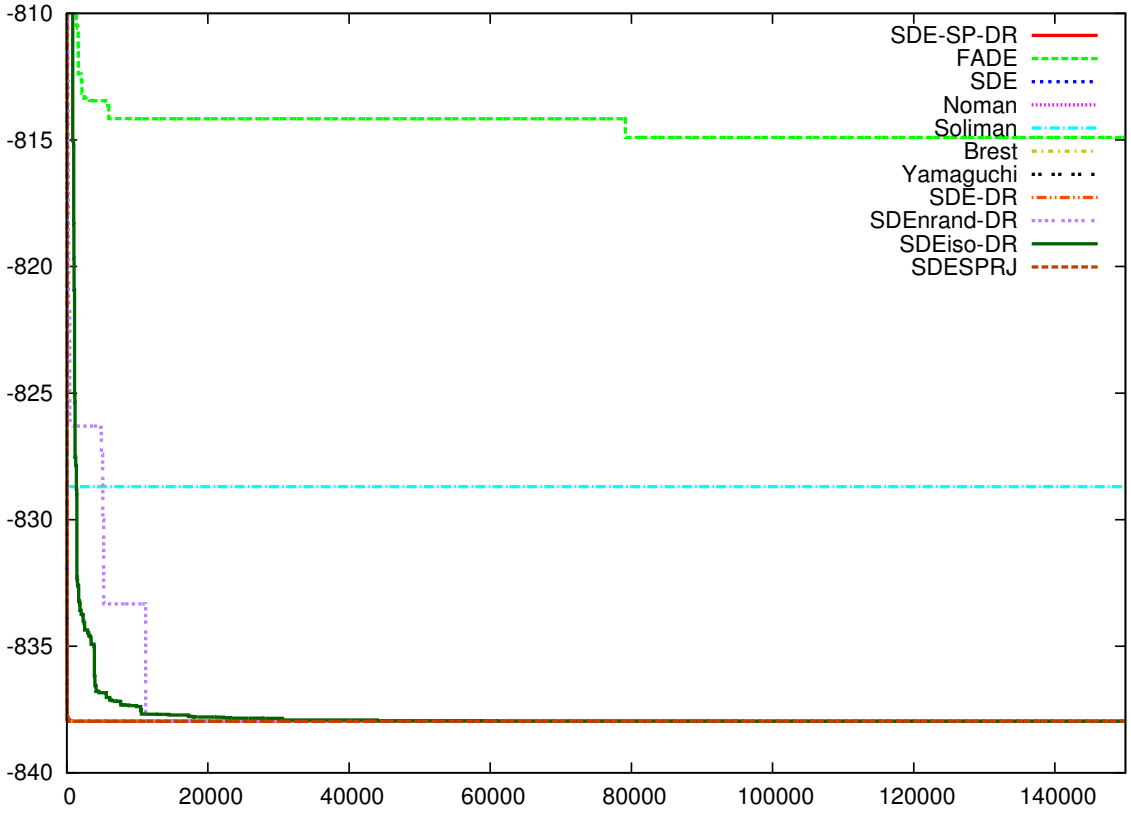


(b) 密集域拡大図

図 4.14: 50 次元の Rosenbrock 関数における各手法の最良解平均

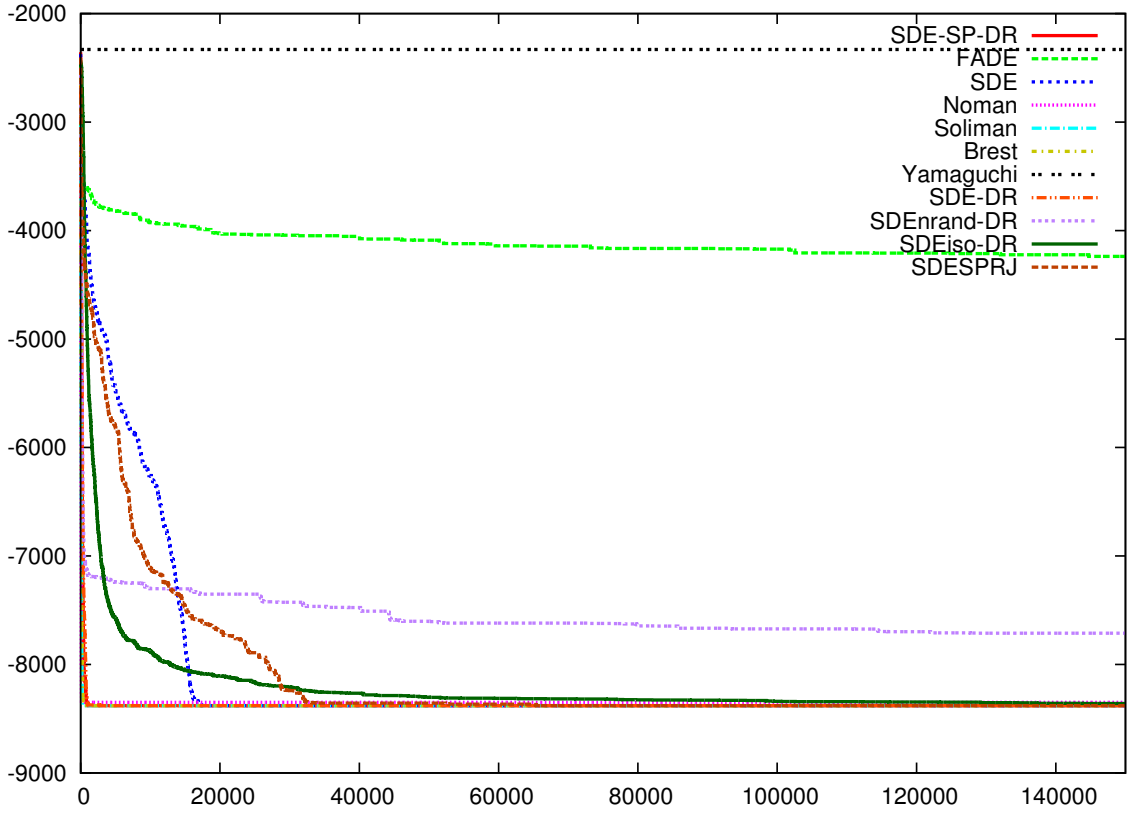


(a) 全体図

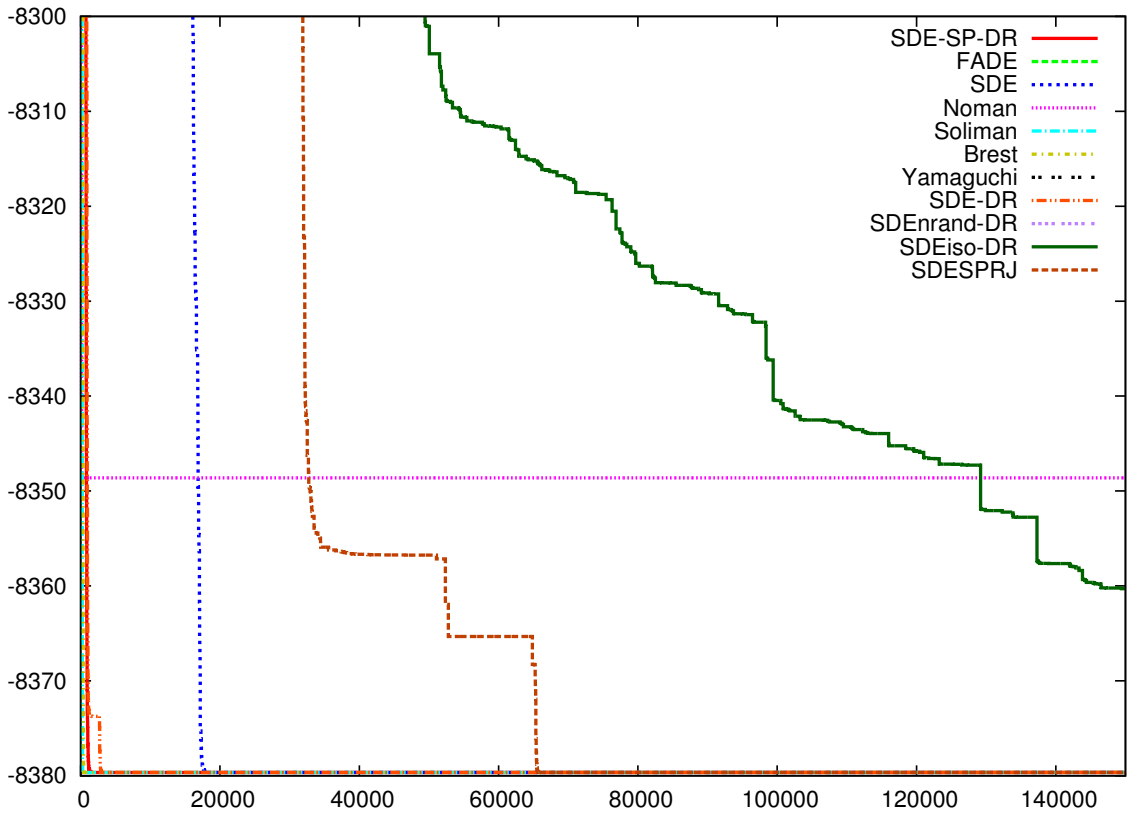


(b) 密集域拡大図

図 4.15: 2 次元の Schwefel 関数における各手法の最良解平均

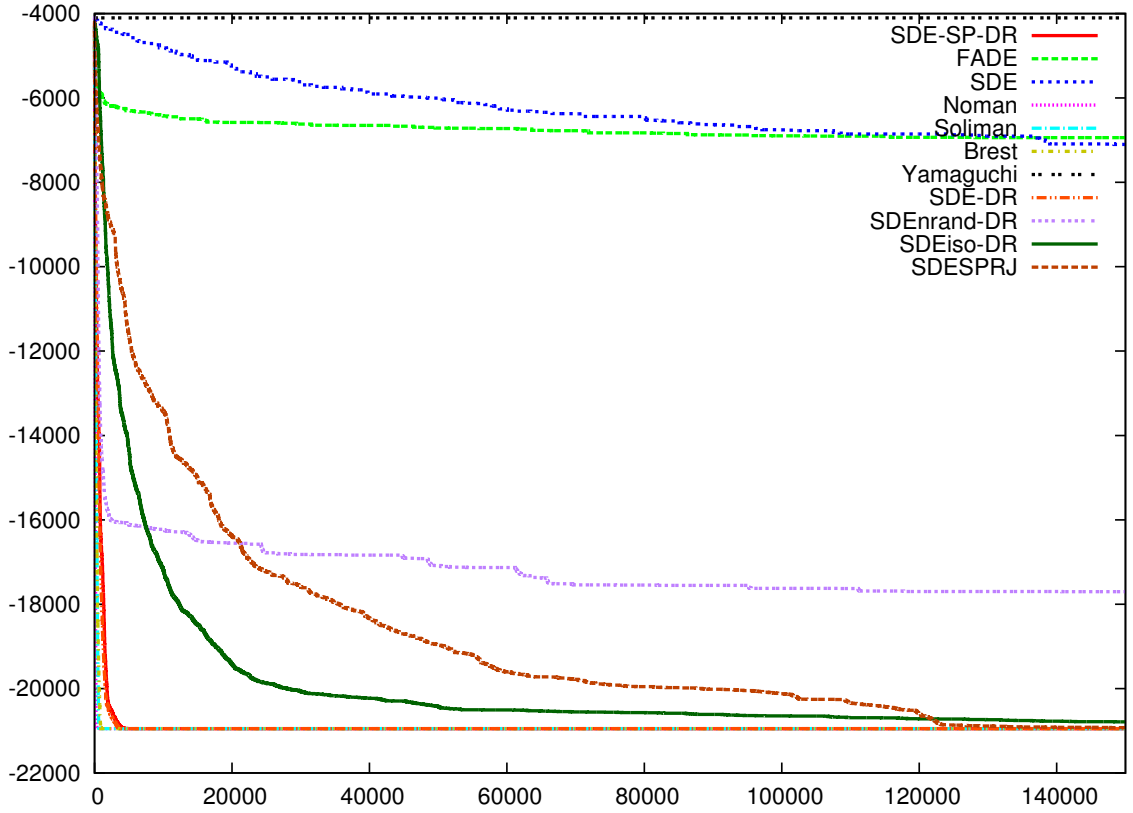


(a) 全体図

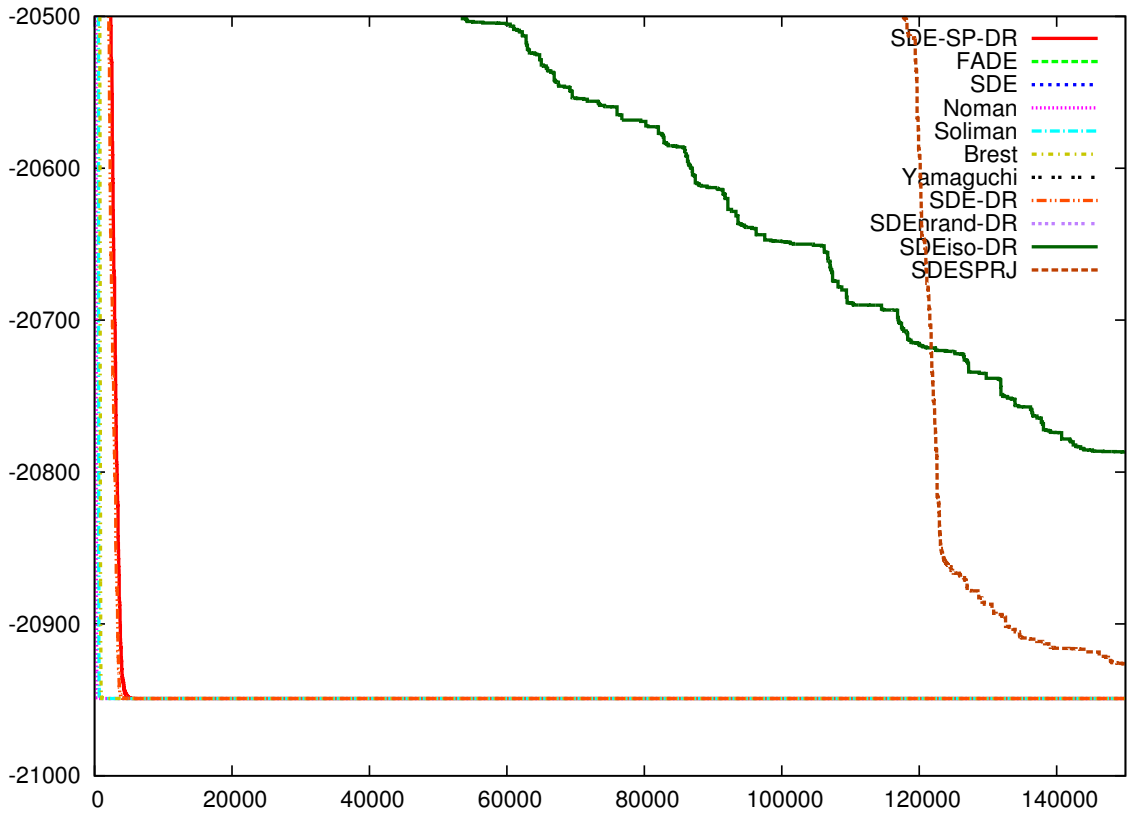


(b) 密集域拡大図

図 4.16: 20 次元の Schwefel 関数における各手法の最良解平均

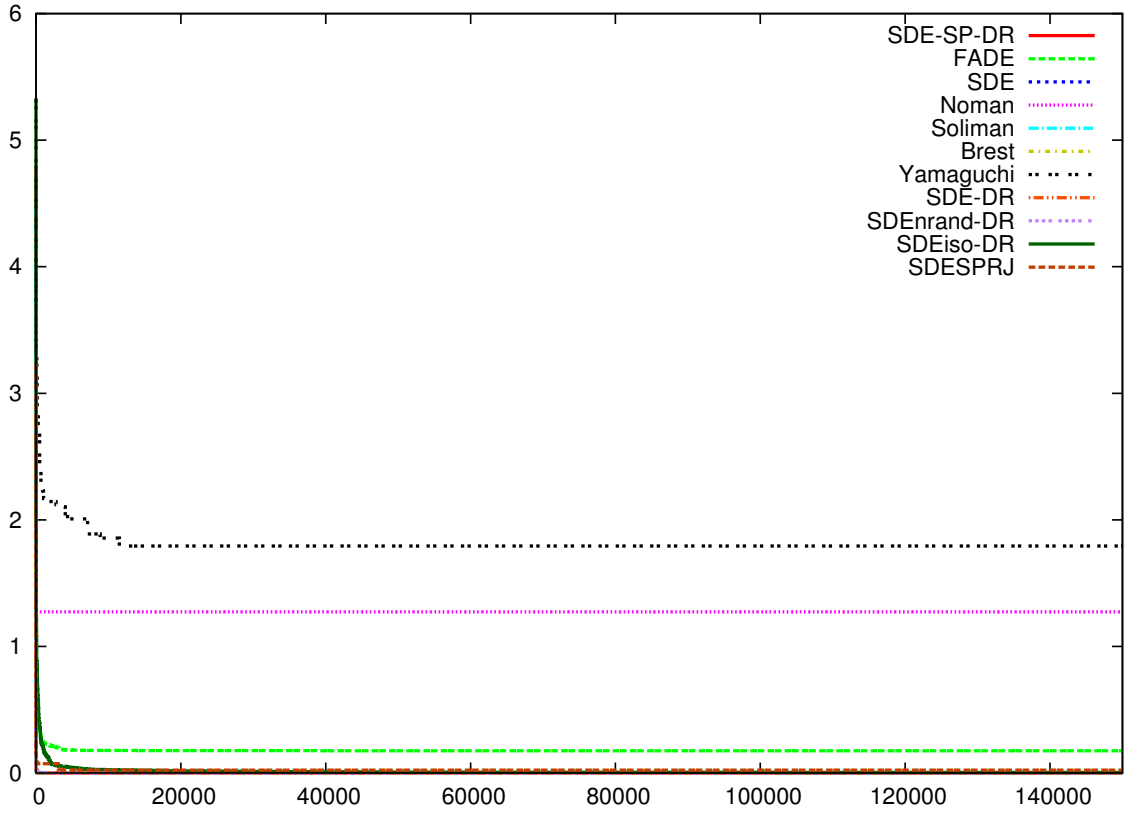


(a) 全体図

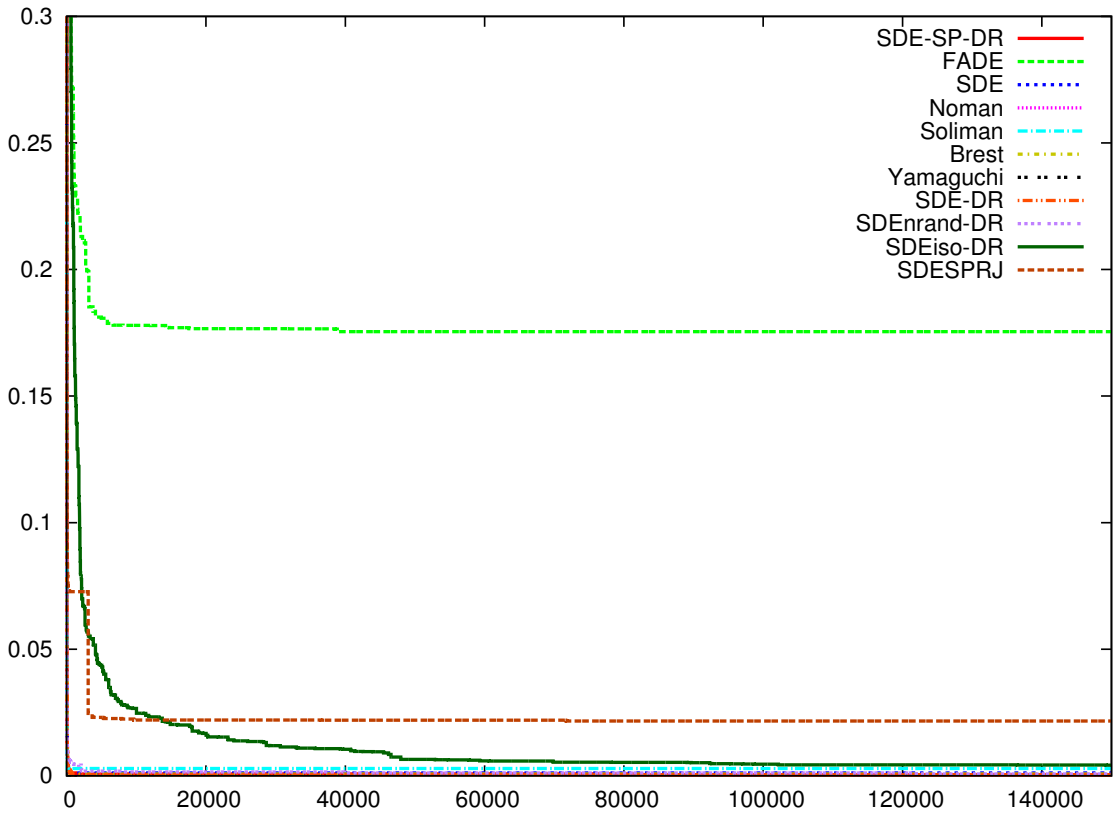


(b) 密集域拡大図

図 4.17: 50 次元の Schwefel 関数における各手法の最良解平均

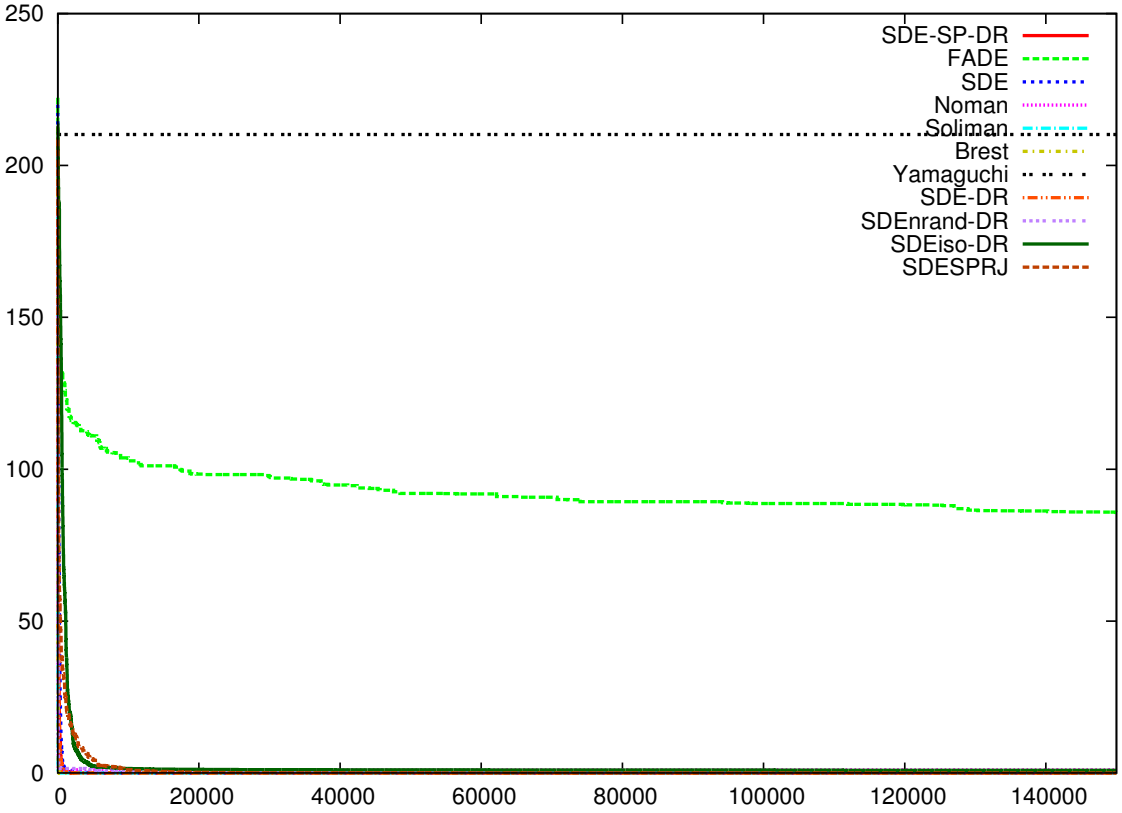


(a) 全体図

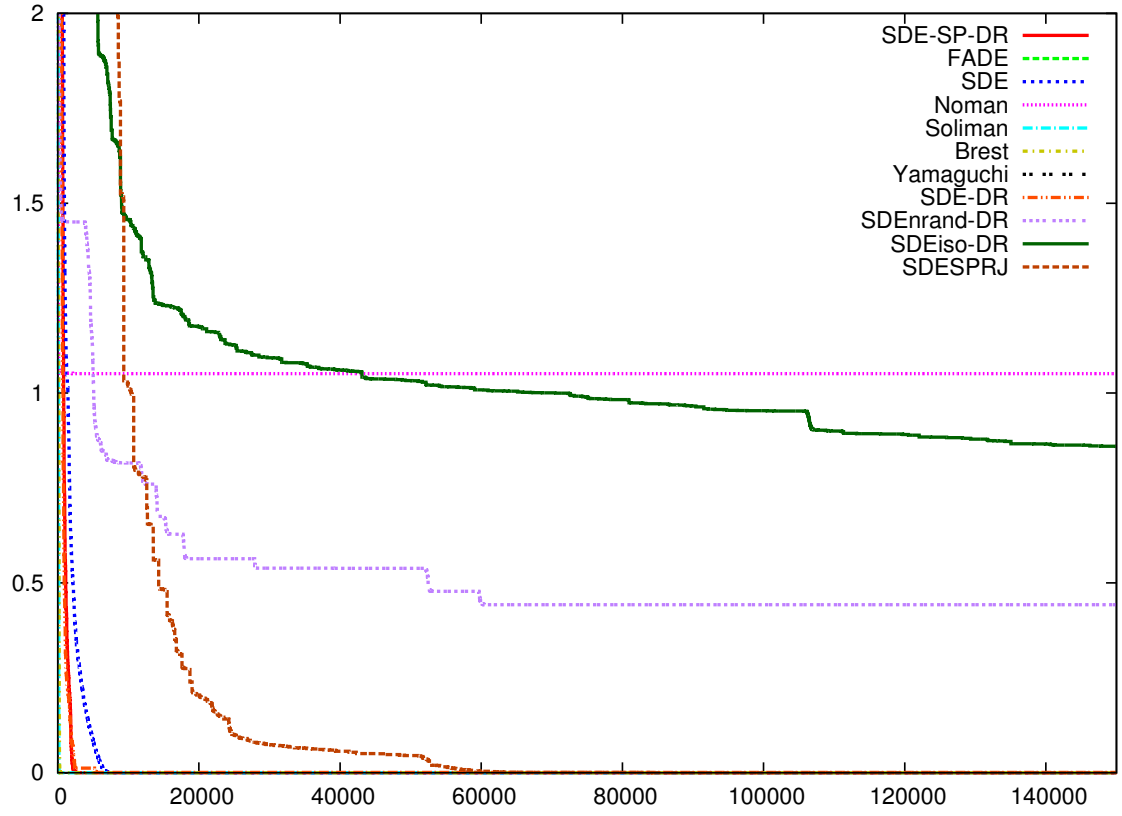


(b) 密集域拡大図

図 4.18: 2 次元の Griwank 関数における各手法の最良解平均

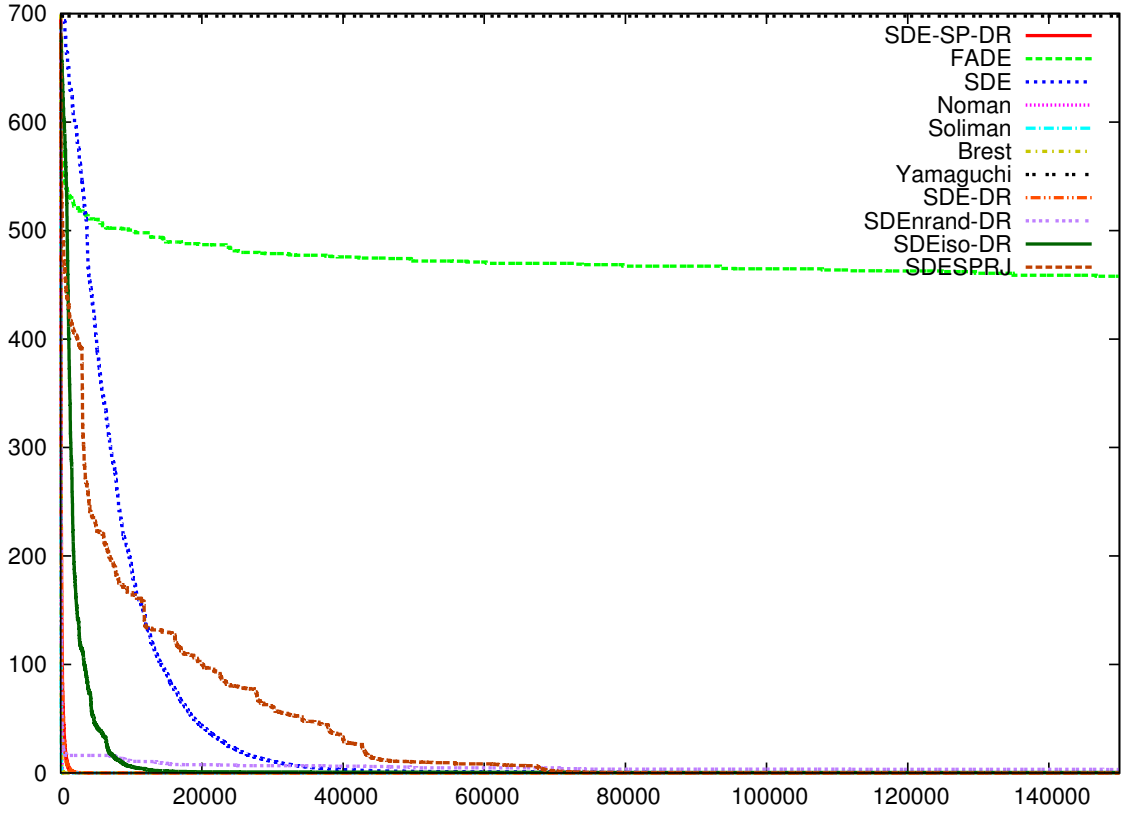


(a) 全体図

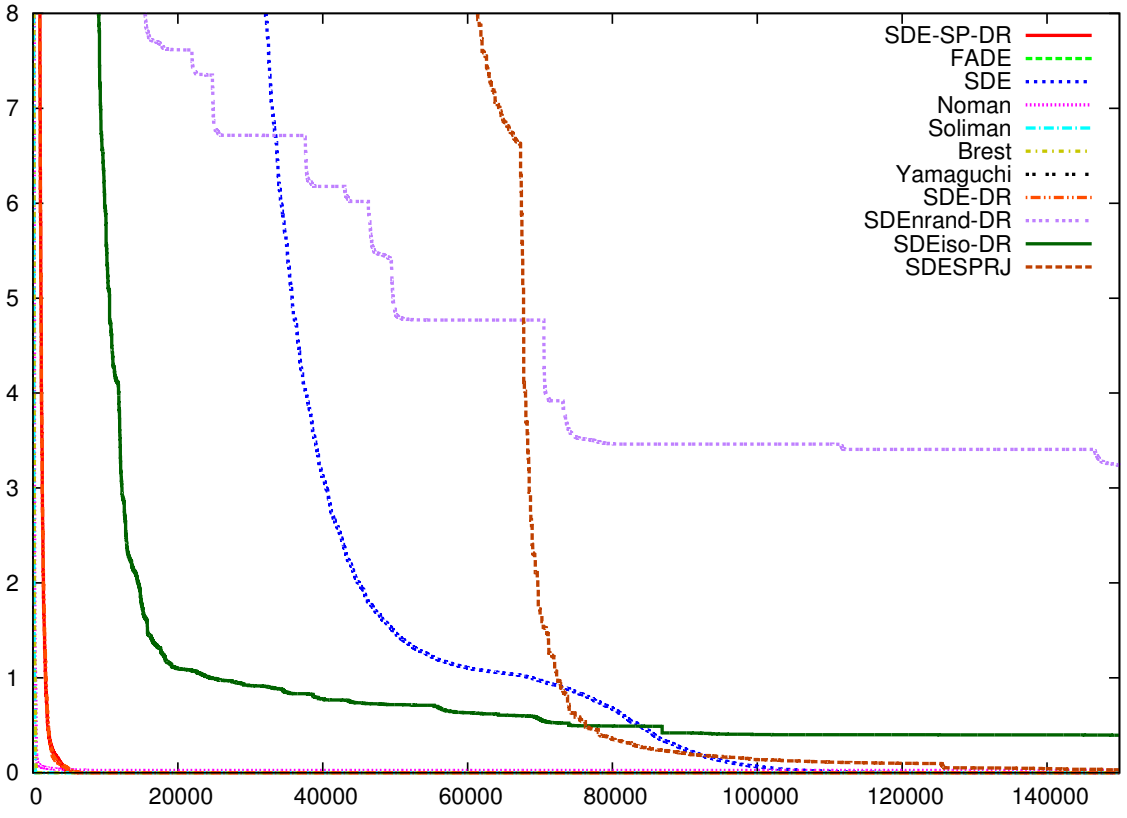


(b) 密集域拡大図

図 4.19: 20 次元の Griwank 関数における各手法の最良解平均

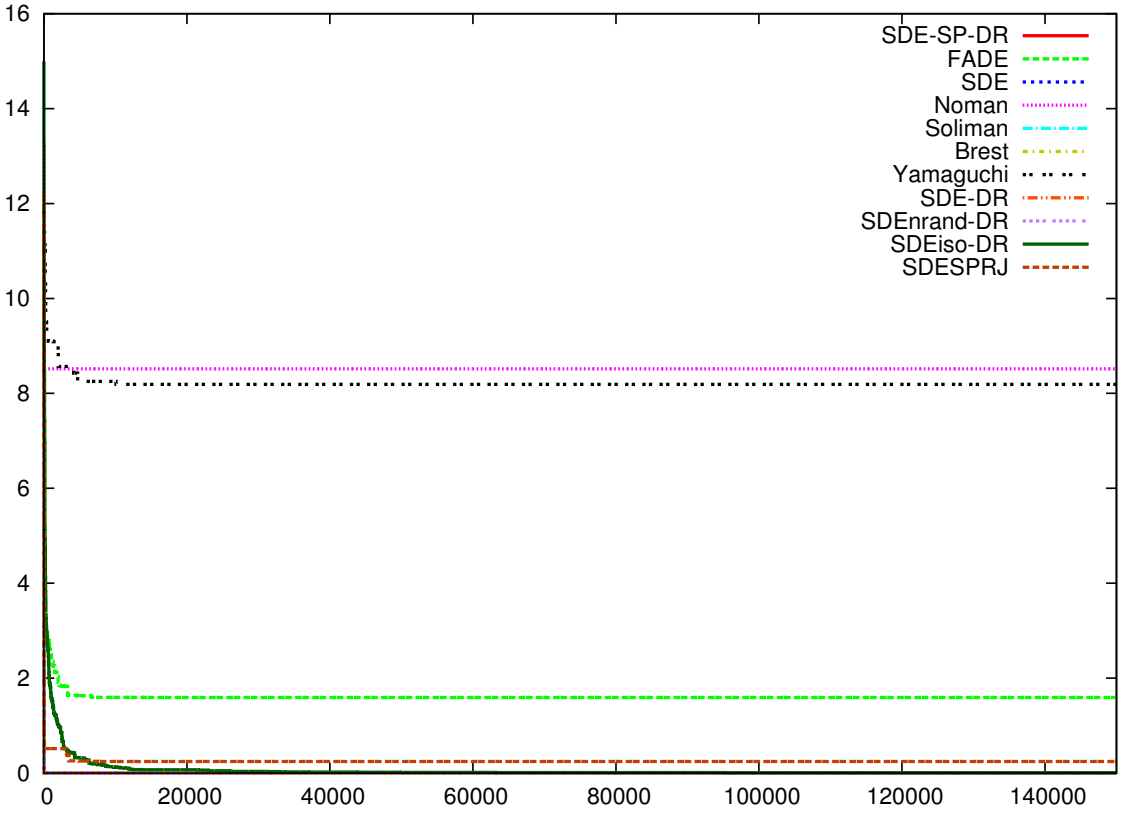


(a) 全体図

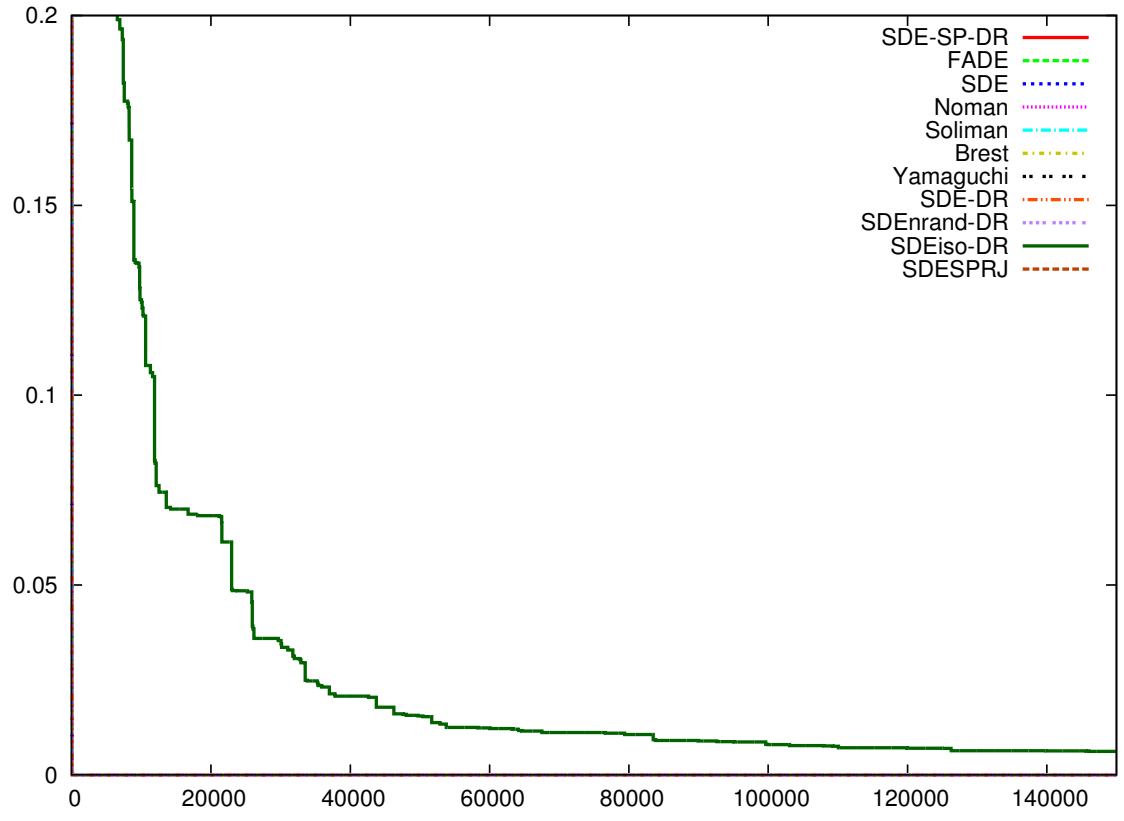


(b) 密集域拡大図

図 4.20: 50 次元の Griwank 関数における各手法の最良解平均

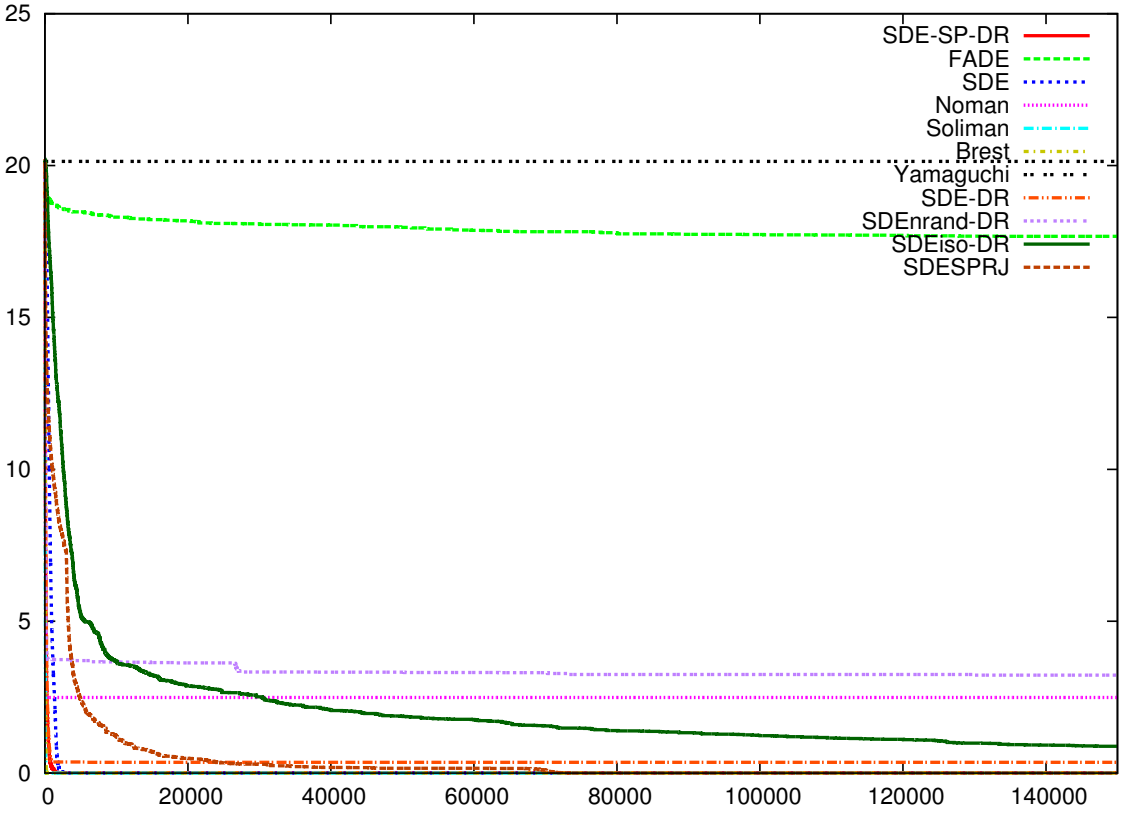


(a) 全体図

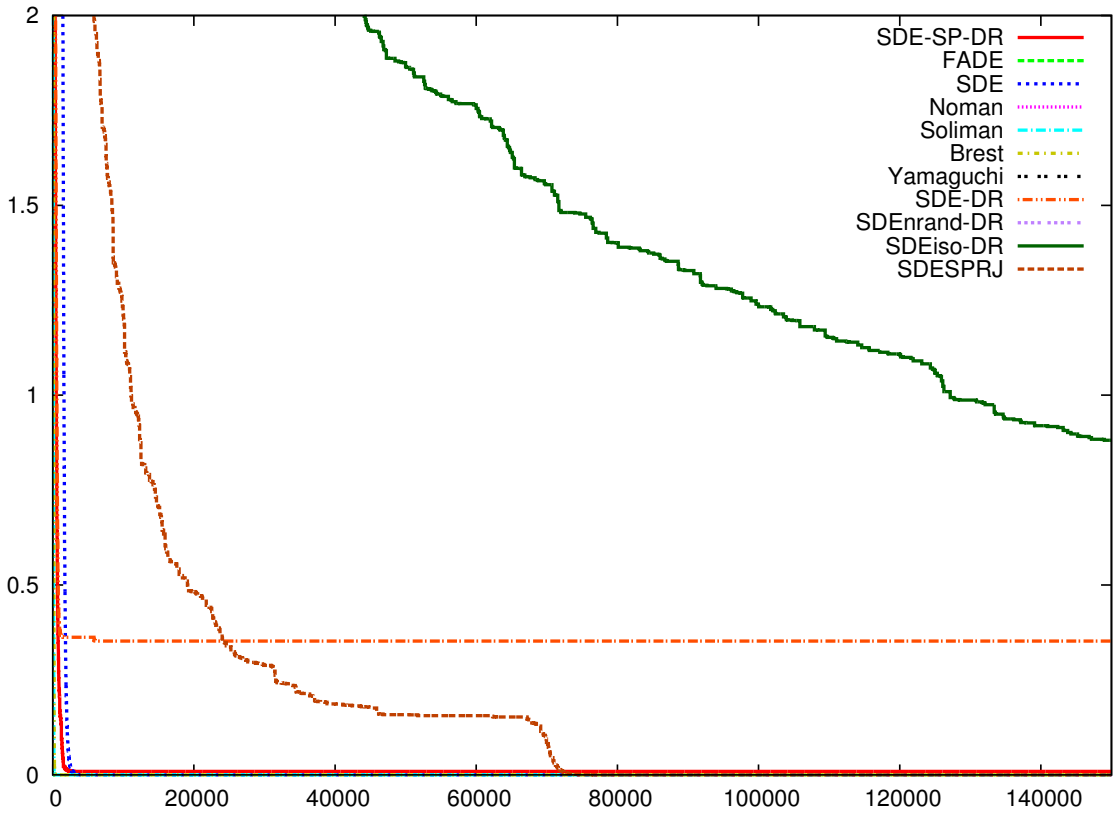


(b) 密集域拡大図

図 4.21: 2 次元の Ackley 関数における各手法の最良解平均

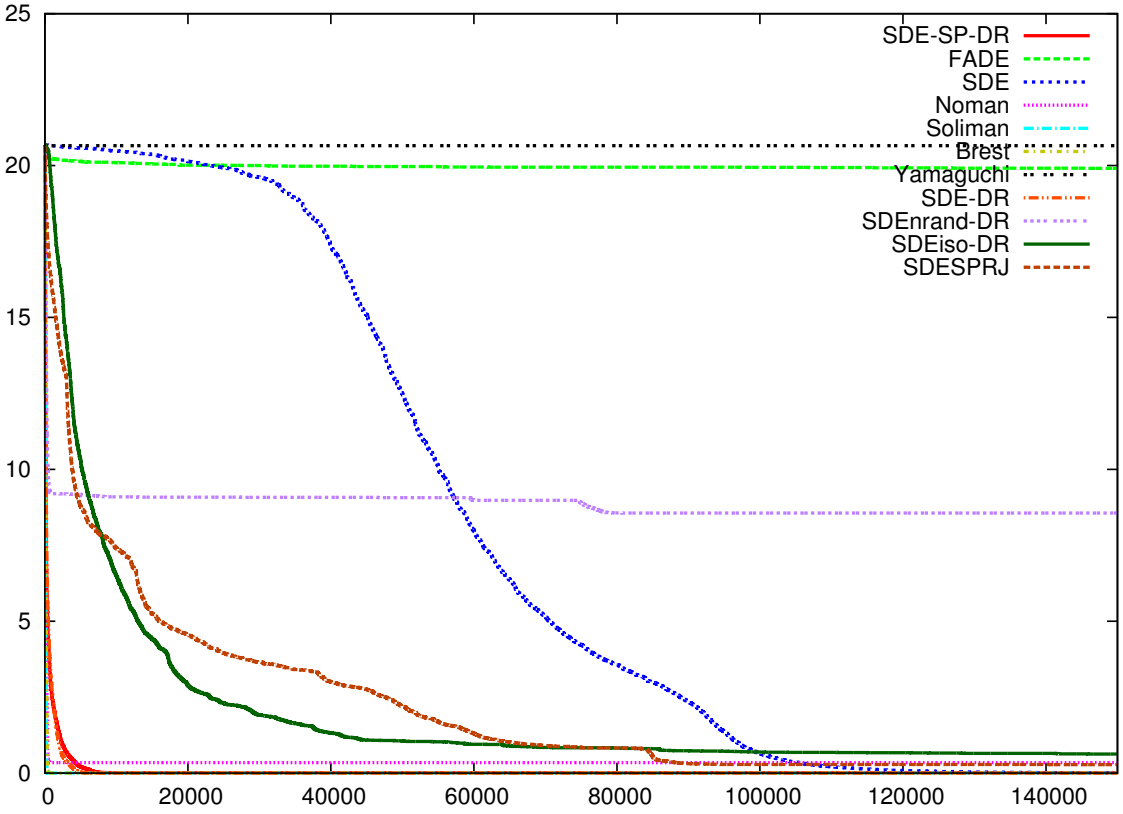


(a) 全体図

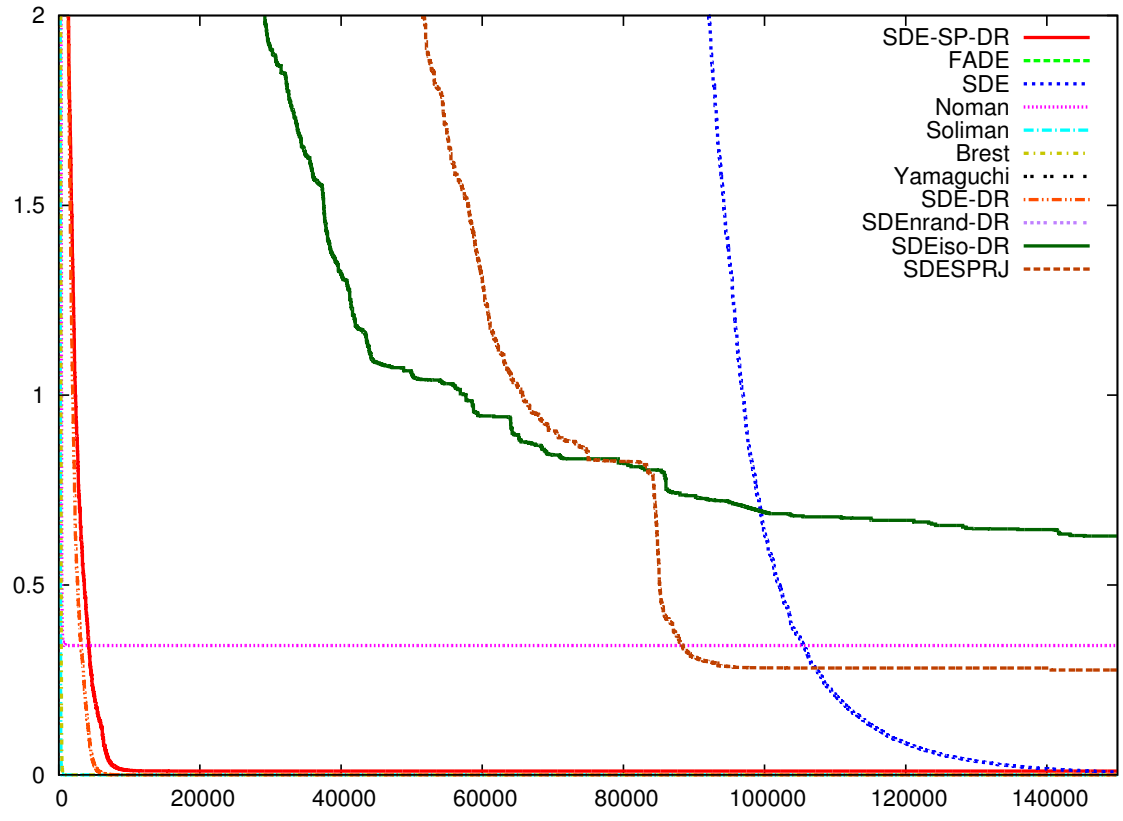


(b) 密集域拡大図

図 4.22: 20 次元の Ackley 関数における各手法の最良解平均



(a) 全体図



(b) 密集域拡大図

図 4.23: 50 次元の Ackley 関数における各手法の最良解平均

Lennard-Jones Clusters Optimization Problem による実験

本節では LJ-Problem を用いた最適化実験による結果比較を行い，SDE-SP-DR の性能を確認する．まず，最適化の環境は以下の通りである．

次元 30 次元（10 粒子）

摂動回数上限 3000000 回

個体数 300

試行回数 10 回

性能指標 全試行を俯瞰して最良の最良解，最悪の最良解，平均の最良解

本問題は報告解は存在するものの，最適解が発見されている訳ではないため，発見される最良解を指標とする．次に，比較手法を以下に示す．本問題は計算時間が莫大に掛かる問題であるため，比較手法は節，節にて高い性能を上げた四つの手法に絞った．

- DE に本稿での提案手続きを適用した手法（SDE-DR）
- Orman らによる Self-adaptive Differential Evolution（SDE） [25]
- Soliman らによる DE の改良（Soliman） [65]
- Brest らによる DE の改良（Brest） [66]

実験結果を表 4.9 に，最良解平均推移を図 4.24 に示す．表 4.9 の報告値 [71] の結果は，Hoare による報告のものを用いている．表 4.9 から，SDE-DP-DR が最も良い解を発見しており，また最悪の最良解と平均の最良解においても全比較手法の中で最も良い解であることが確認される．従って，SDE-SP-DR は局所解が非常に多い問題においてもある程度安定した最適化性能を持つことが示唆される．また，図 4.24 を確認すると，SDE-SP-DR と SDE が探索の継続に成功しており，Soliman，Brest，SDE-DR は探索の継続に失敗していることが確認される．ただし，SDE は 3000000 摂動以内では SDE-SP-DR 程探索を進められていないことから，LJ-Problem あるいはそれに類似する問題に対して，SDE-SP-DR は有用であることが示唆される．

4.6　本章のまとめ

本稿では DE の改良手法である DE-SP に対して手法実装者によるパラメータ設定を排除する目的にて更なる改良を施された SDE-SP-DR を提案した．次に，性能比較のため同様の目的を持つ関連手法との比較実験について述べ，DE，DE-SP，SDE-SP-RJ と，

表 4.9: 各手法による LJ-Problem 最適化時の最良・最悪・平均解

	最良の最良解	最悪の最良解	平均の最良解
SDE-SP-DR	-28.3723	-27.1521	-27.7143
SDE-DR	-18.817	-7.80509	-11.930448
SDE	-8.26528	-6.42133	-7.10195
Soliman	-26.7461	-21.5368	-25.177425
Brest	-27.5559	-26.1161	-26.8935
報告値 [71]	-28.422532	-	-

他の目的を同じくする六つの関連手法の性能を，NF1，2 最適化実験にて比較した．この結果，SDE-SP-DR はこれらと比較して本実験で設定した基準においては SDE-SP-RJ のほぼ上位互換であり，NF1 に対しては全比較手法の中で最も高い性能を示すこと，他の手法では探索が停滞している一方で探索の継続が可能であることを確認した．一方ベンチマーク関数の最適化実験においては，NF2 の最適化実験で顕在化した SDE-SP-DR の解への収束低能の低下がこちらでも示唆される結果となった．解への収束速度は次元数に対して少なくとも線形に増加すると考えられるため，より収束性能の高い改良は引き続き模索するべきであろう．しかしながら，この実験においても SDE-SP-DR は探索の継続には成功していることが確認されたため，探索の停滞を回避する機能は，これらの目的関数においても働いていることが示唆される．LJ-Problem の最適化実験においては，SDE-SP-DR の性能が最も高く，かつ探索の継続にも成功しており，他の手法においては探索が停滞してしまうか，同一の実験条件内ではより悪い解しか発見できないということが端的に確認された．

以上のことから本稿において，SDE-SP-DR はパラメータ設定が不要な点で SDE-SP-RJ よりも扱いやすく，かつ性能が高いため，DE のパラメータ設定に関わる知見が無いのであれば，SDE-SP-RJ よりは本稿の SDE-SP-DR を用いた方が良いと述べることが可能であり，文献 [62, 61] にては完全に達成されなかった本研究の目的に一旦到達したと考えられるだろう．

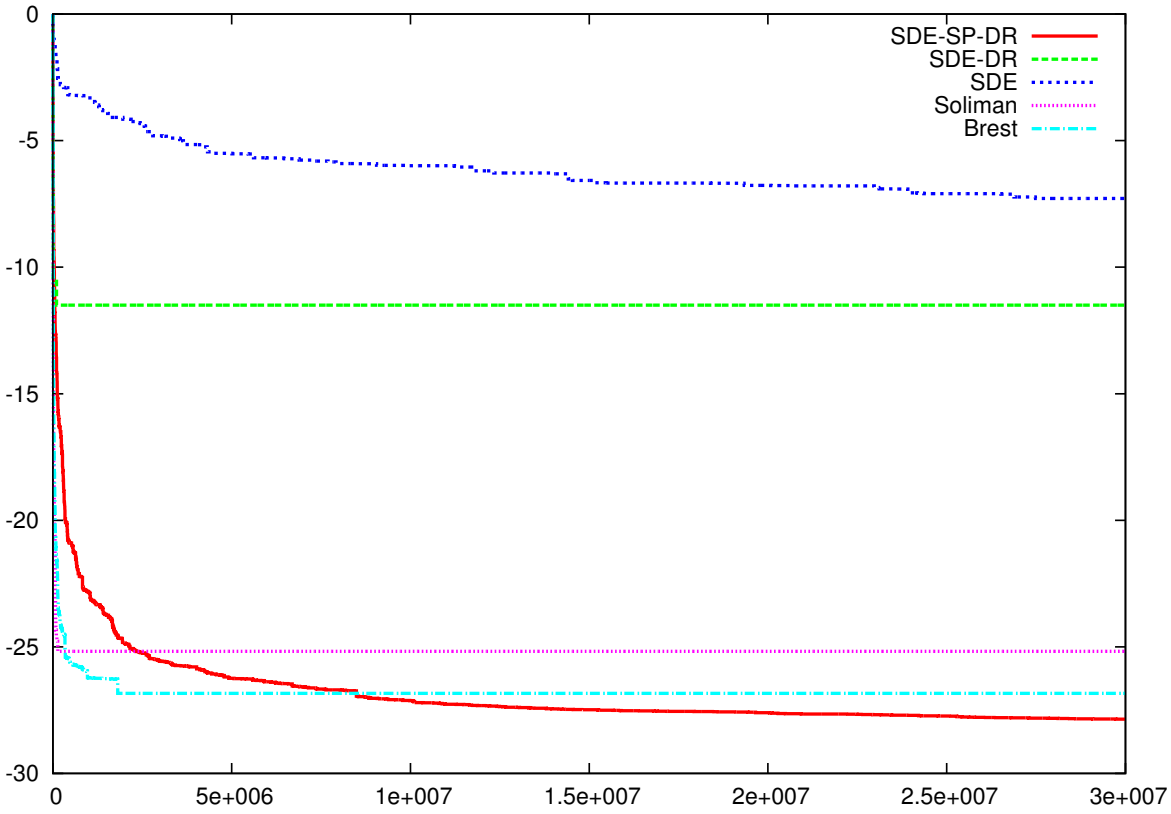


図 4.24: LJ-Problem 最適化時の最良解平均

第5章

結

本稿では，まず 2 章にて DE の改良手法である DE-SP を提案し，他の典型的な確率的最適化手法や DE の改良手法と比較して，最適化性能が最も高く，特に NF1 において顕著に優秀な手法であることを確認した．次に 3 章にて三つの Engineering Optimization Problem とロボットの立位姿勢保持制御問題の最適化実験を行い，DE が Pressure Vessel Optimization Problem において最も良い最適化性能を示すこと，ロボットの立位保持制御問題では DE-SP が最も良い最適化性能を示すこと，他の問題では DE と DE-SP の性能がほぼ同等であることを確認した．また，これらの結果から，DE は局所解が格子状に配置されているような問題で特に優秀である可能性があることを述べた．最後に 4 章では SDE-SP-DR を提案し，NF1，NF2 を用いた最適化実験，5 種のベンチマーク関数を用いた最適化実験，LJ-Problem を用いた最適化実験を行い，多くの摂動回数を必要とはするものの，設定を要するパラメータを完全に廃した DE-SP として提案可能な手法であろうことを確認した．したがって，本研究の二つの目標は最低限には達成されたと考えられる．

ただし，4 章から読み取れる本稿の実験結果からも明らかなように，関連手法と比較した場合に SDE-SP-DR は常に優秀な手法であるとはいえない．これは，DE-SP のパラメータ設定の困難性は廃されたが，本研究の目標達成の次に控えるであろうと考えられる問題の一つである手法選択の困難性を廃するまでには至らなかったことを示す．すなわち本稿における最終的な提案手法 SDE-SP-DR は，「DE-SP を用いようとしたがパラメータの上手い決め方が分からない」場合の解決策にはなるものの，「そもそもどのような最適化手法を用いるか決まっていない」場合の解決策とはなっていない．多数の最適化手法が存在している現在において，この「そもそもどのような最適化手法を用いるか決まっていない」場合において，実装が容易でかつ手法実装者による調整が不要な手法を提案することは重要であると著者らは考えている．すなわち，手法実装者による調整が不要でかつ多くの問題に対して有効な手法の提案が今後の課題であり，後続の研究が，これらを目標として為されることを期待したい．

謝辞

本研究の機会を与え，数々の適切な御指導を頂き，本稿の主査を務めて頂いた加藤昇平教授に深く感謝致します．そして，本稿の副査を努めて頂いた，伊藤孝行教授，犬塚信博教授に深く感謝致します．また，本研究を進めるにあたり，多くの助言を頂き，御協力頂いた加藤研究室の皆様と，参考文献を紹介して下さった株式会社ウサギィ所属の五木田和也様 (TwitterID:@kazoo04，2013 年 12 月 26 日現在) に，厚く御礼申し上げます．

参 考 文 献

[1] K. M. Ragsdell and D. T. Phillips: Optimal Design of a Class of Welded Structures Using Geometric Programming, *Journal of Manufacturing Science and Engineering*, Vol. 98, No. 3, pp. 1021–1025, 1976.

[2] 岩井亮, 加藤昇平: 探索停滞時にパラメータを再設定する Differential Evolution on Scattered Parents, 合同エージェントワークショップ＆シンポジウム JAWS(Joint Agent Workshop and Symposium), pp. B5–1 (6pages), 2012.

[3] E. Sandgren: Nonlinear Integer and Discrete Programming in Mechanical Design Optimization, *Journal of Mechanical Design*, Vol. 112, No. 2, pp. 223–229, 1990.

[4] M. Arakawa and I. Hagiwara: NONLINEAR INTEGER, DISCRETE AND CONTINUOUS OPTIMIZATION USING ADAPTIVE RANGE GENETIC ALGORITHMS, *Proceedings of ASME Design Engineering Technical Conferences*, pp. 1–10, 1997.

[5] C. A. C. Coello: Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems, *Computers in Industry*, Vol. 41, No. 2, pp. 113–127, 2000.

[6] L. C. Cagnina, S. C. Esquivel, and C. A. C. Coello: Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer, *Informatica*, Vol. 32, No. 3, pp. 319–326, 2008.

[7] ANSYS Chemkin-Pro - Combustion Simulation Software. http://www.ansys.com/products/fluids/ansys-chemkin-pro.

[8] H. Agarwal and H. G. Stenger: Development of a predictive kinetic model for homogeneous Hg oxidation data, *Mathematical and Computer Modelling*, Vol. 45, No. 1-2, pp. 109–125, 2007.

[9] S. Grillner and P. Wallen: Central Pattern Generators for Locomotion, with Special Reference to Vertebrates, journal = Annual Review of Neuroscience, Vol. 8, No. 1, pp. 233–261, 1985.

[10] 日栄悠, 加藤昇平, 伊藤英則: FF・FB 制御による包摂アーキテクチャにおける CPG 補償器を用いた歩行制御, 第 69 回情報処理学会全国大会, Vol. 69, No. 2, pp. 471–472, 2007.

[11] S. Kato and M. Ishida: *Dynamic Joint Passivization for Bipedal Locomotion*, chapter 11, pp. 219–236. Intech, 2011.

[12] 星野孝総: 差分進化アルゴリズムによるファジィ制御器の最適化, 高知工科大学紀要, Vol. 8, No. 1, pp. 25–35, 2011.

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi: Optimization by Simulated Annealing, *Science*, Vol. 220, No. 4598, pp. 671–680, 1983.

[14] W. Spendley, G. R. Hext, and F. R. Himsworth: Sequential Application of Simplex Designs in Optimization and Evolutionary Operation, *Technometrics*, Vol. 4, No. 4, pp. 441–461, 1962.

[15] J. A. Nelder and R. Mead: A Simplex Method for Function Minimization, *Computer Journal*, Vol. 7, No. 4, pp. 308–313, 1965.

[16] R. Eberhart and J. Kennedy: A new optimizer using particle swarm theory, In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, pp. 39–43, 1995.

[17] J. H. Holland: Outline for a Logical Theory of Adaptive Systems, *Journal of the ACM (JACM)*, Vol. 9, No. 3, pp. 297–314, 1962.

[18] H. G. Beyer and H. P. Schwefel: Evolution strategies - A comprehensive introduction, *Natural Computing*, Vol. 1, No. 1, pp. 3–52, 2002.

[19] 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.

[20] A. R. Mehrabian and C. Lucas: A novel numerical optimization algorithm inspired from weed colonization, *Ecological Informatics*, Vol. 1, No. 4, pp. 355 – 366, 2006.

[21] S. He, Q. H. Wu, and J. R. Saunders: Group Search Optimizer: An Optimization Algorithm Inspired by Animal Searching Behavior, *Evolutionary Computation, IEEE Transactions on*, Vol. 13, No. 5, pp. 973–990, 2009.

[22] D. Karaboga and B. Basturk: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of Global Optimization*, Vol. 39, pp. 459–471, 2007.

[23] X. S. Yang and S. Deb: Cuckoo Search via Lévy flights, In *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 210–214, 2009.

[24] J. Vesterstrom and R. Thomsen: A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems, *Proceedings of the 2004 Congress on Evolutionary Computation IEEE Cat No04TH8753*, Vol. 2, pp. 1980–1987, 2004.

[25] M. Omran, A. Salman, and A. Engelbrecht: Self-adaptive Differential Evolution, In Y. Hao, J. Liu, Y. Wang, Y. M. Cheung, H. Yin, L. Jiao, J. Ma, and Y. C. Jiao, editors, *Computational Intelligence and Security*, Vol. 3801 of *Lecture Notes in Computer Science*, pp. 192–199. Springer Berlin / Heidelberg, 2005.

[26] 串田淳一, 大場和久, 亀井且有: REAL: Differential Evolution における関数評価回数の削減の提案, 進化計算学会論文誌, Vol. 1, No. 1, pp. 79–88, 2010.

[27] B. Bošković, S. Greiner, J. Brest, and V. Žumer: A Differential Evolution for the Tuning of a Chess Evaluation Function, In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pp. 1851 –1856, 2006.

[28] 今川哲矢, 伊庭斉志: Differential Evolution を用いた外国為替取引システム, 人工知能学会研究会資料, Vol. 5, pp. SIG–FIN–005–01 (7pages), 2010.

[29] T. C. Hu, V. Klee, and D. Larman: Optimization of Globally Convex Functions, *SIAM Journal on Control and Optimization*, Vol. 27, No. 5, pp. 1026–1047, 1989.

[30] K. D. Boese: Cost Versus Distance In the Traveling Salesman Problem, Technical Report CA 90024-1596, UCLA Computer Science Department, 1995.

[31] 池田心, 小林重信: GA の探索における UV 現象と UV 構造仮説, 人工知能学会論文誌, Vol. 17, No. 3, pp. 239–246, 2002.

[32] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis: Finding multiple global optima exploiting differential evolution's niching capability, In *Differential Evolution (SDE), 2011 IEEE Symposium on*, pp. 1–8, 2011.

[33] T. Otani, R. Suzuki, and T. Arita: DE/isolated/1: A New Mutation Operator for Multimodal Optimization with Differential Evolution, In *The 24th Australasian Joint Conference on Artificial Intelligence (AI2011)*, pp. 321–330, 2011.

[34] H. Liu, Z. Cai, and Y. Wang: Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, *Applied Soft Computing*, Vol. 10, pp. 629–640, 2010.

[35] L. C. Y. Lai, C. W. Yeung, and F. H. F. Leung: A New Hybrid Differential Evolution with Wavelet Based Mutation and Crossover, In *Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*, pp. 722–729, 2009.

[36] M. Miki, T. Hiroyasu, and K. Ono: Simulated annealing with advanced adaptive neighborhood, In *Second international workshop on Intelligent systems design and application*, pp. 113–118, Atlanta, GA, USA, 2002. Dynamic Publishers, Inc.

[37] Z. W. Geem, J. H. Kim, and G. V. Loganathan: A New Heuristic Optimization Algorithm: Harmony Search, *SIMULATION*, Vol. 76, No. 2, pp. 60–68, 2001.

[38] 小野功, 佐藤浩, 小林重信: 単峰性正規分布交叉 UNDX を用いた実数値 GA による関数最適化, 人工知能学会誌, Vol. 14, No. 6, pp. 1146–1155, 1999-11-01.

[39] 佐藤浩, 小野功, 小林重信: 遺伝的アルゴリズムにおける世代交代モデルの提案と評価, 人工知能学会誌, Vol. 12, No. 5, pp. 734–744, 1997.

[40] L. J. Eshelman: Real-Coded Genetic Algorithms and Interval-Schemata, *Foundations of Genetic Algorithms*, Vol. 2, pp. 187–202, 1993.

[41] R. A. Krohling and L. dos Santos Coelho: Coevolutionary Particle Swarm Optimization Using Gaussian Distribution for Solving Constrained Optimization Problems, *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, Vol. 36, No. 6, pp. 1407–1416, 2006.

[42] 林孝志郎, 片山謙吾, 南原英生, 成久洋之: 分割階層型 Particle Swarm Optimization, 進化計算研究会 進化計算シンポジウム 2007 講演論文集, pp. 31–34, 2007.

[43] P. Barthelemy, J. Bertolotti, and D. S. Wiersma: A Lévy flight for light, *Nature*, pp. 495–498, 2008.

[44] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller: Equation of State Calculation by Fast Computing Machines, *Journal of Chemical Physics*, Vol. 21, No. 6, pp. 1087–1092, 1953.

[45] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright: Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions, *Siam Journal on Optimization*, Vol. 9, No. 1, pp. 112–147, 1998.

[46] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Hannery: ニューメリカルレシピ・イン・シー　C 言語による数値計算のレシピ, 第 10.4 章, pp. 295–299. 技術評論社.

[47] K. Deb: An Efficient Constraint Handling Method for Genetic Algorithms, *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, No. 2-4, pp. 311–338, 2000.

[48] K. Zielinski and R. Laur: Constrained Single-Objective Optimization Using Differential Evolution, In *IEEE Congress on Evolutionary Computation*, pp. 223–230, 2006.

[49] S. Kukkonen and J. Lampinen: Constrained Rean-Parameter Optimization with Generalized Differential Evolution, In *IEEE Congress on Evolutionary Computation*, pp. 207–214, 2006.

[50] G. G. Yen and M. Daneshyari: Diversity-based Information Exchange among Multiple Swarms in Particle Swarm Optimization, In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pp. 1686 –1693, 2006.

[51] S. Janson and M. Middendorf: A hierarchical particle swarm optimizer and its adaptive variant, *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, Vol. 35, No. 6, pp. 1272 –1282, 2005.

[52] R. Dawkins: *The Extended Phenotype: The Long Reach of the Gene*, chapter 9, pp. 156–164. Oxford University Press.

[53] 岩井亮, 加藤昇平: 二足ロボットの立位姿勢保持 PI-D 制御と Nelder-Mead Simplex 法による PID ゲインの準最適化, 平成 23 年度電気関係学会東海支部連合大会予稿集, pp. P5–3, 2011.

[54] D. K. Shin, Z. Gurdal, and O. H. Griffin, Jr.: A PENALTY APPROACH FOR NONLINEAR OPTIMIZATION WITH DISCRETE DESIGN VARIABLES, *Engineering Optimization*, Vol. 16, No. 1, pp. 29–42, 1990.

[55] J. Golinski: An Adaptive Optimization System Applied to Machine Synthesis, *Mechanism and Machine Theory*, Vol. 8, No. 4, pp. 419–436, 1973.

[56] Open Dynamics Engine - home. http://www.ode.org/.

[57] 杉原知道: 最良重心-ZMP レギュレータに基づく二脚運動の立位可安定性と踏み出し, ロボティクスシンポジア予稿集, Vol. 14, pp. 435–440, 2009.

[58] R. Gämperle, S. D. Müller, and P. Koumoutsakos: A Parameter Study for Differential Evolution, In *WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pp. 293–298. Press, 2002.

[59] J. Liu and J. Lampinen: On Setting the Control Parameter of the Differential Evolution Method, In *Proceedings of the 8th International Conference on Soft Computing (MENDEL 2002)*, pp. 11–18, 2002.

[60] 岩井亮, 加藤昇平: DE-SP での最適化における適応的パラメータ調整, 平成 24 年度電気関係学会東海支部連合大会予稿集, pp. D3–5 (1page), 2012.

[61] 岩井亮, 加藤昇平: 探索の停滞に応じてパラメータを再設定する Differential Evolution on Scattered Parents, 情報処理学会論文誌, Vol. 56, No. 2, pp. 733–743, 2015.

[62] 岩井亮: 親の散在性を考慮した Differential Evolution の性能向上と省パラメータ化, 名古屋工業大学加藤昇平研究室修士論文, pp. 1–151, 2015.

[63] J. Liu and J. Lampinen: A Fuzzy Adaptive Differential Evolution Algorithm, *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, Vol. 9, No. 6, pp. 448–462, 2005.

[64] N. Noman, D. Bollegala, and H. Iba: An Adaptive Differential Evolution Algorithm, In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 2229–2236, 2011.

[65] O. S. Soliman and L. T. Bui: A Self-Adaptive Strategy for Controlling Parameters in Differential Evolution, In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 2837–2842, 2008.

[66] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer: Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems, *Evolutionary Computation, IEEE Transactions on*, Vol. 10, No. 6, pp. 646–657, 2006.

[67] 山口智: Differential Evolution における制御変数の自動調整, *IEEJ Transactions on Electronics, Information and Systems*, Vol. 128, No. 11, pp. 1696–1703, 2008.

[68] T. Takagi and M. Sugeno: Fuzzy Identification of Systems and Its Applications to Modeling and Control, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-15, No. 1, pp. 116–132, 1985.

[69] J. E. Jones: On the Determination of Molecular Fields. II. From the Equation of State of a Gas, *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, Vol. 106, No. 738, pp. 463–477, 1924.

[70] 岩井亮, 加藤昇平: Differential Evolution on Scattered Parents による大域的単峰性に乏しい多峰性探索空間における最適化, 電気学会論文誌 C(電子・情報・システム部門誌) 特集：省電力時代の電子回路技術, Vol. 133, No. 2, pp. 410–417, 2013.

[71] Lennard-Jones Clusters. http://doye.chem.ox.ac.uk/jon/structures/LJ.html.

[72] S. Paterlini and T. Krink: High performance clustering with differential evolution, In *Evolutionary Computation, 2004. CEC2004. Congress on*, Vol. 2, pp. 2004–2011, 2004.

[73] P.-F. Verhulst: Notice sur la loi que la population poursuit dans son accroissement, *Correspondance mathématique et physique*, Vol. 10, , 1838.

[74] M. R. Hoare and P. Pal: Physical cluster mechanics: statics and energy surfaces for monatomic systems, *Advances in Physics*, Vol. 20, No. 84, p. 161, 1971.

[75] Y. Xiang, H. Jiang, W. Cai, and X. Shao: An Efficient Method Based on Lattice Construction and the Genetic Algorithm for Optimization of Large Lennard-Jones Clusters, *The Journal of Physical Chemistry A*, Vol. 108, No. 16, pp. 3586–3592, 2004.

[76] Леонид Генрихович Хачиян: A polynomial algorithm in linear programming (in Russian), *Doklady Akademi Nauk SSSR 224*, pp. 1097–1096, 1979.

[77] N. Karmarkar: A New Polynomial Time Algorithm for Linear Programming, *Combinatorica*, Vol. 4, No. 4, pp. 373–395, 1984.

[78] C. Audet and J. E. Dennis, Jr.: Mesh Adaptive Direct Search Algorithms for Constrained Optimization, *SIAM Journal on Optimization*, Vol. 17, No. 1, pp. 188–217, 2006.

付 録 A

2章の補遺

A.1　4 点バイリニア法

4 点バイリニア法は，$\mathbb{R}^2$ の空間において，$f(a,b)$，$f(a,d)$，$f(c,b)$，$f(c,d)$（ただし $a<c$，$b<d$）が既知であるときに，$f(x,y)$（ただし $a<x<c \wedge b<y<d$）を線形補間する手法である．補間式は次のように表される．

$$f(x,y)=(1-u)(1-v)f(a,b)+u(1-v)f(c,b)+(1-u)vf(a,d)+uvf(c,d) \quad \text{(A.1)}$$

ただし，u，v は，それぞれ x，y の矩形 $(a,b)(c,d)$ の UV 座標値であり，次のように表される．

$$u=\frac{x-a}{c-a} \quad \text{(A.2)}$$

$$v=\frac{y-b}{d-b} \quad \text{(A.3)}$$

A.2　Wavelet Function

Wavelet Function とは，Wavelet 理論における Wavelet を構成する二つの基底関数を構成する要素の一つである．WMWC-DE において利用された Wavelet Function は Morlet Wavelet であり [72]，次の式で表される．

$$\psi(x)=e^{-\frac{x^2}{2}}\cos(5x) \quad \text{(A.4)}$$

A.3　r-K 戦略説

r-K 戦略説は生態学に関する仮設の一つであり，生物の種はどのように子孫を残すかについて，r 戦略と K 戦略の二つの戦略の間で選択を迫られているとする説である．こ

こでの r と K は，ロジスティック式内の内的自然増加率 r と環境収容力 K のことを表している．

A.3.1　ロジスティック式

ロジスティック式とは Verhulst が提案した人口増加モデルとしての微分方程式である [73].

$$\frac{dN}{dt} = r\left(\frac{K-N}{K}\right)N \tag{A.5}$$

ここで，r は内的増加率であり対象とする生物が実現し得る最大の増加率を表し，K は環境収容力であり，現在の環境における個体数の定員（上限）を表す．この式は $0 < N < K$ の下で解析的に求めることができ，次の式として表される．

$$N = \frac{K}{1+e^{rK(t_0-t)}} \quad (t_0\text{は任意初期値}) \tag{A.6}$$

このモデルは非常に単純なものであり，生物学的には成立し得ない仮定に基づいているが，解析的に解が求められるという点が大きな利点となっており，様々な利用がされている．

A.3.2　r 戦略

r 戦略は個体群の内的増加率を高くする方向への進化を行う戦略の事である．即ち，同じ時間でできる限り大量の子孫を残し，なるべく早く成長し，寿命は短めとする戦略である．現在の個体数が少ない場合は有効な戦略であると考えられているが，個体数が増え，環境収容力により頭打ちとなると，大量に残した子孫のほとんどが無駄となってしまう．

A.3.3　K 戦略

K 戦略は環境収容力を高くする方向への進化を行う戦略，あるいは環境収容力に応じて個体数を維持する方向への進化を行う戦略である．即ち，個体の大きさを小さくし，エネルギー効率をよくし，寿命は長めとする戦略である．K 戦略を採る生物は，結果として少数の確実に育つ子を産むという形態に落ち着くとされる．IWO での K 戦略はこの「少数を確実に育てる」戦略に基づいている．

A.4　実験に用いたパラメータ

2.5 節の実験において，各ベンチマーク関数にて各手法に適用した用いたパラメータをここに列挙する．

Rastrigin Function

- DE-SP: $(F, C_R, M) = (0.3, 0.5, 3)$
- DE: $(F, C_R) = (0.3, 0.5)$
- DE-SP($M = 0$): $(F, C_R, M) = (0.3, 0.5, 0)$
- DE2: $(F, C_R, M) = (0.3, 0.5, 3)$
- DE/nrand/1: $(F, C_R) = (1.3, 0.3)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (0.5, 0.5, 49, 48)$
- PSO-DE: $(F, C_R) = (0.7, 0.3)$
- WMWC-DE: $(g, \zeta_{wm}, C_R) = (10000, 1.0, 0.5)$
- SA: $(T, \lambda) = (1000, 0.9999)$
- SA/AAN: $(T, \lambda, \boldsymbol{m}_p, p_{\max}, p_{\min}) = (1000, \dfrac{\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}}{2}, 0.6, 0.4)$
- NMS: $(\alpha, \beta, \gamma, \delta) = (1, 2, 0.5, 0.0001)$
- HS: $(P_{\text{hmcr}}, P_{\text{par}}, \delta_{\text{HS}}) = (0.9, 0.8, 10)$
- GA(Uni+Elite): $(P_m) = (0.05)$（ただし，以下 P_m は突然変異率を表す.）
- GA(UNDX+MGG): $(P_m, \alpha, \beta) = (0.05, 0.5, 0.35)$
- GA(BLX-α+MGG): $(P_m, T_{\text{cross}}, \alpha) = (0.05, 25, 0.05)$
- $(1+1)$-ES: $(\sigma, c_{\frac{1}{5}}) = (50, 0.9999)$
- $(\mu+\lambda)$-ES: $(\mu, \lambda, \tau, \tau', \beta, c_{\mu+\lambda}) = (50, 50, \dfrac{1}{\sqrt{2\sqrt{2}}}, \dfrac{1}{\sqrt{4}}, 0.0873, 1)$
- PSO: $(w, c_1, c_2) = (0.95, 1, 1)$
- PSO-KS: 設定パラメータなし
- DH-PSO: $(H, N_{h_1}, N_{h_2}, w, c_1, c_2, S_{\text{migration}}, S_{\text{exchange}}, S_{\text{reset}})$
  $= (2, 5 \times 5, 25, 0.95, 1, 1, 1000, 1000, 1000)$

- IWO: $(\sigma_{\text{initial}}, \sigma_{\text{final}}, n, N_{\text{first}}) = (100, 0.000001, 1, 5)$
- GSO: $(N_s, l_{\max}, \theta_{\max}, \alpha_{\max}, a)$
  $= (40, |\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|, \dfrac{\pi}{\lfloor\sqrt{3}+0.5\rfloor^2}, \dfrac{\pi}{2\lfloor\sqrt{3}+0.5\rfloor^2}, \lfloor\sqrt{3}+0.5\rfloor)$
- ABC: $(N_s) = (10)$
- RS: 設定パラメータなし
- LF: $(\beta) = (1.5)$
- CS: $(\beta, \alpha, p_a) = (1.5, \dfrac{|\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|}{100}, 0.25)$

Rosenbrock Function

- DE-SP: $(F, C_R, M) = (0.7, 0.5, 3)$
- DE: $(F, C_R) = (0.5, 0.5)$
- DE-SP($M = 0$): $(F, C_R, M) = (0.7, 0.5, 0)$
- DE2: $(F, C_R, M) = (0.5, 0.5, 3)$
- DE/nrand/1: $(F, C_R) = (0.5, 0.5)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (0.5, 0.5, 49, 48)$
- PSO-DE: $(F, C_R) = (0.5, 0.5)$
- WMWC-DE: $(g, \zeta_{wm}, C_R) = (10000, 1.0, 0.5)$
- SA: $(T, \lambda) = (1000, 0.9999)$
- SA/AAN: $(T, \lambda, \boldsymbol{m}_p, p_{\max}, p_{\min}) = (1000, \dfrac{\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}}{2}, 0.6, 0.4)$
- NMS: $(\alpha, \beta, \gamma, \delta) = (1, 2, 0.5, 0.0001)$
- HS: $(P_{\text{hmcr}}, P_{\text{par}}, \delta_{\text{HS}}) = (0.9, 0.8, 10)$
- GA(Uni+Elite): $(P_m) = (0.05)$（ただし，以下 P_m は突然変異率を表す.）
- GA(UNDX+MGG): $(P_m, \alpha, \beta) = (0.05, 0.5, 0.35)$
- GA(BLX-α+MGG): $(P_m, T_{\text{cross}}, \alpha) = (0.05, 25, 0.05)$
- $(1+1)$-ES: $(\sigma, c_{\frac{1}{5}}) = (50, 0.9999)$
- $(\mu+\lambda)$-ES: $(\mu, \lambda, \tau, \tau', \beta, c_{\mu+\lambda}) = (50, 50, \dfrac{1}{\sqrt{2\sqrt{2}}}, \dfrac{1}{\sqrt{4}}, 0.0873, 1)$

- PSO: $(w, c_1, c_2) = (0.95, 1, 1)$
- PSO-KS: 設定パラメータなし
- DH-PSO: $(H, N_{h_1}, N_{h_2}, w, c_1, c_2, S_{\text{migration}}, S_{\text{exchange}}, S_{\text{reset}})$
  $= (2, 5 \times 5, 25, 0.95, 1, 1, 1000, 1000, 1000)$
- IWO: $(\sigma_{\text{initial}}, \sigma_{\text{final}}, n, N_{\text{first}}) = (100, 0.000001, 1, 5)$
- GSO: $(N_s, l_{\max}, \theta_{\max}, \alpha_{\max}, a)$
  $= (40, |\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|, \dfrac{\pi}{\lfloor\sqrt{3}+0.5\rfloor^2}, \dfrac{\pi}{2\lfloor\sqrt{3}+0.5\rfloor^2}, \lfloor\sqrt{3}+0.5\rfloor)$
- ABC: $(N_s) = (10)$
- RS: 設定パラメータなし
- LF: $(\beta) = (1.5)$
- CS: $(\beta, \alpha, p_a) = (1.5, \dfrac{|\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|}{100}, 0.25)$

Schwefel Function

- DE-SP: $(F, C_R, M) = (0.2, 0.5, 3)$
- DE: $(F, C_R) = (0.4, 0.4)$
- DE-SP($M = 0$): $(F, C_R, M) = (0.2, 0.5, 0)$
- DE2: $(F, C_R, M) = (0.4, 0.4, 3)$
- DE/nrand/1: $(F, C_R) = (0.2, 0.6)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (0.4, 0.4, 49, 48)$
- PSO-DE: $(F, C_R) = (0.7, 0.5)$
- WMWC-DE: $(g, \zeta_{wm}, C_R) = (10000, 1.0, 0.5)$
- SA: $(T, \lambda) = (1000, 0.9999)$
- SA/AAN: $(T, \lambda, \boldsymbol{m}_p, p_{\max}, p_{\min}) = (1000, \dfrac{\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}}{2}, 0.6, 0.4)$
- NMS: $(\alpha, \beta, \gamma, \delta) = (1, 2, 0.5, 0.0001)$
- HS: $(P_{\text{hmcr}}, P_{\text{par}}, \delta_{\text{HS}}) = (0.9, 0.8, 10)$
- GA(Uni+Elite): $(P_m) = (0.05)$（ただし，以下 P_m は突然変異率を表す.）
- GA(UNDX+MGG): $(P_m, \alpha, \beta) = (0.05, 0.5, 0.35)$

- GA(BLX-α+MGG): $(P_m, T_{\mathrm{cross}}, \alpha) = (0.05, 25, 0.05)$
- $(1+1)$-ES: $(\sigma, c_{\frac{1}{5}}) = (50, 0.9999)$
- $(\mu+\lambda)$-ES: $(\mu, \lambda, \tau, \tau', \beta, c_{\mu+\lambda}) = (50, 50, \dfrac{1}{\sqrt{2\sqrt{2}}}, \dfrac{1}{\sqrt{4}}, 0.0873, 1)$
- PSO: $(w, c_1, c_2) = (0.95, 1, 1)$
- PSO-KS: 設定パラメータなし
- DH-PSO: $(H, N_{h_1}, N_{h_2}, w, c_1, c_2, S_{\mathrm{migration}}, S_{\mathrm{exchange}}, S_{\mathrm{reset}})$
  $= (2, 5\times 5, 25, 0.95, 1, 1, 1000, 1000, 1000)$
- IWO: $(\sigma_{\mathrm{initial}}, \sigma_{\mathrm{final}}, n, N_{\mathrm{first}}) = (100, 0.000001, 1, 5)$
- GSO: $(N_s, l_{\max}, \theta_{\max}, \alpha_{\max}, a)$
  $= (40, |\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|, \dfrac{\pi}{\lfloor\sqrt{3}+0.5\rfloor^2}, \dfrac{\pi}{2\lfloor\sqrt{3}+0.5\rfloor^2}, \lfloor\sqrt{3}+0.5\rfloor)$
- ABC: $(N_s) = (10)$
- RS: 設定パラメータなし
- LF: $(\beta) = (1.5)$
- CS: $(\beta, \alpha, p_a) = (1.5, \dfrac{|\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|}{100}, 0.25)$

Griewank Function

- DE-SP: $(F, C_R, M) = (0.5, 0.5, 3)$
- DE: $(F, C_R) = (0.5, 0.5)$
- DE-SP$(M=0)$: $(F, C_R, M) = (0.5, 0.5, 0)$
- DE2: $(F, C_R, M) = (0.5, 0.5, 3)$
- DE/nrand/1: $(F, C_R) = (1.8, 0.3)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (0.5, 0.5, 49, 48)$
- PSO-DE: $(F, C_R) = (0.3, 0.9)$
- WMWC-DE: $(g, \zeta_{wm}, C_R) = (10000, 1.0, 0.5)$
- SA: $(T, \lambda) = (1000, 0.9999)$
- SA/AAN: $(T, \lambda, \boldsymbol{m}_p, p_{\max}, p_{\min}) = (1000, \dfrac{\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}}{2}, 0.6, 0.4)$

- NMS: $(\alpha, \beta, \gamma, \delta) = (1, 2, 0.5, 0.0001)$
- HS: $(P_{\mathrm{hmcr}}, P_{\mathrm{par}}, \delta_{\mathrm{HS}}) = (0.9, 0.8, 10)$
- GA(Uni+Elite): $(P_m) = (0.05)$（ただし，以下 P_m は突然変異率を表す.）
- GA(UNDX+MGG): $(P_m, \alpha, \beta) = (0.05, 0.5, 0.35)$
- GA(BLX-α+MGG): $(P_m, T_{\mathrm{cross}}, \alpha) = (0.05, 25, 0.05)$
- $(1+1)$-ES: $(\sigma, c_{\frac{1}{5}}) = (50, 0.9999)$
- $(\mu+\lambda)$-ES: $(\mu, \lambda, \tau, \tau', \beta, c_{\mu+\lambda}) = (50, 50, \dfrac{1}{\sqrt{2\sqrt{2}}}, \dfrac{1}{\sqrt{4}}, 0.0873, 1)$
- PSO: $(w, c_1, c_2) = (0.95, 1, 1)$
- PSO-KS: 設定パラメータなし
- DH-PSO: $(H, N_{h_1}, N_{h_2}, w, c_1, c_2, S_{\mathrm{migration}}, S_{\mathrm{exchange}}, S_{\mathrm{reset}})$
  $= (2, 5 \times 5, 25, 0.95, 1, 1, 1000, 1000, 1000)$
- IWO: $(\sigma_{\mathrm{initial}}, \sigma_{\mathrm{final}}, n, N_{\mathrm{first}}) = (100, 0.000001, 1, 5)$
- GSO: $(N_s, l_{\max}, \theta_{\max}, \alpha_{\max}, a)$
  $= (40, |\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|, \dfrac{\pi}{\lfloor\sqrt{3}+0.5\rfloor^2}, \dfrac{\pi}{2\lfloor\sqrt{3}+0.5\rfloor^2}, \lfloor\sqrt{3}+0.5\rfloor)$
- ABC: $(N_s) = (10)$
- RS: 設定パラメータなし
- LF: $(\beta) = (1.5)$
- CS: $(\beta, \alpha, p_a) = (1.5, \dfrac{|\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|}{100}, 0.25)$

Ackley Function

- DE-SP: $(F, C_R, M) = (0.5, 0.5, 3)$
- DE: $(F, C_R) = (0.5, 0.5)$
- DE-SP($M = 0$): $(F, C_R, M) = (0.5, 0.5, 0)$
- DE2: $(F, C_R, M) = (0.5, 0.5, 3)$
- DE/nrand/1: $(F, C_R) = (0.5, 0.5)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (0.4, 0.5, 49, 48)$

- PSO-DE: $(F, C_R) = (0.5, 0.5)$
- WMWC-DE: $(g, \zeta_{wm}, C_R) = (10000, 1.0, 0.5)$
- SA: $(T, \lambda) = (1000, 0.9999)$
- SA/AAN: $(T, \lambda, \boldsymbol{m}_p, p_{\max}, p_{\min}) = (1000, \dfrac{\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}}{2}, 0.6, 0.4)$
- NMS: $(\alpha, \beta, \gamma, \delta) = (1, 2, 0.5, 0.0001)$
- HS: $(P_{\text{hmcr}}, P_{\text{par}}, \delta_{\text{HS}}) = (0.9, 0.8, 10)$
- GA(Uni+Elite): $(P_m) = (0.05)$（ただし，以下 P_m は突然変異率を表す.）
- GA(UNDX+MGG): $(P_m, \alpha, \beta) = (0.05, 0.5, 0.35)$
- GA(BLX-α+MGG): $(P_m, T_{\text{cross}}, \alpha) = (0.05, 25, 0.05)$
- $(1+1)$-ES: $(\sigma, c_{\frac{1}{5}}) = (50, 0.9999)$
- $(\mu+\lambda)$-ES: $(\mu, \lambda, \tau, \tau', \beta, c_{\mu+\lambda}) = (50, 50, \dfrac{1}{\sqrt{2\sqrt{2}}}, \dfrac{1}{\sqrt{4}}, 0.0873, 1)$
- PSO: $(w, c_1, c_2) = (0.95, 1, 1)$
- PSO-KS: 設定パラメータなし
- DH-PSO: $(H, N_{h_1}, N_{h_2}, w, c_1, c_2, S_{\text{migration}}, S_{\text{exchange}}, S_{\text{reset}})$
  $= (2, 5 \times 5, 25, 0.95, 1, 1, 1000, 1000, 1000)$
- IWO: $(\sigma_{\text{initial}}, \sigma_{\text{final}}, n, N_{\text{first}}) = (100, 0.000001, 1, 5)$
- GSO: $(N_s, l_{\max}, \theta_{\max}, \alpha_{\max}, a)$
  $= (40, |\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|, \dfrac{\pi}{\lfloor\sqrt{3}+0.5\rfloor^2}, \dfrac{\pi}{2\lfloor\sqrt{3}+0.5\rfloor^2}, \lfloor\sqrt{3}+0.5\rfloor)$
- ABC: $(N_s) = (10)$
- RS: 設定パラメータなし
- LF: $(\beta) = (1.5)$
- CS: $(\beta, \alpha, p_a) = (1.5, \dfrac{|\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|}{100}, 0.25)$

Noisy Function 1

- DE-SP: $(F, C_R, M) = (1.0, 0.5, 3)$
- DE: $(F, C_R) = (0.5, 0.1)$
- DE-SP($M = 0$): $(F, C_R, M) = (1.0, 0.5, 0)$
- DE2: $(F, C_R, M) = (0.5, 0.1, 3)$
- DE/nrand/1: $(F, C_R) = (0.4, 0.1)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (1.1, 0.5, 49, 48)$
- PSO-DE: $(F, C_R) = (0.7, 0.4)$
- WMWC-DE: $(g, \zeta_{wm}, C_R) = (10000, 1.0, 0.5)$
- SA: $(T, \lambda) = (1000, 0.9999)$
- SA/AAN: $(T, \lambda, \boldsymbol{m}_p, p_{\max}, p_{\min}) = (1000, \dfrac{\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}}{2}, 0.6, 0.4)$
- NMS: $(\alpha, \beta, \gamma, \delta) = (1, 2, 0.5, 0.0001)$
- HS: $(P_{\text{hmcr}}, P_{\text{par}}, \delta_{\text{HS}}) = (0.2, 0.7, 10)$
- GA(Uni+Elite): $(P_m) = (0.05)$（ただし，以下 P_m は突然変異率を表す.）
- GA(UNDX+MGG): $(P_m, \alpha, \beta) = (0.05, 0.5, 0.35)$
- GA(BLX-α+MGG): $(P_m, T_{\text{cross}}, \alpha) = (0.05, 25, 0.05)$
- $(1 + 1)$-ES: $(\sigma, c_{\frac{1}{5}}) = (50, 0.9999)$
- $(\mu + \lambda)$-ES: $(\mu, \lambda, \tau, \tau', \beta, c_{\mu+\lambda}) = (50, 50, \dfrac{1}{\sqrt{2\sqrt{2}}}, \dfrac{1}{\sqrt{4}}, 0.0873, 1)$
- PSO: $(w, c_1, c_2) = (0.95, 1, 1)$
- PSO-KS: 設定パラメータなし
- DH-PSO: $(H, N_{h_1}, N_{h_2}, w, c_1, c_2, S_{\text{migration}}, S_{\text{exchange}}, S_{\text{reset}})$
  $= (2, 5 \times 5, 25, 0.95, 1, 1, 1000, 1000, 1000)$
- IWO: $(\sigma_{\text{initial}}, \sigma_{\text{final}}, n, N_{\text{first}}) = (100, 0.000001, 1, 5)$
- GSO: $(N_s, l_{\max}, \theta_{\max}, \alpha_{\max}, a)$
  $= (40, |\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|, \dfrac{\pi}{\lfloor\sqrt{3} + 0.5\rfloor^2}, \dfrac{\pi}{2\lfloor\sqrt{3} + 0.5\rfloor^2}, \lfloor\sqrt{3} + 0.5\rfloor)$
- ABC: $(N_s) = (10)$

- RS: 設定パラメータなし
- LF: $(\beta) = (1.5)$
- CS: $(\beta, \alpha, p_a) = (1.5, \dfrac{|\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|}{100}, 0.25)$

Noisy Function 2

- DE-SP: $(F, C_R, M) = (1.6, 0.2, 3)$
- DE: $(F, C_R) = (1.4, 0.1)$
- DE-SP($M = 0$): $(F, C_R, M) = (1.6, 0.2, 0)$
- DE2: $(F, C_R, M) = (1.4, 0.1, 3)$
- DE/nrand/1: $(F, C_R) = (1.3, 0.2)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (1.9, 0.1, 49, 48)$
- PSO-DE: $(F, C_R) = (0.9, 0.6)$
- WMWC-DE: $(g, \zeta_{wm}, C_R) = (10000, 1.0, 0.5)$
- SA: $(T, \lambda) = (1000, 0.9999)$
- SA/AAN: $(T, \lambda, \boldsymbol{m}_p, p_{\max}, p_{\min}) = (1000, \dfrac{\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}}{2}, 0.6, 0.4)$
- NMS: $(\alpha, \beta, \gamma, \delta) = (1, 2, 0.5, 0.0001)$
- HS: $(P_{\text{hmcr}}, P_{\text{par}}, \delta_{\text{HS}}) = (0.9, 0.8, 10)$
- GA(Uni+Elite): $(P_m) = (0.05)$（ただし，以下 P_m は突然変異率を表す.）
- GA(UNDX+MGG): $(P_m, \alpha, \beta) = (0.05, 0.5, 0.35)$
- GA(BLX-α+MGG): $(P_m, T_{\text{cross}}, \alpha) = (0.05, 25, 0.05)$
- $(1+1)$-ES: $(\sigma, c_{\frac{1}{5}}) = (50, 0.9999)$
- $(\mu+\lambda)$-ES: $(\mu, \lambda, \tau, \tau', \beta, c_{\mu+\lambda}) = (50, 50, \dfrac{1}{\sqrt{2\sqrt{2}}}, \dfrac{1}{\sqrt{4}}, 0.0873, 1)$
- PSO: $(w, c_1, c_2) = (0.95, 1, 1)$
- PSO-KS: 設定パラメータなし
- DH-PSO: $(H, N_{h_1}, N_{h_2}, w, c_1, c_2, S_{\text{migration}}, S_{\text{exchange}}, S_{\text{reset}})$
  $= (2, 5 \times 5, 25, 0.95, 1, 1, 1000, 1000, 1000)$

- IWO: $(\sigma_{\text{initial}}, \sigma_{\text{final}}, n, N_{\text{first}}) = (100, 0.000001, 1, 5)$
- GSO: $(N_s, l_{\max}, \theta_{\max}, \alpha_{\max}, a)$
  $= (40, |\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|, \dfrac{\pi}{\lfloor\sqrt{3}+0.5\rfloor^2}, \dfrac{\pi}{2\lfloor\sqrt{3}+0.5\rfloor^2}, \lfloor\sqrt{3}+0.5\rfloor)$
- ABC: $(N_s) = (10)$
- RS: 設定パラメータなし
- LF: $(\beta) = (1.5)$
- CS: $(\beta, \alpha, p_a) = (1.5, \dfrac{|\boldsymbol{\chi}^{\max} - \boldsymbol{\chi}^{\min}|}{100}, 0.25)$

付 録B

3章の補遺

B.1　測定値微分先行型PID制御

本節では測定値微分先行型PID制御（PI-D制御）の詳細について述べる．まず，通常のPID制御について述べ，続いてPID制御の一改良であるPI-D制御について述べる．

B.1.1　PID制御

PID制御は，フィードバック制御の一つであり，操作量（システムへ入力する値）を現在の制御量（システムで制御する値および現在の出力値）と目標値の偏差に比例する値，その積分に比例する値，その微分に比例する値の和で決定する手法である．これは，操作量を$x(t)$，現在の制御量を$y(t-1)$，目標値を$y_d(t-1)$とすると，次の様に表される．

$$x(t) = K_P(y(t-1)-y_d(t-1))+K_I\int(y(t-1)-y_d(t-1))\,dt+K_D\frac{d}{dt}(y(t-1)-y_d(t-1)) \tag{B.1}$$

ここで，K_P，K_I，K_Dはそれぞれ比例ゲイン，積分ゲイン，微分ゲインである．これら四つは使用者によって与えられる定数である．

この制御において，比例項は制御量を目標値に近づける役割を持ち，現在の制御量が目標値から離れているほど大きな値となる．しかし，比例項だけでは，偏差が小さくなった時に操作量が過小となり，制御量が目標値に到達する前に制御が停止してしまう．このときに残る偏差を残留偏差という．積分項は，この残留偏差を積分して操作量にフィードバックし，操作量が過小となる事を防ぎ，残留偏差を解消する働きを持つ．また，比例項と積分項だけでは，外乱などによる制御量の変化や，目標値の変更による偏差の急変に対応することは困難である．微分項は，偏差の変化に応じて制御量を調整するため，この問題を緩和する働きを持つ．

B.1.2　測定値微分先行型 PID 制御

単純な PID 制御では，目標値に変更が発生した際，微分項が大きく変化し，それに応じて操作量も大きく変化する．したがって，システムの応答性が高い場合，過剰な制御が発生して不安定化することがある．この現象を微分キックという．この問題は，微分項には偏差ではなく，現在の制御量のみを用いることで回避できる．このような制御を測定値微分先行型 PID 制御（PI-D 制御）といい，次の様に表される．

$$x(t) = K_P(y(t-1) - y_d(t-1)) + K_I \int (y(t-1) - y_d(t-1))\, dt + K_D \frac{d}{dt} y(t-1) \quad \text{(B.2)}$$

本研究では，この PI-D 制御をモータの制御器として用いた．

B.2　実験に用いたパラメータ

3.3 節の実験において，各問題にて各手法に適用したパラメータをここに列挙する．

B.2.1　圧力器の最適設計問題

- DE-SP: $(F, C_R, M) = (1.0, 0.0, 1)$
- DE: $(F, C_R) = (1.0, 0.8)$
- DE/nrand/1: $(F, C_R) = (1.0, 0.8)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (1.0, 0.8, 48, 49)$
- PSO-DE: $(F, C_R) = (1.0, 0.7)$

B.2.2　減速器の最適設計問題

- DE-SP: $(F, C_R, M) = (0.5, 0.4, 1)$
- DE: $(F, C_R) = (0.5, 0.4)$
- DE/nrand/1: $(F, C_R) = (0.5, 0.4)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (0.5, 0.5, 48, 49)$
- PSO-DE: $(F, C_R) = (1.0, 0.3)$

B.2.3　溶接梁の最適設計問題

- DE-SP: $(F, C_R, M) = (0.8, 0.6, 1)$
- DE: $(F, C_R) = (0.5, 0.5)$
- DE/nrand/1: $(F, C_R) = (0.5, 0.5)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (0.6, 0.5, 48, 49)$
- PSO-DE: $(F, C_R) = (0.5, 0.6)$

B.2.4　二足ロボットの立位保持制御問題

- DE-SP: $(F, C_R, M) = (0.5, 0.1, 1)$
- DE: $(F, C_R) = (0.5, 0.1)$
- DE-SP($M = 0$): $(F, C_R) = (1, 0.5)$
- DE2: $(F, C_R, M) = (0.5, 0.1)$
- DE/nrand/1: $(F, C_R) = (0.5, 0.1)$
- DE/isolated/1: $(F, C_R, N_w, N_d) = (0.5, 0.1, 48, 49)$
- PSO-DE: $(F, C_R) = (0.9, 0.5)$

B.3　Lennard-Jones Clusters Optimization Problem の最適化実験

本節ではDEとDE-SPによるLJ-Problem最適化実験について簡潔に触れる．本問題は一試行に莫大な計算時間を要するため，資料として十分と考えられるほどのパラメータ試行錯誤が現実的に不可能である．したがって，本実験は本文資料ではなく予備資料と扱うこととし，ここで簡潔に触れるのみとする．

B.3.1　実験設定

最適化において，発見された最良解を比較する．比較手法は従来手法であるDEとし，また，Hoareらによる報告値 [74] とも比較する．実験時のパラメータは粒子数 $N_{\mathrm{particles}} = 3$，4，8，20それぞれの場合に実験を行い，個体数250，摂動回数上限2500000回，DE-SPのパラメータは $(F, C_R, M) = (0.7, 0, 10)$，DEのパラメータは $(F, C_R) = (0.2, 0.8)$ とした．

表 B.1: 最適化によって得られた Lennard-Jones ポテンシャルと Hoare による報告値

	$N_{\mathrm{particle}} = 3$	$N_{\mathrm{particle}} = 4$	$N_{\mathrm{particle}} = 8$	$N_{\mathrm{particle}} = 20$
DE-SP	-3	-6	-19.821489	-68.680297
DE	-3	-6	-19.821489	-76.756871
Hoare [74]	-	-	-19.821489	**-77.177043**

B.3.2　実験結果

実験の結果と Hoare による報告値を表 B.1 に示す．この結果から，両手法とも現在報告されている最適解は発見できず，DE-SP は DE よりも成績が悪いことが認められる．この原因として，Lennard-Jones Clusters Optimization Problem を解く際，格子状構造，あるいは多面体構造を仮定して粒子を配置することで最適解に近い状態を導き出しやすい，という性質が挙げられる [75]．DE は，図 2.2 で認められるように，現在の親個体の位置を通る座標軸に平行な直線上に子個体を生成しやすい性質を持っている．この性質により DE は子個体の多様性に比較的乏しく，NoisyFunction の最適化では性能を発揮しないが，この問題では格子状に準じて子個体が生成されることが有利に働くと考えられるため，DE の方がよい最適化性能を示したと考えられる．

付 録C

どの最適化手法を選ぶとよいか

本研究では，目的関数の微分が困難であるような場合や，制約条件を含む場合，また関数の形で書き下せない問題といった，最適化問題の中では比較的解の発見が困難である問題を扱っていた．しかし実際に最適化問題を解かなければならない状況はこれらの状況に限ったものではなく，より容易な場合もあればより困難な場合もある．本研究では主に直接探索法，特にDE類縁手法を扱っているが，対象とする目的関数によってはDE類縁以外の直接探索法や，直接探索法以外の探索手法を適用することをより先に検討した方がよい場合も考えられる．

本章では，著者が行った調査や実験の知見から，想定される幾つかの状況において最初に適用を検討するとよいと勧められる手法を述べる．すなわち，本章に述べられている内容は著者の私見に近く，明確な根拠が必ずしもないことを予め断っておく．

目的関数が線形計画問題に置き換えられる　この場合は**厳密に最適解が導出可能である**．したがって楕円体法 [76] やカーマーカー法 [77] の適用をまず検討するべきである．また，厳密に最適解が導出可能であるため，解の最適性を保証しない直接探索法を使うことはお勧めできない．

目的関数の微分が可能で最適解の存在する領域に目星が付きその領域の単峰性が期待できる　この場合は勾配法，特に最適勾配法や共役勾配法の適用を第一に検討する．実装の容易性を求めるのならば最急降下法を検討してもよいだろう．目星の付いた領域に定義域を限定し，手法を適用すればよい．

目的関数の微分は困難だが最適解の存在する領域に目星が付きその領域の単峰性が期待できる　この場合は山登り法，SA，本稿では取り上げなかったがMesh Adaptive Direct Search（MADS） [78] も有効であると考えられる．目星の付いた領域に定義域を限定し，手法を適用すればよい．

目的関数が微分不能で計算資源が少ない　この場合はNMSを最初に検討し，得られた解の質に満足が出来ないのならばDEの適用を検討する．

目的関数が微分不能で計算資源はある程度確保できる　まず,「ある程度」というのは多点探索手法を用いて対象の目的関数を探索する場合に，個体数を目的関数の次元数の10倍，最大摂動回数を1000000回として試行すると，遅くとも一ヶ月で計算が終了することを目安としている．

この場合はPSO，DE，DE-SPを始めに検討するとよいだろう．パラメータの設定が困難である場合はPSO-KS，SDE，そして本稿の提案手法であるSDE-SP-DRを検討するとよいだろう．

どうしても上手くいかない　目的関数が複雑過ぎる場合や目的関数の次元数が大きすぎるなどの理由で，望む質の解が発見できないことがままある．

目的関数が複雑であると想定される場合は常にRSの適用は視野に入れておくべきである．特に定義域が明確である場合はRSによる探索を望む質の解が求められるまで続けることが最も現実的な探索法であることがままある．定義域が明確でない場合はRandom Walkの適用を視野に入れる．ただしこの場合は可能な限り多くの個体数を用い，可能な限り多くの摂動回数を確保する必要がある．

付 録D

既発表文献

論文誌（査読有り）

[1] 岩井 亮，加藤 昇平: Differential Evolution on Scattered Parents による大域的単峰性に乏しい多峰性探索空間における最適化, 電気学会論文誌C（電子・情報・システム部門誌）, Vol. 133, No. 2, pp. 410-417, 2013.

[2] 岩井 亮，加藤 昇平: 探索の停滞に応じてパラメータを再設定する Differential Evolution on Scattered Parents, 情報処理学会論文誌, Vol. 56, No. 2, pp. 733-743, 2015.

国際会議（査読有り）

[1] Ryo Iwai and Shohei Kato: Optimization in Multi-Modal Continuous Space with Little Globally Convex using Differential Evolution on Scattered Parents, In *IEEE International Conference on Systems Man and Cybernetics*, pp. 2002-2007, 2012.

[2] Ryo Iwai and Shohei Kato: Differential Evolution on Scattered Parents with Resetting Parameters and Restarting Optimization when Optimization Stagnate, In *The Workshop on New Directions of Evolutionary Computation and Intelligent Systems at the IEEE 2015 International Congress on Evolutionary Computation*, 7 pages, 2015.

その他

[1] 岩井 亮，加藤 昇平: 二足ロボットの立位姿勢保持PI-D制御と Nelder-Mead Simplex 法による PID ゲインの準最適化，In 平成 *23* 年度電気関係学会東海支部連合大会，P5-3 (1 page)，2011.

[2] 岩井 亮，加藤 昇平: 大域的単峰性に乏しく多峰な探索空間に有効な Differential Evolution の一改良，In 第 *9* 回情報学ワークショップ，pp. 107-111，2011.

[3] 岩井 亮，加藤 昇平: 局所解からの離脱を目的とした Differential Evolution on Scattered Parent の提案，In 情報処理学会第 *74* 回全国大会，Vol. 74, No. 1, pp. 477-478，2012.

[4] 岩井 亮，加藤 昇平: DE-SP での最適化における適応的パラメータ調整，In 平成 *24* 年度電気関係学会東海支部連合大会，D3-5 (1 page)，2012.

[5] 岩井 亮，加藤 昇平: 探索停滞時にパラメータを再設定する Differential Evolution on Scattered Parents，In *Joint Agent Workshop and Symposium,* B5-1 (6 pages)，2012.

[6] 岩井 亮，加藤 昇平: 探索停滞判定時に探索を再初期化する Differential Evolution on Scattered Parents, 情報処理学会研究報告 *IPSJ SIG Technical Report (IPSJ-SIG-ICS)*, Vol. 2015-ICS-179, No. 17-4-17, 6 pages, 2015.

受賞

[1] 学生奨励賞, 局所解からの離脱を目的とした Differential Evolution on Scattered Parent の提案, 第 74 回情報処理学会全国大会, 情報処理学会, Mar. 8, 2012.

[2] 学生優秀論文賞, 探索停滞時にパラメータを再設定する Differential Evolution on Scattered Parents Joint Agent Workshop and Symposium, JAWS2012/iJAWS2012 委員会, Oct. 26, 2012.

[3] 学生論文奨励賞, 探索の停滞に応じてパラメータを再設定する Differential Evolution on Scattered Parents, 情報処理学会東海支部, May 18, 2015.

[4] 名古屋工業大学学生研究奨励 副学長賞, 名古屋工業大学, Mar. 2, 2016.