博士論文

関数最適化問題に対する適応型差分進化法の研究

田邊遼司

謝辞

本論文は東京大学大学院総合文化研究科広域科学専攻広域システム科学系博士課程在籍時の 3 年間の研究成果の一部をまとめたものです. この 3 年間, 適切かつ手厚いご指導をしていただきました, 福永 Alex 准教授に心より感謝いたします. いつまでも在籍していたいと思うほど充実した博士課程, 及び研究生活でした. 重ね重ね, 本当にお世話になりました.

本論文の副査をお引き受けいただきました, 池上高志教授, 伊庭斉志教授, 植田一博教授, 山口泰教授 (五十音順) に深く感謝をいたします. 数々の貴重なご意見を頂き, 本論文の完成度を高めることができました. 重ねて, 感謝いたします.

本論文の初稿を添削してくれた, 福永研究室博士課程の浅井政太郎君に感謝します. また, ミーティングにて建設的な意見をくれた, 福永研究室の皆様にも感謝します.

博士課程 2 年, 及び 3 年時に, 日本学術振興会に特別研究員 (DC2) として採用していただき, 奨励費などのご支援をいただきました. 厚くお礼申し上げます.

最後に, 経済と精神の両面で支援してくれる両親に感謝します.

目次

表目次

図目次

List of Algorithms

List of Acronyms

SA Simulated Annealing
EA Evolutionary Algorithm
GA Genetic Algorithm
GP Genetic Programming
ES Evolution Strategy
CMA-ES Covariance Matrix Adaptation Evolution Strategy
EP Evolutionary Pragramming
EDA Estimation of Distribution Algorithm
PSO Particle Swarm Optimization
ABC Artificial Bee Colony
SBX Simulated Binary Crossover
BLX Blend Crossover
PBX Parent Centric Blend Crossover
SPX Simplex Crossover
UNDX Unimodal Normal Distribution Crossover
PCX Parent Centric Crossover
MGG Minimal Generation Gap
G3 Generalized Generation Gap
JGG Just Generation Gap
DE Differential Evolution
jDE (Janez's) self-adaptive Differential Evolution
EPSDE Ensemble of Mutation and Crossover Strategies and Parameters in DE
JADE (Jingqiao and Arthur's) Adaptive Differential Evolution
MDE Modified Differential Evolution

SHADE Success-History based Adaptive Differential Evolution
DEMO Differential Evolution for Multiobjective Optimization
SEC Shuffled Exponential Crossover
BBOB Black Box Optimization Benchmarking
COCO COmparing Continuous Optimisers
CEC IEEE Congress on Evlutionary Computation

List of Symbols

$f : \mathbb{R}^D \to \mathbb{R}$	目的関数
$H_{f,g} : \mathbb{R}^D \to \mathbb{R}$	f, g から構成されるハイブリッド関数
D	次元数 (決定変数の数)
N	集団数 (個体の数)
t	世代数 (反復回数)
$\boldsymbol{x}^{i,t} = (x_1^{i,t}, ..., x_D^{i,t})^T$	世代 t における $i \in \{1, ..., N\}$ 番目の個体
$\boldsymbol{v}^{i,t} = (v_1^{i,t}, ..., v_D^{i,t})^T$	世代 t における $i \in \{1, ..., N\}$ 番目の変異個体
$\boldsymbol{u}^{i,t} = (u_1^{i,t}, ..., u_D^{i,t})^T$	世代 t における $i \in \{1, ..., N\}$ 番目の子個体
F	DE におけるスケール係数であり, $F \in (0, 1]$
C	DE における交叉率であり, $C \in (0, 1]$
τ_F	jDE の F に関する適応メタパラメータであり, $\tau_F \in (0, 1)$
τ_C	jDE の C に関する適応メタパラメータであり, $\tau_C \in (0, 1)$
μ_F	JADE, 及び MDE の F に関する適応メタパラメータであり, $\mu_F \in (0, 1]$
μ_C	JADE, 及び MDE の C に関する適応メタパラメータであり, $\mu_C \in [0, 1]$
$\boldsymbol{S}^F, \boldsymbol{S}^C$	JADE, MDE, 及び SHADE において使用される, 世代ごとの成功した F と C の集合
H	SHADE におけるメモリ $\boldsymbol{M}^F, \boldsymbol{M}^C$ の大きさ
$\boldsymbol{M}^F = (M_1^F, ..., M_H^F)$	SHADE の F に関する適応メタパラメータであり, $M_j^F \in (0, 1]$
$\boldsymbol{M}^C = (M_1^C, ..., M_H^C)$	SHADE の C に関する適応メタパラメータであり, $M_j^C \in [0, 1]$
$o_{inc}(n_t), o_{dec}(n_t)$	一次関数を用いた oracle 関数
$o_{sin}(n_t)$	sin 関数を用いた oracle 関数
$o_{rw}(n_t)$	ランダムウォークを用いた oracle 関数

第 1 章

はじめに

1.1 研究の背景

関数最適化問題は, 目的関数 $f : \mathbb{R}^D \to \mathbb{R}$ が与えられた時に目的関数値 $f(\boldsymbol{x})$ を最小化する D 次元の実数値ベクトル $\boldsymbol{x} = (x_1, ..., x_D)^T$ を求める, 工学分野において一般的かつ重要な最適化問題である. 最適解が解析的に求まる関数最適化問題インスタンスは少ないため, これまでに様々な最適化手法が提案されている. しかし, 目的関数が非凸関数であった場合, 最適解を求めることは困難である.

目的関数 f が凸関数かつ微分可能である場合は, ニュートン法を始めとする f の勾配情報を利用する探索手法が有用である. しかし, f が微分不可能である場合は適用することが困難であり, また非凸関数においては大域的最適解に収束する保証は無い. 1960 年代に提案された Nelder-Mead 法 [167] や Hooke-Jeeves 法 [103] などの解 $\boldsymbol{x}$ の目的関数値 $f(\boldsymbol{x})$ のみを利用して探索を行う決定的直接探索法は, 現実的な計算資源 (実行時間や解評価回数) においてユーザが許容できる精度の解を求めることを目的とする. これらの手法は比較的低次元な問題においては 2016 年現在においても有用であるものの, 次元数の増加に伴い探索性能が比較的劣るようになる [92]. また, 多峰性関数において局所的最適解に陥らずに大域的最適解を求める探索能力は高いとは言えない.

ランダムサーチ [31] や擬似焼きなまし法 [121] といった目的関数値 $f(\boldsymbol{x})$ のみを利用して探索を行う確率的最適化手法は, 対象問題が多峰性関数であっても計算資源を際限なく与えた場合に限り大域的最適解への収束が保証されている. また, これらの手法は目的関数 f の勾配情報を利用しないため, 対象問題の目的関数が微分不可能であったとしても適用可能である. しかし, 擬似焼きなまし法においては大域的最適解への収束が保証されるためには, 改悪解への移動受理確率を制御する温度パラメータの冷却スケジュールを非常に遅く減少させる必要があるため, 多くの場合において最適性が保証される条件で探索を行うことは現実的でない.

進化アルゴリズム (Evolutionary Algorithm, EA) は複数の解を用いて探索を行う近似的最適化アルゴリズムの総称である. 直接探索法と同様に, EA は目的関数値 $f(\boldsymbol{x})$ のみを利用して探索を行う. 関数最適化問題に対する EA は, Genetic Algorithm (GA) [102], Evolution Strategy (ES) [15], Evolutionary Programming (EP) [260], Covariance Matrix

Adaptation Evolution Strategy (CMA-ES) [96], Estimation of Distribution Algorithm (EDA) [201, 206], Particle Swarm Optimization (PSO) [120], Ant Colony Optimization [50, 214], Artificial Bee Colony (ABC) [116, 117], Differential Evolution [220, 221] など, これまでに様々な枠組みが提案されている. また, 複数の解を保持して探索を行うので, Nelder-Mead 法などの局所探索法と比べて局所解に探索が陥る可能性が低いと期待されている. EA は 1960 年代から研究されており [65, 66], 1960 年代に Fogel により EP [68], 及び Rechenberg と Schwefel により ES [15] が, 1970 年代には Holland によって GA [102] がそれぞれ提唱されている. その後, 計算機性能の向上に伴い, 1990 年代ごろからより活発に EA の研究が始められ, 2016 年現在までに数多くの EA の提案, 及びその実問題への適用事例が報告されている.

EA の実用上の問題点として, ある最適化問題における EA の探索性能が自身の制御パラメータの設定に大きく依存することがあげられる [54, 146, 56, 119]. 単純な GA を例にとっても, 多峰性の問題では局所解に探索が誤って収束されないように集団数を十分大きく, また多様性を維持するために突然変異率を高く設定する必要がある. 一方, 多峰性問題に適したパラメータ設定, すなわち多様性維持を重視した設定を使用した場合, 単峰性の問題では探索の収束速度が遅いため, 計算資源を多く費やしてしまうこととなる. そのため, 小さな集団数, 及び低い突然変異率が単峰性の問題では好ましい. このように, 対象問題がどのような問題性質を有するのかによって, 適切な EA のパラメータ設定は大きく変化する. そのため, EA を最適化ツールとして使用するユーザは, 解きたい対象問題ごとに EA のパラメータ設定を適切に調整する必要がある.

この問題に対して, EA の探索中に対象問題, 及び探索状況に適したパラメータ設定を求めるパラメータ制御がこれまでに研究されている [54, 119]. EA におけるパラメータ制御法の枠組みは, 次の 3 つに分類される [54]:

1. 決定的パラメータ制御 (deterministic parameter control)
2. 適応的パラメータ制御 (adaptive parameter control)
3. 自己適応的パラメータ制御 (self-adaptive parameter control)

より詳しくは 4 章で説明するが, パラメータ制御法を用いた EA はユーザによるパラメータ設定の調整を必要とせずとも, ある程度良好な探索性能が期待される.

Differential Evolution (DE) [220, 221, 188] は 1995 年に Storn と Price によって提案された, 主に関数最適化問題を対象とした EA である. 多くの EA とは異なり, DE は生物進化, 及び特定の生物の振る舞いからではなく, Nelder-Mead 法の反射操作に着想を得た手法である. DE は比較的単純な枠組みでありながら, 他の EA と比べ比較的良好な性能を示すことが報告されている [43]. 例えば, 最も基本的な DE アルゴリズム [221] は, 1996 年に開催された IEEE の進化計算系の会議である IEEE Congress on Evolutionary Computation (CEC) の境界制約付き関数最適化問題のコンペティションでは 3 位 [19], 1997 年では 1 位 [187], 2005

年では 10 次元の部門では 2 位を占めている[*1]. その他にも, DE アルゴリズムを改良した手法が, 2006 年と 2010 年の制約付き最適化問題において 1 位[*2], 2009 年の多目的最適化問題では 1 位を占めている[*3]. このような背景から, 近年では DE についての研究数が増加している.

先に述べたとおり, 一般的に EA の探索性能は使用するパラメータ設定に大きく依存することが知られている. この一般認識に反して DE が提案された当初は, DE の探索性能は制御パラメータの設定に対してロバストであると報告されている [220, 221]. しかし, 後年の研究結果から, 多くの EA と同様に DE の探索性能も使用するパラメータ設定に大きく依存することが判明した [71, 196, 274, 27]. ここで, 基本的な DE の制御パラメータは集団数 N, スケール係数 F, 交叉率 C である. F は差分突然変異の大きさを調整し, C は親個体から子個体へと受け継がれる決定変数の数を制御する.

多くの実問題においては対象問題の情報が事前に知ることのできない black-box optimization 環境であるため, 対象問題ごとにこれらのパラメータ設定を調整する必要がある. しかし, 実問題においてはパラメータチューニングが困難な状況が多々ある. 例えば, 実問題においては目的関数 f による解評価にかかる計算コストが高価である場合があり [112], このような実問題にて幾度も DE を実行することは現実的ではない. また, 適切なパラメータ設定を調整するためには, DE に関する知識が必要である. そのため, 最適化手法に関する知識が乏しいが DE を最適化ツールとして利用することのみが目的のユーザに, 大きな負担をかけることになる. 以上に述べたように, パラメータ設定問題は DE の実用化に当たり大きな課題となっている.

このような背景から, 探索中に自動的にパラメータ設定が調整されユーザによるパラメータチューニングが不要な DE におけるパラメータ制御法が, 2005 年ごろから徐々に研究され始めた. 先に述べた EA における分類と同様に, DE におけるパラメータ制御法は, (1) 決定的パラメータ制御, (2) 適応的パラメータ制御, (3) 自己適応的パラメータ制御に分けられる. その中でも, 現在の探索状況の情報を基に使用するパラメータ設定を調整する, (2) 適応的パラメータ制御に関する研究が近年増加しており, DE におけるパラメータ制御法の主流となっている [43]. また, jDE [27], EPSDE [157], JADE [267], MDE [110] といったこれまでに優れた適応 DE がいくつか提案されている.

1.2 本論文の目的と内容

多くの優れた適応 DE が提案されている一方, その適応手法に関する知見は乏しい. ほぼ全ての先行研究 [27, 189, 157, 267, 110] は新たな適応 DE を提案し, ベンチマーク集合にてその探索性能を評価するに留まっており, なぜ提案する適応 DE が既存手法よりも良い性能を示すのかといった考察や解析はしていない. そのため, 異なるパラメータ適応手法間でなぜ探索

[*1] https://www.lri.fr/~hansen/cec2005.html (2016 年 1 月 25 日確認)
[*2] http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-06/CEC06.htm, http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC10-Const/CEC10-Const.htm (2016 年 1 月 25 日確認)
[*3] http://dces.essex.ac.uk/staff/zhang/moeacompetition09.htm (2016 年 1 月 25 日確認)

性能に優劣が発生するのか, 優れた適応手法を設計するためには何が重要であるのかといったことは不明である.

適応 DE の適応手法の振る舞いを調査した研究は, 少なからず行われている [27, 30, 189, 267, 157, 36]. しかし, 先行研究の多くは適応メタパラメータ (F, C の生成に使用される, 適応手法により直接調整されるパラメータ), 又は生成された F, C の値を探索の経過ごとに図示し, 視覚に基づく定性的な議論にとどまっている. このような視覚に基づく解析法は, ある問題インスタンスにおいて特定の適応手法がどのように振る舞うのかを大まかに把握するには有用である. しかし, それ以上の議論, すなわち特定の適応手法が優れた性能を示す理由を解明するためには, あまり有用ではない. また, Karafotias らに指摘されているように DE だけではなく EA 全体においても, あるパラメータ適応手法が有用な理由を明らかにした研究は少ない [119].

また, CMA-ES [96] にリスタート戦略を導入した restart CMA-ES [9, 88, 149] が, 現在の black-box optimization 環境における関数最適化問題に対する state-of-the-art な手法であり [92], 適応 DE の探索性能はそれに及ばないとされている [43]. そのため, 適応 DE は比較的単純なアルゴリズムであり良好な探索性能を示すものの, EA を実問題における最適化ツールとして利用するユーザの最初の選択肢としては考えづらい.

上記に述べた適応 DE における 2 つの問題点に対して, 本論文の目的は次のとおりである:

1. 先行研究では適応 DE の適応手法を定性的にしか解析できていなかったのに対して, 本論文では定量的に解析する枠組みを確立する. この枠組みを用いて適応手法を解析することで, 従来の定性的な解析方法では得られなかった, どの適応手法が優れた適応性能を有するのか, 高性能な適応手法を設計するためには何が求められているのか, 及び特定の問題クラスにおける適応 DE の探索失敗現象といった事項を解明する.
2. 上記により得られた適応手法に関する知見を元に, 現在の state-of-the-art な EA である restart CMA-ES と同等以上に効率的な適応 DE を設計する. これにより, 適応 DE が実問題における最適化ツールとして, より多くのユーザに使用されることを目指す.

(1) の目的のために, 適応 DE のパラメータ適応能力の解析のための, 理想化されたパラメータ適応の軌跡である oracle パラメータを用いた新たなシミュレーション法を提案する (5 章参照). 本シミュレーション法ではパラメータ値を独立して評価することが可能であり, これまで難しかったパラメータ適応手法の適応能力についての議論を可能とする. また, 異なる適応過程におけるパラメータ系列間の距離の概念を新たに導入し, ハイブリッド関数 [231, 132, 136] における適応 DE のパラメータ適応の振る舞いを解析する (7 章参照). これにより, ハイブリッド関数にて適応手法がパラメータ適応に失敗したために, 適応 DE の探索が失敗する現象を明らかにし, 適応 DE の問題点を指摘する.

(2) の目的のために, DE における交叉オペレータの分析と評価を行う (8 章参照). その結果から, 近年良好な性能を有することが報告されている DE の exponential 交叉は, 誤った性能評価がされている可能性が高いことを指摘する. 各章にて得られた知見を元に, 新しい効率的な適応手法を提案する (6 章参照). 目的関数値 $f(\boldsymbol{x})$ のみを利用する伝統的な探索手法, 及び

restart CMA-ES と比較し, より探索性能が優れていることを示す.

1.3 本論文の構成

本論文の構成は次のとおりである. 2 章, 3 章, 及び 4 章は本論文の研究背景を, 5 章, 6 章, 7 章, 8 章は本論文の成果を報告している. また, 付録 A では関数最適化問題に対する伝統的な探索手法, 付録 B 章では Zaharie によるフラットランドスケープにおける DE の収束性解析, 付録 C 章では 5 章と 6 章にてアルゴリズムの探索性能評価のために使用した BBOB benchmarks について述べている. 付録 D 章では著者の発表文献をまとめている.

2 章. Black-box optimization 環境における関数最適化問題とその探索手法

本論文が対象とする, black-box optimization 環境における関数最適化問題について述べる. また, 本問題に対する探索手法として直接探索法, 及び EA について説明する. 最後に, 本問題に対する探索手法の性能評価方法について述べる.

3 章. Differential Evolution

本論文にて扱う, EA の一手法である Differential Evolution (DE) [220, 221] について述べる. 始めに基本的な DE アルゴリズムについて説明し, 次に DE の突然変異戦略, 交叉手法や世代交代モデルといった各オペレータの特徴, 及び性質について説明する. 最後に, 探索空間の変換操作に対する不変性や各交叉手法の性質といった DE の理論的な性質について述べる.

4 章. EA, 及び DE におけるパラメータ設定問題とパラメータ制御

本章では, 本論文の対象である DE におけるパラメータ制御法について主に述べる. 始めに, EA におけるパラメータ設定問題について説明する. 次に, パラメータ設定問題の解決方法であるパラメータチューニングとパラメータ制御について, それぞれ説明する. 特に後者については, (1) 決定的パラメータ制御, (2) 適応的パラメータ制御, (3) 自己適応的パラメータ制御の 3 つのアプローチについて詳細に述べる. その後, DE におけるパラメータ設定問題, 及びパラメータ制御に関する先行研究を説明する.

5 章. 適応 DE の適応手法の解析

適応 DE の適応手法を (i) ベンチマーク問題集における性能評価, (ii) 本章にて提案する新たなシミュレーション法により解析する.

(i) の目的は「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しを評価し, どの適応 DE の適応手法が一般的に, 又は特定のオペレータを用いた場合に優れているのかを明らかにすることである. 8 種類の突然変異戦略と 2 種類の交叉手法の組み合わせ, 計 16 種類のオペレータにおける適応手法の探索性能を, BBOB benchmarks [93] にて評価する. その結果から, 適応手法間の優劣関係は使用するオペレータに大きく依存し, あらゆるオペレータにおいて優れた性能を示す適応手法は無いことが判明した.

(ii) の目的は「適応手法」の良し悪しを「遺伝的オペレータ」とは独立して評価し, どの適応手法が優れた適応性能を有するのか, 高性能な適応手法を設計するためには何が

求められているのかを明らかにすることである．「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しは (i) の実験にて比較的容易に検証可能であるが，「適応手法」のみを独立して評価することはいくつかの理由から困難であり，著者の知る限りこれまで行われていない．その最も大きな障害は，適応手法が生成したパラメータ値自体の良し悪しを判別することが困難な点である．この問題点を解消するために，理想的なパラメータ適応の軌跡の代理である oracle パラメータを用いた新たなシミュレーション法を提案する．提案シミュレーション法ではパラメータ値を独立して評価することが可能であり，これまで困難であったパラメータ適応手法の適応能力についての議論を初めて可能とする．本シミュレーション法により従来の適応 DE のパラメータ適応能力を解析することで，既存の適応手法には問題点があることを明らかにする．

6 章．Success-History based Adaptive Differential Evolution

5 章で明らかになった既存の適応手法の問題点を考慮した，DE のための新たな適応手法である Success-History based Adaptive DE (SHADE) を提案する．SHADE では探索中に得られた対象問題に適したパラメータ設定をメモリに保存し，このメモリ内の要素を用いて新たにパラメータを生成することで，制御パラメータの自動調整を行う．

始めに，5 章にて提案する oracle パラメータを用いたシミュレーション法における SHADE のパラメータ適応性能を計測する．その結果から，既存の適応 DE が適応に失敗する oracle パラメータが世代ごとに急激に変化するモデルにおいても，SHADE は良好な適応性能を有することを示す．

次に，ベンチマーク問題集において既存の適応 DE と比較することで，SHADE の有用性を示す．さらに，SHADE は有限差分法を用いた準ニュートン法や Nelder-Mead 法 [167] といった目的関数の勾配情報を必要としない伝統的な探索手法よりも優れており，近年の black-box optimization 環境における state-of-the-art な EA である BIPOP-CMA-ES [88] に匹敵する性能を有することを示す．

7 章．ハイブリッド関数における適応 DE のパラメータ適応の振る舞いの解析

6 章では SHADE を提案し，既存手法と比較することでその有用性を実証するが，SHADE があらゆる問題に対しても完璧に対応可能な，万能な適応手法であると主張しているわけではない．本章では SHADE を含む，適応 DE の適応手法がハイブリッド関数 [231, 132, 136] においてパラメータ適応に失敗する現象を扱う．

具体的には，ハイブリッド関数における，SHADE を含む適応 DE の (1) 探索性能，及び (2) パラメータ適応の挙動を解析する．(1) については，適応 DE をハイブリッド関数に適用した結果，問題性質が互いに異なるベンチマーク関数から構成されたハイブリッド関数は，適応 DE にとって困難な問題であることがわかった．この理由を明らかにするため，(2) にて適応 DE のパラメータ適応の振る舞いを解析する．まず，この目的のために，適応過程におけるパラメータ系列間の距離に基づくパラメータ適応の解析方法を新たに提案する．そして，提案手法を用いてハイブリッド関数における適応手法のパラメータ適応の振る舞いを解析し，適応 DE の探索が失敗する現象を明らかにする．

8 章．DE における交叉オペレータの分析と評価

5 章と 7 章では, 適応 DE の適応手法の解析を行う. また, 6 章では新たな適応手法である SHADE を提案する. 以上の章は適応手法を興味の対象としていたのに対して, 本章では DE における交叉オペレータの分析と評価を行う.

本章ではまず, DE の代表的な交叉手法である exponential 交叉は, Rosenbrock 関数 $f(\boldsymbol{x}) = \sum_{i=1}^{D-1}\left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$ などの人工的に作成したベンチマーク関数に存在する, 依存関係にある決定変数が変数番号上において隣接しているという非現実的な性質を利用可能である点を指摘する. そして, この非現実的な性質を解消した場合, exponential 交叉の探索性能は binomial 交叉と比べ劣るようになることを示す. また, バイアスを持たない Shuffled Exponential Crossover (SEC) [188] を含む大規模な交叉オペレータの実験的評価を行い, (1) 多くのベンチマーク関数において binomial 交叉が最も良好な性能を有し, (2) SEC は exponential 交叉と比べて同等以上の性能を持つことを示す.

9 章. おわりに

本論文にて得られた研究成果をまとめる. また, 今後の課題についても言及する.

第 2 章

Black-box optimization 環境における関数最適化問題とその探索手法

本章では, 本論文にて扱う関数最適化問題, 及び本問題に対する探索手法について述べる. 始めに, 2.1.1 節にて関数最適化問題の定義について説明する. ここでは, 本論文の対象問題である black-box optimization 問題についても述べる. 次に, 2.2 節では, 関数最適化問題の探索手法が直面する問題性質について説明する. 2.3 節では, black-box optimization 環境における関数最適化問題に対する伝統的な探索手法について簡単に述べる. なお, より詳細な説明, 及び目的関数の微分情報を用いた探索手法については, 付録 A 章を参照されたい. そして, 2.4 節にて進化アルゴリズムについて述べ, 最後に 2.5 節にて black-box optimization 問題に対する探索手法の性能評価方法について説明する.

2.1 関数最適化問題

本節では, 本論文にて扱う関数最適化問題の定義について述べる. 始めに 2.1.1 節にて一般的な関数最適化問題を定式化し, 2.1.2 節にて線形計画問題, 及び非線形計画問題について述べる. 次に 2.1.3 節にて境界制約付き関数最適化問題について説明する. 最後に, 2.1.4 節にて本論文が対象とする black-box optimization 環境における境界制約付き関数最適化問題について述べる. なお, 2.1.1, 2.1.2, 2.1.3 節の各定義や数式表記などは [277, 279, 278] を参考にしている.

2.1.1 関数最適化問題の定式化

目的関数 $f : \mathbb{R}^D \to \mathbb{R}$, 及び実行可能領域 $\boldsymbol{S} \subseteq \mathbb{R}^D$ が与えられた時に, 目的関数 f を最小化する解 $\boldsymbol{x} = (x_1, ..., x_D)^T$ (D 個の決定変数から成る実数値ベクトル) を求める問題を, 関数最適化問題と呼ぶ. 一般的な関数最適化問題は, 以下のように記述できる:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \boldsymbol{x} \in \boldsymbol{S} \tag{2.1}$$

式 (2.1) は最小化問題に限定しているが, 最大化問題 $\max_{\boldsymbol{x}} f(\boldsymbol{x})$ を扱う場合においても, $\max_{\boldsymbol{x}} f(\boldsymbol{x}) = -\left(\min_{\boldsymbol{x}} -f(\boldsymbol{x})\right)$ と変換可能であるため, 一般性を失わない. $\boldsymbol{x} \in \boldsymbol{S}$ ならば $\boldsymbol{x}$ を実行可能解と呼び, $\boldsymbol{x} \notin \boldsymbol{S}$ であれば $\boldsymbol{x}$ を実行不可能解と呼ぶ. 以下, 本論文では特に断りが無い限り, 簡便のため実行可能解を単に解と呼ぶ.

次式を満たす解 $\boldsymbol{x}^*$, つまり目的関数 f を最小化する解 $\boldsymbol{x}^*$ を大域的最適解と呼ぶ:

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \boldsymbol{S} \tag{2.2}$$

ここで, 最適解の目的関数値 $f(\boldsymbol{x}^*)$ を最適値と呼ぶ. 式 (2.1) にて定めた関数最適化問題は, 最適解 $\boldsymbol{x}^*$ を実行可能領域 $\boldsymbol{S}$ の中から求める問題と言い換えることができる. また, ある解 $\boldsymbol{x}^l \in \boldsymbol{S}$ の近傍 $\boldsymbol{S}^n(\boldsymbol{x}^l)$ について次式が成り立つ時, $\boldsymbol{x}^l$ を局所最適解と呼ぶ:

$$f(\boldsymbol{x}^l) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \boldsymbol{S} \cap \boldsymbol{S}^n(\boldsymbol{x}^l) \tag{2.3}$$

ここで, $\boldsymbol{S}^n(\boldsymbol{x}^l) = \{\boldsymbol{x} | \epsilon > \|\boldsymbol{x}^l - \boldsymbol{x}\|\}$ であり, ϵ は任意の正数, $\|\boldsymbol{x}\|$ は $\boldsymbol{x}$ のユークリッドノルムを表す. 全ての局所最適解 $\boldsymbol{x}^l$ において, 目的関数値が最小となる局所最適解解 $\boldsymbol{x}^l$ を大域的最適解 $\boldsymbol{x}^*$ と解釈することができる. 以下では, 簡便のため大域的最適解 $\boldsymbol{x}^*$ を単に最適解, 局所最適解 $\boldsymbol{x}^l$ を単に局所解と呼ぶ.

2.1.2 線形計画問題と非線形計画問題

M 個の不等式制約条件 $g_1, ..., g_M$, 及び L 個の等式制約条件 $h_1, ..., h_L$ が与えられた場合, 式 (2.1) の実行可能領域 $\boldsymbol{S}$ は, $\boldsymbol{S} = \{\boldsymbol{x} \mid \forall i : g_i(\boldsymbol{x}) \leq 0, \forall j : h_j(\boldsymbol{x}) = 0\}$ と表せられる. この時, 目的関数 f が $\boldsymbol{x}$ の一次関数であり, M 個の不等式制約条件, 及び L 個の等式制約条件が $\boldsymbol{x}$ の一次式で与えられる問題を線形計画問題と呼ぶ. 単体法が 1947 年に提案されて以来, 線形計画問題はオペレーションズリサーチの分野において長年研究されており, 効率的に最適解を求める手法が開発されている.

一方, 目的関数 f, 及び各制約条件が非線形である場合を非線形計画問題と呼ぶ. 非線形計画問題の中でも, 対象問題が凸関数[*1]かつ二階微分可能であれば, 停留点と最適解が一致するため, 勾配ベクトルとヘッセ行列を求めることにより, 比較的容易に最適解を求めることが可能である. しかし, 非線形計画問題において対象問題が非凸関数であった場合, 最適解を求めることは困難である.

2.1.3 境界制約付き最適化問題

解の各決定変数 x_j, $j \in \{1, ..., D\}$ ごとに上限値 $x_j^{\max}$, 及び下限値 $x_j^{\min}$ が以下のように与えらえれる問題を, 境界制約付き最適化問題と呼ぶ:

$$\boldsymbol{S} = \{\boldsymbol{x} \mid x_j^{\min} \leq x_j \leq x_j^{\max}, j = 1, ..., D\} \tag{2.4}$$

[*1] 任意の 2 つの解 $\boldsymbol{x}^1, \boldsymbol{x}^2$, 及び任意の正数 $\alpha, \beta > 0$ について, $f\left(\frac{\alpha\boldsymbol{x}^1 + \beta\boldsymbol{x}^2}{\alpha+\beta}\right) \leq \frac{\alpha f(\boldsymbol{x}^1) + \beta f(\boldsymbol{x}^2)}{\alpha+\beta}$ が成り立つ時, 目的関数 f は凸関数であると呼ぶ. 凸関数においては局所解 $\boldsymbol{x}^l$ が 1 つのみ存在するため, 局所解 $\boldsymbol{x}^l$ と最適解 $\boldsymbol{x}^*$ が一致する.

境界制約は多くの実問題において与えられる一般的な制約であり, 境界制約付き最適化問題は標準的な関数最適化問題である. 実行不可能解 $\boldsymbol{x} \notin \boldsymbol{S}$ については $f(\boldsymbol{x}) = \infty$ とすれば, 式 (2.4) は $\boldsymbol{S} \in \mathbb{R}^D$ として与えられる無制約最適化問題に変換可能である. また, ある決定変数 x_j が $[x_j^{\min}, x_j^{\max}]$ の範囲外の値をとった場合は, 超えた方の境界値で置き換える, すなわち $x_j = x_j^{\min}$, または $x_j = x_j^{\max}$ などとすれば, 容易に実行不可能解を実行可能解に修正できる.

境界制約付き最適化問題において, 目的関数 f が局所解 $\boldsymbol{x}^l$ で微分可能であり, 式 (2.3) が成り立つのであれば, 次式が成立する:

$$\nabla f(\boldsymbol{x}) = \boldsymbol{0} \tag{2.5}$$

ここで, $\nabla f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, ..., \frac{\partial f(\boldsymbol{x})}{\partial x_D}\right)^T$ である. また, 式 (2.5) を一次の最適性条件と呼び, 局所解であればこの条件を満たす. しかし, 停留点 (鞍点) においても式 (2.5) が成り立つことから, 目的関数 f が凸関数である仮定がない場合は十分条件ではない. 目的関数 f が $\boldsymbol{x}^l$ で二階微分可能であり, 式 (2.3) が成り立つのであれば, 式 (2.5), 及びヘッセ行列 $\nabla^2 f(\boldsymbol{x})$ [*2] が半正値対称行列[*3]となる. これを 2 次の最適性条件と呼ぶ.

2.1.4　本論文が対象とする black-box optimization 問題

本論文では特に断りの無い限り, 2.1.3 節にて述べた境界制約付き最適化問題に加え, 式 (2.1) にて目的関数 f が明示的に与えられず, 目的関数値 $f(\boldsymbol{x})$ しか利用できない black-box optimization 環境 [88] を扱う. Black-box optimization 環境は多くの実問題において現れうる. 例えば, ロボットにおけるマルチセンサの統合処理 [115], 翼形状設計 [174], 車体設計 [129] など計算機シミュレーションにより解 $\boldsymbol{x}$ の目的関数値 $f(\boldsymbol{x})$ を計算する場合, f は式として与えられない.

Black-box optimization 環境では目的関数 f の勾配情報, 及び関数最適化問題を解くにあたって次節で説明する重要な問題性質を, 事前に利用, 及び知ることはできない. Fitness Distance Correlation (FDC) [113] や Dispersion Metric [152] といった地形解析法を使用することで, 目的関数値 $f(\boldsymbol{x})$ から対象問題がどのような問題性質を有するのかを, ある程度推測することができる. しかし, 地形解析法は多くの解評価を必要とするため, 解評価のための計算コストが高価であり計算資源を潤沢に費やせない場合は, 現実的ではない.

Black-box optimization 環境では勾配情報を利用できないため, A.1 節にて述べる最急降下法やニュートン法は適用不可能である. 伝統的な探索手法には, A.1.3 節にて説明する

[*2] 目的関数 f の二階偏微分係数を要素とするヘッセ行列 $\nabla^2 f(\boldsymbol{x})$ は, 次式のように与えられる:

$$\nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_D \partial x_1} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_D \partial x_D} \end{pmatrix} \tag{2.6}$$

なお, $\nabla^2 f(\boldsymbol{x})$ は対称行列である.

[*3] 全ての固有値が正, 又は 0 の対称行列を半正値対称行列と呼ぶ. なお, 行列 $\boldsymbol{A}$ が半正値対称行列であれば, 任意の $\boldsymbol{x}$ に対して $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \geq 0$ となる.

目的関数値空間において推定した勾配ベクトルを用いた準ニュートン法, 2.3 節にて述べる Nelder-Mead 法 [167], Hooke-Jeeves 法 [103], Simulated Annealing [121] などがあげられる.

2.2 関数最適化問題の探索手法が直面する問題性質

本章では, 2.1.4 節にて述べた black-box optimization 環境における関数最適化問題に取り組むにあたっての重要な問題性質である, 次元数, 単峰性/多峰性, 大域的単峰性/大域的多峰性, 変数分離化/変数分離不可, 悪スケール性について順次述べる.

2.2.1 次元数

対象問題の次元数 D が高くなるほど, 実行可能領域 $\boldsymbol{S} = \prod_{j=1}^{D} S_j$ が指数的に拡大されるため, 探索アルゴリズムによる探索が困難となる. そのため, 1, 2 次元と低次元な問題では有用な手法が, 必ずしも 10 ,20 次元といった問題においても優れているとは限らない. 例えば, 1965 年に提案された Nelder-Mead 法 [167] は, 対象問題が 2 次元であれば近年開発された state-of-the-art な探索手法に匹敵する性能を示すが, 3 次元以上では他手法と比べて劣る傾向が見られる [92].

また, black-box optimization の文献では $100 \sim 1,000$ 次元規模の問題を特別に高次元最適化問題と呼び, 他の問題と区別して研究に取り組まれている [100, 132]. また, 高次元最適化問題においてはアルゴリズムの計算量が膨大となるため, 準ニュートン法や CMA-ES [96] などのいくつかの手法は適用するのが困難となり, 計算量を減らすような何かしらの工夫が求められる [142, 199].

2.2.2 単峰性, 及び多峰性

探索空間中に局所解 $\boldsymbol{x}^l$ が一つしか存在せず, $\boldsymbol{x}^l$ と最適解 $\boldsymbol{x}^*$ が一致する性質を単峰性と呼ぶ. 反対に, 局所解が 2 つ以上存在する性質を多峰性と呼ぶ. 多峰性関数においても, 局所解の数や深さに応じて, 定性的に「弱い多峰性」や「強い多峰性」などと呼ぶことがある.

局所解が多数存在する多峰性関数においては, 最急降下法などの局所探索手法は局所解に誤って収束してしまい, 最適解を求めるのに失敗する可能性が高い. そのため, 多峰性の性質を持つ多峰性関数は, 単峰性の性質を有する単峰性関数よりも難しい問題クラスであると考えられている. この認識は, 関数最適化問題のみではなく組合せ最適化問題においても共通である.

2.2.3 大域的単峰性, 及び大域的多峰性

本節では, 2.2.2 節で述べた単峰性, 及び多峰性といった性質とは異なり, 大域的に見た場合における探索空間の構造について述べる. 探索空間を巨視的に見たとき, 大域的には単峰性に見える問題性質を大域的単峰性と呼ぶ [21]. ここで, 2.2.2 節で述べた単峰性とは異なり, 微視的においては局所解が無数に存在する多峰性関数であっても構わない. この問題クラスにおい

ては, 最適解に近づくにつれほぼ単調に局所解の質が向上していく, つまり目的関数値が改善されていく性質が見られる.

一方, 巨視的に多峰性である問題性質を, 大域的多峰性と呼ぶ [130]. ここで, 1990 年代においては騙し構造と呼ばれていたが, 近年の関数最適化問題に対する EA の研究では大域的多峰性と呼ばれることが多い. 大域的多峰性を持つ関数においては, 最適解から離れた領域に有望な局所解が存在する. 一般的に, 大域的多峰性を有する問題に探索手法を適用した場合, 最適解を含まない方の谷に探索が収束してしまうため, 難しい問題クラスであると考えられている [108, 153].

2.2.4　変数分離化, 及び変数分離不可

決定変数を各次元 $j \in \{1, ..., D\}$ ごとに分離し, $\sum_{j=1}^{D} f_j(x_j)$ のように目的関数を分解することで, D 次元の最適化問題を D 個の 1 次元の最適化問題に分解可能な性質を変数分離化と呼ぶ. この性質を明示的に利用する手法を用いた場合, 2.2.1 節にて述べた次元数増加の影響を, ほとんど無視できる. しかし, 変数分離化な性質を有する実問題が存在することは少ないと考えられている [203, 252, 93].

反対に, 複数の変数同士が互いに依存関係を有し, 分解出来ない性質を変数分離不可と呼ぶ. 一般的に, 変数分離化な問題は各次元ごとに探索を行えるため容易であり, 複数の変数間の依存関係を考慮する必要のある変数分離不可な問題は困難であるとされている [97]. このような背景から, 全ての変数が依存関係にあるようなベンチマーク関数が 2005 年頃から使用され始めた [223, 93].

しかし, 近年では全ての変数同士が依存関係にあるのは不自然であることが指摘されている [231, 132]. それについて, Tan らは実問題においては変数を部分的に分解可能な partial separability [82] の性質が頻繁に現れうると述べている [231]. ここで, 目的関数 f を $f(\boldsymbol{x}) = \sum_{j=1}^{m} g_j(\boldsymbol{x}_j)$ のように m 個の部分関数 g_j $(j \in \{1, ..., m\})$ に分解可能である場合, そのような関数は部分的に分解可能であるという. ここで, $\boldsymbol{x}_j$ は g_j についての $\boldsymbol{x}$ の互いに異なる決定変数から構成されるベクトルである. 対象問題が部分的に分解可能であり, かつどの決定変数同士が依存関係にあるかを正確に特定し分類可能であれば, f を m 個の部分問題として扱うことができる [256, 172]. しかし, 決定変数同士の依存関係を把握するためには相応の解評価を繰り返す必要があるため, 解評価コストが高価な場合は現実的なアプローチではない.

2.2.5　悪スケール性

各設計変数ごとに目的関数 f への貢献度合いが大きく異なる問題性質を, 悪スケール性と呼ぶ. これは, 例えばメートル (長さ) とグラム (重さ) のように, それぞれ異なるスケールを持つ決定変数を有する問題において現れうる [176, 7]. 正値対称行列 $\boldsymbol{H}$ に対して目的関数が $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}$ と表せられた時, $\boldsymbol{H}$ の条件数[*4]が 1 より大きければ, 目的関数 f は悪スケー

[*4] 固有値の最大値と最小値の割合を条件数と呼ぶ.

ル性を有するという [97].

一般的に, 悪スケール性を有する問題は探索が困難である. 目的関数値空間の等高線が楕円形になるため, 付録 A.1.1 節にて述べる最急降下法は直前の移動方向と直交する方向に移動, つまりジグザグに移動することになるため, 多くの解評価回数を要する. また, EA を悪スケール性を有する関数に適用した場合, 目的関数への影響力が強い設計変数が始めに最適化され, 影響力の弱い設計変数の探索が十分に行われないため, 例え単峰性関数であったとしても集団が初期収束を起こしてしまう可能性が, 手法によっては高い [97].

2.3 関数最適化問題に対する伝統的な直接探索法

対象問題の目的関数 f の微分情報が利用可能, かつ f が凸関数であった場合は, 付録 A.1 にて述べられているニュートン法などの手法が有用である. しかし, 目的関数値 $f(\boldsymbol{x})$ の情報しか利用できない black-optimization 環境, 又は f が非凸関数であった場合は, そのような手法は適用不可能, 又は効率的な探索が期待できない場合がある. このような背景から, 対象問題の目的関数値 $f(\boldsymbol{x})$ のみを利用して探索を行う手法が研究されており, 1958 年にランダムサーチ [31] が提案されて以降, Hooke-Jeeves 法 [103], Rosenbrock 法 [200], Nelder-Mead 法 [167], Solis-Wets 法 [215], 疑似焼きなまし法 [121] などがこれまでに提案されてきた [122].

次節の 2.3.1 節では, その中でも最も一般的な手法の一つであり, かつ 3 章にて説明する Differential Evolution (DE) [220, 221] の基礎となった Nelder-Mead 法 [167] について述べる. なお, その他の手法については, 付録 A.2 節を参照されたい.

2.3.1 Nelder-Mead 法

Nelder-Mead 法 [167] は, 1965 年に Nelder と Mead により提案された, 勾配情報を利用しない決定的探索アルゴリズムである. Nelder-Mead 法では $D+1$ 個の探索点から構成される単体を反射, 拡張, 縮小といったオペレータにより新たな探索点を生成し, 現探索点集合における最悪探索点と比較, 置換することで探索を行う.

Algorithm 1 に, Nelder-Mead 法を示す. ここで, $\boldsymbol{x}^{m,t}$ は最大目的関数値を持つ $\boldsymbol{x}^{D+1,t}$ を除く探索点 $\{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{D,t}\}$ の重心である. また, $\boldsymbol{x}^{r,t}$ は反射 (reflection), $\boldsymbol{x}^{e,t}$ は拡張 (expansion), $\boldsymbol{x}^{c,t}$ は縮小 (contraction) の操作により生成された新たな探索点である. α_r, α_e, α_c はそれぞれ反射, 拡張, 縮小オペレータに使用する制御パラメータである. 提案論文 [167] にて行われたパラメータ設定の調査では, $\alpha_r = 1$, $\alpha_e = 2$, $\alpha_c = 0.5$ が最も適切であった.

Nelder-Mead 法は比較的古く提案された単純な手法であるが, 対象問題の次元数が 2 次元程度であれば, 近年提案された state-of-the-art な進化アルゴリズムと同等以上の性能を持つことが報告されている [92]. また, 5 次元程度の問題においては, 他の局所探索法と比較しても良好な探索性能を示す [184]. しかし, それ以上の次元数の問題に適用した場合, 良好な性能は望めない. また, 1 次元の凸関数においてのみ局所解に収束する保証を持ち [125], 2 次元においても拡張操作を省略した Nelder-Mead 法は同様の保証を有するが, $D \leq 3$ においては不明で

Algorithm 1: Nelder-Mead 法

$t \leftarrow 1$;
$D+1$ 個の探索点 $\{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{D+1,t}\}$ を探索空間 $\boldsymbol{S}$ 中に一様ランダムに生成する;
while 探索終了条件を満たしていない **do**
    各探索点を $f(\boldsymbol{x}^{1,t}) \leq ... \leq f(\boldsymbol{x}^{D+1,t})$ となるように目的関数値に基づき並び替える;
    $\boldsymbol{x}^{m,t} = \frac{1}{D}\sum_{i=1}^{D} \boldsymbol{x}^{i,t}$;
    // 反射操作
    $\boldsymbol{x}^{r,t} = \boldsymbol{x}^{m,t} + \alpha_r(\boldsymbol{x}^{m,t} - \boldsymbol{x}^{D+1,t})$;
    if $f(\boldsymbol{x}^{1,t}) \leq f(\boldsymbol{x}^{r,t})$ **and** $f(\boldsymbol{x}^{r,t}) < f(\boldsymbol{x}^{D+1,t})$ **then**
        $\boldsymbol{x}^{D+1,t+1} = \boldsymbol{x}^{r,t}$;
    else if $f(\boldsymbol{x}^{r,t}) < f(\boldsymbol{x}^{1,t})$ **then**
        // 拡張操作
        $\boldsymbol{x}^{e,t} = \boldsymbol{x}^{m,t} + \alpha_e(\boldsymbol{x}^{r,t} - \boldsymbol{x}^{m,t})$;
        if $f(\boldsymbol{x}^{e,t}) < f(\boldsymbol{x}^{r,t})$ **then**
            $\boldsymbol{x}^{D+1,t+1} = \boldsymbol{x}^{e,t}$;
        else
            $\boldsymbol{x}^{D+1,t+1} = \boldsymbol{x}^{r,t}$;
    else
        // 縮小操作
        $\boldsymbol{x}^{c,t} = \boldsymbol{x}^{m,t} + \alpha_c(\boldsymbol{x}^{m,t} - \boldsymbol{x}^{D+1,t})$;
        if $f(\boldsymbol{x}^{c,t}) < f(\boldsymbol{x}^{D+1,t})$ **then**
            $\boldsymbol{x}^{D+1,t+1} = \boldsymbol{x}^{c,t}$;
        else
            // 全ての探索点を縮小する
            for $i = 1$ **to** $D+1$ **do**
                $\boldsymbol{x}^{i,t+1} = \frac{(\boldsymbol{x}^{i,t}+\boldsymbol{x}^{1,t})}{2}$;
    $t \leftarrow t+1$;

ある．さらに, Nelder-Mead 法を実行することにより得られる解の精度は, 特に多峰性関数においては初期探索点に強く依存する．

2.4　進化アルゴリズム

進化アルゴリズム (Evolutionary Algorithm, EA) は複数の解を用いて探索を行う近似的最適化アルゴリズムの総称である．関数最適化問題に対する EA は, Genetic Algorithm (GA) [102], Evolution Strategy (ES) [15], Evolutionary Programming (EP) [260], Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [96], Estimation of Distribution Algorithm (EDA) [201, 206], Particle Swarm Optimization (PSO) [120], Ant Colony Optimization [50, 214], Artificial Bee Colony (ABC), [116, 117] Differential Evolution [220, 221]

など, これまでに様々な枠組みが提案されている.

直接探索法と同様に, EA は目的関数値 $f(\boldsymbol{x})$ のみを利用して探索を行う. 勾配情報を利用しないため, 対象問題が微分不可能であったとしても適用可能であるため, 広い問題クラスに適用可能である. また, 複数の解を保持して探索を行うので, SA などの局所探索法と比べて局所解に探索が陥る可能性が低いと期待されている.

本章では, 特に代表的な EA の手法である GA, ES, CMA-ES, PSO について説明する. 始めに 2.4.1 節にてビットストリング GA について述べ, 2.4.2 節にて実数値 GA について説明する. 次に, 2.4.3 節にて $(1+1)$-ES, 2.4.4 節にて CMA-ES について述べる. 最後に, 2.4.5 節にて PSO について説明する. なお, 本章内にて用語を統一するため, 各手法ごとの独自の表現を避け, 解を個体, 新たに生成した解を子個体, 反復を世代などと呼んでいる.

2.4.1 ビットストリング GA (Bit String GA)

Genetic Algorithm (GA) [102] は, 生物の進化から着想を得た EA である. 交叉, 突然変異といった遺伝的オペレータを使用して子個体を生成し, 現世代 t の集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ 内の各個体の目的関数値に応じて, 次世代 $t+1$ の集団 $\boldsymbol{P}^{t+1}$ に生存する個体を決定する.

GA は他の EA と同様に, ナップサック問題, TSP などの様々な最適化問題に適用可能である. 近年において, GA を最適化問題に適用する際は, 多くの場合は対象問題の解をそのまま個体として扱う. 一方, 比較的初期の GA 研究においては, 解 (例えば TSP の場合は巡回路) を $\{0,1\}$ ベクトルに変換し, これを個体として扱っていた [102, 70].

1975 年に提案されたビットストリング GA [102] では, 関数最適化問題の解である D 次元実数値ベクトル $\boldsymbol{x} = (x_1, ..., x_D)^T$ を, $\{0,1\}$ のバイナリベクトルで表現し, これを個体として扱う. 交叉や突然変異といったオペレータはバイナリベクトルで表現された個体に適用される. 一方, 目的関数 f により個体を評価する際には, 再びバイナリベクトルを実数値ベクトルに変換する.

これまでに様々な GA モデルが提案されているが, 最も一般的な steady state GA の枠組みにおけるビットストリング GA を以下で説明する. 各世代 t において, 何らかの方法で集団 $\boldsymbol{P}^t$ から, 交叉の親個体 $\boldsymbol{x}^{a,t}, \boldsymbol{x}^{b,t}$ となる個体を選択する. ここで, 親個体の選択は重複を許さない非復元選択が一般的である. 選択方法としては, ランダム選択, ルーレット選択, トーナメント選択, ランキング選択などがあげられる. ランダム選択では, 集団中の個体を一様ランダムに 2 個体選択する. ルーレット選択では, 各個体の目的関数値に比例する確率で個体を選択する:

$$p_{i,t} = \frac{1/f(\boldsymbol{x}^{i,t})}{\sum_{j=1}^{N} 1/f(\boldsymbol{x}^{j,t})} \tag{2.7}$$

ここで, $p_{i,t}$ は個体 $\boldsymbol{x}^{i,t}$ が選択される確率である. ルーレット選択では, 目的関数値 $f(\boldsymbol{x})$ が低い個体ほど, 選択される確率が高くなる[*5]. トーナメント選択では, 集団中の個体を N_T 個ラ

[*5] ルーレット選択は, 3.3.1 節にて述べる, 順序を保持した目的関数値の変換に対する不変性を持たない. そのため, 近年ではルーレット選択はほとんど使用されていない.

ンダムに選択し, その中で最も低い目的関数値を有する個体を選択する. ランキング選択では, まず各目的関数値に従い $f(\boldsymbol{x}^{1,t}) \leq f(\boldsymbol{x}^{2,t}) \leq ... \leq f(\boldsymbol{x}^{\lambda,t})$ となるように各個体を並び替える. そして, 各個体のランクに基づき選択確率 p を決定する. 次式はランキング選択の実装例であり, 並び替えられた個体 $\boldsymbol{x}^{i,t}$ の選択確率 $p_{i,t}$ を表す:

$$p_{i,t} = \frac{N - i + 1}{\sum_{j=1}^{\mu}(N - j + 1)} \tag{2.8}$$

次に, 選ばれた親個体 $\boldsymbol{x}^{a,t}, \boldsymbol{x}^{b,t}$ に交叉オペレータを適用することにより, 子個体 $\boldsymbol{x}^{c,t}$ を生成する. 代表的な交叉手法には, 1 点交叉, 2 点交叉や一様交叉などがあげられる. 1 点交叉では, $\{1, ..., D_b\}$ の範囲の中から一様ランダムに交叉点を選択し, その交叉点を基準に $\boldsymbol{x}^{a,t}, \boldsymbol{x}^{b,t}$ の要素を交換する. ここで, D_b は個体のベクトル長である.

一様交叉では, 各成分 $j = 1, ..., D_b$ ごとに一様分布に従う区間 $[0, 1]$ の乱数 $\mathrm{rand}[0, 1]$ を生成し, 乱数値が 0.5 以下であれば $\boldsymbol{x}^{a,t}$ の要素を, そうでなければ $\boldsymbol{x}^{b,t}$ の要素を子個体 $\boldsymbol{x}^{c,t}$ に受け継ぐ:

$$x_j^{c,t} = \begin{cases} x_j^{a,t} & \text{if } \mathrm{rand}[0, 1] \leq 0.5 \\ x_j^{b,t} & \text{otherwise} \end{cases} \tag{2.9}$$

交叉により生成された子個体 $\boldsymbol{x}^{c,t}$ は, 突然変異オペレータが適用される. 各成分 j ごとに, 予め定めた突然変異確率 $p_m \in [0, 1]$ に従い, 突然変異を行うか否かを決定する. バイナリ表現型の GA において一般的なビット反転突然変異では, $x_j^{c,t}$ が 0 ならば 1, 1 ならば 0 というように記号を反転させる.

子個体 $\boldsymbol{x}^{c,t}$ の生成後, $\boldsymbol{x}^{c,t}$ はバイナリ表現から実数値表現に変換され, 目的関数により評価される. その後, 集団 $\boldsymbol{P}^t$ から置き換え対象となる個体 $\boldsymbol{x}^{r,t}$ を任意の方法で選択する. 一般的に, ランダム選択やルーレット選択といった, 先述の親個体の選択方法と同様の方法を使用する. ただし, 親個体の選択では有望な個体, すなわち低い目的関数値を有する個体ほど選択されやすいバイアスを使用していたが, 置き換え対象となる個体を選択する場合は, 有望でない個体, すなわち高い目的関数値を有する個体ほど選択されやすいようバイアスをかける. 例えば, 式 (2.8) のランキング選択は, 次のように修正される:

$$p_{i,t} = \frac{i}{\sum_{j=1}^{\mu}(j)} \tag{2.10}$$

選択された個体 $\boldsymbol{x}^{r,t}$ と子個体 $\boldsymbol{x}^{c,t}$ の目的関数値を比べ, もし $f(\boldsymbol{x}^{c,t}) \leq f(\boldsymbol{x}^{r,t})$ であれば, $\boldsymbol{x}^{r,t+1} = \boldsymbol{x}^{c,t}$ とする. 最後に, 以上に述べた steady state GA を, Algorithm 2 に示す.

関数最適化問題において, 本節で説明したビットストリング GA の性能は乏しい. これは, 真の探索空間とビットストリング GA の変換された探索空間の構造が異なり, 決定変数同士の依存関係や悪スケール性といった情報が失われるため, 交叉, 突然変異といったオペレータを行ったとしても親個体が持つ有用な情報を子個体に引き継げないためである. グレイコード表現を用いることでこの問題は多少解決されることが知られているが, 根本的な解決とはならない. この問題を解消した, 次節で説明する実数値 GA の研究に引き継がれる形で, 実用的な関数最適化問題に対するビットストリング GA の研究は, 近年ではほとんど行われていない.

Algorithm 2: Steady State モデルを使用したビットストリング GA

```
1  t ← 1;
2  集団 P^t = {x^{1,t}, ..., x^{N,t}} の各個体を探索空間 S 中に一様ランダムに生成する;
3  while 探索終了条件を満たしていない do
4      集団 P^t から, 2 個体選択し親個体とする;
5      親個体に交叉オペレータを適用し, 子個体 x^{c,t} を生成;
6      子個体 x^{c,t} に突然変異オペレータを適用;
7      集団 P^t から, 置き換え対象となる 1 個体 x^{r,t} を選択;
8      if f(x^{c,t}) ≤ f(x^{r,t}) then
9          x^{r,t+1} = x^{c,t};
10     t ← t + 1;
```

2.4.2 実数値 GA

ビットストリング GA の限界が指摘されるようになり, 実数値ベクトル $\boldsymbol{x}$ をそのまま個体として扱う実数値 GA (Real-Coded GA) の開発が行われるようになった. 一般的に実数値 GA は突然変異を使用せず, 交叉オペレータのみで子個体を生成する. 代表的な実数値 GA の交叉手法には, Simulated Binary Crossover (SBX) [45], Blend Crossover (BLX-α) [58], Parent Centric Blend Crossover (PBX-α) [151], Simplex Crossover (SPX) [243], Unimodal Normal Distribution Crossover (UNDX) [175], Parent Centric Crossover (PCX) [46] などがある.

ここでは, 一般的な BLX-α, PBX-α, PCX について説明する. BLX-α では, 子個体 $\boldsymbol{x}^{c,t}$ の各要素 $x_j^{c,t}$ は, 2 個体の親個体 $\boldsymbol{x}^{a,t}$, $\boldsymbol{x}^{b,t}$ の各変数の区間 $[x_j^{a,t}, x_j^{b,t}]$ を α 倍拡張した範囲内に一様ランダムに生成される:

$$x_j^{c,t} = \mathrm{rand}[\min(x_j^{a,t}, x_j^{b,t}) - \alpha I, \max(x_j^{a,t}, x_j^{b,t}) + \alpha I] \tag{2.11}$$

ここで, $I = |x_j^{a,t} - x_j^{b,t}|$ である. また, α の推奨値は 0.5 である.

BLX-α の子個体が生成されうる分布の中心は, 2 個体の親個体 $\boldsymbol{x}^{a,t}$, $\boldsymbol{x}^{b,t}$ の平均 $\boldsymbol{m}^t = \frac{\boldsymbol{x}^{a,t}+\boldsymbol{x}^{b,t}}{2}$ と一致する. 一方, PBX-α では, $\boldsymbol{x}^{a,t}$ と $\boldsymbol{x}^{b,t}$ のいずれかをランダムに選択し, 選ばれた個体を中心とする:

$$x_j^{c,t} = \mathrm{rand}[x_j^{p,t} - \alpha I, x_j^{p,t} + \alpha I] \tag{2.12}$$

ここで, $x_j^{p,t}$ は $\boldsymbol{x}^{a,t}$ と $\boldsymbol{x}^{b,t}$ の選択された方の j 番目の要素である. なお, PBX-α の提案論文 [151] における α の推奨値は 1.0 である.

上記の BLX-α, 及び PBX-α では集団中の 2 個体の親個体を使用して子個体を生成した. 一方, PCX では集団中の任意の μ 個体 $\{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{\mu,t}\}$ を親個体集団とする. ここで, $\mu \geq 2$ であ

る次式に PCX の交叉方法を示す:

$$\boldsymbol{x}^{c,t} = \boldsymbol{x}^{p,t} + w_\zeta \boldsymbol{d}^{p,t} + \sum_{i=1, i\neq p}^{\mu} w_\eta \overline{D}_t \boldsymbol{e}^{i,t} \tag{2.13}$$

$\boldsymbol{x}^{p,t}$ は親個体集団 $\{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{\mu,t}\}$ から任意の方法で選択した個体である．$\boldsymbol{d}^{p,t}$ は親個体集団の重心 $\boldsymbol{m}^t = 1/\mu \sum_{i=1}^{\mu} \boldsymbol{x}^{i,t}$ から $\boldsymbol{x}^{p,t}$ に向かう方向ベクトル $\boldsymbol{d}^{p,t} = \boldsymbol{x}^{p,t} - \boldsymbol{m}^t$ である．$\overline{D}_t$ は，$\boldsymbol{x}^{p,t}$ を除く $\mu - 1$ 個体から $\boldsymbol{d}^{p,t}$ への垂直距離 $D_{i,t}$ の平均である．$\boldsymbol{e}^{i,t}$ は $\boldsymbol{x}^{p,t}$ を除く $\mu - 1$ 個体の，$\boldsymbol{d}^{p,t}$ への直交部分空間を張る正規直交基底である．w_ζ, w_η は平均 0，分散 σ_ζ^2, σ_η^2 の正規分布に従う乱数である．親個体数 μ の推奨サイズは 3，σ_ζ^2, σ_η^2 の推奨値は共に 0.1 である．

[46] では，子個体分布の中心が親個体集合の重心に一致する交叉手法を mean centric crossover，親個体集合のいずれかの親個体に一致する交叉手法を parent centric crossover と呼び，分類している．BLX-α, SPX, UNDX が mean centric crossover，SBX, PBX-α, PCX が parent centric crossover に分類される．Mean centric crossover は集団を探索領域 $\boldsymbol{S}$ に一様ランダムに初期化した場合，$\boldsymbol{S}$ の中心付近に収束しやすい．そのため，最適解 $\boldsymbol{x}^*$ が探索空間の中心付近に位置する場合は高精度の解を求めやすい一方，境界付近に位置する場合は良好な結果が望めないことが指摘されている [67, 238]．

実数値 GA において，交叉手法と並んで重要な要素は世代交代モデルである．世代交代モデルは，どのように親個体集合を選択し，交叉により新しく生成した子個体をどの基準にて現世代の個体と置き換えるかを決定する．これまでに，CHC [58], Minimal Generation Gap (MGG) [205], Just Generation Gap (JGG) [2], Generalized Generation Gap (G3) [46] などが提案されているが，最も一般的な G3 を Algorithm 3 に示す．ここで，λ は子個体生成数であり，PCX を用いた場合の推奨値は $\lambda = 2$ である．これまで様々な実数値 GA の交叉オペレータ，世代交代モデルが提案されているが，交叉に PCX，世代交代モデルに G3 を用いた G3-PCX が現在の state-of-the-art とされている．G3 を世代交代モデルに使用した場合，PCX の子個体分布の中心となる親個体 $\boldsymbol{x}^{p,t}$ は，集団中の最良個体 $\boldsymbol{x}^{best,t}$ とすることが推奨されている [46]．これは，最良個体付近に子個体を常に生成することを意味し，単峰性関数においては高速な収束を示す．その一方，多峰性関数においては局所解に誤って収束してしまう可能性が高く，また例え低次元な単峰性関数であったとしても，他の EA と比べ巨大な集団サイズを必要とするなど，課題が多い [182]．

2.4.3　ES

Evolution Strategy (ES) [15] は，1960 年代に Schwefel, Rechenberg, Bienert によって工業設計問題を解くために提案された，主に関数最適化問題を対象とした EA である．(μ, λ)-ES など，様々なバリエーションが存在するが [15]，以下では最も基本的な $(1 + 1)$-ES について説明する．

$(1 + 1)$-ES では，他の EA とは異なり 1 個体しか探索に利用しない．各世代 t において，個

Algorithm 3: G3

$t \leftarrow 1$;
集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の各個体を探索空間 $\boldsymbol{S}$ 中に一様ランダムに生成する;
while 探索終了条件を満たしていない **do**
  最も低い目的関数値を有する個体 $\boldsymbol{x}^{best,t}$ と $\mu - 1$ 個体を集団 $\boldsymbol{P}^t$ からランダムに選択し, 親個体集団とする;
  λ 個の子個体を交叉により生成する;
  集団 $\boldsymbol{P}^t$ からランダムに 1 個体 $\boldsymbol{x}^{r,t}$ 選択;
  λ 個の子個体と $\boldsymbol{x}^{r,t}$ の集合において, 最も低い目的関数値を有する個体 $\boldsymbol{x}^{cbest,t}$ を選択;
  if $f(\boldsymbol{x}^{cbest,t}) \leq f(\boldsymbol{x}^{r,t})$ **then**
    $\boldsymbol{x}^{r,t+1} = \boldsymbol{x}^{cbest,t}$;
  $t \leftarrow t + 1$;

体 $\boldsymbol{x}^t$ に突然変異操作を加えることで, 変異個体 $\boldsymbol{u}^t$ を生成する:

$$\boldsymbol{u}^t = \boldsymbol{x}^t + \sigma \cdot \mathrm{randn}(\boldsymbol{0}, \boldsymbol{I}) \tag{2.14}$$

ここで, $\mathrm{randn}(\boldsymbol{0}, \boldsymbol{I})$ は多変量標準正規分布に従う, 各成分ごとに独立な乱数である. σ は突然変異の大きさを決めるステップサイズである.

変異個体 $\boldsymbol{u}^t$ の生成後, 目的関数値を算出し, 現在の個体 $\boldsymbol{x}^t$ よりも優れていた場合のみ世代交代を行う:

$$\boldsymbol{x}^{t+1} = \begin{cases} \boldsymbol{u}^t & \text{if } f(\boldsymbol{u}^t) \leq f(\boldsymbol{x}^t) \\ \boldsymbol{x}^t & \text{otherwise} \end{cases} \tag{2.15}$$

ここで問題となるのが, σ の設定である. σ を対象問題ごとに適切にパラメータチューニングする必要があるが, σ を適応的に調整する方法として, 1/5 success rule が提案されている. 1/5 success rule では, 世代交代を行う式 (2.15) において, $f(\boldsymbol{u}^t) \leq f(\boldsymbol{x}^t)$ となった場合を成功, $f(\boldsymbol{u}^t) > f(\boldsymbol{x}^t)$ であった場合を失敗と呼ぶ. この適応戦略では, 突然変異が成功する率を 1/5 になるように σ を調整する. 1/5 success rule では, D 世代ごとに, 過去 H 世代における成功, 及び失敗した回数を保持し, H 世代ごとに突然変異の成功率 r_s を求める:

$$r_s = \frac{\text{過去 } H \text{ 世代間に成功した回数}}{\text{過去 } H \text{ 世代間に成功した回数} + \text{過去 } H \text{ 世代間に失敗した回数}} \tag{2.16}$$

ここで H の推奨値は $H = 10D$ である. そして, r_s の値に基づき, σ を更新する:

$$\sigma = \begin{cases} c\sigma & \text{if } r_s < 1/5 \\ \sigma/c & \text{if } r_s > 1/5 \\ \sigma & \text{otherwise} \end{cases} \tag{2.17}$$

ここで, $c \in [0, 1]$ は σ の増減幅を決定する制御パラメータである. c の推奨値は 0.87 である. 成功率 r_s が 1/5 より高ければ σ を減少させ, 1/5 未満であれば σ を増加させる.

Algorithm 4: $(1+1)$-ES

```
t ← 1;
x^t, σ を初期化する;
while 探索終了条件を満たしていない do
    式 (2.14) により変異個体 u^t を生成;
    式 (2.15) により x^{t+1} を求める;
    if t modulo D == 0 then
        式 (2.17) により σ を更新;
    t ← t + 1;
```

最後に, 以上に述べた $(1+1)$-ES の全体を Algorithm 4 に示す. $(1+1)$-ES は非常に単純な手法であるため実装が容易である利点を持つが, 1 個体しか使用していないため大域的探索性能に乏しい. また, 変数間の依存関係を考慮していないため, 変数分離不可な問題に対応することが困難である. これらの問題点を解決するために, 次節で述べる CMA-ES [96] が提案された.

2.4.4　CMA-ES

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [96] は, 前節で説明した ES をより洗練させた手法である. 近年の関数最適化問題に対する state-of-the-art な手法として知られている [92]. CMA-ES では, 平均ベクトル $\boldsymbol{m} = (m_1, ..., m_D)^T$, 共分散行列 $\sigma\boldsymbol{C}$ の多変量正規分布に従う乱数を用いた突然変異により, 新たな子個体を生成する. そして, 各子個体の目的関数値を考慮し, 分布の平均を $\boldsymbol{m}$, 突然変異の大きさを σ, 分布の形状を $D \times D$ の共分散行列 $\boldsymbol{C}$ を更新することで, 新たな突然変異分布を得る. この操作を探索終了条件を満たすまで繰り返す.

これまでに多くの CMA-ES アルゴリズムが提案されているが [95, 96, 94, 111], 以下では, [87] の解説により説明されている近年標準的な CMA-ES である, (μ_w, λ)-CMA-ES について述べる. なお, 以下の説明にて使用する制御パラメータの推奨設定を表 2.1 に示している. 表中の μ_{eff} については, 後述の重みベクトル $\boldsymbol{w} = (w_1, ..., w_\mu)$ に対して, $\mu_{eff} = (\sum_{i=1}^{\mu} w_i^2)^{-1}$ である.

探索の開始時 $t = 1$ において, $\boldsymbol{m}^t$ を探索空間内に一様ランダムに生成し, $\boldsymbol{C}^t = \boldsymbol{I}$, $\boldsymbol{p}^{\sigma,t} = \boldsymbol{0}$, $\boldsymbol{p}^{c,t} = \boldsymbol{0}$ とする. また, 探索領域 $[A, B]^D$ について, $\sigma_t = 0.5(A - B)$ とする. なお, CMA-ES は 3.3.1 節にて説明する探索空間の線形変換に対する不変性を有するため, 探索領域を D 次元の超立方体 $[A, B]^D$ に正規化を行ったとしても, 探索性能に問題は無い.

各世代 t の始めに $\boldsymbol{C}^t$ を固有値分解し, $\boldsymbol{C}^t = \boldsymbol{B}^t\boldsymbol{D}^t(\boldsymbol{B}^t)^T$ を得る. ここで, $\boldsymbol{B}^t$ は固有ベクトル, $\boldsymbol{D}^t$ は各要素が固有値の対角行列 (固有値ベクトル) である. その後, 各個体 $\boldsymbol{x}^{i,t}$

表 2.1: (μ_w, λ)-CMA-ES における推奨パラメータ設定 [87].

パラメータ名	設定
子個体数	$\lambda = 4 + \lfloor 3\ln(D) \rfloor$, $\mu = \lfloor \lambda/2 \rfloor$
σ の更新	$c_\sigma = \frac{\mu_{eff}+2}{D+\mu_{eff}+3}$, $d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{eff}-1}{D+1}} - 1\right) + c_\sigma$
$\boldsymbol{C}$ の更新	$c_c = \frac{4}{D+4}$, $\mu_{cov} = \mu_{eff}$, $c_{cov} = \frac{1}{\mu_{cov}} \frac{2}{(D+\sqrt{2})^2} + \left(1 - \frac{1}{\mu_{cov}}\right) \min\left(1, \frac{2\mu_{eff}-1}{(D+2)^2+\mu_{eff}}\right)$

$(i \in \{1, ..., \lambda\})$ は次式により生成される:

$$\boldsymbol{x}^{i,t} = \boldsymbol{m}^t + \sigma_t \boldsymbol{B}^t \boldsymbol{D}^t (\boldsymbol{B}^t)^T \mathrm{randn}(\boldsymbol{0}, \boldsymbol{I}) \tag{2.18}$$

ここで, λ は個体数である. $\mathrm{randn}(\boldsymbol{0}, \boldsymbol{I})$ は多変量標準正規分布に従う, 各成分ごとに独立な乱数である. また, 式 (2.18) の右辺は $\mathrm{randn}(\boldsymbol{m}, \sigma_t (\boldsymbol{C}^t)^2)$ と同意義である.

λ 個の子個体を生成し目的関数により評価した後, 各目的関数値に従い $f(\boldsymbol{x}^{1,t}) \leq f(\boldsymbol{x}^{2,t}) \leq ... \leq f(\boldsymbol{x}^{\lambda,t})$ となるように各個体を並び替える. その後, 次に述べるように $\boldsymbol{m}^t$, σ_t, $\boldsymbol{C}^t$ を更新する.

次世代 $t+1$ の突然変異分布の中心となる平均ベクトル $\boldsymbol{m}^{t+1}$ は, λ 個の子個体から上位 μ 個体の重み付き平均により得られる:

$$\boldsymbol{m}^{i,t+1} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}^{i,t} \tag{2.19}$$

$$w_i = \frac{\ln(\mu+1) - \ln i}{\sum_{j=1}^{\mu} (\ln(\mu+1) - \ln j)} \tag{2.20}$$

ここで, μ は親個体数であり, $\mu \leq \lambda$ である. また, w_i は $w_1 \geq w_2 \geq ... \geq w_\mu > 0$ であり, $\sum_{i=1}^{\mu} w_i = 1$ である.

次世代の突然変異の大きさを制御するステップサイズ σ_{t+1} は, 次式により更新される:

$$\boldsymbol{p}^{\sigma,t+1} = (1 - c_\sigma)\boldsymbol{p}^{\sigma,t+1} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}} \boldsymbol{C}^{t,-1/2} \frac{\boldsymbol{m}^{t+1} - \boldsymbol{m}^t}{\sigma_t} \tag{2.21}$$

$$\sigma_{t+1} = \sigma_t \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}^{\sigma,t+1}\|}{\mathrm{E}\|\mathrm{randn}(\boldsymbol{0}, \boldsymbol{I})\|} - 1\right)\right) \tag{2.22}$$

$$\tag{2.23}$$

ここで, $\mathrm{E}||\mathrm{randn}(\boldsymbol{0}, \boldsymbol{I})||$ は $\mathrm{randn}(\boldsymbol{0}, \boldsymbol{I})$ のユークリッドノルムの期待値であり, 通常はその近似値 $\mathrm{E}||\mathrm{randn}(\boldsymbol{0}, \boldsymbol{I})|| \approx \sqrt{D}(1 - 1/4D + 1/21D^2)$ が使用される.

最後に, 突然変異分布の形状を制御する共分散行列 $\boldsymbol{C}^{+1}$ の更新を行う:

$$h_{\sigma,t+1} = \begin{cases} 1 & \text{if } \frac{\|\boldsymbol{p}^{\sigma,t+1}\|}{\sqrt{1-(1-c_\sigma)^{2(t+1)}}} < (1.5 + \frac{1}{D-0.5})\mathrm{E}\|\mathrm{randn}(\boldsymbol{0}, \boldsymbol{I})\| \\ 0 & \text{otherwise} \end{cases} \tag{2.24}$$

Algorithm 5: CMA-ES

$t \leftarrow 1$;
$\boldsymbol{p}^{\sigma,t} = \boldsymbol{0}$, $\boldsymbol{p}^{c,t} = \boldsymbol{0}$, $\boldsymbol{C}^t = \boldsymbol{I}$, $\sigma_t = 0.5(A-B)$ とする;
while 探索終了条件を満たしていない **do**
　式 (2.18) に従い, $\boldsymbol{m}^t, \sigma_t, \boldsymbol{C}^t$ を用いて λ 個の子個体を生成;
　各個体の目的関数値に基づき, 個体を並び替える;
　式 (2.19) により $\boldsymbol{m}^{t+1}$ を更新;
　式 (2.21), (2.22) により σ_{t+1} を更新;
　式 (2.24), (2.25), (2.26) により $\boldsymbol{C}^{t+1}$ を更新;
　$t \leftarrow t+1$;

$$\boldsymbol{p}^{c,t+1} = (1-c_c)\boldsymbol{p}^{c,t+1} + h_{\sigma,t+1}\sqrt{c_c(2-c_c)\mu_{eff}}\frac{\boldsymbol{m}^{t+1}-\boldsymbol{m}^t}{\sigma_t} \tag{2.25}$$

$$\boldsymbol{C}^{t+1} = (1-c_{cov})\boldsymbol{C}^t + \frac{c_{cov}}{\mu_{cov}}\left(\boldsymbol{p}^{c,t+1}(\boldsymbol{p}^{c,t+1})^T + \delta(h_{\sigma,t+1})\boldsymbol{C}^t\right) \tag{2.26}$$

$$+ c_{cov}\left(1-\frac{1}{\mu_{cov}}\right)\sum_{i=1}^{\mu} w_i \mathrm{OP}\left(\frac{\boldsymbol{x}^{i,t+1}-\boldsymbol{m}^t}{\sigma_t}\right) \tag{2.27}$$

ここで, OP は外積を表す. また, $\delta(h_{\sigma,t+1}) = (1-h_{\sigma,t+1})c_c(2-c_c)$ である.

最後に, 以上の説明をまとめた (μ_w, λ)-CMA-ES を Algorithm 5 に示す. CMA-ES は軸変換に対して回転不変性を持つことから, 変数分離不可な問題に対しては他の EA よりも高い性能を示す [94, 10]. 特に, 単峰性関数における性能は, 他の EA よりも優れている. また, 強い悪スケール性を有する場合は凸関数においても A.1.3 節にて説明した準ニュートン法よりも高い探索性能を示す [10].

CMA-ES の問題点は, 個体数 λ の適切な設定が対象問題の問題性質に大きく依存することである. 対象問題が単峰性関数であれば, λ を小さくすれば収束速度が向上する. 一方, 多峰性関数であった場合は λ を大きく設定しなければ, 探索が局所解に誤って収束する可能性が高くなる. しかし, black-box optimization においては, 対象問題の景観を事前に知ることは不可能であるため, 対象問題ごとに適切にパラメータチューニングを行う必要がある.

このパラメータ設定問題を解消するために, 決定的に集団数 λ を調整する戦略がいくつか提案されている. リスタート毎に集団数 λ を 2 倍づつ増加させる IPOP-CMA-ES [9], それに加えて初期ステップサイズ σ_1 を減少させる NEW IPOP-CMA-ES (NIPOP-CMA-ES) [149], IPOP-CMA-ES と multi-start の CMA-ES の 2 つの戦略を使用する BIPOP-CMA-ES[88] などの枠組みが, 代表例としてあげられる.

2.4.5　PSO

Particle Swarm Optimization (PSO) は 1995 年に Kennedy と Eberhart に提案された, 生物の群れの振る舞いから着想を得た進化アルゴリズムである [120]. 集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$

Algorithm 6: PSO

```
t ← 1;
各個体 {x^{1,t}, ..., x^{N,t}}, 及び各速度ベクトル {v^{1,t}, ..., v^{N,t}} を初期化;
while 探索終了条件を満たしていない do
    for i = 1 to N do
        v^{i,t} = w v^{i,t} + c_1 U^{1,i,t}(x^{pbest,i,t} − x^{i,t}) + c_1 U^{2,i,t}(x^{gbest,t} − x^{i,t});
        x^{i,t+1} = x^{i,t} + v^{i,t+1};
    for i = 1 to N do
        if f(x^{i,t+1}) ≤ f(x^{pbest,i,t} then
            x^{pbest,i,t+1} = x^{i,t+1};
    t ← t + 1;
```

の各個体 $\boldsymbol{x}^{i,t}$ は速度ベクトル $\boldsymbol{v}^{i,t}$, 及び探索中に得られた最良探索点 $\boldsymbol{x}^{pbest,i,t}$ を持つ.

実数値 GA においては交叉オペレータと世代交代モデルが主なアルゴリズムの構成要素であったように, PSO においては速度ベクトルの更新方法 (新たな解の生成方法) と, 個体群のトポロジーが重要な要素である. 速度ベクトルの更新方法には, Inertia weight model [210], Constriction coefficients [37] や Fully Informed Particle Swarm (FIPS) [158], 個体群のトポロジーには gbest model [120, 210], lbest [222] などがこれまでに提案されている. 以下では [210] にて提案された最も基本的な gbest/inertia weight model について述べる.

探索の開始時 $t = 1$ において, 集団 $\boldsymbol{P}^t$ と各個体の速度ベクトルと最良探索点を初期化する. そして, 各世代の始めに, 次式により速度ベクトルを更新する:

$$\boldsymbol{v}^{i,t+1} = w\boldsymbol{v}^{i,t} + c_1\boldsymbol{U}^{1,i,t}(\boldsymbol{x}^{pbest,i,t} - \boldsymbol{x}^{i,t}) + c_1\boldsymbol{U}^{2,i,t}(\boldsymbol{x}^{gbest,t} - \boldsymbol{x}^{i,t}) \tag{2.28}$$

ここで, w は慣性項, c_1, c_2 は制御パラメータである. $\boldsymbol{U}^{1,i,t}, \boldsymbol{U}^{2,i,t}$ は $D \times D$ の対角行列であり, 各要素はそれぞれ $[0, 1]$ 区間内の一様乱数により与えられる. また, $\boldsymbol{x}^{gbest,t}$ は集団 $\boldsymbol{P}^t$ において最も低い目的関数値を有する $\boldsymbol{x}^{pbest}$, つまり $f(\boldsymbol{x}^{gbest,t}) \leq f(\boldsymbol{x}^{pbest,i,t}), \forall \boldsymbol{x}^{pbest,i,t} \in \boldsymbol{P}^t$ である.

速度ベクトルの更新後, 各個体の現在の探索点の更新を行う:

$$\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t} + \boldsymbol{v}^{i,t+1} \tag{2.29}$$

新たな探索点を目的関数にて評価した後, $\boldsymbol{x}^{pbest,i,t}$ を更新する:

$$\boldsymbol{x}^{pbest,i,t+1} = \begin{cases} \boldsymbol{x}^{i,t+1} & \text{if } f(\boldsymbol{x}^{i,t+1}) \leq f(\boldsymbol{x}^{pbest,i,t}) \\ \boldsymbol{x}^{pbest,i,t} & \text{otherwise} \end{cases} \tag{2.30}$$

以上のステップを終了条件を満たすまで繰り返す. 最後に, Algorithm 6 にここで記述した PSO アルゴリズムを示す. PSO アルゴリズムは比較的単純な手法であり実装が容易なため, 関数最適化問題に対する EA の中でも最も研究数が多い手法の一つである. しかし, 式 (2.28) にて各決定変数ごとに独立して摂動を求めているため, 複数の変数が依存関係にある変数分離

不可な問題に対応できない [97]. また, 常に最良探索点に向かう方向に新たな探索点を生成するため, 多峰性関数における探索性能が乏しい [135].

2.5　探索手法の性能評価のためのベンチマーク集合

本節では, 関数最適化問題に対する EA の性能評価方法について述べる. 始めに, 2.5.1 節にて実験的な性能評価を使用する背景について述べ, ベンチマーク関数を用いた性能評価の妥当性について議論する. 次に, 2.5.2 節にてベンチマーク関数を使用した場合に起こりえる性能誤評価, 及びその解決方法について説明する. 最後に, 2.5.3 節にて, 近年の代表的なベンチマークセット, 及び本論文にて主に使用する BBOB benchmarks について説明する.

2.5.1　ベンチマーク関数を用いた性能評価の妥当性

対象問題の目的関数 f が二階微分可能である場合における, 最急降下法 (A.1.1 節参照) とニュートン法 (A.1.2 節参照) の探索性能の優劣について考える. この場合, 最急降下法は一次収束, ニュートン法は二次収束であるので, ニュートン法が明らかに優れていると言える. 一方, 確率的探索手法である EA の探索性能に優劣をつけることは, 容易ではない. 特定の手法間における特定の性質を有する関数, つまり限定的な状況下において「なぜ手法 A は手法 B よりも優れているのか?」を理論的に説明する方法論は, これまでに提案されている [3]. しかし, 理論的に一般的な問題クラスにおける手法間の優劣を判定することは困難である. また, 全ての問題において他の手法よりも平均性能が良い手法は存在しないことが No Free Lunch の定理より知られている [254].

上記の背景から, EA の探索性能を比較するには, 計算機実験による実験的評価が一般的である. 例えば, 信号処理 [219], ロボットにおけるマルチセンサの統合処理 [115], 電磁気学 [195], web 検索におけるランキング関数の学習 [22], 集積回路設計 [141] など, 特定の実問題に適用し, 得られた解の質で性能に優劣をつける方法が考えられる. しかし, ソフトウェア, 及びハードウェア上の問題から, 他の計算機環境において得られた結果を再現することは容易ではない. また, その場合は得られた実験結果を一般化することが困難である. 例えば, ある論文において, その著者が実問題 P において, 手法 A と手法 B を比較し, 手法 A の方が優れているとする. この結論は, 実問題 P においてのみ有効な結論であり, 別の実問題 P' においてはあてはまらない. また, 後年, 別の著者が新たに手法 C を設計し, 手法 A, B との比較を試みたとする. この時, 全く同じ実問題 P にて手法を評価する必要があるが, その著者に実問題 P に関する知識, 及び実装できる環境が整っていない場合, これは非常に困難である.

以上のように, 実問題を用いて EA の性能を評価することは困難である. そのため, 2.2 節で述べた実問題に現れうる問題性質を有する, 人工的に作成したベンチマーク関数を用いた性能評価が一般的な方法である. ここで, ベンチマーク関数は $f(\boldsymbol{x}) = \sum_{j=1}^{D} x_j^2$ のように与えられる. 代表的なベンチマーク関数には Sphere 関数 $f(\boldsymbol{x}) = \sum_{j=1}^{D} x_j^2$, Rastrigin 関数 $f(\boldsymbol{x}) = \sum_{i=1}^{D} \big(x_i^2 - 10\cos 2\pi(x_i) + 10\big)$, Rosenbrock 関数 $f(\boldsymbol{x}) = \sum_{i=1}^{D-1} \big(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\big)$

などがある．これらのベンチマーク関数は, 実問題に現れうる問題性質を端的に有するように設計されている．例えば, Sphere 関数は単峰性, Rastrigin 関数は多峰性, Rosenbrock 関数は変数分離不可といった性質を有する．このような関数における EA の性能を評価することにより, 「単峰性関数である Sphere 関数において, 手法 A は手法 B よりも優れていた．この結果から, 対象問題が単峰性の性質を有する場合, 手法 A の方が有効である.」や, 「一方, 多峰性関数である Rastrigin 関数では, 手法 B の方が手法 A よりも優れていた．このことから, 対象問題が複数の局所解を有する場合は手法 B を使用するべきである」といった, より一般性のある定性的な知見が得られる．また, ベンチマーク関数の定義は比較的単純な数式により与えられるため, 任意の計算機環境においても実装が容易であり, 実験を再現することは難しくない.

2.5.2 ベンチマーク関数を使用した場合に起こりえる性能誤評価, 及びその解決方法

2.5.1 節では, ベンチマーク関数を用いた性能評価の妥当性について述べた．しかし, 人工的に作成したベンチマーク関数は, EA の性能を誤評価する危険性があることが指摘されている [67, 203, 252, 231, 138, 154]．最も大きな問題点は, 人工的なベンチマーク関数はしばしば実問題では一般的ではない性質を暗に有することである．以下では, この性質を「非現実的な性質」と呼ぶ．例えば, [260] などのベンチマークセットで見られる古典的な関数の多くは, 次のような非現実的な性質を持つことが知られている [138]:

1. 最適解が全ての変数において同じ値を取る,
2. 最適解が原点 $(0, ..., 0)^T$ と一致する,
3. 最適解が探索範囲の中心付近に存在する

これらの非現実的な性質を利用する手法は, 過大評価されてしまう恐れがある．例えば, 最適解が全ての変数において同じ値を取る関数 (性質 1) では, 探索中に得られた良解 $\boldsymbol{x}$ において, ある次元 $i \in \{1, ..., D\}$ における決定変数 x_i を他の次元 j の決定変数 x_j に $x_j = x_i$ とコピーするオペレータを有するアルゴリズムによる, 効率的な探索が可能である．このような手法には, それぞれ [138, 139] で指摘されているよう, MAGA [272] や Best-so-far ABC algorithm [17] などが挙げられる.

また, 最適解が原点 $(0, ..., 0)^T$ と一致する関数 (性質 2) は, 新たな解 $\boldsymbol{x}$ を原点に近づける, 又は原点と一致させるようなオペレータを使用すれば, 関数の景観に影響されることなく, 必ず最適解を求めることが可能である．Liang らは, 最適解が原点にあるという性質のために課題評価されてしまった手法の実例として, MAGA [272] をあげている [138]．また, 近年提案された Fireworks Algorithm [226] も, 原点付近に新たな個体を生成するようなオペレータを使用しており, そのために特定の問題において性能が過大評価されていたことが, [270] にて指摘されている.

最適解が探索範囲の中心付近に存在する関数 (性質 3) では, 探索空間の中心を偏重して探索するような手法を用いた場合, 効率的に探索できる [67]．例えば, BLX-α [58], UNDX [175]

や SPX [243] といった実数値 GA の mean centric crossover は, 子個体分布の中心が親個体の重心に一致するため, 初期化領域の中心に集団が収束する傾向がある. そのため, 集団を探索領域に一様ランダムに初期化した場合, (3) の性質を持つ関数を容易に探索することができる. また, 速度ベクトルの生成に複数の particle の重心を使用する PSO アルゴリズムも, 同様の傾向が見られることが指摘されている [164].

先述の 3 つの非現実的な性質以外にも, 多くの関数において最適値 $f(\boldsymbol{x}^*)$ が 0 であることが指摘されている [223]. このような関数では, 集団中の個体の目的関数値が 0 に近づくほど最適解に近づいていると判断でき, この情報を探索に利用することが可能である[*6] [242]. また, Rastrigin 関数のように規則的に局所解が存在する場合, 一定のステップサイズで探索が行える可能性があることが, MacNish に指摘されている [154]. De Jong's benchmarks [114] や 13 classical functions [260] に含まれるベンチマーク関数は, 変数分離可な性質を有する関数が比較的多く, 変数分離不可な関数は少ない. そのため, 前者の性質に特化した手法がこうしたベンチマーク集合では有利になり過大評価してしまう恐れがあることを, Salomon と Whitley らは指摘している [203, 252]. この問題に対して, Salomon [203] は $D \times D$ の回転行列を使用する方法を, Whitley ら [252] は変数分離不可な関数で複数の決定変数をラッピングする方法を提案している. しかし, Tan らは実問題においては変数を部分的に分解可能な partial separability (2.2.4 節参照) の性質が頻繁に現れうるとし, 全ての変数同士が依存関係にあるのは不自然であると指摘している [231, 132]. 実問題には大域的単峰性関数と大域的多峰性関数 (2.2.3 節参照) が現れうる. しかし, Lunacek らは既存の多くのベンチマーク関数は前者に分類され, 後者の問題性質を有する関数がほとんど無いことを指摘している [153]. 解決策として, 同論文において Lunacek らは大域的単峰性を有するベンチマーク関数を大域的多峰性関数に変換する方法を提案している.

2.5.3　代表的なベンチマーク集合と BBOB benchmarks

全ての問題において他の手法よりも平均性能が良い手法は存在しないことが, No Free Lunch の定理より知られている [254]. そのため, 2.2 節にて述べた各問題性質における EA の性能を評価する必要があり, 様々な問題性質を有する複数の関数から成るベンチマークセットによる実験的性能評価が一般的である. 以下は近年の代表的なベンチマークセットである:

- De Jong's benchmarks [114]
- 13 classical functions [260]
- CEC2005 benchmarks [223]
- CEC2013 benchmarks [137]
- CEC2014 benchmarks [136]

[*6] [242] が対象問題例としてあげている学習誤差を 0 にするパラメータを求める問題では, $f(\boldsymbol{x}^*) = 0$ となる可能性が高い. そのため, $f(\boldsymbol{x}^*) = 0$ な関数が非現実的であるとは, 一概には言えない. しかし, black-box optimization 環境における手法の探索性能を評価するには, $f(\boldsymbol{x}^*) = 0$ であることを前提とすることは好ましくない.

- CEC2010 LSGO benchmarks [231]
- CEC2013 LSGO benchmarks [132]
- SOCO benchmarks [100]
- BBOB benchmarks [93]

De Jong's benchmarks, 13 classical functions は前節で述べた「非現実的な性質」を有するベンチマーク関数を多く含んでいるため, 今日では使用される頻度は減少している. CEC2010 LSGO benchmarks, 及び SOCO benchmarks は 1000 次元規模の高次元最適化問題に対する手法のためのベンチマーク集合である. また, CEC2005/2013/2014 benchmarks, 特に CEC2005 benchmarks は今日において最も広く使用されているベンチマーク集合である. しかし, CEC2005 benchmarks は black-box optimization 環境の関数最適化問題のコンペティションのために設計されたベンチマーク集合である. そのため, 2.5.1 節にて述べた「単峰性関数である Sphere 関数において, 手法 A は手法 B よりも優れていた. この結果から, 対象問題が単峰性の性質を有する場合, 手法 A の方が有効である.」といった, 一般性のある定性的な知見を得ることが目的な場合には, このようなベンチマーク集合が必ずしも適しているとは限らない.

一方, BBOB benchmarks はこのような目的のために, Hansen らによって設計されたベンチマーク集合である. BBOB benchmarks は国際会議 Genetic and Evolutionary Computation Conference (GECCO) の 2009 年, 2010 年, 2012 年, 2013 年, 2015 年に開催されたワークショップ「Black-Box-Optimization-Benchmarking」にて, どの手法がどの関数クラスにおいて優れているか, 又は black box optimization 環境を想定した全ての関数クラスにおいて優れている手法について議論するために使用されている. また, Hansen らが管理する COmparing Continuous Optimisers (COCO)[*7] というウェブサイト上に, 過去の Black-Box-Optimization-Benchmarking に参加した手法の実験データが公開されているため, 手法間の比較が容易である.

BBOB benchmarks は 24 個のベンチマーク関数 $f_1 \sim f_{24}$ から構成されており, $f_1 \sim f_5$ は変数分離可な関数, $f_6 \sim f_9$ は弱い悪スケール性関数, $f_{10} \sim f_{14}$ は強い悪スケール性関数, $f_{15} \sim f_{19}$ は大域的単峰性の多峰性関数, $f_{20} \sim f_{24}$ は弱い大域的構造を有する関数である. 表 2.2 に, BBOB benchmarks [91, 93] の 24 個の関数 $f_1 \sim f_{24}$ の大まかな問題性質 [159] を示す. 各ベンチマーク関数の定義, 詳細な問題性質, どのような性能を計測する意図でその関数は設計されたかといった詳細は, 付録 C を参照されたい. 表 2.2 から, $f_1 \sim f_5$ を除くベンチマーク関数が, 一般的に探索が困難とされる変数分離不可な関数であることがわかる. また, plateau の問題性質以外は $f_1 \sim f_{24}$ によりほぼ均等に網羅されていることがわかる.

以上に述べた利点から, 本論文ではベンチマーク集合として BBOB benchmarks を使用することにした.

[*7] http://coco.gforge.inria.fr/

表 2.2: BBOB benchmarks [91, 93] の 24 個の関数 $f_1 \sim f_{24}$ の大まかな問題性質 [159]. より詳しくは, 付録 C を参照されたい. 各列名について, 「多峰性」は対象問題の多峰性の度合い (2.2.2 節参照), 「大域的構造」は巨視的に見た目的関数値空間の構造 (2.2.3 節参照), 「変数分離可 」は各決定変数が分離可能か否か (2.2.4 節参照), 「変数スケール」は条件数の度合い (2.2.5 節参照), 「一様性」は解空間の異なる領域における各問題性質の類似性 , 「plateau」は目的関数値空間の一部が平坦になる性質を表す.

f	関数名	詳細	多峰性	大域的構造	変数分離可	変数スケール	一様性	plateau
f_1	Sphere	付録 C.2.1 節	none	none	high	none	high	none
f_2	Ellipsoidal	付録 C.2.2 節	none	none	high	high	high	none
f_3	Rastrigin	付録 C.2.3 節	high	strong	high	low	high	none
f_4	Büche Rastrigin	付録 C.2.4 節	high	strong	high	low	high	none
f_5	Linear Slope	付録 C.2.5 節	none	none	high	none	high	none
f_6	Attractive Sector	付録 C.2.6 節	none	none	none	low	med.	none
f_7	Step Ellipsoidal	付録 C.2.7 節	none	none	none	low	high	small
f_8	Rosenbrock	付録 C.2.8 節	low	none	none	none	med.	none
f_9	Rotated Rosenbrock	付録 C.2.9 節	low	none	none	none	med.	none
f_{10}	Ellipsoidal	付録 C.2.10 節	none	none	none	high	high	none
f_{11}	Discus	付録 C.2.11 節	none	none	none	high	high	none
f_{12}	Bent Cigar	付録 C.2.12 節	none	none	none	high	high	none
f_{13}	Sharp Ridge	付録 C.2.13 節	none	none	none	low	med.	none
f_{14}	Different Powers	付録 C.2.14 節	none	none	none	low	med.	none
f_{15}	Rastrigin	付録 C.2.15 節	high	strong	none	low	high	none
f_{16}	Weierstrass	付録 C.2.16 節	high	medium	none	med.	high	none
f_{17}	Schaffers F7	付録 C.2.17 節	high	medium	none	low	med.	none
f_{18}	Schaffers F7-Ill	付録 C.2.18 節	high	medium	none	high	med.	none
f_{19}	Griewank-Rosenbrock	付録 C.2.19 節	high	strong	none	none	high	none
f_{20}	Schwefel	付録 C.2.20 節	medium	deceptive	none	none	high	none
f_{21}	Gallagher 101	付録 C.2.21 節	medium	none	none	med.	high	none
f_{22}	Gallagher 21	付録 C.2.22 節	low	none	none	med.	high	none
f_{23}	Katsuura	付録 C.2.23 節	high	none	none	none	high	none
f_{24}	Bi-Rastrigin	付録 C.2.24 節	high	weak	none	low	high	none

第 3 章

Differential Evolution

本章では, 本論文にて扱う EA の一手法である Differential Evolution (DE) [220, 221] について述べる. DE は初めに 1995 年にテクニカルレポート [220], 次に 1997 年に論文誌論文 [221] として Storn と Price に発表された. 2.4 節にて述べた多くの EA とは異なり, DE は生物進化, 及び特定の生物の振る舞いからではなく, 2.3.1 節にて説明した Nelder-Mead 法の反射操作に着想を得た手法である. DE は関数最適化問題を対象とする多くの EA と同様に, DE の集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の各個体 $\boldsymbol{x}^{i,t}$ $(i \in \{1, ..., N\})$ は, 対象問題の解ベクトル $\boldsymbol{x}^{i,t} = (x_1^{i,t}, ..., x_D^{i,t})^T$ で表現される. ここで, N は集団数であり, D は次元数である. そして, 集団 $\boldsymbol{P}^t$ の個体間の差分ベクトルを使用する突然変異, 及び交叉を用いて新たな子個体 $\boldsymbol{u}^{i,t}$[*1]を生成する. 各個体番号 i において, 個体 $\boldsymbol{x}^{i,t}$ と子個体 $\boldsymbol{u}^{i,t}$ を対比較することにより, 世代交代を行う.

DE は以上のような比較的単純な枠組みでありながら, 他の EA と比べ比較的良好な性能を示すことが報告されている [43]. 例えば, 3.1 節にて説明する基本的な DE アルゴリズムは, 1996 年に開催された, IEEE の進化計算系の会議である IEEE Congress on Evolutionary Computation (CEC) の境界制約付き関数最適化問題のコンペティションでは 3 位, 1997 年では 1 位, 2005 年では 10 次元の部門では 2 位を占めている. その他にも, DE アルゴリズムを改良した手法が, 2006 年と 2010 年の制約付き最適化問題において 1 位, 2009 年の多目的最適化問題では 1 位を占めている. このような背景から, 近年では DE についての研究数が増加している.

本章では始めに 3.1 節にて基本的な DE アルゴリズムについて説明し, 3.2 節にて DE の突然変異戦略, 交叉手法や世代交代モデルといった各オペレータの特徴, 及び性質について説明する. 最後に 3.3 節にて, 不変性, 各交叉手法の性質といった DE の理論的な性質について述べる.

[*1] DE においては子個体を「trial vector」, trial vector を生成する過程で発生する変異個体を「mutant vector」と呼ぶ. しかし, 用語を本論文内にて統一することで可読性を向上させるため, trial vector を子個体, mutant vector を変異個体と呼ぶこととした.

3.1　基本的な Differential Evolution アルゴリズム

本節では基本的な DE アルゴリズム [220, 221] について述べる．始めに 3.1.1 節にて [221] にて記述されている最も古典的かつ基本的な DE アルゴリズムについて説明する．その後, 3.1.2 節にて任意の突然変異戦略, 及び交叉手法を使用できるように一般化した DE アルゴリズムについて述べる．

3.1.1　最も基本的な DE アルゴリズム

本節では, [221] にて DE/rand/1/bin と呼ばれている, 基本的な DE アルゴリズムを説明する (Algorithm 8)．ここで, [220, 221] では DE/X/Y/Z というような表記方法を使用していた．「DE」は DE アルゴリズムであること，「X」は差分突然変異を加えられる個体の選択方法，「Y」は差分突然変異に使用する差ベクトルの数，「Z」は交叉手法である．しかし, この表記方法は柔軟性に乏しく近年においてはあまり使われていないため, 本論文では使用しない．

まず, 探索の開始時 $t = 1$ において, 各個体 $\boldsymbol{x}^{i,t}, i \in \{1, ..., N\}$ は探索領域内 $\boldsymbol{S} = [x_j^{\min}, x_j^{\max}]^D, j \in \{1, ..., D\}$ に一様ランダムに初期化される (Algorithm 8, 1 行目):

$$x_j^{i,t} = (x_j^{\max} - x_j^{\min}) \, \mathrm{rand}[0, 1] + x_j^{\min} \tag{3.1}$$

ここで, rand$[0, 1]$ は区間 $[0, 1]$ の一様分布に従う乱数である．集団の初期化後, 以下に述べる突然変異戦略を用いた変異個体の生成, 親個体と変異個体の交叉による子個体の生成, 生存選択といった手順を, 探索の終了条件を満たすまで繰り返す．ここで, ユーザが定めた計算資源 (最大評価回数, 実行時間など) を使い切った場合, あるいはユーザが許容できる精度の解が求まった場合などが, 一般的な探索終了条件である．

各世代 t において, 個体 $\boldsymbol{x}^{i,t}$ ごとに変異個体 $\boldsymbol{v}^{i,t}$ を集団中の複数の個体に突然変異戦略を適用することで生成する (Algorithm 8, 4 行目):

$$\boldsymbol{v}^{i,t} = \boldsymbol{x}^{r_1,t} + F_{i,t} \left(\boldsymbol{x}^{r_2,t} - \boldsymbol{x}^{r_3,t} \right) \tag{3.2}$$

ここで, スケール係数 $F_{i,t} \in (0, 1]$ は突然変異の大きさを調整する制御パラメータである．通常の DE では, 特に断りの無い限り, $F_{i,t} = F$ と全ての個体で同一の値を取る．r_1, r_2, r_3 は i と互いと異なるように $\{1, ..., N\}$ からランダムに選択した個体番号である[*2]．また, 式 (3.2) は rand/1 と呼ばれる突然変異戦略である．

次に, 親個体 $\boldsymbol{x}^{i,t}$ と変異個体 $\boldsymbol{v}^{i,t}$ に binomial 交叉を適用することで, 子個体 $\boldsymbol{u}^{i,t}$ を生成する (Algorithm 8, 5 行目)．Binomial 交叉を Algorithm 7 に示す．ここで, rand$[0, 1]$ は $[0, 1]$ 区間内に一様ランダムに生成した乱数, randi$[1, D]$ は $\{i, ..., D\}$ からランダムに選択した決定変数番号である．Algorithm 7 に示す binomial 交叉において, 変異個体 $\boldsymbol{v}^{i,t}$ の最低でも 1 変数

[*2] DE の提案論文 [221] では, 差ベクトルが加えられる $\boldsymbol{x}^{r_1,t}$ を base vector と呼んでいた．しかし, この呼称は近年ではほとんど使われていないため, 本論文でも特に使用しないこととした．

Algorithm 7: Binomial 交叉

```
1  j_rand ← randi[1, D];
2  for j = 1 to D do
3      if rand[0, 1) < C_{i,t} 又は j == j_rand then
4          u_j^{i,t} = v_j^{i,t};
5      else
6          u_j^{i,t} = x_j^{i,t};
```

Algorithm 8: [221] にて述べられている基本的な DE アルゴリズム

```
1  t ← 1, 集団 P^t = {x^{1,t}, ..., x^{N,t}} の初期化;
2  while 探索終了条件を満たしていない do
3      for i = 1 to N do
4          式 (3.2) により変異個体 v^{i,t} を生成;
5          x^{i,t} と v^{i,t} に Algorithm 7 の binomial 交叉を適用し, 子個体 u^{i,t} を生成;
6      for i = 1 to N do
7          if f(u^{i,t}) ≤ f(x^{i,t}) then
8              x^{i,t+1} = u^{i,t};
9          else
10             x^{i,t+1} = x^{i,t};
11     t ← t + 1;
```

が子個体 $\boldsymbol{u}^{i,t}$ に受け継がれることが, j_{rand} により保証されている. また, 交叉率 $C_{i,t} \in [0, 1]$ は親個体 $\boldsymbol{x}^{i,t}$ と変異個体 $\boldsymbol{v}^{i,t}$ のどちらの要素をより多く子個体 $\boldsymbol{u}^{i,t}$ に継承させるかを調整する制御パラメータである. 先述の F と同様に, 通常の DE では特に断りの無い限り, $C_{i,t} = C$ と全ての個体で同一の値を取る. これら 2 つの特徴が, GA における一様交叉と異なる点である.

全ての個体が子個体を生成した後, 次世代 $t+1$ に残る個体を決定する (Algorithm 8, 6 ∼ 10 行目). DE では, 各個体 $\boldsymbol{x}^{i,t}$ と子個体 $\boldsymbol{u}^{i,t}$ を比較し, 優れている方の個体が次世代 $t+1$ の集団に残る:

$$\boldsymbol{x}^{i,t+1} = \begin{cases} \boldsymbol{u}^{i,t} & \text{if } f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t}) \\ \boldsymbol{x}^{i,t} & \text{otherwise} \end{cases} \tag{3.3}$$

3.1.2　一般化した基本的な DE アルゴリズム

3.1.1 節では, [221] にて述べられている基本的な DE アルゴリズムを説明した. DE の提案論文 [221] では, 突然変異戦略に rand/1, 交叉手法に binomial 交叉を使用することを前提としていたが, これまでに様々なオペレータが提案されている [220, 59, 188, 267, 43]. そこで, 本節では任意の突然変異戦略, 及び交叉手法を使用できるように一般化した基本的な DE アルゴリズムについて述べる.

Algorithm 8 を一般化した基本的な DE アルゴリズムを Algorithm 9 に示す. また, 近年代表的な突然変異戦略を表 3.1 に示す. Algorithm 9 の 3 行目において, 表 3.1 のいずれかの突然変異戦略を使用することで, 変異個体 $\boldsymbol{v}^{i,t}$ を生成する.

r_1, r_2, r_3, r_4, r_5 は i と互いと異なるように $\{1, ..., N\}$ からランダムに選択した個体番号である. $\boldsymbol{x}^{best,t}$ は世代 t における集団 $\boldsymbol{P}^t$ において, 最も低い目的関数値を有する個体, つまり $f(\boldsymbol{x}^{best,t}) \leq f(\boldsymbol{x}^{i,t}), \forall \boldsymbol{x}^{i,t} \in \boldsymbol{P}^t$ である. $\boldsymbol{x}^{pbest,t}$ は集団 $\boldsymbol{P}^t$ を評価値に基づき降順に並び替え, 上位 $\max(\lfloor N \times p \rfloor, 2)$ 個体からランダムに選択した個体である. また, $p \in [0, 1]$ である. current-to-$p$best/1 と rand-to-$p$best/1 の $\tilde{\boldsymbol{x}}^{r_2,t}$, $\tilde{\boldsymbol{x}}^{r_3,t}$ は, 集団 $\boldsymbol{P}^t$ とアーカイブ $\boldsymbol{A}^t$ の集合からランダムに選択した個体である. 式 (3.3) の生存選択にて, 子個体 $\boldsymbol{u}^{i,t}$ との比較に敗れた親個体 $\boldsymbol{x}^{i,t}$ はアーカイブ $\boldsymbol{A}^t = \{\boldsymbol{y}^{1,t}, ..., \boldsymbol{y}^{A,t}\}$ に追加される. また, アーカイブ $\boldsymbol{A}^t$ に属する個体数がアーカイブサイズ A を超えた場合は, $\{1, ..., A\}$ からランダムに選択した個体がアーカイブ $\boldsymbol{A}^t$ から除外される.

Algorithm 9 の 4 行目において, $\boldsymbol{x}^{i,t}$ と $\boldsymbol{v}^{i,t}$ に Algorithm 7 の binomial 交叉, Algorithm 10 の exponential 交叉, 後の 8 章にて述べる Algorithm 22 の Shuffled Exponential Crossover (SEC) のいずれかの交叉手法を適用し, 子個体 $\boldsymbol{u}^{i,t}$ を生成する.

3.2　DE の各オペレータの特徴

本節では, DE の突然変異戦略, 交叉手法や世代交代モデルといった各オペレータの特徴, 及び性質について説明する. 3.2.1 節にて突然変異戦略, 3.2.2 節では交叉手法について述べる. 3.2.3 節では DE の世代交代モデルについて説明する.

3.2.1　突然変異戦略の特徴, 及び性質

他の EA と比べた際の特徴

GA や ES の遺伝的オペレータは生物進化から, PSO の速度更新方法は鳥などの群の振る舞いに着想を得たのに対し, DE の差分突然変異は Nelder-Mead 法 [167] の反射操作に着想を得ている [221]. そのため, DE は EA に分類されるものの, 探索中に確率モデルを構築し新たな個体をサンプリングする EDA [201, 206] と同様に, 生物進化から直接発想を得た手法ではない.

一般的な GA では交叉の必要性が問われているが [216], 2.4.2 節にて述べた一般的な実数値

Algorithm 9: Algorithm 8 を一般化した基本的な DE アルゴリズム

```
t ← 1, 集団 P^t = {x^{1,t}, ..., x^{N,t}} の初期化;
while 探索終了条件を満たしていない do
    for i = 1 to N do
        表 3.1 の任意の突然変異戦略により変異個体 v^{i,t} を生成;
        x^{i,t} と v^{i,t} に Algorithm 7 の binomial 交叉, Algorithm 10 の exponential 交叉
        のいずれかの交叉手法を適用し, 子個体 u^{i,t} を生成;
    for i = 1 to N do
        if f(u^{i,t}) ≤ f(x^{i,t}) then
            x^{i,t+1} = u^{i,t};
        else
            x^{i,t+1} = x^{i,t};
    t ← t + 1;
```

表 3.1: DE の代表的な 8 つの突然変異戦略. 通常の DE では特に断りの無い限り, $F_{i,t} = F$ と全ての個体で同一の値を取る.

突然変異名	定義
rand/1	$\boldsymbol{v}^{i,t} = \boldsymbol{x}^{r_1,t} + F_{i,t}\left(\boldsymbol{x}^{r_2,t} - \boldsymbol{x}^{r_3,t}\right)$
rand/2	$\boldsymbol{v}^{i,t} = \boldsymbol{x}^{r_1,t} + F_{i,t}\left(\boldsymbol{x}^{r_2,t} - \boldsymbol{x}^{r_3,t}\right) + F_{i,t}\left(\boldsymbol{x}^{r_4,t} - \boldsymbol{x}^{r_5,t}\right)$
best/1	$\boldsymbol{v}^{i,t} = \boldsymbol{x}^{best,t} + F_{i,t}\left(\boldsymbol{x}^{r_1,t} - \boldsymbol{x}^{r_2,t}\right)$
best/2	$\boldsymbol{v}^{i,t} = \boldsymbol{x}^{best,t} + F_{i,t}\left(\boldsymbol{x}^{r_1,t} - \boldsymbol{x}^{r_2,t}\right) + F_{i,t}\left(\boldsymbol{x}^{r_3,t} - \boldsymbol{x}^{r_4,t}\right)$
current-to-rand/1	$\boldsymbol{v}^{i,t} = \boldsymbol{x}^{i,t} + F_{i,t}\left(\boldsymbol{x}^{r_1,t} - \boldsymbol{x}^{i,t}\right) + F_{i,t}\left(\boldsymbol{x}^{r_2,t} - \boldsymbol{x}^{r_3,t}\right)$
current-to-best/1	$\boldsymbol{v}^{i,t} = \boldsymbol{x}^{i,t} + F_{i,t}\left(\boldsymbol{x}^{best,t} - \boldsymbol{x}^{i,t}\right) + F_{i,t}\left(\boldsymbol{x}^{r_1,t} - \boldsymbol{x}^{r_2,t}\right)$
current-to-$p$best/1	$\boldsymbol{v}^{i,t} = \boldsymbol{x}^{i,t} + F_{i,t}\left(\boldsymbol{x}^{pbest,t} - \boldsymbol{x}^{i,t}\right) + F_{i,t}\left(\boldsymbol{x}^{r_1,t} - \tilde{\boldsymbol{x}}^{r_2,t}\right)$
rand-to-$p$best/1	$\boldsymbol{v}^{i,t} = \boldsymbol{x}^{r_1,t} + F_{i,t}\left(\boldsymbol{x}^{pbest,t} - \boldsymbol{x}^{r_1,t}\right) + F_{i,t}\left(\boldsymbol{x}^{r_2,t} - \tilde{\boldsymbol{x}}^{r_3,t}\right)$

GA においては交叉が主探索オペレータであり, 突然変異は集団内の個体の多様性維持のための補助的なオペレータである (2.4.2 節参照). そのため, 洗練された実数値 GA においては交叉のみで十分な探索を行えるため, 突然変異を使用していない [175, 243, 46]. 一方, DE における差分突然変異は交叉と同等に重要な主探索オペレータである.

DE の突然変異戦略では表 3.1 に示すように集団の個体の差ベクトルを利用している. その

Algorithm 10: exponential 交叉

$\boldsymbol{u}^{i,t} = \boldsymbol{x}^{i,t}$, $k \leftarrow 1$, $j \leftarrow \mathrm{randi}[1, D]$;
repeat
　$u_j^{i,t} = v_j^{i,t}$, $j \leftarrow 1 + j$ modulo D, $k \leftarrow k + 1$;
until $\mathrm{rand}[0, 1) < C_{i,t}$ かつ $k < D$;

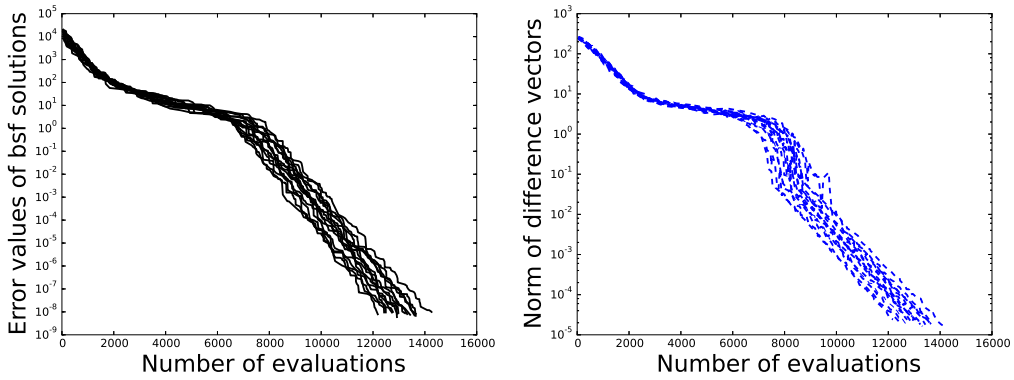
ため, 一様ランダムに集団が初期化されたために個体が探索空間の広範囲に疎に点在する探索序盤では, 個体同士の距離が離れているために差ベクトルが大きい, つまり与えられる摂動が探索空間に対して比較的大きくなる. 一方, 探索が経過するにつれ個体が探索空間の特定領域に収束していくため, 加えられる摂動が比較的小さくなる. そのため, CMA-ES (2.4.4 節参照) のように突然変異の大きさをステップサイズ σ を用いて明示的に制御せずとも, DE では突然変異の大きさを探索の経過に合わせて調整することができる. また, 一般的に実数値 GA における突然変異の大きさは探索状況に依らず一定であるが [48], DE では先述のように探索状況に応じて突然変異の大きさが変動する.

以上の説明の実例として, 通常の DE アルゴリズム (Algorithm 8 参照) を 10 次元の Rastrigin 関数 $f(\boldsymbol{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos 2\pi(x_i) + 10\right)$ に適用した際の, (a) 最良解 $\boldsymbol{x}^{bsf}$ と最適解 $\boldsymbol{x}^*$ の目的関数の誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$, 及び (b) 各世代における差ベクトルのユークリッドノルム $\|(\boldsymbol{x}^{r_1,t} - \boldsymbol{x}^{r_2,t})\|$ の平均値の推移を図 3.1 に示す. なお, 実行可能領域は $[-5.12, 5.12]^D$ である. 探索が経過するにつれ, 差分突然変異の大きさもほぼ単調に縮小していることが, 図 3.1 からわかる.

PSO では式 (2.28) に示すように現探索点からの各個体の過去の最良探索点 $\boldsymbol{x}^{pbest,i,t}$, 及び全ての個体の過去の最良探索点 $\boldsymbol{x}^{gbest,t}$ に対する差ベクトルを使用する. そのため, 個体間の差ベクトルを利用するという点では, DE の差分突然変異は PSO の速度ベクトルの更新に類似している. しかし, PSO では慣性項 w を用いて 1 世代前の移動方向を考慮するのに対して, DE では前世代の移動方向を特に利用しない. また, PSO では各決定変数ごとに独立して一様乱数を発生させ移動幅を決定するのに対し, DE ではスケール係数 F により全ての決定変数の移動幅のスケールは同一である.

各突然変異戦略の特徴

表 3.1 に示した, rand/1, rand/2, best/1, best/2, current-to-rand/1, current-to-best/1, current-to-$p$best/1 [267], rand-to-$p$best/1 [266] の 8 つの突然変異戦略の特徴について述べる. 変異個体 $\boldsymbol{v}^{i,t}$ を, 最良個体 $\boldsymbol{x}^{best,t}$ の付近に生成する best/1 と best/2, 対象個体から $\boldsymbol{x}^{best,t}$ に向かうように生成する current-to-best/1 は, rand/1, rand/2 などに比べてより局所的探索能力が強い戦略である. 反対に, rand/1, rand/2 のように集団からランダムに選択した個体 $\boldsymbol{x}^{r_1,t}$ の付近, 及び $\boldsymbol{x}^{r_1,t}$ への方向に変異個体 $\boldsymbol{v}^{i,t}$ を生成する current-to-rand/1 は, 大域的探索能力に優れた戦略である. rand/1, rand/2, best/1, best/2 と比べ, current-to-rand/1 と current-to-best/1 は親個体 $\boldsymbol{x}^{i,t}$ に差ベクトルを加えるため, $\boldsymbol{x}^{i,t}$ 付近に変異個体 $\boldsymbol{v}^{i,t}$ を生



(a) $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ の推移　(b) $\|(\boldsymbol{x}^{r_1,t} - \boldsymbol{x}^{r_2,t})\|$ の推移

図 3.1: 通常の DE アルゴリズム (Algorithm 8 参照) を 10 次元の Rastrigin 関数に適用した際の, (a) 最良解 $\boldsymbol{x}^{bsf}$ と最適解 $\boldsymbol{x}^*$ の目的関数の誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$, 及び (b) 各世代における差ベクトルのユークリッドノルム $\|(\boldsymbol{x}^{r_1,t} - \boldsymbol{x}^{r_2,t})\|$ の平均値の推移. 突然変異戦略に rand/1, 交叉に binomial 交叉を使用し, $N = 50$, $F = 0.5$, $C = 0.1$ とした. 横軸は評価回数の経過を表し, 15 試行分のデータを示している. なお, 15 試行全てにおいて最適解を求めることに成功している.

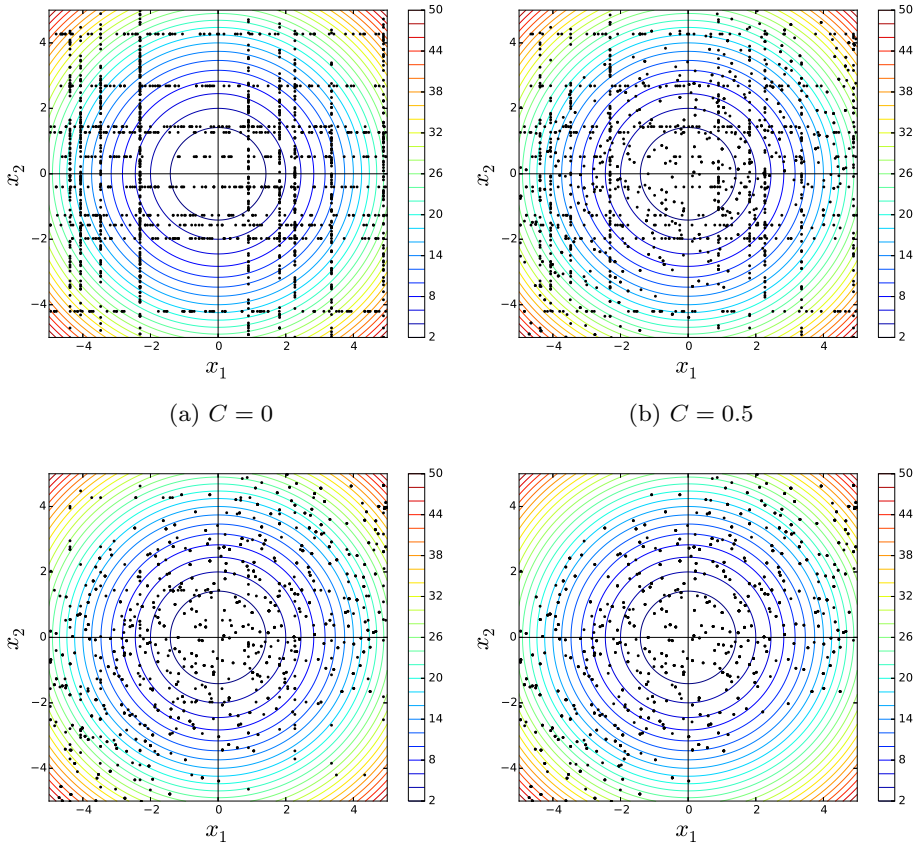
成しやすいという特徴がある. また, DE の突然変異戦略では, rand/1 より rand/2, best/1 より best/2 の方といったように, 加える差ベクトルの数が多い程, より多様性に富んだ変異個体 $\boldsymbol{v}^{i,t}$ を生成しやすい.

current-to-$p$best/1 は current-to-rand/1 と current-to-best/1 を, 制御パラメータ p を導入することで一般化した突然変異戦略である. $p \in [0, 1]$ は局所的探索能力と大域的探索能力を調整するパラメータであり, $p = 1$ では current-to-rand/1, $p = 0$ ではほぼ current-to-best/1 となる. そのため, current-to-$p$best/1 は p を適切に設定することで, 探索能力のバランスが取れた効率的な突然変異戦略となり得る. これは rand-to-$p$best/1 についても同様のことが言える. また, サイズ $A > 1$ のアーカイブ $\boldsymbol{A}$ を使用した場合, 単純に生成可能な差ベクトル $(\boldsymbol{x}^{r_1,t} - \boldsymbol{x}^{r_2,t})$ の数が $(N-2)(N-3)$ から $(N-2)(N+A-3)$ へと増加するので, 多様な解を生成しやすい.

表 3.1 以外にも, 各個体の目的関数値の重み付けを用いた trigonometric mutation [59], 一定の確率で使用する突然変異を変更する either-or [188], リングトポロジーを用いた突然変異戦略 [40], ランク選択を用いた突然変異 [224, 78] などが, これまでに提案されている.

3.2.2 DE における交叉の種類と交叉率 C の影響

DE の代表的な交叉手法である binomial 交叉と exponential 交叉は, それぞれ GA における一様交叉と 1, 2 点交叉に類似している. 両者の相違点は, 遺伝される決定変数の数の分布 (3.3.2 節参照), 及び遺伝される決定変数が変数番号上において連続的か否かである [188, 140]. 比較的初期の研究では, binomial 交叉は exponential 交叉と比べ優れていると報告する研究が多い [220, 161]. 2005 年 ~2010 年頃までの DE に関する研究においても, 多くが binomial 交



(a) $C = 0$　(b) $C = 0.5$

(c) $C = 0.9$　(d) $C = 1$

図 3.2: 2 次元の Sphere 関数において, 世代交代を行わない DE アルゴリズムにより生成された 2,000 個の子個体の分布. 突然変異戦略に rand/1, 交叉に binomial 交叉を使用し, 集団数 $N = 10$, $F = 0.5$ とした. (a), (b), (c), (d) にて, それぞれ C を 0, 0.5, 0.9, 1.0 としている. なお, [224] の Figure 1, 及び [43] の Figure 4 を参考に再実験を行い, 新たに作成した.

叉を使用している [196, 27, 28, 257, 267, 189]. 一方, 近年では exponential 交叉の有用性を報告する研究が数多く存在する [170, 99, 29, 269, 128, 268]. 特に, Soft Computing Journal の $D = 50, 100, 200, 5000, 1,000$ 規模の高次元関数最適化問題に対する EA の特集号[*3]に掲載された SOUPDE [251], DE-D^{40} + M$^{\mathrm{m}}$ [73], GODE [246], GaDE [258], jDElscop [29], SaDE-MMTS [269], MOS [128] といった 7 つの DE アルゴリズムは, 全て exponential 交叉を使用している. この結果から, Zhao と Suganthan は高次元問題においては exponential 交叉の方が binomial 交叉よりも優れていると結論づけている [268].

交叉率 C は, 親個体 $\boldsymbol{x}^{i,t}$ と変異個体 $\boldsymbol{v}^{i,t}$ の, どちらの要素を子個体 $\boldsymbol{u}^{i,t}$ により多く受け継がせるかを制御する. C が低ければ, 親個体 $\boldsymbol{x}^{i,t}$ の要素が多く子個体 $\boldsymbol{u}^{i,t}$ に受け継がれる, つまり, いくつかの決定変数が新たな要素に置き換わる. 一方, C が高ければ, 変異個体 $\boldsymbol{v}^{i,t}$ の

[*3] http://sci2s.ugr.es/EAMHCO (2015 年 11 月 10 日確認)

要素が多く子個体 $\boldsymbol{u}^{i,t}$ に受け継がれる, つまり, 多くの決定変数が新たな要素に置き換わる. $C = 0.0$ では, 1 次元ごとの探索が可能となるため変数分離化な多峰性関数において適切な設定である. 一方, $C = 1.0$ では軸変換に対する回転不変性 (3.3.1 節参照) を持つため, 変数分離不可な問題において適切な設定である.

C の影響を確認するため, 図 3.2 に, $D = 2$ の Sphere 関数 $f(\boldsymbol{x}) = \sum_{i=1}^{D} x_i^2$ において世代交代を行わない DE アルゴリズムにより生成された 2,000 個の子個体の分布を示す. ここで, 図 3.2 は, [224] の Figure 1, 及び [43] の Figure 4 を参考に再実験を行い作成した. 突然変異戦略に rand/1, 交叉に binomial 交叉を使用し, 集団数 $N = 10$, $F = 0.5$ とした. 図 3.2(a), (b), (c), (d) にて, それぞれ C を 0, 0.5, 0.9, 1.0 としている. C の値が低いほど垂直, 又は水平方向に並んだ子個体が生成されやすいが, C の値が増加するにつれこの傾向が見られなくなることが, 図 3.2 よりわかる.

3.2.3 DE における世代交代モデル

DE においては, 個体番号 $i \in \{1, ..., N\}$ ごとに親個体 $\boldsymbol{x}^{i,t}$ と子個体 $\boldsymbol{u}^{i,t}$ を比較する. そして, 子個体が親個体よりも優れていた場合のみ, つまり $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ であった場合のみ, $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$ となる. そのため, DE の世代交代モデルは探索中に得られた最良個体 (最良解) を現集団 $\boldsymbol{P}^t$ に常に保存する, エリート戦略である. また, 各個体番号 i ごとに子個体の生成と世代交代を独立して行うため並列性が高い [190].

このような親個体と子個体の対比較に基づく世代交代は, GA の枠組みにおいて, 多峰性関数での複数の局所解を得るための手法であるニッチング法の枠組みで 1995 年に提案された Deterministic Crowding [155], 及びそれを GPU を用いた超並列計算機環境向けに洗練させたモデル [241] においても採用されている. また, EDA [239] と ACO [240] の枠組みにおいても, 類似した世代交代モデルが提案されている.

エリート戦略である点に関して, 更新した探索点の目的関数値を考慮しない PSO (2.4.5 節参照) や, 探索点を保持しない CMA-ES (2.4.4 節参照) といった非エリート戦略の EA とは異なる. 同じエリート戦略という意味では, G3 を始めとする steady state 型の GA の世代交代モデル (2.4.1, 2.4.2 節参照) と類似点がある. しかし, GA の多くの世代交代モデルは集団中から置き換え候補個体を選択するため, 選択圧が強く, 集団中の個体の多様性が失われやすい [205]. 一方, DE では親個体の要素をある程度受け継いだ子個体により親個体を置き換えるため, エリート保存戦略でありながら集団中の個体の多様性を維持することが期待できる. また, DE の世代交代は親個体 $\boldsymbol{x}^{i,t}$ と子個体 $\boldsymbol{u}^{i,t}$ の目的関数値の比較により決定的に行われる. これに対して, SA (付録 A.2.3 節参照) の改悪解への移動を確率的に許容する戦略を DE の世代交代モデルに導入することで, 多峰性関数における DE の探索性能を向上できるという報告がある [42].

式 (3.3) に示す通り, $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ と, 子個体と親個体の目的関数値が等しい場合においても, $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$ となる. このため, 目的関数値空間の一部が平坦になる性質 (plateau) を含んでいたとしても, DE は停滞することなく探索を行える. 実際, 式 (3.3) において, $\leq$ を $<$

とした場合, plateau の性質を含む問題では探索性能が劣化する [188].

3.3　DE の理論的な性質

3.2 節では, DE の突然変異戦略, 交叉手法や世代交代モデルといった各オペレータの特徴, 及び性質を関連文献, 及び関連手法をあげながら説明した. 対して, 本節では DE の理論的な性質について述べる.

始めに 3.3.1 節にて探索空間の変換操作に対する不変性について述べ, 3.3.2 節では binomial 交叉と exponential 交叉の性質について説明する. なお, Zaharie [262] による集団数 N, スケール係数 F, 交叉率 C の関係とフラットランドスケープにおける収束性の解析については, 付録 B 章を参照されたい.

3.3.1　不変性

目的関数, 及び探索空間の変換について不変性を持つ手法は, 変換後の問題においても同様の探索性能を有することが保証されている. 例えば問題 A と変換後の問題 A' において, もし行った変換操作に対して対象手法が不変性を有するのであれば, A と A' での探索性能は変わらない. つまり, 問題 A に対する理論的, 及び実験的な方法にて得られた結果を, 特別な手順を加えること無く問題 A' に適用できる. そのため, 関数最適化問題に対する探索手法では, 不変性は重要な性質である [86, 74]. ただし, [10] において述べられているように, 不変性を持つ手法が, 持たない手法よりも探索性能が優れているとは限らない.

DE は以下の 3 つの関数最適化問題において重要な不変性を有する:

1. 探索空間の線形変換に対する不変性 (linear transformation)
2. 順序を保持した目的関数値の変換に対する不変性 (order-preserving transformation)
3. 座標軸の回転に対する不変性 (rotation transformation)

ここで, 上記全ての不変性を有する手法には, DE 以外にも CMA-ES があげられる [86, 10].

探索空間の線形変換に対する不変性 (不変性 1) については, 例えば探索領域 $\boldsymbol{S} = [x_j^{\min}, x_j^{\max}]^D, j \in \{1, ..., D\}$ を, $\alpha > 0$ について $\boldsymbol{S} = [\alpha \cdot x_j^{\min}, \alpha \cdot x_j^{\max}]^D$ のように α 倍拡大したとしても, 探索過程が不変となる性質である. DE, 及び 2 章にて紹介したほとんどの手法が, この不変性を有する [*4].

順序を保持した目的関数値の変換に対する不変性 (不変性 2) については, DE は個体間の目的関数値の比較に基づき探索を行う. 例えば, 探索終了までに生成した解を, 各目的関数値により $f(\boldsymbol{x}^1) \leq f(\boldsymbol{x}^2) \leq ...$ のように並び替えたとする. この時, 単調関数 g について目的関数 f を $g \circ f = h$ と変換したとしても, $h(\boldsymbol{x}^1) \leq h(\boldsymbol{x}^2) \leq ...$ と順序は保持されているので, DE の探索過程は変換後の関数 h においても変わらない. DE のみではなく, ほとんどの EA がこの

[*4] 例えば, $\beta > 0$ について $\boldsymbol{u}^t = \boldsymbol{x}^t + \beta\,\mathrm{randn}(0, 1)$ として新たな探索点 $\boldsymbol{u}^t$ を生成する手法は, (1) の不変性を持たない. 一方, $\boldsymbol{u}^t = \boldsymbol{x}^t + \beta\,(\boldsymbol{x}^{\max} - \boldsymbol{x}^{\min})\,\mathrm{randn}(0, 1)$ とするような手法は (1) の不変性を有する.

性質を有する [*5]. 一方, A.1 節にて紹介した BFGS を始めとする目的関数の勾配情報を利用する探索手法は, この不変性を持たない [10].

最後に 座標軸の回転に対する不変性 (不変性 3) とは, $D \times D$ の回転行列 $\boldsymbol{R}$ について, $f : \boldsymbol{x} \to f(\boldsymbol{x})$ と $f_{\boldsymbol{R}} : \boldsymbol{Rx} \to f(\boldsymbol{Rx})$ においても, 探索過程が変わらない性質である. 回転不変性を有する探索手法は, 変数分離可な関数, 及び変数分離不可な関数においても探索性能は変わらない. 2.4 節にて述べた EA の中では, SPX と PCX を用いた実数値 GA, 及び CMA-ES が回転不変性を有するが, ビットストリング GA と PSO はこの性質を持たない. DE は使用する交叉手法に依らず $C = 1$ とした場合, この回転不変性を有する [188].

3.3.2 交叉

本節では, 3.1 節にて紹介した DE の binomial 交叉と exponential 交叉の理論的な性質について, 主に [188, 263, 140] を参考に述べる.

Binomial 交叉では, D 回の独立したベルヌーイ試行を行い, 変異個体 $\boldsymbol{v}^{i,t}$ から子個体 $\boldsymbol{u}^{i,t}$ に受け継がれる決定変数を決める. ここで, 受け継がれる決定変数の数を確率変数 L とおくと, L の確率関数 $P(L = h)$ は D と C により決定される二項分布となる [188, 263]. すなわち, $P(L = h) = {}_h\mathrm{C}_D(C)^h(1 - C)^{D-h}$ である. ただし, Algorithm 7 の 3 行目の j_{rand} の存在のため, 必ず 1 つの決定変数が $\boldsymbol{v}^{i,t}$ から受け継がれるので, $P(L = h)$ は正しくは次のとおりである [140]:

$$P(L = h) = {}_{h-1}\mathrm{C}_{D-1}C^{h-1}(1 - C)^{D-h} \tag{3.4}$$

ただし, 式 (3.4) は 3 次元以上において成り立つ.

変異個体 $\boldsymbol{v}^{i,t}$ の各決定変数 v_j^t ($j \in \{1, ..., D\}$) が子個体に受け継がれる確率 p_m は, 単純に考えると C である. しかし, 先述と同様に Algorithm 7 の 3 行目の j_{rand} の存在のため, 必ず $\boldsymbol{v}^{i,t}$ から 1 つの決定変数が, $1/D$ の確率で受け継がれることが保証されているため, 実際には次式のようになる [263]:

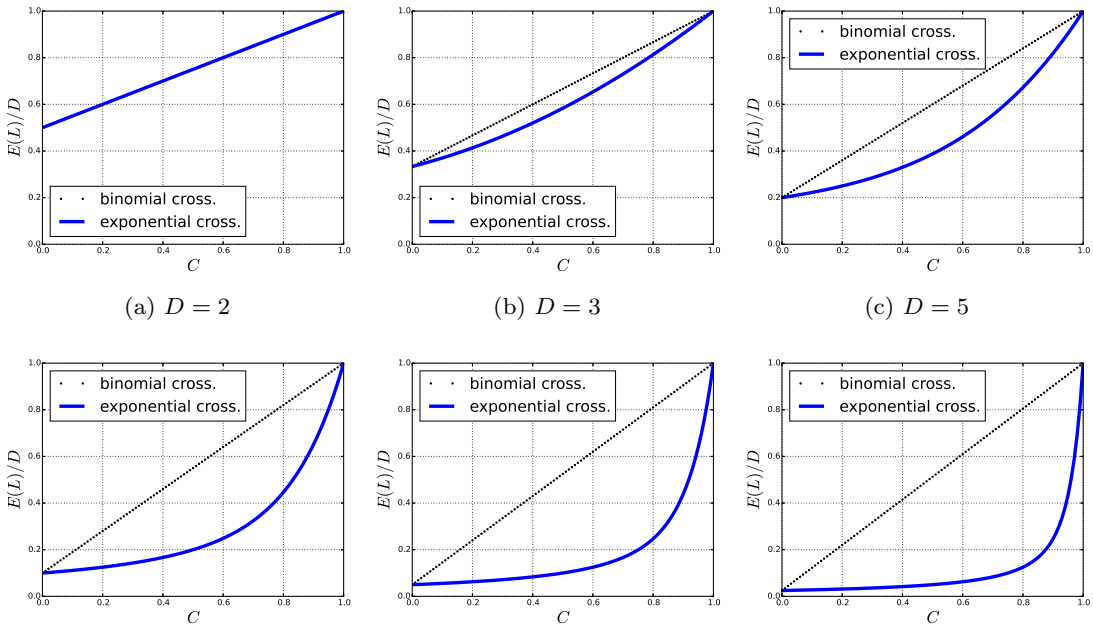
$$p_m = C\left(1 - \frac{1}{D}\right) + \frac{1}{D} \tag{3.5}$$

また, 確率 p の事象を n 回行う二項分布の期待値は np となるため, L の期待値 $E(L)$ は次のように求まる [263]:

$$\begin{aligned} E(L) &= D \cdot p_m \\ &= (D - 1)C + 1 \end{aligned} \tag{3.6}$$

次に, exponential 交叉について考える. exponential 交叉では, 確率 C で起こりえる成功事象, 及び確率 $1 - C$ で起こる失敗事象から構成されるベルヌーイ試行を, 失敗事象が発生する,

[*5] (2) の不変性を持たない EA の例として, 目的関数値を直接利用する式 (2.7) のルーレット選択を用いた GA があげられる [74]. 同様の理由で ABC [116] もこの不変性を持たない. 一方, 式 (2.8) のランキング選択を用いた GA は, (2) の不変性を有する.



(a) $D = 2$　(b) $D = 3$　(c) $D = 5$

(d) $D = 10$　(e) $D = 20$　(f) $D = 40$

図 3.3: $D = 2, 3, 5, 10, 20, 40$ における, binomial 交叉と exponential 交叉の C (横軸) に対する交叉長 L の期待値 $E(L)$ (縦軸) の推移. $E(L)$ は次元数 D にて正規化している. なお, [263] の Figure 1, 及び [140] の Figure 10 を参考に新たに作成した.

又は試行回数が D となるまで繰り返す. そのため, exponential 交叉の L の確率関数は幾何分布に従い, $P(L = h) = (1 - C)C^h$ となる [263]. しかし, Algorithm 10 に示す exponential 交叉では, $\{1, ..., D\}$ から一様ランダムに選択した交叉の始点となる決定変数は, 必ず子個体 $\boldsymbol{u}^{i,t}$ に受け継がれる. そのため, $P(L = h) = (1 - C)C^{h-1}$ と修正する必要がある. また, Algorithm 10 において $L = D$ となった場合, 試行が停止することを考慮しなくてはならない. 以上をまとめた, exponential 交叉の L の確率関数は次のように与えられる:

$$P(L = h) = \begin{cases} (1 - C)C^{h-1} & \text{if } h < D \\ C^{D-1} & \text{otherwise (if } h = D) \end{cases} \tag{3.7}$$

また, 期待値 $E(L)$ は式 (3.7) より, 以下のように求まる:

$$\begin{aligned} E(L) &= \left((1 - C) \sum_{h=1}^{D-1} \left(h\, C^{h-1} \right) \right) + D \left(C^{D-1} \right) \\ &= \frac{1 - C^D}{1 - C} \end{aligned} \tag{3.8}$$

最後に, $D = 2, 3, 5, 10, 20, 40$ における, binomial 交叉と exponential 交叉の C に対する $E(L)$ の推移を, 図 3.3 に示す. なお, 図 3.3 は [263] の Figure 1, 及び [140] の Figure 10 を参考に新たに作成した. 図 3.3 から, $D = 2$ においては binomial 交叉と exponential 交叉の $E(L)$ は完全に一致していることがわかる. 一方, 次元数の増加に伴いそれぞれの交叉手法の

特徴が明確になる．binomial 交叉においては C と $E(L)$ は線形であるが, exponential 交叉では非線形である．そのため, 同一の C 値においても次元数によって $E(L)/D$ が大きく異なる．特に, $D = 20, 40$ においては, $C = 0.8$ 付近を堺に $E(L)/D$ が急激に増加していることがわかる．この結果は, exponential 交叉においては対象問題の次元数に応じて C を適切に調整する必要があり, 値の範囲によっては $E(L)$ が C に対して鈍感, 又は敏感となりえることを意味している．

第 4 章

EA, 及び DE におけるパラメータ設定問題とパラメータ制御

3 章で説明した DE は, 比較的単純なアルゴリズムでありながら優れた探索性能を有することが報告されている. 一方, DE の探索性能は自身の制御パラメータ (集団数 N, スケール係数 F, 及び交叉率 C) の設定に大きく依存し, 対象問題ごとに適切に設定できない場合は良好な結果が望めないという問題を抱えている. このパラメータ設定問題は DE だけではなく, ほぼ全ての EA が抱えている問題である.

本章ではパラメータ設定問題に対する解決策である EA, 及び DE におけるパラメータ制御について主に述べる. これにより, 後述の 5 ～ 8 章の準備とする.

始めに, 4.1 節にて DE を除く EA におけるパラメータ設定問題について説明する. 次に, 4.2 節にて, パラメータ設定問題の解決方法であるパラメータチューニング (4.2.1 節) とパラメータ制御 (4.2.2 節) について, それぞれ説明する. 特に後者については, (1) 決定的パラメータ制御, (2) 適応的パラメータ制御, (3) 自己適応的パラメータ制御の 3 つのアプローチについて詳細に述べる. その後, 4.3 節にて DE のパラメータ設定問題について説明し, 4.4 節にて DE におけるパラメータ制御に関する研究をまとめる. 最後に 4.5 節にて, 近年の代表的な適応 DE について述べる.

4.1 進化アルゴリズムにおけるパラメータ設定問題

2015 年現在において, 制御パラメータを一つも持たない EA は, ほぼ存在しない. Harik と Lobo の Parameter-less GA [98] のように, 制御パラメータを有さないと主張する手法はこれまでにいくつか提案されている [23, 77]. しかし, ロバストな探索が期待されるパラメータ設定 (集団数や突然変異率など) を EA の設計者が提示することにより, ユーザが設定すべきパラメータ設定が無いという意味での「Parameter-less」であり, 真の意味での「Parameter-less」ではない.

一般的に, ある最適化問題における EA の探索性能は, 自身の制御パラメータの設定に大きく依存する [54, 146, 56, 119]. 単純な GA を例にとっても, 多峰性の問題では局所解に探索が

誤って収束されないように集団数を十分大きく, また多様性を維持するために突然変異率を高く設定する必要がある. 一方, 単峰性の問題において多峰性問題に適したパラメータ設定, すなわち多様性維持を重視した設定を使用した場合は探索の収束速度が遅いため, 計算資源 (最大評価回数や実行時間など) を多く費やしてしまうこととなる. そのため, 小さな集団数, 及び低い突然変異率が単峰性の問題では好ましい. このように, 対象問題がどのような問題性質を有するのかによって, EA の適切なパラメータ設定は大きく変化する. そのため, EA を最適化ツールとして使用するユーザは, 解きたい対象問題ごとに EA のパラメータ設定を適切に調整する必要がある.

しかし, 実問題においては解の評価にシミュレーションなどを使用するため, 一つの解を評価するのに膨大な計算時間がかかるような問題が存在する [112, 208]. 実例として, 翼形状設計 [174], 回路設計 [141], 車体設計 [129] などがあげられる. それぞれ 1 つの解を評価するのにかかる時間は, [174] では約 11 分, [141] においては 10 ～ 12 分, [129] では約 60 分である. こうした実問題において, パラメータ設定を調整するために膨大な計算資源 (解評価の制限) を使用することは現実的に困難である. この問題はスーパコンピュータなどの超高性能計算機を使用することにより, ある程度解決できるように思われる. しかし, 計算機の高速化に応じてより高精度なシミュレーション (例えば, 超音速機の飛行シミュレーション [171]) が実現可能となるため, 解評価にかかる計算時間は結果としてこれまでと大差が無い場合がある. そのため, 将来においても上記に述べた問題点が本質的に解消されることは, ほとんど無いと考えられる.

理論的に最適なパラメータ設定を得ようという試みは, EA の研究初期においていくつか存在する [13, 76, 101, 37]. しかし, そのようなアプローチは極端に単純な特定の問題においてのみ有用であり, 現実的には有用ではないと Eiben らに指摘されている [54]. 加えて, [213] にて指摘されているように, 考えうる全ての問題に対して最適な EA のパラメータ設定は No Free Lunch の定理 [254] から存在しない. また, 探索序盤であれば有望な探索領域を特定するために大域的な探索が好ましいが, 少しでも精度の高い解を求める能力が必要とされる探索終盤では局所的な探索能力が求められるため, 適切なパラメータ設定は対象問題のみならず EA の探索状況にも依存する. つまり, 最適なパラメータ設定 θ^* は探索を通して固定されず, 世代 t に依存する θ_t^* という形式で与えられる. そのため, 理論的に最適なパラメータ設定を使用したとしても, 後述のパラメータ制御を用いた手法に劣る可能性がある [235, 265, 39].

また, 本研究にて取り扱う black-box optimization 環境 (2.1.4 節参照) においては, 対象問題がどのような景観, 性質を有するのかを事前に知ることはできず, 先の例のように特定の問題性質を有する関数においてのみ効率的なパラメータ設定を使用することは, 危険性が高い.

4.2　パラメータチューニングとパラメータ制御

本節では, パラメータ設定問題に対するアプローチである, パラメータチューニング[*1] [*2]とパラメータ制御について述べる. ここで, パラメータチューニングとパラメータ制御は, 1999 年に行われた Eiben らによるサーベイ論文 [54] で使われた名称, 分類方法であり, 以下のような階層を持つ:

1. パラメータチューニング (parameter tuning)
2. パラメータ制御 (parameter control)
   (1) 決定的パラメータ制御 (deterministic parameter control)
   (2) 適応的パラメータ制御 (adaptive parameter control)
   (3) 自己適応的パラメータ制御 (self-adaptive parameter control)

パラメータチューニングは, 対象問題に EA を適用する以前に, いくつかのベンチマーク関数にて対象問題における EA の性能が向上するようなパラメータ設定を求める方法である. 一方, パラメータ制御は対象問題において EA が探索している最中に, 対象問題, 及び探索状況に適したパラメータ設定を求める方法である. 1990 年代頃までは, パラメータ設定問題に対するアプローチを体系立てて分類する方法がほとんど存在しなかったため, 用語の混同が見られた [54]. しかし, 上記の Eiben らによる分類により, それぞれのアプローチの差異や特徴が明確になった.

本節では, Eiben らによる分類方法に従い, 各アプローチの説明をする. 始めに 4.2.1 節にてパラメータチューニングについて述べた後, 4.2.2 節にてパラメータ制御について説明する.

4.2.1　パラメータチューニング

最適化問題としてのパラメータチューニング

パラメータチューニングは, チューニング対象となる探索アルゴリズム A, A のチューニング対象となる k 個のパラメータ設定の範囲 $\boldsymbol{\Theta} \subseteq \Theta_1 \times ... \times \Theta_k$, 訓練問題集 $\boldsymbol{I} = (I_1, ...)$ を入力として与えられた時, 出力として $\boldsymbol{I}$ における A の探索性能が最も良くなるようなパラメータ設定 $\boldsymbol{\theta}^* = (\theta_1^*, ..., \theta_k^*)$ を求める最適化問題として定式化できる [107]. ここで, パラメータ設定 $\boldsymbol{\theta}$ がどの程度優れているかを効用と呼ぶ. 一般的に, EA においては探索中に得られた最良解の質や最適解に到達するまでに費やした計算資源 (解評価回数や計算時間) にて効用を判定する.

パラメータチューニングを最適化問題として扱った場合, 任意の直接探索手法を適用することで探索が行えるが, 効用を最大化するパラメータ設定 $\boldsymbol{\theta}^*$ を求めることは容易ではない. 例えば GA における突然変異率は実数値型, 集団数は整数型, 交叉オペレータはカテゴリカル

[*1] 現在では, より広義な意味でアルゴリズムコンフィグレーション (algorithm configuration) と呼ばれることもある [107].
[*2] パラメータチューニングはオフラインチューニング, パラメータ制御はオンラインチューニングとも呼ばれることがある [69].

型[*3]で表現される．そのため，A のチューニング対象となるパラメータ設定 $\boldsymbol{\theta}$ は，単一の型のベクトルで表現することが難しい．また，Hooke-Jeeves 法 (A.2.2 節) や Nelder-Mead 法 (2.3.1 節) などの決定的探索手法の場合は問題にならないが，疑似焼きなまし法 (A.2.3 節) や EA のような確率的探索手法では，実行毎に計測される効用が異なる．そのため，例え同一の訓練問題インスタンスにおいても，得られる効用にノイズが含まれる．さらに，パラメータ設定 $\boldsymbol{\theta}$ の効用を計測するためには，実際に対象アルゴリズム A を訓練問題集 $\boldsymbol{I}$ にて実行する必要がある．例えば A を $\boldsymbol{I}$ の 1 問にて 1 回実行するのに 10 秒かかったとする．この時，$|\boldsymbol{I}| = 360$ であった場合は，$\boldsymbol{I}$ の全てのインスタンスにおいて A を実行するのに 1 時間程度かかる．このような場合，パラメータ設定 $\boldsymbol{\theta}$ の効用を計測する上限回数を数万回に設定することは現実的に困難であり，高々数百程度しか設けることができない，つまり限られた計算資源の下で探索を行う必要がある．これらの要因が，パラメータチューニングを困難とさせている．

自動パラメータチューナー

パラメータチューニングのために，これまでに数多くの手法が提案されてきた [56]．ユーザによる手動チューニングを除いた場合，恐らく最も古いアプローチとして 1986 年に Grefenstette によって提案された GA のパラメータ値空間を探索する GA (meta-GA) があげられる [81]．その後，実験計画法 [1] やラテン超方格サンプリング [165] といった少サンプル数における効率的なサンプリング法に基づく枠組みが提案された．

2000 年代の始めに，F-race [20]，Sequential Parameter Optimization (SPO) [18]，Relevance Estimation and VAlue Calibration of EA parameters (REVAC) [166] などの自動パラメータチューナーが開発された．ここで，自動パラメータチューナーでは，チューニング対象アルゴリズム A や訓練問題集 $\boldsymbol{I}$ といった入力を受け取った後，何らかの方法でパラメータ設定 $\boldsymbol{\theta}$ を生成し，$\boldsymbol{\theta}$ を用いた探索アルゴリズム A を訓練問題集 $\boldsymbol{I}$ の中の 1 つ以上の問題インスタンスで実行することで，$\boldsymbol{\theta}$ の効用を判断する．このパラメータ設定の生成と評価を繰り返し，最後にチューナーの探索にて得られた最良のパラメータ設定 $\boldsymbol{\theta}^{best}$ を出力する．

しかし，$\boldsymbol{\theta}$ の要素は先に述べた実数値型，整数型，カテゴリカル型のいずれかの型に分類されるが，SPO を始めとする多くの自動パラメータチューナーはカテゴリカル型を扱えないという欠点があった．これに対して，2009 年に提案された ParamILS [107] は，パラメータ設定の全ての要素をカテゴリカル型に変換して探索を行うことで，全ての型を扱うことが可能であった．ParamILS の成功後，Gender-based GA (GGA) [5]，Iterative F-Race (irace) [148]，Sequential Model-based Algorithm Configuration (SMAC) [106] など様々な自動チューナーが今日に至るまで提案されている．

一般的に，自動チューナーは訓練問題集 $\boldsymbol{I}$ を対象問題の代表的な分布としてみなし，その問題の分布に対してアルゴリズム A の性能がロバストになるようなパラメータ設定を求めることを試みる．そのため，自動チューナーにより得られたパラメータ設定を使用した場合，もし

[*3] 順序の意味や異なるパラメータ間の距離などの情報が明示的に与えられない，離散的なパラメータの型 [148]．DE における突然変異戦略 (表 3.1 の rand/1, rand/2 など) や，交叉手法の種類 (binomial 交叉，exponential 交叉) などがカテゴリカル型のパラメータの例としてあげられる．

ユーザーの解きたい対象問題が訓練問題集 $\boldsymbol{I}$ の構成要素に類似していれば, 実際にアルゴリズム A をチューニングせずとも良好な結果が期待できる. しかし, 4.3 節にて述べるように対象問題, 及び現在の探索状況ごとに探索手法の適切なパラメータ設定が大きく異なる場合, パラメータチューニングによるアプローチが必ずしも成功するとは限らない.

4.2.2　パラメータ制御

1999 年に行われた Eiben らによるサーベイ論文 [54] では, 探索中にパラメータを動的に変化させるパラメータ制御の枠組みを, さらに次の 3 つに分類している:

1. 決定的パラメータ制御 (deterministic parameter control)
2. 適応的パラメータ制御 (adaptive parameter control)
3. 自己適応的パラメータ制御 (self-adaptive parameter control)

以下では, 上記 3 つのパラメータ制御について, それぞれ説明する. なお, 関数最適化問題が対象問題ではない EA も含まれていることに注意されたい. また, 本節では数多くのパラメータ制御手法が描写されるため, 各手法の特徴を把握しづらい可能性がある. そのため, 表 4.1 に本節にて説明する EA の各パラメータ制御手法の概要を示す.

1. 決定的パラメータ制御

決定的パラメータ制御では, 現在の探索状況を考慮せずに, 予め定めたスケジュールに従い決定的にパラメータ設定を変更する. 例えば, GA において探索経過 (実行時間や使用した評価回数) に従い突然変異率を減少させていく手法 [64] などが, 決定的パラメータ制御に分類される. また. 探索が経過するにつれ改悪解への移動を受理する確率が減少していく SA (A.2.3 節参照) も, この決定的パラメータ制御に分類されると考えれる.

GA 以外にも, PSO においては速度更新式 (式 (2.28)) の慣性項 w を探索経過と共に減少 [210], 又は増加 [271] させる制御方法が設計されている. これらは, 探索序盤では大域的探索能力に適したパラメータ設定を使用し, 探索の経過に伴い徐々に局所的探索能力が強い設定に変更していくという枠組みに基づいている.

集団数については, 世代の経過に従い徐々に減少させる方法 [127], 予め定めた探索資源を使い切った際に集団数を $\alpha\%$ に縮小させる方法 [72] などが提案されている. [127] や [72] は集団数を減少させるが, 反対に世代が経過する度に個体を集団に追加する方法 [44], リスタート毎に 2 倍づつ増やす方法 [9] なども提案されている.

以上に述べたアプローチは, 探索の経過に従い何らかの制御パラメータを増加, 又は減少させていた. それ以外にも, 突然変異率や交叉率を世代数 t に対する正弦波によって調整する方法 [118], PSO における慣性項 w を一様ランダムに生成する方法 [52], GA における集団数を増減, 減少を繰り返してのこぎりの歯のように変動させる方法 [123] などが提案されている.

表 4.1: 4.2.2 節にて説明する各パラメータ制御手法の概要. ここで, 本来ならば「適応的パラメータ制御」に分類されるべきはずの手法に, 「自己適応的パラメータ制御」という名称がつけられている場合が少なくない [267]. そこで, 本表では名称ではなく, 実際のアルゴリズムから判断してパラメータ制御手法を分類した.

制御方法の分類	手法名	対象 EA	対象パラメータ
決定的パラメータ制御	Decreasing-IW [210]	PSO	慣性項 w
	Increasing-IW [271]	PSO	慣性項 w
	Random-IW [52]	PSO	慣性項 w
	[64]	GA	突然変異率
	SVPS [127]	GA	集団数
	saw-tooth GA [123]	GA	集団数
	GL-25 [72]	実数値 GA	集団数
	IPSO [44]	PSO	集団数
	IPOP-CMA-ES [9]	CMA-ES	集団数
	ES [118]	ES	任意の制御パラメータ
適応的パラメータ制御	1/5 success rule [15]	ES	ステップサイズ σ
	Self-adaptive [235]	GA	突然変異率
	AER [4]	実数値 GA	交叉拡張率
	Self-adaptive [49]	実数値 GA	交叉拡張率
	GAVaPS[6]	GA	集団数
	APGA [14]	GA	集団数
	D-MAB [38]	GA	交叉手法
	AOS [63]	GA	交叉手法
自己適応的パラメータ制御	Self-adaptive ES [16]	ES	ステップサイズ σ
	EP [16]	EP	ステップサイズ σ
	FEP [260]	EP	ステップサイズ σ
	Self-adaptive [47]	実数値 GA	交叉拡張率
	Self-adaptive [12]	GA	突然変異率
	Adaptive CCG [204]	GA	遺伝される決定変数の数
	GASAP [55]	GA	集団数

2. 適応的パラメータ制御

適応的パラメータ制御では, 現在の探索状況の情報を基に使用するパラメータ設定を調整する. 最初に設計された適応的パラメータ制御は, 恐らく 2.4.3 節にて述べた $(1+1)$-ES の 1/5 success rule [15] である. 1/5 success rule では, 式 (2.14) に表せられる突然変異にて, 摂動の大きさを調整するステップサイズ σ を単純な仮定の下で適応的に変化させる. より洗練された適応手法を用いた手法としては, CMA-ES [96] (2.4.4 節参照) があげられる.

GA においては, これまでに様々な適応方法が提案されている [218, 235, 49, 4]. Thierens の適応方法 [235] では, 個体ごとに突然変異率 p_m を割り当て, (1) p_m, (2) p_m を減少させた率, (3) p_m を増加させた率のそれぞれの突然変異率の設定で子個体を生成する. そして, 最良の目的関数値を持つ子個体の p_m を次世代の個体に受け継がせる. [49] では実数値 GA (2.4.2 節参照) において, 生成した子個体が親個体よりも目的関数値が低ければ交叉拡張率を高く, 反対に目的関数値が高ければ交叉拡張率を低くするという, Nelder-Mead 法 (2.3.1 節参照) に類似した適応方法を提案している. 同様に実数値 GA において, [4] では選択した親個体集団の重心と, 交叉オペレータにより生成した子個体集団の中で目的関数値が優れている上位個体の重心を計測し, その重心の移動距離に基づき交叉拡張率を適応的に調整している.

集団数を適応的に変化させる方法として, Genetic Algorithm with Varying Population Size (GAVaPS) [6] が 1994 年に提案された. GAVaPS では, 各個体に lifetime を導入し, 集団に一定の世代以上生存した個体を削除する. GAVaPS の提案後, それを基にした様々な改良制御法が提案されている [14]. しかし, GAVaPS を始めとする集団数適応法を使用した場合, 集団数の設定が不必要となる代わりに 5 ～ 6 個の新たな制御パラメータを導入することになる問題点が 2005 年に Lobo と Lima に指摘されている [145]. このことから, 集団数のパラメータ制御には近年では先に述べた決定的なアプローチを採用する傾向が見られる.

突然変異率や集団数といった実数型, 及び整数型で表現される制御パラメータ以外にも, 交叉方法などのカテゴリカル型のパラメータを適応的に調整する Adaptive Operator Selection (AOS) に関する研究がされている [236]. Spears は一点交叉と一様交叉のどちらの交叉手法を使用するかを各個体ごとに割り当て, 適応的に選択している [217]. AOS においては, 使用するオペレータ候補 $o_1, o_2, \ldots$ をどのような基準で選択するのかが問題となる. [38] では AOS を multi-armed bandits 問題として扱い, Upper Confidence Bound (UCB) [8] に基づく適応的選択法を提案している. Fialho らは AOS を (1) オペレータ候補 $o_1, o_2, \ldots$ の報酬をどのよう与えるのか, (2) 報酬に基づきどのように選択するべきかの 2 つの課題に分け, これまでに提案されてきた適応的選択方法を実験的に解析している [63].

3. 自己適応的パラメータ制御

最後に, 自己適応的パラメータ制御では, 各個体 $\boldsymbol{x}^{i,t}$ ごとにその個体が使用するパラメータ設定 $\boldsymbol{\theta}_{i,t}$ を保持している. そして, 交叉や突然変異といった遺伝的オペレータをパラメータ設定 $\boldsymbol{\theta}_{i,t}$ にも適用し生存選択を行う. このように優れたパラメータ設定を解とともに探索することで, 対象問題に適したパラメータ設定へと調整する.

最も代表的な自己適応的パラメータ制御法として, Self-adaptive ES [16] があげられる. 1/5 success rule (2.4.3 節参照) とは異なり, 突然変異にて摂動の大きさを調整するステップサイズ σ にも突然変異を加える:

$$\sigma'_t = \sigma_t \exp\left(\tau \cdot \mathrm{randn}(0,1)\right) \tag{4.1}$$

ここで, $\tau \in [0,1]$ は学習率である. 式 (4.1) により生成した新たな σ'_t を用いて, 式 (2.14) の突然変異操作を行う. その後, 変異個体 $\boldsymbol{u}^t$ が個体 $\boldsymbol{x}^t$ よりも目的関数値が低かった場合のみ $\sigma_{t+1} = \sigma'_t$ とし, そうでなければ $\sigma_{t+1} = \sigma_t$ とする. この Self-adaptive ES とほぼ同様の適応方法が Evolutionary Programming [16, 260] においても採用されている. また, [47] では Self-adaptive ES の適応手法を実数値 GA に導入している.

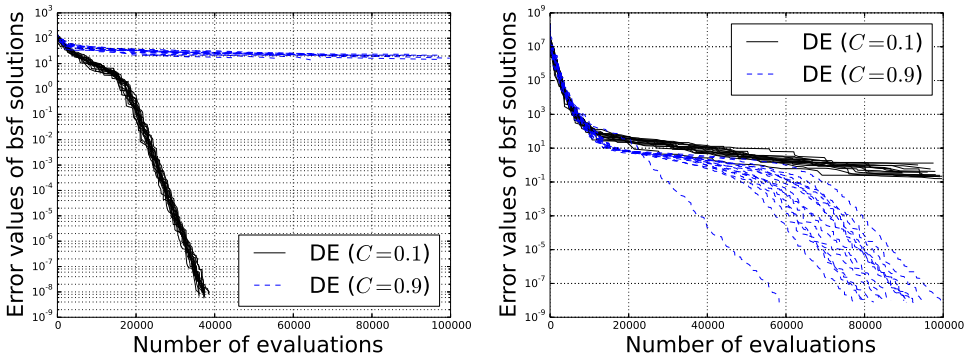
その他にも, Bäck は解 $\boldsymbol{x}$ の決定変数 $(x_1, ..., x_D)$ ごとに突然変異率 $\boldsymbol{p} = (p_1, ..., p_D)$ を用意し, 交叉や突然変異といった操作を $\boldsymbol{p}$ にも適用し, $\boldsymbol{p}$ を自己適応的に調整する方法を提案している [12]. [204] では, GA において交叉により受け継がれる決定変数の数を制御するパラメータを, 実数値 GA の交叉オペレータ [45] を用いて新たに生成している. [55] では, 各個体に集団数を決定するパラメータ p を割り当て, 各世代において全個体の p の値に基づき集団数を決定する, 自己適応的集団数調整法を提案している.

以上では自己適応的パラメータ制御法について述べたが, Zhang と Sanderson に指摘されているとおり, Self-adaptive DE [27] (後述の 4.5.1 節参照) のように本来ならば「適応的パラメータ制御」に分類されるべきはずの手法が, 「自己適応的パラメータ制御」という名称で呼ばれる場合が少なくない [267]. この混同のため, パラメータ制御が適応的であるか自己適応的であるかは, 名称からだけでは判断し難い.

4.3 DE におけるパラメータ設定問題

4.1 節にて述べたとおり, 一般的に EA の探索性能は使用するパラメータ設定に大きく依存することが知られている. この一般認識に反して DE が提案された当初は, DE の探索性能は制御パラメータの設定に対してロバストであると報告されている [220, 221]. しかし, 後年の研究結果から, 多くの EA と同様に DE の探索性能も使用するパラメータ設定に大きく依存することが判明した [71, 196, 274, 27]. ここで, 基本的な DE の制御パラメータは集団数 N, スケール係数 F, 交叉率 C である.

2002 年に報告された Gämperle らのパラメータ調査研究 [71] が, 恐らく DE のパラメータ設定問題を扱う最も古い事例である. Gämperle らは 2, 5, 20 次元の Sphere, Rastrigin, Rosenbrock 関数において, 通常の DE アルゴリズム (Algorithm 8) の適切なパラメータ設定を調査しており, 集団数 N は $\{3D, ..., 8D\}$ の範囲内, $F = 0.6$, C は $[0.3, 0.9]$ の範囲内にて設定するのが良いと結論づけている [71]. 一方, Rönkkönen らは, F は $(0.4, 0.95]$ の範囲内, C は対象問題が変数分離化であれば $[0.0, 0.2]$, 変数分離不可であれば 0.9 付近が経験的に良いと述べている [196]. Zielinski らは目的関数による解評価の計算コストが比較的安価な 16 次元の実問題において実験を行い, $F = 0.7$, $C = 0.9$ が良いとしている [274]. また, 集団数 N は



(a) Rastrigin 関数 ($D = 10$)　(b) Rosenbrock 関数 ($D = 10$)

図 4.1: $C = 0.1$, 及び 0.9 とした通常の DE アルゴリズム (Algorithm 8) を 10 次元の Rastrigin 関数と Rosenbrock 関数に適用した結果. 横軸は評価回数の経過, 縦軸は探索中に得られた最良解の目的関数値 $f(\boldsymbol{x}^{bsf})$ と最適値 $f(\boldsymbol{x}^*)$ の誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ である. 突然変異戦略に rand/1, 交叉に binomial 交叉を使用し, 集団数 $N = 100$, $F = 0.5$ とした. 15 試行分のデータを示している.

30 が適切であったとしている. Brest らの調査 [27] では, F と C が互いに依存関係にあり, それぞれの値を考慮する必要があるとしている. 悪スケール性を有する変数分離不可な単峰性関数における Auger らの調査 [10] では, $N = 10D$, $F = 0.8$, $C = 1$ が適切であるとしている. 同時に, F と C の適切な設定は対象問題によって大きく異なるため, 通常の DE アルゴリズムは扱いづらいと指摘している.

以上の DE のパラメータ設定に関する調査結果をまとめると, DE の適切なパラメータ設定は対象問題ごとに異なるため, DE を実問題に適用する際は, ユーザは対象問題に適したパラメータ設定を求めるために試行錯誤を繰り返す必要があると言える. 実例を示すために, 図 4.1 に, $C = 0.1$, 及び 0.9 とした通常の DE アルゴリズム (Algorithm 8) を 10 次元の Rastrigin 関数と Rosenbrock 関数に適用した 15 試行分のデータを示す. 集団数 $N = 50$, $F = 0.5$ とした. 変数分離化な Rastrigin 関数において, $C = 0.1$ では全ての試行で最適解を求めることに成功しているのに対し, $C = 0.9$ では全試行で失敗している. 一方, 変数分離不可な Rosenbrock 関数ではこの結果は反対となる. $C = 0.9$ の場合は全ての試行で最適解を求めることに成功しているが, $C = 0.1$ では全ての試行で失敗している. このように DE の適切なパラメータ設定は対象問題の問題性質に大きく依存し, 解きたい問題ごとに適切にパラメータチューニングを行う必要がある.

4.4　DE におけるパラメータ制御

4.3 節にて述べたように, 多くの EA と同様に DE の適切なパラメータ設定は対象問題の問題性質に強く依存する. 多くの実問題においては対象問題の情報が事前に知ることのできない black-box optimization 環境であるため, 各実問題ごとにパラメータチューニングを行う必要がある. しかし, 実問題においてはパラメータチューニングが困難な状況が多々ある. 例えば,

表 4.2: 4.4 節にて説明する, DE における各パラメータ制御手法の概要. ここで, 本来ならば「適応的パラメータ制御」に分類されるべきはずの手法に, 「自己適応的パラメータ制御」という名称がつけられている場合が少なくない [267]. そこで, 本表では名称ではなく, 実際のアルゴリズムから判断してパラメータ制御手法を分類した.

制御方法の分類	手法名	対象パラメータ
決定的パラメータ制御	DERSF, DETVSF [41]	F
	SaDE [189]	F
	SaNSDE [259]	F
	Randomized scale factor [133]	F
	CoDE [247]	F と C
	BiDE [249]	F と C
	FiADE[75]	F と C
	IDP [232]	F と C
適応的パラメータ制御	jDE [27]	F と C
	jDE2 [26]	F と C
	aDE [169]	F と C
	SaDE [189]	C
	SaNSDE [257]	F の生成方法と C
	CDE [244]	F と C
	EPSDE [157]	F, C, 及び突然変異戦略
	JADE [267]	F と C
	MDE [110]	F と C
	ZEPDE [60]	F と C
	DADE [144]	F と C
	GaDE [258]	F と C
	μJADE [33]	F と C
自己適応的パラメータ制御	SDE [173]	F
	DESAP [233]	F, C, 集団数
	DEGL [40]	突然変異戦略の貪欲性

実問題においては目的関数 f による解評価にかかる計算コストが高価である場合があり [112], このような実問題にて幾度も DE を実行することは現実的ではない. また, 適切なパラメータ設定を調整するためには, DE に関する知識が必要である. そのため, 最適化手法に関する知識は乏しいが, DE を道具として利用することのみが目的のユーザに大きな負担をかけることになる. このパラメータ設定問題は, DE の実用化に当たり大きな問題となる.

以上の背景から, 探索中に自動的にパラメータ設定が調整され, ユーザによるパラメータチューニングが不要な DE のパラメータ制御法が, 2005 年ごろから徐々に研究され始めた. 最初に提案された手法は, 主に GA や PSO の研究分野において提案されてきた適応手法から着想を得た手法であった [41, 143, 233]. 例えば, 適応的ではないが, 決定的パラメータ制御に分類される手法として, Das ら [41] が 2005 年に提案した DE with Random Scale Factor (DERSF) と DE with Time Varying Scale Factor (DETVSF) があげられる. DERSF では各世代において F を $(0.5, 1)$ の範囲内に一様ランダムに生成する. DETVSF では初期世代の F を 1.2 に設定し, 最終世代にて $F = 0.4$ となるように世代の経過に対して線形に F の値を減少させる. 以上に述べた DERSF と DETVSF は, いずれも PSO の枠組みにて提案された慣性項の決定的制御法 [211, 52] を DE に流用した手法である. 同様に, [143] では GA の枠組みにて提案されたファジィ制御を用いて制御パラメータを適応的に調整する方法 [131] を基に, Fuzzy Adaptive DE (FADE) を提案している. また, Teo は様々な自己適応型 GA 手法から着想を得た, F と C に加えて集団数 N も自己適応的に調整する DE with Self-Adapting Populations (DESAP) を設計している [233].

しかし, 2006 年に提案された jDE [27] (後述の 4.5.1 節参照) の成功をきっかけに, DE 独自のスキーマに合った適応 DE が設計されるようになった. 以下では, DE におけるパラメータ制御法を, 1. 決定的パラメータ制御 (4.4.1 節), 2. 適応的パラメータ制御 (4.4.2 節), 3. 自己適応的パラメータ制御 (4.4.3 節) といった各分類ごとに述べる. また, 表 4.1 と同じく, 表 4.2 に本節にて紹介する DE における各パラメータ制御法の概要を示す. なお, 4.4.2 節にて述べる, 近年の代表的な適応 DE である jDE [27], EPSDE [157], JADE [267] については, 4.5 節にて詳細に説明する.

4.4.1　決定的パラメータ制御

現在の探索状況を考慮せずに, 予め定めたスケジュールに従い決定的にパラメータ設定を変更する決定的パラメータ制御については, 先に説明した Das らの手法 [41] の他にも, 様々な手法がこれまでに提案されている. Self-adaptive DE (SaDE) [189] では, 各世代 t, 及び個体 $\boldsymbol{x}^{i,t}$ ごとに平均 0.5, 標準偏差 0.3 の正規分布に従う乱数により $F_{i,t}$ を生成している. 同様のアプローチが Yang らにおいても提案されている [259]. Composite DE (CoDE) [247] では, 各世代において各個体 $\boldsymbol{x}^{i,t}$ に rand/1/bin, rand/2/bin, current-to-rand/1 の 3 つの突然変異戦略を, 予め用意した F と C の 3 つのペア[*4] とランダムに組み合わせ, 3 つの子個体を生成す

[*4] $\{F, C\}$ について, $\{1.0, 0.1\}$, $\{1.0, 0.9\}$, $\{0.8, 0.2\}$ である.

る．ここで，各ペアは経験的に良いとされる F と C の組み合わせである．Wang らは二峰性分布に従いランダムに各個体 $\boldsymbol{x}^{i,t}$ の $F_{i,t}$ と $C_{i,t}$ を生成する BiDE を提案している [249]:

$$F_{i,t} = \begin{cases} \text{randc}(0.65, 0.1) & \text{if rand}(0,1) < 0.5 \\ \text{randc}(1.0, 0.1) & \text{otherwise} \end{cases} \tag{4.2}$$

$$C_{i,t} = \begin{cases} \text{randc}(0.1, 0.1) & \text{if rand}(0,1) < 0.5 \\ \text{randc}(0.95, 0.1) & \text{otherwise} \end{cases} \tag{4.3}$$

ここで，$\text{randc}(\mu, \sigma)$ は位置パラメータ μ と尺度パラメータ σ のコーシー分布に従う乱数である．[249] と同様の手法が，[133] においても提案されている．

Ghosh らは各世代 t の集団において，目的関数値が最も低い個体 $\boldsymbol{x}^{best,t}$ との目的関数値の差 $|\boldsymbol{x}^{i,t} - \boldsymbol{x}^{best,t}|$ を利用して，各個体に F と C を割り当てる Fitness-Adaptive DE (FiADE) を提案している [75][*5]．また，個体を目的関数値に基づき並び替え，得られたランクを用いて各個体の $F_{i,t}$ と $C_{i,t}$ を生成する Individual-Dependent Parameter (IDP) が [232] にて提案されている．

4.4.2 適応的パラメータ制御

現在の探索状況の情報を基に使用するパラメータ設定を調整する適応的パラメータ制御については，DE ではこれまでに様々な手法が提案されているが，概ね以下のような枠組みである．以下では，式 (3.3) に示した生存選択において，親個体 $\boldsymbol{x}^{i,t}$ よりも優れた子個体 $\boldsymbol{u}^{i,t}$ を生成できた場合，つまり $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ となった場合を「成功」と呼び，そうでない場合を「失敗」と呼んでいる:

(i) 各世代 t において，集団中の各個体 $\boldsymbol{x}^{i,t}$ に何らかの方法で $F_{i,t}$, $C_{i,t}$ を割り当て，そのパラメータを使用した突然変異，及び交叉操作により子個体 $\boldsymbol{u}^{i,t}$ を生成する

(ii) $\boldsymbol{u}^{i,t}$ を目的関数により評価し $f(\boldsymbol{u}^{i,t})$ を得て，$F_{i,t}$ と $C_{i,t}$ のペアが成功したか否かを式 (3.3) により判定する

(iii) 各世代 t の終了時に，成功した $F_{i,t}$ と $C_{i,t}$ のペアを，何らかの形で次世代 $t+1$ のパラメータ適応に反映される

以上の (i) パラメータ $F_{i,t}$, $C_{i,t}$ の生成，(ii) 成功か失敗かの判定，(iii) 次世代へのフィードバックといった一連の試行錯誤を繰り返すことで，探索状況に適したパラメータ設定へと調整していく．

ここで，DE における適応的パラメータ制御では，「対象問題，及び現在の探索状況に適したパラメータ設定を使用したために，優れた子個体を生成することができた」という仮定の下，

[*5] 決定的パラメータ制御の定義は，「現在の探索状況を考慮せずに，予め定めたスケジュールに従い決定的にパラメータ設定を変更する調整方法」である．一方，FiADE [75]，及び IDP [232] は目的関数値，及びランク情報を利用するため，「現在の探索状況を考慮」していると言え，適応的パラメータ制御に分類される可能性がある．しかし，「予め定めたスケジュールに従い」パラメータ設定を生成しているため，FiADE と IDP を本論文では決定的パラメータ制御に分類する．

成功したパラメータ設定を何らかの形で次世代に反映させる．この考えは, 4.2.2 節にて述べた他の EA のいくつかの適応的パラメータ制御 [235, 49, 15] においても使われている．ただし, CMA-ES (2.4.4 節参照) や [4] にて提案された適応的実数値 GA のように解空間における個体間の距離などは, 適応 DE では利用されていない [*6].

Self-adaptive DE (jDE) [27] では探索中は基本的には子個体 $\boldsymbol{u}^{i,t}$ の $F_{i,t}$, $C_{i,t}$ は親個体 $\boldsymbol{x}^{i,t}$ から受け継がれる．しかし, それらのパラメータ設定は予め定めた一定の確率でランダムに変更され, 成功した場合のみ変更後のパラメータを受け継ぐ．また, いくつかの jDE の改良手法が提案されている [26, 169].

SaDE [189] では先述のように F はランダムに生成するが, C については過去に H 世代において成功した C 値の中央値[*7]を平均とする正規分布に従う乱数にて, 各世代ごとに各個体の $C_{i,t}$ を生成する．Self-adaptive Neighborhood Search DE (SaNSDE) [257] では, F を正規分布, 又はコーシー分布に従い生成するかの割合を制御するパラメータ, 及び C を SaDE と同様の適応方法で自動調整する．

Competitive DE [244] では, 先述の CoDE のように予め K 個の F と C のペアを用いる．そして, 各ペアが世代 t までの成功した回数 $n_{k,t}$, $k \in \{1, ..., K\}$ を保持し, 世代 t において k 番目のペアが使用される確率 $p_{k,t}$ を $p_{k,t} = \frac{n_{k,t}+\epsilon}{\sum_{j=1}^{k}(n_{j,t}+\epsilon)}$ として求める．ここで, $\epsilon > 0$ は $p_{k,t} = 0$ となる場合を防ぐための十分に小さな定数である．

EPSDE [157] は F と C について, それぞれ $[0.4, 0.9]$, $[0.1, 0.9]$ の範囲から離散的に 0.1 刻みで保管されている F-pool, C-pool を使用する．F と C に加えて, 突然変異戦略の適応も同様のプールを用いて行う．探索の始めに, 各プールからランダムに各個体 $\boldsymbol{x}^{i,t}$ にパラメータを割り当てる．そして探索中は, 個体 $\boldsymbol{x}^{i,t}$ ごとに割り当てられている $F_{i,t}, C_{i,t}$ の組合せを用いて子個体 $\boldsymbol{u}^{i,t}$ を生成する．成功した場合は $F_{i,t}, C_{i,t}$ の組合せは次世代の個体 $\boldsymbol{x}^{i,t+1}$ へと受け継がれる．一方, 失敗した場合は各プールからランダムに新たなパラメータが再度割り当てられる．

JADE [267] は F と C の自動調整のため, それぞれ適応パラメータ $\mu_F \in (0, 1]$, $\mu_C \in [0, 1]$ を使用する．各世代 t の始めに, 個体 $\boldsymbol{x}^{i,t}$ ごとに $F_{i,t}$ と $C_{i,t}$ をそれぞれ μ_F と μ_C を位置パラメータ, 平均とするコーシー分布, 正規分布に従う乱数にて割り当てる．そして, 世代 t の終了時に成功した F と C に基づき, μ_F, μ_C を更新する．このようにサンプリングと更新を繰り返しながら探索を行うことで, μ_F, μ_C は対象問題, 及び現探索状況に適したパラメータ設定に徐々に近づくことが期待される．また, いくつかの JADE の改良手法が提案されている [60, 144, 258, 33, 110].

[*6] いくつかの適応 DE では, 親個体から子個体の目的関数値の改善値 $|f(\boldsymbol{u}^{i,t}) - f(\boldsymbol{x}^{i,t})|$ を用いて, 各成功したパラメータ設定に重み付けをし次世代に反映させる方法が提案されている [257, 179]．ただし, 方法を用いた場合, DE は 3.3.1 節にて述べた順序を保持した目的関数値の変換に対する不変性を失う．

[*7] 平均値を使用する実装法も存在する [191].

4.4.3　自己適応的パラメータ制御

最後に, 個体 $\boldsymbol{x}^{i,t}$ ごとにその個体が使用するパラメータ設定を保持する自己適応的パラメータコントロールについて述べるが, 4.2.2 節にて述べた GA や ES などの EA と比べ, DE の枠組みではあまり研究されておらず, 著者の知る限り以下に述べる 3 件しか存在しない. このことから, DE におけるパラメータ制御研究は, 先に述べた決定的パラメータ制御, あるいは適応的パラメータ制御にほとんどの場合分類されると言える.

Omran らは, 式 (3.2) の rand/1 突然変異戦略を用いて各個体 $\boldsymbol{x}^{i,t}$ の $F_{i,t}$ を新たに生成する Self-adaptive DE を提案している [173]. ただし, [173] では C の自己適応は行わない. また, 先に述べた Teo の自己適応的 DE [233] も Omran らと同様の方法で F と C を調整する. F と C の自己適応ではないが, [40] では突然変異戦略の貪欲性を制御するパラメータを rand/1 突然変異戦略を用いて生成するという, 自己適応的調整を採用している.

4.5　近年の代表的な適応 DE の詳細な説明

本節では 4.4.2 節にて簡潔に説明した jDE [27], EPSDE [157], JADE [267] について, より詳細に説明する. なお, この 3 手法は近年において代表的な適応 DE であるため, 次章以降の本論文において, 比較手法として使用することとした.

4.5.1　Self adaptive DE (jDE)

2006 年に提案された jDE[*8] [27] は Algorithm 8 に示す古典的な DE [221] に適応手法を導入した適応 DE である. つまり, jDE では突然変異戦略には rand/1, 交叉手法には binomial 交叉を使用する. 4.4 節にて述べた FADE [143] と比較して, jDE は単純なアルゴリズムでありながら良好な探索性能を示すことが報告されている [27]. jDE は近年の適応 DE の標準的手法として考えられており, 様々な研究に比較対象として使用されている [189, 157, 267, 247, 110].

jDE では, 集団 $\boldsymbol{P}^t = (\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t})$ の各個体 $\boldsymbol{x}^{i,t}$ はそれぞれ $F_{i,t}$, 及び $C_{i,t}$ を保持する. 探索の開始時 $t = 1$ においては, 各個体 $\boldsymbol{x}^{i,t}$ のパラメータは $F_{i,t} = 0.5$, $C_{i,t} = 0.9$ と初期化されている (Algorithm 11, 2 行目). 探索中, 子個体 $\boldsymbol{u}^{i,t}$ の $F_{i,t}$, $C_{i,t}$ は基本的に親個体 $\boldsymbol{x}^{i,t}$ から受け継がれる. しかし, それらのパラメータ設定は次の更新式 (4.4), (4.5) で示すように, 予め定めた一定の確率で各世代ごとにランダムに変更される (Algorithm 11, 5 ～ 12 行目):

$$F'_{i,t} = \begin{cases} \mathrm{rand}[0.1, 1] & \text{if } \mathrm{rand}[0, 1] < \tau_F \\ F_{i,t} & \text{otherwise} \end{cases} \tag{4.4}$$

[*8] 「jDE」という名称は [27] の中では使用されておらず, 単に「Self-adaptive DE」と呼ばれている. その後, 数多くの他の適応 DE と区別するために, jDE という名称が付けられた. ここで, jDE の「j」は, 第一著者 (Janez Brest) の first name の頭文字であると思われる.

Algorithm 11: jDE

1 $t \leftarrow 1$, 集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の初期化;
2 各個体 $\boldsymbol{x}^{i,t}$ について, $F_{i,t} = 0.5$, $C_{i,t} = 0.9$ とする;
3 **while** 探索終了条件を満たしていない **do**
4 **for** $i = 1$ **to** N **do**
5 **if** rand$[0,1] \leq \tau_F$ **then**
6 $F'_{i,t} = \text{rand}[0.1, 1]$;
7 **else**
8 $F'_{i,t} = F_{i,t}$;
9 **if** rand$[0,1] \leq \tau_C$ **then**
10 $C'_{i,t} = \text{rand}[0, 1]$;
11 **else**
12 $C'_{i,t} = C_{i,t}$;
13 式 (3.2) の rand/1 を用いて $\boldsymbol{v}^{i,t}$ を生成;
14 $\boldsymbol{x}^{i,t}$ と $\boldsymbol{v}^{i,t}$ に binomial 交叉 (Algorithm 7) を適用し, $\boldsymbol{u}^{i,t}$ を生成;
15 **for** $i = 1$ **to** N **do**
16 **if** $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
17 $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t+1} = F'_{i,t}$, $C_{i,t+1} = C'_{i,t}$;
18 **else**
19 $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$, $F_{i,t+1} = F_{i,t}$, $C_{i,t+1} = C_{i,t}$;
20 $t \leftarrow t + 1$;

$$C'_{i,t} = \begin{cases} \text{rand}[0,1] & \text{if rand}[0,1] < \tau_C \\ C_{i,t} & \text{otherwise} \end{cases} \tag{4.5}$$

更新式 (4.4), (4.5) において, τ_F, $\tau_C \in (0, 1]$ はそれぞれ F, C のパラメータ適応を制御するパラメータであり, 値が高いほど頻繁にランダム生成した新たなパラメータが使用されることとなる.

式 (4.4), (4.5) により生成した $F'_{i,t}$, 及び $C'_{i,t}$ を用いて突然変異戦略に rand/1, 及び binomial 交叉を使用することで, 子個体 $\boldsymbol{u}^{i,t}$ を生成する (Algorithm 11, 13 ～ 14 行目). 全ての個体が子個体を生成した後, 世代交代を行うが, この際成功した場合のみ, つまり $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ となった場合のみ, 次世代の $\boldsymbol{x}^{i,t+1}$ へと $F'_{i,t}$ と $C'_{i,t}$ は受け継がれる (Algorithm 11, 15 ～ 19 行目). これにより, 過去の探索にて成功したパラメータ設定を保持しつつも, 一定の確率で新たなパラメータ設定を試すことが可能である.

4.5.2 Ensemble of Mutation and Crossover Strategies and Parameters in DE (EPSDE)

Mallipeddi らに提案された EPSDE [157] は, F と C に加えて突然変異戦略の適応も行う. EPSDE では, F と C について, それぞれ $[0.4, 0.9]$, $[0.1, 0.9]$ の範囲から離散的に 0.1 刻

Algorithm 12: EPSDE

1 $t \leftarrow 1$, 集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の初期化;
2 F-pool $= \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$;
3 C-pool $= \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$;
4 S-pool $=$ { rand/1, best/2, current-to-rand/1 };
5 各個体 $\boldsymbol{x}^{i,t}$ について, $S_{i,t}, F_{i,t}, C_{i,t}$ を各プールからランダムに割り当てる;
6 **while** 探索終了条件を満たしていない **do**
7 **for** $i = 1$ **to** N **do**
8 $F_{i,t}$ を用いた突然変異戦略 $S_{i,t}$ により $\boldsymbol{v}^{i,t}$ を生成;
9 $\boldsymbol{x}^{i,t}$ と $\boldsymbol{v}^{i,t}$ に交叉率 $C_{i,t}$ の binomial 交叉 (Algorithm 7) を適用し, $\boldsymbol{u}^{i,t}$ を生成;
10 **for** $i = 1$ **to** N **do**
11 **if** $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
12 $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $S_{i,t+1} = S_{i,t}, F_{i,t+1} = F_{i,t}, C_{i,t+1} = C_{i,t}$;
13 **else**
14 $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$;
15 各個体 $\boldsymbol{x}^{i,t+1}$ について, $S_{i,t+1}, F_{i,t+1}, C_{i,t+1}$ を各プールからランダムに割り当て直す;
16 $t \leftarrow t + 1$;

みで保管されている F-pool, C-pool を使用する. 同様に, 突然変異戦略については rand/1, best/2, current-to-rand/1 を保持する S-pool を用いる. EPSDE は同研究グループにて提案された SaDE [189] と比べ, 比較的単純なアルゴリズムでありながら, SaDE や jDE [27] を含む他の適応 DE よりも優れた探索性能を有することが報告されている.

探索の開始時 $t = 1$ においては, 3 つの各プールから一様ランダムに各個体 $\boldsymbol{x}^{i,t}$ にパラメータを割り当てる (Algorithm 12, 5 行目). そして探索中は, 各個体 $\boldsymbol{x}^{i,t}$ ごとに割り当てられている $S_{i,t}, F_{i,t}, C_{i,t}$ の組合せを用いて子個体 $\boldsymbol{u}^i$ を生成する (Algorithm 12, 8 ∼ 9 行目).

全ての個体が子個体を生成した後, 世代交代を行うが, この際成功した場合のみ, つまり $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ となった場合は, 次世代の $\boldsymbol{x}^{i,t+1}$ においても $S_{i,t}, F_{i,t}, C_{i,t}$ を使用する. 一方, 失敗した場合は S-pool, F-pool, C-pool から再度ランダムに新たなパラメータが割り当てられる (Algorithm 12, 15 行目). jDE では新たに生成したパラメータ設定は, 成功した場合のみ次世代へと受け継がれるが, EPSDE では失敗した場合のみ次世代にて使用するパラメータ設定を新たに生成する.

4.5.3　Adaptive DE (JADE)

2009 年に提案された JADE[*9] [267] は, 今日において比較的良好な性能を示す適応 DE であり, いくつかの JADE の改良手法も提案されている [60, 144, 258, 33]. [267] では, 独自の

[*9] JADE の「JA」は, 提案者である Jingqiao Zhang と Arthur C. Sanderson のファーストネームの頭文字を合わせたものであると考えられる.

Algorithm 13: JADE

1 $t \leftarrow 1$, 集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の初期化;
2 $\mu_F \leftarrow 0.5$, $\mu_C \leftarrow 0.5$;
3 **while** 探索終了条件を満たしていない **do**
4 　$\boldsymbol{S}^F \leftarrow \emptyset$, $\boldsymbol{S}^C \leftarrow \emptyset$;
5 　**for** $i = 1$ **to** N **do**
6 　　$F_{i,t} = \text{randc}(\mu_F, 0.1)$;
7 　　$C_{i,t} = \text{randn}(\mu_C, 0.1)$;
8 　　current-to-$p$best/1 (表 3.1) を用いて $\boldsymbol{v}^{i,t}$ を生成;
9 　　$\boldsymbol{x}^{i,t}$ と $\boldsymbol{v}^{i,t}$ に binomial 交叉 (Algorithm 7) を適用し, $\boldsymbol{u}^{i,t}$ を生成;
10 　**for** $i = 1$ **to** N **do**
11 　　**if** $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
12 　　　$\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t} \rightarrow \boldsymbol{S}^F$, $C_{i,t} \rightarrow \boldsymbol{S}^C$;
13 　　**else**
14 　　　$\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$;
15 　**if** $\boldsymbol{S}^F, \boldsymbol{S}^C \neq \emptyset$ **then**
16 　　$\mu_F \leftarrow (1-c) \cdot \mu_F + c \cdot \text{mean}_L(\boldsymbol{S}^F)$;
17 　　$\mu_C \leftarrow (1-c) \cdot \mu_C + c \cdot \text{mean}_A(\boldsymbol{S}^C)$;
18 　$t \leftarrow t + 1$;

パラメータ適応手法の他に, 新たな突然変異戦略として current-to-$p$best/1 (表 3.1) を提案している.

JADE では F と C の自動調整のため, それぞれに対して適応手法により直接調整されるパラメータである適応メタパラメータ $\mu_F \in (0, 1]$, $\mu_C \in [0, 1]$ を使用する. 探索開始時 $t = 1$ においては, μ_F と μ_C はそれぞれ 0.5 に初期化されている (Algorithm 13, 2 行目). 各世代 t の始めに, 各個体 $\boldsymbol{x}^{i,t}$ の $F_{i,t}$ と $C_{i,t}$ は, μ_F と μ_C を用いてそれぞれ次のように生成する (Algorithm 13, 6 ∼ 7 行目):

$$F_{i,t} = \text{randc}(\mu_F, 0.1) \tag{4.6}$$
$$C_{i,t} = \text{randn}(\mu_C, 0.1) \tag{4.7}$$

ここで, randc(μ, σ) は位置パラメータ μ と尺度パラメータ σ のコーシー分布に従う乱数, randn(μ, σ) は平均 μ, 標準偏差 σ の正規分布に従う乱数である. $F_{i,t}$ の値が $F_{i,t} > 1$ の場合は $F_{i,t} = 1$ とし, $F_{i,t} \leq 0$ の場合は再び式 (4.6) を用いて生成を行う. 生成された $C_{i,t}$ の値が $[0, 1]$ 区間外の場合は, 超えた方の境界値で置き換えられる. 両裾が正規分布よりも広いコーシー分布を F の生成に使用するのは, 初期収束を防ぐために多様な F の値を DE では必要とするからである.

$F_{i,t}$, 及び $C_{i,t}$ を用いて current-to-$p$best/1 突然変異戦略, 及び binomial 交叉を使用することで子個体 $\boldsymbol{u}^{i,t}$ を生成する (Algorithm 13, 8 ∼ 9 行目). 全ての個体が子個体を生成した後に世代交代を行うが, この際成功した場合のみ, つまり $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ となった場合のみ,

$F_{i,t} \to \boldsymbol{S}^F$, $C_{i,t} \to \boldsymbol{S}^C$ とする (Algorithm 13, 12 行目). ここで, $\boldsymbol{S}^F, \boldsymbol{S}^C$ は世代 t において成功した F と C の集合である.

各世代の終了時に, $\boldsymbol{S}^F, \boldsymbol{S}^C$ に基づき次の更新式 (4.8), (4.9) を用いて μ_F, μ_C を更新する (Algorithm 13, 15 ~ 17 行目):

$$\mu_F \leftarrow (1-c) \cdot \mu_F + c \cdot \mathrm{mean}_L(\boldsymbol{S}^F) \tag{4.8}$$

$$\mu_C \leftarrow (1-c) \cdot \mu_C + c \cdot \mathrm{mean}_A(\boldsymbol{S}^C) \tag{4.9}$$

更新式 (4.8), (4.9) において, $c \in [0, 1]$ は学習率であり, c の値が高いほど直近の世代の $\boldsymbol{S}^F, \boldsymbol{S}^C$ が μ_F, μ_C に反映されることとなる. また, $\mathrm{mean}_A(\cdot)$ は算術平均, $\mathrm{mean}_L(\cdot)$ は階数 2 のレーマー平均 (Lehmer mean)[*10] である. 低い F の値を集団中の全ての個体が使用した場合は初期収束を起こしやすいため, 集合の高い値に偏りやすいレーマー平均を μ_F の更新に使用している. このように式 (4.8), (4.9) によるパラメータ生成と式 (4.8), (4.9) によるパラメータ更新を繰り返すことで, μ_F, μ_C は対象問題, 及び現在の探索状況に適したパラメータ設定に徐々に近づいていくことが期待される.

[*10] 階数 2 のレーマー平均は次のように求められる: $\mathrm{mean}_L(S) = \frac{\sum_{s \in \boldsymbol{S}} s^2}{\sum_{s \in \boldsymbol{S}} s}$. ここで, $\boldsymbol{S} = (s_1, ..., s_{|S|})$ は $\boldsymbol{S}^F, \boldsymbol{S}^C$ のいずれかである.

第 5 章

適応 DE の適応手法の解析

4 章にて述べたように, 多くの EA と同様に DE の適切なパラメータ設定は対象問題の問題性質に強く依存する. そのため, 探索中に自動的にパラメータ設定が調整されユーザによるパラメータチューニングが不要なパラメータ制御法が近年活発に研究されている. 4.4 節では DE におけるパラメータ制御法を, (1) 決定的パラメータ制御 (4.4.1 節), (2) 適応的パラメータ制御 (4.4.2 節), (3) 自己適応的パラメータ制御 (4.4.3 節) といった各分類ごとに紹介した. その中でも, 現在の探索状況の情報を基に使用するパラメータ設定を調整する, (2) 適応的パラメータ制御に関する研究が近年増加しており, DE におけるパラメータ制御法の主流となっており [43], これまでにいくつかの優れた適応 DE が提案されている. しかし, 多くの優れた適応 DE が提案されている一方, その適応手法に関する知見は乏しい.

本章では適応 DE の適応手法を (i) ベンチマーク問題集における性能評価, (ii) 本章にて提案する新たなシミュレーション法により解析する. (i) の目的は「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しを評価し, どの適応 DE の適応手法が一般的に, 又は特定のオペレータを用いた場合に優れているのかを明らかにすることである. 16 種類のオペレータにおける適応 DE の適応手法の探索性能を, BBOB benchmarks [93] にて評価する. その結果から, 適応手法間の優劣関係は使用するオペレータに大きく依存し, あらゆるオペレータにおいて優れた性能を示す適応手法は無いことがわかった. また, 本来の適応 DE の枠組みにおいて使用されることが前提とされているオペレータ以外を用いた場合, 各文献にて報告されている探索性能と比べ劣る傾向が確認された.

(ii) の目的は「適応手法」の良し悪しを「遺伝的オペレータ」とは独立して評価し, どの適応手法が優れた適応性能を有するのか, 高性能な適応手法を設計するためには何が求められているのかを明らかにすることである. 「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しは (i) の実験にて比較的容易に検証可能であるが, 「適応手法」のみを独立して評価することはいくつかの理由から困難であり, 著者の知る限りこれまで行われていない. この目的のため, 理想的なパラメータ適応の軌跡の代理である oracle パラメータを用いた新たなシミュレーション法を提案する. 提案シミュレーション法ではパラメータ値を独立して評価することが可能であり, これまで困難であったパラメータ適応手法の適応能力についての議論を可能とする. 本シミュレーション法により従来の適応 DE のパラメータ適応能力を解析することで,

既存の適応手法の問題点を明らかにする.

5.1 はじめに

4.1 節にて述べたとおり, 一般的に EA の探索性能は使用するパラメータ設定に大きく依存する [146, 54]. また, 4.3 節にて説明したように, 多くの EA と同様に DE の探索性能も用いる制御パラメータに大きく依存することが知られている [71, 196, 274, 27]. ここで, 基本的な DE の制御パラメータは集団数 N, スケール係数 F, 交叉率 C である. DE の適切なパラメータ設定は対象問題ごとに異なるため, ユーザは対象問題に適した DE のパラメータ設定を求めるために試行錯誤を繰り返す必要がある.

これは DE の実用化にあたり大きな問題となるため, 近年ではパラメータ設定を探索中に自動的に調整する DE のパラメータ制御法に関する研究が数多く行われている (4.4 節参照). EA におけるパラメータ制御法は, (1) 決定的パラメータ制御 (4.4.1 節), (2) 適応的パラメータ制御 (4.4.2 節), (3) 自己適応的パラメータ制御 (4.4.3 節) といった 3 種類に大別できるが, DE においては (2) 適応的パラメータ制御に関する研究が主流である [43]. 近年の代表的な適応 DE には, jDE [27], SaDE [189], EPSDE [157], JADE [267], MDE [110] などがあげられる.

しかし, 多くの優れた適応 DE が提案されている一方, その適応手法に関する知見は乏しい. 例えば, ほぼ全ての先行研究 [27, 189, 157, 267, 110] では, 適応 DE を新たに提案しベンチマーク集合にてその探索性能を評価している. そして, 提案する適応 DE は有用であると結論づけている. 一方, なぜ提案する適応 DE が既存手法よりも良い性能を示すのか, どのような特徴を有するのかといった考察や解析はしていない. そのため, 異なるパラメータ適応手法間にてなぜ探索性能に優劣が発生するのか, 優れた適応手法を設計するためには何が重要であるのかといったことは不明である. また, 適応手法のパラメータ適応の振る舞いを調査した先行研究はいくつか存在するが, 極めて限定的な情報しか得ることができていない.

以下の 5.1.1 節, 及び 5.1.2 節では, それぞれ適応手法の性能評価, 及びパラメータ適応の振る舞いの解析についての問題点を, 先行研究をあげながら述べる. 最後に 5.1.3 節にて, これら 2 つの問題点に対する本研究の概要を説明する.

5.1.1 適応手法の性能評価の問題点

適応 DE は複数の構成要素から成るが, 一般的に「適応 DE」という単語はこの複合したアルゴリズムを意味する. 例えば, 4.5.3 節にて解説した「JADE」 は, 突然変異戦略に current-to-p-best/1, 交叉手法に binomial 交叉, そして JADE 独自の適応手法を使用した複合体を指す. 同様に, 4.5.1 節にて述べた「jDE」 [27] は, 突然変異戦略に rand/1, 交叉手法に binomial 交叉, 適応手法に jDE 独自のものを使用した複合体のことである.

こうした複合体同士の比較 (例えば, jDE と JADE の比較) は数多く行われているものの, 純粋に適応手法同士 (例えば, jDE と JADE の適応手法の比較) を比較した研究は少ないことが, 近年指摘されている [273, 51]. そのため, 優れた「適応 DE」が提案されている一方, どの

「適応 DE の適応手法」が優れているか, また様々なオペレータと各適応手法を組み合わせた際の探索性能は良く知られていない.

Zielinski らは jDE [27] や SaDE [189] などの適応 DE アルゴリズムの構成要素を組み換え, どの構成要素がアルゴリズムの探索性能に貢献しているのかを調査している [273]. 総計 8 種類のアルゴリズムを制約付き最適化問題のベンチマーク集合にて評価し, SaDE の適応手法が比較的良好であると結論づけている. Drozdik らは多目的最適化問題に対するいくつかの適応 DE の適応手法を, 多目的 DE アルゴリズムである Differential Evolution for Multiobjective Optimization (DEMO) [194] の枠組みに導入し, 多目的最適化問題にて性能比較をしている [51]. 一般的に適応 DE の適応手法は F と C の両方のパラメータを自動調整するが, Segura らは F についての適応手法のみを比較した [207]. 3 つの適応 DE (cDE [244], jDE, JADE) の F の適応手法, 及び正規分布とコーシー分布に従う乱数にて F を生成する手法を比較した. このような本来の枠組みから適応手法を独立させて性能評価をするアプローチは, Ant Colony Optimization (ACO) [178] や Adaptive Operator Selection (AOS) [63] の枠組みにおいても行われている.

しかし, 先行研究 [273, 51] では, 制約付き最適化問題や多目的最適化問題に焦点を当てており, 最も基本的な単目的最適化問題における調査はしていない. さらに, いずれの先行研究 [273, 51, 207] においても, 突然変異戦略と交叉手法には数種類程度の限られた組み合わせしか使用していない. これらのことから, 数多くの適応 DE が提案されている一方, どの適応 DE の適応手法が一般的に, 又は特定のオペレータを用いた場合に優れているのかは未だに明らかにされていない.

5.1.2　適応手法の解析の問題点

適応 DE の適応手法の振る舞いを調査した研究は, 少なからず行われている [27, 30, 189, 267, 157, 36]. しかし, 先行研究の多くは適応メタパラメータ (F, C の生成に使用される, 適応手法により直接調整されるパラメータ), 又は生成された F, C の値を探索の経過ごとに後述の図 5.6 のように図示し, 視覚に基づく定性的な議論に留まっている. 視覚に基づく解析法は, ある問題インスタンスにおいて特定の適応手法がどのように振る舞うのかを大まかに把握するには有用である. しかし, それ以上の議論, すなわち特定の適応手法が優れた適応性能を示す理由を解明するためには, あまり有用ではない. また, Karafotias らに指摘されているように, DE だけではなく EA 全体においても, ある適応手法が有用な理由を明らかにした研究は少ない [119].

適応手法の解析を難しくさせている最も大きな障害は, 適応手法が生成したパラメータ値自体の良し悪しを判別することが困難な点である. 「遺伝的オペレータ」と「適応手法」の良し悪しは生成した新たな解 $\boldsymbol{x}$ の目的関数値 $f(\boldsymbol{x})$ により判断可能であるのに対して, 「適応手法」のみを独立して評価することは, 非常に困難である.

例えば, $\boldsymbol{p} = \{p_1, ..., p_n\}$ を適応手法により生成された, EA の n 個のパラメータ集合であるとする. DE の場合は $n = 2$ であり $\boldsymbol{p} = \{F, C\}$ である. EA ではパラメータ集合 $\boldsymbol{p}$ を用いた

遺伝的オペレータにより, 新たな解 $\boldsymbol{x}$ を生成する. DE では, 突然変異戦略と交叉手法がこの遺伝的オペレータに該当する. そして新しく生成した解 $\boldsymbol{x}$ は, 対象問題の目的関数 f にて評価することで, 目的関数値 $f(\boldsymbol{x})$ を得る. この時, 遺伝的オペレータとパラメータ集合 $\boldsymbol{p}$ を生成した適応手法の良し悪しは, 解 $\boldsymbol{x}$ の良し悪し, すなわち目的関数値 $f(\boldsymbol{x})$ により判別可能である. 一方, 適応手法を遺伝的オペレータとは独立して評価することは困難である. これは, (1) パラメータ値空間 $\boldsymbol{S}^p \subseteq \mathbb{R}^n$, (2) 解空間 $\boldsymbol{S}^x \subseteq \mathbb{R}^D$, (3) 目的関数空間 $\boldsymbol{S}^f \subseteq \mathbb{R}$ という 3 つの空間が存在し, それぞれを分けて考えることが通常はできないためである.

一般的に, 良好な $f(\boldsymbol{x})$ を有する $\boldsymbol{x}$ を生成できたのは遺伝的オペレータとパラメータ集合 $\boldsymbol{p}$ の相乗効果のためであり, 分離して考えることは困難である. また, 遺伝的オペレータの確率的な性質から, 本来ならば適切なパラメータ $\boldsymbol{p}$ を用いたとしても必ず良解を生成できるとは限らず, 反対に不適切な $\boldsymbol{p}$ を使用したとしても良い解を生成する可能性は存在する. 以上の要因から, 「遺伝的オペレータ」とは独立に「適応手法」の良し悪しを決定するのは困難である.

5.1.3 本章の概要と構成

本章では 5.1.1 節, 及び 5.1.2 節で述べた適応 DE の適応手法の問題点を, (i) ベンチマーク問題集における性能評価, (ii) 本章にて提案する新たなシミュレーション法により取り組む. なお, 本研究では近年の代表的な適応 DE である jDE [27], EPSDE [157], JADE [267], MDE [110] の適応手法を調査対象とした.

(i) の目的は「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しを評価し, どの適応 DE の適応手法が一般的に, 又は特定のオペレータを用いた場合に優れているのかを明らかにすることである. 16 種類のオペレータにおける適応 DE の適応手法の探索性能を, BBOB benchmarks [93] にて評価する. これにより, 5.1.1 節にて述べた, どの適応 DE の適応手法が優れているか, また様々なオペレータと各適応手法を組み合わせた際の探索性能, 各適応手法の特徴などのこれまで不明であった点を明らかにする.

(i) では「遺伝的オペレータ」と「適応手法」の組み合わせに興味の焦点を当てていたのに対して, (ii) の目的は「適応手法」の良し悪しを独立して評価し, 特定の適応手法が優れた探索性能を示す理由, 高性能な適応手法を設計するためにはどのような機能が求められているのかを明らかにすることである. 「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しは (i) の実験にて検証可能であるが, 「適応手法」のみを独立して評価することは 5.1.2 節にて述べた理由から困難であり, 著者の知る限りこれまで行われていない. 最も大きな障害は, パラメータ値空間と解空間を分離して考えることが困難, つまり適応手法が生成したパラメータ値自体の良し悪しをオペレータから独立して判別することが困難な点である. これを解消するために, 理想的なパラメータ適応の軌跡の代理である oracle パラメータを用いた新たなシミュレーション法を提案する. 提案シミュレーション法ではパラメータ値を独立して評価することが可能であり, これまで困難であったパラメータ適応手法の適応能力についての議論を可能とする. 本シミュレーション法により従来の適応 DE のパラメータ適応能力を解析することで, 既存の適応手法の問題点を明らかにする.

Algorithm 14: 一般化した jDE の適応手法

$t \leftarrow 1$, 集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の初期化;
各個体 $\boldsymbol{x}^{i,t}$ について, $F_{i,t} = 0.5$, $C_{i,t} = 0.9$ とする;
while 探索終了条件を満たしていない **do**
    for $i = 1$ **to** N **do**
        if rand$[0, 1] \leq \tau_F$ **then**
            $F'_{i,t} = \text{rand}[0.1, 1]$;
        else
            $F'_{i,t} = F_{i,t}$;
        if rand$[0, 1] \leq \tau_C$ **then**
            $C'_{i,t} = \text{rand}[0, 1]$;
        else
            $C'_{i,t} = C_{i,t}$;
        表 3.1 の任意の突然変異戦略により変異個体 $\boldsymbol{v}^{i,t}$ を生成;
        $\boldsymbol{x}^{i,t}$ と $\boldsymbol{v}^{i,t}$ に binomial 交叉 (Algorithm 7), 又は SEC (Algorithm 22) のいずれかの交叉手法を適用し, 子個体 $\boldsymbol{u}^{i,t}$ を生成;
    for $i = 1$ **to** N **do**
        if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t+1} = F'_{i,t}$, $C_{i,t+1} = C'_{i,t}$;
        else
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$, $F_{i,t+1} = F_{i,t}$, $C_{i,t+1} = C_{i,t}$;
    $t \leftarrow t + 1$;

本章の構成は以下のとおりである. 始めに 5.2 節にて, 本章で解析対象とする 4 つの適応 DE (jDE, EPSDE, JADE, MDE) の適応手法を一般化させた枠組みについて述べる. 先に述べた目的 (i) のために, 5.3 節にて様々なオペレータを使用した場合における各適応手法の探索性能を, BBOB benchmarks にて評価する. 次に, 目的 (ii) のために, 5.4 節では理想的なパラメータ適応の軌跡の代理である oracle パラメータを用いたシミュレーション法を提案する. そして, 5.5 節にて提案シミュレーション法による適応手法のパラメータ適応能力の評価実験を行う. 最後に 5.6 節にて本章をまとめる.

5.2　一般化した適応 DE の適応手法

本節では, 本章にて解析対象とする jDE [27], EPSDE [157], JADE [267], MDE [110] の適応手法を一般化させた枠組みについて述べる. これにより, 後の 5.3 節, 5.4 節, 5.5 節の準備とする.

8.1 節で述べたとおり, 本研究の興味は「適応 DE」ではなく, 「適応 DE における F, C の適応手法」である. そのため, 例えば MDE では current-to-$p$best/1 と類似した current-to-gr_best/1 という突然変異戦略や, p-best 交叉という独自の交叉方法を使用しているが, 表 3.1,

Algorithm 15: 一般化した EPSDE の適応手法

$t \leftarrow 1$, 集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の初期化;
F-pool $= \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$;
C-pool $= \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$;
各個体 $\boldsymbol{x}^{i,t}$ について, $F_{i,t}, C_{i,t}$ を各プールからランダムに割り当てる;
while 探索終了条件を満たしていない **do**
    for $i = 1$ **to** N **do**
        表 3.1 の任意の突然変異戦略により変異個体 $\boldsymbol{v}^{i,t}$ を生成;
        $\boldsymbol{x}^{i,t}$ と $\boldsymbol{v}^{i,t}$ に binomial 交叉 (Algorithm 7), 又は SEC (Algorithm 22) のいずれかの交叉手法を適用し, 子個体 $\boldsymbol{u}^{i,t}$ を生成;
    for $i = 1$ **to** N **do**
        if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t+1} = F_{i,t}, C_{i,t+1} = C_{i,t}$;
        else
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$;
            各個体 $\boldsymbol{x}^{i,t+1}$ について, $F_{i,t+1}, C_{i,t+1}$ を各プールからランダムに割り当て直す;
    $t \leftarrow t + 1$;

及び Algorithm 7, 22 に示す任意の突然変異戦略, 及び交叉手法が使用できるよう, 柔軟な枠組みに修正する必要がある. つまり, 適応 DE という複合体から適応手法という構成要素のみを抜き取り, 任意のオペレータを使用できるように修正する. このようなアプローチは, 5.1.1 節にて述べたとおり, 先行研究 [273, 178, 51, 207] でも行われている. ここで, jDE, EPSDE, JADE, MDE の「F, C の適応手法」ではなく, 「様々な構成要素から成るアルゴリズム全体」については, 4.5 節を参照されたい.

Algorithm 14, 15, 16, 17 に, jDE (Algorithm 11), EPSDE (Algorithm 12), JADE (Algorithm 13), MDE[*1]を一般化した枠組みを示す. ここで, Algorithm 17 において, $\text{mean}_P(\boldsymbol{S}) = \left(\frac{1}{|\boldsymbol{S}|}\sum_{s\in\boldsymbol{S}} s^{1.5}\right)^{\frac{1}{1.5}}$ である.

Algorithm 14, 15, 16, 17 は任意の突然変異戦略, 及び交叉手法を使用できるように, 元のアルゴリズムに修正されている. 以下, 本章では Algorithm 14, 15, 16, 17 に示した一般化された適応 DE の適応手法を解析対象とする.

5.3 実験 1: BBOB benchmarks における適応手法の探索性能の比較

本節では, 様々な突然変異戦略と交叉手法の組み合わせにおける適応 DE の適応手法の性能を, BBOB benchmarks [91, 93] にて評価する. 本節の目的は, 「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しを評価し, これまで不明であった (5.1.1 節参照), 各適応 DE の

[*1] 本来の MDE は JADE のアルゴリズムと類似しているため, 4.5 節でのアルゴリズムの記載は省略した.

Algorithm 16: 一般化した JADE の適応手法

$t \leftarrow 1$, 集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の初期化;
$\mu_F \leftarrow 0.5$, $\mu_C \leftarrow 0.5$;
while 探索終了条件を満たしていない **do**
    $\boldsymbol{S}^F \leftarrow \emptyset$, $\boldsymbol{S}^C \leftarrow \emptyset$;
    for $i = 1$ **to** N **do**
        $F_{i,t} = \mathrm{randc}(\mu_F, 0.1)$;
        $C_{i,t} = \mathrm{randn}(\mu_C, 0.1)$;
        表 3.1 の任意の突然変異戦略により変異個体 $\boldsymbol{v}^{i,t}$ を生成;
        $\boldsymbol{x}^{i,t}$ と $\boldsymbol{v}^{i,t}$ に binomial 交叉 (Algorithm 7), 又は SEC (Algorithm 22) のいずれかの交叉手法を適用し, 子個体 $\boldsymbol{u}^{i,t}$ を生成;
    for $i = 1$ **to** N **do**
        if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t} \rightarrow \boldsymbol{S}^F$, $C_{i,t} \rightarrow \boldsymbol{S}^C$;
        else
            $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$;
    if $\boldsymbol{S}^F, \boldsymbol{S}^C \neq \emptyset$ **then**
        $\mu_F \leftarrow (1-c) \cdot \mu_F + c \cdot \mathrm{mean}_L(\boldsymbol{S}^F)$;
        $\mu_C \leftarrow (1-c) \cdot \mu_C + c \cdot \mathrm{mean}_A(\boldsymbol{S}^C)$;
    $t \leftarrow t + 1$;

構成要素とは独立してどの適応手法が優れているのか, 様々なオペレータと各適応手法を組み合わせた際の探索性能を明らかにする.

5.3.1　実験設定

本節の性能評価実験では, ベンチマーク集合に BBOB benchmarks [91, 93] を使用した. BBOB benchmarks については, 2.5.3 節, より詳しくは付録 C 節を参照されたい.

次元数 D は $2, 5, 10, 20$ とした. 最大評価回数は $10^4 \times D$ 回とし, 試行回数は 15 回とした. 上記を含む全ての実験設定は全て BBOB benchmarks が定めた規定に従っている. また, 手法の性能評価指標には COCO software[*2] を用いて作成した累積経験分布関数 [91, 93] を使用した (付録 C.3 節参照).

各適応手法のパラメータ設定には, 各参考文献の推奨値を用いた. jDE では $\tau_F = 0.1, \tau_C = 0.1$, EPSDE では F-pool$= \{0.4, ..., 0.9\}$, C-pool$= \{0.1, ..., 0.9\}$, JADE では $c = 0.1$ とした. Pošík と Klema の BBOB benchmarks における JADE の性能評価に関する先行研究 [185] を参考に, 集団数 N は $5 \times D$ とした. 突然変異戦略には表 3.1 に示す 8 つの戦略を使用した. current-to-$p$best/1 と rand-to-$p$best/1 の 2 つの制御パラメータは $p = 0.05$, $A = N$ とした

[*2] http://coco.gforge.inria.fr/

Algorithm 17: 一般化した MDE の適応手法

$t \leftarrow 1$, 集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の初期化;
$\mu_F \leftarrow 0.5$, $\mu_C \leftarrow 0.5$;
while 探索終了条件を満たしていない **do**
  $\boldsymbol{S}^F \leftarrow \emptyset$, $\boldsymbol{S}^C \leftarrow \emptyset$;
  for $i = 1$ **to** N **do**
    $F_{i,t} = \mathrm{randc}(\mu_F, 0.1)$;
    $C_{i,t} = \mathrm{randn}(\mu_C, 0.1)$;
    表 3.1 の任意の突然変異戦略により変異個体 $\boldsymbol{v}^{i,t}$ を生成;
    $\boldsymbol{x}^{i,t}$ と $\boldsymbol{v}^{i,t}$ に binomial 交叉 (Algorithm 7), 又は SEC (Algorithm 22) のいずれかの交叉手法を適用し, 子個体 $\boldsymbol{u}^{i,t}$ を生成;
  for $i = 1$ **to** N **do**
    if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
      $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t} \rightarrow \boldsymbol{S}^F$, $C_{i,t} \rightarrow \boldsymbol{S}^C$;
    else
      $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$;
  if $\boldsymbol{S}^F, \boldsymbol{S}^C \neq \emptyset$ **then**
    $c_F \leftarrow \mathrm{rand}(0.0, 0.2]$, $c_C \leftarrow \mathrm{rand}(0.0, 0.1]$;
    $\mu_F \leftarrow (1 - c_F) \cdot \mu_F + c_F \cdot \mathrm{mean}_P(\boldsymbol{S}^F)$;
    $\mu_C \leftarrow (1 - c_C) \cdot \mu_C + c_C \cdot \mathrm{mean}_P(\boldsymbol{S}^C)$;
  $t \leftarrow t + 1$;

[267, 266]. また, 交叉手法には Algorithm 7, 22 に示す binomial 交叉と, 後の 8 章にて説明する Shuffled Exponential Crossover (SEC) を使用した. 3 章にて述べたとおり DE における交叉手法としては exponential 交叉が一般的であるが, 後の 8 章にて指摘する問題点を有するため, 本章では除外した.

適応 DE の適応手法の比較手法として, 通常の DE (Algorithm 9) を用いた. ここで, 通常の DE については, $F = 0.5$, $C = 0.9$ とした. この F と C の設定は, DE アルゴリズムの標準的な設定として, 幅広く使用されている [27, 193, 267].

本章の実験では, 全ての DE アルゴリズムにリスタート戦略を導入した. ここで, リスタート戦略は EA の探索停滞時に集団や制御パラメータを再初期化することで, 探索をやり直す枠組みである [70]. DE におけるリスタート戦略の枠組みはこれまでにいくつか提案されているが [179, 255, 134, 264], 本章では以下の (1), (2) に示す, Zhabitsky と Zhabitskaya に提案されたリスタート戦略 [264] にて使用された戦略を採用した. 本章ではさらに (3) を加えた[*3].

[*3] リスタート条件 (1), (2) のみの場合, いくつかの多峰性関数において, 探索が明らかに停滞しているにも関わらず, 停滞を感知できない場合が予備実験において見られた. そのため, 本論文では (3) を新たに加えた. また, (3) のみと (1), (2), (3) 全てを導入した DE アルゴリズムを比較したところ, 大きな性能差は見られなかったもののいくつかの関数においては後者のほうが優れている場合があった. そのため, 3 つのリスタート条件全てを導入することにした.

DE アルゴリズムの探索中に次のリスタート条件 (1), (2), (3) のいずれかを満たした場合, 集団の全ての個体や適応メタパラメータを再初期化し探索をリスタートする:

(1) 解 $\boldsymbol{x}$ に基づくリスタート条件

次の Δx_j が十分小さい場合, 解空間において DE の探索が十分収束していると見なしリスタートする. 本論文では $\varepsilon_x = 10^{-12}$ とした.

$$\exists j \; \Delta x_j < \varepsilon_x \max_{i=1,\ldots,N} \{x_j^i\} \tag{5.1}$$

$$\Delta x_j = \max_{i=1,\ldots,N} \{x_j^i\} - \min_{i=1,\ldots,N} \{x_j^i\} \tag{5.2}$$

(2) 解の評価値 f に基づくリスタート条件

次の Δf が十分小さい場合, 目的関数空間において DE の探索が十分収束していると見なし, リスタートする. 本論文では $\varepsilon_f = 10^{-12}$ とした.

$$\Delta f < \varepsilon_f | \max_{i=1,\ldots,N} \{f_i\}| \tag{5.3}$$

$$\Delta f = | \max_{i=1,\ldots,N} \{f_i\} - \min_{i=1,\ldots,N} \{f_i\}| \tag{5.4}$$

(3) 最良解の更新頻度に基づくリスタート条件

最後のリスタートからの最良解の更新が n_{stop} 回の評価回数を費やしても行われなかった場合, 探索を停滞と見なしリスタートする. n_{stop} の値は予備実験において多くの場合適切であった $n_{\mathrm{stop}} = 500 \times D$ とした.

5.3.2　実験結果

図 5.1 ～ 5.4 に, 突然変異戦略に rand/1, rand/2, best/1, best/2, current-to-rand/1, current-to-best/1, current-to-$p$best/1, 及び rand-to-$p$best/1 (表 3.1 参照), 交叉手法に binomial 交叉 (Algorithm 7) と SEC (Algorithm 22) を用いた場合における, 2, 5, 10, 20 次元の BBOB benchmarks の 24 個のベンチマーク関数 $f_1 \sim f_{24}$ における実験結果を示す. 図 5.1 ～ 5.4 では, 通常の DE (Algorithm 9), 及び 4 つの一般化した適応 DE (jDE, EPSDE, JADE, MDE) を比較している. また, 8 つの突然変異戦略における最大評価回数 $10^4 \times D$ での時点での平均順位を, 交叉手法ごとに図 5.5 に示す.

以下では, 各手法の実験結果を要約する:

通常の DE

図 5.1 ～ 5.2 から, binomial 交叉を用いた場合は $D = 2$ と対象問題が低次元である際に比較的良好な性能を示す. しかし, $D = 5, 10, 20$ と次元数が増加した際の性能は乏しい. この傾向は, 図 5.5 からも確認できる.

一方, SEC においては, 次元数に依らずに比較的良好な探索性能を示している. 例えば $D = 20$ の rand/1, best/1, best/2, current-to-$p$best/1, rand-to-$p$best/1 の結果のように, 特定の次元数, 及び突然変異戦略との組み合わせにおいては, 比較手法中最良の性

能である．図 5.5 においても，$D = 2$ では 4 位，$D = 5$ では MDE と同順で 4 位であるが，$D = 10$ では 3 位，$D = 20$ では 2 位と $D = 10, 20$ では他の適応 DE と比べても劣らない性能を示している．

jDE

図 5.1 ～ 5.4 から，rand/1, rand/2 を使用した場合，交叉手法に依らず jDE は比較的良好な性能を示している．これは，jDE は本来 rand/1 の使用を前提に設計された適応手法であるためだと思われる．その他にも，例えば図 5.4(b) の current-to-best/1/SEC における結果のように，特定の次元数，及び突然変異戦略と交叉手法の組み合わせにおいて，最も良い性能を示している．

また，図 5.5 から，ここで比較した適応 DE の中では最も古くに提案された手法であるにも関わらず，次元数，及び使用する交叉手法に依らずに jDE は比較的良好な性能を示している．Binomial 交叉においては $D = 2, 5$ で最も良く，$D = 10, 20$ では JADE に次いで良い．また，exponential 交叉を用いた場合は，$D = 5, 10$ では EPSDE に次いで良く，$D = 2, 20$ では最良の性能を示している．

EPSDE

図 5.1 ～ 5.2 から，binomial 交叉を用いた場合は，$D = 5$ においては best/1，及び current-to-rand/1 を用いた場合の結果のように特定の突然変異戦略の下では良好な性能を示すものの，全体的に他手法に劣っている．特に，$D = 2, 20$ においては，図 5.5 では通常の DE よりも劣る性能を示している．

一方，exponential 交叉においては (図 5.3 ～ 5.4)，$D = 5, 10$ では比較的良好な性能を示し，図 5.5 から最良の順位を示していることがわかる．

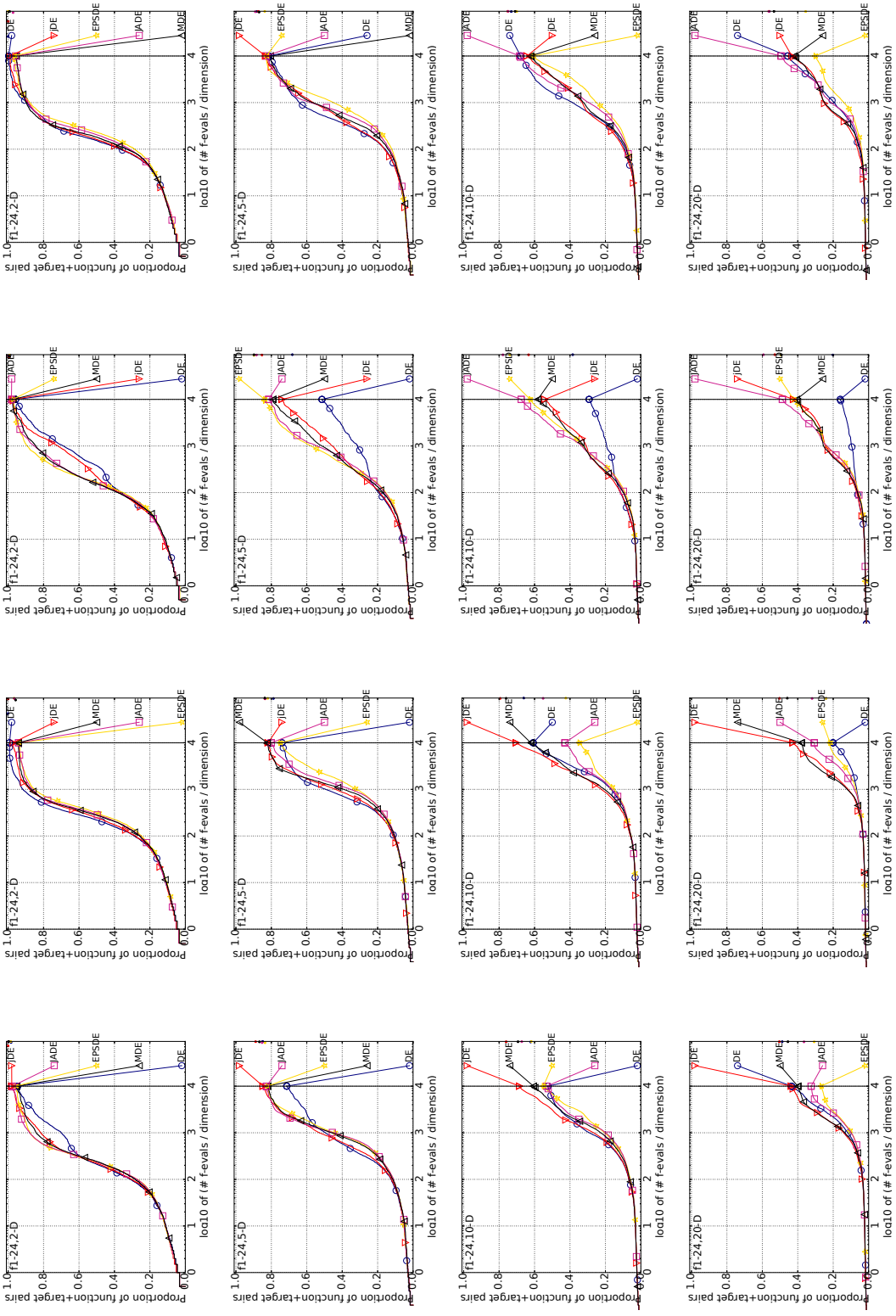
JADE

Binomial 交叉を使用した場合，JADE は比較的良好な性能を示す傾向が見られた．図 5.1 ～ 5.2 から，$D = 10, 20$ において，rand/1, rand/2 における結果と $D = 10$ での current-to-$p$best/1 における結果を除き，最良の性能を示している．この傾向は，図 5.5 からも確認できる．ここで，rand/1, 及び rand/2 における JADE の乏しい性能は，先行研究 [267, 207] での実験結果と一致する．

しかし，SEC を用いた場合，ほぼ全ての突然変異戦略において JADE の性能が他手法と比べ劣るようになる傾向が顕著に見られた．特に，$D = 10, 20$ においては，ほとんどの場合，比較手法中最も劣る性能を JADE は示している (図 5.3 ～ 5.4)．

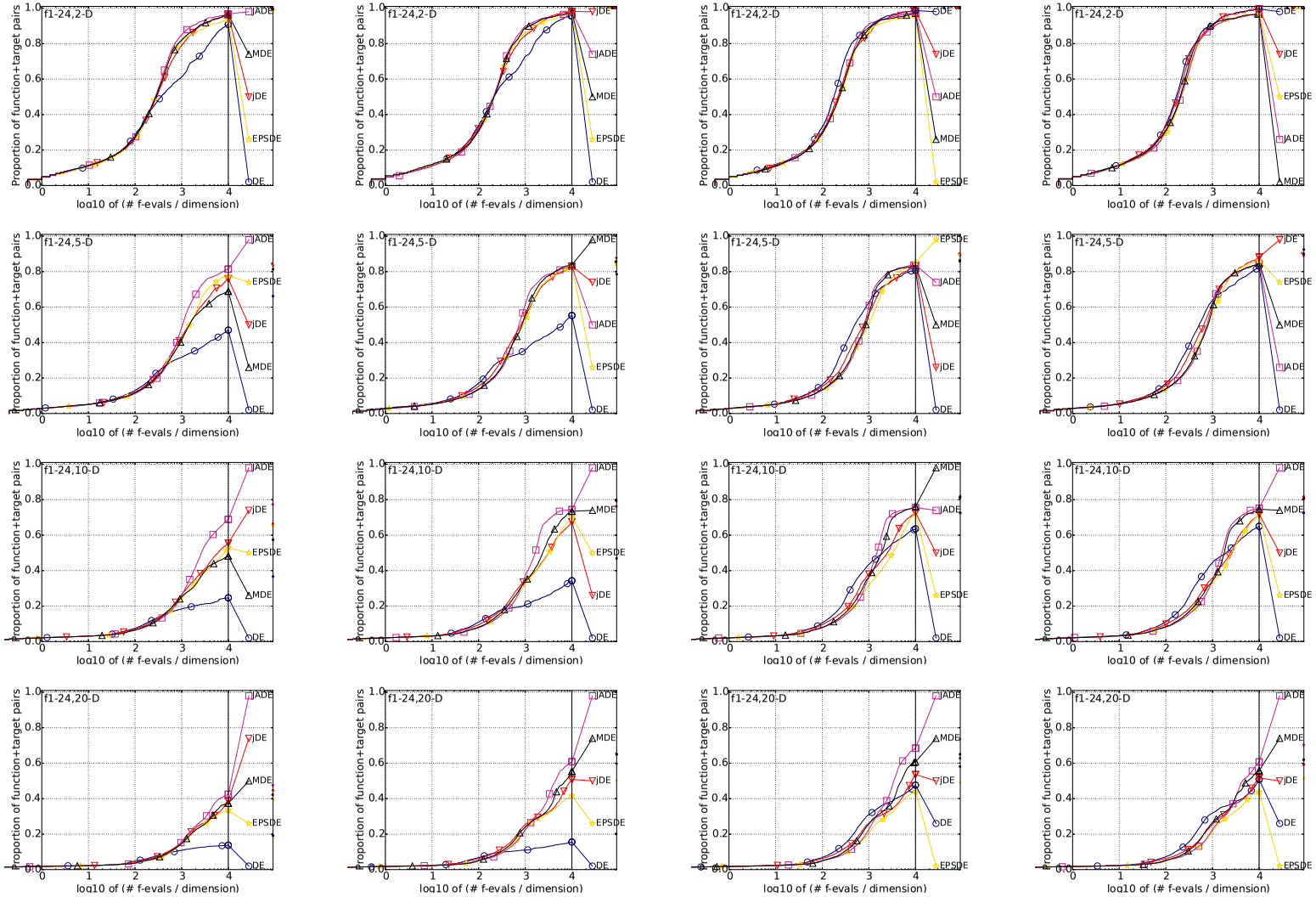
MDE

使用する交叉方法を問わず，図 5.1 ～ 5.4，及び図 5.5 から全体的に MDE は他手法と比べ劣る性能を示している．また，4.4 節にて説明したとおり，MDE は JADE に類似したパラメータ適応手法であるが，SEC を用いた場合の $D = 10, 20$ における結果を除き，多くの突然変異戦略において JADE よりも劣っている．



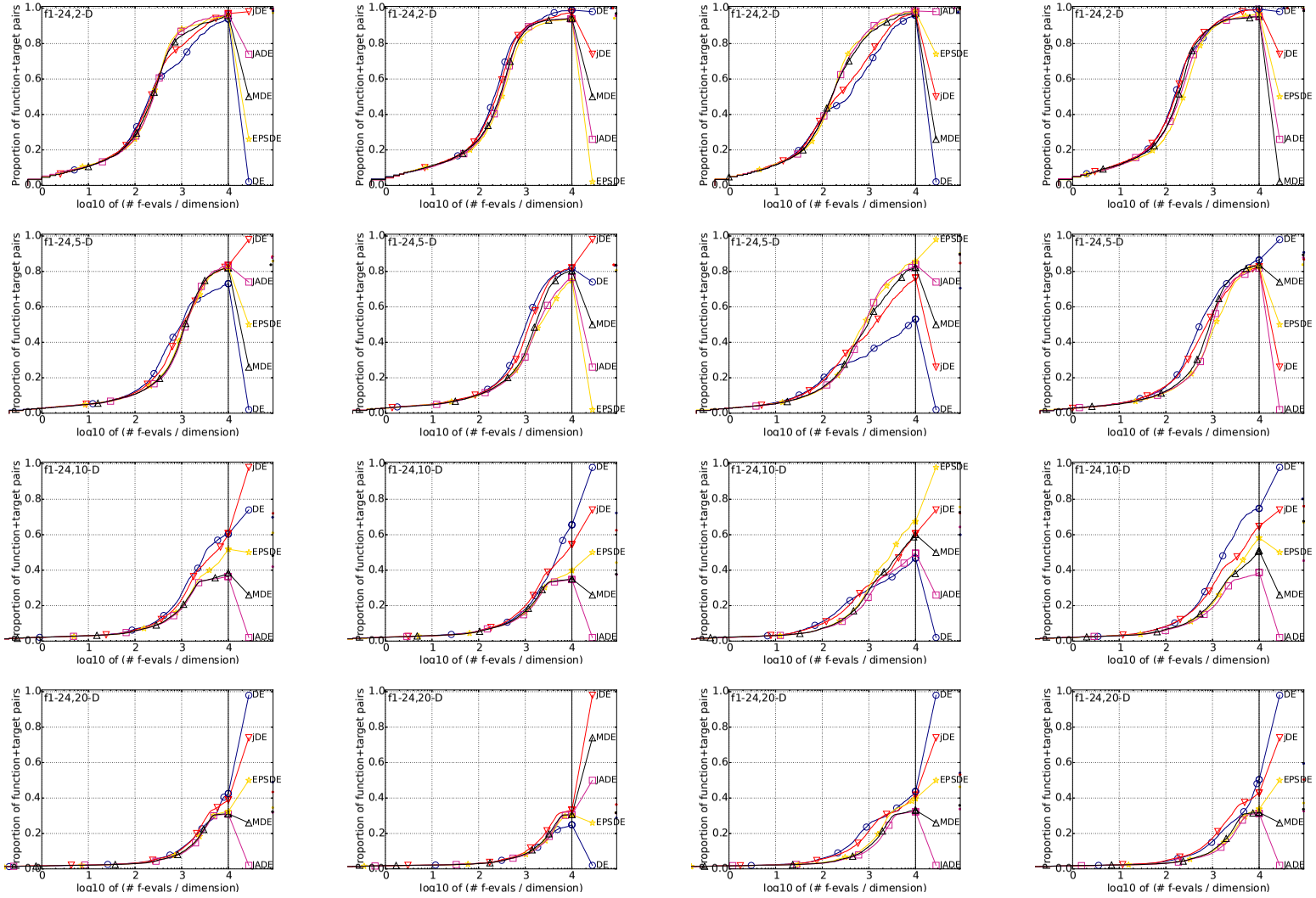
(a) rand/1/bin　(b) rand/2/bin　(c) best/1/bin　(d) best/2/bin

図 5.1: 突然変異戦略に rand/1, rand/2, best/1, 及び best/2, 交叉手法に binomial 交叉を用いた場合における，通常の DE (Algorithm 9)，及び 4 つの一般化した適応 DE (jDE, EPSDE, JADE, MDE) の比較結果 (上から，$D = 2, 5, 10, 20$)．BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している．縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する．横軸は対応する評価回数を次元数で割った値であり，対数スケールを取っている．より詳細は，付録 C.3 節を参照されたい．



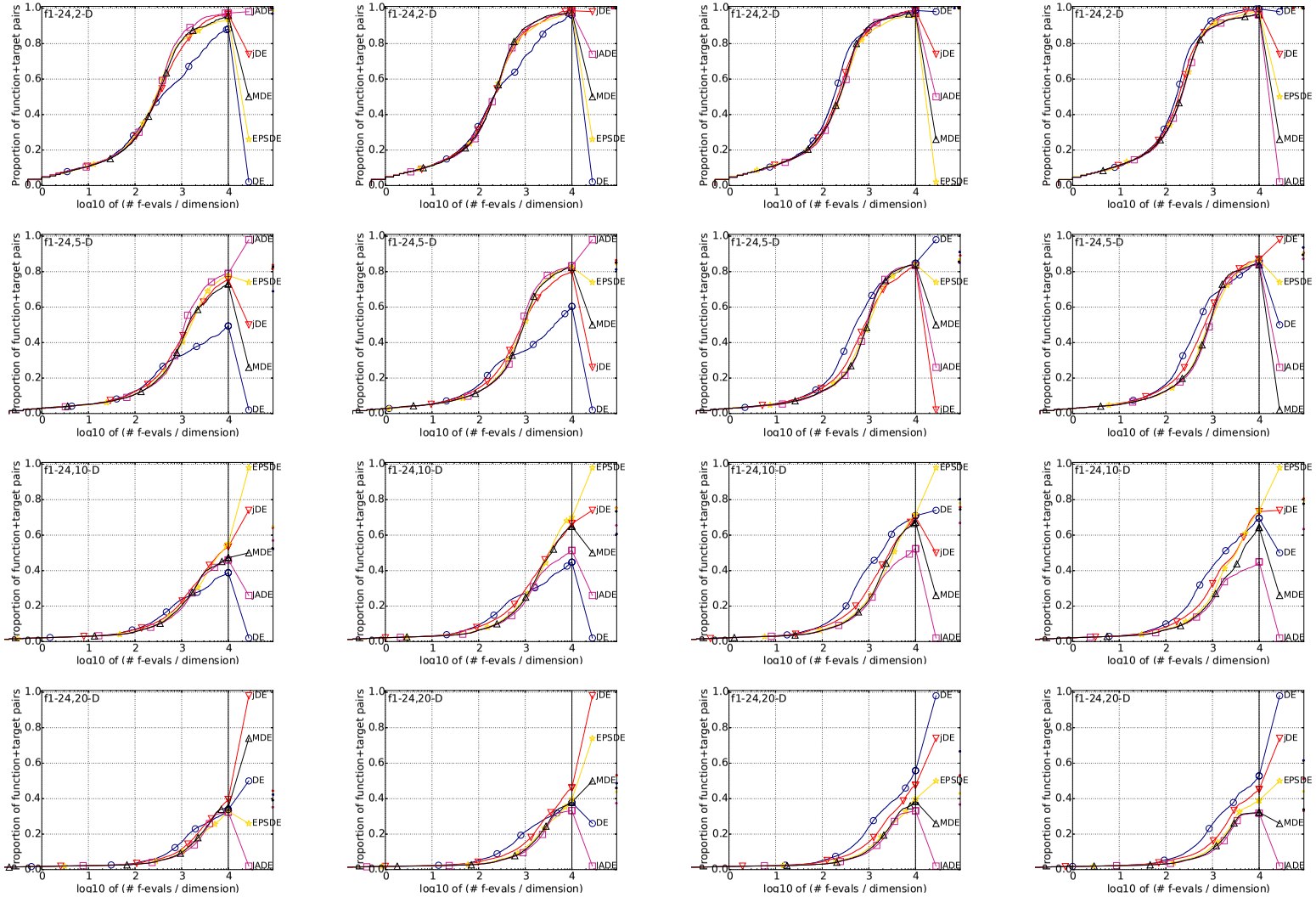
(a) current-to-rand/1/bin (b) current-to-best/1/bin (c) current-to-*p*best/1/bin (d) rand-to-*p*best/1/bin

図 5.2: 突然変異戦略に current-to-rand/1, current-to-best/1, current-to-*p*best/1, 及び rand-to-*p*best/1, 交叉手法に binomial 交叉を用いた場合における, 通常の DE (Algorithm 9), 及び 4 つの一般化した適応 DE の比較結果 (上から, $D = 2, 5, 10, 20$). BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.



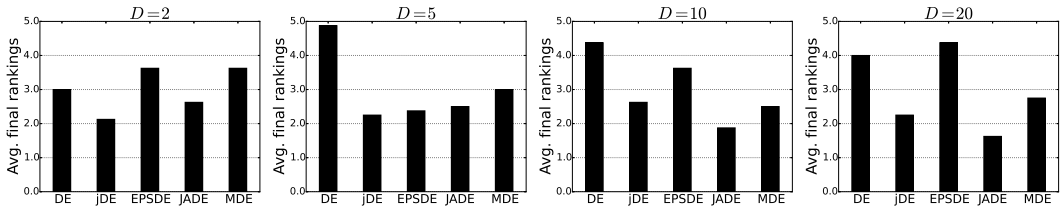
(a) rand/1/SEC (b) rand/2/SEC (c) best/1/SEC (d) best/2/SEC

図 5.3: 突然変異戦略に rand/1, rand/2, best/1, 及び best/2, 交叉手法に SEC を用いた場合における, 通常の DE (Algorithm 9), 及び 4 つの一般化した適応 DE (jDE, EPSDE, JADE, MDE) の比較結果 (上から, $D = 2, 5, 10, 20$). BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.

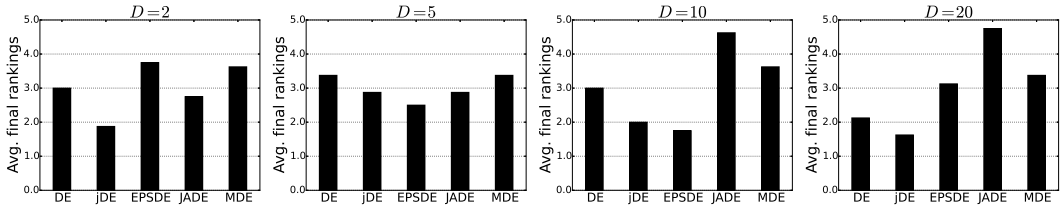


(a) current-to-rand/1/SEC (b) current-to-best/1/SEC (c) current-to-$p$best/1/SEC (d) rand-to-$p$best/1/SEC

図 5.4: 突然変異戦略に current-to-rand/1, current-to-best/1, current-to-$p$best/1, 及び rand-to-$p$best/1, 交叉手法に SEC を用いた場合における, 通常の DE (Algorithm 9), 及び 4 つの一般化した適応 DE の比較結果 (上から, $D = 2, 5, 10, 20$). BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.



(a) binomial 交叉



(b) SEC

図 5.5: 最大評価回数 $10^4 \times D$ の時点での, 8 つの突然変異戦略を使用した場合の交叉手法ごとの平均順位 ($D = 2, 5, 10, 20$). 平均順位が低いほど, どの突然変異戦略においても平均的に優れていることを意味する.

5.3.3　考察

本節では, 5.3.2 節にて得られた実験結果について考察する. 本研究にて使用した jDE, EPSDE, JADE, MDE は, 本来 binomial 交叉の使用を前提として設計された適応手法である. そのためか, JADE のように交叉手法を SEC に変えた途端に性能が大幅に劣化する適応手法も見られた. これは恐らく, 3.3.2 節にて説明したとおり, 交叉により子個体に受け継がれる決定変数の期待値 $E(L)$ が, binomial 交叉と SEC[*4]では異なるためであると考えられる. また, SEC においては対象問題の次元数が増加した場合, 値の範囲によっては $E(L)$ が C に対して鈍感, 又は敏感となりえることも, JADE の乏しい性能の要因の一つであると推測される.

MDE [110] の提案論文では, jDE, JADE などよりも MDE は優れていると結論づけているが, 本実験では MDE の適応手法は多くの場合において jDE, JADE に劣っていた. そのため, MDE 全体の優れた性能は, current-to-gr_best/1 と p-best 交叉の組み合わせによる相乗効果によるものだと考えられる. 一方, jDE は使用する突然変異戦略と交叉手法の組み合わせに依らず, 比較的良好な性能を示していた. また, jDE, EPSDE, JADE, MDE の提案論文では, いずれも探索中においてパラメータの変更を行わない通常の DE (Algorithm 9) よりもこれらの適応 DE は優れていると結論づけているが, 本実験では突然変異戦略と交叉手法の組み合わせによってはこの限りではなかった. この傾向は, SEC を用いた $D = 20$ における結果では顕著となった.

全体的に, あらゆる突然変異戦略と交叉手法の組み合わせにおいて他手法よりも優れている

[*4] SEC の $E(L)$ は, exponential 交叉と同様である (8 章参照).

適応手法は, 本実験では確認されなかった. このことは, 特定のオペレータの組み合わせに適した適応手法を使用することの有用性を示唆している, 有益な知見である. 例えば, 突然変異戦略を適応的に選択する枠組みがこれまでに提案されてきているが [189, 157, 62, 79], 著者の知る限り, 選択されたオペレータに依らずに同一の適応手法を使用している. この点について, 本実験で得られた知見から, 各オペレータに適した適応手法を使用することでさらなる性能向上が期待できる.

多くの先行研究 [189, 267, 157, 110] では, 次元数に依らず提案手法が優れていると結論づけており, 次元数に対する性能変化について議論した研究は, 著者の知る限りほとんどない. しかし, 図 5.1 ～ 5.4, 及び図 5.5 から適応手法の優劣関係は次元数に依存することがわかる. このことから, これまではベンチマーク関数ごとの優劣しか議論していなかったが, 適応手法の性能評価をする際には次元数についても考慮する必要があると言える.

5.4 Oracle パラメータに基づくシミュレーション法

5.3 節では, 様々な突然変異戦略と交叉手法の組み合わせにおける適応 DE の適応手法の性能を, BBOB benchmarks [91, 93] にて評価した. これにより, 「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しを評価し, どの適応 DE の適応手法が一般的に, 又は特定のオペレータを用いた場合に優れているのかを明らかにした. しかし, 適応手法の違いが DE アルゴリズムの探索性能差に影響を及ぼす理由は不明である. 例えば, 5.3 節の binomial 交叉を用いた場合における実験結果から, 「JADE の適応手法は優れている」ということは言える. しかし, 「なぜ JADE の適応手法は優れているのか?」という点に関しては明らかではない.

本節ではこの問題に取り組むために, 「適応手法」の良し悪しを「遺伝的オペレータ」とは独立して評価する枠組みとして, oracle パラメータを用いた新たなシミュレーション法を提案する. oracle パラメータは理想的なパラメータ適応の軌跡の代理であり, 任意の oracle 関数によって与えられる. 複数の oracle 関数を使用することで, 様々な形状, 及び傾向を有する oracle パラメータへの適応手法の追従能力に関する知見が得られる. これにより, これまで困難であったパラメータ適応手法の適応能力についての議論を可能とする.

始めに, 5.4.1 節にて伝統的な視覚に基づく適応手法の解析方法の限界について述べる. 次に, oracle パラメータの概念を説明するための準備として, 5.4.2 節にて優れた適応手法と理想的なパラメータの定義について述べる. そして, 5.4.3 節にて oracle パラメータを用いたシミュレーション法を説明する. 最後に, 5.4.4 節にて提案シミュレーション法の妥当性について議論する.

5.4.1 視覚に基づく適応手法の解析法の限界について

5.1.2 節にて述べたとおり, 適応 DE の適応手法の振る舞いを調査した先行研究は少なからず存在するが [27, 30, 189, 267, 157, 36], それらの多くは適応メタパラメータ, 又は F, C の値を図に示し, 視覚に基づく定性的な議論にとどまっている. 視覚に基づく解析法の限界につ

いて議論するために, jDE, EPSDE, JADE, MDE を BBOB benchmarks の 10 次元の f_3, f_8 に適用した際の, 探索経過に対する各適応手法の適応メタパラメータの推移を図 5.6 に示す. なお, 突然変異戦略には current-to-$p$best/1, 交叉手法には binomial 交叉を使用している. 横軸から, f_3 では JADE, MDE, jDE, EPSDE, f_8 では JADE, MDE, EPSDE, jDE の順により少ない評価回数で最適解に到達できていることがわかる.

先行研究 [27, 30, 189, 267, 157, 36] ではこのようにして適応手法を解析しようと試みていた. しかし, 図 5.6 からわかることは, 次の定性的な情報のみである:

(i) 異なる適応 DE の適応手法は, 異なる適応プロセスを示す.
(ii) 適応手法は, 異なる問題ごとに異なる適応プロセスを示す.
(iii) 同一の問題においても, 探索経過状況に応じて適応メタパラメータは異なる値を示す.

以上の 3 点から得られる情報は限られており, 適応手法の違いがなぜ DE アルゴリズムの探索性能差に影響を及ぼすのかは不明である. 例えば, 5.3 節の binomial 交叉を用いた場合における実験結果から, 「JADE の適応手法は優れている」ということは言える. しかし, 「なぜ JADE の適応手法は優れているのか?」という点に関しては明らかではなく, 図 5.6 を用いてこれを定量的に説明することは, 容易ではない. 以上に議論したように, 視覚に基づく適応手法の解析法から得られる情報は限られており, このアプローチにより DE アルゴリズムの探索性能と適応手法の性能を解析することは困難である.

5.4.2　優れた適応手法と理想的なパラメータの定義

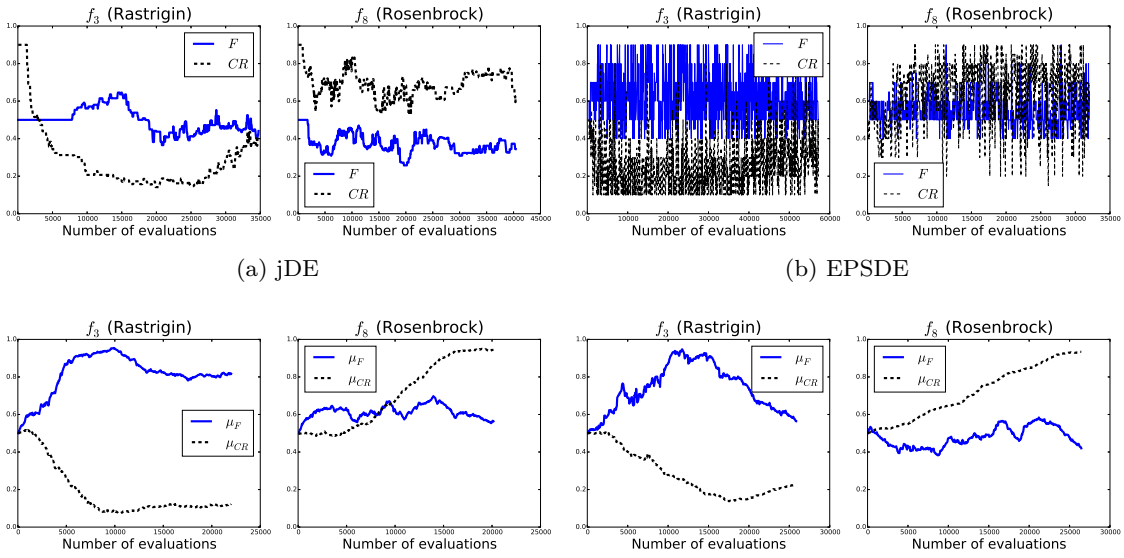
5.4.1 節にて議論したとおり, 多くの先行研究で行われている視覚に基づく適応手法の解析法では, DE アルゴリズムの探索性能と適応手法の性能を結び付けて議論することは困難である. これは, 5.1.2 節にて述べたとおり, 良好な目的関数値 $f(\boldsymbol{x})$ を有する解 $\boldsymbol{x}$ を生成できたのは, 遺伝的オペレータと n 個の制御パラメータの集合 $\boldsymbol{p} = \{p_1, ..., p_n\}$ の相乗効果のためであり, 適応手法の良し悪しを遺伝的オペレータとは独立して評価することは困難なためである.

そこで, oracle パラメータを用いた新たなシミュレーション法を本章では提案する. oracle パラメータは理想的なパラメータ適応の軌跡の代理であり, 任意の oracle 関数によって与えられる. これまでは, 5.3 節にて行ったように解 $\boldsymbol{x}$ の目的関数値 $f(\boldsymbol{x})$ からパラメータ $\boldsymbol{p}$ の良し悪しについて議論しており, $\boldsymbol{p}$ を独立して評価することは困難であった. 一方, 本シミュレーション法では $\boldsymbol{p}$ を独立して評価することが可能であり, これまで難しかったパラメータ適応手法の適応能力についての議論が可能である.

本節では oracle パラメータを説明するための準備として, 理想的な (ideal) パラメータ θ^* について述べる. まず, 本研究では「理想的な適応手法」を次のように定義する:

理想的な適応手法の定義　理想的な適応手法は, 異なる対象問題, 及び変化する探索状況において理想的なパラメータを生成可能である.

次に「優れた適応手法」を以下のように定義する:



(a) jDE (b) EPSDE

(c) JADE (d) MDE

図 5.6: current-to-$p$best/1/bin を用いた各適応 DE の適応手法を 10 次元の f_3, f_8 に適用した際の, 評価回数の経過に対する適応メタパラメータの推移. 15 試行中の最適解に到達できた各試行について, それまでに費やした評価回数が中央値である 1 試行のデータを示している. jDE については全ての個体に割り当てられている F, C の値の中央値, EPSDE については各世代において成功した F, C の値の中央値を図示している. また, JADE, MDE については μ_F, μ_C の値を示している.

優れた適応手法の定義 優れた適応手法は, 上記の理想的な適応手法に近いパラメータを生成可能である.

DE の制御パラメータである F, C の適切なパラメータ設定は対象問題ごとに異なり, かつ探索状況に応じて刻々と変化することが, 一般的に知られている [71, 196, 274, 27, 267]. そのため, 上記の定義はこれまでの DE のパラメータ調査研究, 及び適応 DE の設計に関する知見から逸脱することのない定義である. また, 上記の定義は適応 DE のみならず, 他の EA においても当てはまると考えられる. 上記のような優れた適応手法を組み込んだ DE アルゴリズムは, 優れた探索性能を実現できると考えられる.

次に, 上記の定義に沿った, 適応手法の評価方法を考える. 本章の 5.3 節では 8 つの突然変異戦略と 2 つの交叉手法を組み合わせた計 16 種類のオペレータの下で jDE, EPSDE, JADE, MDE の適応手法を評価した. 一方, ここでは使用する突然変異戦略や交叉手法といったオペレータとは独立した適応手法の評価方法について考える. まず, 上記の定義における「理想的なパラメータ」を次のように定める:

理想的なパラメータの定義

探索中のある地点において, 予め定めた DE アルゴリズムの性能評価指標を最適化するパラメータを, 理想的なパラメータと呼ぶ.

ここで, DE アルゴリズムの性能評価指標とは, 探索中に得られる解の質 [136], Expected Run Time (ERT) [91, 93], anytime 性能 [275] などの任意の指標である.

適応手法を解析する上で, 理想的なパラメータ値 θ^* の概念は有益である. 例えば, 5.3 節でも使用した jDE, EPSDE, JADE, MDE の適応手法を実行し, θ^* にどの程度近いパラメータ値を生成できるかを調査すれば, 「DE アルゴリズムの性能評価指標」とは独立して, 適応手法の良し悪しを定量的に評価可能である. また, 突然変異戦略や交叉手法といったオペレータからも独立した, 一般性のある知見が得られることが期待される.

しかし, 上記で定めた理想的なパラメータ値 θ^* を知ることは, いくつかの理由から困難である. まず, 適切なパラメータ値は探索状況に応じて変化するため, 例えば $F = 0.5$, $C = 0.9$ といったような固定された値ではない. そのため, 各探索経過地点, 例えば評価回数の経過 $1, 2, 3, ...$ ごとに, $\theta_1^*, \theta_2^*, \theta_3^*, ...$ のように異なる理想的なパラメータ値が存在する. しかし, これまでにどのようなパラメータ値を使用して個体を生成してきたかによって探索状況が変化するため, ある地点 t における θ_t^* を独立して求めることはできない. そのため, 理想的なパラメータ列 $\boldsymbol{\theta}^* = (\theta_1^*, \theta_2^*, \theta_3^*, ...)$ を求めることは困難である.

5.4.3　Oracle パラメータを用いたシミュレーション法

5.4.2 節にて述べたとおり, 適応 DE の適応手法を解析するために理想的なパラメータ列 $\boldsymbol{\theta}^* = (\theta_1^*, \theta_2^*, \theta_3^*, ...)$ は有用であるが, 実際に $\boldsymbol{\theta}^*$ を得ることはほぼ不可能である. そこで, 本研究では $\boldsymbol{\theta}^*$ の代換え案として, $\boldsymbol{\theta}^*$ の代理を人為的に作成し, これを用いて適応手法の性能を評価するシミュレーション法の枠組みを提案する. ここで, 任意の oracle 関数によって与えられる人工的に作成した $\boldsymbol{\theta}^*$ を, oracle パラメータ $\boldsymbol{\theta}^o = (\theta_1^o, \theta_2^o, \theta_3^o, ...)$ と呼ぶ.

実際の問題における $\boldsymbol{\theta}^*$ を適応手法が追従しているかを確認することは困難であるが, もし $\boldsymbol{\theta}^*$ が例えば直線, 正弦波, 又はランダムウォークの性質といった「ある傾向」を持つことを前提とした場合, 適応手法がその傾向にどの程度追従できるかを把握できる. これにより, ベンチマーク関数, 対象問題の次元数, 使用するオペレータの種類といった様々な要因とは独立に, 適応手法の適応能力が解析可能である.

4.4.2 節にて述べたとおり, 多くの適応 DE では各世代 t の集団中 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の各個体 $\boldsymbol{x}^{i,t}$, $i \in \{1, ..., N\}$ に何らかの方法で $F_{i,t}$, $C_{i,t}$ を割り当て, そのパラメータを使用して子個体 $\boldsymbol{u}^{i,t}$ を生成する. そして, 成功したパラメータ, つまり式 (3.3) に示す DE の生存選択において親個体 $\boldsymbol{x}^{i,t}$ よりも優れた子個体 $\boldsymbol{u}^{i,t}$ を生成できた $F_{i,t}$, $C_{i,t}$ を何らかの方法で次世代 $t+1$ のパラメータ適応に反映させる. このような (1) パラメータ $F_{i,t}$, $C_{i,t}$ の生成, (2) その評価 (成功か失敗かの判定), (3) 次世代へのフィードバックといった一連の試行錯誤を繰り返すことで, 探索状況に適したパラメータ値へと調整していく. このとき, (2) の生成したパラメータ値が成功か否かの判定を前節で述べた oracle パラメータを用いて行う[*5].

[*5] DE の特徴として, 式 (3.3) に示すように親個体 $\boldsymbol{x}^{i,t}$ よりも優れた子個体 $\boldsymbol{u}^{i,t}$ を生成できた場合, つまり $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ となった場合, 必ず $\boldsymbol{x}^{i,t+1}$ は $\boldsymbol{u}^{i,t}$ となる. 本シミュレーション法はこの DE の決定的な生存選択モデルを前提とし, その特徴を利用している. 一方, 2.4 節で述べた生存選択が決定的ではない GA な

Algorithm 18: Oracle パラメータを用いたシミュレーション法の概要

```
1  t ← 1;
2  while 最大サンプリング数を満たしていない do
3      θ^o_t を予め定めた oracle 関数により求める;
4      for i = 1 to N do
5          JADE における式 (4.6), (4.7) のように, 各適応 DE の適応手法のサンプリング法により,
           パラメータ θ_{i,t} を生成;
6      for i = 1 to N do
7          if rand[0, 1] ≤ p_a(θ_{i,t}) then
8              θ_{i,t} を成功とする;
9          else
10             θ_{i,t} を失敗とする;
11     JADE の μ_F と μ_C の更新式である式 (4.8), (4.9) のように, 各適応 DE の適応手法により, θ
       に関する適応メタパラメータを更新;
12     t ← t + 1;
```

Oracle パラメータを用いたシミュレーション法の概要を Algorithm 18 に示す．ここで，パラメータ $\theta_{i,t}$ は，各個体 $\boldsymbol{x}^{i,t}$ の (i) $F_{i,t}$, (ii) $C_{i,t}$, (iii) $F_{i,t}$ と $C_{i,t}$ の対のいずれかを意味する．各世代 t の始めに，oracle パラメータである θ_t^o を予め定めた oracle 関数により求める[*6]．ここで，本研究で使用した oracle 関数については，後述の 5.5.1 節を参照されたい．Algorithm 18 の line 4–5 にて，例えば jDE ならば式 (4.4), (4.5), JADE ならば式 (4.6), (4.7) といったように，解析対象となる各適応手法に従い $\theta_{i,t}$ を生成する．次に，line 6–10 にて生成したパラメータ $\theta_{i,t}$ が成功したか否かを評価する．本研究では以下の式により定めた $p_a(\theta_{i,t}) \in [0, \beta]$ の確率で $\theta_{i,t}$ を成功とする:

$$p_a(\theta_{i,t}) = \max(-\alpha d_{i,t} + \beta, 0) \tag{5.5}$$

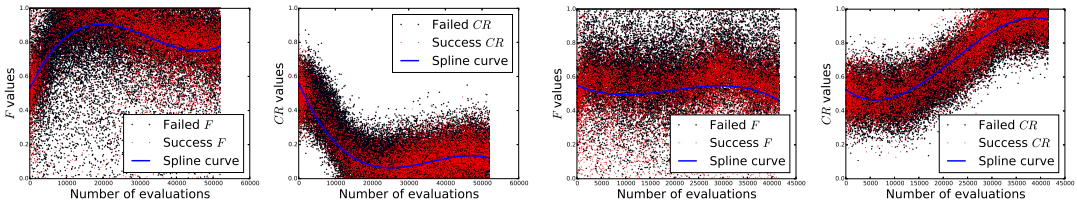
ここで，$d_{i,t}$, $i \in \{1, ..., N\}$ は $\theta_{i,t}$ と世代 t における oracle パラメータ θ_t^o の距離 $d_{i,t} = |\theta_{i,t} - \theta_t^o|$ である．また，α と β は本モデルの制御パラメータであり，α は傾きを，β は $p_a(\theta_{i,t})$ の最大値を調整する．距離 $d_{i,t}$ の値が小さいほど，つまり $\theta_{i,t}$ が oracle パラメータ θ_t^o に近いほど，成功する確率値 $p_a(\theta_{i,t})$ は高くなる．また，$d_{i,t} = 0$, つまり $\theta_{i,t} = \theta_t^o$ にて $p_a(\theta_{i,t})$ は最大確率値 β となる．反対に $d_{i,t}$ の値が大きいほど，つまり $\theta_{i,t}$ が θ_t^o から遠くに位置するほど，成功する確率値 $p_a(\theta_{i,t})$ は低くなる．

各世代 t の終わりに，Algorithm 18 の line 6–10 の判定に従い，例えば JADE ならば式 (4.8), (4.9) のように，各適応手法に従い $\theta_{i,t}$ に関する適応メタパラメータを更新する．

最後に，本シミュレーションモデルにおける適応手法の性能評価指標は，次式で求められる

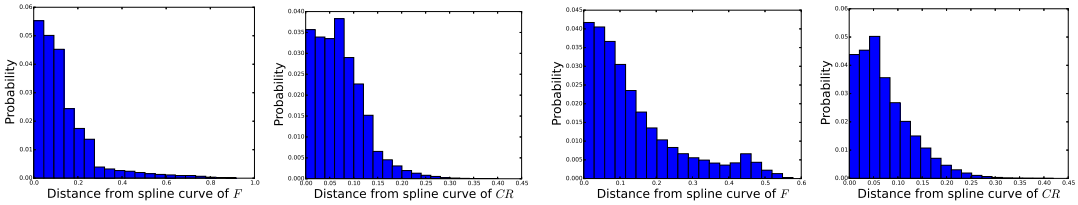
どの一部の他の EA の場合，本シミュレーション法をそのまま適用することはできない．

[*6] Oracle パラメータを費やした評価回数に依存させることも可能であるが，本研究では経過世代数 t に依存させることにした．



(a) f_3 (Rastrigin 関数)　　(b) f_8 (Rosenbrock 関数)

図 5.7: 10 次元の f_3, f_8 に current-to-$p$best/1/bin を用いた JADE を適用した際に生成した, 成功, 及び失敗した全てのパラメータの図. 左が F, 右が C に関する図であり, 横軸は評価回数の経過, 縦軸は F, C の値を表す. また, 成功したパラメータ列についての平滑化スプライン曲線も示している. 15 試行中の最適値に到達できた各試行について, それまでに費やした評価回数が中央値である 1 試行のデータを示している.



(a) f_3 (Rastrigin 関数)　　(b) f_8 (Rosenbrock 関数)

図 5.8: 10 次元の f_3, f_8 に current-to-$p$best/1/bin を用いた JADE を適用した際に生成した, 成功したパラメータ列の平滑化スプライン曲線からの距離に対する成功率の推移. 左が F, 右が C に関する図である. 15 試行にて得られた全てのデータを使用している.

成功したパラメータ数の割合 $r_s \in [0, 1]$ である:

$$r_s = \frac{\text{成功したパラメータの数}}{\text{生成したパラメータの数}} \tag{5.6}$$

r_s の値が大きいほど, 理想的なパラメータ適応の軌跡の代理である oracle パラメータ列 $\boldsymbol{\theta}^o$ を沿うように適応手法は機能できる, つまり 5.4.2 節にて定めた「優れた適応手法」であることを意味する. 異なる適応手法を実行することにより得られた r_s の値を比較することにより, これまで困難であった適応手法の適応能力についての議論が可能と成る.

5.4.4　Oracle パラメータを用いたシミュレーション法の妥当性

本節では, 5.4.3 節にて説明した oracle パラメータを用いたシミュレーション法の妥当性について議論する.

本シミュレーション法は, 5.4.2 節で定義した「優れた適応手法」という指標の下に成り立つ. 本来ならば理想的なパラメータ列 $\boldsymbol{\theta}^*$ を使用することが望ましいが, 5.4.2 節で述べたように $\boldsymbol{\theta}^*$ を知ることはほぼ不可能である. そのため, 探索過程の一部が一次関数, 周期的な関数, ランダムな関数でモデル化できたとした場合, そのようなパラメータの変動に追従可能かを検

証するために, 様々な oracle パラメータの形状を用意してシミュレーションを行う必要がある. 後述の 5.5 節では, 一次関数, sin 関数, ランダムウォークの 3 種類の特徴的な oracle 関数を用いて実験を行うが, より現実的な, つまり理想的なパラメータを再現するような oracle 関数の導入は, 今後の課題としてあげられる.

式 (5.5) では oracle パラメータからの距離が離れるほど, 生成したパラメータ θ が成功する確率値 $p_a(\theta)$ が低くなる. これは理想的なパラメータ θ^* からの距離が離れるほど現在の探索状況に不適切なパラメータであり, 良い個体を生成し式 (3.3) の生存選択による個体の入れ替えが起こる可能性が低くなることを仮定している. 以下ではこの設定, 及び仮定の妥当性について述べる.

図 5.7 に 10 次元の f_3, f_8 に current-to-$p$best/1/bin を用いた JADE を適用した際に生成した, 成功, 及び失敗した全てのパラメータの図を示す. ここで, 本研究にて使用した 4 つの適応 DE (jDE, EPSDE, JADE, MDE) の適応手法の内, 5.3 節の実験では JADE が比較的良好な性能を示していたため, JADE をここでは使用することにした. また, 成功したパラメータ列についての平滑化スプライン曲線も示している. これは理想的な oracle パラメータ列の粗い近似とみなすことができる. JADE では式 (4.6), 及び (4.7) に示すように, コーシー分布, 及び正規分布に従う乱数を使用して F, 及び C をそれぞれ生成している. そのため, ある程度多様なパラメータを生成しているが, 平滑化スプライン曲線から離れるほど失敗しているパラメータが多いように思える.

これをより詳しく調査するために, 図 5.8 に 10 次元の f_3, f_8 に current-to-$p$best/1/bin を用いた JADE を適用した際に生成した, 成功したパラメータ集合についての平滑化スプライン曲線からの距離に対する成功率の推移を示す. 図 5.8 から, F, C といったパラメータの種類, 及びベンチマーク関数に依らず, 成功したパラメータ集合の平滑化スプライン曲線からの距離が離れるほど, ほぼ単調に成功率が下がっている.

以上より, 理想的な oracle パラメータから離れるほど, 良い個体を生成する確率が減少, つまり生成したパラメータ θ が成功する確率が低くなるという仮定は, 現実と乖離することはないと考えられる. そのため, 式 (5.5) はある程度妥当なモデルと考えられる.

5.5　実験 2: oracle パラメータを用いたシミュレーション法による適応手法の適応能力の解析

Oracle パラメータを用いたシミュレーション法を 5.4 節にて提案した. 本節では, 提案シミュレーション法による適応 DE の適応手法のパラメータ適応能力の評価実験について述べる.

5.3 節では様々なオペレータを使用した場合における各適応手法の探索性能, つまり「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しを BBOB benchmarks にて評価した. 一方, 本節では「適応手法」の良し悪しを「遺伝的オペレータ」から独立して評価し, 特定の適応手法が優れた探索性能を示す理由, 高性能な適応手法を設計するためにはどのような機能が

求められているのかを明らかにすることを目的とする．

始めに 5.5.1 節にて，oracle 関数の設定を含む本節の実験設定について述べる．次に，5.5.2 節では 3 種類の oracle 関数における実験結果について説明する．最後に 5.5.3 節にて，本節にて得られた結果と 5.3 節での BBOB benchmarks における実験結果について考察する．

5.5.1　実験設定

5.3 節での実験と同様に，jDE, EPSDE, JADE, MDE の 4 つの適応 DE の適応手法を評価対象とした．式 (5.6) にて定めた成功したパラメータ数の割合 r_s を，各適応手法の性能評価指標に用いた．最大評価回数は 5×10^4 回とし，試行回数は 101 回とした．集団数 N は 50 とした．

EPSDE と jDE を除き，各適応手法の制御パラメータは，5.3.1 節にて述べた設定と同じ値を使用した．EPSDE は F と C について，それぞれ $[0.4, 0.9]$, $[0.1, 0.9]$ の範囲から離散的に 0.1 刻みで保管されている F-pool, C-pool を使用する．そのため，EPSDE はこの範囲外のパラメータを生成することが不可能である．公平な比較のため，他の適応手法と同じ範囲にするために，F, C のプールともに $[0.0, 1.0]$ の範囲から離散的に 0.1 刻みで設定，すなわち F-pool $= \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, C-pool $= \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ とした．また，jDE は $F_{i,t}$ を式 (4.4) を用いて $[0.1, 1.0]$ の区間内に生成するが，EPSDE と同様の理由で $[0.0, 1.0]$ 区間内に変更した．JADE, MDE の適応メタパラメータの初期値に合わせ，oracle パラメータの初期値 θ_1^o は 0.5 とした．それに伴い，jDE の $C_{i,t}$ の初期値を 0.5，EPSDE の初期パラメータセットはランダム生成ではなく 0.5 とした．JADE の学習率 c などの制御パラメータの設定の妥当性に関する議論は，後の 6.5.2 節を参照されたい．

式 (5.5) において，α と β は本シミュレーションモデルの制御パラメータである．α の値を $0.5, 1, 3$ と変えて予備実験をしたところ，α が増加するにつれ単調に成功率が減少していく傾向が，ほぼ全ての適応手法において見られた．しかし，定性的な傾向には影響がほとんど無かったため，$\alpha = 1$ における結果のみを示す．一方，β については $[0.1, 1.0]$ の範囲にて 0.1 刻みで変化させた．

Oracle 関数の設定

本節の実験にて使用する oracle 関数について述べる．世代 t における oracle パラメータ θ_t^o は，以下で説明する oracle 関数 $o(n_t)$ により与えられるものとする．n_t は世代 t までに費やした評価回数を最大評価回数で割った値であり，$n_t \in (0, 1]$ である．ここで，oracle パラメータを用いたシミュレーション法においては，世代 t や評価回数は探索の経過を表すのではないことに注意されたい．

Oracle 関数 $o(n_t)$ の設計には (1) 一次関数[*7], (2) sin 関数, (3) ランダムウォークの 3 種類

[*7] 指数関数を oracle 関数 $o(n_t)$ に使用する実験を行ったが，一次関数を用いた実験結果と大きな差は見られなかった．

の異なる関数を使用した．F, C に依らず, パラメータ θ の範囲は $[0.0, 1.0]$ とし, oracle パラメータ θ^o が取り得る範囲は $[0.1, 0.9]$ とした.

始めに, (1) 一次関数を用いた 2 種類の oracle 関数を次に示す:

$$o_{inc}(n_t) = 0.4n_t + 0.5 \tag{5.7}$$
$$o_{dec}(n_t) = -0.4n_t + 0.5 \tag{5.8}$$

ここで, 式 (5.7), (5.8) は, 0.5 からそれぞれ 0.9, 及び 0.1 まで線形に増加, 及び減少する関数である．式 (5.7), (5.8) において傾きを 0.4 としたのは, 先述の oracle パラメータ θ^o が取り得る範囲 $[0.1, 0.9]$ 内に, $o_{inc}(n_t), o_{dec}(n_t)$ の値が収まるようにするためである．単調に増加, 及び減少する 2 種類の oracle 関数 $o_{inc}(n_t)$, $o_{dec}(n_t)$ を使用することで, 各適応手法の適応傾向に偏りがあるかどうかを調査する.

次に, (2) sin 関数を用いた oracle 関数を以下に示す:

$$o_{sin}(n_t) = 0.4 \cdot \sin(\omega n_t) + 0.5 \tag{5.9}$$

式 (5.9) において, 振幅は 0.4, 初期位相は 0.5 とした．そして, 角周波数 ω の値を 10, 20, 30, 40 と変更することで, oracle パラメータの変動具合の強弱を調整した．sin 関数を用いた oracle 関数 $o_{sin}(n_t)$ により, 周期的に変化する oracle パラメータを各適応手法がどの程度まで追従できるかを調査する.

最後に, (3) ランダムウォークを用いた oracle 関数を以下に示す:

$$o_{rw}(n_t) = o_{rw}(n_{t-1}) + s \cdot \mathrm{rand}[-1, 1] \tag{5.10}$$

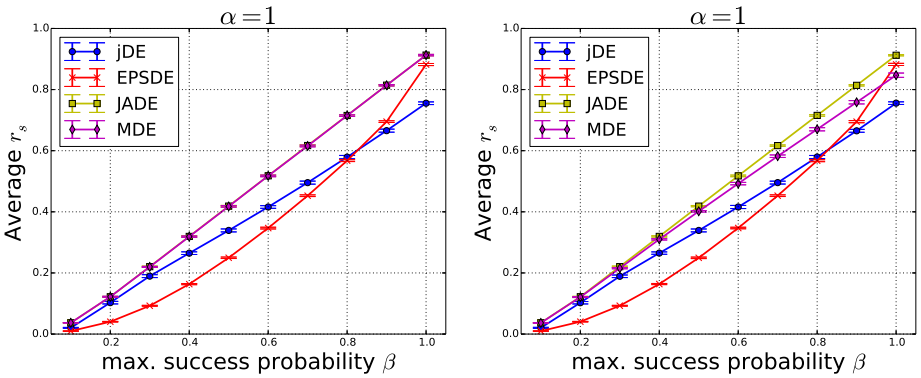
ここで, $o_{rw}(n_1) = 0.5$ である．s はランダムウォークのステップサイズを表しており, 各世代ごとに加えられる摂動の大きさを調整する．本実験では, s を 0.01 から 0.1 まで, 0.01 刻みで変化させた．ただし, $o_{rw}(n_t)$ の値が境界値 $[0.1, 0.9]$ のいずれかを超えた場合は, 以下のように超過分を反射させることで修正した:

$$o_{rw}(n_t) = \begin{cases} 2 \times 0.9 - o_{rw}(n_t) & \text{if } o_{rw}(n_t) > 0.9 \\ 2 \times 0.1 - o_{rw}(n_t) & \text{if } o_{rw}(n_t) < 0.1 \end{cases} \tag{5.11}$$

また, 101 試行分のランダムウォークのデータを予め用意し, 各手法における実験環境を統一した．式 (5.9) の sin 関数を用いた oracle 関数 $o_{sin}(n_t)$ とは対照的に, $o_{rw}(n_t)$ にて実験を行うことで, 不定に変化する oracle パラメータに各適応手法がどの程度まで追従可能かという知見が得られる.

5.5.2 各 oracle 関数における実験結果

本節では, 式 (5.7), (5.8), (5.9), (5.10) の各 oracle 関数における実験結果を述べる．なお, Algorithm 18 のパラメータ θ を, (i) C, (ii) F, (iii) F と C のペアの 3 通りに変えて実験を行ったところ, 各適応手法間における結果の傾向に変化はほとんど見られなかった．そのため, 本章では (i) C の結果のみを記載する.



(a) oracle 関数 $o_{inc}(n_t)$　　(b) oracle 関数 $o_{dec}(n_t)$

図 5.9: 式 (5.7), 及び (5.8) に示す oracle 関数 $o_{inc}(n_t)$, 及び $o_{dec}(n_t)$ を用いたシミュレーション実験の結果. 横軸は式 (5.5) における β の値, 縦軸は式 (5.6) の r_s の 101 試行の平均値である. r_s の値が高いほど適応性能が高い手法であることを表す. また, 図中のエラーバーは標準偏差を表す.

一次関数に基づく oracle 関数 $o_{inc}(n_t)$, $o_{dec}(n_t)$ における実験結果

図 5.9 に, 式 (5.7), (5.8) に示す一次関数を oracle 関数として使用した実験結果を示す. ここで, oracle 関数 $o_{inc}(n_t)$ と $o_{dec}(n_t)$ の違いは, oracle パラメータが増加するか減少するかという点である. jDE, EPSDE, JADE は両関数においてほとんど r_s に変化は無い. 一方, MDE は, $o_{inc}(n_t)$ での結果と比べ, $o_{dec}(n_t)$ では $\beta \geq 0.3$ にて r_s が減少する傾向が見られる.

両関数において, β の値が 1.0 に近づくほど全ての手法にて単調に r_s も上昇している. $\beta = 0.1$ では全ての適応手法にほとんど差はみられない. 一方, β が高くなるにつれて, $o_{inc}(n_t)$ と $o_{dec}(n_t)$ の両関数にて jDE の性能が, 比較的劣るようになる. 反対に, EPSDE の r_s は次第に高くなる, つまり他手法と比べ優れた性能を示している.

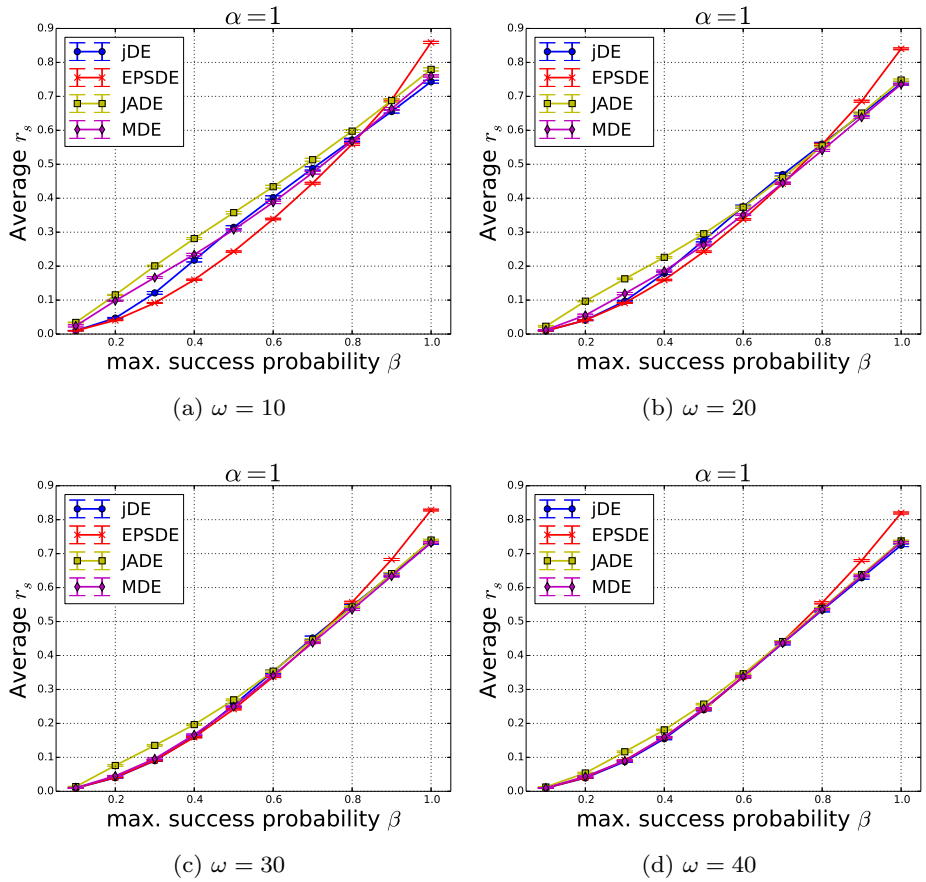
sin 関数に基づく oracle 関数 $o_{sin}(n_t)$ における実験結果

図 5.10 に, 式 (5.9) に示す sin 関数に基づく oracle 関数 $o_{sin}(n_t)$ にて $\omega = 10, 20, 30, 40$ とした場合の実験結果を示す. ここで, ω の値が増加していくほど, つまり図 5.10(a), (b), (c), (d) の順に, oracle パラメータに適応手法が追従していくことが困難となる.

全ての ω の値において, β の値が 1 の時は EPSDE が最も高い r_s を示している. しかし, β の値が低くなるほど, 他手法と比べ EPSDE は劣るようになる. 一方, JADE は β の値が低い場合においても, 比較的高い r_s を示す. ただし, JADE が他手法と比べて優位となる β の範囲は, ω の値が大きくなるにつれて狭まっていく傾向が見られる.

ランダムウォークに基づく oracle 関数 $o_{rw}(n_t)$ における実験結果

図 5.11 に, 式 (5.10) に示すランダムウォークに基づく oracle 関数 $o_{rw}(n_t)$ における実験結果を示す. ここで, 1 世代におけるランダムウォークのステップサイズの大きさを制御する s



(a) $\omega = 10$ (b) $\omega = 20$

(c) $\omega = 30$ (d) $\omega = 40$

図 5.10: 式 (5.9) に示す oracle 関数 $o_{sin}(n_t)$ を用いたシミュレーション実験の結果. 横軸は式 (5.5) における β の値, 縦軸は式 (5.6) の r_s の 101 試行の平均値である. r_s の値が高いほど適応性能が高い手法であることを表す. (a), (b), (c), (d) は, 式 (5.9) の角周波数 ω の値を 10, 20, 30, 40 と変更した場合の結果である. また, 図中のエラーバーは標準偏差を表す.

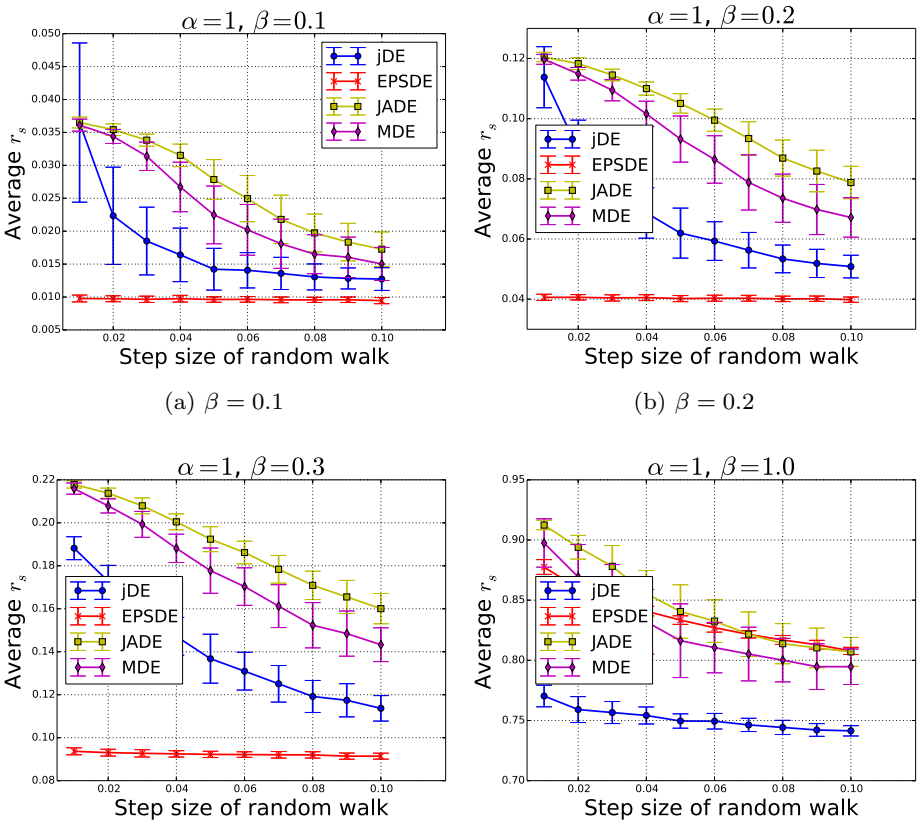
の値が増加していくほど, oracle パラメータに適応手法が追従していくことが困難となる.

図 5.11(d) から, $\beta = 1$ の場合は図 5.9 と図 5.10 の結果と同様に, EPSDE が比較的良好である. しかし, 図 5.11(a), (b), (c) に示すように $\beta = 0.1, 0.2, 0.3$ においては最も劣る性能を EPSDE は示している. 一方, s が増加するにつれ r_s が単調に減少していくものの, JADE は多くの β と s の設定の下, 最も優れた性能を示している. また, MDE と JADE については, β と s の値に依らず, MDE が劣っている.

5.5.3 考察

本節では 5.5.2 節にて得られた各 oracle 関数における実験結果, 及び BBOB benchmarks における適応手法の探索性能を比較した 5.3 節の結果について議論する.

5.3 節にて行った, 8 種類の突然変異戦略と 2 つの交叉手法の組み合わせにおける各適応手法の性能評価実験では, binomial 交叉では JADE, SEC では EPSDE と jDE が比較的優れて



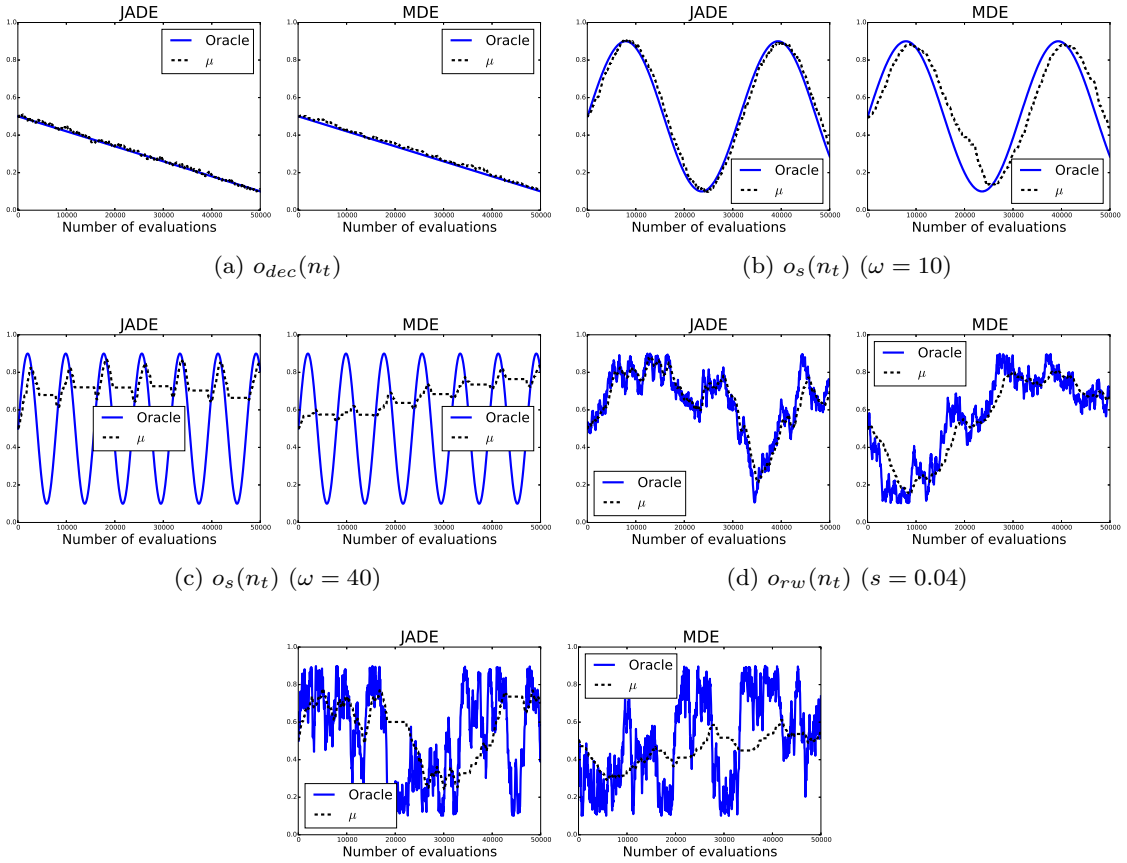
(a) $\beta = 0.1$　　(b) $\beta = 0.2$

(c) $\beta = 0.3$　　(d) $\beta = 1.0$

図 5.11: 式 (5.10) に示すランダムウォークに基づく oracle 関数 $o_{rw}(n_t)$ を用いたシミュレーション実験の結果．横軸は一世代におけるランダムウォークのステップサイズの大きさを制御する s の値, 縦軸は式 (5.6) の r_s の 101 試行の平均値である．r_s の値が高いほど適応性能が高い手法であることを表す．(a), (b), (c), (d) は, 式 (5.5) における β の値を 0.1, 0.2, 0.3, 1.0 と変更した場合の結果である．また, 図中のエラーバーは標準偏差を表す．

いた．「JADE」 [267] は突然変異戦略に current-to-$p$best/1, 交叉手法に binomial 交叉, 適応手法に「JADE の適応手法」を使用した複合体である．JADE の提案論文 [267] ではこの複合体である「JADE」の探索性能が優れていると結論づけているが, 正確には「JADE の適応手法」が binomial 交叉においては優れていたために, 「JADE」の良好な性能が実現されたと言える．しかし, JADE の適応手法が優れている理由を説明することは容易ではない．多くの先行研究は図 5.6 のように適応メタパラメータを図示し, 視覚に基づく適応手法の解析を試みていた．しかし, 5.4.1 節にて議論したとおり, このようなアプローチによりアルゴリズムの探索性能と適応手法の性能を結び付けて議論することは困難である．

そこで, 5.4 節では理想的なパラメータ適応の軌跡の代理である oracle パラメータを用いた新たなシミュレーション法を提案した．5.5 節にて得られた実験結果では, jDE, EPSDE, MDE の適応性能が乏しいシミュレーションモデルにおいて, JADE は優れた性能を示しており, 5.4.2 節にて定めた 「優れた適応手法」に比較手法の内では最も近いことが確認された．そ



(a) $o_{dec}(n_t)$ (b) $o_s(n_t)$ $(\omega = 10)$

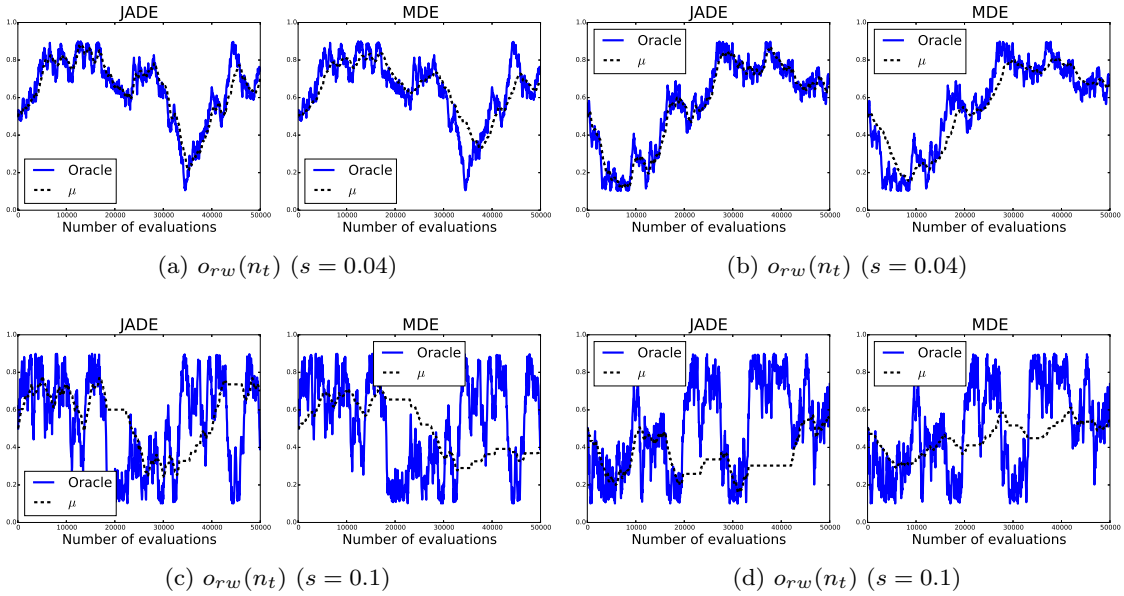
(c) $o_s(n_t)$ $(\omega = 40)$ (d) $o_{rw}(n_t)$ $(s = 0.04)$

(e) $o_{rw}(n_t)$ $(s = 0.1)$

図 5.12: 各 oracle 関数を用いたシミュレーション実験における, JADE と MDE の適応プロセスの比較 ($\beta = 0.1$). JADE と MDE における適応メタパラメータ μ の値を示している. 横軸は評価回数, 縦軸はパラメータの値を表す. 101 試行における, 式 (5.6) の r_s が中央値となる 1 試行のデータを示している. なお, oracle 関数 $o_{rw}(n_t)$ では 1 から 101 試行の各試行ごとに異なる oracle パラメータのインスタンスを使用しているため, 図中の oracle パラメータは JADE と MDE にて異なる. 同一のインスタンスにおける比較結果は図 5.13 を参照されたい.

のために, 5.3 節のベンチマークセットを用いた評価実験において, JADE は優れた探索性能を示していたと言える. 言い換えると, 5.3 節におけるベンチマークセットにおいて JADE が優れた探索性能を示していた理由は, 各探索状況における理想的なパラメータを近似する能力が優れていたためであると考えられる. しかし, 例えば図 5.1 に示す突然変異戦略に rand/1 を用いた実験では, JADE は他の適応手法と比べて劣る探索性能を示していたように, 以上の結論は万能ではない. そのため, oracle パラメータを用いたシミュレーション法における結果と実際のベンチマーク集合における実験結果がどの程度の相関を有するのか, より詳細な調査が今後の課題として残される.

5.5.2 節で示した実験結果から, oracle パラメータに追従することが容易な $\beta = 1.0$ という実験設定においては, EPSDE が優れていた. これは, $\beta = 1.0$ という非現実的に高い最大成功



(a) $o_{rw}(n_t)$ $(s = 0.04)$　　(b) $o_{rw}(n_t)$ $(s = 0.04)$

(c) $o_{rw}(n_t)$ $(s = 0.1)$　　(d) $o_{rw}(n_t)$ $(s = 0.1)$

図 5.13: ランダムウォークに基づく oracle 関数を用いたシミュレーション実験における, JADE と MDE の適応プロセスの比較 ($\beta = 0.1$). 横軸は評価回数, 縦軸はパラメータの値を示し, μ の値をプロットしている. (a), (c) については, JADE において, 101 試行における式 (5.6) の r_s が中央値となる 1 試行におけるインスタンスにおけるデータを示している. 一方, (b), (d) については, MDE において, 101 試行における式 (5.6) の r_s が中央値となる 1 試行におけるインスタンスにおけるデータを示している.

確率値と, 失敗しない限り同じパラメータを使い続けるという EPSDE の性質が合ったための結果だと考えられる. 反対に, $\beta = 0.1$ などの低い最大成功確率値においては, JADE が比較的優れていた. これは, 過去数世代において成功したパラメータ列の付近に新たなパラメータを生成する JADE は, β が低い場合においても oracle パラメータの近辺に重点的にパラメータを生成できたためであると考えられる. ここで, binomial 交叉により突然変異個体 $\boldsymbol{v}^{i,t}$ から子個体 $\boldsymbol{u}^{i,t}$ へと受け継がれる決定変数の数の期待値 $E(L)$ は, 図 3.3 に示すとおり線形であり, C の値が増加, 減少するに比例して $E(L)$ も増加, 減少する. 一方, SEC における $E(L)$ は, 対象問題の次元数が増加した場合, 値の範囲によっては $E(L)$ が C に対して鈍感 , 又は敏感となりえる. このことから, binomial 交叉において JADE が優れていたのは, C の値と $E(L)$ が比例関係にあり, 過去数世代において成功したパラメータ列の付近に新たなパラメータを生成する JADE に適していたためだと考えられる. 反対に, SEC において EPSDE が優れていたのは, $E(L)$ が C の狭い範囲において急激に変動するため, 失敗した場合はランダムにパラメータを再初期化し, 失敗しない限り同じパラメータを使い続けるという EPSDE の機能が適していたためだと考えられる.

MDE は JADE と類似した適応手法であるにも関わらず, 多くの場合 JADE よりも劣っていた. これについてさらなる議論をするために, 図 5.12 に $\beta = 0.1$ における各 oracle 関数における JADE と MDE の適応プロセスを示す. また, oracle 関数 $o_{rw}(n_t)$ での同一インスタンスにおける比較結果は, 図 5.13 に示している. 図 5.12(a), (b) から, 比較的緩やかに oracle パ

ラメータが変動するモデルにおいては, JADE の μ はほぼ oracle パラメータと重なっていることがわかる．また, 図 5.12(d) から, 世代 t ごとに不規則に oracle が変動するモデルにおいても, ある程度追従することができている．

一方, MDE においては世代の経過に対して oracle パラメータが緩やかに変化する図 5.12 (b), (d) においても, 適応メタパラメータ μ が oracle パラメータからやや離れている．この oracle パラメータを追従する能力の違いが原因となり, JADE と MDE の各モデルに対する優劣の変化が発生したのだと考えられる．また, 5.3 節にて行われた BBOB benchmarks における実験では, 多くの場合 MDE は JADE に劣っていたが, このパラメータ適応能力が性能差の要因であったと考えられる．

しかし, 図 5.12 (c), (e) のように oracle パラメータが世代 t ごとに急激に変化するモデルにおいては, JADE と MDE の両手法とも対応することがほとんどできていない．このことから, 既存の適応 DE の適応手法を改善する余地が残されていると言える．そのため次の 6 章では, この既存の適応手法の問題点を考慮した, DE のための新たな適応手法である Success-History based Adaptive DE (SHADE) を提案する．

5.6 おわりに

近年, 現探索状況の情報を基に使用するパラメータ設定を調整する適応的パラメータ制御に関する研究が, DE におけるパラメータ制御法の研究の主流となっている [43]．また, 4.5 節で述べたように, これまでにいくつかの優れた適応 DE が提案されている．しかし, 多くの優れた適応 DE が提案されている一方, その適応手法に関する知見は乏しい．そこで, 本章では近年の代表的な適応 DE である jDE [27], EPSDE [157], JADE [267], MDE [110] の適応手法の解析を行った．

第一に, 5.3 節では 8 種類の突然変異戦略と 2 種類の交叉手法の組み合わせの計 16 種類のオペレータにおける各適応手法の性能を, BBOB benchmarks [91, 93] を用いて評価した．実験結果から, 本来の枠組みにおいて使用されることが前提とされている突然変異戦略と交叉手法の組み合わせ以外のオペレータを用いた場合, 各文献にて報告されている性能と比べ劣る傾向が見られた．例えば, JADE では本来 binomial 交叉の使用を前提として設計された適応手法であるためか, 交叉手法を SEC に変えた途端に性能が大幅に劣化していた．一方, jDE は使用する突然変異戦略と交叉手法の組み合わせに依らず, 比較的良好な性能を示していた．しかし, 一般的に適応 DE は通常の DE (Algorithm 9) よりも探索性能が優れているとされているが, 対象問題の次元数, 及び用いる突然変異戦略と交叉手法の組み合わせによっては通常の DE に劣る傾向が確認された．

第二に, 5.5 節では理想的なパラメータ適応の軌跡の代理である oracle パラメータを用いた新たなシミュレーション法により適応手法を解析した．始めに伝統的な視覚に基づく適応手法の解析法の問題点を指摘し, 次に理想的なパラメータ適応の軌跡の代理である oracle パラメータを用いた新たなシミュレーション法を提案した．本シミュレーション法ではパラメータ値を独立して評価することができ, これまで難しかった適応手法の適応能力の解析を可能とす

る．つまり, 提案シミュレーション法では, ベンチマーク関数, 対象問題の次元数, 使用するオペレータの種類といった様々な要因とは独立に, 適応手法の性能評価がある程度可能である．実験結果から, 世代の経過に対して緩やかに変化する oracle パラメータにすら適応することが難しい適応 DE があることがわかった．また, この結果を基に, 5.3 節にて行われた BBOB benchmarks における実験結果における各手法間の性能差について議論した．

本章では生成したパラメータ θ と oracle パラメータ θ^o の距離に線形に比例して成功と判定される確率が減少する, 式 (5.5) のみをシミュレーションに使用した．最も単純な線形式を用いて一定の結果を得られたことにより, 提案シミュレーションモデルの妥当性はある程度実証されたと考えられるが, ガンマ分布や正規分布などの様々な確率分布を用いた同様の実験を行い, その差異を調査することが今後の課題として残される．

Algorithm 18 のパラメータ θ について, F, C の両方を使用した場合においても, それぞれ同じ oracle 関数を使用していた．しかし, 実際では F と C に求められる適応能力の傾向は異なると考えられる．そこで, F と C のそれぞれに対して異なる oracle 関数を用いた場合, 適応手法はどのような傾向を示すのかを調査することが今後の課題として考えられる．

通常の DE における F と C の理論的な解析研究は付録 B 章に示すように少なからず行われている．一方, 適応手法におけるパラメータ適応に対する理論的な研究は, 著者の知る限りこれまでにほとんど行われていない．これは 5.1.2 節にて述べたとおり, 適応手法を遺伝的オペレータとは独立して評価することは困難なためである．しかし, 適応手法を独立して評価可能な本シミュレーション法により, これまで困難であった適応手法の理論的な解析が可能となったと考えられる．適応手法の追従能力と oracle 関数の傾き (例えば, ランダムウォークを用いた oracle 関数 $o_{rw}(n_t)$ では, パラメータを平滑化した際の傾き) の関係性の解析などが, 今後の課題として残される．

特定の適応 DE の適応手法を実験的に解析するツールとして, 本研究にて提案した oracle パラメータを用いたシミュレーション法は今後の適応手法の設計に有用なアプローチであると言える．本章では F, C のパラメータ適応手法のみを扱ったが, 突然変異戦略の適応手法 [189, 157, 79, 62], 及びその他の制御パラメータの適応手法 [40, 261] を解析する際にも有用であると思われる．さらに, EA 全体においても特定の適応手法が優れた適応性能を示す理由を解明した研究は少ないため [119], 本シミュレーション法を用いることにより新たな知見が EA 全体においても得られることが期待できる．ACO [178] や AOS [63] を始めとする他の適応 EA への応用は, 今後の課題として考えられる．

第 6 章

Success-History based Adaptive Differential Evolution

5 章では, 現在の探索状況の情報を基に使用するパラメータ設定を調整する適応的パラメータ制御を用いた適応 DE の解析を行った. その結果, 既存の適応 DE の適応手法にはいくつかの問題点があり, 改善の余地が残されていることがわかった.

そこで, 本章では既存の適応手法の問題点を考慮した, DE のための新たな適応手法である Success-History based Adaptive DE (SHADE) を提案する. SHADE では探索中に得られた対象問題に適したパラメータ設定をメモリに保存し, このメモリ内の要素を用いて新たにパラメータを生成することで, 制御パラメータの自動調整を行う. 5 章にて使用した既存の適応 DE と比較することで, SHADE の有用性を示す. また, DE 以外の black-box optimization 環境における関数最適化問題に対する state-of-the-art な手法と比較する.

なお, 本章は著者が第一著者である先行研究 [227, 229] に基づいている.

6.1 はじめに

EA におけるパラメータ制御法は, (1) 決定的パラメータ制御 (4.4.1 節), (2) 適応的パラメータ制御 (4.4.2 節), (3) 自己適応的パラメータ制御 (4.4.3 節) に大別される [54] (4 章参照). その中でも, DE においては (2) 適応的パラメータ制御に関する研究が近年増加しており, パラメータ制御法の主流となっている [43]. また, 4.5 節で述べたように, これまでにいくつかの優れた適応 DE が提案されている. しかし, 多くの優れた適応 DE が提案されている一方, その適応手法に関する知見は乏しかった. これに対して, 5 章では適応 DE の解析を行った. その結果, 既存の適応 DE の適応手法にはいくつかの問題点があり, 改善の余地が残されていることがわかった.

第一に, これまで提案されてきたほぼ全ての適応 DE の適応手法は, 特定の突然変異戦略, 及び交叉手法の組合せを使用することを前提として設計されている. そのため, そのオペレータの組合せ以外を使用した場合, 探索性能の低下が手法によっては見られた. 例えば, JADE では本来 binomial 交叉の使用を前提として設計された適応手法であるためか, 交叉手法を SEC

(後述の 8 章参照) に変えた途端に性能が大幅に劣化していた．また, 一般的に適応 DE は通常の DE (Algorithm 9) よりも探索性能が優れているとされているが [27, 157, 267, 110], 対象問題の次元数, 及び用いる突然変異戦略と交叉手法の組み合わせによっては通常の DE に劣る傾向が見られた．

第二に, 理想的なパラメータ適応の軌跡の代理である oracle パラメータを用いたシミュレーション法において, 世代の経過に対して緩やかに変化する oracle パラメータにすら適応することが難しい適応 DE が存在することがわかった．例えば, 図 5.12(b) に示すように, MDE は世代の経過に対して緩やかな正弦波状に変化する oracle パラメータにおいてすら, 追従することに失敗している．また, 図 5.12 (c), (e) のように oracle パラメータが世代 t ごとに急激に変化するモデルにおいては, 5.3 節の実験にて binomial 交叉を用いた場合比較手法の中では良好な探索性能を示した JADE においても, 適応することがほとんどできていない．

上記の考察から, 既存の適応 DE の適応手法は改善の余地が残されていると言える．そこで, 本章では既存の適応手法の問題点を考慮した, DE のための新たな適応手法である Success-History based Adaptive DE (SHADE) を提案する．SHADE では探索中に得られた対象問題に適したパラメータ設定をメモリに保存し, このメモリ内の要素を用いて新たにパラメータを生成することで, 制御パラメータの自動調整を行う．5 章にて使用した既存の適応 DE と比較することで, SHADE の有用性を示す．また, DE 以外の black-box optimization 環境における関数最適化問題に対する state-of-the-art な手法と比較する．

本章の構成は次の通りである．始めに 6.2 節にて提案手法である SHADE について説明する．次に, 6.3 節にて oracle パラメータを用いたシミュレーション法における SHADE の適応性能を評価する．その後, 6.4 節にて BBOB benchmarks にて従来の適応 DE との比較を行う．6.5 節ではメモリサイズ H の設定の違いが SHADE の探索性能に与える影響を調査する．また, 6.6 節では SHADE に集団数 N にパラメータ制御法の一手法である決定的集団数減少法を導入した, L-SHADE を提案する．Black-box optimization 環境における関数最適化問題の state-of-the-art な探索手法との比較を 6.7 節にて行い, 最後に 6.8 節にて本章をまとめる．

6.2　Success-History based Adaptive DE (SHADE)

本節では, 本章の提案手法である SHADE の枠組みについて述べる．始めに 6.2.1 節にて近年の代表的な適応 DE である JADE や MDE などの適応メタパラメータ μ_F, μ_C を使用する適応手法の問題点について述べる．次に, 6.2.2 節にて SHADE を説明する．最後に, 6.2.3 節にて既存の適応手法との関連性, 及び SHADE の利点について述べる．

6.2.1　JADE, MDE などの適応メタパラメータ μ_F, μ_C を使用する適応手法の問題点

5.3 節にて行った, 様々なオペレータを用いた各適応手法の BBOB benchmarks における性能評価実験では, binomial 交叉を用いた場合, JADE は比較的良好な探索性能を示していた．

また, 5.5 節にて行った oracle パラメータに基づくシミュレーション実験においても, JADE は比較手法の中では良好な適応性能を示していた. しかし, そのような JADE の適応手法であっても, 6.1 節にて述べたとおり, 十分な適応能力を有するとは言い難い. 本節では JADE, 及び MDE などの適応メタパラメータ μ_F, μ_C を使用する適応手法の問題点について述べる.

JADE, 及び MDE では F と C の自動調整のため, それぞれ適応メタパラメータ μ_F, μ_C を使用する (4.4.2 節参照). 各世代 t の始めに, 個体 $\boldsymbol{x}^{i,t}$ ごとに $F_{i,t}$ と $C_{i,t}$ をそれぞれ μ_F を位置パラメータとするコーシー分布, μ_C を平均とする正規分布に従う乱数にて割り当てる. そして, 世代 t の終了時に成功したパラメータ集合 $\boldsymbol{S}^F, \boldsymbol{S}^C$ に基づき, μ_F, μ_C を更新する.

しかし, DE の確率的な性質から, 「理想的な」パラメータとは大きく異なる F, C でも親個体よりも優れた子個体を生成し $\boldsymbol{S}^F, \boldsymbol{S}^C$ にそのようなパラメータが含まれる可能性は十分にある. このような場合に式 (4.6), (4.7) により μ_F, μ_C の更新を行うと, 更新後の μ_F, μ_C は対象問題, 及び現在の探索状況に適したパラメータ設定とは異なる値へと変化してしまう可能性がある. この好ましくない現象は, JADE の学習率 $c \in [0, 1]$ を十分に小さく設定することで, 防ぐことができる. しかし, c を小さく設定し過ぎた場合, 世代ごとの $\boldsymbol{S}^F, \boldsymbol{S}^C$ の情報を利用することがほとんどできず, μ_F, μ_C が初期値からほとんど変動しない, つまりパラメータ適応手法がうまく機能しない恐れがある. JADE, 及び MDE は, 学習率 c の設定に関するこのようなジレンマを有する.

6.2.2 SHADE

6.2.1 節での議論を元に設計した, 本章の提案手法である SHADE の枠組みについて本節では述べる. なお, 4.4 節で説明したような, 適応 DE を含むパラメータ制御機能を付加した DE アルゴリズムは, 多くの場合特定の突然変異戦略と交叉手法の組合せを使用することを前提としている. それに対して本節で述べる SHADE は, 任意の突然変異戦略と交叉手法の組合せを使用できるよう, 柔軟な枠組みとなるよう設計した.

SHADE は, F と C についてそれぞれ大きさ H のメモリ $\boldsymbol{M}^F, \boldsymbol{M}^C$ を用いてパラメータ適応を行う. ここで, $\boldsymbol{M}^F = (M_1^F, ..., M_H^F)$, $\boldsymbol{M}^C = (M_1^C, ..., M_H^C)$ であり, 全ての要素を 0.5 に初期化, つまり $j \in \{1, ..., H\}$ について $M_j^F \leftarrow 0.5, M_j^C \leftarrow 0.5$ とする (Algorithm 19, 2 行). なお, $M_j^F \in (0, 1]$, 及び $M_j^C \in [0, 1]$ である.

各世代 t において, 集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の個体 $\boldsymbol{x}^{i,t}$ ごとに, まず $\{1, ..., H\}$ の範囲から一様ランダムにメモリ番号 r_i を選択する (Algorithm 19, 7 行). そして, r_i 番目のメモリの要素 $M_{r_i}^F, M_{r_i}^C$ を用いて, 各個体 $\boldsymbol{x}^{i,t}$ の $F_{i,t}$ と $C_{i,t}$ を生成する (Algorithm 19, 8, 9 行):

$$F_{i,t} = \mathrm{randc}(M_{r_i}^F, 0.1) \tag{6.1}$$

$$C_{i,t} = \mathrm{randn}(M_{r_i}^C, 0.1) \tag{6.2}$$

ここで, $\mathrm{randc}(\mu, \sigma)$ は位置パラメータ μ と尺度パラメータ σ のコーシー分布に従う乱数, $\mathrm{randn}(\mu, \sigma)$ は平均 μ, 標準偏差 σ の正規分布に従う乱数である. $F_{i,t}$ の値が $F_{i,t} > 1$ の場合は $F_{i,t} = 1$ とし, $F_{i,t} \leq 0$ の場合は再び式 (4.6) を用いて生成を行う. $C_{i,t}$ の値が $[0, 1]$ 区間

外の場合は, 超えた方の境界値で置き換えられる. F, C の生成にコーシー分布, 正規分布を使用する理由, 及び生成した $F_{i,t}, C_{i,t}$ の修正方法などは, 4.5.3 節にて説明した JADE と同様である.

そして, 各個体 $\boldsymbol{x}^{i,t}$ は割り当てられた $F_{i,t}$ と $C_{i,t}$ を使用し, 表 3.1, 及び Algorithm 7, 22 に示す任意の突然変異戦略, 及び交叉手法により子個体 $\boldsymbol{u}^{i,t}$ を生成する (Algorithm 19, 10, 11 行). 集団中の全ての個体が子個体を生成した後に世代交代を行うが, この際成功した場合のみ, つまり $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ となった場合のみ, $F_{i,t} \rightarrow \boldsymbol{S}^F$, $C_{i,t} \rightarrow \boldsymbol{S}^C$ とする (Algorithm 19, 14 行目).

多くの適応 DE では, 生存選択において親個体 $\boldsymbol{x}^{i,t}$ よりも優れた子個体 $\boldsymbol{u}^{i,t}$ を作成できた場合を「成功」と呼ぶ. そして, 成功したのは使用した $F_{i,t}$ と $C_{i,t}$ のパラメータ設定が, 対象問題, 及び現探索状況に適していたためと仮定し, これらを何らかの方法で適応手法にフィードバックする. SHADE においては, 各世代 t の終了時に成功した $F_{i,t}$, $C_{i,t}$ の集合 $\boldsymbol{S}^F, \boldsymbol{S}^C$ の階数 2 のレーマー平均 $\mathrm{mean}_L(\boldsymbol{S}^F)$, $\mathrm{mean}_L(\boldsymbol{S}^C)$ を用いて, メモリ $\boldsymbol{M}^F, \boldsymbol{M}^C$ の $k \in \{1, ..., H\}$ 番目の要素 M_k^F, M_k^C を更新する (Algorithm 19, 18, 19 行)[*1]:

$$M_k^F \leftarrow \mathrm{mean}_L(\boldsymbol{S}^F) \tag{6.3}$$

$$M_k^C \leftarrow \mathrm{mean}_L(\boldsymbol{S}^C) \tag{6.4}$$

ここで, k は更新するメモリの要素を決定するパラメータである. 探索開始時 $t = 1$ では $k = 1$ であるが, 以後更新を行う度に 1 づつ増加し, $k > H$ となった場合は再度 $k = 1$ とする (Algorithm 19, 20 行).

以上の (1) メモリ $\boldsymbol{M}^F, \boldsymbol{M}^C$ を用いた $F_{i,t}$, $C_{i,t}$ の生成, (2) 個体 $\boldsymbol{x}^{i,t}$ ごとに割り当てられた $F_{i,t}$, $C_{i,t}$ を使用して子個体 $\boldsymbol{u}^{i,t}$ を生成, (3) 成功したパラメータ集合 $\boldsymbol{S}^F, \boldsymbol{S}^C$ を元にメモリ $\boldsymbol{M}^F, \boldsymbol{M}^C$ の一要素を更新といった手順を, 終了条件を満たすまで繰り返す. これにより, メモリ $\boldsymbol{M}^F, \boldsymbol{M}^C$ には対象問題, 及び現在の探索状況に適した, かつ多様な F, C のパラメータ設定が含まれていくことが期待される.

6.2.3　既存の適応 DE と比べた SHADE の新規性や利点について

本節では既存の適応 DE との関連性, 及び SHADE の利点について述べる. SHADE では各世代 t の始めに, 個体 $\boldsymbol{x}^{i,t}$ ごとにランダムに選択したメモリ $\boldsymbol{M}^F = (M_1^F, ..., M_H^F)$, $\boldsymbol{M}^C = (M_1^C, ..., M_H^C)$ の要素を位置パラメータ, 及び平均とするコーシー分布, 及び正規分布に従う乱数により $F_{i,t}$ と $C_{i,t}$ を生成する. そして, 世代 t の終了時に $\boldsymbol{S}^F, \boldsymbol{S}^C$ に基づき, 式 (6.3), (6.4) によりメモリ $\boldsymbol{M}^F, \boldsymbol{M}^C$ の一要素を更新する.

成功したパラメータ集合 $\boldsymbol{S}^F, \boldsymbol{S}^C$ を利用するという点では JADE に類似しているが, SHADE ではメモリ $\boldsymbol{M}^F, \boldsymbol{M}^C$ の一要素を更新するのみである. そのため, 先に述べたような

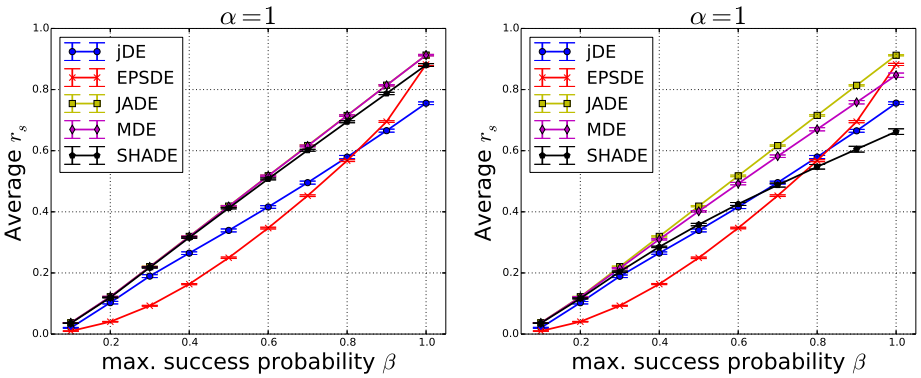
[*1] $C \in [0, 1]$ であるため, $\boldsymbol{S}^C$ の全ての要素が 0 であり, 式 (6.4) のレーマー平均の計算時にゼロ除算が発生する可能性がある. これを防ぐため, プログラム上の実装では $\forall S_j^C = 0 | S_j^C \in \boldsymbol{S}^C$ の場合は, $M_k^C \leftarrow 0$ としている.

Algorithm 19: SHADE

$t \leftarrow 1$, 集団 $\boldsymbol{P}^t = \{\boldsymbol{x}^{1,t}, ..., \boldsymbol{x}^{N,t}\}$ の初期化;
各メモリ $\boldsymbol{M}^F = (M_1^F, ..., M_H^F)$, $\boldsymbol{M}^C = (M_1^C, ..., M_H^C)$ の全ての要素を 0.5 に初期化;
$k \leftarrow 1$;
while 探索終了条件を満たしていない **do**
  $\boldsymbol{S}^F \leftarrow \emptyset$, $\boldsymbol{S}^C \leftarrow \emptyset$;
  for $i = 1$ **to** N **do**
    $\{1, ..., H\}$ からランダムに r_i を選択;
    $F_{i,t} = \mathrm{randc}(M_{r_i}^F, 0.1)$;
    $C_{i,t} = \mathrm{randn}(M_{r_i}^C, 0.1)$;
    表 3.1 の任意の突然変異戦略により変異個体 $\boldsymbol{v}^{i,t}$ を生成;
    $\boldsymbol{x}^{i,t}$ と $\boldsymbol{v}^{i,t}$ に binomial 交叉 (Algorithm 7), 又は SEC (Algorithm 22) のいずれかの交叉手法を適用し, 子個体 $\boldsymbol{u}^{i,t}$ を生成;
  for $i = 1$ **to** N **do**
    if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ **then**
      $\boldsymbol{x}^{i,t+1} = \boldsymbol{u}^{i,t}$, $F_{i,t} \rightarrow \boldsymbol{S}^F$, $C_{i,t} \rightarrow \boldsymbol{S}^C$;
    else
      $\boldsymbol{x}^{i,t+1} = \boldsymbol{x}^{i,t}$;
  if $\boldsymbol{S}^F, \boldsymbol{S}^C \neq \emptyset$ **then**
    $M_k^F \leftarrow \mathrm{mean}_L(\boldsymbol{S}^F)$;
    $M_k^C \leftarrow \mathrm{mean}_L(\boldsymbol{S}^C)$;
    $k \leftarrow (k \bmod H) + 1$;
  $t \leftarrow t + 1$;

理想的なパラメータから外れたパラメータ設定が $\boldsymbol{S}^F, \boldsymbol{S}^C$ に含まれていた場合においても, 唯一の適応パラメータである μ_F, μ_C を $\boldsymbol{S}^F, \boldsymbol{S}^C$ に近づけるように更新する JADE とは異なり, SHADE では H 個の要素から成るメモリ $\boldsymbol{M}^F, \boldsymbol{M}^C$ の一要素のみが影響を受けるだけである. 仮に $\boldsymbol{M}^F, \boldsymbol{M}^C$ の一要素が理想的なパラメータから外れていた場合でも, 式 (6.1), (6.2) に示すように個体 $\boldsymbol{x}^{i,t}$ ごとに一様ランダムに選択したメモリの要素 $M_{r_i}^F, M_{r_i}^C$ を元に $F_{i,t}, C_{i,t}$ を生成するため, その要素が選択される確率は $1/H$ であり, H が大きくなるほどその影響は小さいと考えられる.

また, 世代 t において成功したパラメータ集合 $\boldsymbol{S}^F, \boldsymbol{S}^C$ のレーマー平均値を直接 $\boldsymbol{M}^F, \boldsymbol{M}^C$ の要素に代入する SHADE では, $\boldsymbol{S}^F, \boldsymbol{S}^C$ の情報, つまり現探索状況に適していると思われるパラメータ設定を次世代 $t+1$ において即座に利用することができる. 以上の理由から, 6.2.1 節にて述べた JADE, 及び MDE などの適応メタパラメータ μ_F, μ_C を使用する適応手法が抱える問題点を SHADE は解消していると考えられる. 実際に, 次の 6.3 節にて述べる oracle パラメータを用いたシミュレーション法による解析にて, SHADE は JADE や MDE などと比べ適応手法として優れていることを実証する.



(a) oracle 関数 $o_{inc}(n_t)$　　(b) oracle 関数 $o_{dec}(n_t)$

図 6.1: 式 (5.7), 及び (5.8) に示す oracle 関数 $o_{inc}(n_t)$, 及び $o_{dec}(n_t)$ を用いたシミュレーションにおける SHADE と他の適応 DE との比較結果．横軸は式 (5.5) における β の値, 縦軸は式 (5.6) の r_s の 101 試行の平均値である．また, 図中のエラーバーは標準偏差を表す．

6.3　Oracle パラメータを用いたシミュレーションにおける適応性能の評価

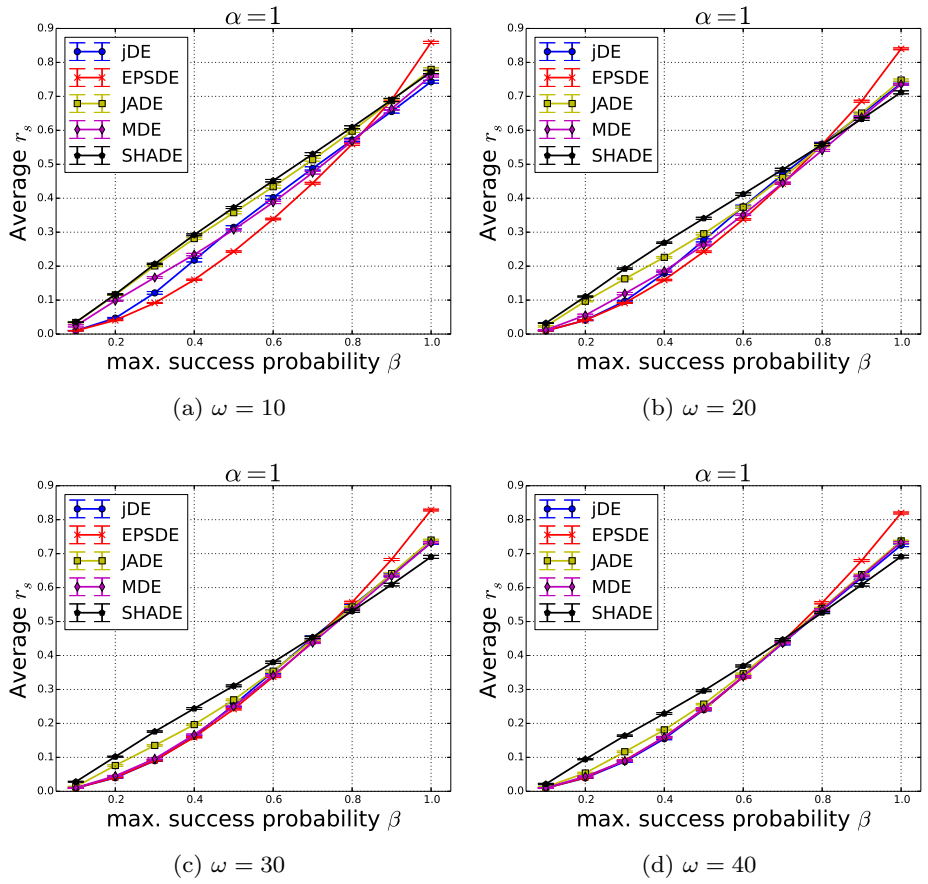
本節では, 5.4.3 節にて述べた oracle パラメータを用いたシミュレーション法により, SHADE のパラメータ適応能力を解析する．始めに 6.3.1 節にて (1) 一次関数, (2) sin 関数, (3) ランダムウォークの 3 種類の異なる oracle 関数における結果を報告し, 次に 6.3.2 節にて考察を行う．

6.3.1　各 oracle 関数における実験結果

以下では, 3 種類の異なる oracle 関数 (5.5.1 節参照) を用いたシミュレーションにおける SHADE と 5 章で解析対象とした 4 つの適応 DE (jDE [27], EPSDE [157], JADE [267], MDE [110]) の比較結果について述べる．

SHADE のメモリサイズ H については, JADE の学習率 $c = 0.1$ の設定を参考に $H = 10$ とした[*2]．後に説明する 6.5 節にて, $H = \{1, 3, 5, 7, 10, 15, 20, 40, 80, 160\}$ として BBOB benchmarks にて性能評価実験をしたところ, SHADE の探索性能, 及び適応性能は H の設定に依存することがわかった．また, 多くの場合 $H \in \{5, ..., 20\}$ が適切な設定範囲であり, $H = 10$ という設定を使用しても問題は無いことがわかった．なお, その他の全ての実験設定は 5.5.1 節にて述べた設定と同様にした．

[*2] JADE の μ_F, μ_C の更新式 (4.8), (4.9) において, ある世代における成功したパラメータ集合 $\boldsymbol{S}^F, \boldsymbol{S}^C$ の影響は, 約 $1/c$ 世代後に消失する [267]．そのため, $c = 0.1$ とした場合, $1/0.1 = 10$ と約 10 世代で $\boldsymbol{S}^F, \boldsymbol{S}^C$ の影響がほぼ無くなる．一方, SHADE のメモリ更新式 (6.3), (6.4) では多くの場合 H 世代後に $\boldsymbol{S}^F, \boldsymbol{S}^C$ の影響が消失する．以上のことを考慮し, $\boldsymbol{S}^F, \boldsymbol{S}^C$ の有効期間が JADE とほぼ同等になるように, $H = 10$ とした．



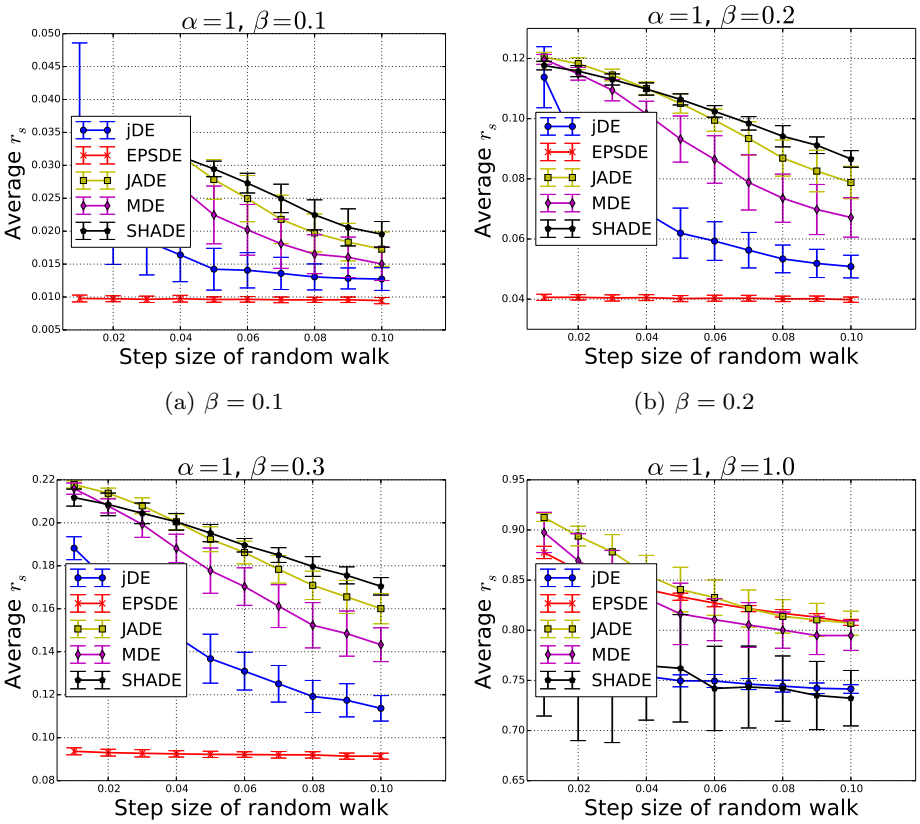
(a) $\omega = 10$ (b) $\omega = 20$

(c) $\omega = 30$ (d) $\omega = 40$

図 6.2: 式 (5.9) に示す oracle 関数 $o_{sin}(n_t)$ を用いたシミュレーションにおける SHADE と他の適応 DE との比較結果．横軸は式 (5.5) における β の値，縦軸は式 (5.6) の r_s の 101 試行の平均値である．(a), (b), (c), (d) は，式 (5.9) の角周波数 ω の値を 10, 20, 30, 40 と変更した場合の結果である．また，図中のエラーバーは標準偏差を表す．

式 (5.7), (5.8) の一次関数，式 (5.9) の sin 関数，式 (5.10) のランダムウォークを用いた oracle 関数における実験結果を，図 6.1, 6.2, 6.3 にそれぞれ示す．図 6.1 から，一次関数に基づく oracle 関数 $o_{inc}(n_t)$ と $o_{dec}(n_t)$ の両者において SHADE の r_s は JADE，及び MDE に劣っていることがわかる．ここで，oracle 関数 $o_{inc}(n_t)$ と $o_{dec}(n_t)$ の違いは，oracle パラメータが増加するか減少するかという点である．さらに，$o_{dec}(n_t)$ における結果を示す図 6.1(b) において，$\beta \in [0.8, 1.0]$ では比較手法中最も劣る性能を SHADE は示している．図 6.1(a), (b) において SHADE の性能が異なるのは，集合の高い値に偏りやすいレーマー平均をメモリ $\boldsymbol{M}$ の更新に使用しているためだと考えられる．

図 6.2 は sin 関数に基づく oracle 関数における結果を示している．ここで，ω の値が増加していくほど，つまり図 6.2(a), (b), (c), (d) の順に，oracle パラメータに適応手法が追従していくことが困難となる．SHADE は $\omega > 20$ において，$\beta = 1$ の場合に最も劣る．しかし，$\beta \in [0.1, 0.6]$ では，SHADE は他の手法よりも優れた性能を示している．この傾向は，ω の値



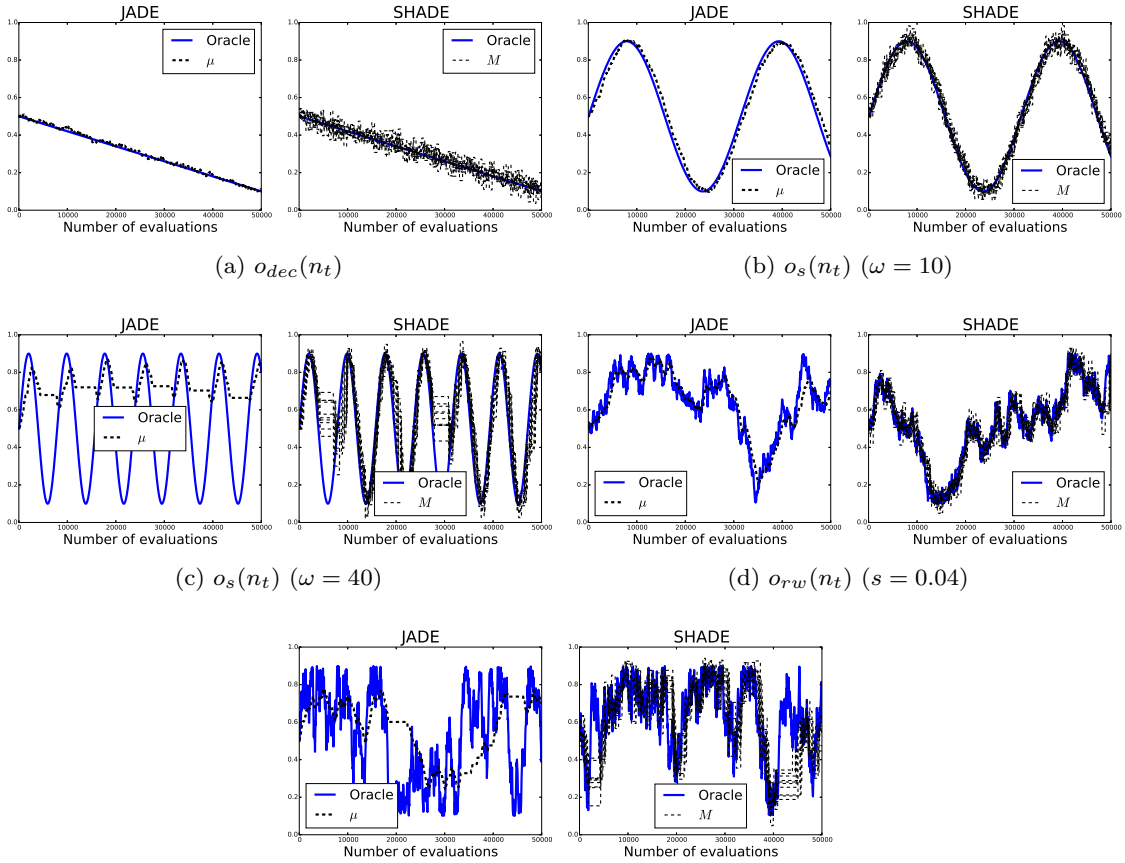
(a) $\beta = 0.1$　(b) $\beta = 0.2$

(c) $\beta = 0.3$　(d) $\beta = 1.0$

図 6.3: 式 (5.10) に示すランダムウォークに基づく oracle 関数 $o_{rw}(n_t)$ を用いたシミュレーションにおける SHADE と他の適応 DE との比較結果．横軸は一世代におけるランダムウォークのステップサイズの大きさを制御する s の値, 縦軸は式 (5.6) の r_s の 101 試行の平均値である．(a), (b), (c), (d) は, 式 (5.5) における β の値を 0.1, 0.2, 0.3, 1.0 と変更した場合の結果である．また, 図中のエラーバーは標準偏差を表す．

が増加するほど, つまり oracle パラメータが短い期間で急激に変化するようになるほど, 強く見られるようになる．

最後に, 図 6.3 は式 (5.10) に示すランダムウォークに基づく oracle 関数 $o_{rw}(n_t)$ における結果を示している．1 世代におけるランダムウォークのステップサイズの大きさを制御する s の値が増加していくほど, oracle パラメータに適応手法が追従していくことが困難となる．SHADE は $\beta = 1$ では jDE と同様の性能を示すものの, $\beta = 0.1, 0.2, 0.3$ では jDE, EPSDE よりも全ての s の値において優れている．JADE と SHADE の比較では, ランダムウォークのステップサイズ s が小さい場合, つまり oracle パラメータが緩やかに推移する場合は JADE の方が高い成功率を示す．反対に s が 0.05 付近より高い場合, つまり oracle パラメータが世代ごとに急激に変動する場合は, JADE は SHADE と比べ劣るようになる．また, SHADE と MDE については, β と s の値に依らずに MDE が劣っている．



(a) $o_{dec}(n_t)$ (b) $o_s(n_t)$ $(\omega = 10)$

(c) $o_s(n_t)$ $(\omega = 40)$ (d) $o_{rw}(n_t)$ $(s = 0.04)$

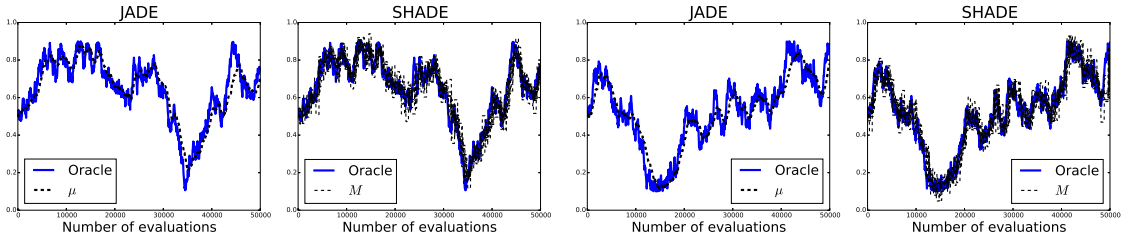
(e) $o_{rw}(n_t)$ $(s = 0.1)$

図 6.4: 各 oracle 関数を用いたシミュレーション実験における, SHADE と JADE の適応プロセスの比較 ($\beta = 0.1$). 横軸は評価回数, 縦軸はパラメータの値を示し, JADE では μ, SHADE ではメモリ $\boldsymbol{M}$ の全ての要素をプロットしている. 101 試行における, 式 (5.6) の r_s が中央値となる 1 試行のデータを示している. なお, oracle 関数 $o_{rw}(n_t)$ では 1 から 101 試行の各試行ごとに異なる oracle パラメータのインスタンスを使用しているため, 図中の oracle パラメータは SHADE と JADE にて異なる. 同一のインスタンスにおける比較結果は図 6.5 を参照されたい.

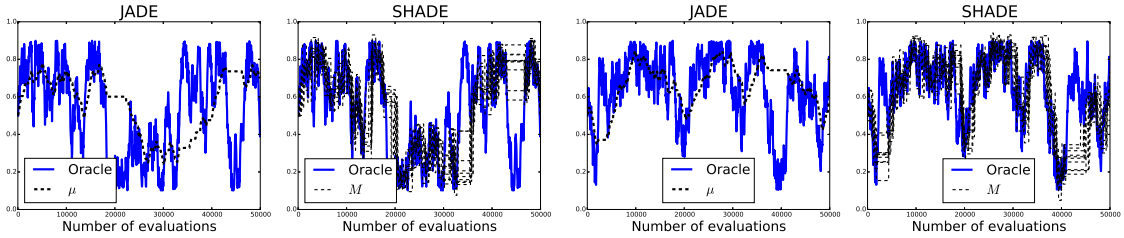
6.3.2 考察

本節では, 6.3 節にて述べた各 oracle 関数における実験結果について議論する. 実験結果から, $\beta = 0.1$ などの低い最大成功確率値においては, SHADE と JADE が比較的優れていた. この 2 手法において, $\beta = 0.1$ においても JADE は図 6.1, 図 6.2(a), 図 6.3(a) の $s \in [0.01, 0.03]$ などの oracle パラメータの形状が単純なモデルでは SHADE よりも高い r_s を示していた. 一方, SHADE はそれ以外の oracle パラメータの追従が困難なモデルにおいて, JADE よりも良好であった.

これについてさらなる議論をするために, 図 6.4 に $\beta = 0.1$ における各 oracle 関数における



(a) $o_{rw}(n_t)$ $(s = 0.04)$　(b) $o_{rw}(n_t)$ $(s = 0.04)$



(c) $o_{rw}(n_t)$ $(s = 0.1)$　(d) $o_{rw}(n_t)$ $(s = 0.1)$

図 6.5: ランダムウォークに基づく oracle 関数を用いたシミュレーション実験における, SHADE と JADE の適応プロセスの比較 ($\beta = 0.1$). 横軸は評価回数, 縦軸はパラメータの値を示し, JADE では μ, SHADE ではメモリ $\boldsymbol{M}$ の全ての要素をプロットしている. (a), (c) については, JADE において, 101 試行における式 (5.6) の r_s が中央値となる 1 試行におけるインスタンスにおけるデータを示している. 一方, (b), (d) については, SHADE において, 101 試行における式 (5.6) の r_s が中央値となる 1 試行におけるインスタンスにおけるデータを示している.

JADE と SHADE の適応プロセスを示す. また, oracle 関数 $o_{rw}(n_t)$ での同一インスタンスにおける比較結果は, 図 6.5 に示している. 図 6.4(a), (b) から, 比較的緩やかに oracle パラメータが変動するモデルにおいては, JADE の μ はほぼ oracle パラメータと重なっていることがわかる. 一方, SHADE のメモリ $\boldsymbol{M}$ の要素は oracle パラメータの周囲を覆うように幅広い値を保持しながら追従している. この違いが, 図 6.1 に示したように, 単純なモデルにおいては JADE が SHADE と比べて優れていた理由として考えられる. 反対に, 図 5.12(c), (e) のように oracle パラメータの変動が急なモデルでは, JADE の μ は明らかに適応に失敗している. 一方, SHADE は概ね oracle パラメータに適応することができている. つまり, JADE に対して SHADE は保持すべき適応メタパラメータにある程度の幅を持たせることにより, 急激な環境の変化にも対応できるロバストさを獲得していると言える.

6.4　BBOB benchmarks における SHADE の性能評価実験

6.3 節では, 5.4.3 節にて述べた oracle パラメータを用いたシミュレーション法による, SHADE のパラメータ適応能力の評価を行った. 実験結果から, 既存の適応 DE の適応手法が追従することが困難である複雑な oracle パラメータの形状においても, SHADE はある程度適応可能であり良好な性能を示すことがわかった. そのため, もし 5.4 節にて述べた, 「理想

のパラメータ適応の軌跡の代理である oracle パラメータ集合 $\boldsymbol{\theta}^o$ を沿うように機能できる適応手法は, 5.4.2 節にて定めた優れた適応手法である」という仮説が正しければ, ベンチマーク集合における性能評価実験においても SHADE は優れていることが期待できる. そこで, 本節では様々な突然変異戦略と交叉手法の組み合わせにおける SHADE の性能を, 5.3 節と同様に BBOB benchmarks (2.5.3 節, 又は付録 C 節参照) にて評価することにした.

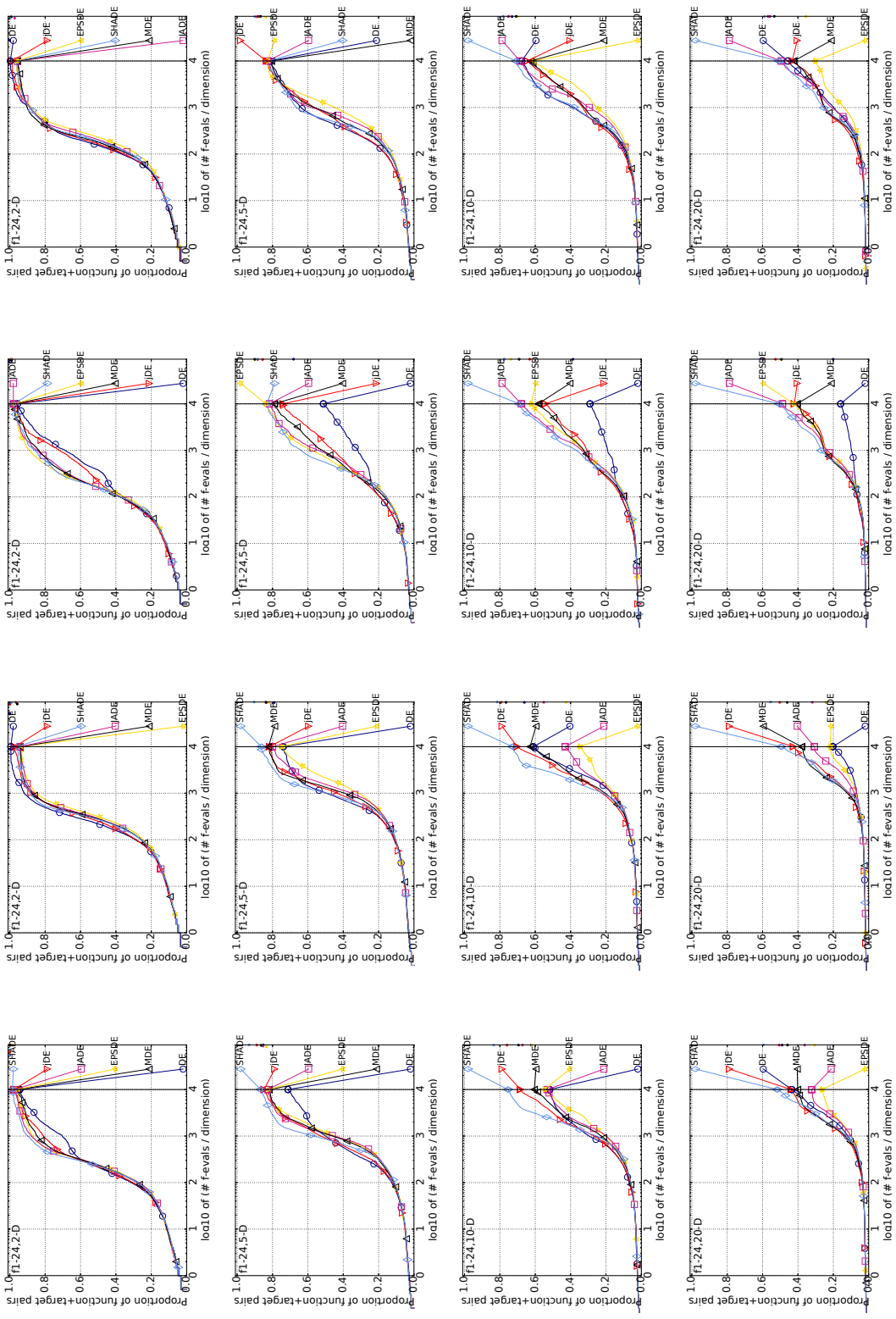
始めに 6.4.1 節にて通常の DE (Algorithm 9), 及び 4 つの適応 DE (jDE, EPSDE, JADE, MDE) と SHADE の比較結果を述べる. 次に, 6.4.2 節にて得られた結果について考察する.

6.4.1 実験結果

以下では, SHADE の性能評価実験の結果について述べる. なお, 本実験では, 5.4 節と同様に SHADE のメモリサイズ H を $H = 10$ とした. その他の設定は, 全て 5.3 節での実験設定と同様とした.

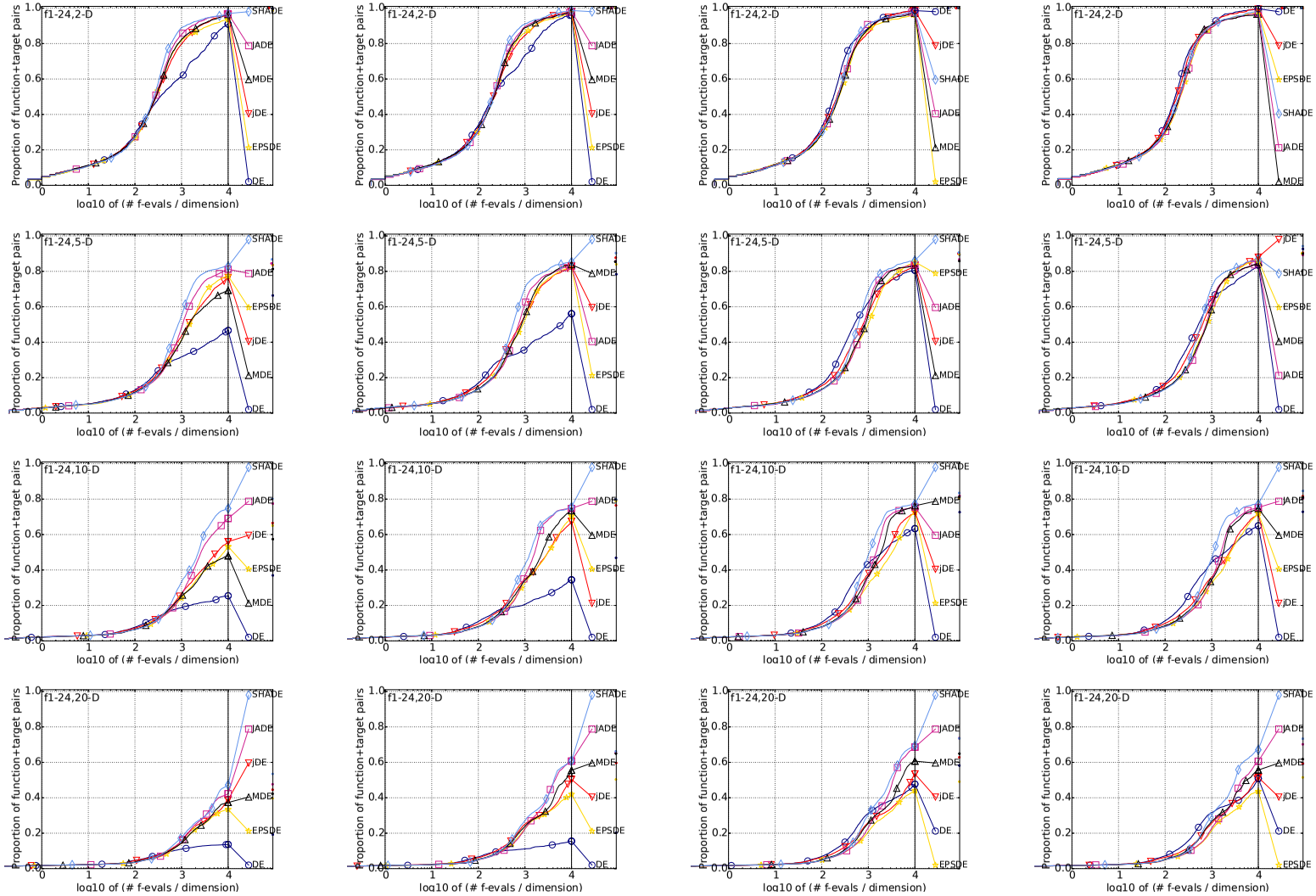
図 6.6 ～ 6.9 に, 突然変異戦略に rand/1, rand/2, best/1, best/2, current-to-rand/1, current-to-best/1, current-to-$p$best/1, 及び rand-to-$p$best/1 (表 3.1 参照), 交叉手法に binomial 交叉 (Algorithm 7) と SEC (Algorithm 22) を用いた場合における, 2, 5, 10, 20 次元の BBOB benchmarks の 24 個のベンチマーク関数 $f_1 \sim f_{24}$ における実験結果を示す. また, 8 つの突然変異戦略における最大評価回数 $10^4 \times D$ での時点での平均順位を, 交叉手法ごとに図 6.10 に示す.

図 6.6 ～ 6.9 から, 例えば突然変異戦略に best/1, 交叉手法に binomial 交叉を用いた $D = 2, 5$ における結果 (図 6.6(c)) のように特定の次元数, 及び特定の交叉手法と突然変異戦略の組み合わせを使用した場合は他手法に劣る場合があるものの, 多くの場合において SHADE は比較的良好な性能を示している. このことは, 図 6.10 において, 全ての次元数, 及び交叉手法の下で比較手法中最良の平均順位を示していることからも確認できる. 特に, binomial 交叉を用いた場合, $D = 10, 20$ にて 8 つの全ての突然変異戦略を用いた場合において最も良い性能を示している.



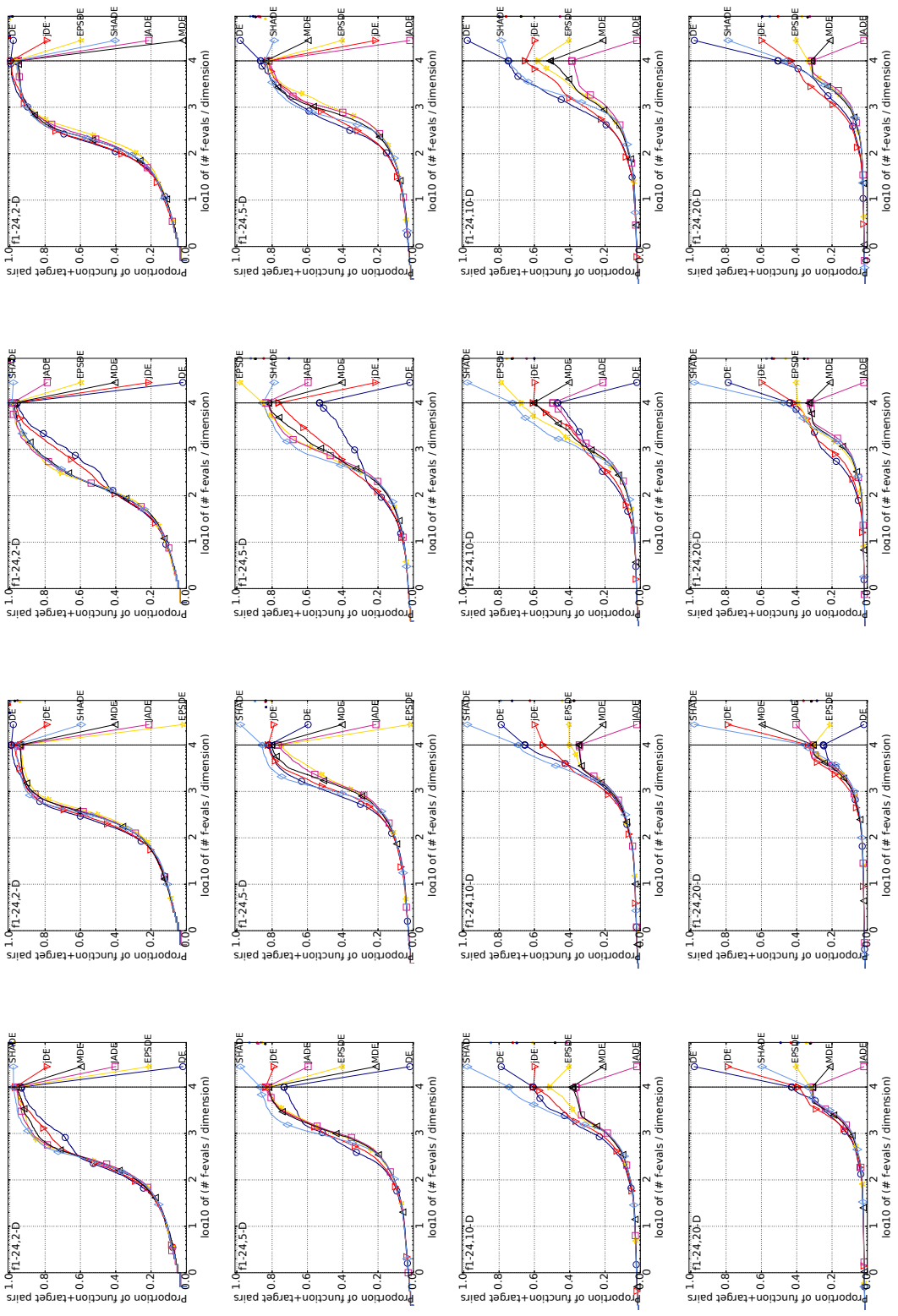
(a) rand/1/bin (b) rand/2/bin (c) best/1/bin (d) best/2/bin

図 6.6: 突然変異戦略に rand/1, rand/2, best/1, 及び best/2, 交叉手法に binomial 交叉を用いた場合における, SHADE と通常の DE (Algorithm 9), 及び 4 つの一般化した適応 DE (jDE, EPSDE, JADE, MDE) の比較結果 (左から, $D = 2, 5, 10, 20$). BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.



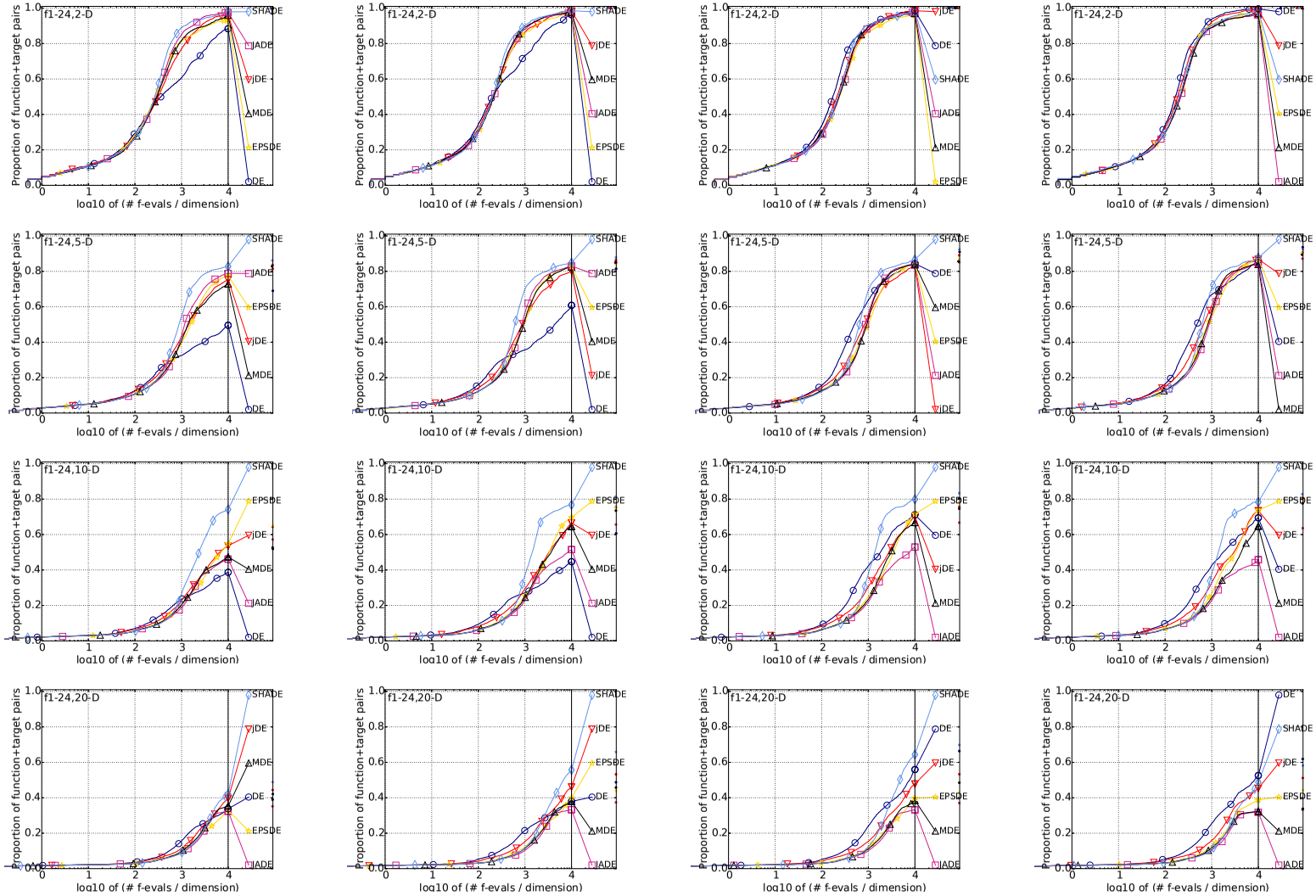
(a) current-to-rand/1/bin (b) current-to-best/1/bin (c) current-to-*p*best/1/bin (d) rand-to-*p*best/1/bin

図 6.7: 突然変異戦略に current-to-rand/1, current-to-best/1, current-to-*p*best/1, 及び rand-to-*p*best/1, 交叉手法に binomial 交叉を用いた場合における, SHADE と通常の DE (Algorithm 9), 及び 4 つの一般化した適応 DE の比較結果 (左から, $D = 2, 5, 10, 20$). BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.



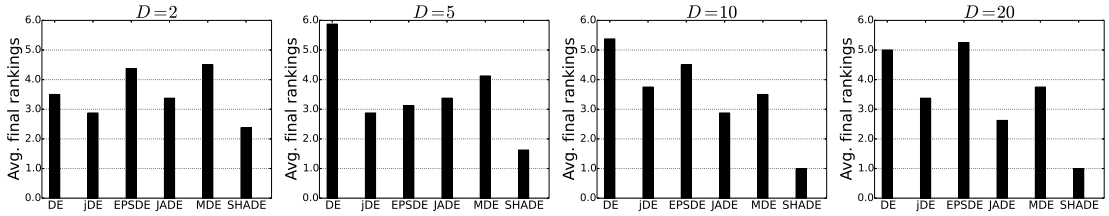
(a) rand/1/SEC　(b) rand/2/SEC　(c) best/1/SEC　(d) best/2/SEC

図 6.8: 突然変異戦略に rand/1, rand/2, best/1, 及び best/2, 交叉手法に SEC を用いた場合における, SHADE と通常の DE (Algorithm 9), 及び 4 つの一般化した適応 DE (jDE, EPSDE, JADE, MDE) の比較結果 (左から, $D = 2, 5, 10, 20$). BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.

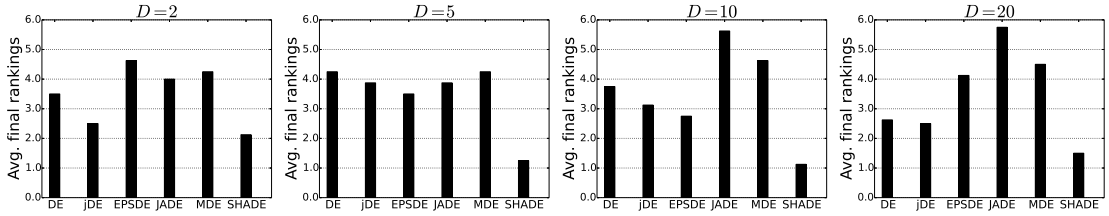


(a) current-to-rand/1/SEC (b) current-to-best/1/SEC (c) current-to-*p*best/1/SEC (d) rand-to-*p*best/1/SEC

図 6.9: 突然変異戦略に current-to-rand/1, current-to-best/1, current-to-*p*best/1, 及び rand-to-*p*best/1, 交叉手法に SEC を用いた場合における, SHADE と通常の DE (Algorithm 9), 及び 4 つの一般化した適応 DE の比較結果 (左から, $D = 2, 5, 10, 20$). BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.



(a) binomial 交叉



(b) SEC

図 6.10: SHADE を含む 6 手法における, 最大評価回数 $10^4 \times D$ の時点での, 8 つの突然変異戦略を使用した場合の交叉手法ごとの平均順位 ($D = 2, 5, 10, 20$). 平均順位が低いほど, どの突然変異戦略においても平均的に優れていることを意味する.

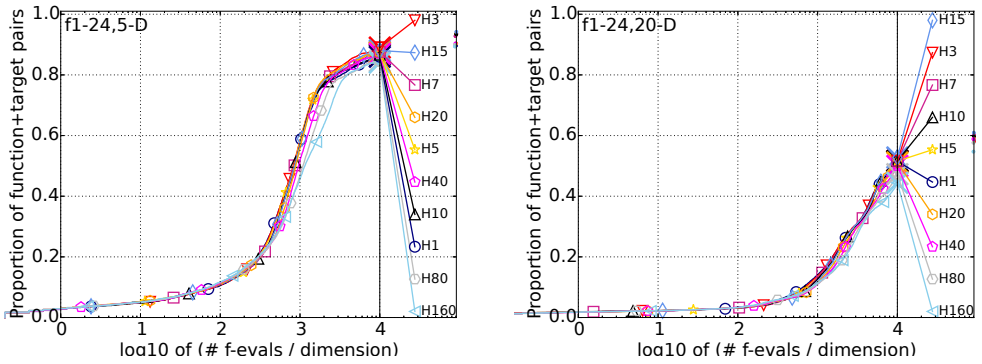
6.4.2　考察

本節では, 6.3 節と 6.4.1 節にて行った 2 つの実験結果に対する考察を行う. 6.3 節では, 5 章にて提案した理想化されたパラメータ適応の軌跡の代理である oracle パラメータを用いたシミュレーション法により, SHADE の適応能力を調査した. その結果, jDE, EPSDE, JADE, MDE といった既存の適応 DE の性能が乏しい, 世代の経過に対して急な変動を示す oracle パラメータにおいても, SHADE はある程度追従が可能であることがわかった. そのため, SHADE は 5.4 節にて述べた優れた適応手法であると考えられる.

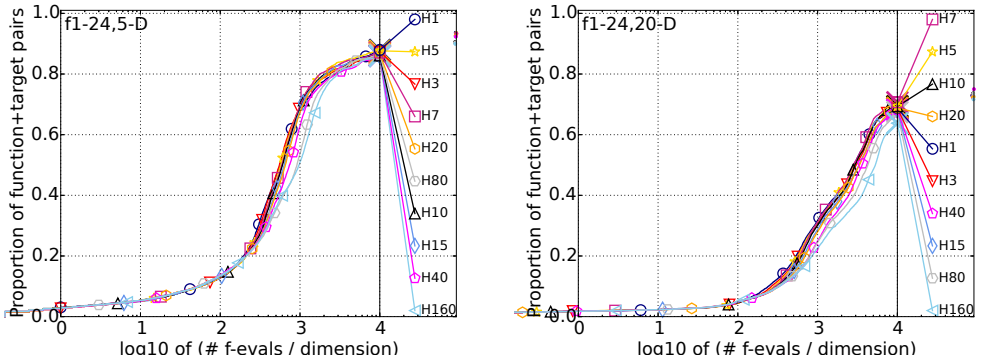
6.4.1 節にて行った, 8 種類の突然変異戦略と 2 つの交叉手法の組み合わせにおける各適応手法の, BBOB benchmarks を用いた性能評価実験では, 多くの場合 jDE, EPSDE, JADE, MDE と比べて SHADE が優れていた. また, 5.3 節の実験では JADE のように交叉手法を変えた途端に性能が大幅に劣化する手法も見られたが, SHADE は binomial 交叉と SEC の両者において良好な性能を示していることから, 汎用性の高い適応手法であると考えられる. この評価実験において SHADE が優れた性能を示していた理由は, 各探索状況における理想的なパラメータを近似する能力が SHADE は優れていたためであると, 先述の結果から言える.

6.5　SHADE におけるメモリサイズ H の影響

6.3 節と 6.4 節では, SHADE の制御パラメータであるメモリサイズ H は, JADE の学習率 c の推奨値 $c = 0.1$ に合わせて $H = 10$ としていた. しかし, $H = 10$ という設定は適切か否か,



(a) rand/1/bin



(b) current-to-$p$best/1/bin

図 6.11: 様々なメモリサイズ H を用いた SHADE の性能比較結果 ($D = 5, 20$). BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.

また異なる H の設定が SHADE の探索性能にどのような影響を与えるのかは不明である. そこで, 本節では SHADE の制御パラメータであるメモリサイズ H の設定が与える影響を調査する.

始めに 6.5.1 節にて, BBOB benchmarks におけるメモリサイズ H の設定の探索性能への影響を調査し, 次に 6.5.2 節にて oracle パラメータを用いたシミュレーション法における同様の調査を行う.

6.5.1 BBOB benchmarks におけるメモリサイズ H の設定が探索性能に与える影響の調査

本節では BBOB benchmarks におけるメモリサイズ H の設定が SHADE の探索性能に与える影響を調査する. 本実験では, $H = \{1, 3, 5, 7, 10, 15, 20, 40, 80, 160\}$ とし, その他の設定は

全て 6.4 節と同様にした．また, 交叉手法には binomial 交叉, 突然変異戦略には最も基本的な rand/1 と, 6.4 節の実験にて SHADE に比較的適していることがわかった current-to-$p$best/1 の 2 手法を使用した．

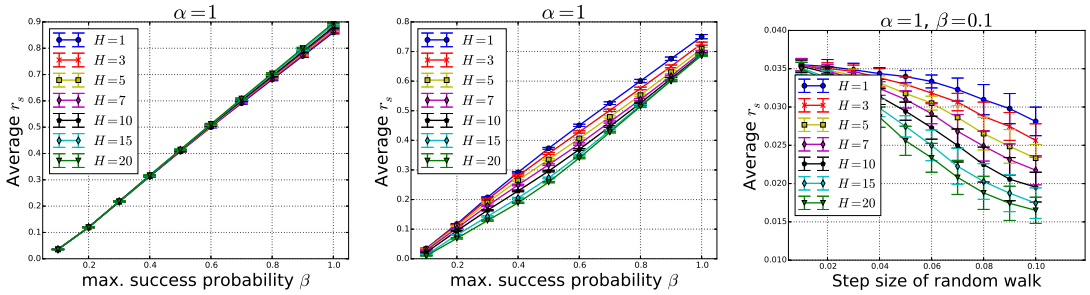
図 6.11 に, $D = 5, 20$ における様々なメモリサイズ H を用いた SHADE の性能比較結果を示す．図から, $D = 5$ にて rand/1 では $H = 3$, current-to-$p$best/1 では $H = 1$ が比較的良好な性能を示している．また, $H = 5, 7, 15, 20$ における性能は $H = 10$ とほとんど差は見られない．しかし, $D = 5, 20$, 及び rand/1 と current-to-$p$best/1 を使用した場合において, 多くの場合 $H = 80, 160$ が探索過程, 及び最終的に得られる解の質の両面で他の設定に劣っている．このことから, SHADE の探索性能は H の設定にある程度依存し, 適切な H の設定は対象問題の次元数, 及び使用するオペレータに依存すると言える．また, 多くの場合 $H \in \{5, ..., 20\}$ が適切な設定範囲であり, 6.3 節と 6.4 節にて使用した $H = 10$ という設定を用いても, 特に問題は無いと考えられる．

6.5.2　Oracle パラメータを用いたシミュレーション法における, メモリサイズ H の設定が適応性能に与える影響の調査

6.5.1 節では, BBOB benchmarks におけるメモリサイズ H の設定が SHADE の探索性能に与える影響を調査した．対して, 本節では oracle パラメータを用いたシミュレーション法におけるメモリサイズ H の設定が適応性能に与える影響を調査する．本実験では, $H = \{1, 3, 5, 7, 10, 15, 20\}$ とし, その他の設定は全て 6.4 節と同様にした．なお, その他の全ての実験設定は 5.5.1 節にて述べた設定と同様にした．

図 6.12 に一次関数 $o_{inc}(n_t)$, sin 関数 $o_{sin}(n_t)$, ランダムウォーク $o_{rw}(n_t)$ の 3 種類の oracle 関数における実験結果を示す．なお, sin 関数 $o_{sin}(n_t)$ の角周波数 ω については, $\omega = 40$ としている．また, ランダムウォーク $o_{rw}(n_t)$ では最大成功確率値 β を 0.1 としている．図から, $o_{inc}(n_t)$ における結果を除き, $H = 1$ での場合が r_s 値が最も高く, H が増加するにつれ次第に r_s 値が減少していくことがわかる．この結果から, oracle パラメータを用いたシミュレーション法では, $H = 1$ が最も適切であると言える．

しかし, これは 6.5.1 節にて行った BBOB benchmarks での実験にて得られた $H \in \{5, ..., 20\}$ が適切であるという結果に反する．SHADE における本実験で得られたこのような結果は, JADE の学習率 c のパラメータ設定においても同様の傾向が確認された．この結果から, 5.5 節, 及び 6.3 節での実験では, 各適応手法のベンチマーク集合における性能評価実験にて得られた推奨値を使用していたが, oracle パラメータを用いたシミュレーション法においても適切な設定であるとは限らないと言える．そのため, 実際の問題とシミュレーション環境における適切なパラメータ設定の関連性に関するさらなる調査が, 今後の課題として残される．



(a) $o_{inc}(n_t)$ (b) $o_{sin}(n_t)$ ($\omega = 40$) (c) $o_{rw}(n_t)$ ($\beta = 0.1$)

図 6.12: 様々なメモリサイズ H を用いた SHADE の oracle パラメータを用いたシミュレーション法における性能比較結果．一次関数 $o_{inc}(n_t)$, sin 関数 $o_{sin}(n_t)$, ランダムウォーク $o_{rw}(n_t)$ の 3 種類の oracle 関数における実験結果を示している．$o_{sin}(n_t)$ については $\omega = 40$, $o_{rw}(n_t)$ では $\beta = 0.1$ としている．縦軸は式 (5.6) の r_s の 101 試行の平均値である．(a), (b) での横軸は式 (5.5) における β の値, (c) では一世代におけるランダムウォークのステップサイズの大きさを制御する s の値を表す．また, 図中のエラーバーは標準偏差を表す．

6.6 L-SHADE: 決定的集団数減少法を用いた SHADE

6.4 節では SHADE を従来の適応 DE の適応手法と BBOB benchmarks にて比較することで, その有用性を確認した．また, 6.5 節ではメモリサイズ H の設定が SHADE の探索性能に与える影響を調査した．いずれの実験においても, 集団数 N は探索を通して $N = 5 \times D$ と固定していたが, 一般的に同一の対象問題においても適切なパラメータ設定は探索状況に応じて異なる．そのため, N についてのパラメータ制御法を導入することで, SHADE の探索性能をさらに改善できると考えられる．そこで, 本節では集団数 N にパラメータ制御法の一つである決定的集団数減少法 [127] を導入した際の効果を検証する．

パラメータ制御法は (1) 決定的パラメータ制御, (2) 適応的パラメータ制御, (3) 自己適応的パラメータ制御の 3 つに分類されるが, 4.2.2 節にて述べたように, 集団数 N のパラメータ制御には (1) の決定的パラメータ制御法が使用されることが近年では多い．一般的に, F や C とは異なり現世代の集団数の良し悪しを判定することは容易ではなく, 探索中に得られた情報を元に適切な集団数を調整することは困難であるため適応には複雑な制御機能が必要となる．その結果として, 集団数の設定が不必要となる代わりに 5 ～ 6 個の新たな制御パラメータを調整しなければならない [145]．このため, 予め定めたスケジュールに従い決定的にパラメータ設定を変更する (1) 決定的パラメータ制御法が, 集団数の制御においては比較的有用である．

4.2.2 節にて述べたように, EA における集団数の決定的パラメータ制御法はこれまでにいくつか提案されているが [127, 72, 44, 9], 本節では Laredo らに提案された世代の経過に対して決定的に集団数 N を減少させる方法 [127] を使用することにした．ここで, [127] で提案された手法は本来 GA での使用を前提としているが, 柔軟な枠組みであるため DE においても使用

可能である．また, DE においても決められた世代数ごとに集団数を半分つつ減少させる制御法が, 比較的良好な性能を示すことが報告されている [28].

Laredo らの手法 [127] では, τ と ρ という 2 つの制御パラメータを用いて集団数の減少スケジュールを決定していたが, ここでは最も単純な線形に集団数を減少させるスケジュールを採用した．以下では, この制御法を Linear Population Size Reduction (LPSR) と呼び, LPSR を導入した SHADE を L-SHADE と呼ぶ. LPSR では, 探索開始時 $t = 1$ における初期集団数 N_{init} から探索終了時の N_{fin} まで, 世代 t の経過に対して線形に集団数を減少させる．各世代 t の終了時に, 次世代 $t + 1$ の集団数 N_{t+1} を以下のように求める:

$$N_{t+1} = \left\lfloor (N_{\text{fin}} - N_{\text{init}}) \frac{n_t}{n_{\max}} + N_{\text{init}} \right\rfloor \tag{6.5}$$

ここで, $n_{\max}$ は最大評価回数, n_t は世代 t 終了時までに費やした評価回数である．N_{fin} は使用する突然変異戦略に必要な最小集団数である．例えば表 3.1 において, rand/2 であれば $N_{\text{fin}} = 6$, current-to-$p$best/1 ならば $N_{\text{fin}} = 4$ である．なお, この LPSR は探索の状況を一切考慮せずに決定的に集団数を調整する決定的パラメータ制御法であり, 適応的パラメータ制御法ではない (4.2.2 節参照)．ここで, 擬似焼きなまし法 (A.2.3 節参照) と同様に, 探索が開始される事前に最大評価回数 $n_{\max}$ が定まっていない場合, L-SHADE を使用することは困難である．また, 想定された最大評価回数未満で探索を停止した場合, L-SHADE よりも通常の SHADE の方が高精度の解を得られる可能性がある.

図 6.13 に, $D = 2, 5, 10, 20$ における L-SHADE と SHADE の性能比較結果を示す. SHADE, 及び L-SHADE の両手法において, 交叉手法に binomial 交叉, 突然変異戦略に current-to-$p$best/1 を使用した．これは, 6.4 節の実験にて比較的良好な性能を示した組合せである．なお, L-SHADE については, 予備実験から $N_{\text{init}} = 20 \times D$ とした．その他の設定は, 全て 6.4 節で述べた SHADE と同様にした．$D = 5, 10, 20$ において, 最大評価回数の時点, つまり探索終了時では L-SHADE は SHADE よりも優れた性能を示している．このことから, LPSR を導入することにより SHADE の探索性能を向上させることは可能であると言える．しかし, 探索過程における性能は, L-SHADE は明らかに SHADE よりも劣っている．これは, 探索序盤から中盤においては集団数が巨大であり, L-SHADE の探索の収束が遅いためであると考えられる．一方, 集団数が減少される探索終盤では, 局所的な探索が進んだために良解を求めることに成功したのだと思われる.

以上のことから, 対象問題において使用可能な最大評価回数が探索の事前に判明しており, 探索開始から最大評価回数を使い切り探索が停止するまでの期間にて得られる解の質を考慮しなくてもよい環境では, L-SHADE は SHADE よりも優れていると言える.

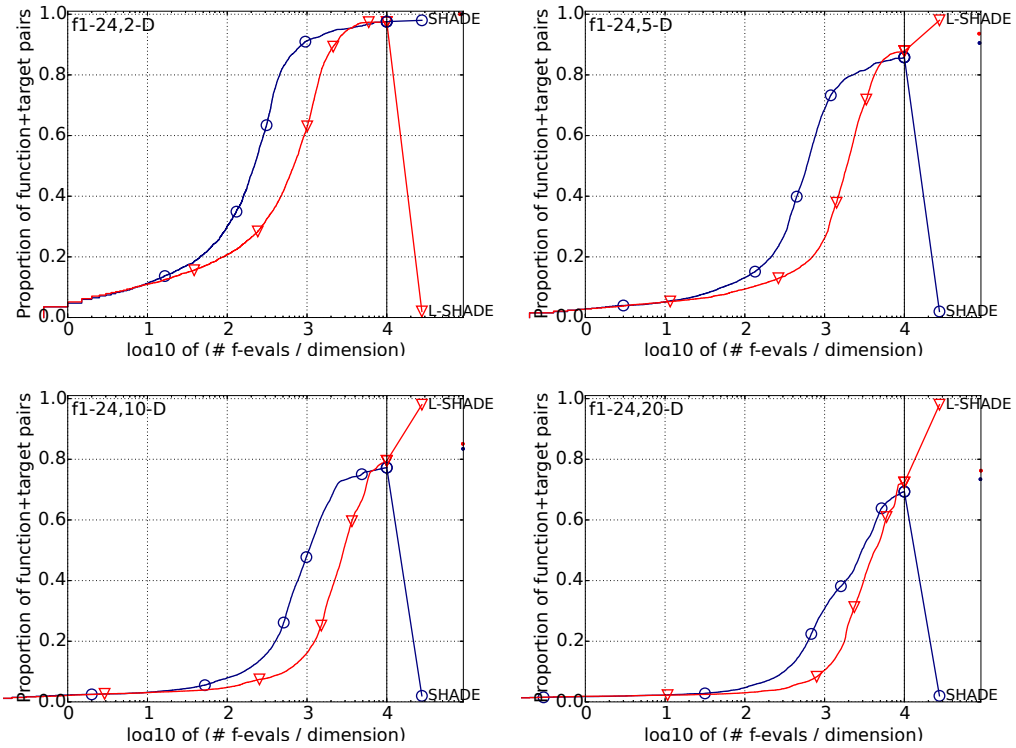


図 6.13: L-SHADE と SHADE の性能比較 ($D = 2, 5, 10, 20$). 両手法において, 交叉手法に binomial 交叉, 突然変異戦略に current-to-$p$best/1 を使用している. BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.

6.7 Black-box optimization 環境における関数最適化問題の state-of-the-art な探索手法との比較

本章で行われたこれまでの実験結果から, 提案手法である SHADE は従来の適応 DE よりも比較的優れていることがわかった. 一方, 2.4.4 節にて紹介した CMA-ES [96] にリスタート戦略を導入した手法が, black-box optimization 環境における関数最適化問題に対する state-of-the-art な手法である [92]. この restart CMA-ES の枠組みよりも優れた探索性能を持つ DE アルゴリズムは, 著者の知る限り既存研究には存在しない. そのため, SHADE が restart CMA-ES を始めとする他の EA と比べた際の探索性能を評価することは興味深い. また, 準ニュートン法 (A.1.3 節参照) や Nelder-Mead 法 (2.3.1 節参照) といった目的関数の勾配情報を必要としない伝統的な探索手法と比べ, SHADE がどの程度の性能を有するのかを確認することは有意義である. そこで, 本節では DE 以外の black-box optimization 環境における関数最適化問題に対する探索手法と SHADE の性能を比較する.

比較手法には, 以下に述べる 12 手法を用いた:

1. Random Search (RS) [31]　付録 A.2.1 節参照.
2. fminbnd [183]　商用ソフトウェアである MATLAB の fminbnd 関数 (黄金分割法に基づく直線探索手法) を利用した手法.
3. BFGS の公式を用いた準ニュートン法 (BFGS) [197]　付録 A.1.3 節参照. なお, 実装には商用ソフトウェアである MATLAB の fminfunc 関数を利用している.
4. Nelder-Mead 法 (Nelder) [167, 89]　2.3.1 節参照.
5. NEW Unconstrained Optimization Algorithm (NEWUOA) [186, 198]　2006 年に Powell に提案された, 各反復において目的関数の二次モデルを構築し最小化する信頼領域法に基づく決定的探索方法 [186].
6. ビットストリング GA (SimpleGA) [168]　2.4.1 節参照.
7. Particle Swarm Optimization (PSO) [210, 57]　2.4.5 節参照
8. Generalized Generation Gap with Parent-Centric Crossover (G3-PCX) [46]　2.4.2 節参照.
9. Memetic Algorithms Based on Local Search Chains (MA-LS-C) [162, 163]　交叉手法に BLX-α (式 (2.11)), 世代交代モデルに steady state (Algorithm 2) を用いた実数値 GA に, 局所探索法として CMA-ES (2.4.4 節参照) を組み込んだ手法.
10. BI-population CMA-ES (BIPOP-CMA-ES) [88]　2.4.4 節参照.
11. Hybrid CMA-ES (HCMA) [150]　BIPOP-CMA-ES[*3]に未評価の解の目的関数値を予測する surrogate モデル [149] を導入し, さらに NEWUOA と変数分離可な関数を効率的に探索可能な直線探索法である STEP [225] を適応的に使用する手法. 探索開始後はまず NEWUOA を決められた評価回数だけ実行し, その後 surrogate モデルを導入した BIPOP-CMA-ES と STEP を同時に実行する. そして, ある一定の期間にてより優れた解を獲得できた方の手法に, 残りの全ての計算資源 (評価回数) を与える. なお, HCMA は複数のアルゴリズムを使用する Algorithm Portfolio [105] の枠組みに分類される.
12. 通常の DE (Algorithm 9)　突然変異戦略に current-to-$p$best/1, 交叉手法に binomial 交叉を用いた通常の DE (Algorithm 9).

DE, SHADE, 及び L-SHADE の実験データ以外は全て COCO データベース[*4]からダウンロードした. SHADE, 及び L-SHADE については, 6.6 節にて得られた, 交叉手法に binomial 交叉, 突然変異戦略に current-to-$p$best/1 を用いた際の実験データを使用した.

図 6.14 に BBOB benchmarks における DE 以外の探索手法との SHADE の比較結果を示す ($D = 2, 5, 10, 20$). $D = 2$ においては, SHADE が比較手法の中では最良の性能を示している. その一方, 1965 年に提案されたにも関わらず, Nelder-Mead 法が SHADE, L-SHADE 及び CMA-ES 系以外の手法よりも良好な性能を示している. この傾向は先行研究 [92] と一致する. $D = 5, 10, 20$ と次元数が増加するにつれ, HCMA が探索過程, 及び最終的に得られ

[*3] 正確には, リスタート毎に初期大域的ステップサイズ σ_1 (2.4.4 節参照) を徐々に減少させる戦略 [149] を BIPOP-CMA-ES に組み込んでいる.

[*4] coco.gforge.inria.fr

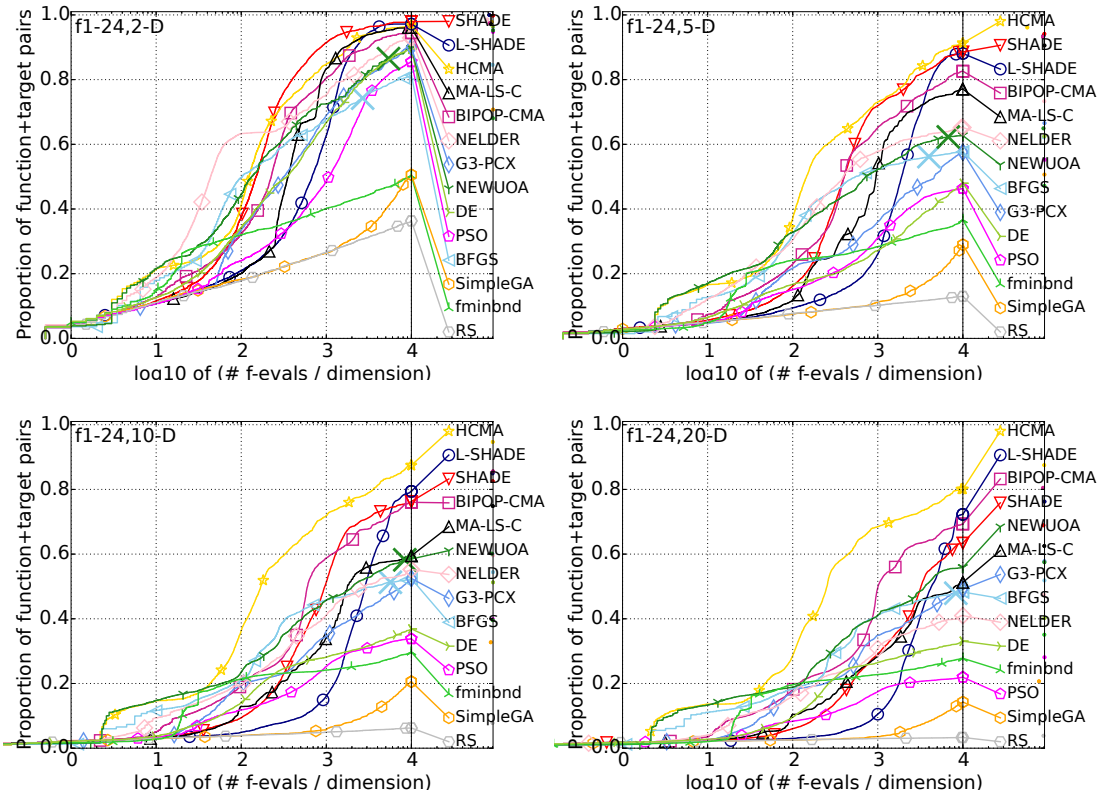
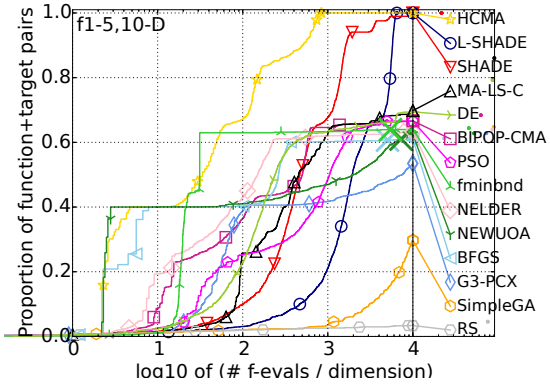


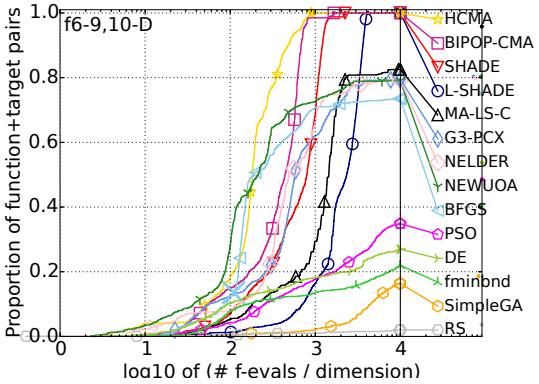
図 6.14: BBOB benchmarks における DE 以外の探索手法との SHADE の比較結果 ($D = 2, 5, 10, 20$). BBOB benchmarks の 24 個の関数 $f_1 \sim f_{24}$ における実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.

る解の質の両面で最良の性能を示している. 一方, L-SHADE, SHADE は HCMA に劣るものの, $D = 10$ にて両手法, $D = 20$ にて L-SHADE が近年の代表的な restart CMA-ES である BIPOP-CMA-ES よりも優れた性能を示している. また, $D = 5, 10, 20$ にて, Nelder-Mead 法や準ニュートン法と比べ, SHADE, 及び L-SHADE は最大評価回数の時点で 2 割以上の問題インスタンスを解くことに成功している. 実数値 GA の state-of-the-art である G3-PCX や関数最適化問題に対する代表的な EA である PSO やビットストリング GA と比べても, SHADE においては同様のことが言える.

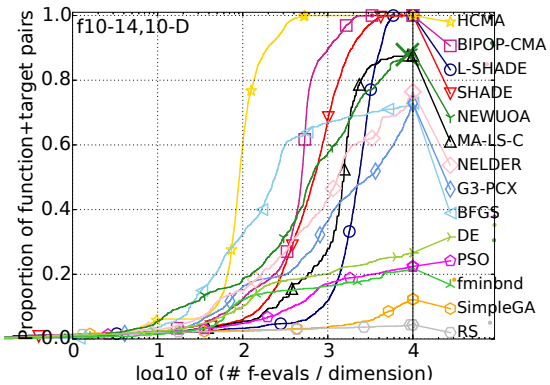
より詳細にアルゴリズムごとの特徴を把握するため, BBOB benchmarks の各関数クラスごとにおける比較結果を, 図 6.15 に示す ($D = 10$). 変数分離化な関数クラスである $f_1 \sim f_5$ では, SHADE, 及び L-SHADE が良好な性能を示している. これは, C を低い値に適応した際, 変数分離化な問題性質を SHADE は探索に利用可能であるためであると考えられる. 弱い悪スケール性 ($f_6 \sim f_9$), 強い悪スケール性 ($f_{10} \sim f_{14}$) の関数クラスでは, 収束速度の点では BIPOP-CMA-ES に劣るものの, SHADE と L-SHADE はほぼ全ての問題インスタンスを解



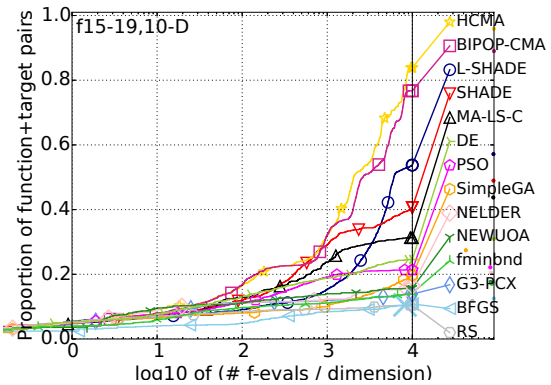
(a) $f_1 \sim f_5$ (変数分離化)



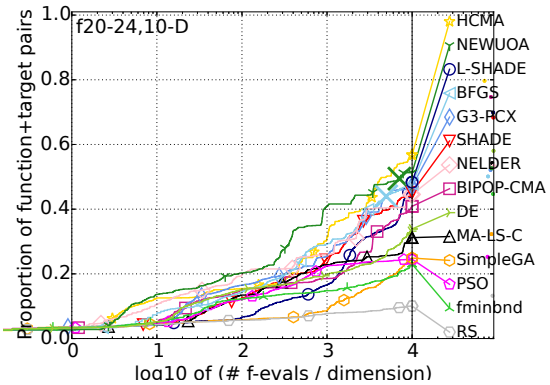
(b) $f_6 \sim f_9$ (弱い悪スケール性)



(c) $f_{10} \sim f_{14}$ (強い悪スケール性)



(d) $f_{15} \sim f_{19}$ (多峰性)



(e) $f_{20} \sim f_{24}$ (大域的構造の弱い多峰性)

図 6.15: BBOB benchmarks の各関数クラスにおける DE 以外の探索手法との SHADE の比較結果 ($D = 10$). BBOB benchmarks の各関数クラスごとの実験結果から作成した累積経験分布関数の図を示している. 縦軸についての値が高いほどより多くの関数インスタンスにおいて高い精度の解が得られていることを意味する. 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている. より詳細は, 付録 C.3 節を参照されたい.

くことに成功している．変数分離不可な多峰性関数クラスである $f_{15} \sim f_{19}$ では，収束速度と得られる解の質の両面で BIPOP-CMA-ES に劣っている．これは，先行研究 [94] の結果とある程度一致する．大域的構造の弱い多峰性関数クラス ($f_{20} \sim f_{24}$) では NEWUOA，準ニュートン法，及び G3-PCX に劣る性能を SHADE は示している．これは，一般的に EA は大域的構造の弱い問題クラスを不得意とするため [153]，局所探索法，及び G3-PCX のような局所的探索能力が非常に強い EA を繰り返しリスタートさせた方が良解が得られたためであると考えられる．

図 6.14, 6.15 に示す結果から，SHADE，及び L-SHADE は，準ニュートン法や Nelder-Mead 法といった目的関数の勾配情報を必要としない伝統的な探索手法よりも優れており，近年の代表的な restart CMA-ES であり black-box optimization 環境における state-of-the-art な一手法でもある BIPOP-CMA-ES に匹敵する性能を有すると言える．

一方，$D = 2$ における結果を除き，surrogate モデルを使用し，かつ複数のアルゴリズムから構成される Algorithm Portfolio [105] の枠組みに分類される HCMA (詳しくは先述) と比べた場合，SHADE は劣っていた．ここで，CMA-ES は回転不変性 (3.3.1 節参照) を有するため，複数の変数同士が依存関係にある変数分離不可な問題を効率的に探索できる．一方，対象問題が変数分離化な場合，各次元ごとに独立して探索することで効率的な探索が実現できるが，CMA-ES にはこの性質を利用することはできない．このように，対象問題の問題性質を事前に知ることができない black-box optimization 環境にて一つの探索手法しか利用しない場合，対象問題がその手法が不得意とする問題性質を有すると，良好な結果が望めない恐れがある．そこで，HCMA では各次元ごとに直線探索を行う変数分離化な関数において高性能な STEP [225] を CMA-ES と同時に，かつ適応的に実行することで，CMA-ES の問題点を補っている．このように探索失敗の危険性を少しでも減らすべく，複数の探索手法を実行する枠組みを Algorithm Portfolio と呼ぶ．

一般的に Algorithm Portfolio の枠組みを用いた手法は単一のアルゴリズムよりも良いとされているため [245, 180]，HCMA よりも優れた単一のアルゴリズムを設計することは困難である．そのため，(1) SHADE に surrogate モデルを導入し，(2) SHADE が不得意とする関数クラス (例えば，$f_{15} \sim f_{19}$) を得意とする手法を組み込み Algorithm Portfolio を構築することで，より高い探索性能を有する手法が実現されると考えられる．これを確かめることが今後の課題として残される．

6.8 まとめ

5 章での解析結果から，既存の適応 DE の適応手法にはいくつかの問題点があり，改善の余地が残されていることがわかった．これに対して本章では，既存の適応手法の問題点を考慮した，DE のための新たな適応手法である Success-History based Adaptive DE (SHADE) を提案した．SHADE では探索中に得られた対象問題に適したパラメータ設定をメモリに保存し，このメモリ内の要素を用いて新たにパラメータを生成することで，制御パラメータの自動調整を行う．

始めに, SHADE を理想的なパラメータ適応の軌跡の代理である oracle パラメータを用いたシミュレーション法にて, 他の適応 DE と比較した．その結果, jDE, EPSDE, JADE, MDE といった適応 DE の適応手法が適応に失敗する, oracle パラメータが世代ごとに急激に変化するモデルにおいても, SHADE は良好な性能を示すことがわかった．そのため, SHADE は 5.4 節にて述べた優れた適応手法であると考えられる．

次に, 8 種類の突然変異戦略と 2 つの交叉手法の組み合わせにおける各適応手法の, BBOB benchmarks を用いてベンチマーク集合における探索性能を評価した．その結果, 多くの場合 jDE, EPSDE, JADE, MDE と比べて SHADE が優れた性能を示した．また, これまで提案されてきたほぼ全ての適応 DE の適応手法は特定の突然変異戦略, 及び交叉手法の組合せを使用することを前提として設計されているためか, そのオペレータの組合せ以外を使用した場合, 5.3 節の実験結果のように探索性能の低下が手法によっては見られた．一方, SHADE は binomial 交叉と SEC の両者において良好な性能を示していることから, 既存の適応手法と比較してある程度汎用性の高い適応手法であると考えられる．また, black-box optimization 環境における関数最適化問題に対する伝統的な手法, 及び state-of-the-art な手法と比較したところ, DE 以外の手法と比べても SHADE は比較的優れた性能を示すことがわかった．このことから, SHADE は今後有望な探索手法として期待される．

第 7 章

SHADE を含む適応 DE の問題点について: ハイブリッド関数における適応 DE のパラメータ適応の振る舞いの解析

6 章では DE における新たな適応手法 SHADE を提案し, 既存手法と比較することでその有用性を実証した. しかし, SHADE があらゆる問題に対しても完璧に対応可能な, 万能な適応手法であると主張しているわけではない. 本章では SHADE を含む, 適応 DE の適応手法の問題点について述べる.

既存のほとんどのベンチマーク関数は, 全ての決定変数にて一様に同じ問題性質を有する. 一方, 実問題では決定変数ごとに異なる問題性質を有する可能性がある [82]. Tang らはこの問題性質を表すために, 複数のベンチマーク関数を合成したハイブリッド関数を提案した [231]. その後, 近年ではベンチマーク集合におけるハイブリッド関数の使用頻度が徐々に高まっているものの, EA の探索性能にどのような影響を与えるかは不明である.

本章ではハイブリッド関数における適応 DE のパラメータ適応の振る舞いを解析する. 解析対象となる適応 DE には, 5 章にて比較的良好な性能を示した jDE [27] (4.5.1 節参照), JADE [267] (4.5.3 節参照), 及び, 6 章にて提案した SHADE を使用した. 特に SHADE は black-box optimization 環境における関数最適化問題に対する伝統的な手法, 及び state-of-the-art な手法と比較しても良好な性能を有する.

しかし, 3 つの適応 DE をハイブリッド関数に適用した結果, 問題性質が互いに異なるベンチマーク関数から構成されたハイブリッド関数は, 適応 DE にとって困難な問題であることがわかった. このようなハイブリッド関数において適応 DE の探索が失敗する現象を解明するため, 適応手法のパラメータ適応の振る舞いを調査した. その結果, 適応 DE は対象問題の各決定変数における問題性質の一様性を暗黙の仮定としているために, 非一様な問題性質を有する問題には対応することが困難であることが明らかになった.

なお, 本章は著者が第一著者である先行研究 [230] に基づいている.

7.1　はじめに

5 章, 及び 6 章にて使用した BBOB benchmarks の 24 個のベンチマーク関数を含む, 既存のほとんどのベンチマーク関数は, 全ての決定変数にて一様に同じ問題性質を有する. 一方, 実問題では決定変数ごとに異なる問題性質を有する可能性がある [82]. このような問題性質は輸送ネットワーク, 回路理論 , 画像処理といった様々な実問題において現れうる [82].

Tang らはこの問題性質を表すために, 複数のベンチマーク関数を合成したハイブリッド関数を提案し, CEC2010 LSGO benchmarks を設計した [231]. その後, CEC2013 LSGO benchmarks [132], CEC2014 benchmarks [136] といった様々なベンチマーク集合にて, このハイブリッド関数が使用されるようになった. ハイブリッド関数では, まず解 $\boldsymbol{x} = (x_1, ..., x_D)^T$ の各決定変数は複数のグループに分けられ, 決定変数グループごとに異なるベンチマーク関数で評価される. 最後に, それぞれにおける目的関数値の総和をとり, その値を解 $\boldsymbol{x}$ の目的関数値とする. ここで, 互いに異なるベンチマーク関数を構成に使用した場合, ハイブリッド関数は従来のベンチマーク関数とは異なり一様な問題性質を持たない.

しかし, 近年ではベンチマーク集合におけるハイブリッド関数の使用頻度が徐々に高まっているものの, EA の探索性能にどのような影響を与えるかは不明である. 2.5.1 節で述べたようにベンチマーク関数を用いた性能評価実験の利点の一つは, 実問題に現れうる問題性質を端的に有するように設計されたベンチマーク関数にて探索手法の性能を評価することで, 一般性のある定性的な知見が得られることである. 一方, ハイブリッド関数を用いて探索手法の性能を評価することにより, どのような知見が得られるのかについてはこれまで研究されていない. 先行研究として, ハイブリッド関数が有する問題性質の一つである partial separability (2.2.4 節参照) に特殊化した手法がいくつか提案されているが [264, 34], これらの研究の興味はいかに効率的な探索を実現できるかであり, partial separability が EA に与える影響は調査されていない.

そこで, 本章ではハイブリッド関数が有する各決定変数における非一様な問題性質が, 適応 DE の適応手法にどのような影響を及ぼすのか解析する. 解析対象となる適応 DE には, 5 章にて比較的良好な性能を示した jDE [27] (4.5.1 節参照), JADE [267] (4.5.3 節参照), 及び, 6 章にて提案した SHADE を使用した. 特に SHADE は black-box optimization 環境における関数最適化問題に対する伝統的な手法, 及び state-of-the-art な手法と比較しても良好な性能を有する.

始めに, 様々なベンチマーク関数を組み合わせて構成したハイブリッド関数に適応 DE を適用したところ, 問題性質が互いに異なるベンチマーク関数から構成されたハイブリッド関数は, 適応 DE にとって困難な問題であることがわかった. 例えば, Sphere 関数 $f(\boldsymbol{x}) = \sum_{i=1}^{D} x_i^2$ のような二次関数から構成されるハイブリッド関数においても, 多峰性関数である Rastrigin 関数と組み合わせた場合, 適応 DE の多くの試行が最適解を求めることに失敗していた. 次に, ハイブリッド関数における適応手法のパラメータ適応の振る舞いを調査することで, このような適応 DE の探索失敗現象を解明する.

本章の構成は次の通りである．始めに 7.2 節にて本章で取り扱う 2 つの構成関数から成るハイブリッド関数について説明する．次に，7.3 節にてハイブリッド関数における適応 DE の探索性能を評価する．その後，7.4 節にてハイブリッド関数における適応手法のパラメータ適応過程を解析し，適応 DE の探索が失敗する現象を明らかにする．最後に 7.5 節にて本章をまとめる．

7.2　ハイブリッド関数の設計方法

本節では，本章で取り扱う 2 つの構成関数から成るハイブリッド関数について説明する．始めに 7.2.1 節にて先行研究におけるハイブリッド関数の設計方法を述べ，その後 7.2.2 節にて本研究におけるハイブリッド関数の設計方法を説明する．

7.2.1　先行研究におけるハイブリッド関数の設計方法

CEC2010 LSGO benchmarks [231] では，まず決定変数を 50 個のグループに均等に，かつ排他的に分ける．そして，特定のグループについてのみ，各グループ内の決定変数間において依存関係を有するように，個別に回転行列を適用する．また，あるグループについては，別のベンチマーク関数を割り当てる場合がある．

CEC2013 LSGO benchmarks [132] は，CEC2010 LSGO benchmarks をより実問題らしくするために，いくつかの工夫を加えたベンチマーク集合である．CEC2010 LSGO benchmarks とは異なり，CEC2013 LSGO benchmarks ではグループごとに決定変数の数が不均一になるように分ける．また，CEC2010 LSGO benchmarks では決定変数を排他的に分けていたが，CEC2013 LSGO benchmarks ではいくつかの決定変数は複数のグループに割り当てられており，変数分解法 [256, 172] (2.2.4 節参照) を適用するのが困難となるようにしている．さらに，グループごとに目的関数値への貢献度合いが異なるよう，重み付けをしている．

CEC2014 benchmarks [136] におけるハイブリッド関数は，解の決定変数を 3 ～ 5 個のグループに排他的に分け，それぞれ異なるベンチマーク関数にて評価し，得られた各目的関数値の総和を解 $\boldsymbol{x}$ の真の目的関数値としている．また，CEC2010 LSGO benchmarks と同様に，グループごとに独立して回転行列を適用することでグループ内の決定変数間に依存関係を持たせている．

7.2.2　本研究におけるハイブリッド関数の設計方法

本章では 7.2.1 節にて述べた先行研究 [231, 132, 136] の生成方法を単純化し，決定変数を分割するグループの数を 2，つまり 2 つの構成関数 f, g を使用し，回転行列は用いずに異なるベンチマーク関数を構成関数に使用することで，ハイブリッド関数 $H_{f,g}$ を設計した．これは，最も単純なハイブリッド関数の実装における適応 DE の探索性能を解析するためである．以下では，本研究にて使用したハイブリッド関数 $H_{f,g}$ の設計方法を説明する．

D 次元の解 $\boldsymbol{x} = (x_1, ..., x_D)^T$ を，2 つの任意の構成関数 f と g から成るハイブリッド関数 $H_{f,g}$ にて評価し，目的関数値 $H_{f,g}(\boldsymbol{x})$ を得る手順を考える．始めに，$\boldsymbol{x}$ の各決定変数

$(x_1, ..., x_D)$ を, (1) f で評価するグループ $\boldsymbol{x}^f = (x_1^f, ..., x_{D_f}^f)^T$ と, (2) g で評価するグループ $\boldsymbol{x}^g = (x_1^g, ..., x_{D_g}^g)^T$ の 2 つに排他的に分ける. ここで, D_f, D_g はそれぞれ $\boldsymbol{x}^f, \boldsymbol{x}^g$ の次元数であり, $D = D_f + D_g$ である. 本章では構成関数 $f : \mathbb{R}^{D_f} \to \mathbb{R}$ と $g : \mathbb{R}^{D_g} \to \mathbb{R}$ にて評価される決定変数の数の割合を調整するため, 比率 $r \in [0, 1]$ をハイブリッド関数の制御パラメータとして導入した. $D_f = \lfloor D \times r \rfloor$, 及び $D_g = D - D_f$ とし, 必ず $D = D_f + D_g$ となるようにした. この時, 例えば r を高い値に設定した場合, $\boldsymbol{x}^f$ にはより多くの決定変数が割り当てられることとなる. そして, $\boldsymbol{x}^f$ に割り当てられる D_f 個の決定変数は, $(x_1, ..., x_D)$ からランダムに選択し, 選ばれなかった残りの D_g 個の決定変数を $\boldsymbol{x}^g$ に割り当てた[*1].

解 $\boldsymbol{x}$ を $\boldsymbol{x}^f$, $\boldsymbol{x}^g$ に分けた後, ハイブリッド関数 $H_{f,g}$ における解 $\boldsymbol{x}$ の目的関数値を次式により得る:

$$H_{f,g}(\boldsymbol{x}) = w_f f(\boldsymbol{x}^f) + w_g g(\boldsymbol{x}^g) \tag{7.1}$$

ここで, $f(\boldsymbol{x}^f)$ と $g(\boldsymbol{x}^g)$ は任意の構成関数 f, 及び g にて $\boldsymbol{x}^f$, 及び $\boldsymbol{x}^g$ を評価することで得られる. 例えば, $D = 30$ のハイブリッド関数 $H_{f,g}$ において f, 及び g がそれぞれ Sphere 関数と Rastrigin 関数であったとする. また, $r = 0.4$ であった場合, $D_f = \lfloor 30 \times 0.4 \rfloor = 12$, $D_g = D - D_f = 30 - 12 = 18$ である. この際, $\boldsymbol{x}$ の 12 個の決定変数から成る $\boldsymbol{x}^f$ が Sphere 関数, 18 個の決定変数から成る $\boldsymbol{x}^g$ が Rastrigin 関数にて評価され, 各目的関数値の総和がハイブリッド関数 $H_{\text{Sphere,Rastrigin}}$ における解の目的関数値 $H_{\text{Sphere,Rastrigin}}(\boldsymbol{x})$ となる.

式 (7.1) において, w_f, w_g は任意の正の実数である. w_f, w_g を調整することで, 7.2.1 節にて述べた CEC2013 LSGO benchmarks [132] のように各グループごとに目的関数値 $H_{f,g}(\boldsymbol{x})$ への貢献度合いが異なるよう, 重み付けが可能である. しかし, 本章では最も単純なハイブリッド関数の実装における適応 DE の探索性能を解析するため, $w_f = w_g = 1$ とした.

7.3　2 つの構成関数から成るハイブリッド関数における適応 DE の性能評価

本節では, 2 つの構成関数から成るハイブリッド関数における適応 DE の性能評価を行う. 始めに 7.3.1 節にて実験設定について説明する. その後, 7.3.2 節にて構成関数が互いに異なる性質を有するハイブリッド関数, 7.3.3 節にて構成関数が互いに似た問題性質を有するハイブリッド関数における実験結果と考察をそれぞれ述べる.

[*1] $\boldsymbol{x} = (x_1, ..., x_D)^T$ において, $(x_1, ..., x_{D_f})$ を $\boldsymbol{x}^f$ に, $(x_{D_f+1}, ..., x_D)$ を $\boldsymbol{x}^g$ に割り当てるといったように, 単純に連続した決定変数列をそれぞれに割り当てる方法が考えられる. しかし, この方法では依存関係にある決定変数同士が隣接している場合があり, この性質は 8 章にて述べたように exponential 交叉などの変数番号が連続した複数の決定変数を子個体に受け継がせるオペレータを使用することで, 探索に利用可能である. これを防ぐため, 本論文では $\boldsymbol{x}^f$, $\boldsymbol{x}^g$ に割り当てる決定変数は, $(x_1, ..., x_D)$ からランダムに選択した.

7.3.1　実験設定

本実験には, 一般的なベンチマーク関数である Sphere 関数, Schwefels 1.2 関数, Rosenbrock 関数, Rastrigin 関数, Ackley 関数を 2 つ組み合わせることでハイブリッド関数を設計した. 表 7.1 に, 各ベンチマーク関数の定義と性質を示す. ベンチマーク関数ごとに探索範囲が異なるが, 簡便のため探索領域は $[-100, 100]^D$ とし, 評価の際に各決定変数の値を元の探索領域に伸縮した. また, 最適解 $\boldsymbol{x}^*$ の位置を各次元ごとに $[-80, 80]$ の範囲に一様ランダムに生成したベクトルにシフトした. $\boldsymbol{z} = (z_1, ..., z_D)^T$, $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}$ であり, 各関数の最適解の位置を $\boldsymbol{o} = (o_1, ..., o_D)^T$ だけシフトさせている. $\boldsymbol{o}$ は各次元 $\{1, ..., D\}$ において $[-80, 80]$ の探索範囲内に一様ランダムに生成した実数値ベクトルである. いずれの関数も, $\boldsymbol{z} = (0, ..., 0)^T$ にて最適値 $f_{\text{best}} = 0$ を取る. これらの関数の設定は, 全て CEC2014 benchmarks [136] におけるハイブリッド関数と同様である.

表 7.2 に, 表 7.1 の 5 つのベンチマーク関数を組み合わせることで設計した, 本研究にて使用する 8 つのハイブリッド関数を示す. ここで, 7.2.2 節にて説明した通り, $H_{f,g}$ は関数 f と g から構成されていることを表す. 例えば, $H_{\text{Sph,Ras}}$ は Sphere 関数と Rastrigin 関数から成ることを意味する. 表 7.2 中の上段から 6 個のハイブリッド 関数は, 例えば $H_{\text{Sph,Ras}}$ は単峰性関数である Sphere 関数と多峰性関数である Rastrigin 関数から構成されているように, 互いに異なる問題性質を持つベンチマーク関数を組み合わせて設計されている. 一方, 残りの 2 個のハイブリッド関数である $H_{\text{Sph,Sch}}$, 及び $H_{\text{Ras,Ack}}$ は, 構成関数がそれぞれ単峰性関数, 及び多峰性関数同士と, 互いに似た問題性質を有する. 本章では前者の問題クラスをヘテロジニアス, 後者をホモジニアスと呼び, それぞれ 7.3.2 節と 7.3.3 節にて使用する.

次元数は CEC2014 benchmarks [136] に合わせ, $D = 30, 50$ とした. 7.2.2 節にて述べた構成関数の比率 $r \in [0, 1]$ は, 0.0 から 1.0 まで 0.1 刻みで変化させた. また, 探索中に得られた最良解の目的関数値 $f(\boldsymbol{x}^{bsf})$ と最適値 $f(\boldsymbol{x}^*)$ との誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ が 10^{-8} 以下になった場合は, その探索を成功とみなし打ち切った. 反対に, 評価回数が $D \times 10,000$ を超えた場合は, その探索を失敗とした. 各問題, 及び各 r の値について試行回数は 50 回とし, 探索成功率, つまり成功した探索の回数を試行回数 50 で割った値を各手法の性能評価指標として使用した.

解析対象となる適応 DE には, 5 章にて比較的良好な性能を示した jDE [27] (Algorithm 11), JADE [267] (Algorithm 13), 及び, 6 章にて提案した SHADE (Algorithm 19) を使用した. JADE, 及び SHADE については集団数 $N = 100$, jDE については $N = 50$ とした[*2]. SHADE については, 6.4 節の実験にて比較的良好な組合せであった突然変異戦略に current-to-$p$best/1, 交叉手法に binomial 交叉を使用した. その他の設定は, 5 章, 及び 6 章のパラメータ設定と同様である.

また, 突然変異戦略に rand/1, 交叉手法に binomial 交叉を用いた通常の DE (Algorithm 8) も比較のため評価した. 通常の DE のパラメータ設定は, 広く使われている組合せである, 集

[*2] jDE に関しては $N = 100$ を使用した場合, Schwefel 1.2 と Rosenbrock 関数において次元数に依らずに与えられた評価回数下で収束することが出来なかったため, 集団数を半分の 50 とした.

表 7.1: 7 章のハイブリッド関数の設計のために使用した, 5 個のベンチマーク関数. $\boldsymbol{z} = (z_1, ..., z_D)^T$ については本文を参照されたい.

関数名	定義	問題性質
Sphere 関数	$f(\boldsymbol{x}) = \sum_{i=1}^{D} z_i^2$	変数分離可, 単峰性
Schwefels 1.2 関数	$f(\boldsymbol{x}) = \sum_{i=1}^{D} (\sum_{j=1}^{i} z_j)^2$	変数分離不可, 単峰性
Rosenbrock 関数	$f(\boldsymbol{x}) = \sum_{i=1}^{D-1} \left(100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2\right)$	変数分離不可, 弱い多峰性
Rastrigin 関数	$f(\boldsymbol{x}) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos 2\pi(z_i) + 10\right)$	変数分離可, 強い多峰性
Ackley 関数	$f(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{1/n \sum_{i=1}^{n} z_i^2}\right)$ $-\exp\left(1/n \sum_{i=1}^{n} \cos(2\pi z_i)\right) + 20 + \mathrm{e}$	変数分離可, 弱い多峰性

表 7.2: 7 章にて使用した 8 個のハイブリッド関数. 上段から 6 個のハイブリッド 関数は, 互いに異なる問題性質を持つベンチマーク関数を構成関数 f, g として設計されている. 一方, 残りの 2 個のハイブリッド関数である $H_{\mathrm{Sph,Sch}}$, 及び $H_{\mathrm{Ras,Ack}}$ は構成関数 f, g が互いに似た問題性質を有する. 本章では前者の問題クラスをヘテロジニアス, 後者をホモジニアスと呼ぶ.

ハイブリッド関数の種類	ハイブリッド関数 $H_{f,g}$	構成関数 f	構成関数 g
ヘテロジニアス	$H_{\mathrm{Sph,Ras}}$	Sphere	Rastrigin
	$H_{\mathrm{Sph,Ack}}$	Sphere	Ackley
	$H_{\mathrm{Sch,Ras}}$	Schwefel 1.2	Rastrigin
	$H_{\mathrm{Sch,Ack}}$	Schwefel 1.2	Ackley
	$H_{\mathrm{Ros,Ras}}$	Rosenbrock	Rastrigin
	$H_{\mathrm{Ros,Ack}}$	Rosenbrock	Ackley
ホモジニアス	$H_{\mathrm{Sph,Sch}}$	Sphere	Schwefel 1.2
	$H_{\mathrm{Ras,Ack}}$	Rastrigin	Ackley

団数 $N = 50$[*3], $F = 0.5$, $C = 0.1, 0.9$ を使用した [27, 193, 267]. $C = 0.1, 0.9$ はそれぞれ変数分離可, 及び変数分離不可な関数に適している [196]. 以下本章では, 各 C の値を用いた通常の DE を DE-C0.1, DE-C0.9 と表記する.

[*3] 一般的には $N = 100$ であるが, jDE の集団数の設定と同様の理由でこの値を用いた.

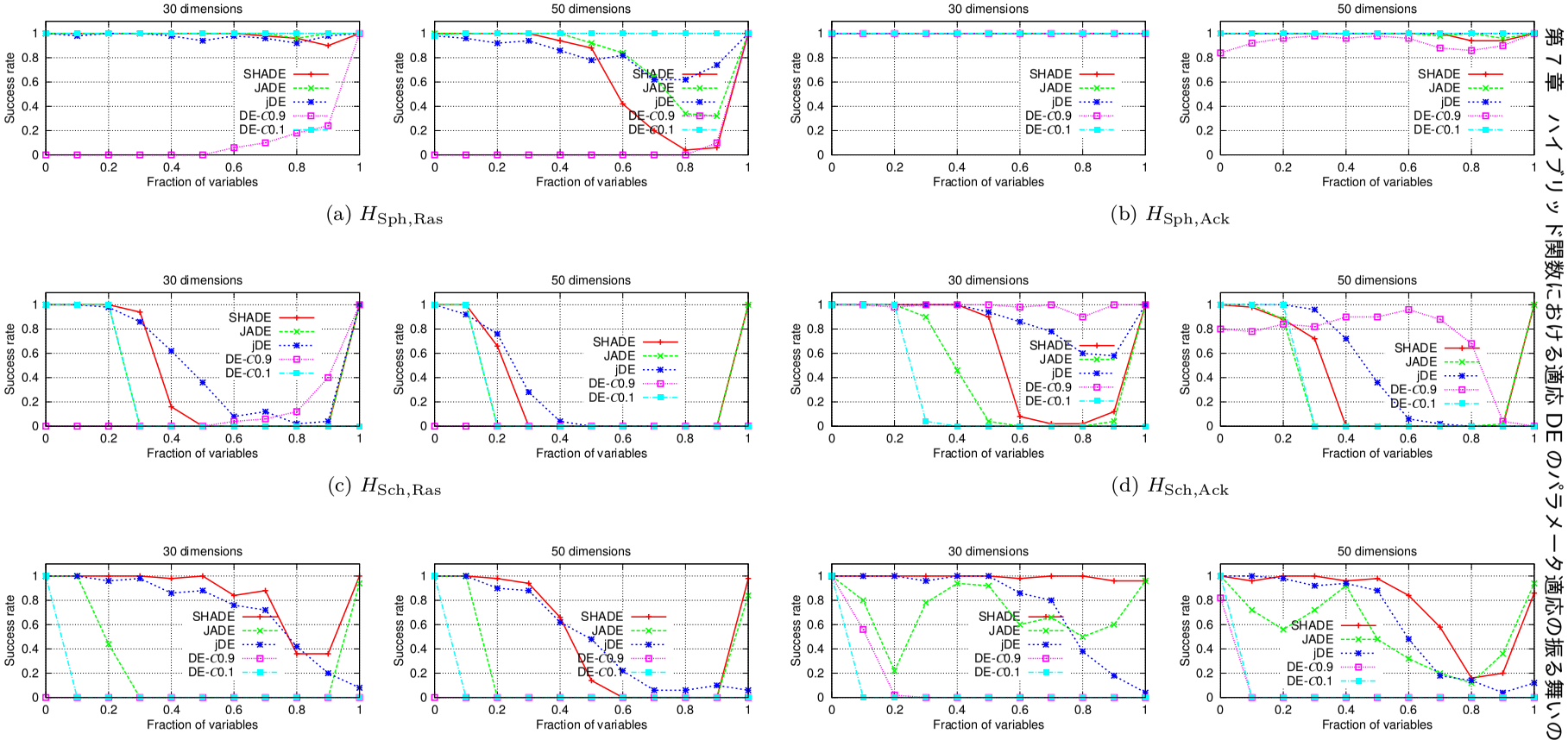
7.3.2　構成関数が互いに異なる性質を有するハイブリッド関数における実験結果と考察

図 7.1 に構成関数が互いに異なる性質を有するハイブリッド関数 ($H_{\mathrm{Sph,Ras}}$, $H_{\mathrm{Sph,Ack}}$, $H_{\mathrm{Sch,Ras}}$, $H_{\mathrm{Sch,Ack}}$, $H_{\mathrm{Ros,Ras}}$, $H_{\mathrm{Ros,Ack}}$) における, 構成関数 f, g の構成比率 r に対する各手法の探索成功率を示す. なお, $D = 30, 50$ である. 図 7.1 から, 以下のような傾向が見られた:

- 図 7.1(b) の 30 次元における $H_{\mathrm{Sph,Ack}}$ の結果のみ, r の値に依らず全ての手法の探索成功率が 1 となった.
- 適応 DE (SHADE, JADE, jDE) は, 50 次元における Schwefel 1.2 関数と Rosenbrock 関数での jDE の結果を除き, $r = 0.0, 1.0$ の時に高い探索成功率を示す. このことは, 表 7.1 に示した構成関数 f, g 単独において, 適応 DE は比較的容易に最適解を求めることが可能であることを意味する.
- 適応 DE の探索成功率は r が 0 に近づくほど高くなるが, 多くの場合 $r = 0.5 \sim 0.9$ にて乏しい成功率を取る. しかし, $r = 1.0$ において再度高い成功率を示す. 結果的に V 字, 又は U 字のような成功率の推移を示す.
- 多くの場合, $D = 50$ における探索成功率は $D = 30$ での結果と比べ低い値を取る. これは, 高次元関数ほど探索が困難であるためであると考えられる.

一般的に, 単峰性関数よりも多峰性関数の方が, 探索が局所解に誤って収束してしまい最適解を求めるのに失敗する可能性があるため, 探索が困難である. そのため, $H_{\mathrm{Sph,Ras}}$, $H_{\mathrm{Sph,Ack}}$, $H_{\mathrm{Sch,Ras}}$, $H_{\mathrm{Sch,Ack}}$ では Rastrigin 関数, 及び Ackley 関数の構成比率が高くなるほど, 各ハイブリッド関数のインスタンスは単調に探索が困難になることが予想された. しかし, こうした多峰性関数の比率が高いほど, 適応 DE の探索成功率が高くなる現象が実験では見られた. 一方, ハイブリッド関数の構成要素の 8 ∼ 9 割が Sphere 関数, 及び Schwefels 1.2 関数といった単峰性関数の時が SHADE, JADE, jDE の探索が最も困難となる場合があった. この傾向は多峰性関数同士の Rosenbrock 関数と Rastrigin, Ackley 関数のハイブリッド関数である $H_{\mathrm{Ros,Ack}}$, $H_{\mathrm{Ros,Ras}}$ においても観測された.

特に, 図 7.1(a) の $H_{\mathrm{Sph,Ras}}$ における実験結果は, black-box optimization 環境における関数最適化問題に対する伝統的な手法, 及び state-of-the-art な手法と比較しても良好な性能を有する SHADE が, ほぼ Sphere 関数と等しいハイブリッド関数において探索が頻繁に失敗するという興味深い結果である. また, 一般的に DE は変数分離可な関数においては, 対象問題が多峰性関数であった場合においても, 他の EA と比べ良好な探索性能を示すことが知られている [94, 224]. しかし, $H_{\mathrm{Sph,Ras}}$ は決定変数を各次元 $j \in \{1, ..., D\}$ ごとに分解可能な関数であるにも関わらず DE の探索が困難であった. このことから, $H_{\mathrm{Sph,Ras}}$ は変数分離可であるにも関わらずに DE の探索が困難な関数の初めての事例であると考えられる. さらに, DE–C0.1 が $H_{\mathrm{Sph,Ras}}$ においては比較手法中最も高い成功率を示すという, 適応 DE は通常の DE よりも探索能力が高いというこれまでの知見に反する結果となった.

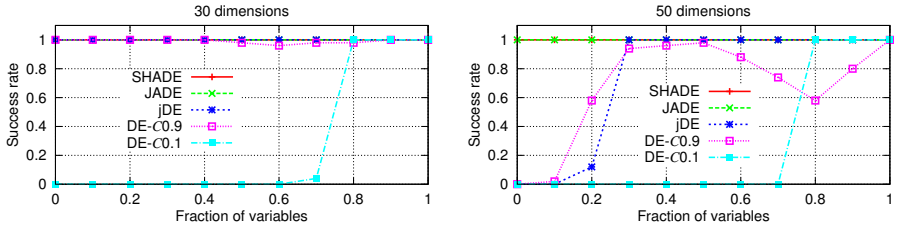


(a) $H_{\mathrm{Sph,Ras}}$ (b) $H_{\mathrm{Sph,Ack}}$

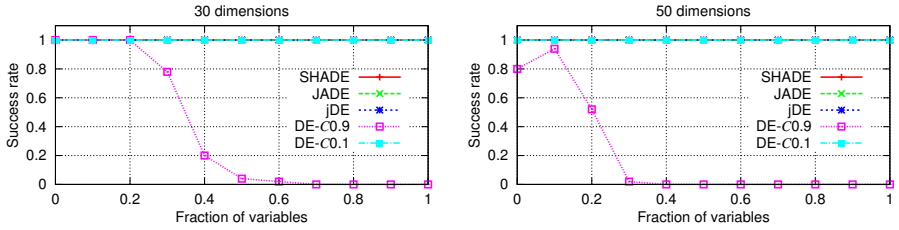
(c) $H_{\mathrm{Sch,Ras}}$ (d) $H_{\mathrm{Sch,Ack}}$

(e) $H_{\mathrm{Ros,Ras}}$ (f) $H_{\mathrm{Ros,Ack}}$

図 7.1: 構成関数 f, g が互いに異なる性質を有するハイブリッド 関数 ($H_{\mathrm{Sph,Ras}}$, $H_{\mathrm{Sph,Ack}}$, $H_{\mathrm{Sch,Ras}}$, $H_{\mathrm{Sch,Ack}}$, $H_{\mathrm{Ros,Ras}}$, $H_{\mathrm{Ros,Ack}}$) における, 構成比率 r に対する適応 DE (jDE, JADE, SHADE), 及び通常の DE (Algorithm 8) の 50 試行中の探索成功率の推移 ($D = 30, 50$).



(a) $H_{\mathrm{Sph,Sch}}$



(b) $H_{\mathrm{Ras,Ack}}$

図 7.2: 構成関数 f, g が互いに似た性質を有するハイブリッド関数 ($H_{\mathrm{Sph,Sch}}$, $H_{\mathrm{Ras,Ack}}$) における, 構成比率 r に対する適応 DE (jDE, JADE, SHADE), 及び通常の DE (Algorithm 8) の 50 試行中の探索成功率の推移 ($D = 30, 50$).

7.3.3　構成関数が互いに似た問題性質を有するハイブリッド関数における実験結果と考察

7.3.2 節では, 構成関数が互いに異なる問題性質を有するハイブリッド関数において, 特定の構成比率 r の下, 適応 DE の探索成功率が著しく低下する現象が見られた. この結果から, 任意の 2 つのベンチマーク関数を組み合わせただけでも, このような探索失敗現象が起こり得る可能性がある. そこで, 本節では 7.3.2 節とは反対に, 構成関数が互いに類似した性質を有するハイブリッド関数において同様の実験を行った. 実験には表 7.2 のハイブリッド関数 $H_{\mathrm{Sph,Sch}}$ と $H_{\mathrm{Ras,Ack}}$ を用いた. ここで, $H_{\mathrm{Sph,Sch}}$ は互いに単峰性関数, $H_{\mathrm{Ras,Ack}}$ は互いに多峰性関数から構成されている.

30, 50 次元における $H_{\mathrm{Sph,Sch}}$ と $H_{\mathrm{Ras,Ack}}$ における結果を図 7.2 に示す. SHADE, JADE, jDE は全ての比率 r において, $H_{\mathrm{Sph,Sch}}$, 及び $H_{\mathrm{Ras,Ack}}$ では成功率はほぼ 1 である. これは, 多峰性関数同士の組合せである $H_{\mathrm{Ras,Ack}}$ よりも, 図 7.1(a) に結果を示した単峰性関数が含まれる $H_{\mathrm{Sph,Ras}}$ の方が探索が困難であるという, 直感に反する結果である. この結果から, 7.3.2 節で述べたように, ハイブリッド関数において構成関数が互いに異なる問題性質を有する場合に限り, 特定の比率 r の下で DE の探索性能が悪化するのであり, $H_{\mathrm{Sph,Sch}}$, 及び $H_{\mathrm{Ras,Ack}}$ のように互いに似た 2 つの関数を組み合わせただけでは, このような特徴的な探索失敗現象は起こる可能性は低いと言える.

しかし, 図 7.1(b) に結果を示した $H_{\mathrm{Sph,Ack}}$ では, 互いに異なる問題性質を有する他のハイ

ブリッド関数とは異なり, r の値に関わらず全ての手法において探索成功率が高い．これは, Sphere 関数は単峰性, Ackley 関数は比較的弱い多峰性であり, かつ両者とも変数分離可と互いに性質が似ていたため, 本節で議論した $H_{\mathrm{Sph,Sch}}$ と $H_{\mathrm{Ras,Ack}}$ における結果と同様の傾向を示したのだと考えられる．

7.4　ハイブリッド関数における適応 DE のパラメータ適応過程の解析

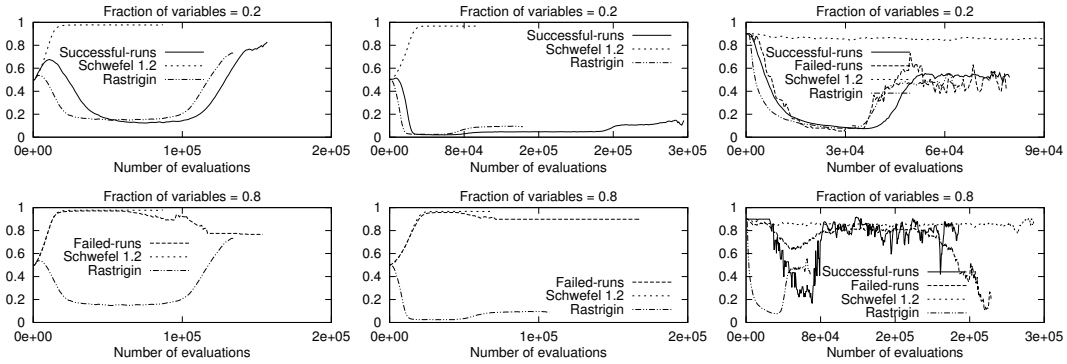
本節では 7.3.2 節にて確認された, 構成関数が互いに異なる問題性質を有するハイブリッド関数において, 特定の構成比率 r の下, 適応 DE の探索が失敗する現象を明らかにする．始めに, 7.4.1 節にてハイブリッド関数における適応 DE のパラメータ適応過程の視覚に基づく解析を行う．次に, 定量的な傾向を確認するために, 7.4.2 節において適応過程におけるパラメータ系列間の距離に基づく解析方法を提案し, 7.4.3 節にて結果を示す．最後に, 7.4.4 節にてこれらの解析結果に対して考察する．

7.4.1　ハイブリッド関数における適応 DE のパラメータ適応過程の視覚に基づく解析

本節では, 7.3.2 節にて見られたハイブリッド関数における適応 DE の探索失敗現象を明らかにするため, SHADE, JADE, jDE のパラメータ適応過程を可視化し解析する．なお, 5.4.1 節にて指摘したとおり, 視覚に基づく適応手法の解析法にて得られる情報は限られているが, ある問題インスタンスにおいて特定の適応手法がどのように振る舞うのかを大まかに把握することが可能であるため, 本節では本アプローチを採用した．

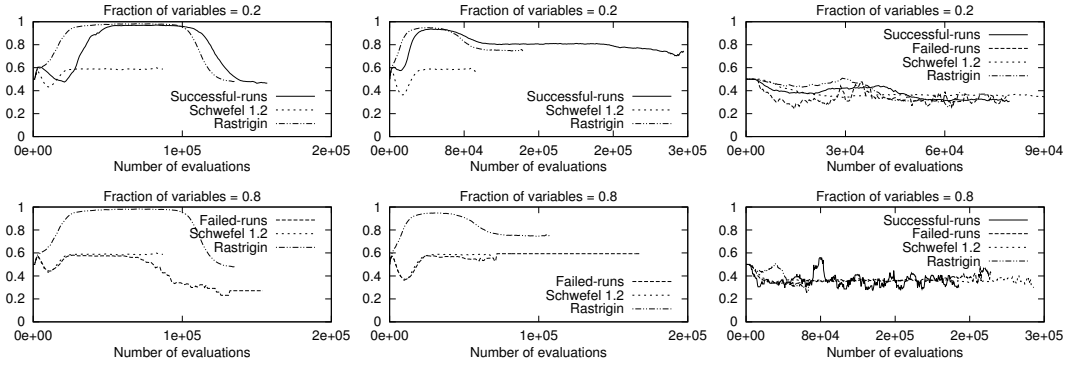
図 7.3, 7.4 に, $r = 0.2, 0.8$ とした 30 次元の $H_{\mathrm{Sch,Ras}}$ における SHADE, JADE, jDE の C, F 値のパラメータ適応過程をそれぞれ示す．ここで, 各手法ごとに異なるパラメータ適応機能を有するため, それぞれの適応手法における C, F 値の代表値を表している．SHADE に関しては, 世代ごとのメモリ $\boldsymbol{M}^F = (M_1^F, ..., M_H^F)$, $\boldsymbol{M}^C = (M_1^C, ..., M_H^C)$ の中央値を示している．JADE については, μ_C, μ_F の世代ごとの値を示している．また, jDE では, 集団中の各個体 $\boldsymbol{x}^{i,t}$, $i \in \{1, ..., N\}$ に割り当てられている $C_{i,t}, F_{i,t}$ の中央値を図に示している．なお, 全ての収束曲線は, 最良解の更新がされなくなるまでの適応パラメータの 50 回試行の平均値である．各図中の Success-run は探索成功時, Failed-run は探索失敗時のパラメータ適応過程を意味する．50 試行全てにおいて探索が成功, 又は失敗しか無かった場合はどちらか一方しか図に示していない．また, 比較のため $r = 0.0$, 及び $r = 1.0$ とした場合の $H_{\mathrm{Sch,Ras}}$, つまり Rastrigin 関数と Schwefels 1.2 関数における各手法の探索が成功した際のパラメータ適応過程を示している．

図 7.3, 7.4 から, SHADE, JADE, jDE は Schwefels 1.2 関数では C は高い値に, F は 0.5 付近の値に適応していることがわかる．一方, Rastrigin 関数では C は低く, F は高い値に収束している．このことから, Schwefels 1.2 関数と Rastrigin 関数では, 適応 DE は両ベンチマー



(a) SHADE　(b) JADE　(c) jDE

図 7.3: 30 次元の $H_{\mathrm{Sch,Ras}}$ における SHADE, JADE, jDE の C 値のパラメータ適応過程. $r = 0.2, 0.8$ における結果をそれぞれ上段, 下段に示している. 全ての収束曲線は, 最良解の更新がされなくなるまでの適応パラメータの 50 試行の平均値である. 各図中の Success-run は探索成功時, Failed-run は探索失敗時のパラメータ適応過程を意味する. 横軸は評価回数の経過を表しており, 縦軸は各手法の適応手法における C 値の代表値を表している.



(a) SHADE　(b) JADE　(c) jDE

図 7.4: 30 次元の $H_{\mathrm{Sch,Ras}}$ における SHADE, JADE, jDE の F 値のパラメータ適応過程. $r = 0.2, 0.8$ における結果をそれぞれ上段, 下段に示している. 全ての収束曲線は, 最良解の更新がされなくなるまでの適応パラメータの 50 試行の平均値である. 各図中の Success-run は探索成功時, Failed-run は探索失敗時のパラメータ適応過程を意味する. 横軸は評価回数の経過を表しており, 縦軸は各手法の適応手法における F 値の代表値を表している.

ク関数にて大きく異なるパラメータ設定に適応していると言える. これらの傾向は, これまでの DE のパラメータ設定に関する調査研究 [196, 27] 及び JADE, jDE のパラメータ適応に関する実験結果とほぼ一致する [27, 267].

次に, $r = 0.2$ における $H_{\mathrm{Sch,Ras}}$, つまり Rastrigin 関数に対して多くの決定変数が割り当てられるハイブリッド関数における適応 DE の振る舞いについて述べる. 図 7.3, 7.4 から,

$r = 0.2$ における $H_{\mathrm{Sch,Ras}}$ と Rastrigin 関数における SHADE, JADE の C, F のパラメータ適応過程はほぼ一致することがわかる．また, jDE においては $C_{i,t}, F_{i,t}$ の中央値を使用しているためかノイズが大きく, Success-run と Failed-run にて若干の差はあるものの, 基本的には SHADE, JADE と同様に Rastrigin 関数におけるパラメータ適応過程と同様の傾向を示す．一方, $r = 0.8$ における $H_{\mathrm{Sch,Ras}}$, つまり Schwefels 1.2 関数の構成比率の高い場合ではその反対となり, C, F のパラメータ適応過程は Schwefels 1.2 関数でのものと似た傾向を示す．

上記の結果から, 適応 DE をハイブリッド関数に適用した場合, 適応手法は 2 つの構成関数の内, 関数の構成比率 r が高く, 解の改善のしやすい構成関数部分に制御パラメータが適応される傾向があると言える．この傾向は, 図 7.3, 7.4 に示した $H_{\mathrm{Sch,Ras}}$ における結果のみではなく, 本研究に使用した $H_{\mathrm{Sph,Ras}}$, $H_{\mathrm{Sch,Ack}}$, $H_{\mathrm{Ros,Ras}}$, $H_{\mathrm{Ros,Ack}}$ といった構成関数が互いに異なる問題性質を有するハイブリッド関数においても見られた．

7.4.2　適応過程におけるパラメータ系列間の距離に基づく, パラメータ適応の解析方法の提案

7.4.1 節での適応 DE のパラメータ適応過程の視覚に基づく解析では, 構成関数が互いに異なる問題性質を有するハイブリッド関数である $H_{\mathrm{Sch,Ras}}$ において, 適応 DE (jDE, JADE, SHADE) のパラメータ適応過程は 2 つの構成関数の内, 関数の構成比率 r が高く, 解の改善のしやすい構成関数部分に制御パラメータが適応される傾向が見られた．また, この傾向は $H_{\mathrm{Sph,Ras}}$, $H_{\mathrm{Sch,Ack}}$, $H_{\mathrm{Ros,Ras}}$, $H_{\mathrm{Ros,Ack}}$ といったハイブリッド関数においても見られる．しかし, 5.4.1 節にて指摘したとおり, 視覚に基づく適応手法の解析法にて得られる情報は定性的であり, この結果に基づき定量的な議論をするのは困難である．そこで, 異なるインスタンスでの適応過程におけるパラメータ系列間の距離の概念を新たに導入し, 各 $r \in [0, 1]$ に対する全てのハイブリッド関数におけるパラメータ適応過程を要約する方法を提案する．

例えば, jDE を Sphere 関数に 2 回適用し, 1 回目は 5 世代, 2 回目は 4 世代目で探索が打ち切られたとする．また, この場合の C の代表値 (7.4.1 節参照) が 1 回目では $(0.7, 0.6, 0.5, 0.7, 0.4)$, 2 回目では $(0.9, 0.8, 0.3, 0.5)$ であったとする．この時, 適応 DE は 1 世代ごとに適応パラメータの更新を行うので, 世代ごとに適応パラメータの 2 回の試行の平均値を独立して求めることができ, この例における C の平均的な適応過程におけるパラメータ系列は $(0.8, 0.7, 0.4, 0.6, 0.4)$ となる．同様に, jDE を Rastrigin 関数に 2 回適用し, C の平均的なパラメータ系列 $(0.4, 0.5, 0.4)$ が得られたとする．この時, 世代ごとの Sphere 関数と Rastrigin 関数における 2 つのパラメータ系列間の距離は $(0.4, 0.2, 0.0)$ であり, 全世代を通じての 2 つのパラメータ系列の平均距離は $(0.4 + 0.2 + 0.0)/3 = 0.2$ である．ここで, 各試行において探索終了までの世代数が異なる場合は, 世代数が小さい方のパラメータ系列に合わせる．Algorithm 20 に, 上記の例を一般化した適応過程におけるパラメータ系列間の距離の算出方法を示す．

この時, 平均距離 $\delta(\overline{\boldsymbol{p}}^{h_1}, \overline{\boldsymbol{p}}^{h_2})$ は異なる 2 つの関数 h_1, h_2 において, 2 つの適応過程におけるパラメータ系列 $\overline{\boldsymbol{p}}^{h_1}, \overline{\boldsymbol{p}}^{h_2}$ がどの程度互いに類似していたかの近似指標となる．$\delta(\overline{\boldsymbol{p}}^{h_1}, \overline{\boldsymbol{p}}^{h_2})$ の

Algorithm 20: 全世代を通じての 2 つのパラメータ系列間の平均距離を計算する関数 $\delta(\overline{\boldsymbol{p}}^{h_1}, \overline{\boldsymbol{p}}^{h_2})$

input : $\overline{\boldsymbol{p}}^{h_1}, \overline{\boldsymbol{p}}^{h_2}$: 関数 h_1, h_2 における適応過程のパラメータ系列の平均
output : δ: $\overline{\boldsymbol{p}}^{h_1}$ と $\overline{\boldsymbol{p}}^{h_2}$ における全世代を通じての 2 つのパラメータ系列間の距離

$t_{\max} \leftarrow \min(|\overline{\boldsymbol{p}}^{h_1}|, |\overline{\boldsymbol{p}}^{h_2}|)$;
$\delta \leftarrow 0$;
for $t = 1$ **to** $t_{\max}$ **do**
    $\delta \leftarrow \delta + |\overline{p}_t^{h_1} - \overline{p}_t^{h_2}|$;
$\delta \leftarrow \delta / t_{\max}$;

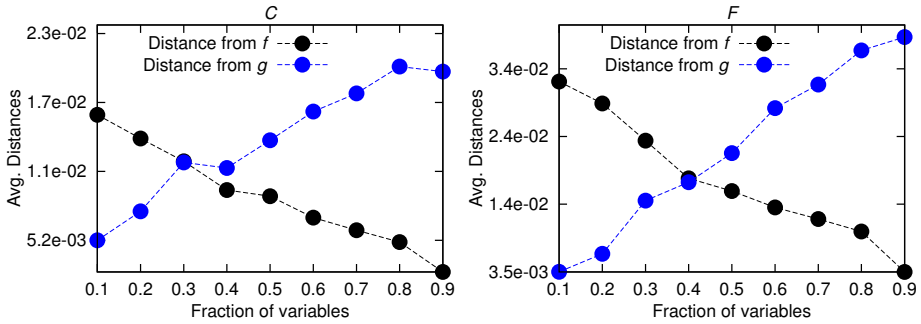
Algorithm 21: 解析対象となる全てのハイブリッド関数における平均距離の算出法 (図 7.5 中のデータの算出法)

input : 解析対象となる全てのハイブリッド関数についての $\overline{\boldsymbol{p}}^{H_{f,g}}$, $\overline{\boldsymbol{p}}^{f}$, $\overline{\boldsymbol{p}}^{g}$
output : $\overline{\delta}_{H_{f,g},f}$, $\overline{\delta}_{H_{f,g},g}$: 解析対象となる全てのハイブリッド関数における平均距離

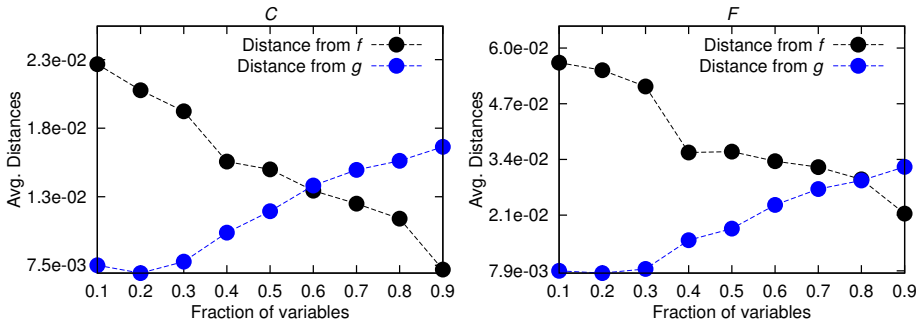
$\overline{\delta}_{H_{f,g},f} \leftarrow 0$;
$\overline{\delta}_{H_{f,g},g} \leftarrow 0$;
for $H_{f,g} \in \boldsymbol{I}$ **do**
    $\overline{\delta}_{H_{f,g},f} \leftarrow \overline{\delta}_{H_{f,g},f} + \delta(\overline{\boldsymbol{p}}^{H_{f,g}}, \overline{\boldsymbol{p}}^{f})$;
    $\overline{\delta}_{H_{f,g},g} \leftarrow \overline{\delta}_{H_{f,g},g} + \delta(\overline{\boldsymbol{p}}^{H_{f,g}}, \overline{\boldsymbol{p}}^{g})$;
$\overline{\delta}_{H_{f,g},f} \leftarrow \overline{\delta}_{H_{f,g},f} / |\boldsymbol{I}|$;
$\overline{\delta}_{H_{f,g},g} \leftarrow \overline{\delta}_{H_{f,g},g} / |\boldsymbol{I}|$;

値が小さいほど 2 つのパラメータ適応過程は互いに似ており, 反対に大きいほど異なることを意味する.

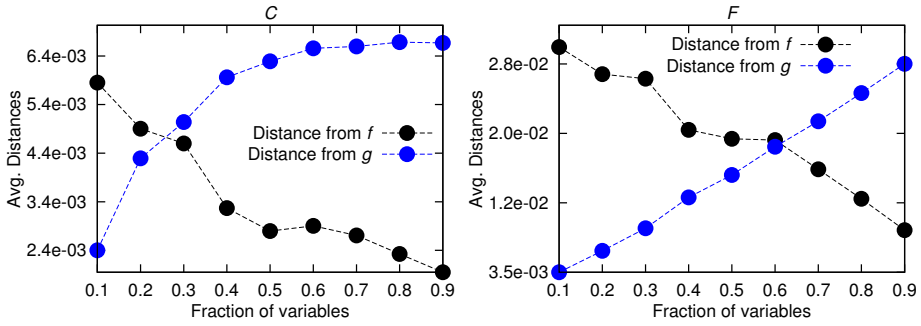
複数のインスタンス集合 $\boldsymbol{I} = (I_1, I_2, ...)$ における平均距離についても同様に求められる. ここで, 本研究の興味は, 特定の r におけるハイブリッド関数 $H_{f,g}$ とその構成関数におけるパラメータ系列の距離 $\delta(H_{f,g}, f)$, 及び $\delta(H_{f,g}, g)$ の全てのインスタンスにおけるパラメータ系列間の平均距離 $\overline{\delta}^r_{H_{f,g},f}$, $\overline{\delta}^r_{H_{f,g},g}$ である. Algorithm 21 に, $\overline{\delta}^r_{H_{f,g},f}$, $\overline{\delta}^r_{H_{f,g},g}$ の算出方法を示す. 例えば, 特定の r において, $\delta(H_{\mathrm{Sph,Ras}}, \mathrm{Sphere}) = 0.4$, 及び $\delta(H_{\mathrm{Sch,Ack}}, \mathrm{Schwefel\ 1.2}) = 0.6$ であったとする. この場合, 距離の平均値は $(0.4 + 0.6)/2 = 0.5$ である. 同様に, $\delta(H_{\mathrm{Sph,Ras}}, \mathrm{Rastrigin}) = 0.6$ であり, $\delta(H_{\mathrm{Sch,Ack}}, \mathrm{Ackley}) = 0.8$ であれば, 距離の平均値は $(0.6 + 0.8)/2 = 0.7$ である. この時, その r 値において, ハイブリッド関数 $H_{f,g}$ と構成関数 f でのパラメータ適応過程は類似している一方, $H_{f,g}$ と構成関数 g でのパラメータ適応過程の類似度は, f でのものと比べて $0.7 - 0.5 = 0.2$ 低いといった定量的な議論が可能である.



(a) SHADE



(b) JADE



(c) jDE

図 7.5: 互いに異なる問題性質を有する構成関数から成るハイブリッド関数 ($H_{\mathrm{Sph,Ras}}$, $H_{\mathrm{Sph,Ack}}$, $H_{\mathrm{Sch,Ras}}$, $H_{\mathrm{Sch,Ack}}$, $H_{\mathrm{Ros,Ras}}$, $H_{\mathrm{Ros,Ack}}$) について, Algorithm 21 を用いて $r \in \{0.1, ..., 0.9\}$ の値ごとに求めた, C, F の適応過程におけるパラメータ系列間の平均距離 $\overline{\delta}^{r}_{H_{f,g},f}$, $\overline{\delta}^{r}_{H_{f,g},g}$. SHADE, JADE, jDE における結果を示している.

7.4.3　ハイブリッド関数における, 異なる適応過程でのパラメータ系列間の距離に基づく解析結果

本節では, 7.4.2 節にて提案した異なるインスタンスでの適応過程におけるパラメータ系列間の距離計測方法を使用して, ハイブリッド関数における適応 DE のパラメータ適応を解析する.

図 7.5 に, 表 7.2 の互いに異なる問題性質を有する構成関数から成るハイブリッド関数

($H_{\mathrm{Sph,Ras}}$, $H_{\mathrm{Sph,Ack}}$, $H_{\mathrm{Sch,Ras}}$, $H_{\mathrm{Sch,Ack}}$, $H_{\mathrm{Ros,Ras}}$, $H_{\mathrm{Ros,Ack}}$) について, Algorithm 21 を用いて $r \in 0.1, ..., 0.9$ の値ごとに求めた, C, F の $\overline{\delta}^{r}_{H_{f,g},f}$, $\overline{\delta}^{r}_{H_{f,g},g}$ を適応手法ごとに示す. なお, 図 7.5 の各 r の結果は, 探索成功, 失敗を問わず, 全ての試行のデータを使用して作成した. 一方, 構成関数 f, g における端的なパラメータ適応過程とハイブリッド関数 $H_{f,g}$ におけるパラメータ適応過程を比較するため, 構成関数 f, g の平均距離については, 50 試行中成功した試行のみを使用している. SHADE と JADE の図では 30, 50 次元の結果に基づいているが, jDE については 50 次元の Schwefel 1.2 関数にて探索成功率が 0 であったため, 30 次元の結果のみを示している.

図 7.5 から, 全ての適応 DE について次のことが言える. r の値が小さい, つまりハイブリッド関数 $H_{f,g}$ が主に構成関数 g から成る場合, その r 値での $H_{f,g}$ と g におけるパラメータ適応過程の平均距離 $\overline{\delta}^{r}_{H_{f,g},g}$ も同様に小さくなる. これは, $H_{f,g}$, 及び g における適応 DE のパラメータ適応過程が, 互いに類似していることを意味する. しかし, r の値が増加するにつれ, $\overline{\delta}^{r}_{H_{f,g},g}$ も単調に増加しており, g におけるパラメータ適応過程との類似性は徐々に低くなっている. 一方, r が増加するほど $\overline{\delta}^{r}_{H_{f,g},f}$ は減少し, 反対に f でのパラメータ適応過程と似るようになる. この傾向は, C, F の両者において確認できる.

上記の結果から, 異なる問題性質を有する 2 つの構成関数 f, g から成るハイブリッド関数 $H_{f,g}$ において, 適応 DE の C, F についてのパラメータ適応過程は, f, 又は g の内, より多くの変数が割り当てられた方の関数でのものと類似すると言える. この結果は, 7.4.1 節における視覚に基づく解析結果と傾向が一致している.

7.4.4　考察

本節では 7.4.1 節, 及び 7.4.3 節にて得られた解析結果について考察する. 6 章で提案した SHADE, 及び JADE, jDE などの既存の適応 DE (4.4.2 節参照) の多くは, 世代 t において親個体 $\boldsymbol{x}^{i,t}$ よりも優れた子個体 $\boldsymbol{u}^{i,t}$ を生成できた, つまり $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$ となったパラメータ設定 $\boldsymbol{s}^{i,t} = \{C_{i,t}, F_{i,t}\}$ を, 対象問題に適した有用な設定と見なし, 何らかの方法に基づき次世代 $t+1$ の C, F の生成に反映させる. つまり, 適応 DE の適応手法は 「対象問題, 及び現在の探索状況に適したパラメータ設定を使用したために, 優れた子個体を生成することができた」という近視眼的な仮定に基づいている. 本章にて扱った互いに異なる問題性質を有する構成関数から成るハイブリッド関数においては, この特徴のために適応 DE の探索, 及びパラメータ適応が失敗すると考えられる.

例えば, 2 つの構成関数 f, g から成るハイブリッド関数 $H_{f,g}$ において, f の方がより多くの決定変数が割り当てられているとする[*4]. 解 $\boldsymbol{x}$ の目的関数値 $H_{f,g}(\boldsymbol{x})$ は f, g の目的関数値の合計となるため (7.2.2 節参照), より多くの決定変数が割り当てられた f の方が, 割り当てられる決定変数が少ない g よりも, 目的関数値の改善に貢献しやすい[*5]. 本章の解析結果から, 適応

[*4] ここでは例として, f の方が g よりも多くの決定変数が割り当てられているとしているが, 反対の場合においても同様のことが言える.

[*5] 式 (7.1) にて重み w_g を w_f よりも十分に大きく取った場合, この限りではない.

DE の適応手法は f における性能を向上させようと, f に適したパラメータ値に素早く適応する傾向がある一方, g におけるパラメータ適応は無視されやすいと言える. この場合, f に属する決定変数を最適化した後, 適応 DE は g に適したパラメータ値へと適応することに失敗する可能性がある.

この傾向は, 7.3.2 節において述べた, 図 7.1(c) にて観測された適応 DE の特徴的なパラメータ適応の振る舞いを説明している. $H_{\mathrm{Sch,Ras}}$ にて, SHADE, JADE, jDE は $r = 0.2$ の時は成功率 1, つまり全ての探索にて最適解を求めることに成功している一方, $r = 0.8$ の時は成功率は 0 に近い. これは, 先述の適応 DE の近視眼的なパラメータ適応の性質のためであると考えられる. $H_{\mathrm{Sch,Ras}}$ の構成関数である Schwefel 1.2 関数と Rastrigin 関数は, 問題性質が互いに大きく異なる. Schwefel 1.2 は単峰性かつ変数分離不可な関数であり, Rastrigin 関数は多峰性かつ変数分離可な性質を有する. また, 先述の通りそれぞれの関数において適切なパラメータ設定は大きく異なる. 実際, 図 7.3, 7.4 では適応 DE は大きく異なるパラメータ適応過程をそれぞれの関数にて示している.

図 7.3, 7.4 において, $r = 0.8$ の時, 適応 DE は単峰性かつ変数分離不可な Schwefel 1.2 関数に適したパラメータ値に適応している. これは, 一方の Rastrigin 関数に対しては不適切なパラメータ値へと誤って適応していることを意味する. 実際, Schwefel 1.2 関数に適したパラメータ設定は局所探索能力が強いため, 多峰性かつ変数分離可な Rastrigin 関数に同じ設定を適用した場合, 無数に存在する局所解に探索が陥りやすいため不適切である. 同様の結論が, 図 7.1(e) の $H_{\mathrm{Ros,Ras}}$ の結果についても当てはまる. Rosenbrock 関数と Rastrigin 関数は両者とも多峰性関数であるが, Rosenbrock 関数は変数分離不可な関数であり, C を比較的高い値に設定しなければ, 効率的な探索が望めない. しかし, Rastrigin 関数では低い C 値が適切であるため, 片方の構成関数に適切なパラメータ値へと適応しようとすると, 残りの構成関数へのパラメータ適応が阻害されてしまう.

一般的に, Rastrigin 関数, 及び Ackley 関数といった多峰性関数から成るハイブリッド関数は, 割り当てられる決定変数が少ない場合, 適応 DE は多峰性関数に適切なパラメータ適応ができないために局所解に探索が陥り, 結果としてハイブリッド関数全体における探索成功率は低下すると考えられる. 一方, $r = 0.2$, つまり Rastrigin 関数, 及び Ackley 関数に割り当てられている決定変数が多い場合において探索成功率が高いのは, 多峰性関数へのパラメータ適応は Sphere 関数や Schwefel 1.2 関数などの単峰性関数への収束速度を遅らせる原因となり得るが, 単峰性関数を探索する能力を完全には損なわないためであると考えられる.

7.5　おわりに

本章では, ハイブリッド関数 [231, 132, 136] における適応 DE のパラメータ適応の振る舞いを解析した. 解析対象となる適応 DE には, 5 章にて比較的良好な性能を示した jDE [27] (4.5.1 節参照), JADE [267] (4.5.3 節参照), 及び, 6 章にて提案した SHADE を使用した. 始めに 3 つの適応 DE をハイブリッド関数に適用した結果, 問題性質が互いに異なるベンチマーク関数から構成されたハイブリッド関数は, 適応 DE にとって困難な問題であることがわかっ

た．次に，この適応 DE の探索失敗現象を解明するため，適応手法のパラメータ適応過程を解析した．解析結果から，互いに異なる問題性質を有する構成関数から成るハイブリッド関数に適応 DE を適用した場合，特定の構成比率 r の下では，どちらかの構成関数に適したパラメータ設定に適応してしまい，残りの関数に対しては不適切なパラメータ設定となり，探索が失敗することがわかった．

本章の主な成果は，互いに異なる問題性質を有する構成関数から成るハイブリッド関数において，適応 DE の探索が失敗する現象を，パラメータ適応の振る舞いを解析することによって明らかにしたことである．この知見は，現存する適応 DE は対象問題の性質の一様性を暗黙に仮定しており，非一様な問題性質を有する問題には対応できない可能性を示唆している．この結果から，各次元 $j \in \{1, ..., D\}$ について $\{C_1, ..., C_D\}$，及び $\{F_1, ..., F_D\}$ のように，C, F を決定変数ごとに割り当てるといった，これまでの適応 DE とは異なる適応手法が有望であると思われる．また，これまでに決定変数間の依存関係を明示的に特定し，探索に利用する DE の交叉手法が提案されている [264, 34]．このような交叉手法は，ハイブリッド関数においても有望である可能性が高い．これらを確かめることが，今後の課題としてあげられる．

また，5 章でも述べたとおり，適応 DE の適応手法の振る舞いを調査した先行研究 [27, 30, 189, 267, 157, 36] のほとんどが視覚に基づく解析である．それに対して 7.4.2 節では，異なる関数インスタンス間の適応過程におけるパラメータ系列間の距離を計測し要約する方法を提案し，ハイブリッド関数において適応 DE が探索に失敗する現象をこの提案手法を用いて明らかにした．この距離に基づくパラメータ適応の解析方法の提案が，本章の副次的な成果であると考えられる．本提案解析法はハイブリッド関数のみではなく，様々な種類の最適化問題における適応 DE の適応手法の解析に使用可能であると思われ，多目的最適化問題などにおける適応 DE のパラメータ適応の振る舞いを解析するのに有用であると考えられる．これを確かめることも，今後の課題として残される．

第 8 章

DE における交叉オペレータの分析と評価

5 章と 7 章では, 適応 DE の適応手法の解析を行った. また, 6 章では新たな適応手法である Success-History based Adaptive DE (SHADE) を提案した. これらの章は適応手法を興味の対象としていたのに対して, 本章では DE における交叉オペレータを扱う.

3.1.2 節, 及び 3.2.2 節にて説明したように, binomial 交叉と exponential 交叉は DE において一般的に使用されている交叉手法である. DE が提案された当初は binomial 交叉の方が優れているとされていたが, 近年では exponential 交叉の有用性を報告する研究が増加しており, どちらの交叉手法が有用であるか明確でない.

本章ではまず, exponential 交叉は Rosenbrock 関数などの人工的に作成したベンチマーク関数に存在する, 依存関係にある決定変数が変数番号上において隣接しているという非現実的な性質を利用可能である点を指摘する. 本論文が対象とする black-box optimization 環境では依存関係にある変数同士が必ず隣接しているという仮定を設けることは不適切であり, この非現実的な性質を解消した場合, exponential 交叉の探索性能は binomial 交叉と比べ劣るようになることを示す. また, バイアスを持たない Shuffled Exponential Crossover (SEC) を含む大規模な交叉オペレータの実験的評価を行い, (1) 多くのベンチマーク関数において binomial 交叉が最も良好な性能を有し, (2) SEC は exponential 交叉と比べて同等以上の性能を持つことを示す.

なお, 本章は著者が第一著者である先行研究 [228, 276] に基づいている.

8.1 はじめに

DE では突然変異と交叉を用いて新たな解を生成するため, 探索性能はこれらのオペレータの選択に大きく依存する [161]. DE の代表的な交叉手法にはそれぞれ GA における一様交叉と 1, 2 点交叉 (2.4.1 節参照) に類似した, binomial 交叉と exponential 交叉がある [220] (3.1 節参照). 両者の相違点は, 遺伝される変数の数の分布, 及び遺伝される決定変数が変数番号上

において連続的か否かである [140][*1].

GA における 1, 2 点交叉は, 変数番号上において隣接している決定変数同士は子個体に共に遺伝されやすい一方, 互いに離れている決定変数同士が共に子個体に受け継がれる可能性は低いという, 決定変数の位置に関するバイアスを持つことが, 1989 年に Caruana らに指摘されている [35]. そのため, 例えば解 $\boldsymbol{x} = (x_1, ..., x_D)^T$ において, 依存関係にある決定変数が x_1 と x_2 のように変数番号上で隣接している問題においては, 依存関係にある決定変数同士を同時に子個体に継承する可能性が高いため, 一様交叉よりも 1, 2 点交叉は有用であることが期待される. 一方, 依存関係にある決定変数が変数番号上で隣接していない問題においては, この限りではない. このバイアスは GA の探索性能に好ましくない影響を与えるため, 近年では GA を利用する際に 1, 2 点交叉が使われることは少なくなっている. 当然ながら, 1, 2 点交叉に類似した exponential 交叉にもこのバイアスが存在することが指摘されている [188]. しかし, このことは特に問題視されていない.

一般的に, 本論文が取り扱う対象問題の性質を事前に知ることができない black-box optimization 環境では, どの決定変数同士が依存関係にあるのかは不明である[*2]. そのため, DE においては遺伝される決定変数が変数番号上においてランダムに選択される binomial 交叉の方が, exponential 交叉よりも優れていることが予想される. 実際に, 比較的初期の研究では, binomial 交叉は exponential 交叉と比べ優れていると報告する研究が多かった [220, 161]. また, 2005 年 ~2010 年頃までの DE に関する研究においても, 多くが binomial 交叉を使用している [196, 27, 28, 257, 267, 189]. しかし, 直感に反して, 近年では exponential 交叉の有用性を報告する研究が数多く存在する [99, 29, 269, 128, 268]. 特に, 高次元最適化問題においては exponential 交叉の方が binomial 交叉よりも優れていると報告する研究が増加している [99, 268].

ここで, 2.5.1 節にて説明したとおり, 関数最適化問題に対する EA の探索性能は, 一般的にベンチマーク関数にて実行することにより評価される. しかし, 2.5.2 節にて述べたように, 人工的に作成したベンチマーク関数を使用した実験的評価では EA の性能を誤評価する危険性がある [67, 203, 252, 231, 138, 154]. 最も大きな問題点は, 最適解が全ての決定変数において同じ値を取る, 最適解が原点 $(0, ..., 0)$ と一致する, 最適解が探索範囲の中心付近に存在するなど, 人工的なベンチマーク関数はしばしば実問題には存在しない性質を暗に有することである. その結果, これらの非現実的な性質を利用する手法は過大評価されてしまう恐れがある.

本章では exponential 交叉は Rosenbrock 関数などの人工的に作成した関数に存在する, 依存関係にある決定変数同士が隣接しているという, 非現実的な性質を利用可能であることを指摘する. そして, この性質を解消した場合, exponential 交叉の性能は大幅に劣るようになることを実験的に示す. また, バイアスを持たない Shuffled Exponential Crossover (SEC) は, exponential 交叉と比べ同等以上の性能を持つことを示す. ここで, SEC は Price らに簡潔

[*1] Binomial 交叉と exponential 交叉のより詳細な相違点については, 3.2.2 節, 及び 3.3.2 節を参照されたい.

[*2] いくつかの決定変数に同時に摂動を加えその解の目的関数値を計測することにより, 依存関係にある決定変数を自動的に検出する方法はこれまでに提案されているが [177, 234, 172], 決定変数同士の依存関係を把握するためには相応の解評価を繰り返す必要があるため, 解評価コストが高価な場合は現実的なアプローチではない.

に提案された, shuffle crossover [35] の仕組みを exponential 交叉に導入した交叉手法である [188]. exponential 交叉は DE の枠組みが提案された 1995 年当初から使用されている交叉手法であるが [220], 本章にて得られた実験結果から, 今後はバイアスを解消した SEC を使用することを推奨する.

本章の構成は次の通りである. 始めに 8.2 節では, 依存関係にある変数が隣接している非現実的な関数について説明する. 8.3 節にて adjacent 関数と distributed 関数における exponential 交叉の性能を評価する. shuffled exponential 交叉の説明と評価は 8.4 節にて行い, exponential 交叉の過大評価については 8.5 節でまとめる. 最後に 8.6 節にて本章をまとめる.

8.2　依存関係にある変数同士が隣接している非現実的なベンチマーク関数

8.2.1　問題提起

EA の探索性能を評価するには, Sphere 関数, Rastrigin 関数, Rosenbrock 関数などの人工的に作成したベンチマーク関数を用いた実験的評価が一般的である. これらのベンチマーク関数は, 単峰性/多峰性, 変数分離可/変数分離不可, 悪スケール性などの実問題に現れうる問題性質を端的に有するように設計されている. しかし, 人工的に作成した関数は, EA の性能を誤評価する危険性があることが指摘されている [67, 203, 252, 231, 138, 154]. 2.5.2 節にて述べたように, 人工的なベンチマーク関数はしばしば実問題には存在しない性質を持ち, この非現実的な性質を利用する手法は過大評価されてしまう恐れがある.

2.5.2 節にて説明した, 最適値 $f(\boldsymbol{x}^*)$ が 0, 規則的に局所解が存在するなどの実問題には存在しない非現実的な問題性質以外にも, いくつかの変数分離不可な関数に見られる「依存関係にある変数同士が隣接している」という性質が挙げられる. 例えば, 最も頻繁に使用されている変数分離不可な関数である Rosenbrock 関数 $f(\boldsymbol{x}) = \sum_{i=1}^{D-1}\left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$ では, 解 $\boldsymbol{x} = (x_1, ..., x_D)^T$ において隣接している変数 x_i と x_{i+1}, $i \in \{1, ..., D-1\}$ にのみ依存関係が存在する. しかし, 実問題においては依存関係にある変数同士が必ず隣接しているという保証は無く, 人工的に作成した関数において頻繁に見られる問題性質である.

ただし, Bouzarkouna らが例として取り上げているように, 当然ながら依存関係にある変数同士が隣接している実問題は存在すると考えられる [24]. 例えば, GECCO2014 にて開催されたコンペティション[*3]でも取り上げられた, Wind Farm Layouts Optimization Problem [253] では与えられた複数の風車の設置位置が x-y 座標系で与えられる. 本問題の解は i 番目の風車の座標を (x_i, y_i) とすると, $(x_1, y_1, x_2, y_2, x_3, y_3, ...)$ となり, 明らかに依存関係にある変数 (x_i, y_i) が隣接している. この際, この問題性質を明示的に利用する探索手法を設計することにより, 高い質の解が得られることが期待できる. しかし, 本論文が取り扱う対象問題の性

[*3] http://www.sigevo.org/gecco-2014/competitions.html (2016 年 1 月 25 日確認)

質を事前に知ることができない black-box optimization 環境では, 依存関係にある決定変数同士が隣接している性質を仮定することは好ましくない.

8.2.2 依存関係にある変数同士が隣接していないベンチマーク関数の作成方法

8.2.1 節にて述べた, 依存関係にある変数同士が隣接しているという性質を持つベンチマーク関数は, DE の exponential 交叉を用いることで効率的に探索することができる. これは, 3.1.2 節, 及び 3.2.2 節にて説明したように, exponential 交叉は変数番号が連続した複数の決定変数を子個体に受け継がせやすく, 決定変数同士の依存関係を破壊すること無く探索が行えるからである. 以上をまとめると, exponential 交叉を用いた DE アルゴリズムを Rosenbrock 関数のように依存関係にある変数同士が隣接しているベンチマーク関数にて性能評価した場合, DE アルゴリズムを誤って実際以上に高い探索性能を持つと結論づけてしまう恐れがある.

そのため, 公平な性能評価実験のために, 依存関係にある決定変数同士が隣接している問題性質をベンチマーク関数から除外する必要がある. この問題点は, CEC2010 LSGO benchmarks [231] のベンチマーク関数の設計に使用された[*4], 例えば $D = 5$ について $(1, 2, 3, 4, 5)$ という変数番号順列を, $(3, 5, 1, 4, 2)$ のようにランダムに並び変えた順列を使用することで容易に解消できる.

以下では, 依存関係にある決定変数同士が隣接していないベンチマーク関数の作成方法について述べる. 始めに, $\boldsymbol{P} = (P_1, ..., P_D)$ を, 変数番号 $\{1, ..., D\}$ がランダムな順序で全て保持されている順列とする. この順列 $\boldsymbol{P}$ を使用して, $\boldsymbol{x}' = (x_{P_1}, ..., x_{P_D})^T$ のように解 $\boldsymbol{x}$ の変数番号をランダムに入れ替える. 例えば, $D = 5$ の問題において, $\boldsymbol{P} = (3, 5, 1, 4, 2)$ の場合, 解 $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5)^T$ は $\boldsymbol{x}' = (x_3, x_5, x_1, x_4, x_2)^T$ となる. そして, 変換された $\boldsymbol{x}'$ を $\boldsymbol{x}$ の代わりに任意のベンチマーク関数の目的関数 f を用いて評価し, $f(\boldsymbol{x}')$ を $f(\boldsymbol{x})$ として与える. 以上の一連の変換操作を使用した場合, 依存関係にある変数同士が隣接しているという問題性質は解消され, この性質を探索に利用することはほぼ不可能となる.

8.3 Adjacent 関数, 及び Distributed 関数における Exponential 交叉の性能評価

8.3.1 実験設定

本節では exponential 交叉を, (1) 依存関係にある変数同士が変数番号上で隣接している関数, (2) 8.2.2 節で述べた決定変数の並び替え法を用いて設計した, 依存関係にある変数同士がランダムに変数番号上に配置してある関数の 2 通りにおいて性能評価をする. 以下では

[*4] CEC2010 LSGO benchmarks [231] の多くのベンチマーク関数は, 部分的に分解可能な性質 (2.2.4 節参照) を有するように設計されている. この際, 依存関係にある変数同士が変数番号上で分散するように, 本節で述べる並び替える手順を導入している.

表 8.1: 8 章の DE アルゴリズムの性能評価実験に使用した, 3 つの変数分離不可なベンチマーク関数. $\boldsymbol{z} = (z_1, ..., z_D)^T$ については本文を参照されたい.

関数名	定義	実行可能領域
Rosenbrock 関数	$f(\boldsymbol{x}) = \sum_{i=1}^{D-1} \left(100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2\right)$	$[-30, 30]^D$
Schwefels 1.2 関数	$f(\boldsymbol{x}) = \sum_{i=1}^{D} (\sum_{j=1}^{i} z_j)^2$	$[-100, 100]^D$
Block-rotated Ellipsoid 関数	$f(\boldsymbol{x}) = \sum_{i=1}^{D-1} \sum_{j=1}^{2} \left(\alpha^{j-1} \left(\boldsymbol{R}^i \cdot (z_i, z_{i+1})\right)\right)^2$	$[-5, 5]^D$

Caruana らの分類 [35] に従い, (1) の関数クラスを adjacent 関数, (2) を distributed 関数と呼ぶ.

実験には, 表 8.1 に示す 3 つの変数分離不可な関数を使用した. Rosenbrock 関数と Schwefels 1.2 関数は, 変数分離不可な関数における EA の性能評価のために古くから頻繁に使用されているベンチマーク関数である [260]. Block-rotated Ellipsoid 関数 [199, 24] は, 明示的に z_i と z_{i+1} にのみ依存関係があるよう設計された関数である. ここで, $\boldsymbol{z} = (y_{P_1}, y_{P_2}, ..., y_{P_{D-1}}, y_{P_D})^T$, $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{o}$ であり, 各関数の最適解の位置を $\boldsymbol{o} = (o_1, ..., o_D)^T$ だけシフトさせている. $\boldsymbol{o}$ は各次元 $\{1, ..., D\}$ において探索範囲内に一様ランダムに生成した実数値ベクトルである. いずれの関数も, $\boldsymbol{z} = (0, ..., 0)^T$ にて最適値 $f_{\text{best}} = 0$ を取る. $\boldsymbol{P} = (P_1, ..., P_D)$ は, 各要素が $\{1, ..., D\}$ の値を取る任意の並び順の順列である. ここで, adjacent 関数では $\boldsymbol{P} = (1, 2, ..., D-1, D)$ であり, distributed 関数では $\boldsymbol{P} = (6, 1, 3, ...)$ のように 8.2.2 にて説明したとおりランダムに並び替えた順列を使用した. $\boldsymbol{R}^i$ は [203] にて提案された方法により一様ランダムに生成した 2×2 の回転行列である. なお, 全ての試行において同じ $\boldsymbol{P}$, $\boldsymbol{R}^i$, $\boldsymbol{o}$ を使用した.

次元数 D は 10, 20, 30, 40, 50, 60, 70, 80 次元とした. 各問題について, 50 回の独立した試行を行った. 各試行において, 探索中に得られた最良解の目的関数値 $f(\boldsymbol{x}^{bsf})$ と最適値 $f(\boldsymbol{x}^*)$ との誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ が 10^{-8} 以下になった場合, その探索を成功として扱う. 反対に評価回数が 2.0×10^6 を超えた場合は, その探索は失敗とし, 打ち切った.

本章の探索手法の性能評価指標には, Success Performance 1 (SP1) [94] と呼ばれる[*5], 成功した探索における評価回数の平均を成功率で割った値を, 探索手法の性能評価指標とした. CEC2005 Competition on Real-Parameter Optimization における手法間の順位付けにも SP1 は使用されている[*6]. SP1 は対象問題において任意の許容精度の解を求めるまでに費やした計算資源 (ここでは解評価回数) の期待値を表し, 本指標が低い手法ほど, 素早く, かつ安定して高精度の解を求めることができていることを意味する.

実験には 3.1.2 節にて述べた通常の DE (Algorithm 9 参照), 及び 4.5 節で説明した jDE

[*5] [10] にて言及されているとおり, ENES [187], 及び Q-measure [61, 250] と呼ばれる性能評価指標と SP1 は等価である.

[*6] https://www.lri.fr/~hansen/cec2005.html (2016 年 1 月 25 日確認)

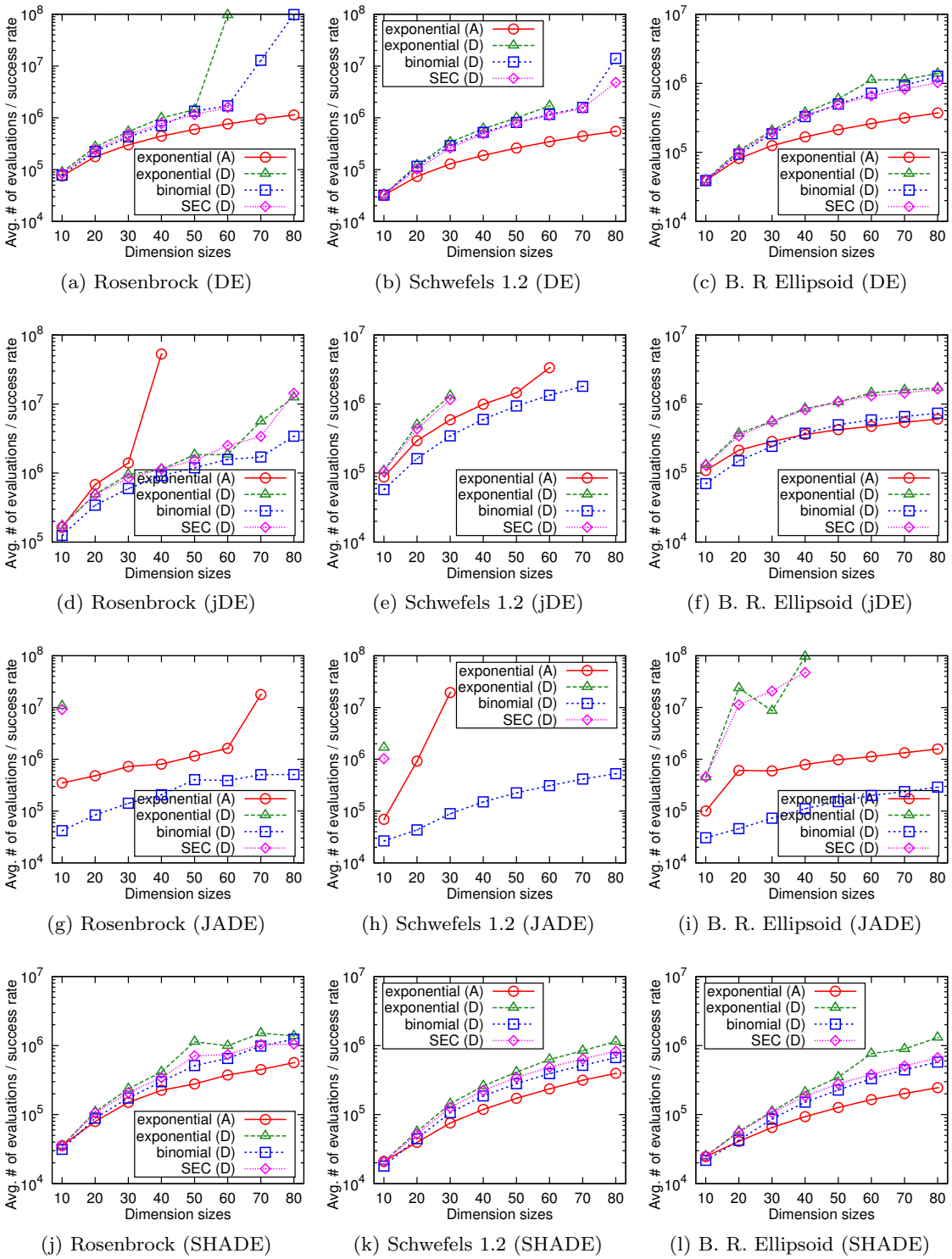
[27] (Algorithm 11 参照), JADE [267] (Algorithm 13 参照) といった適応 DE アルゴリズムを使用した. さらに, 6 章にて提案した SHADE も用いた. ここで, jDE [27], JADE [267] は, binomial 交叉を使用することを前提に設計された F, C を自動調整する適応 DE であるため, exponential 交叉を使用した場合は適応手法が機能しない恐れがあることに注意されたい. 通常の DE のパラメータ設定について, 集団数 $N = 100$, $F = 0.5$ とした [27, 267]. 突然変異戦略には, 表 3.1 の中でも最も基本的な rand/1 を使用した. 交叉には exponential 交叉の他に, 比較のために binomial 交叉も評価した. C については, 各交叉手法, 各ベンチマーク関数, 及び各次元ごとに 0.90 から 0.01 刻みで 0.99 まで値を変えて実験を行った. そして, 本実験の性能評価指標である SP1 が最も低い値を取るデータを使用した. jDE, JADE の制御パラメータについては, 各論文 [27, 267] に示されている推奨パラメータ設定を使用した. すなわち, 集団数 $N = 100$ とし, jDE については $\tau_F = 0.1$, $\tau_C = 0.1$, JADE については $c = 0.1$, current-to-$pbest$/1 の p, A については, $p = 0.05, A = N$ とした. SHADE については, JADE と同様に current-to-$p$best/1 を使用した. その他の設定は, 及び 6 章のパラメータ設定と同様である.

8.3.2 実験結果と考察

各 DE アルゴリズムの実験結果を図 8.1 に示す. shuffled exponential 交叉 (SEC) については 8.4 節にて説明する. また, binomial 交叉, 及び SEC については adjacent, distributed 関数において性能の変化はほとんど見られなかったため, 後者の関数クラスにおける結果のみを示している.

通常の DE (図 8.1(a)(b)(c)) では, 全ての次元数における adjacent 関数において, exponential 交叉は binomial 交叉よりも優れている. 対照的に, distributed 関数では全ての次元数, 及びベンチマーク関数において, exponential 交叉の性能は大幅に下がり, binomial 交叉よりも劣っている. その性能差は, 次元数が増加するほど拡がっている. 特に, distributed-Rosenbrock 関数と distributed-Schwefels 1.2 関数では, 70 次元以上では exponential 交叉は全ての試行において探索が失敗している. 制御パラメータの自動調整を行う適応 DE においても, jDE の Rosenbrock 関数 (図 8.1(d)) における結果を除き, 同様の傾向が見られる.

以上の結果より, Rosenbrock 関数や Schwefels 1.2 関数などの変数分離不可な関数における exponential 交叉の性能は, 依存関係にある変数の変数番号上での並び順に大きく影響すると言える. その結果, 本質的には同一の関数であるにも関わらず, 依存関係にある変数が隣接していない場合, exponential 交叉の性能は binomial 交叉よりも劣るようになる. 本実験で使用した Rosenbrock 関数と Schwefels 1.2 関数は, DE の提案以来から今日に至るまで, 頻繁に使用されている [192, 99, 29, 269, 128, 268]. そのため, exponential 交叉の性能はこれまで過大評価されてきたと考えられる.



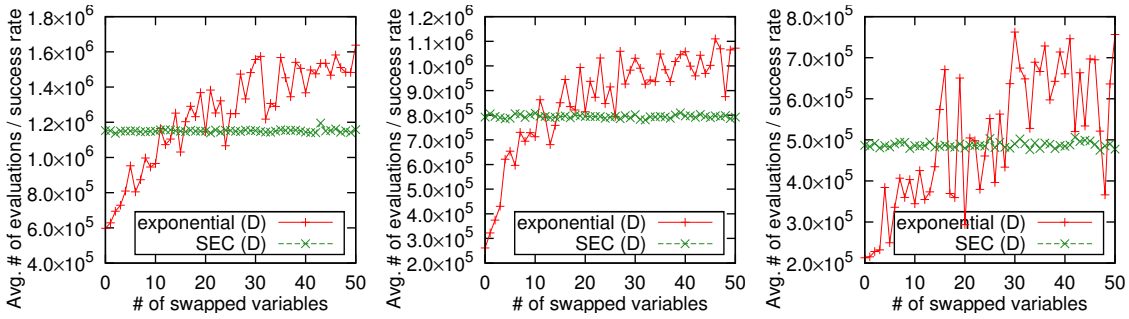
(a) Rosenbrock (DE)　(b) Schwefels 1.2 (DE)　(c) B. R Ellipsoid (DE)

(d) Rosenbrock (jDE)　(e) Schwefels 1.2 (jDE)　(f) B. R. Ellipsoid (jDE)

(g) Rosenbrock (JADE)　(h) Schwefels 1.2 (JADE)　(i) B. R. Ellipsoid (JADE)

(j) Rosenbrock (SHADE)　(k) Schwefels 1.2 (SHADE)　(l) B. R. Ellipsoid (SHADE)

図 8.1: Adjacent 関数, 及び distributed 関数における, 各交叉手法を用いた通常の DE, jDE, JADE, SHADE の比較結果. 図中の (A) と (D) は, それぞれ adjacent 関数と distributed 関数における結果であることを意味する. 横軸は次元数 $D \in \{10, ..., 80\}$, 縦軸は $f_{\text{target}} = 10^{-8}$ とした SP1 (成功した探索における評価回数の平均を成功率で割った値) の値である. なお, 50 回全ての試行において探索が失敗し成功率 p_s が 0 となる場合は, 該当データを削除している.



(a) Rosenbrock (b) Schwefels 1.2 (c) B. R. Ellipsoid

図 8.2: 変数の非隣接性の度合いを変化させた場合における, exponential 交叉と SEC を用いた通常の DE (Algorithm 9) の性能推移. 次元数 D は 50 である. 横軸はランダムに変数を入れ替えた回数 ($n \in \{1, ..., D\}$), 縦軸は $f_{\text{target}} = 10^{-8}$ とした SP1 (成功した探索における評価回数の平均を成功率で割った値) の値である. なお, 50 回全ての試行において探索が失敗し, 成功率が 0 となる場合は該当データを削除している.

8.3.3 変数の並び順が Exponential 交叉に与える影響

8.3.2 節では, exponential 交叉の探索性能が distributed 関数において悪化することを示した. しかし, 依存関係にある決定変数の配置が exponential 交叉にどの程度の影響を与えるのかは定かではない. そこで, 本節では決定変数の非隣接性の度合いを変化させた場合における, exponential 交叉の探索性能の推移を調査することにした. 具体的には表 8.1 のベンチマーク関数における順列 $\boldsymbol{P}$ について, $\{1, ..., D\}$ をランダムに選択した 2 つの変数番号を入れ替えるという操作を $n \in \{1, ..., D\}$ 回行うことで, 決定変数の非隣接性の度合いを調整した. この方法では n が大きくなるほど元の変数番号は保持されづらくなり, 依存関係にある変数が隣接している数は少なくなる[*7]. DE には通常の DE を使用し, パラメータ設定は 8.3.1 節と同様の設定を使用した. 各 n の値について, 最良の C を使用した. また, 関数の次元数は 50 とした.

図 8.2 に実験結果を示す. 図から, 全ての関数において exponential 交叉は $n = 0$, つまり adjacent 関数の場合が最も性能が良い. しかし, n が増加するに伴い性能は劣るようになる. また, 図から視認するのは困難であるが, Block-rotated Ellipsoid 関数では n の値によっては成功率が 0 となる場合があった. この結果から, 依存関係にある決定変数の変数番号上での配置に対して, exponential 交叉の性能は非常に敏感であると言える.

[*7] 一様性を保証している Fisher-Yates shuffle 法とは異なり, この方法では一様ランダムな並び替えを実現する有限な n の値は存在しないため, ランダムに順列の要素を並び替える方法としては問題がある. しかし, ここでは決定変数の非隣接性の度合い調整のためだけにこの方法を使用するため, 特に問題ないと考えられる.

8.4　Shuffled Exponential Crossover (SEC) の性能評価

8.3 節の実験結果から, distributed 関数における exponential 交叉の探索性能は乏しいことがわかった. これは決定変数の位置に関するバイアスを exponential 交叉は持つためである. そこで, 本節ではこのバイアスを解消した Shuffled Exponential Crossover (SEC) について述べ, また実験にて distributed 関数における性能が改善されていることを確認する.

8.4.1　Shuffled Exponential Crossover (SEC) と関連研究

Caruana ら [35] に指摘されているように, 1 点交叉では変数番号が互いに近い決定変数同士が子個体に共に遺伝されやすい一方, 互いに離れている決定変数同士が共に子個体に受け継がれる可能性は低い. この傾向は exponential 交叉にも当てはまり, そのために 8.3 節の実験では, distributed 関数において exponential 交叉の探索性能は乏しかったと考えられる. この決定変数の位置に関するバイアスを解消するため, Caruana らは shuffle crossover を提案した [35]. shuffle crossover では, 始めに 2 つの親個体の変数番号をランダムに並び替える. その後, 1 点交叉を適用し, ランダムに並び替えられた変数番号を元の番号順に並び替える. shuffle crossover は一様交叉に類似しているが, 子個体に受け継がれる変数の数の分布が異なる. また, 先に述べた 1 点交叉に内在するバイアスを, shuffle crossover では解消している.

Price らは, 1 点交叉と同様の問題が DE の exponential 交叉においても起こり得るが, 同じように shuffle crossover のメカニズムを適用することで, この問題は解消できると述べている [188]. 本論文ではこの交叉を, Shuffled Exponential Crossover (SEC) と呼ぶ. SEC を Algorithm 22 に示す. ここで, 通常の DE では特に断りの無い限り, $C_{i,t} = C$ と全ての個体で同一の値を取る. SEC では exponential 交叉とは異なり, 遺伝される決定変数が変数番号上において連続的ではない. しかし, 3.3.2 節にて述べた, 受け継がれる決定変数の数 L の期待値 $E(L)$ は, exponential 交叉と同様に式 (3.6) で与えられる性質を持つ.

[188] では exponential 交叉の問題点についての言及と SEC の提案を簡潔にしているが, 実験的な評価は一切していない. Lin らは non-consecutive exponential 交叉という, SEC と同一の交叉手法を提案し, 評価をしている [140]. しかし, 本章のように依存関係にある変数の並び順に関する調査はしていない. 現に, Lin らは「in non-separable ridge functions (Rosenbrock and Schwefels 1.2 function), differential evolution algorithms with consecutive crossover are more reliable than those with non-consecutive crossover」という, 8.3 節で得られた知見に反する, 不正確な結論を述べている. つまり, [188, 140] の両研究においても, 8.2 節にて述べた依存関係にある決定変数同士が隣接している非現実的なベンチマーク関数と exponential 交叉の好ましくない関係については認知されていない.

Algorithm 22: Shuffled Exponential Crossover (SEC)

$\boldsymbol{u}^{i,t} \leftarrow \boldsymbol{x}^{i,t}$;
変数番号 $\{1, ..., D\}$ をランダムに並び替えた順列 $\boldsymbol{S}^{i,t} = (s_1^{i,t}, ..., s_D^{i,t})$ を生成;
$k \leftarrow 1$;
repeat
  $j \leftarrow s_k^{i,t}$;
  $u_j^{i,t} \leftarrow v_j^{i,t}$;
  $k \leftarrow k + 1$;
until $\mathrm{rand}[0, 1) < C_{i,t}$ and $k < D$;

8.4.2 実験結果

図 8.1 に, 通常の DE と 3 つの適応 DE (jDE, JADE, SHADE) における SEC の実験結果を示す. ここでの実験設定は, 8.3 節と同様である. exponential 交叉とは異なり, しかし binomial 交叉とは同様に, SEC の性能は関数が adjacent か distributed であるかに影響されないため, distributed 関数における結果のみを示す. その根拠として, 図 8.2 に, $n = 0$ において最良な C を用いた通常の DE における SEC の実験結果を同様に示す. exponential 交叉とは異なり, SEC の性能は n の値に対して独立であることがわかる.

通常の DE における実験結果を示す図 8.1(a)(b)(c) から, 低次元の distributed 関数において, SEC は exponential 交叉よりもわずかに優れていることがわかる. しかし, 60 次元の distributed Rosenbrock 関数では, SEC は exponential 交叉よりも大幅に優れている. distributed-Schwefels 1.2 関数では, exponential 交叉を用いた場合は全ての試行において探索が失敗する 70 次元以上においてさえ, SEC は binomial 交叉と同程度の性能を示している. 通常の DE における結果と同様に, 適応 DE アルゴリズムにおいても distributed 関数では多くの場合 SEC が exponential 交叉よりも優れている.

以上の結果から, exponential 交叉は対象関数が adjacent か distributed によって性能が大きく変化するのに対して, SEC の性能はそれらに対して影響を受けないと言える. また, distributed 関数では手法に依らず SEC の方が優れている場合が多い. そのため, SEC は exponential 交叉よりも有用な交叉手法であると考えられる.

8.5 Exponential 交叉の過大評価の可能性

8.2.2 節での議論, 及び 8.3 節での実験結果から, DE における exponential 交叉はこれまで過大評価されている可能性が高い. 本節では, まず 8.5.1 節にて代表的なベンチマークセットにおける adjacent 関数を列挙し, 誤評価の可能性が高いベンチマークセットを指摘する. 次に 8.5.2 節にて, adjacent 関数が存在しないと考えられるベンチマークセットである CEC2014

表 8.2: 代表的なベンチマークセットにおける adjacent 関数の数.

ベンチマークセット	adjacent 関数の数	ベンチマーク関数
13 classical [260]	2 / 13	F_3, F_5
CEC2005 [223]	4 / 25	F_2, F_4, F_6, F_{13}
CEC LSGO 2010 [231]	2 / 20	F_{19}, F_{20}
BBOB [90]	1 / 24	F_8
SOCO [100]	10 / 19	$F_3, F_8, F_9, F_{11}, F_{12}, F_{13}, F_{14}, F_{16}, F_{17}, F_{18}$

benchmarks [136] にて binomial 交叉, exponential 交叉, SEC の 3 つの交叉手法の性能評価を行う.

8.5.1　代表的なベンチマークセットにおける Adjacent 関数について

本章での実験結果から, 表 8.1 にあげたような依存関係にある変数同士が隣接しているベンチマーク関数を含むベンチマークセットを用いて評価された exponential 交叉に関する実験結果は, 改めて見直す必要があると言える. しかし, そのような関数には表 8.1 に示される関数以外にも, Schaffer $F7$ 関数, Whitley's composite functions [252] など, 一般的に使用されているベンチマークセットに頻出する関数がある.

表 8.2 に, 代表的なベンチマークセットの構成関数の総数と, その中に含まれる adjacent 関数の数を示す. adjacent 関数の数が多いベンチマークセットは, exponential 交叉のように依存関係にある変数同士が隣接している性質を利用できる手法を過大評価してしまう可能性が高い. 特に, SOCO benchmarks [100] は 19 個の関数の内 10 個が adjacent 関数であることから, 他のベンチマークセットと比べ, この傾向が端的に見られる. 実際, SOCO benchmarks が使用された Soft Computing Journal の高次元最適化問題に対する EA の特集号[*8]に掲載された SOUPDE [251], DE-D^{40} + M$^{\mathrm{m}}$ [73], GODE [246], GaDE [258], jDElscop [29], SaDE-MMTS [269], MOS [128] といった 7 つの DE アルゴリズムは, 全て exponential 交叉を使用している. また, binomial 交叉よりも exponential 交叉の方が優れていると結論づけた Zhao と Suganthan の調査研究 [268] も, SOCO benchmarks を使用している. このように, いくつかの先行研究では adjacent 関数の数が多いベンチマークセットを使用したために, exponential 交叉を過大評価している可能性が高いように考えられる.

[*8] http://sci2s.ugr.es/EAMHCO (2015 年 11 月 10 日確認)

8.5.2 CEC2014 Benchmarks での各交叉手法の比較

8.5.1 節で述べたように, 代表的なベンチマークセットのほとんどが adjacent 関数を含んでおり, exponential 交叉の探索性能を正しく評価することが困難である. そのため, adjacent 関数が存在しないベンチマークセットにおいて, 交叉手法を比較することは興味深いことである. 近年提案された 30 個の関数 $F_1^{\mathrm{cec}} \sim F_{30}^{\mathrm{cec}}$ から構成される CEC2014 benchmarks [136] は, adjacent 関数を含まない. そこで, 通常の DE と, jDE, JADE, SHADE における SEC, exponential 交叉, binomial 交叉 の性能を CEC2014 benchmarks (D =10, 30, 50 次元) にて評価することにした.

実験設定は, CEC2014 にて開催されたコンペティション [136] に従い, 最大評価回数を $D \times 10,000$, 試行回数を 51 とした. 各 DE アルゴリズムのパラメータ設定は, 8.3.1 節での設定と同様とした. また, 探索中に得られた最良解の目的関数値 $f(\boldsymbol{x}^{bsf})$ と最適値 $f(\boldsymbol{x}^*)$ との誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ が 10^{-8} 以下になった場合は, 誤差値は 0 とした. $F_1^{\mathrm{cec}} \sim F_3^{\mathrm{cec}}$ は変数分離不可な単峰性関数, $F_4^{\mathrm{cec}} \sim F_{16}^{\mathrm{cec}}$ は多峰性関数であり $F_8^{\mathrm{cec}}, F_{10}^{\mathrm{cec}}$ のみ変数分離可である, $F_{17}^{\mathrm{cec}} \sim F_{22}^{\mathrm{cec}}$ は決定変数ごとに異なる問題性質を有するよう, 複数のベンチマーク関数を用いた部分的に分解可能なハイブリッド関数 (7 章参照) である. $F_{23}^{\mathrm{cec}} \sim F_{30}^{\mathrm{cec}}$ は $F_1^{\mathrm{cec}} \sim F_{22}^{\mathrm{cec}}$ の中から選択した複数のベンチマーク関数を合成した関数である. また, 実行可能領域は $\boldsymbol{S} = [-100, 100]^D$ である.

表 8.3 8.4, 8.5, 8.6 に, 通常の DE, jDE, JADE, 及び SHADE における SEC と binomial 交叉, exponential 交叉の比較結果を示す. 表から, 30 次元における SHADE の結果を除き, CEC2014 benchmarks においては全ての手法について SEC は exponential 交叉よりも優れていることがわかる. また, SEC と binomial 交叉の比較では, 10 次元における通常の DE の結果を除き, binomial 交叉の方が良い性能を示している. 以上の結果から, black-box optimization 環境においては, 3 種類の交叉手法の中では binomial 交叉が最も良い交叉手法であると考えられる.

一方, 表 8.3, 8.4, 8.5, 8.6 から, CEC2014 benchmarks 全体での性能は binomial 交叉 に劣っているものの, いくつかの関数においては SEC が優れている場合が見られる. このことから, binomial 交叉と SEC といった異なる交叉手法を対象問題に応じて適応的に選択するような戦略を構築することは, 優れた探索手法の実現のためには有望なアプローチであると考えられる. また , 本章では jDE, JADE, SHADE といった適応 DE を評価対象手法として使用したが, SHADE 以外の手法は交叉方法として binomial 交叉を用いることを前提として設計された手法であり, SEC を使用することは本来ならば不適切な可能性がある. そのため, SEC の使用を前提とした開発された新たな適応 DE が, binomial 交叉を使用する jDE, JADE といった手法よりも優れている可能性は十分に考えられる. 実際, 任意の突然変異戦略と交叉手法を組み合わせて使用することを前提に設計した SHADE (6 章参照) は, SEC を用いた場合でも良好な性能を示していた.

表 8.3: 10, 30, 50 次元の CEC2014 benchmarks [136] での, 突然変異戦略に rand/1 を用いた通常の DE アルゴリズム (Algorithm 9) における SEC に対する binomial/exponential 交叉の比較結果. 表中のデータは, 探索中に得られた最良解の目的関数値 $f(\boldsymbol{x}^{bsf})$ と最適値 $f(\boldsymbol{x}^*)$ との誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ の 51 試行における平均値である. 各関数において最良のデータを太字で示している. 誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ の 51 試行分のデータに対して行った, 有意水準 $p = 0.05$ の Wilcoxon の順位和検定を行った. 表中の各関数における記号 $+, \approx, -$ は, 比較交叉手法が SEC と比べてそれぞれ有意に優れている, 有意に劣っている, 有意差無しであったこと表す. また, 表下の 「# of better」, 「# of worse 」 , 「# of no sig. 」の項目は, 比較交叉手法が SEC と比べてそれぞれ有意に優れている, 有意に劣っている, 有意差無しであった関数の総数を表す.

(a) $D = 10$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_2^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_3^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_4^{cec}	**6.48e+00**	9.46e+00≈	7.58e+00≈
F_5^{cec}	1.97e+01	1.96e+01+	**1.91e+01**+
F_6^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_7^{cec}	3.75e-02	3.70e-02≈	**2.91e-02**≈
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	1.14e+01−
F_9^{cec}	**1.18e+01**	1.18e+01≈	1.42e+01≈
F_{10}^{cec}	**4.19e+00**	4.58e+00≈	5.19e+02−
F_{11}^{cec}	**5.54e+02**	5.98e+02≈	8.66e+02−
F_{12}^{cec}	4.71e-01	**4.67e-01**≈	6.55e-01−
F_{13}^{cec}	1.25e-01	1.26e-01≈	**1.22e-01**≈
F_{14}^{cec}	1.43e-01	**1.34e-01**≈	1.51e-01≈
F_{15}^{cec}	1.36e+00	**1.32e+00**≈	1.77e+00−
F_{16}^{cec}	**2.31e+00**	2.33e+00≈	2.51e+00−
F_{17}^{cec}	3.86e-01	**3.58e-01**+	4.88e-01−
F_{18}^{cec}	1.56e-01	1.62e-01−	**1.43e-01**≈
F_{19}^{cec}	**2.94e-01**	3.21e-01≈	3.26e-01−
F_{20}^{cec}	**4.77e-02**	7.42e-02−	8.72e-02≈
F_{21}^{cec}	**2.27e-01**	3.47e-01−	3.09e-01≈
F_{22}^{cec}	**1.16e-01**	1.97e-01−	2.34e-01−
F_{23}^{cec}	**3.23e+02**	3.29e+02≈	3.29e+02≈
F_{24}^{cec}	1.15e+02	1.17e+02−	**1.13e+02**≈
F_{25}^{cec}	1.47e+02	1.44e+02≈	**1.42e+02**≈
F_{26}^{cec}	1.00e+02	1.00e+02≈	**1.00e+02**≈
F_{27}^{cec}	6.06e+01	5.07e+01≈	**4.68e+01**+
F_{28}^{cec}	3.66e+02	**3.63e+02**≈	3.71e+02≈
F_{29}^{cec}	**2.03e+02**	2.04e+02≈	2.11e+02−
F_{30}^{cec}	**4.62e+02**	4.63e+02−	4.64e+02−
	+ (better)	2	2
	− (worse)	6	11
	≈ (no sig.)	22	17

(b) $D = 30$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	6.74e+04	1.05e+05−	**3.29e+04**+
F_2^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_3^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_4^{cec}	**9.38e-02**	4.79e-01−	2.06e-01−
F_5^{cec}	2.05e+01	**2.05e+01**≈	2.07e+01−
F_6^{cec}	5.36e-01	3.85e-01≈	**3.74e-01**+
F_7^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	1.88e+01−
F_9^{cec}	8.81e+01	9.14e+01≈	**3.60e+01**+
F_{10}^{cec}	**2.30e+01**	2.66e+01−	1.16e+03−
F_{11}^{cec}	**3.83e+03**	3.83e+03≈	3.89e+03−
F_{12}^{cec}	**7.04e-01**	7.35e-01≈	7.54e-01−
F_{13}^{cec}	2.89e-01	3.15e-01≈	**2.77e-01**≈
F_{14}^{cec}	**2.54e-01**	2.54e-01≈	2.59e-01≈
F_{15}^{cec}	9.73e+00	9.60e+00≈	**6.64e+00**+
F_{16}^{cec}	**1.09e+01**	1.09e+01≈	1.09e+01≈
F_{17}^{cec}	1.86e+02	1.83e+02≈	**1.25e+02**≈
F_{18}^{cec}	1.00e+01	1.12e+01≈	**7.78e+00**+
F_{19}^{cec}	3.32e+00	3.40e+00≈	**3.08e+00**≈
F_{20}^{cec}	9.98e+00	9.39e+00≈	**7.58e+00**+
F_{21}^{cec}	9.29e+01	7.98e+01≈	**6.33e+01**+
F_{22}^{cec}	3.32e+01	3.65e+01≈	**2.81e+01**+
F_{23}^{cec}	**3.15e+02**	**3.15e+02**≈	**3.15e+02**≈
F_{24}^{cec}	2.16e+02	**2.15e+02**≈	2.18e+02≈
F_{25}^{cec}	2.03e+02	2.03e+02−	**2.03e+02**≈
F_{26}^{cec}	1.00e+02	1.00e+02≈	**1.00e+02**≈
F_{27}^{cec}	3.64e+02	3.64e+02≈	**3.47e+02**+
F_{28}^{cec}	7.90e+02	**7.85e+02**+	8.00e+02−
F_{29}^{cec}	6.52e+02	6.63e+02≈	**5.72e+02**+
F_{30}^{cec}	5.92e+02	6.27e+02≈	**5.25e+02**+
	+ (better)	1	11
	− (worse)	4	7
	≈ (no sig.)	25	12

(c) $D = 50$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	5.62e+05	6.57e+05≈	**3.85e+05**+
F_2^{cec}	**9.65e+00**	6.73e+02−	2.08e+02≈
F_3^{cec}	**1.30e-05**	3.12e-04−	1.93e-01−
F_4^{cec}	7.50e+01	8.40e+01≈	**7.06e+01**≈
F_5^{cec}	**2.06e+01**	2.06e+01≈	2.08e+01−
F_6^{cec}	2.40e+00	3.05e+00≈	**1.50e+00**+
F_7^{cec}	**0.00e+00**	**0.00e+00**≈	1.45e-04≈
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	3.46e+01−
F_9^{cec}	1.97e+02	2.00e+02≈	**5.85e+01**+
F_{10}^{cec}	**3.49e+01**	4.28e+01−	1.60e+03−
F_{11}^{cec}	7.61e+03	7.69e+03≈	**6.98e+03**+
F_{12}^{cec}	8.13e-01	8.10e-01≈	**5.46e-01**+
F_{13}^{cec}	4.30e-01	4.42e-01≈	**3.96e-01**+
F_{14}^{cec}	**2.75e-01**	2.81e-01≈	3.17e-01−
F_{15}^{cec}	2.14e+01	2.10e+01≈	**9.11e+00**+
F_{16}^{cec}	1.96e+01	1.97e+01≈	**1.93e+01**+
F_{17}^{cec}	1.46e+04	1.65e+04≈	**1.25e+04**≈
F_{18}^{cec}	4.40e+01	4.93e+01≈	**2.35e+01**+
F_{19}^{cec}	**8.15e+00**	8.61e+00≈	9.12e+00−
F_{20}^{cec}	**4.55e+01**	5.83e+01−	4.90e+01≈
F_{21}^{cec}	1.02e+03	1.18e+03≈	**7.16e+02**+
F_{22}^{cec}	**5.09e+02**	5.64e+02≈	5.91e+02≈
F_{23}^{cec}	**3.44e+02**	**3.44e+02**≈	**3.44e+02**≈
F_{24}^{cec}	**2.62e+02**	2.63e+02≈	2.70e+02−
F_{25}^{cec}	2.06e+02	2.06e+02≈	**2.05e+02**+
F_{26}^{cec}	1.00e+02	1.00e+02+	**1.00e+02**+
F_{27}^{cec}	**3.56e+02**	3.74e+02−	3.69e+02−
F_{28}^{cec}	**1.08e+03**	1.10e+03−	1.08e+03≈
F_{29}^{cec}	**8.87e+02**	1.15e+03−	9.13e+02≈
F_{30}^{cec}	**7.93e+03**	7.98e+03−	8.27e+03−
	+ (better)	1	12
	− (worse)	8	9
	≈ (no sig.)	21	9

表 8.4: 10, 30, 50 次元の CEC2014 benchmarks [136] での, jDE (Algorithm 11) における SEC に対する binomial/exponential 交叉の比較結果. 表中のデータは, 探索中に得られた最良解の目的関数値 $f(\boldsymbol{x}^{bsf})$ と最適値 $f(\boldsymbol{x}^*)$ との誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ の 51 試行における平均値である. 各関数において最良のデータを太字で示している. 誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ の 51 試行分のデータに対して行った, 有意水準 $p = 0.05$ の Wilcoxon rank-sum test を行った. 表中の各関数における記号 $+, \approx, -$ は, 比較交叉手法が SEC と比べてそれぞれ有意に優れている, 有意に劣っている, 有意差無しであったこと表す. また, 表下の 「# of better」, 「# of worse 」 , 「# of no sig. 」の項目は, 比較交叉手法が SEC と比べてそれぞれ有意に優れている, 有意に劣っている, 有意差無しであった関数の総数を表す.

(a) $D = 10$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	1.45e-05	1.09e-03≈	**0.00e+00**+
F_2^{cec}	9.28e-07	3.21e-05−	**0.00e+00**+
F_3^{cec}	1.70e-07	2.54e-09+	**0.00e+00**+
F_4^{cec}	3.93e+00	**1.29e+00**≈	1.42e+01≈
F_5^{cec}	1.86e+01	**1.81e+01**≈	1.98e+01−
F_6^{cec}	1.52e-01	1.07e-02+	**3.25e-06**+
F_7^{cec}	2.16e-02	1.90e-02≈	**1.89e-02**≈
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_9^{cec}	5.67e+00	**5.62e+00**≈	5.99e+00≈
F_{10}^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_{11}^{cec}	2.67e+02	**2.48e+02**≈	2.96e+02≈
F_{12}^{cec}	**3.31e-01**	3.37e-01≈	3.87e-01−
F_{13}^{cec}	1.64e-01	1.61e-01≈	**1.42e-01**+
F_{14}^{cec}	1.60e-01	1.66e-01≈	**1.56e-01**≈
F_{15}^{cec}	9.40e-01	**8.92e-01**≈	9.94e-01≈
F_{16}^{cec}	**2.08e+00**	2.08e+00≈	2.12e+00≈
F_{17}^{cec}	4.81e+01	4.87e+01≈	**1.15e+01**+
F_{18}^{cec}	3.22e+00	4.16e+00−	**1.37e+00**+
F_{19}^{cec}	3.49e-01	3.78e-01≈	**3.32e-01**≈
F_{20}^{cec}	4.80e-01	1.07e+00−	**2.06e-01**+
F_{21}^{cec}	3.88e+00	1.04e+01−	**4.99e-01**+
F_{22}^{cec}	3.58e-01	7.02e-01−	**1.72e-01**+
F_{23}^{cec}	**3.29e+02**	**3.29e+02**≈	**3.29e+02**≈
F_{24}^{cec}	1.13e+02	1.13e+02≈	**1.13e+02**≈
F_{25}^{cec}	**1.27e+02**	1.31e+02≈	1.27e+02−
F_{26}^{cec}	1.00e+02	1.00e+02≈	**1.00e+02**+
F_{27}^{cec}	**4.68e+01**	6.18e+01≈	7.94e+01−
F_{28}^{cec}	**3.57e+02**	3.58e+02−	3.61e+02≈
F_{29}^{cec}	2.23e+02	2.24e+02≈	**2.22e+02**+
F_{30}^{cec}	4.73e+02	4.77e+02−	**4.66e+02**+
	+ (better)	2	13
	− (worse)	7	4
	≈ (no sig.)	21	13

(b) $D = 30$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	3.17e+05	3.75e+05≈	**6.54e+04**+
F_2^{cec}	**0.00e+00**	3.38e-09−	**0.00e+00**≈
F_3^{cec}	7.75e-06	7.25e-05−	**0.00e+00**+
F_4^{cec}	5.74e+01	5.35e+01≈	**5.08e+00**+
F_5^{cec}	2.03e+01	**2.03e+01**≈	2.03e+01−
F_6^{cec}	1.36e+01	1.31e+01+	**3.84e+00**+
F_7^{cec}	**0.00e+00**	3.67e-10≈	**0.00e+00**≈
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_9^{cec}	5.05e+01	5.09e+01≈	**4.25e+01**+
F_{10}^{cec}	**4.08e-04**	8.16e-04≈	2.04e-03≈
F_{11}^{cec}	**2.25e+03**	2.26e+03≈	2.43e+03−
F_{12}^{cec}	**3.87e-01**	3.91e-01≈	4.63e-01−
F_{13}^{cec}	3.46e-01	3.47e-01≈	**2.97e-01**+
F_{14}^{cec}	**2.64e-01**	2.66e-01≈	2.76e-01−
F_{15}^{cec}	6.97e+00	6.77e+00≈	**5.66e+00**+
F_{16}^{cec}	**9.75e+00**	9.79e+00≈	9.89e+00≈
F_{17}^{cec}	5.30e+04	5.65e+04≈	**1.61e+03**+
F_{18}^{cec}	6.66e+02	7.03e+02≈	**1.78e+01**+
F_{19}^{cec}	6.94e+00	6.97e+00≈	**4.49e+00**+
F_{20}^{cec}	5.40e+01	5.12e+01≈	**1.16e+01**+
F_{21}^{cec}	3.69e+03	3.20e+03≈	**3.03e+02**+
F_{22}^{cec}	2.31e+02	2.13e+02≈	**9.91e+01**+
F_{23}^{cec}	**3.15e+02**	**3.15e+02**≈	**3.15e+02**≈
F_{24}^{cec}	2.26e+02	2.26e+02≈	**2.24e+02**+
F_{25}^{cec}	2.04e+02	2.04e+02≈	**2.03e+02**+
F_{26}^{cec}	1.00e+02	1.00e+02≈	**1.00e+02**+
F_{27}^{cec}	4.08e+02	4.11e+02−	**3.56e+02**+
F_{28}^{cec}	8.53e+02	8.53e+02≈	**7.88e+02**+
F_{29}^{cec}	1.30e+03	1.30e+03≈	**8.07e+02**+
F_{30}^{cec}	1.78e+03	1.61e+03+	**1.49e+03**+
	+ (better)	2	20
	− (worse)	3	4
	≈ (no sig.)	25	6

(c) $D = 50$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	1.34e+06	1.38e+06≈	**4.91e+05**+
F_2^{cec}	1.49e+03	1.63e+03≈	**8.17e-09**+
F_3^{cec}	5.28e+01	3.94e+01≈	**7.18e-09**+
F_4^{cec}	9.48e+01	9.12e+01≈	**8.91e+01**≈
F_5^{cec}	**2.04e+01**	2.04e+01≈	2.04e+01−
F_6^{cec}	2.87e+01	2.90e+01≈	**8.33e+00**+
F_7^{cec}	1.82e-05	8.07e-05−	**0.00e+00**+
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_9^{cec}	1.22e+02	1.24e+02≈	**9.34e+01**+
F_{10}^{cec}	**9.80e-04**	1.22e-03≈	3.43e-03−
F_{11}^{cec}	4.95e+03	**4.90e+03**≈	5.21e+03−
F_{12}^{cec}	4.17e-01	**4.09e-01**≈	4.94e-01−
F_{13}^{cec}	4.05e-01	3.99e-01≈	**3.90e-01**≈
F_{14}^{cec}	3.14e-01	**3.13e-01**≈	3.40e-01≈
F_{15}^{cec}	1.56e+01	1.56e+01≈	**1.21e+01**+
F_{16}^{cec}	1.82e+01	**1.82e+01**≈	1.82e+01≈
F_{17}^{cec}	7.26e+05	6.73e+05≈	**2.15e+04**+
F_{18}^{cec}	9.80e+02	1.08e+03≈	**4.56e+02**+
F_{19}^{cec}	1.78e+01	1.78e+01≈	**1.43e+01**+
F_{20}^{cec}	6.85e+02	9.12e+02−	**5.17e+01**+
F_{21}^{cec}	1.28e+05	1.28e+05≈	**8.36e+03**+
F_{22}^{cec}	7.53e+02	7.29e+02≈	**5.46e+02**+
F_{23}^{cec}	**3.44e+02**	**3.44e+02**≈	**3.44e+02**≈
F_{24}^{cec}	**2.61e+02**	2.62e+02≈	2.68e+02−
F_{25}^{cec}	2.11e+02	2.12e+02≈	**2.07e+02**+
F_{26}^{cec}	1.00e+02	1.00e+02≈	**1.00e+02**+
F_{27}^{cec}	9.69e+02	1.03e+03≈	**4.33e+02**+
F_{28}^{cec}	1.29e+03	1.29e+03≈	**1.11e+03**+
F_{29}^{cec}	1.66e+03	1.70e+03≈	**1.08e+03**+
F_{30}^{cec}	**8.64e+03**	8.66e+03≈	8.65e+03≈
	+ (better)	0	18
	− (worse)	2	5
	≈ (no sig.)	28	7

表 8.5: 10, 30, 50 次元の CEC2014 benchmarks [136] での, JADE (Algorithm 13) における SEC に対する binomial/exponential 交叉の比較結果. 表中のデータは, 探索中に得られた最良解の目的関数値 $f(\boldsymbol{x}^{bsf})$ と最適値 $f(\boldsymbol{x}^*)$ との誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ の 51 試行における平均値である. 各関数において最良のデータを太字で示している. 誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ の 51 試行分のデータに対して行った, 有意水準 $p = 0.05$ の Wilcoxon rank-sum test を行った. 表中の各関数における記号 $+, \approx, -$ は, 比較交叉手法が SEC と比べてそれぞれ有意に優れている, 有意に劣っている, 有意差無しであったこと表す. また, 表下の 「# of better」, 「# of worse 」 , 「# of no sig. 」の項目は, 比較交叉手法が SEC と比べてそれぞれ有意に優れている, 有意に劣っている, 有意差無しであった関数の総数を表す.

(a) $D = 10$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	2.78e+04	7.13e+04−	**0.00e+00**+
F_2^{cec}	5.94e+01	9.72e+01≈	**0.00e+00**+
F_3^{cec}	3.71e+01	2.53e+01≈	**3.52e-05**+
F_4^{cec}	2.32e+00	**1.32e+00**≈	2.34e+01−
F_5^{cec}	**1.74e+01**	1.77e+01≈	1.77e+01≈
F_6^{cec}	1.15e+00	4.86e-01+	**9.92e-02**+
F_7^{cec}	1.87e-02	2.01e-02≈	**1.05e-02**+
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_9^{cec}	4.13e+00	4.21e+00≈	**3.29e+00**+
F_{10}^{cec}	**2.45e-03**	4.90e-03≈	9.80e-03−
F_{11}^{cec}	1.17e+02	1.34e+02≈	**8.83e+01**+
F_{12}^{cec}	**2.34e-01**	2.38e-01≈	2.55e-01≈
F_{13}^{cec}	1.33e-01	1.29e-01≈	**8.16e-02**+
F_{14}^{cec}	1.13e-01	1.14e-01≈	**1.02e-01**≈
F_{15}^{cec}	6.71e-01	6.38e-01≈	**5.54e-01**+
F_{16}^{cec}	1.82e+00	1.85e+00≈	**1.63e+00**+
F_{17}^{cec}	7.37e+03	1.10e+04≈	**3.14e+00**+
F_{18}^{cec}	1.92e+02	5.38e+02−	**4.00e-01**+
F_{19}^{cec}	**2.69e-01**	3.05e-01−	2.71e-01≈
F_{20}^{cec}	9.93e-01	8.75e+01−	**3.25e-01**+
F_{21}^{cec}	7.03e+02	8.54e+02≈	**8.45e-01**+
F_{22}^{cec}	1.93e-01	2.35e-01−	**1.87e-01**≈
F_{23}^{cec}	**3.04e+02**	3.17e+02−	3.29e+02−
F_{24}^{cec}	1.11e+02	1.13e+02−	**1.09e+02**+
F_{25}^{cec}	1.26e+02	1.28e+02≈	**1.25e+02**+
F_{26}^{cec}	1.00e+02	1.00e+02≈	**1.00e+02**+
F_{27}^{cec}	4.94e+01	**4.28e+01**≈	6.55e+01−
F_{28}^{cec}	**3.68e+02**	3.74e+02−	4.03e+02−
F_{29}^{cec}	3.12e+02	3.19e+02≈	**2.33e+02**+
F_{30}^{cec}	5.43e+02	5.58e+02≈	**4.76e+02**+
	+ (better)	1	19
	− (worse)	8	5
	≈ (no sig.)	21	6

(b) $D = 30$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	1.68e+07	1.51e+07≈	**1.81e+03**+
F_2^{cec}	5.37e+01	2.43e+02−	**0.00e+00**+
F_3^{cec}	4.28e+02	6.94e+02≈	**8.40e-05**+
F_4^{cec}	5.99e+01	5.10e+01+	**7.82e-02**+
F_5^{cec}	2.02e+01	**2.02e+01**≈	2.03e+01−
F_6^{cec}	1.33e+01	1.30e+01≈	**9.01e+00**+
F_7^{cec}	**1.53e-04**	2.24e-04−	5.32e-04−
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_9^{cec}	4.29e+01	4.41e+01≈	**2.56e+01**+
F_{10}^{cec}	**2.45e-03**	**2.45e-03**≈	6.94e-03−
F_{11}^{cec}	**1.65e+03**	1.70e+03≈	1.67e+03≈
F_{12}^{cec}	2.27e-01	**2.19e-01**≈	2.52e-01−
F_{13}^{cec}	2.94e-01	3.00e-01≈	**2.05e-01**+
F_{14}^{cec}	**2.16e-01**	2.20e-01≈	2.24e-01≈
F_{15}^{cec}	5.29e+00	5.14e+00≈	**3.12e+00**+
F_{16}^{cec}	9.51e+00	9.53e+00≈	**9.39e+00**≈
F_{17}^{cec}	2.54e+06	2.83e+06≈	**1.12e+03**+
F_{18}^{cec}	1.62e+04	1.69e+04≈	**9.06e+01**+
F_{19}^{cec}	7.21e+00	7.20e+00≈	**4.47e+00**+
F_{20}^{cec}	5.28e+03	5.44e+03≈	**2.38e+03**+
F_{21}^{cec}	4.33e+05	5.03e+05−	**1.07e+04**+
F_{22}^{cec}	2.07e+02	2.09e+02≈	**1.60e+02**+
F_{23}^{cec}	3.15e+02	3.15e+02≈	**3.15e+02**+
F_{24}^{cec}	2.28e+02	2.28e+02≈	**2.26e+02**+
F_{25}^{cec}	2.08e+02	2.08e+02−	**2.05e+02**+
F_{26}^{cec}	**1.00e+02**	1.00e+02≈	1.02e+02−
F_{27}^{cec}	4.04e+02	4.13e+02≈	**3.42e+02**+
F_{28}^{cec}	8.71e+02	8.68e+02≈	**7.93e+02**+
F_{29}^{cec}	1.71e+03	1.67e+03≈	**7.28e+02**+
F_{30}^{cec}	4.10e+03	4.13e+03≈	**1.48e+03**+
	+ (better)	1	21
	− (worse)	4	5
	≈ (no sig.)	25	4

(c) $D = 50$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	1.60e+07	1.73e+07≈	**1.86e+04**+
F_2^{cec}	4.19e+04	4.26e+04≈	**0.00e+00**+
F_3^{cec}	5.81e+03	6.68e+03≈	**3.66e+03**+
F_4^{cec}	8.53e+01	8.69e+01≈	**5.47e+00**+
F_5^{cec}	2.03e+01	**2.03e+01**≈	2.04e+01−
F_6^{cec}	2.58e+01	2.56e+01≈	**1.67e+01**+
F_7^{cec}	5.49e-03	6.06e-03≈	**4.88e-03**≈
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_9^{cec}	1.00e+02	1.02e+02≈	**5.41e+01**+
F_{10}^{cec}	**3.43e-03**	6.61e-03≈	8.57e-03−
F_{11}^{cec}	3.82e+03	3.83e+03≈	**3.82e+03**≈
F_{12}^{cec}	2.07e-01	**2.05e-01**≈	2.61e-01−
F_{13}^{cec}	3.71e-01	3.73e-01≈	**3.17e-01**+
F_{14}^{cec}	**2.72e-01**	2.72e-01≈	2.95e-01−
F_{15}^{cec}	1.25e+01	1.25e+01≈	**7.48e+00**+
F_{16}^{cec}	1.78e+01	1.79e+01≈	**1.77e+01**≈
F_{17}^{cec}	6.64e+06	6.55e+06≈	**2.47e+03**+
F_{18}^{cec}	1.41e+04	1.41e+04≈	**1.63e+02**+
F_{19}^{cec}	1.80e+01	1.86e+01≈	**1.55e+01**+
F_{20}^{cec}	2.16e+04	2.16e+04≈	**6.23e+03**+
F_{21}^{cec}	3.38e+06	3.57e+06≈	**4.63e+04**+
F_{22}^{cec}	5.79e+02	5.69e+02≈	**4.95e+02**+
F_{23}^{cec}	3.45e+02	3.46e+02−	**3.44e+02**+
F_{24}^{cec}	2.58e+02	**2.58e+02**≈	2.74e+02−
F_{25}^{cec}	**2.16e+02**	2.16e+02−	2.22e+02−
F_{26}^{cec}	**1.00e+02**	1.00e+02−	1.02e+02−
F_{27}^{cec}	8.77e+02	8.59e+02≈	**4.55e+02**+
F_{28}^{cec}	1.42e+03	1.43e+03≈	**1.14e+03**+
F_{29}^{cec}	2.66e+03	2.61e+03≈	**9.04e+02**+
F_{30}^{cec}	1.07e+04	1.05e+04≈	**1.00e+04**+
	+ (better)	0	19
	− (worse)	3	7
	≈ (no sig.)	27	4

表 8.6: 10, 30, 50 次元の CEC2014 benchmarks [136] での, 突然変異戦略に current-to-$p$best/1 を用いた SHADE (Algorithm 19) における SEC に対する binomial/exponential 交叉の比較結果. 表中のデータは, 探索中に得られた最良解の目的関数値 $f(\boldsymbol{x}^{bsf})$ と最適値 $f(\boldsymbol{x}^*)$ との誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ の 51 試行における平均値である. 各関数において最良のデータを太字で示している. 誤差値 $|f(\boldsymbol{x}^{bsf}) - f(\boldsymbol{x}^*)|$ の 51 試行分のデータに対して行った, 有意水準 $p = 0.05$ の Wilcoxon rank-sum test を行った. 表中の各関数における記号 $+, \approx, -$ は, 比較交叉手法が SEC と比べてそれぞれ有意に優れている, 有意に劣っている, 有意差無しであったこと表す. また, 表下の 「# of better」, 「# of worse 」 , 「# of no sig. 」の項目は, 比較交叉手法が SEC と比べてそれぞれ有意に優れている, 有意に劣っている, 有意差無しであった関数の総数を表す.

(a) $D = 10$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_2^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_3^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_4^{cec}	**1.34e+01**	1.67e+01≈	2.87e+01−
F_5^{cec}	1.63e+01	1.75e+01≈	**1.54e+01**≈
F_6^{cec}	1.58e-01	2.64e-03+	**0.00e+00**+
F_7^{cec}	4.89e-03	**4.72e-03**≈	7.75e-03≈
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_9^{cec}	3.40e+00	3.45e+00≈	**2.92e+00**+
F_{10}^{cec}	2.45e-03	**1.22e-03**≈	1.47e-02−
F_{11}^{cec}	9.26e+01	9.62e+01≈	**7.57e+01**≈
F_{12}^{cec}	1.54e-01	1.56e-01≈	**1.42e-01**≈
F_{13}^{cec}	1.20e-01	1.21e-01≈	**7.15e-02**+
F_{14}^{cec}	9.97e-02	1.00e-01≈	**9.95e-02**≈
F_{15}^{cec}	5.31e-01	5.00e-01≈	**4.92e-01**≈
F_{16}^{cec}	1.86e+00	1.86e+00≈	**1.52e+00**+
F_{17}^{cec}	1.34e+00	**1.26e+00**≈	1.66e+00≈
F_{18}^{cec}	**1.79e-01**	3.71e-01−	2.67e-01≈
F_{19}^{cec}	**1.92e-01**	2.37e-01≈	2.16e-01−
F_{20}^{cec}	**1.91e-01**	4.03e-01−	2.04e-01≈
F_{21}^{cec}	4.09e-01	4.04e-01≈	**3.10e-01**≈
F_{22}^{cec}	**1.58e-01**	2.61e-01−	2.57e-01−
F_{23}^{cec}	3.29e+02	**3.23e+02**≈	3.29e+02≈
F_{24}^{cec}	1.10e+02	1.11e+02≈	**1.09e+02**+
F_{25}^{cec}	**1.16e+02**	1.17e+02≈	1.22e+02−
F_{26}^{cec}	1.00e+02	1.00e+02≈	**1.00e+02**+
F_{27}^{cec}	**4.70e+01**	8.02e+01≈	1.07e+02−
F_{28}^{cec}	**3.72e+02**	3.79e+02≈	3.97e+02−
F_{29}^{cec}	2.22e+02	2.22e+02≈	**2.22e+02**≈
F_{30}^{cec}	4.66e+02	**4.64e+02**≈	4.71e+02≈
	+ (better)	1	6
	− (worse)	3	7
	≈ (no sig.)	26	17

(b) $D = 30$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	1.32e+03	1.62e+03≈	**8.53e+02**+
F_2^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_3^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_4^{cec}	1.24e+00	**0.00e+00**≈	**0.00e+00**≈
F_5^{cec}	2.01e+01	2.01e+01≈	**2.01e+01**+
F_6^{cec}	1.19e+01	1.12e+01+	**5.82e-01**+
F_7^{cec}	**0.00e+00**	**0.00e+00**≈	1.45e-04≈
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_9^{cec}	4.11e+01	4.00e+01≈	**1.51e+01**+
F_{10}^{cec}	5.31e-03	**4.90e-03**≈	1.02e-02−
F_{11}^{cec}	1.70e+03	1.75e+03≈	**1.44e+03**+
F_{12}^{cec}	1.79e-01	1.83e-01≈	**1.61e-01**+
F_{13}^{cec}	3.02e-01	3.05e-01≈	**2.16e-01**+
F_{14}^{cec}	2.13e-01	**2.10e-01**≈	2.39e-01−
F_{15}^{cec}	3.15e+00	3.20e+00≈	**2.53e+00**+
F_{16}^{cec}	9.72e+00	9.66e+00≈	**9.05e+00**+
F_{17}^{cec}	**8.63e+02**	1.02e+03−	1.02e+03−
F_{18}^{cec}	3.14e+01	**2.85e+01**≈	5.54e+01−
F_{19}^{cec}	**4.06e+00**	4.15e+00≈	4.37e+00−
F_{20}^{cec}	**1.32e+01**	1.36e+01≈	1.36e+01≈
F_{21}^{cec}	**2.01e+02**	2.35e+02≈	2.39e+02≈
F_{22}^{cec}	**1.06e+02**	1.28e+02≈	1.08e+02≈
F_{23}^{cec}	**3.15e+02**	**3.15e+02**≈	**3.15e+02**≈
F_{24}^{cec}	2.25e+02	**2.24e+02**+	2.26e+02≈
F_{25}^{cec}	**2.03e+02**	2.03e+02≈	2.04e+02−
F_{26}^{cec}	1.00e+02	**1.00e+02**≈	1.02e+02−
F_{27}^{cec}	4.04e+02	4.06e+02−	**3.23e+02**+
F_{28}^{cec}	**8.14e+02**	8.21e+02≈	8.28e+02−
F_{29}^{cec}	7.26e+02	7.23e+02≈	**7.16e+02**≈
F_{30}^{cec}	1.26e+03	**1.17e+03**≈	1.56e+03≈
	+ (better)	2	10
	− (worse)	2	8
	≈ (no sig.)	26	12

(c) $D = 50$

関数	SEC	exponential 交叉	binomial 交叉
F_1^{cec}	3.97e+04	6.02e+04−	**1.74e+04**+
F_2^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_3^{cec}	**0.00e+00**	1.29e-06−	**0.00e+00**≈
F_4^{cec}	**1.55e+01**	2.52e+01−	2.28e+01≈
F_5^{cec}	2.02e+01	2.02e+01≈	**2.01e+01**+
F_6^{cec}	2.52e+01	2.57e+01≈	**5.27e+00**+
F_7^{cec}	3.48e-03	3.09e-03≈	**1.55e-03**≈
F_8^{cec}	**0.00e+00**	**0.00e+00**≈	**0.00e+00**≈
F_9^{cec}	9.63e+01	1.01e+02−	**3.31e+01**+
F_{10}^{cec}	7.59e-03	**7.35e-03**≈	1.10e-02≈
F_{11}^{cec}	3.76e+03	3.81e+03≈	**3.38e+03**+
F_{12}^{cec}	1.77e-01	1.71e-01≈	**1.60e-01**+
F_{13}^{cec}	3.87e-01	3.77e-01≈	**3.12e-01**+
F_{14}^{cec}	**2.41e-01**	2.46e-01≈	2.88e-01−
F_{15}^{cec}	6.92e+00	6.88e+00≈	**5.79e+00**+
F_{16}^{cec}	1.82e+01	1.82e+01≈	**1.75e+01**+
F_{17}^{cec}	2.87e+03	2.89e+03≈	**2.58e+03**≈
F_{18}^{cec}	1.25e+02	**1.14e+02**≈	1.49e+02−
F_{19}^{cec}	1.15e+01	1.16e+01≈	**9.68e+00**+
F_{20}^{cec}	**9.08e+01**	1.20e+02−	1.98e+02−
F_{21}^{cec}	**1.23e+03**	1.35e+03≈	1.27e+03≈
F_{22}^{cec}	4.65e+02	4.81e+02≈	**3.77e+02**+
F_{23}^{cec}	**3.44e+02**	**3.44e+02**≈	**3.44e+02**≈
F_{24}^{cec}	**2.59e+02**	2.60e+02≈	2.74e+02−
F_{25}^{cec}	**2.06e+02**	2.06e+02≈	2.12e+02−
F_{26}^{cec}	**1.00e+02**	1.00e+02≈	1.02e+02≈
F_{27}^{cec}	7.94e+02	8.81e+02≈	**4.49e+02**+
F_{28}^{cec}	1.18e+03	1.19e+03≈	**1.14e+03**+
F_{29}^{cec}	9.08e+02	9.22e+02≈	**8.89e+02**≈
F_{30}^{cec}	**8.59e+03**	8.60e+03≈	9.43e+03−
	+ (better)	0	13
	− (worse)	5	6
	≈ (no sig.)	25	11

8.6　おわりに

本章では, 始めに DE の代表的な交叉手法である exponential 交叉が, Rosenbrock 関数などの人工的に作成したベンチマーク関数に現れる, 依存関係にある決定変数が隣接しているという性質を探索に利用している問題点を指摘した. 実験では, この非現実的な問題性質が存在しない場合, exponential 交叉の性能は乏しくなることを示した. この結果から, Rosenbrock 関数などを含むベンチマークセット [260, 223, 231, 100, 90] を用いた性能評価実験では, exponential 交叉の性能を実際よりも過大に評価していたと考えられる [192, 99, 29, 269, 128, 268]. そのため, exponential 交叉の性能評価を再度行う必要があると思われる. また, exponential 交叉が有するバイアスを解消した SEC は, 全体的に exponential 交叉よりも優れており, 関数によっては binomial 交叉と同程度の性能を持つことを示した. 本章で得られた知見から, 8.2.1 節にてあげた例のように事前に依存関係にある決定変数が隣接していることが判明している実問題を除き, SEC, 又は binomial 交叉を exponential 交叉の代わりに今後は使用すべきである.

本章で示した DE における exponential 交叉のように, 明示的, 又は非明示的に依存関係にある決定変数が隣接している性質を探索に利用できる手法は, ベンチマークセットを用いた実験では過大評価してしまう恐れがある. これを防ぐため, 8.2.2 節で説明した Tang らの方法 [231] を用いて, 変数の並び順をランダムに並び替える操作を全てのベンチマーク関数に適用することを推奨する. また, 本章は単目的境界制約付き関数最適化問題のみを取り扱ったが, 依存関係にある変数が隣接しているような人工的なベンチマーク関数は, 様々な関数最適化問題においてみられる. 例えば, 多目的最適化問題においては KUR [124], WFG [104], 制約付き最適化問題では CEC2010 Constrained Optimization Benchmarks [156] などがあげられる. これらのベンチマーク関数についても, 決定変数が隣接している性質を探索に利用できる手法の性能誤評価の可能性があると考えられる.

本章では現在 DE の交叉手法として最も主流である binomial 交叉, 及び exponential 交叉を新たに分析, 及び評価の対象とし, 上記に要約した新たな知見を得ることができた. しかし, その他にも orthogonal crossover [248], eigenvector-based crossover operator [84], hybrid linkage crossover [34] など, DE における交叉手法がこれまでにいくつか提案されている. これらの交叉手法における分析が, 今後の課題として残される.

第 9 章

おわりに

9.1 本論文のまとめ

本論文は「関数最適化問題に対する適応型差分進化法の研究」と題し, 探索中に得られた情報を基に使用するパラメータ設定を調整する適応的パラメータ制御法を用いた DE (適応 DE) に関する研究成果を各章にて述べた. 以下では, 5, 6, 8, 7 章にて得られた成果をまとめる:

5 章. 適応 DE の適応手法の解析

多くの優れた適応 DE が提案されている一方, その適応手法に関する知見は乏しい. そこで, 適応 DE の適応手法を (i) ベンチマーク問題集における性能評価, (ii) 本章にて提案する新たなシミュレーション法により解析した.

(i) の目的は「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しを評価することであり, 様々な突然変異戦略と交叉手法の組み合わせにおける適応手法の性能を, BBOB benchmarks [93] にて評価した. その結果, 使用されることが前提とされている突然変異戦略と交叉手法の組み合わせ以外のオペレータを用いた場合, 各文献にて報告されている性能と比べ劣る傾向が明らかになった.

次に, (ii) について, 理想的なパラメータ適応の軌跡の代理である oracle パラメータを用いた新たなシミュレーション法により, 適応 DE の適応手法を解析した. 一般的に, 適応手法の解析は困難であるが [119], その最も大きな障害は, 適応手法が生成したパラメータ値自体の良し悪しを判別することが困難な点である. 「遺伝的オペレータ」と「適応手法」の組み合わせの良し悪しは (i) の実験にて実証できたものの, 「適応手法」のみを独立して評価することは, 非常に困難である. これは, (a) パラメータ値空間, (b) 解空間, (c) 目的関数空間という 3 つの空間が存在し, それぞれを分けて考えることが通常はできないためである. そこで, 本研究では (a) パラメータ値空間のみを独立して扱うシミュレーション法を提案した. 本シミュレーション法ではパラメータ値を独立して評価することが可能であり, これまで難しかったパラメータ適応手法の適応能力についての議論が可能となった. つまり, 提案シミュレーション法では, ベンチマーク関数, 対象問題の次元数, 使用するオペレータの種類といった様々な要因とは独立に, 適応手

法の性能評価がある程度可能である．提案シミュレーション法を用いた実験結果から，oracle パラメータが世代ごとに急激に変化するシミュレーションモデルにおいては，既存の適応 DE では適応することが困難であることがわかった．

6 章．Success-History based Adaptive Differential Evolution

5 章での適応 DE の解析結果から，既存の適応 DE の適応手法にはいくつかの問題点があり，改善の余地が残されていることがわかった．そこで，6 章では既存の適応手法の問題点を考慮した，DE のための新たな適応手法である Success-History based Adaptive DE (SHADE) を提案した．

始めに oracle パラメータを用いたシミュレーション法における SHADE のパラメータ適応性能を計測した．その結果，既存の適応 DE の適応手法が適応に失敗する，oracle パラメータが世代ごとに急激に変化するモデルにおいても，SHADE は良好な適応性能を示すことがわかった．次に，BBOB benchmarks を用いてベンチマーク集合における探索性能を評価した．その結果，使用する然変異戦略，及び交叉手法に依らず，多くの場合 SHADE が他の適応 DE よりも優れていることがわかった．さらに，SHADE は，有限差分法を用いた準ニュートン法や Nelder-Mead 法 [167] といった目的関数の勾配情報を必要とせず目的関数値 $f(\boldsymbol{x})$ のみを利用する伝統的な探索手法よりも優れており，近年の black-box optimization 環境における state-of-the-art な EA である BIPOP-CMA-ES [88] に匹敵する性能を有することが確認された．

7 章．ハイブリッド関数における適応 DE のパラメータ適応の振る舞いの解析

6 章では DE における新たな適応手法 SHADE を提案し，既存手法と比較することでその有用性を実証した．しかし，SHADE があらゆる問題に対しても完璧に対応可能な，万能な適応手法であると主張しているわけではない．本章では SHADE を含む，適応 DE の適応手法がパラメータ適応に失敗する現象について述べた．具体的には，実問題にて現れうる非一様な問題性質 [82] を有するよう設計された，複数のベンチマーク関数から成るハイブリッド関数 [231, 132, 136] における，適応 DE の (1) 探索性能，及び (2) パラメータ適応の挙動を解析した．

(1) については，適応 DE をハイブリッド関数に適用した結果，問題性質が互いに異なるベンチマーク関数から構成されたハイブリッド関数は，適応 DE にとって困難な問題であることがわかった．このハイブリッド関数における適応 DE の探索失敗現象を明らかにするため，(2) にて適応手法のパラメータ適応の振る舞いを解析した．

まず，この目的のために，異なる関数インスタンスでの適応過程におけるパラメータ系列間の距離を計測し要約する方法を提案した．そして，提案手法を用いて解析したところ，互いに異なる問題性質を有する構成関数から成るハイブリッド関数では，どちらかの構成関数に適したパラメータ値に適応してしまい残りの関数に対しては不適切なパラメータ設定となり，探索が失敗することが明らかになった．これは，ハイブリッド関数は適応 DE の探索が困難な問題クラスであることを意味し，著者の知る限り初めて報告される事例である．また，距離に基づくパラメータ適応の振る舞いの解析方法の提案が本章の副次的な成果であり，本手法は様々な種類の最適化問題における適応 DE の適応手

法の解析に今後使用されることが期待される．

8 章．DE における交叉オペレータの分析と評価

5 章と 7 章では，適応 DE の適応手法の解析を行った．また，6 章では新たな適応手法である SHADE を提案した．以上の章は適応手法を興味の対象としていたのに対して，本章では DE における交叉オペレータを扱った．

Binomial 交叉と exponential 交叉は，DE において一般的に使用されている交叉手法である．DE が提案された当初は binomial 交叉の方が優れているとされていたが [220, 161]，近年では exponential 交叉の有用性を報告する研究 [99, 29, 269, 128, 268] が増加しており，どちらのオペレータが有用であるか明確でない．8 章ではまず，exponential 交叉は Rosenbrock 関数などの人工的に作成したベンチマーク関数に存在する，依存関係にある決定変数が変数番号上において隣接しているという非現実的な性質を利用可能である点を指摘した．次に，ベンチマーク関数からこの非現実的な性質を解消した場合，exponential 交叉の探索性能は binomial 交叉と比べ劣るようになることを示した．また，バイアスを持たない Shuffled Exponential Crossover (SEC) [188] を含む大規模な交叉オペレーターの実験的評価を行い，(1) 多くのベンチマーク関数において binomial 交叉が最も良好な性能を有し，(2) SEC は exponential 交叉と比べて同等以上の性能を持つことを示した．本章で得られた知見から，事前に依存関係にある決定変数が隣接していることが判明している実問題を除き，SEC，又は binomial 交叉を exponential 交叉の代わりに使用することを，今後は推奨する．

9.2　本論文にて提案した適応手法 SHADE の応用について

6 章にて提案した適応 DE の適応手法 SHADE は，既に油槽シミュレータの制御パラメータ最適化問題 [7] や 3 次元物体の表面反射特性パラメータ推定 [53] などに適用され，既存手法と比較して良好な性能が報告されている．そのため，実問題において既存の探索手法では得られなかった高精度の近似解が，SHADE を使用することで獲得できることが期待される．

また，探索停滞時に親個体の選択方法を切り替える戦略 [85]，複数の突然変異戦略を適応的に選択する戦略 [80]，binomial 交叉と exponential 交叉の 2 つの交叉手法を使用する戦略 [32]，島型モデル [181] といった様々なアルゴリズムの要素を加えることで，SHADE の探索性能をさらに向上できることが報告されている．さらに，2014 年度の国際会議 IEEE CEC にて開催された CEC2014 Competition on Real-Parameter Single Objective Optimization にて，6.6 節にて提案した L-SHADE [229] が 1 位[*1]，2015 年度の CEC2015 Competition on Real-Parameter Single Objective Optimization では，L-SHADE に基づく手法 [83, 11, 202] が上位 1, 2, 3 位を占めている[*2]．このことから，SHADE は効率的な探索手法として，今後さらなる発展が期待される．

[*1] http://www3.ntu.edu.sg/home/EPNSugan/index_files/CEC2014/CEC2014.htm
[*2] http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2015/CEC2015.htm

9.3　今後の課題

今後の課題として, 5 章にて提案した oracle パラメータを用いたシミュレーション法のさらなる調査が課題として考えられる. 本シミュレーション法ではパラメータ値を独立して評価することができ, これまで難しかった適応手法の適応能力の解析を可能とする. 本論文では適応 DE の F, C のパラメータ適応手法のみを解析対象としたが, その他の制御パラメータの適応手法を解析する際にも有用であると思われる. EA 全体においても特定の適応手法が優れた性能を示す理由を解明した研究は少ないため [119], 本シミュレーション法を用いることにより新たな知見が EA 全体においても得られることが期待できる. また, 特定の探索手法間, 及び遺伝的オペレータの性能差が強調される関数を Genetic Programming (GP) により自動生成する枠組みがこれまでに提案されているが [126, 212], この枠組みを本シミュレーション法に適用することで, それぞれの適応手法が得意, 不得意とする oracle パラメータを生成でき, さらなる知見が得られると考えられる. これは今後の課題として残される.

SHADE を含む, 適応 DE の適応手法は 「対象問題, 及び現在の探索状況に適したパラメータ設定を使用したために, 優れた子個体を生成することができた」という近視眼的な仮定に基づいている. しかし, 7 章にて扱った互いに異なる問題性質を有する構成関数から成るハイブリッド関数においては, この特徴のために適応 DE の探索, 及びパラメータ適応が失敗することが明らかになった. この知見を元に, これまでとは全く異なるような適応 DE における適応手法を設計することが, 今後の課題としてあげられる.

参考文献

[1] B. Adenso-Díaz and M. Laguna. Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search. *Operations Research*, 54(1):99–114, 2006.

[2] Y. Akimoto. *Design of Evolutionary Computation for Continuous Optimization*. PhD thesis, Tokyo Institute of Technology, 2011.

[3] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi. Theoretical analysis of evolutionary computation on continuously differentiable functions. In *GECCO*, pages 1401–1408, 2010.

[4] Y. Akimoto, J. Sakuma, I. Ono, and S. Kobayashi. Adaptation of expansion rate for real-coded crossovers. In *GECCO*, pages 739–746, 2009.

[5] C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In *CP*, pages 142–157, 2009.

[6] J. Arabas, Z. Michalewicz, and J. J. Mulawka. Gavaps - a genetic algorithm with varying population size. In *ICEC*, pages 73–78, 1994.

[7] C. Aranha, R. Tanabe, R. Chassagne, and A. S. Fukunaga. Optimization of oil reservoir models using tuned evolutionary algorithms and adaptive differential evolution. In *IEEE CEC*, pages 877–884, 2015.

[8] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[9] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *IEEE CEC*, pages 1769–1776, 2005.

[10] A. Auger, N. Hansen, J. M. P. Zerpa, R. Ros, and M. Schoenauer. Experimental comparisons of derivative free optimization algorithms. In *SEA*, pages 3–15, 2009.

[11] N. H. Awad, M. Z. Ali, and R. G. Reynolds. A Differential Evolution algorithm with success-based parameter adaptation for CEC2015 learning-based optimization. In *IEEE CEC*, pages 1098–1105, 2015.

[12] T. Bäck. Self-Adaptation in Genetic Algorithms. In *ECAL*, pages 263–271, 1992.

[13] T. Bäck. Optimal Mutation Rates in Genetic Search. In *ICGA*, pages 2–8, 1993.

[14] T. Bäck, A. E. Eiben, and N. A. L. van der Vaart. An Empirical Study on GAs "Without Parameters". In *PPSN*, pages 315–324, 2000.

[15] T. Bäck, C. Foussette, and P. Krause. *Contemporary Evolution Strategies*. Natural Computing Series. Springer, 2013.

[16] T. Bäck and H. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[17] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul. The best-so-far selection in artificial bee colony algorithm. *Appl. Soft Comput.*, 11(2):2888–2901, 2011.

[18] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. Sequential parameter optimization. In *IEEE CEC*, pages 773–780, 2005.

[19] H. Bersini, M. Dorigo, S. Langerman, G. Seront, and L. M. Gambardella. Results of the First International Contest on Evolutionary Optimisation (1st ICEO). In *ICEC*, pages 611–615, 1996.

[20] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A Racing Algorithm for Configuring Metaheuristics. In *GECCO*, pages 11–18, 2002.

[21] K. D. Boese, A. B. Kahng, and S. Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Oper. Res. Lett.*, 16(2):101–113, 1994.

[22] D. Bollegala, N. Noman, and H. Iba. Rankde: learning a ranking function for information retrieval using differential evolution. In *GECCO*, pages 1771–1778, 2011.

[23] P. A. N. Bosman, J. Grahl, and D. Thierens. Benchmarking Parameter-Free AMaLGaM on Functions With and Without Noise. *Evolutionary Computation*, 21(3):445–469, 2013.

[24] Z. Bouzarkouna, A. Auger, and D. Y. Ding. Local-meta-model CMA-ES for partially separable functions. In *GECCO*, pages 869–876, 2011.

[25] J. Bräuninger. A variable metric algorithm for unconstrained minimization without evaluation of derivatives. *Numerische Mathematik*, 36(4):359–373, 1981.

[26] J. Brest, B. Boskovic, S. Greiner, V. Zumer, and M. S. Maucec. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Comput.*, 11(7):617–629, 2007.

[27] 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. *IEEE Tran. Evol. Comput.*, 10(6):646–657, 2006.

[28] J. Brest and M. S. Maučec. Population size reduction for the differential evolution algorithm. *Applied Intelligence*, 29(3):228–247, 2008.

[29] J. Brest and M. S. Maučec. Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Comput.*, 15(11):2157–2174, 2011.

[30] J. Brest, A. Zamuda, B. Bošković, S. Greiner, and V. Žumer. An analysis of the control parameters' adaptation in DE. In *Advances in Differential Evolution*, pages

89–110. Springer, 2008.

[31] S. H. Brooks. A discussion of random methods for seeking maxima. *Operations Research*, 6(2):244–251, 1958.

[32] P. Bujok and J. Tvrdík. Adaptive Differential Evolution: SHADE with Competing Crossover Strategies. In *ICAISC*, pages 329–339, 2015.

[33] M. Leach C. Brown, Y. Jin and M. Hodgson. μ jade: Adaptive differential evolution with a small population. *Soft Comput.*, page in press, 2011.

[34] Y. Cai and J. Wang. Differential evolution with hybrid linkage crossover. *Inf. Sci.*, 320:244–287, 2015.

[35] R. Caruana, L. J. Eshelman, and J. D. Schaffer. Representation and Hidden Bias II: Eliminating Defining Length Bias in Genetic Search via Shuffle Crossover. In *IJCAI*, pages 750–755, 1989.

[36] C. Chen and T. Chiang. Adaptive differential evolution: A visual comparison. In *IEEE CEC*, pages 401–408, 2015.

[37] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evolutionary Computation*, 6(1):58–73, 2002.

[38] L. Da Costa, Á. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *GECCO*, pages 913–920, 2008.

[39] M. A. M. d. Oca, T. Stützle, M. Birattari, and M. Dorigo. Frankenstein's PSO: A Composite Particle Swarm Optimization Algorithm. *IEEE Trans. Evolutionary Computation*, 13(5):1120–1132, 2009.

[40] S. Das, A. Abraham, U. K. Chakraborty, and Amit Konar. Differential evolution using a neighborhood-based mutation operator. *IEEE Tran. Evol. Comput.*, 13(3):526–553, 2009.

[41] S. Das, A. Konar, and U. K. Chakraborty. Two improved differential evolution schemes for faster global search. In *GECCO*, pages 991–998, 2005.

[42] S. Das, A. Konar, and U. K. Chakraborty. Annealed Differential Evolution. In *IEEE CEC*, pages 1926–1933, 2007.

[43] S. Das and P. N. Suganthan. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Tran. Evol. Comput.*, 15(1):4–31, 2011.

[44] M. A. Montes de Oca, T. Stützle, K. Van den Enden, and M. Dorigo. Incremental social learning in particle swarms. *IEEE Trans. on Systems, Man, and Cybernetics, PartB*, 41(2):368–384, 2011.

[45] K. Deb and R. B Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[46] K. Deb, A. Anand, and D. Joshi. A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization. *Evol. Comput.*, 10(4):345–369, 2002.

[47] K. Deb and H. Beyer. Self-Adaptation in Real-Parameter Genetic Algorithms with Simulated Binary Crossover. In *GECCO*, pages 172–179, 1999.

[48] K. Deb and M. Goya. A Combined Genetic Adaptive Search (geneAS) for Engineering Design. *Computer Science and Informatics*, 26(4):30–45, 1996.

[49] K. Deb, K. Sindhya, and T. Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *GECCO*, pages 1187–1194, 2007.

[50] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[51] M. Drozdik, H. E. Aguirre, Y. Akimoto, and K. Tanaka. Comparison of Parameter Control Mechanisms in Multi-objective Differential Evolution. In *LION*, pages 89–103, 2015.

[52] R. C. Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. In *IEEE CEC*, volume 1, pages 94–100, 2001.

[53] S. Eguchi, Y. Matsugano, H. Sakaguchi, S. Ono, H. Fukuda, R. Furukawa, and H. Kawasaki. A Comparative Study on Self-Adaptive Differential Evolution Algorithms for Test Functions and a Real-World Problem. In *LION*, pages 131–136, 2015.

[54] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.*, 3(2):124–141, 1999.

[55] A. E. Eiben, M. C. Schut, and A. R. de Wilde. Is Self-adaptation of Selection Pressure and Population Size Possible? - A Case Study. In *PPSN*, pages 900–909, 2006.

[56] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[57] M. El-Abd and M. S. Kamel. Black-box optimization benchmarking for noiseless function testbed using particle swarm optimization. In *GECCO (Companion)*, pages 2269–2274, 2009.

[58] L. J. Eshelman and J. D. Schaffer. Real-Coded Genetic Algorithms and Interval-Schemata. In *FOGA*, pages 187–202, 1992.

[59] H. Y. Fan and J. Lampinen. A trigonometric mutation operation to differential evolution. *J. of Global Optimization*, 27(1):105–129, 2003.

[60] Q. Fan and X. Yan. Self-Adaptive Differential Evolution Algorithm With Zoning Evolution of Control Parameters and Adaptive Mutation Strategies. *IEEE T. Cybernetics*, page in press, 2013.

[61] V. Feoktistov. *Differential Evolution in Search of Solutions*. Springer, 2006.

[62] Á. Fialho, R. Ros, M. Schoenauer, and M. Sebag. Comparison-Based Adaptive Strategy Selection with Bandits in Differential Evolution. In *PPSN*, pages 194–203,

2010.

[63] Á. Fialho, M. Schoenauer, and M. Sebag. Analysis of adaptive operator selection techniques on the royal road and long k-path problems. In *GECCO*, pages 779–786, 2009.

[64] T. C. Fogarty. Varying the probability of mutation in the genetic algorithm. In *ICGA9*, pages 104–109, 1989.

[65] D. B. Fogel. Revisiting Overlooked Foundations of Evolutionary Computation: Part I. *Cybernetics and Systems*, 41(5):343–358, 2010.

[66] D. B. Fogel. Revisiting Overlooked Foundations of Evolutionary Computation: Part II. *Cybernetics and Systems*, 41(6):407–415, 2010.

[67] D. B. Fogel and H. Beyer. A Note on the Empirical Evaluation of Intermediate Recombination. *Evol. Comput.*, 3(4):491–495, 1995.

[68] L. J. Fogel. Autonomous automata. *Industrial Research Magazine*, 4(2):14–19, 1962.

[69] G. Francesca, P. Pellegrini, T. Stützle, and M. Birattari. Off-line and On-line Tuning: A Study on Operator Selection for a Memetic Algorithm Applied to the QAP. In *EvoCOP*, pages 203–214, 2011.

[70] A. S. Fukunaga. Restart Scheduling for Genetic Algorithms. In *PPSN*, pages 357–366, 1998.

[71] R. Gämperle, S. D. Müller, and P. Koumoutsakos. A parameter study for differential evolution. In *Int. Conf. on Adv. in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pages 293–298, 2002.

[72] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, and A. M. Sánchez. Global and local real-coded genetic algorithms based on parent-centric crossover operators. *European Journal of Operational Research*, 185(3):1088–1113, 2008.

[73] C. García-Martínez, F. J. Rodríguez, and M. Lozano. Role differentiation and malleable mating for differential evolution: an analysis on large-scale optimisation. *Soft Comput.*, 15(11):2109–2126, 2011.

[74] S. Gelly, S. Ruette, and O. Teytaud. Comparison-Based Algorithms Are Robust and Randomized Algorithms Are Anytime. *Evol. Comput.*, 15(4):411–434, 2007.

[75] A. Ghosh, S. Das, A. Chowdhury, and R. Giri. An improved differential evolution algorithm with fitness-based adaptation of the control parameters. *Inf. Sci.*, 181(18):3749–3765, 2011.

[76] D. E. Goldberg, K. Deb, and D. Thierens. Toward a Better Understanding of Mixing in Genetic Algorithms. 計測と制御, 32:10–16, 1993.

[77] B. W. Goldman and W. F. Punch. Fast and Efficient Black Box Optimization Using the Parameter-less Population Pyramid. *Evolutionary Computation*, 23(3):451–479, 2015.

[78] W. Gong and Z. Cai. Differential Evolution With Ranking-Based Mutation Opera-

tors. *IEEE T. Cybernetics*, 43(6):2066–2081, 2013.

[79] W. Gong, Z. Cai, C. X. Ling, and H. Li. Enhanced differential evolution with adaptive strategies for numerical optimization. *IEEE Trans. on Systems, Man, and Cybernetics, PartB*, 41(2):397–413, 2011.

[80] W. Gong, A. Zhou, and Z. Cai. A Multioperator Search Strategy Based on Cheap Surrogate Models for Evolutionary Optimization. *IEEE Trans. Evolutionary Computation*, 19(5):746–758, 2015.

[81] J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transaction on Systems, Man and Cybernetics*, 16(1):122–128, 1986.

[82] A. Griewank and P. L. Toint. On the Unconstrained Optimization of Partially Separable Functions. In *Nonlinear Optimization 1981*, pages 301–312. Academic Press, 1982.

[83] S. Guo, J. S. Hong Tsai, C. Yang, and P. Hsu. A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set. In *IEEE CEC*, pages 1003–1010, 2015.

[84] S. Guo and C. Yang. Enhancing Differential Evolution Utilizing Eigenvector-Based Crossover Operator. *IEEE Trans. Evolutionary Computation*, 19(1):31–49, 2015.

[85] S. Guo, C. Yang, P. Hsu, and J. S. H. Tsai. Improving Differential Evolution With a Successful-Parent-Selecting Framework. *IEEE Trans. Evolutionary Computation*, 19(5):717–730, 2015.

[86] N. Hansen. Invariance, Self-Adaptation and Correlated Mutations and Evolution Strategies. In *PPSN*, pages 355–364, 2000.

[87] N. Hansen. The CMA evolution strategy: a comparing review. In J.A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[88] N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *GECCO (Companion)*, pages 2389–2396, 2009.

[89] N. Hansen. Benchmarking the nelder-mead downhill simplex algorithm with many local restarts. In *GECCO (Companion)*, pages 2403–2408, 2009.

[90] N. Hansen. GECCO BBOB. http://coco.gforge.inria.fr/doku.php, 2014.

[91] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA, 2012.

[92] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Posík. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *GECCO (Companion)*, pages 1689–1696, 2010.

[93] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization

benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009. Updated February 2010.

[94] N. Hansen and S. Kern. Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In *PPSN*, pages 282–291, 2004.

[95] N. Hansen and A. Ostermeier. Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In *ICEC*, pages 312–317, 1996.

[96] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001.

[97] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger. Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems. *Appl. Soft Comput.*, 11(8):5755–5769, 2011.

[98] G. R. Harik and F. G. Lobo. A parameter-less genetic algorithm. In *GECCO*, pages 258–265, 1999.

[99] F. Herrera, M. Lozano, and D. Molina. Components and parameters of de, real-coded chc, and g-cmaes. Technical report, Univ. of Granada, 2010.

[100] F. Herrera, M. Lozano, and D. Molina. Test suite for the spec. iss. of Soft Computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems. Technical report, Univ. of Granada, 2010.

[101] T. Higuchi, S. Tsutsui, and M. Yamamura. Theoretical Analysis of Simplex Crossover for Real-Coded Genetic Algorithms. In *PPSN*, pages 365–374, 2000.

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

[103] R. Hooke and T. A. Jeeves. "Direct Search" Solution of Numerical and Statistical Problems. *Journal of the ACM*, 8(2):212–229, 1961.

[104] S. Huband, P. Hingston, L. Barone, and R. L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.*, 10(5):477–506, 2006.

[105] B. Huberman, R. Lukose, and T. Hogg. An economics approach to hard computational problems. *Science*, 275(5296):51–54, 1997.

[106] F. Hutter, F. H. Hoos, and K. Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In *LION*, pages 507–523, 2011.

[107] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: An Automatic Algorithm Configuration Framework. *J. Artif. Intell. Res. (JAIR)*, 36:267–306, 2009.

[108] K. Ikeda and S. Kobayashi. GA based on the uv-structure hypothesis and its application to JSP. In *PPSN*, pages 273–282, 2000.

[109] L. Ingber and B. Rosen. Genetic algorithms and very fast simulated reannealing: A comparison. *Mathematical and Computer Modelling*, 16(11):87–100, 1992.

[110] Sk. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan. An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization. *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, 42(2):482–500, 2012.

[111] G. A. Jastrebski and D. V. Arnold. Improving evolution strategies through active covariance matrix adaptation. In *IEEE CEC*, pages 2814–2821, 2006.

[112] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput.*, 9(1):3–12, 2005.

[113] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *ICGA*, pages 184–192, 1995.

[114] K. De Jong. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan, 1975.

[115] R. Joshi and A. C. Sanderson. Minimal representation multisensor fusion using differential evolution. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 29(1):63–76, 1999.

[116] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *J. Global Optimization*, 39(3):459–471, 2007.

[117] D. Karaboga and B. Basturk. On the performance of artificial bee colony (ABC) algorithm. *Appl. Soft Comput.*, 8(1):687–697, 2008.

[118] G. Karafotias, M. Hoogendoorn, and A. E. Eiben. Why parameter control mechanisms should be benchmarked against random variation. In *IEEE CEC*, pages 349–355, 2013.

[119] G. Karafotias, M. Hoogendoorn, and A. E. Eiben. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Trans. Evol. Comput.*, 19(2):167–187, 2015.

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

[121] S. Kirkpatrick, D. Gelatt Jr., and M. P. Vecchi. Optimization by Simmulated Annealing. *Science*, 220(4598):671–680, 1983.

[122] T. G. Kolda, R. Michael Lewis, and V. Torczon. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. *SIAM Review*, 45(3):385–482, 2003.

[123] V. K. Koumousis and C. P. Katsaras. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Trans. Evol. Comput.*, 10(1):19–28, 2006.

[124] F. Kursawe. A variant of evolution strategies for vector optimization. In *PPSN*, pages 193–197. Springer, 1991.

[125] 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*, 9(1):112–147, 1998.

[126] W. B. Langdon and R. Poli. Evolving Problems to Learn About Particle Swarm Optimizers and Other Search Algorithms. *IEEE Trans. Evol. Comput.*, 11(5):561–578, 2007.

[127] J. L. J. Laredo, C. Fernandes, J. J. Merelo Guervós, and C. Gagné. Improving Genetic Algorithms Performance via Deterministic Population Shrinkage. In *GECCO*, pages 819–826, 2009.

[128] A. LaTorre, S. Muelas, and J. M. Peña. A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. *Soft Comput.*, 15(11):2187–2199, 2011.

[129] M. N. Le, Y. Ong, S. Menzel, Y. Jin, and B. Sendhoff. Evolution by adapting surrogates. *Evol. Comput.*, 21(2):313–340, 2013.

[130] R. H. Leary. Global Optimization on Funneling Landscapes. *J. Global Optimization*, 18(4):367–383, 2000.

[131] M. A. Lee and H. Takagi. Dynamic Control of Genetic Algorithms Using Fuzzy Logic Techniques. In *ICGA*, pages 76–83, 1993.

[132] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin. Benchmark functions for the cec 2013 special session and competition on large-scale global optimization. Technical report, Univ. of Science and Technology of China, 2013.

[133] X. Li and M. Yin. Modified differential evolution with self-adaptive parameters method. *Journal of Combinatorial Optimization*, pages 1–31, 2014.

[134] Y. Li and J. Zhang. A new differential evolution algorithm with dynamic population partition and local restart. In *GECCO*, pages 1085–1092, 2011.

[135] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3):281–295, 2006.

[136] J. J. Liang, B. Y. Qu, and P. N. Suganthan. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical report, Zhengzhou Univ. and Nanyang Technological Univ., 2013.

[137] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz. Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. Technical report, Nanyang Technological Univ., 2013.

[138] J. J. Liang, P. N. Suganthan, and K. Deb. Novel Composition Test Functions for Numerical Global Optimization. In *Swarm Intell. Symp.*, pages 68–75, 2005.

[139] T. Liao, D. Aydin, and T. Stützle. Artificial bee colonies for continuous optimization:

Experimental analysis and improvements. *Swarm Intell.*, 7(4):327–356, 2013.

[140] C. Lin, A. Qing, and Q. Feng. A comparative study of crossover in differential evolution. *J. Heuristics*, 17(6):675–703, 2011.

[141] B. Liu, Q. Zhang, and G. G. E. Gielen. A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. *IEEE Trans. Evol. Comput.*, 18(2):180–192, 2014.

[142] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(1-3):503–528, 1989.

[143] J. Liu and J. Lampinen. A fuzzy adaptive differential evolution algorithm. *Soft Comput.*, 9(6):448–462, 2005.

[144] X. Fang Liu, Z. Zhan, and J. Zhang. Dichotomy Guided Based Parameter Adaptation for Differential Evolution. In *GECCO*, pages 289–296, 2015.

[145] F. G. Lobo and C. F. Lima. A review of adaptive population sizing schemes in genetic algorithms. In *GECCO*, pages 228–234, 2005.

[146] F. G. Lobo, C. F. Lima, and Z. Michalewicz, editors. *Parameter setting in evolutionary algorithms*. Studies in Computational Intelligence. Springer, 2007.

[147] M. Locatelli. Convergence of a Simulated Annealing Algorithm for Continuous Global Optimization. *J. Global Optimization*, 18(3):219–233, 2000.

[148] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace Package: Iterated Racing for Automatic Algorithm Configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[149] I. Loshchilov, M. Schoenauer, and M. Sebag. Alternative Restart Strategies for CMA-ES. In *PPSN*, pages 296–305, 2012.

[150] I. Loshchilov, M. Schoenauer, and M. Sebag. Bi-population CMA-ES agorithms with surrogate models and line searches. In *GECCO (Companion)*, pages 1177–1184, 2013.

[151] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina. Real-coded memetic algorithms with crossover hill-climbing. *Evol. Comput.*, 12(3):273–302, September 2004.

[152] M. Lunacek and D. Whitley. The dispersion metric and the CMA evolution strategy. In *GECCO*, pages 477–484, 2006.

[153] M. Lunacek, D. Whitley, and A. M. Sutton. The Impact of Global Structure on Search. In *Parallel Problem Solving from Nature - PPSN X, 10th International Conference Dortmund, Germany, September 13-17, 2008, Proceedings*, pages 498–507, 2008.

[154] C. MacNish. Towards unbiased benchmarking of evolutionary and hybrid algorithms for real-valued optimisation. *Connect. Sci.*, 19(4):361–385, 2007.

[155] S. W. Mahfoud. A comparison of parallel and sequential niching methods. In *ICGA*, pages 136–143, 1995.

[156] R. Mallipeddi and P. N. Suganthan. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. Technical report, Nanyang Technological University, 2010.

[157] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11(2):1679–1696, 2011.

[158] R. Mendes, J. Kennedy, and J. Neves. The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Trans. Evol. Comput.*, 8(3):204–210, 2004.

[159] O. Mersmann, M. Preuss, H. Trautmann, B. Bischl, and C. Weihs. Analyzing the BBOB Results by Means of Benchmarking Concepts. *Evol. Comput.*, 23(1):161–185, 2015.

[160] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[161] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello. A comparative study of differential evolution variants for global optimization. In *GECCO*, pages 485–492, 2006.

[162] D. Molina, M. Lozano, C. García-Martínez, and F. Herrera. Memetic Algorithms for Continuous Optimisation Based on Local Search Chains. *Evolutionary Computation*, 18(1):27–63, 2010.

[163] D. Molina, M. Lozano, and F. Herrera. A memetic algorithm using local search chaining forblack-box optimization benchmarking 2009 for noise free functions. In *GECCO (Companion)*, pages 2255–2262, 2009.

[164] C. K. Monson and K. D. Seppi. Exposing origin-seeking bias in PSO. In *GECCO*, pages 241–248, 2005.

[165] R. Myers and E. R. Hancock. Empirical modelling of genetic algorithms. *Evolutionary Computation*, 9(4):461–493, 2001.

[166] V. Nannen and A. E. Eiben. Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. In *IJCAI*, pages 975–980, 2007.

[167] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computation Journal*, 7:308–313, 1965.

[168] M. Nicolau. Application of a simple binary genetic algorithm to a noiseless testbed benchmark. In *GECCO (Companion)*, pages 2473–2478, 2009.

[169] N. Noman, D. Bollegala, and H. Iba. An adaptive differential evolution algorithm. In *IEEE CEC*, pages 2229–2236, 2011.

[170] N. Noman and H. Iba. Accelerating Differential Evolution Using an Adaptive Local Search. *IEEE Tran. Evol. Comput.*, 12(1):107–125, 2008.

[171] S. Obayashi, D. Sasaki, Y. Takeguchi, and N. Hirose. Multiobjective evolutionary

computation for supersonic wing-shape optimization. *IEEE Trans. Evol. Comput.*, 4(2):182–187, 2000.

[172] M. N. Omidvar, X. Li, Y. Mei, and X. Yao. Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization. *IEEE Trans. Evolutionary Computation*, 18(3):378–393, 2014.

[173] M. G. H. Omran, A. A. Salman, and A. P. Engelbrecht. Self-adaptive Differential Evolution. In *CIS*, pages 192–199, 2005.

[174] Y. S. Ong, P. B. Nair, and A. J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal*, 41(4):687–696, 2003.

[175] I. Ono and S. Kobayashi. A real coded genetic algorithm for function optimization using unimodal normal distributed crossover. In *ICGA*, pages 246–253, 1997.

[176] I. Ono, S. Kobayashi, and K. Yoshida. Optimal lens design by real-coded genetic algorithms using UNDX. *Computer methods in applied mechanics and engineering*, 186(2):483–497, 2000.

[177] M. Pelikan, D. E. Goldberg, and S. Tsutsui. Combining The Strengths Of Bayesian Optimization Algorithm And Adaptive Evolution Strategies. In *GECCO*, pages 512–519, 2002.

[178] P. Pellegrini, T. Stützle, and M. Birattari. A critical analysis of parameter adaptation in ant colony optimization. *Swarm Intelligence*, 6(1):23–48, 2012.

[179] F. Peng, K. Tang, G. Chen, and X. Yao. Multi-start JADE with knowledge transfer for numerical optimization. In *IEEE CEC*, pages 1889–1895, 2009.

[180] F. Peng, K. Tang, G. Chen, and X. Yao. Population-based algorithm portfolios for numerical optimization. *IEEE Trans. Evol. Comput.*, 14(5):782–800, October 2010.

[181] R. Polakova, J. Tvrdík, and P. Bujok. Cooperation of optimization algorithms: A simple hierarchical model. In *IEEE CEC*, pages 1046–1052, 2015.

[182] P. Pošík. Bbob-benchmarking the generalized generation gap model with parent centric crossover. In *GECCO (Companion)*, pages 2321–2328, 2009.

[183] P. Posík. Bbob-benchmarking two variants of the line-search algorithm. In *GECCO (Companion)*, pages 2329–2336, 2009.

[184] P. Posík and W. Huyer. Restarted local search algorithms for continuous black box optimization. *Evolutionary Computation*, 20(4):575–607, 2012.

[185] P. Pošík and V. Klema. JADE, an adaptive differential evolution algorithm, benchmarked on the BBOB noiseless testbed. In *GECCO (Companion)*, pages 197–204, 2012.

[186] M. Powell. The NEWUOA software for unconstrained optimization without derivatives. In *Large-scale nonlinear optimization*, pages 255–297. 2006.

[187] K. Price. Differential evolution vs. the functions of the 2 nd ICEO. In *ICEC*, pages

153–157, 1997.

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

[189] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Tran. Evol. Comput.*, 13(2):398–417, 2009.

[190] A. K. Qin, F. Raimondo, F. Forbes, and Y. Ong. An improved CUDA-based implementation of differential evolution on GPU. In *GECCO*, pages 991–998, 2012.

[191] A. Kai Qin and Ponnuthurai N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *Congress on Evolutionary Computation*, pages 1785–1791, 2005.

[192] A. Qing. *Differential Evolution: Fundamentals and Applications in Electrical Engineering*. Wiley-IEEE Press, 2009.

[193] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama. Opposition-based differential evolution. *IEEE Tran. Evol. Comput.*, 12(1):64–79, 2008.

[194] T. Robič and B. Filipič. DEMO: differential evolution for multiobjective optimization. In *EMO*, pages 520–533, 2005.

[195] P. Rocca, G. Oliveri, and A. Massa. Differential evolution as applied to electromagnetics. *IEEE Antennas and Propagation Magazine*, 53(1):38–49, 2011.

[196] J. Rönkkönen, S. Kukkonen, and K. V. Price. Real-Parameter optimization with Differential Evolution. In *IEEE CEC*, pages 506–513, 2005.

[197] R. Ros. Benchmarking the BFGS algorithm on the BBOB-2009 function testbed. In *GECCO (Companion)*, pages 2409–2414, 2009.

[198] R. Ros. Benchmarking the NEWUOA on the BBOB-2009 function testbed. In *GECCO (Companion)*, pages 2421–2428, 2009.

[199] R. Ros and N. Hansen. A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity. In *PPSN*, pages 296–305, 2008.

[200] H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[201] S. Rudlof and M. Köppen. Stochastic hill climbing with learning by vectors of normal distributions. In *WSC*, pages 60–70, 1996.

[202] K. M. Sallam, R. A. Sarker, D. Essam, and S. M. Elsayed. Neurodynamic differential evolution algorithm and solving CEC2015 competition problems. In *IEEE CEC*, pages 1033–1040, 2015.

[203] R. Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39(3):263–278, 1996.

[204] H. Sato, C. A. Coello Coello, H. E. Aguirre, and K. Tanaka. Adaptive Control of

the Number of Crossed Genes in Many-Objective Evolutionary Optimization. In *LION*, pages 478–484, 2012.

[205] H. Satoh, M. Yamamura, and S. Kobayashi. Minimal generation gap model for gas considering both exploration and exploitation. In *the IIZUKA: Methodologies for the Conception, Design, and Application of Intelligent Systems*, page 494497, 1996.

[206] M. Sebag and A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In *PPSN*, pages 418–427, 1998.

[207] C. Segura, C. A. C. Coello, E. Segredo, and C. León. On the adaptation of the mutation scale factor in differential evolution. *Optimization Letters*, 9(1):189–198, 2015.

[208] S. Shan and G. G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41(2):219–241, 2010.

[209] Y. Shang and Y. Qiu. A Note on the Extended Rosenbrock Function. *Evolutionary Computation*, 14(1):119–126, 2006.

[210] Y. Shi and R. Eberhart and. A modified particle swarm optimizer. In *IEEE CEC*, pages 69–73, 1998.

[211] Y. Shi and R. C. Eberhart. Empirical study of particle swarm optimization. In *IEEE CEC*, volume 3, pages 101–106, 1999.

[212] S. Shirakawa, N. Yata, and T. Nagao. Evolving search spaces to emphasize the performance difference of real-coded crossovers using genetic programming. In *IEEE CEC*, pages 1–8, 2010.

[213] S. K. Smit and A. E. Eiben. Parameter Tuning of Evolutionary Algorithms: Generalist vs. Specialist. In *EvoApplicatons*, pages 542–551, 2010.

[214] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173, 2008.

[215] F. Solis and R. J-B. Wets. Minimization by random search techniques. *Mathematics of operations research*, 6(1):19–30, 1981.

[216] W. M. Spears. Crossover or mutation? In *FOGA*, pages 221–237, 1992.

[217] W. M. Spears. Adapting crossover in evolutionary algorithms. In *Evolutionary Programming*, pages 367–384, 1995.

[218] M. Srinivas and L. M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. on Systems, Man, and Cybernetics*, 24(4):656–667, 1994.

[219] R. Storn. Differential evolution design of an iir-filter. In *International Conference on Evolutionary Computation*, pages 268–273, 1996.

[220] R. Storn and K. Price. Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, Interna-

tional Computer Science Institute, Berkeley, CA, 1995.

[221] R. Storn and K. Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Global Optimiz.*, 11(4):341–359, 1997.

[222] P. N. Suganthan. Particle swarm optimiser with neighbourhood operator. In *IEEE CEC*, pages 1958–1962, 1999.

[223] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical report, Nanyang Technological Univ., 2005.

[224] A. M. Sutton, M. Lunacek, and L. D. Whitley. Differential evolution and non-separability: using selective pressure to focus search. In *GECCO*, pages 1428–1435, 2007.

[225] S. Swarzberg, G. Seront, and H. Bersini. STEP: The easiest way to optimize a function. In *IEEE CEC*, pages 519–524, 1994.

[226] Y. Tan and Y. Zhu. Fireworks algorithm for optimization. In *ICSI (1)*, pages 355–364, 2010.

[227] R. Tanabe and A. Fukunaga. Success-History Based Parameter Adaptation for Differential Evolution. In *IEEE CEC*, pages 71–78, 2013.

[228] R. Tanabe and A. Fukunaga. Reevaluating Exponential Crossover in Differential Evolution. In *PPSN*, pages 201–210, 2014.

[229] R. Tanabe and A. S. Fukunaga. Improving the search performance of SHADE using linear population size reduction. In *IEEE CEC*, pages 1658–1665, 2014.

[230] R. Tanabe and A. S. Fukunaga. On the pathological behavior of adaptive differential evolution on hybrid objective functions. In *GECCO*, pages 1335–1342, 2014.

[231] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise. Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization. Technical report, Univ. of Science and Technology of China, 2010.

[232] L. Tang, Y. Dong, and J. Liu. Differential Evolution With an Individual-Dependent Mechanism. *IEEE Trans. Evolutionary Computation*, 19(4):560–574, 2015.

[233] J. Teo. Exploring dynamic self-adaptive populations in differential evolution. *Soft Comput.*, 10(8):673–686, 2006.

[234] M. Tezuka, M. Munetomo, and K. Akama. Linkage Identification by Nonlinearity Check for Real-Coded Genetic Algorithms. In *GECCO*, pages 222–233, 2004.

[235] D. Thierens. Adaptive mutation rate control schemes in genetic algorithms. In *IEEE CEC*, pages 980–985, 2002.

[236] D. Thierens. Adaptive Strategies for Operator Allocation. In *Parameter Setting in Evolutionary Algorithms*, pages 77–90. 2007.

[237] L. Tseng and C. Chen. Multiple trajectory search for large scale global optimization. In *IEEE CEC*, pages 3052–3059, 2008.

[238] S. Tsutsui. Sampling bias and search space boundary extension in real coded genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Las Vegas, Nevada, USA, July 8-12, 2000*, pages 211–218, 2000.

[239] S. Tsutsui. Probabilistic Model-Building Genetic Algorithms in Permutation Representation Domain Using Edge Histogram. In *PPSN*, pages 224–233, 2002.

[240] S. Tsutsui. *c*AS: Ant Colony Optimization with Cunning Ants. In *PPSN*, pages 162–171, 2006.

[241] S. Tsutsui and N. Fujimoto. Solving quadratic assignment problems by genetic algorithms with GPU computation: a case study. In *GECCO (Companion)*, pages 2523–2530, 2009.

[242] S. Tsutsui, A. Ghosh, D. Corne, and Y. Fujimoto. A Real Coded Genetic Algorithm with an Explorer and an Exploiter Populations. In *ICGA*, pages 238–245, 1997.

[243] S. Tsutsui, M. Yamamura, and T. Higuchi. Multi-parent recombination with simplex crossover in real coded genetic algorithms. In *GECCO*, volume 1, pages 657–664, 1999.

[244] J. Tvrdık. Competitive Differential Evolution. In *MENDEL*, pages 7–12, 2006.

[245] J. A. Vrugt, B. A. Robinson, and J. M. Hyman. Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces. *IEEE Trans. Evolutionary Computation*, 13(2):243–259, 2009.

[246] H. Wang, Z. Wu, and S. Rahnamayan. Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. *Soft Comput.*, 15(11):2127–2140, 2011.

[247] Y. Wang, Z. Cai, and Q. Zhang. Differential Evolution With Composite Trial Vector Generation Strategies and Control Parameters. *IEEE Tran. Evol. Comput.*, 15(1):55–66, 2011.

[248] Y. Wang, Z. Cai, and Q. Zhang. Enhancing the search ability of differential evolution through orthogonal crossover. *Inf. Sci.*, 185(1):153–177, 2012.

[249] Y. Wang, H. Li, T. Huang, and L. Li. Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. *Appl. Soft Comput.*, 18:232–247, 2014.

[250] M. Weber, F. Neri, and V. Tirronen. Distributed differential evolution with explorative—exploitative population families. *GPEM*, 10(4):343–371, 2009.

[251] M. Weber, F. Neri, and V. Tirronen. Shuffle or update parallel differential evolution for large-scale optimization. *Soft Comput.*, 15(11):2089–2107, 2011.

[252] D. Whitley, K. Mathias, S. Rana, and J. Dzubera. Evaluating evolutionary algorithms. *Artificial Intelligence*, 85:245–276, 1996.

[253] D. Wilson, S. Cussat-Blanc, K. Veeramachaneni, U. O'Reilly, and H. Luga. A continuous developmental model for wind farm layout optimization. In *GECCO*, pages 745–752, 2014.

[254] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transaction on Evolutionary Computation*, 1(1):67–82, 1997.

[255] M. Yang, C. Li, Z. Cai, and J. Guan. Differential evolution with auto-enhanced population diversity. *IEEE T. Cybernetics*, 45(2):302–315, 2015.

[256] Z. Yang, K. Tang, and X. Yao. Large scale evolutionary optimization using cooperative coevolution. *Inf. Sci.*, 178(15):2985–2999, 2008.

[257] Z. Yang, K. Tang, and X. Yao. Self-adaptive differential evolution with neighborhood search. In *IEEE CEC*, pages 1110–1116, 2008.

[258] Z. Yang, K. Tang, and X. Yao. Scalability of generalized adaptive differential evolution for large-scale continuous optimization. *Soft Comput.*, 15(11):2141–2155, 2011.

[259] Z. Yang, X. Yao, and J. He. Making a difference to differential evolution. In *Advances in Metaheuristics for Hard Optimization*, pages 397–414. 2008.

[260] X. Yao, Y. Liu, and G. Lin. Evolutionary Programming Made Faster. *IEEE Tran. Evol. Comput.*, 3(2):82–102, 1999.

[261] W. Yu, J. Li, J. Zhang, and M. Wan. Differential evolution using mutation strategy with adaptive greediness degree control. In *GECCO*, pages 73–80, 2014.

[262] D. Zaharie. Control of population diversity and adaptation in differential evolution algorithms. In *MENDEL*, pages 41–46, 2003.

[263] D. Zaharie. Influence of crossover on the behavior of Differential Evolution Algorithms. *Appl. Soft Comput.*, 9(3):1126–1138, 2009.

[264] M. Zhabitsky and E. Zhabitskaya. Asynchronous Differential Evolution with Adaptive Correlation Matrix. In *GECCO*, pages 455–462, 2013.

[265] Z. Zhan, J. Zhang, Y. Li, and H. S. Chung. Adaptive Particle Swarm Optimization. *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, 39(6):1362–1381, 2009.

[266] J. Zhang and A. Sanderson. *Adaptive differential evolution: a robust approach to multimodal problem optimization*, volume 1. Springer, 2009.

[267] J. Zhang and A. C. Sanderson. JADE: Adaptive Differential Evolution With Optional External Archive. *IEEE Tran. Evol. Comput.*, 13(5):945–958, 2009.

[268] S. Zhao and P. N. Suganthan. Empirical investigations into the exponential crossover of differential evolutions. *Swarm and Evol. Comput.*, 9:27–36, 2013.

[269] S. Zhao, P. N. Suganthan, and S. Das. Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Comput.*, 15(11):2175–2185, 2011.

[270] S. Zheng, A. Janecek, and Y. Tan. Enhanced fireworks algorithm. In *IEEE Congress on Evolutionary Computation*, pages 2069–2077, 2013.

[271] Y. Zheng, L. Ma, L. Zhang, and J. Qian. Empirical study of particle swarm optimizer with an increasing inertia weight. In *IEEE CEC*, pages 221–226, 2003.

[272] W. Zhong, J. Liu, M. Xue, and L. Jiao. A multiagent genetic algorithm for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(2):1128–1141, 2004.

[273] K. Zielinski, X. Wang, and R. Laur. Comparison of Adaptive Approaches for Differential Evolution. In *PPSN*, pages 641–650, 2008.

[274] K. Zielinski, P. Weitkemper, R. Laur, and K. D. Kammeyer. Parameter study for differential evolution using a power allocation problem including interference cancellation. In *IEEE CEC*, pages 1857–1864, 2006.

[275] S. Zilberstein. Using Anytime Algorithms in Intelligent Systems. *AI Magazine*, 17(3):73–83, 1996.

[276] 田邊遼司 and 福永 Alex. DE における Exponential Crossover の再評価. **進化計算学会論文誌**, 6(1):42–52, 2015.

[277] 相吉英太郎, 岡本卓, and 安田恵一郎. **最適化手法の基礎 – 力学モデルによる理解と実装–**. 森北出版, 2014.

[278] 茨木俊秀. **最適化の数学**. 共立出版, 2011.

[279] 金谷健一. **これなら分かる最適化数学 – 基礎原理から計算手法まで**. 共立出版, 2005.

付録 A

関数最適化問題に対する伝統的な探索手法

本章では関数最適化問題に対する伝統的な探索手法について述べる．始めに, A.1 節にて目的関数 f の微分情報を用いた探索手法について述べる．次に, A.2 節にて目的関数値 $f(\boldsymbol{x})$ のみを用いて探索を行う手法を説明する．

A.1　勾配ベクトルやヘッセ行列を用いた探索手法

本節では, 無制約非線形最適化問題に対する目的関数 f の勾配を用いた伝統的な探索手法について述べる．始めに A.1.1 節にて最急降下法について述べ, A.1.2 節にてニュートン法を説明する．その後, A.1.3 節にて準ニュートン法について述べる．なお, 本章の各探索手法の記述は [277, 279, 278] を参考にした．

A.1.1　最急降下法

最急降下法 (Steepest descent method) は, 現探索点 $\boldsymbol{x}^t$ から目的関数値が最小となる方向を目的関数の勾配を用いて求め, その方向へと探索点を移動する操作を収束するまで繰り返す手法である．具体的には, ある探索点 $\boldsymbol{x}$ において, 目的関数値を最小化する方向が, 勾配ベクトル $\nabla f(\boldsymbol{x})$ の逆方向である性質を利用する．

探索の開始時 $t = 1$ において, 何らかの方法で初期点 $\boldsymbol{x}^t$ を与える．その後, 各反復 t において, $\boldsymbol{x}^t$ における目的関数の勾配ベクトル $\nabla f(\boldsymbol{x}^t)$ を用いて, 新たな探索点を得る:

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + \alpha \left(-\nabla f(\boldsymbol{x}^t)\right) \tag{A.1}$$

ここで, α はステップサイズであり, $\alpha \geq 0$ である．ステップサイズ α は可能な限り大きな値に設定することが好ましいが, 不適切な設定であった場合, 探索が収束しない可能性がある．一方, α を過小に設定してしまうと, 収束速度が遅くなり, 探索終了までに費やす解評価回数が増加してしまう恐れがある．そこで, 式 (A.1) 中の $-\nabla f(\boldsymbol{x}^t)$ の方向の直線上にて, 目的関数値が最小となるような α を直線探索問題として扱い, 二分法や黄金分割法といった直線探索法に

よって近似的に求める方法が一般的である.

$\nabla f(\boldsymbol{x}^t) = \boldsymbol{0}$ となるまで, 以上の操作を繰り返す. 最急降下法の実行終了時の探索点 $\boldsymbol{x}^t$ は, 初期探索点 $\boldsymbol{x}^1$ を適切に与えた場合に限り, 局所解である保証を持つ. なお, 最急降下法の収束は, 各反復毎に目的関数値が一定の割合で減少する一次収束である.

A.1.2　ニュートン法

前節で説明した最急降下法は勾配ベクトル $\nabla f(\boldsymbol{x})$ を利用していたが, ニュートン法ではヘッセ行列の逆行列 $\nabla^2 f(\boldsymbol{x})^{-1}$ を用いて探索を行う. 目的関数 f を現探索点 $\boldsymbol{x}^t$ の周りで二次のテイラー展開を行うと, 次式が得られる:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^t) + \nabla f(\boldsymbol{x}^t)^T(\boldsymbol{x} - \boldsymbol{x}^t) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^t)\nabla^2 f(\boldsymbol{x}^t)(\boldsymbol{x} - \boldsymbol{x}^t) \tag{A.2}$$

式 (A.2) の左辺を $f_{II}(\boldsymbol{x})$ と置くと, $f_{II}(\boldsymbol{x})$ の最適解は元の目的関数 f の高精度な近似であると考えられる. ヘッセ行列が正定値であるとき, $f_{II}(\boldsymbol{x})$ の最適解は, 式 (A.2) を $\boldsymbol{x}^t$ で偏微分することにより求まり, $\boldsymbol{x}^t - \nabla^2 f(\boldsymbol{x})^{-1}\nabla f(\boldsymbol{x})$ である.

ニュートン法では, この性質を利用して探索を行う. 探索の開始時 $t = 1$ において, 何らかの方法で初期点 $\boldsymbol{x}^t$ を与える. その後, 各反復 t において, $\boldsymbol{x}^t$ における目的関数の勾配ベクトル $\nabla f(\boldsymbol{x}^t)$ とヘッセ行列の逆行列 $\nabla^2 f(\boldsymbol{x})^{-1}$ を用いて, 新たな探索点を得る:

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \nabla^2 f(\boldsymbol{x})^{-1}\nabla f(\boldsymbol{x}) \tag{A.3}$$

この式 (A.3) による探索点の更新を, 終了条件を満たすまで繰り返す.

ニュートン法は 2 次収束性を持つため, 一次収束の最急降下法と比べ収束速度が非常に速い. ここで, 2 次収束性とは $t + 1$ 回目の反復で得られる解 $\boldsymbol{x}^{t+1}$ の近似精度が, t 回目の $\boldsymbol{x}^t$ と比べ 2 乗ほど改善される収束である. また, 最急降下法のようにステップサイズ α を求めるために直線探索問題を解く必要が無い.

しかし, ニュートン法は局所解への収束性は保証されていない. また, ヘッセ行列 $\nabla^2 f(\boldsymbol{x})$ の逆行列 $\nabla^2 f(\boldsymbol{x})^{-1}$ を求めるため, $\nabla^2 f(\boldsymbol{x})$ が正則である必要がある. さらに, 逆行列の算出にかかる計算量は $O(D^3)$ であるため, 高次元関数においては実行時間が問題となる.

A.1.3　準ニュートン法

前節で述べたように, 対象問題によってはニュートン法は適用することが困難となる場合がある. このような背景から, ヘッセ行列の逆行列 $\nabla^2 f(\boldsymbol{x})^{-1}$ の近似行列 $\boldsymbol{H}$ を作成し, 各反復毎に更新することで探索を行う準ニュートン法が提案された. 準ニュートン法では探索中に得られた探索点, 及び勾配ベクトルを用いて近似行列 $\boldsymbol{H}$ を構築する構築する.

まず, 探索開始時において, $\boldsymbol{H}^1 = \boldsymbol{I}$ とし, 初期探索点 $\boldsymbol{x}^1$ を何らかの方法で生成する. ここで, $\boldsymbol{I}$ は単位行列である. t 回目の反復において, $t + 1$ の探索点 $\boldsymbol{x}^{t+1}$ を次式を用いて求める.

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \alpha\boldsymbol{H}^t\nabla f(\boldsymbol{x}) \tag{A.4}$$

ここで, 式 (A.1) と同様に $\alpha \geq 0$ はステップサイズであり, 直線探索問題として扱い, 何らかの方法を適用することにより求める. 目的関数 f が微分不可能であり勾配ベクトル $\nabla f(\boldsymbol{x})$ が利用できない場合, 有限差分法を用いて $\nabla f(\boldsymbol{x}) \approx \frac{f(\boldsymbol{x}+\Delta\boldsymbol{x})-f(\boldsymbol{x})}{\Delta\boldsymbol{x}}$ などと推定する [25].

新たな探索点を生成し, 目的関数により評価した後, 近似行列 $\boldsymbol{H}^t$ の更新を行う. 更新則には, 階数 1 公式, Davidon, Fletcher, Powell の公式 (DFP 公式) や Broyden Fletcher Goldfarb Shanno の公式 (BFGS 公式) などが存在するが, 最も一般的な BFGS 公式は次式により与えられる:

$$\boldsymbol{H}^{t+1} = \boldsymbol{H}^t + \left(\frac{\boldsymbol{p}^t (\boldsymbol{p}^t)^T}{(\boldsymbol{p}^t)^T \boldsymbol{q}^t} - \frac{\boldsymbol{H}^t \boldsymbol{q}^t (\boldsymbol{q}^t)^T \boldsymbol{H}^t}{(\boldsymbol{q}^t)^T \boldsymbol{H}^t \boldsymbol{q}^t} \right) \tag{A.5}$$

ここで, $\boldsymbol{p}^t = \boldsymbol{x}^{t+1} - \boldsymbol{x}^t$, $\boldsymbol{q}^t = \nabla f(\boldsymbol{x}^{t+1}) - \nabla f(\boldsymbol{x}^t)$ である. 以上に述べたとおり, 準ニュートン法では反復 t ごとに式 (A.4) による探索点 $\boldsymbol{x}^t$ の更新, 式 (A.5) による近似行列 $\boldsymbol{H}^t$ の更新を繰り返す.

初期探索点 $\boldsymbol{x}^1$ を適切に与えた場合, 準ニュートン法の実行終了時の探索点 $\boldsymbol{x}^t$ は, 局所解である保証を持つ. また, 初期近似行列 $\boldsymbol{H}^1$ が最適解におけるヘッセ行列の逆行列に一致し, かつそれが正則行列であった場合に限り, 準ニュートン法の局所解への収束は 2 次収束に近い超一次収束である. しかし, 実問題においてはそのように $\boldsymbol{H}^1$ を初期化することは困難である.

A.2　関数最適化問題に対する伝統的な直接探索法

2.3.1 節では Nelder-Mead 法 [167] を説明したが, 本節ではその他の目的関数 f の勾配を利用せずに目的関数値 $f(\boldsymbol{x})$ のみを利用して探索を行う手法について述べる. 始めに A.2.1 節にてランダムサーチ [31] について述べる. 次に, A.2.2 節では Hooke-Jeeves 法 [103] について説明する. 最後に疑似焼きなまし法 [121] について述べる.

A.2.1　Random Search

最も単純な確率的最適化方法としてランダムサーチ (Random Search) [31] が考えられる. ランダムサーチでは, 各反復 t 毎に新たな解 $\boldsymbol{x}^t$ を実行可能領域 $\boldsymbol{S}$ 内に一様ランダムに生成し, 最良解 $\boldsymbol{x}^{bsf}$ よりも目的関数値が低ければ, $\boldsymbol{x}^{bsf} = \boldsymbol{x}^t$ とする. この操作を終了条件を満たすまで繰り返す.

最適値 $f(\boldsymbol{x}^*)$ と解 $\boldsymbol{x}$ の誤差値は $|f(\boldsymbol{x}) - f(\boldsymbol{x}^*)|$ であり, $\epsilon > 0$ を許容誤差値とする. 最適解 $\boldsymbol{x}^*$ が実行可能領域 $\boldsymbol{S}$ に存在する場合, $\boldsymbol{x} \in \boldsymbol{S}$ となる解 $\boldsymbol{x}$ をランダムサーチは探索状況に依存せずに生成可能である. そのため, 最大評価回数を際限なく与えた場合, ランダムサーチは任意の $\epsilon > 0$ に対して $|f(\boldsymbol{x}) - f(\boldsymbol{x}^*)| < \epsilon$ となる解 $\boldsymbol{x}$ を必ず求めることができる.

ランダムサーチは極めて単純な探索手法であり, 無限に探索資源を与えた場合は最適解を必ず求めることができる. しかし, 実際に最大評価回数を無限に設定することは非現実的であり, その場合, 他の探索手法と比べてランダムサーチの性能は大きく劣る.

A.2.2　Hooke-Jeeves 法

1961 年に Hooke と Jeeves により提案された Hooke-Jeeves 法 [103] は, 勾配情報を利用しない決定的探索アルゴリズムである.

Hooke-Jeeves 法では各反復 t において, 探索点 $\boldsymbol{x}^t$ に決定変数ごとに摂動を加えることで新たな探索点 $\boldsymbol{u}^t$ を生成する. そして, 新たな探索点 $\boldsymbol{u}^t$ が現在の探索点 $\boldsymbol{x}^t$ よりも目的関数値が改善されている場合, $\boldsymbol{x}^t = \boldsymbol{u}^t$ とする. 全ての各決定変数に摂動を加えて $\boldsymbol{u}^t$ を生成したにも関わらず, 目的関数値が改善されない場合は, 摂動が大きいために適切な $\boldsymbol{u}^t$ を生成できなかったと判断し, 摂動の大きさを制御するステップサイズ $\alpha \in (0, 1]$ を減少させる. 反対に, 目的関数値の改善が見られた場合, 探索の収束の向上のため, 全ての決定変数に摂動を与える. この手順を終了条件を満たすまで繰り返す. 以上に述べた Hooke-Jeeves 法を, Algorithm 23 に示す. ここで, β はステップサイズ α の縮小率であり, 推奨値は $\beta = 0.5$ である.

Hooke-Jeeves 法は, 各決定変数ごとに摂動を加えるため, 変数分離化な関数を比較的に効率よく探索することができる. そのため, 1961 年に開発された手法であるにも関わらず, Multiple Trajectory Search Local Search 1 (MTS-LS1) [237] などの Hooke-Jeeves 法を基にした手法が, 近年でも提案されている. しかし, Hooke-Jeeves 法ではステップサイズ α を縮小することで探索状況に適した α へと調整するが, 縮小した α を再度拡大する機能を持たないため, 一度局所解に誤って収束してしまうと探索が停滞してしまう恐れがある.

A.2.3　Simulated Annealing (SA)

1983 年に Kirkpatrick らに提案された Simulated Annealing (SA, 疑似焼きなまし法) [121] は, 物理現象である焼きなましに着想を得た確率的最適化手法である. ここで, 焼きなましとは加熱した金属を徐々に冷却していくことで結晶体の欠陥を減らす方法であり, 計算機上でシミュレートする方法として Metropolis Monte Carlo 法 [160] が知られている. SA は Traveling Salesperson Problem (TSP) を始めとする組合せ最適化問題に対する手法として本来は提案されたが, 後に関数最適化問題へも拡張された. 以下では, [188] を参考に, SA について説明する.

探索の開始時 $t = 1$ において, 何らかの方法で初期点 $\boldsymbol{x}^t$ を与える. その後, 各反復 t において, 現在の探索点 $\boldsymbol{x}^t$ に摂動を加えることで, 新たな探索点 $\boldsymbol{u}^t$ を生成する:

$$\boldsymbol{u}^t = \boldsymbol{x}^t + \sigma \cdot \mathrm{randn}(\boldsymbol{0}, \boldsymbol{I}) \tag{A.6}$$

ここで, $\mathrm{randn}(\boldsymbol{0}, \boldsymbol{I})$ は多変量標準正規分布に従う, 各成分ごとに独立な乱数である. σ は摂動の大きさを決める制御パラメータである.

新たな探索点 $\boldsymbol{u}^t$ を目的関数にて評価した後, 次式により次反復 $t + 1$ における探索点 $\boldsymbol{x}^{t+1}$

Algorithm 23: Hooke-Jeeves 法

```
t ← 1;
x^t の初期化;
while 探索終了条件を満たしていない do
    for j = 1 to D do
        u^t = x^t;
        u_j^t = x_j^t + α(x_j^max − x_j^min);
        if f(u^t) < f(x^t) then
            x^t = u^t;
        else
            u^t = x^t;
            u_j^t = x_j^t − α(x_j^max − x_j^min);
            if f(u^t) < f(x^t) then
                x^t = u^t;
    t ← t + 1;
    if f(x^t) < f(x^{t−1}) then
        u^t = x^t + (x^t − x^{t−1});
        if f(u^t) < f(x^t) then
            x^t = u^t;
    else
        α ← α ∗ β;
```

Algorithm 24: SA

```
t ← 1;
x^t, T_t を初期化する;
while 探索終了条件を満たしていない do
    式 (A.6) により新たな探索点 u^t を生成;
    式 (A.7) により x^{t+1} を求める;
    式 (A.9) により T_{t+1} を求める;
    t ← t + 1;
```

を得る:

$$\boldsymbol{x}^{t+1} = \begin{cases} \boldsymbol{u}^t & \text{if } f(\boldsymbol{u}^t) \leq f(\boldsymbol{x}^t) \text{ or } p_t^a \leq \text{rand}[0,1] \\ \boldsymbol{x}^t & \text{otherwise} \end{cases} \tag{A.7}$$

ここで, p_t^a は反復 t における受理確率であり, rand$[0,1]$ は区間 $[0,1]$ の一様分布に従う乱数である. p_t^a は Metropolis の受理確率の式 [160] により得られる:

$$p_t^a = \exp\left(-\frac{f(\boldsymbol{u}^t) - f(\boldsymbol{x}^t)}{T_t}\right) \tag{A.8}$$

ここで T_t は世代 t における温度パラメータである．式 (A.8) では, 現在の探索点と新たな探索点の目的関数値の差 $f(\boldsymbol{u}^t) - f(\boldsymbol{x}^t)$ が小さいほど, 温度パラメータ T_t が高いほど受理確率 p_t^a も高い値となる．これは, $f(\boldsymbol{u}^t) > f(\boldsymbol{x}^t)$ となる現在の探索点 $\boldsymbol{x}^t$ に対しての改悪点 $\boldsymbol{u}^t$ への移動を受理する確率が高くなることを意味している．

各反復 t の終了時, 次反復 $t+1$ の温度パラメータ T_{t+1} を求める:

$$T_{t+1} = \frac{T_1}{\ln(t)} \tag{A.9}$$

最後に, 以上に述べた SA を Algorithm 24 に示す．適切な冷却スケジュールを採用した SA は, 対象問題が多峰性関数であっても最適解への収束が保証されている [147]．また勾配を利用しないため, 対象問題が微分不可能であったとしても適用可能であるため, black-box optimization 問題に適用可能である．

しかし, SA の最適解への収束が保証されるためには , 式 (A.9) に示すように収束速度を非常に遅くさせる必要があるため, 多くの場合現実的ではない．そのため, 最適解への収束性の保証を諦め現実的な実行時間で高精度の近似解を得るために, 温度パラメータの更新方法の修正がいくつか行われている [109].

付録 B

Zaharie による集団数 N, スケール係数 F, 交叉率 C の関係とフラットランドスケープにおける収束性解析

4.3 節にて述べたように, DE の探索性能は集団数 N, スケール係数 F, 交叉率 C の設定に大きく依存する. Zaharie [262] は目的関数値空間がフラットランドスケープ[*1]である無制約最適化問題における N, F, C の依存関係, 及び収束性を解析した. 以下では, Zaharie によるパラメータ設定と収束性の理論的な解析 [262] について述べる. なお, 以下の記述は全て [262] に基づく.

ここでは, 突然変異戦略に rand/1, 交叉手法に binomial 交叉を用いた最も基本的な DE アルゴリズム (Algorithm 8 参照) を解析対象とする. ただし, 解析を容易にするために, いくつかの手順を省略したモデルを考える[*2]. まず, 式 (3.2) の rand/1 については, $i \neq r_1 \neq r_2 \neq r_3$ となるように $\{1, ..., N\}$ から個体番号 r_1, r_2, r_3 をランダムに選択するが, ここでは単純化のために, i との重複を許し, 単に $r_1 \neq r_2 \neq r_3$ となるように選ぶ. 次に, Algorithm 7 の binomial 交叉については, j_{rand} の 3 行目の j_{rand} を省略し, 以下のように修正している:

$$u_j^{i,t} = \begin{cases} v_j^{i,t} & \text{if rand}[0,1] < C \\ x_j^{i,t} & \text{otherwise} \end{cases} \tag{B.1}$$

DE においては, 各次元 $j \in \{1, ..., D\}$ について独立に突然変異, 及び交叉を適用するので, 個体 $\boldsymbol{x} = (x_1, ..., x_D)^T$ の任意の決定変数 x_j についてのみを解析対象とすることができる. 以下, 簡便のために, 個体 $\boldsymbol{x}$ の任意の決定変数 x_j を単に x と表記する.

DE アルゴリズムの集団 $\boldsymbol{X} = \{x_1, ..., x_N\}$ における分散 $Var(\boldsymbol{X})$ の期待値 $E(Var(\boldsymbol{X}))$ を考える. ここで, $Var(\boldsymbol{X}) = \overline{\boldsymbol{X}^2} - \overline{\boldsymbol{X}}^2$, 及び $\overline{\boldsymbol{X}} = 1/N \sum_{i=1}^{N} x^i$ である. $Var(\boldsymbol{X})$ ではなく $E(Var(\boldsymbol{X}))$ を扱うのは, $\boldsymbol{X}$ は確率的な要素を含むため, 期待値としてのみ算出可能で

[*1] 目的関数 f が任意の実数 a の写像, つまり $f : \mathbb{R}^D \to a$ である時, 対象問題はフラットランドスケープであるという.

[*2] このような修正をした場合, DE アルゴリズムの探索性能が劣化することが, [188] にて報告されている.

あるためである．また, $\boldsymbol{X}$ と同様に, 変異個体の集合を $\boldsymbol{V} = \{v_1, ..., v_N\}$, 子個体の集合を $\boldsymbol{U} = \{u_1, ..., u_N\}$ とする．

始めに, $\boldsymbol{V}$ についての分散の期待値 $E(Var(\boldsymbol{V}))$ を求める．$E(Var(\boldsymbol{V})) = E(\overline{\boldsymbol{V}^2}) - E(\overline{\boldsymbol{V}}^2)$ であるので, 始めに $E(\overline{\boldsymbol{V}^2})$ と $E(\overline{\boldsymbol{V}}^2)$ をそれぞれ計算する:

$$\begin{aligned} E(\overline{\boldsymbol{V}^2}) &= \frac{1}{N}\left(\sum_{i=1}^{N}\left(x_{r_1^i} + F(x_{r_2^i} - x_{r_3^i})\right)^2\right) \\ &= (2F^2\frac{N}{N-1} + 1)\overline{\boldsymbol{X}^2} - 2F^2\frac{N}{N-1}\overline{\boldsymbol{X}}^2 \end{aligned} \tag{B.2}$$

$$\begin{aligned} E(\overline{\boldsymbol{V}}^2) &= \frac{1}{N^2}\left(\sum_{i=1}^{N}\left(x_{r_1^i} + F(x_{r_2^i} - x_{r_3^i})\right)\right)^2 \\ &= \frac{1}{N}E(\overline{\boldsymbol{V}^2}) + \frac{N-1}{N}\overline{\boldsymbol{X}}^2 \end{aligned} \tag{B.3}$$

ここで, r_1^i, r_2^i, r_3^i は各 $i \in \{1, ..., N\}$ について, $r_1^i \neq r_2^i \neq r_3^i$ となるように $\{1, ..., N\}$ の区間から一様ランダムに選択した個体番号である．式 (B.2), (B.3) を $E(Var(\boldsymbol{V})) = E(\overline{\boldsymbol{V}^2}) - E(\overline{\boldsymbol{V}}^2)$ に代入して整理すると, $E(Var(\boldsymbol{V}))$ は以下のように求まる:

$$\begin{aligned} E(Var(\boldsymbol{V})) &= E(\overline{\boldsymbol{V}^2}) - E(\overline{\boldsymbol{V}}^2) \\ &= (2F^2 + \frac{N-1}{N})Var(\boldsymbol{X}) \end{aligned} \tag{B.4}$$

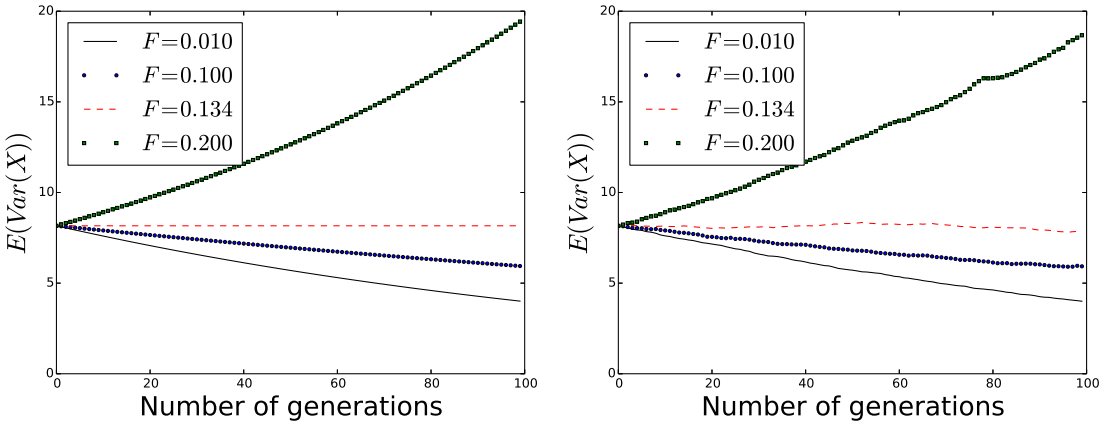
次に, 子個体の集合 $\boldsymbol{U}$ の分散の期待値 $E(Var(\boldsymbol{U}))$ を求める．ここで, 式 (B.1) において, 確率 C により v_i, $(1-C)$ にて x_i の要素が各 u_i となる．そのため, $E(\overline{\boldsymbol{U}^2})$ と $E(\overline{\boldsymbol{U}}^2)$ はそれぞれ以下のように求まる:

$$\begin{aligned} E(\overline{\boldsymbol{U}^2}) &= \frac{1}{N}\sum_{i=1}^{N}E(u_i^2) \\ &= (1-C)\overline{\boldsymbol{X}^2} + C\,E(\overline{\boldsymbol{V}^2}) \end{aligned} \tag{B.5}$$

$$\begin{aligned} E(\overline{\boldsymbol{U}}^2) =& \frac{1}{N^2}E\left(\left(\sum_{i=1}^{N}u_i\right)^2\right) \\ =& \frac{1}{N}E(\overline{\boldsymbol{U}^2}) + \frac{1}{N^2}(N^2 - 2NC + NC^2)\overline{\boldsymbol{X}}^2 \\ & - \frac{1}{N}(1-C)^2\overline{\boldsymbol{X}^2} \end{aligned} \tag{B.6}$$

式 (B.2) $\sim$ (B.6) を $E(Var(\boldsymbol{U})) = E(\overline{\boldsymbol{U}^2}) - E(\overline{\boldsymbol{U}}^2)$ に代入して整理すると, 子個体の集合 $\boldsymbol{U}$ の分散の期待値 $E(Var(\boldsymbol{U}))$ は次式のように求まる．

$$E(Var(\boldsymbol{U})) = \left(2F^2C + \frac{2C}{N} + \frac{C^2}{N} + 1\right)Var(\boldsymbol{X}) \tag{B.7}$$



(a) $E(Var(\boldsymbol{X}))$ の理論値　　(b) $E(Var(\boldsymbol{X}))$ の実験値

図 B.1: フラットランドスケープにおける, 世代経過に対する集団 $\boldsymbol{X}$ の分散の期待値 $E(Var(\boldsymbol{X}))$ の推移. (a) は理論値, (b) は実験値を表す. $N = 50$, $C = 0.2$ とし, F を変化させている. ただし, $F = 0.13416407865...$ は, 式 (B.8) を変形した $F = \sqrt{\frac{(1-\frac{C}{2})}{N}}$ より求めた. 理論値 (a) については, 実験値 (b) の $F = 0.010$ における初期値と同様に, $E(Var(\boldsymbol{X}_1)) = 8.16242590944$ としている. 実験値 (b) においては, 試行回数 50, 次元数 $D = 30$ とし, 各世代において各次元ごとに求めた分散の平均を取り, さらにその分散の平均値の 50 試行分のデータの平均を計算している. なお, 本図は [262] の Figure 2 を参考に同一の実験設定にて再実験を行い, 新たに作成した.

式 (B.7) において, $\left(2F^2C + \frac{2C}{N} + \frac{C^2}{N} + 1\right)$ の値が 1 より大きい場合, $E(Var(\boldsymbol{U})) > Var(\boldsymbol{X})$ となるので世代交代後は集団の多様性が増加する. 一方, 1 未満であった場合は $E(Var(\boldsymbol{U})) < Var(\boldsymbol{X})$ となるので, 世代交代後の集団の多様性が減少することがわかる.

次に, 無限に探索資源を与え, かつ目的関数値空間がフラットランドスケープである無制約最適化問題における DE アルゴリズムの収束性を考える. ある世代 t における $E(Var(\boldsymbol{U}))$ を $E(Var(\boldsymbol{U}^t))$ とおく. この時, 式 (B.7) において $2F^2C + \frac{2C}{N} + \frac{C^2}{N} + 1 > 0$ の場合, $\lim_{t\to\infty} E(Var(\boldsymbol{U}^t)) = \infty$ となり, DE の集団は発散する. 一方, $0 < 2F^2C + \frac{2C}{N} + \frac{C^2}{N} + 1 < 1$ の場合, $\lim_{t\to\infty} E(Var(\boldsymbol{U}^t)) = 0$ となり, DE の集団は探索空間中の一点に収束する.

このような初期収束, 及び探索の発散を防ぐためにも, 式 (B.7) の右辺の括弧で括られた数式を変形し得られた次式を満たすように N, F, C を設定するべきであると, Zaharie は結論づけている:

$$2F^2 - \frac{2}{N} + \frac{C}{N} = 0 \tag{B.8}$$

式 (B.8) が成り立つ場合, $E(Var(\boldsymbol{U})) = Var(\boldsymbol{X})$ となるため, 集団の多様性は世代交代後においても不変である.

最後に, 式 (B.7) を検証するために, 図 B.1 にフラットランドスケープにおける, 世代経過に対する集団 $\boldsymbol{X}$ の分散の期待値 $E(Var(\boldsymbol{X}))$ の理論値, 及び実験値の推移を示す. なお, 図 B.1 は [262] の Figure 2 を参考に同一の実験設定にて再実験を行い, 新たに作成した. 図 B.1

では, 理論値と実験値の推移が概ね一致していることがわかる. また, 本実験では $N = 50$, $C = 0.2$ としており, この場合式 (B.8) を満たす F の値は $F = \sqrt{\frac{(1-\frac{C}{2})}{N}} = 0.13416407865...$ である. $F = 0.13416407865$ においては, 世代が経過したとしても $E(Var(\boldsymbol{X}))$ が一定である. 一方, この値以下の $F = 0.01, 0.1$ では $E(Var(\boldsymbol{X}))$ が世代の経過に伴い減少, この値以上の $F = 0.2$ では $E(Var(\boldsymbol{X}))$ が増加していることがわかる.

付録 C

BBOB benchmarks

本章では BBOB benchmarks [91, 93] の説明をする．BBOB benchmarks はどの探索手法がどの関数クラスにおいて優れているか, 又は black box optimization 環境を想定した全ての関数クラスにおいて優れている手法について議論するために設計されている．BBOB benchmarks は 24 個のベンチマーク関数 $f_1 \sim f_{24}$ から構成されており, $f_1 \sim f_5$ は変数分離可な関数, $f_6 \sim f_9$ は弱い悪スケール性関数, $f_{10} \sim f_{14}$ は強い悪スケール性関数, $f_{15} \sim f_{19}$ は大域的単峰性の多峰性関数, $f_{20} \sim f_{24}$ は弱い大域的構造を有する関数である．表 2.2 に, BBOB benchmarks [91, 93] の 24 個の関数 $f_1 \sim f_{24}$ の大まかな問題性質 [159] を示す．表 2.2 に対して, 本章では各ベンチマーク関数の定義, 詳細な問題性質, どのような性能を計測する意図でその関数は設計されたかといった詳細について述べる．なお, 以下の説明は全て BBOB benchmarks のマニュアル [93] に基づいている．

始めに C.1 節にて $f_1 \sim f_{24}$ の定義に使用する各記号の説明を含む, BBOB benchmarks の一般的な設定について述べる．次に, C.2 節にて各ベンチマーク関数の詳細について説明する．

C.1 一般的な設定

C.1.1 探索範囲

BBOB benchmarks の全ベンチマーク関数において, 探索範囲は $\boldsymbol{S} = [-5, 5]^D$ である．ここで, D は次元数である．また, いくつかの関数において, $\boldsymbol{x} \notin \boldsymbol{S}$ となる $\boldsymbol{x}$ については, ペナルティ関数 $f_{\mathrm{pen}} : \mathbb{R}^D \to \mathbb{R}$ に応じた罰則値が, 目的関数値に加算される:

$$f_{\mathrm{pen}} = \sum_{i=1}^{D} \max(0, |x_i| - 5)^2 \tag{C.1}$$

C.1.2 最適解と最適値

f_5, f_{24} などの一部の関数を除き, 最適解 $\boldsymbol{x}^{\mathrm{opt}}$ は $[-4, 4]^D$ の範囲内に各次元ごとに一様ランダムに位置する．また, 最適値 $f(\boldsymbol{x}^{\mathrm{opt}})$ はコーシー分布に従う乱数にて割り当てている．一部

の関数では $\boldsymbol{x}$ を変換した $\boldsymbol{z} = (z_1, ..., z_D)^T$ を使用しているが, 多くの場合 $\boldsymbol{z} = \boldsymbol{0}$ が最適解である. なお, 以上に述べた設定は, 各次元, 各関数, 各試行ごとに異なる.

C.1.3　探索空間の線形, 及び非線形変換操作

変数分離化な関数を変数分離不可な関数にするために, また関数の条件数を制御するために, 解空間の線形変換を行っている. また, いくつかの関数については, C.1.4 節にて説明する T_{osz} や T_{asy} のように, 解空間, 及び目的関数空間に非線形変換を適用している.

C.1.4　BBOB benchmarks の関数の記述に使用する各記号

以下に BBOB benchmarks の関数の記述に使用する各記号の説明を列挙する:

- $\|.\|$: ユークリッドノルムであり, $\|\boldsymbol{x}\|^2 = \sum_{i=1}^{D} x_i^2$ である.
- $[.]$: 引数の最も近い整数値を表す.
- $\boldsymbol{0}$: $\boldsymbol{0} = (0, ..., 0)^T$
- $\boldsymbol{1}$: $\boldsymbol{1} = (1, ..., 1)^T$
- Λ^{α}: $D \times D$ の対角行列を表し, $i \in \{1, ..., D\}$ 番目の対角要素は $\lambda_{i,i} = \alpha^{\frac{1}{2}\frac{i-1}{D-1}}$ である.
- $\boldsymbol{1}_{-}^{+}$: -1 と 1 が等しい確率で出現する D 次元整数ベクトル
- $\boldsymbol{Q}, \boldsymbol{R}$: グラムシュミットの正規直交化法を用いて生成した回転行列. 各次元, 各関数, 各試行ごとに生成している.
- $T_{\mathrm{asy}}^{\beta} : \mathbb{R}^D \to \mathbb{R}^D$ は解空間を非対称にするための変換である:

$$x_i = \begin{cases} x_i^{1+\beta\frac{i-1}{D-1}\sqrt{x_i}} & \text{if } x_i > 0\,,\ i \in \{1, ..., D\} \\ x_i & \text{otherwise} \end{cases}$$

- $T_{\mathrm{osz}} : \mathbb{R}^n \to \mathbb{R}^n$ は局所的な不規則性を解空間に与える変換である $(n \in \{1, D\})$:

$$x \to \mathrm{sign}(x) \exp\left(\hat{x} + 0.049\left(\sin(c_1\hat{x}) + \sin(c_2\hat{x})\right)\right)$$

$$\hat{x} = \begin{cases} \log|x| & \text{if } x \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}, c_1 = \begin{cases} 10 & \text{if } x > 0 \\ 5.5 & \text{otherwise} \end{cases}, c_2 = \begin{cases} 7.9 & \text{if } x > 0 \\ 3.1 & \text{otherwise} \end{cases}$$

- $\boldsymbol{x}^{\mathrm{opt}}$: 最適解であり, $f(\boldsymbol{x}^{\mathrm{opt}}) = f_{\mathrm{opt}}$ が最小目的関数値となる.

C.2　各ベンチマーク関数の定義, 及び問題性質

本節では, $f_1 \sim f_{24}$ の各ベンチマーク関数の定義, 及び問題性質について述べる. 概ね, 関数の定義を示した後に, その関数の問題性質, 及びその関数に探索手法を適用することで得られ

る知見の説明という形式を取る.

C.2.1 f_1 : Sphere 関数

$$f_1(\boldsymbol{x}) = \|\boldsymbol{z}\|^2 + f_{\mathrm{opt}} \tag{C.2}$$

ここで, $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}}$ である.

問題性質

- 単峰性, 対称性を持ち次元数に対するスケール不変性有する, 変数分離可

得られる知見

- 最も単純な二次関数における探索手法の収束速度

C.2.2 f_2 : Ellipsoidal 関数

$$f_2(\boldsymbol{x}) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} z_i^2 + f_{\mathrm{opt}} \tag{C.3}$$

ここで, $\boldsymbol{z} = T_{\mathrm{osz}}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})$ である.

問題性質

- 単峰性, 悪スケール性 (条件数 = 約 10^6), 変数分離可

得られる知見

- f_1 に対して: 対象手法は探索空間の対称性を利用しているか否か
- f_{10} に対して: 対象手法は変数分離化な性質を利用しているか否か

C.2.3 f_3 : Rastrigin 関数

$$f_3(\boldsymbol{x}) = 10\left(D - \sum_{i=1}^{D}\cos(2\pi z_i)\right) + \|\boldsymbol{z}\|^2 + f_{\mathrm{opt}} \tag{C.4}$$

ここで, $\boldsymbol{z} = \Lambda^{10} T_{asy}^{0.2}\left(T_{osz}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})\right)$ である.

問題性質

- 強い多峰性 (局所解の数: 約 10^D 個), 悪スケール性 (条件数 = 約 10), 変数分離可

得られる知見

- f_2 に対して: 多峰性の影響

C.2.4　f_4 : Büche-Rastrigin 関数

$$
\begin{aligned}
f_4(\boldsymbol{x}) &= 10\left(D - \sum_{i=1}^{D}\cos(2\pi z_i)\right) + \sum_{i=1}^{D} z_i^2 + 100 f_{\mathrm{pen}}(\boldsymbol{x}) + f_{\mathrm{opt}} \qquad \text{(C.5)}\\
z_i &= s_i T_{osz}(x_i - x_i^{\mathrm{opt}}) \text{ for } i = 1, ..., D\\
s_i &= \begin{cases} 10 \times 10^{\frac{1}{2}\,\frac{i-1}{D-1}} & \text{if } z_i > 0 \text{ and } i = 1, 3, 5, ...\\ 10^{\frac{1}{2}\,\frac{i-1}{D-1}} & \text{otherwise} \end{cases}
\end{aligned}
$$

問題性質

- 強い多峰性 (局所解の数: 約 10^D 個), 悪スケール性 (条件数 = 約 10), 変数分離可, 最適解付近にて強い非対称性

得られる知見

- f_3 に対して: 非対称性の影響

C.2.5　f_5 : Linear Slope 関数

$$
\begin{aligned}
f_5(\boldsymbol{x}) &= \sum_{i=1}^{D}(5|s_i| - s_i z_i) + f_{\mathrm{opt}} \qquad \text{(C.6)}\\
s_i &= \mathrm{sign}(x_i^{\mathrm{opt}} 10^{\frac{i-1}{D-1}}) \,, i \in \{1, ..., D\}\\
z_i &= \begin{cases} x_i & \text{if } x_i^{\mathrm{opt}} x_i < 5^2 \,, i \in \{1, ..., D\}\\ x_i^{\mathrm{opt}} & \text{otherwise} \end{cases}
\end{aligned}
$$

問題性質

- 単峰性 (線形関数であり, 探索領域の境界付近に最適解が存在), 変数分離可

得られる知見

- 探索領域の境界付近への収束速度

C.2.6 f_6 : Attractive Sector 関数

$$
\begin{aligned}
f_6(\boldsymbol{x}) &= T_{osz}\left(\sum_{i=1}^{D}(s_i z_i)^2\right)^{0.9} + f_{\mathrm{opt}} \\
\boldsymbol{z} &= \boldsymbol{Q}\Lambda^{10}\boldsymbol{Q}(\boldsymbol{x}-\boldsymbol{x}^{\mathrm{opt}}) \\
s_i &= \begin{cases} 10^2 & \text{if } z_i \times x_i^{\mathrm{opt}} > 0 \\ 1 & \text{otherwise} \end{cases}
\end{aligned}
\tag{C.7}
$$

問題性質

- 単峰性, 変数分離不可, 探索空間の強い非対称性

得られる知見

- f_1 に対して: 強い非対称性の影響

C.2.7 f_7 : Step Ellipsoidal 関数

$$
\begin{aligned}
f_7(\boldsymbol{x}) &= 0.1\max\left(|\hat{z}_1|/10^4, \sum_{i=1}^{D} 10^2 \,{}^{\frac{i-1}{D-1}} z_i^2\right) + f_{\mathrm{pen}}(\boldsymbol{x}) + f_{\mathrm{opt}} \\
\hat{\boldsymbol{z}} &= \Lambda^{10}\boldsymbol{R}(\boldsymbol{x}-\boldsymbol{x}^{\mathrm{opt}}) \\
\tilde{z}_i &= \begin{cases} \lfloor 0.5+\hat{z}_i \rfloor & \text{if } \hat{z}_i > 0.5 \\ \lfloor 0.5+10\hat{z}_i \rfloor/10 & \text{otherwise} \end{cases} \\
\boldsymbol{z} &= \boldsymbol{Q}\tilde{\boldsymbol{z}}
\end{aligned}
\tag{C.8}
$$

問題性質

- 単峰性, 変数分離不可, 悪スケール性 (条件数 = 約 100)

得られる知見

- plateau において停滞せずに探索が行える能力

C.2.8 f_8 : Rosenbrock 関数

$$
f_8(\boldsymbol{x}) = \sum_{i=1}^{D-1}\left(100(z_i - z_{i+1})^2 + (z_i - 1)^2\right) + f_{\mathrm{opt}} \tag{C.9}
$$

ここで, $\boldsymbol{z} = \max\left(1, \frac{\sqrt{D}}{8}\right)(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}}) + 1$, $\boldsymbol{z}^{\mathrm{opt}} = \mathbf{1}$

問題性質

- 弱い多峰性関数 (ただし, 3 次元以下では単峰性関数 [209]), 部分的に変数分離可
- f_8 において, 探索手法は $\boldsymbol{z} = \mathbf{0}$ に一度収束し, その後 $\boldsymbol{z} = \mathbf{1}$ へ移動する振る舞いを示す

得られる知見

- 探索の収束後, 最適解へと再度移動する能力

C.2.9　f_9 : Rosenbrock 関数 (Rotated)

$$f_9(\boldsymbol{x}) = \sum_{i=1}^{D-1} \left(100(z_i - z_{i+1})^2 + (z_i - 1)^2\right) + f_{\mathrm{opt}} \tag{C.10}$$

ここで, $\boldsymbol{z} = \max\left(1, \frac{\sqrt{D}}{8}\right)\boldsymbol{R}\boldsymbol{x} + \frac{1}{2}$, $\boldsymbol{z}^{\mathrm{opt}} = \mathbf{1}$

問題性質

- f_8 に回転行列を適用した関数

得られる知見

- f_8 でのように部分的に分解可能な性質を利用せずに, 探索収束後に再度移動する能力

C.2.10　f_{10} : Ellipsoidal 関数 (Rotated)

$$f_{10}(\boldsymbol{x}) = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} z_i^2 + f_{\mathrm{opt}} \tag{C.11}$$

ここで, $\boldsymbol{z} = T_{osz}\left(\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})\right)$

問題性質

- 単峰性, 悪スケール性 (条件数 = 約 10^6), 変数分離不可

得られる知見

- f_2 に対して: 変数分離不可な影響

C.2.11 f_{11} : Discus 関数

$$f_{11}(\boldsymbol{x}) = 10^6 z_1^2 + \sum_{i=2}^{D} z_i^2 + f_{\mathrm{opt}} \tag{C.12}$$

ここで, $\boldsymbol{z} = T_{osz}\left(\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})\right)$

問題性質

- 局所的な不規則性を有する単峰性, 悪スケール性 (条件数 = 約 10^6), 変数分離不可

得られる知見

- f_{10} に対して: 1 変数のみ極端に異なる変数スケールの影響

C.2.12 f_{12} : Bent Cigar 関数

$$f_{12}(\boldsymbol{x}) = z_1^2 + 10^6 \sum_{i=2}^{D} z_i^2 + f_{\mathrm{opt}} \tag{C.13}$$

ここで, $\boldsymbol{z} = \boldsymbol{R}T_{asy}^{0.5}\left(\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})\right)$

問題性質

- 単峰性, 悪スケール性 (条件数 = 約 10^6), 変数分離不可

得られる知見

- f_{10} に対して: 1 変数のみ極端に異なる変数スケールの影響

C.2.13 f_{13} : Sharp Ridge 関数

$$f_{13}(\boldsymbol{x}) = z_1^2 + 100\sqrt{\sum_{i=2}^{D} z_i^2} + f_{\mathrm{opt}} \tag{C.14}$$

ここで, $\boldsymbol{z} = \boldsymbol{Q}\Lambda^{10}\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})$.

問題性質

- 単峰性, 悪スケール性 (条件数 = 約 10^6), 変数分離不可

得られる知見

- f_{12} に対して: なめらかではない, 微分不可な尾根の探索への影響

C.2.14　f_{14} : Different Powers 関数

$$f_{14}(\boldsymbol{x}) = \sqrt{\sum_{i=1}^{D} |z_i|^{2+4\frac{i-1}{D-1}}} + f_{\mathrm{opt}} \tag{C.15}$$

ここで, $\boldsymbol{z} = \boldsymbol{R}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})$

問題性質

- 単峰性, 悪スケール性 (条件数 = 約 10^6), 変数分離不可

得られる知見

- f_{10} に対して: 最適解に近づくほど, 目的関数値への各決定変数の影響が変化する性質での探索への影響

C.2.15　f_{15} : Rastrigin 関数 (Rotated)

$$f_{15}(\boldsymbol{x}) = 10\left(D - \sum_{i=1}^{D} \cos(2\pi z_i)\right) + \|\boldsymbol{z}\|^2 + f_{\mathrm{opt}} \tag{C.16}$$

$$\tag{C.17}$$

ここで, $\boldsymbol{z} = \boldsymbol{R}\Lambda^{10}\boldsymbol{Q}T_{asy}^{0.2}\left(T_{osz}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})\right)$.

問題性質

- 強い多峰性 (局所解の数: 約 10^D 個), 悪スケール性 (条件数 = 約 10), 変数分離不可

得られる知見

- f_3 に対して: 変数分離不可な影響

C.2.16　f_{16} : Weierstrass 関数

$$f_{16}(\boldsymbol{x}) = 10\left(\frac{1}{D}\sum_{i=1}^{D}\sum_{k=0}^{11}\frac{1}{2^k}\cos\left(2\pi 3^k\left(z_i + \frac{1}{2}\right)\right) - f_0\right)^3 + \frac{10}{D}f_{\mathrm{pen}}(\boldsymbol{x}) + f_{\mathrm{opt}} \tag{C.18}$$

$$\tag{C.19}$$

ここで, $\boldsymbol{z} = \boldsymbol{R}\Lambda^{1/100}\boldsymbol{Q}T_{\mathrm{osz}}\left(\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})\right)$, $f_0 = \sum_{k=0}^{11} \frac{1}{2^k}\cos\left(\frac{2\pi 3^k}{2}\right)$

問題性質

- 強い多峰性, 複数最適解が存在する性質, 巨視的には規則的に局所解が存在 (局所的に不規則), 変数分離不可

得られる知見

- f_{17} に対して: 多峰性の強さの度合い, 及び局所解の規則性の探索への影響

C.2.17 f_{17} : Schaffers F7 関数

$$f_{17}(\boldsymbol{x}) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}\left(\sqrt{s_i} + \sqrt{s_i}\sin^2\left(50 s_i^{\frac{1}{5}}\right)\right)\right)^2 + 10 f_{\mathrm{pen}}(\boldsymbol{x}) + f_{\mathrm{opt}} \tag{C.20}$$

ここで, $\boldsymbol{z} = \Lambda^{10}\boldsymbol{Q}T_{\mathrm{asy}}^{0.5}\left(\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})\right)$, $s_i = \sqrt{z_i^2 + z_{i+1}^2}, i \in \{1, ..., D\}$

問題性質

- 強い多峰性, 不規則に局所解が存在, 非対照な探索空間, 弱い悪スケール性, 変数分離不可

得られる知見

- f_{15} に対して: 規則性の無い局所解の配置が探索に与える影響

C.2.18 f_{18} : Schaffers F7 Function 関数 (moderately ill-conditioned)

$$f_{17}(\boldsymbol{x}) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}\left(\sqrt{s_i} + \sqrt{s_i}\sin^2\left(50 s_i^{\frac{1}{5}}\right)\right)\right)^2 + 10 f_{\mathrm{pen}}(\boldsymbol{x}) + f_{\mathrm{opt}} \tag{C.21}$$

ここで, $\boldsymbol{z} = \Lambda^{1000}\boldsymbol{Q}T_{\mathrm{asy}}^{0.5}\left(\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{opt}})\right)$, $s_i = \sqrt{z_i^2 + z_{i+1}^2}, i \in \{1, ..., D\}$

問題性質

- f_{17} の問題性質に加えて, 悪スケール性 (条件数 = 約 10^3)

得られる知見

- f_{17} に対して: 悪スケール性の影響

C.2.19　f_{19} : Composite Griewank-Rosenbrock Function F8F2 関数

$$f_{19}(\boldsymbol{x}) = \left(\frac{10}{D-1} \sum_{i=1}^{D-1} \left(\frac{s_i}{4000} - \cos(s_i) \right) \right) + 10 + f_{\text{opt}} \tag{C.22}$$

ここで, $\boldsymbol{z} = \max(1, \frac{\sqrt{D}}{8})\boldsymbol{R}\boldsymbol{x} + 0.5$, $s_i = 100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2, i \in \{1, ..., D\}$, $\boldsymbol{z}^{\text{opt}} = \mathbf{1}$

問題性質

- f_9 の問題性質に加えて, 強い多峰性

得られる知見

- f_9 に対して: 多峰性の強弱が探索に与える影響

C.2.20　f_{20} : Schwefel 2.26 関数

$$f_{20}(\boldsymbol{x}) = -\frac{1}{D} \sum_{i=1}^{D} z_i \sin(\sqrt{|z_i|}) + 4.189828872724339 + 100 f_{\text{pen}}(\boldsymbol{z}/100) + f_{\text{opt}} \tag{C.23}$$

ここで,

$$\begin{aligned}
\hat{\boldsymbol{x}} &= 2(\mathbf{1}_-^+ \otimes \boldsymbol{x}) \\
\hat{z}_1 &= \hat{x}_1, \hat{z}_{i+1} = \hat{x}_{i+1} + 0.25(\hat{x}_i + x_i^{\text{opt}}), i \in \{1, ..., D-1\} \\
\boldsymbol{z} &= 100 \left(\Lambda^{10} \hat{\boldsymbol{z}} - \boldsymbol{x}^{\text{opt}} \right) \\
\boldsymbol{x}^{\text{opt}} &= 4.2096874633/2\mathbf{1}_-^+
\end{aligned}$$

問題性質

- 多峰性, 大域的多峰性, 変数分離不可

得られる知見

- f_{17} に対して: 大域的多峰性が探索に与える影響

C.2.21　f_{21} : Gallagher's Gaussian 101-me Peaks 関数

$$f_{21}(\boldsymbol{x}) = T_{osz} \left(10 - \max_{i=1}^{101} w_i \exp \left(-\frac{1}{2D} (\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{R}^T \boldsymbol{C}_i \boldsymbol{R} (\boldsymbol{x} - \boldsymbol{y}) \right) \right)^2 + f_{\text{pen}}(\boldsymbol{x}) + f_{\text{opt}} \tag{C.24}$$

ここで, $\boldsymbol{C}_i = \Lambda^{\alpha_i}/\alpha_i^{1/4}$ であり,

$$w_i = \begin{cases} 1.1 + 8 \times \frac{i-2}{99} & \text{if } 2 \leq i \leq 101 \\ 10 & \text{otherwise} \end{cases}$$

問題性質

- 多峰性 (深さ, 及びサイズがランダムに生成された 101 個の局所解が存在), 大域的構造を持たない, 変数分離不可, 弱い悪スケール性 (最適解の周辺にて条件数 = 約 30)

得られる知見

- 大域的構造を持たない関数における探索性能

C.2.22 f_{22} : Gallagher's Gaussian 21-hi Peaks 関数

$$f_{22}(\boldsymbol{x}) = T_{osz}\left(10 - \max_{i=1}^{21} w_i \exp\left(-\frac{1}{2D}(\boldsymbol{x}-\boldsymbol{y})^T \boldsymbol{R}^T \boldsymbol{C}_i \boldsymbol{R}(\boldsymbol{x}-\boldsymbol{y})\right)\right)^2 + f_{\text{pen}}(\boldsymbol{x}) + f_{\text{opt}} \tag{C.25}$$

ここで, $\boldsymbol{C}_i = \Lambda^{\alpha_i}/\alpha_i^{1/4}$ であり,

$$w_i = \begin{cases} 1.1 + 8 \times \frac{i-2}{19} & \text{if } 2 \leq i \leq 21 \\ 10 & \text{otherwise} \end{cases}$$

問題性質

- 多峰性 (深さ, 及びサイズがランダムに生成された 21 個の局所解が存在), 大域的構造を持たない, 変数分離不可, 悪スケール性 (最適解の周辺にて条件数 = 約 10^3)

得られる知見

- f_{17} に対して: 悪スケール性が探索に与える影響

C.2.23 f_{23} : Katsuura 関数

$$f_{23}(\boldsymbol{x}) = \frac{10}{D^2}\prod_{i=1}^{D}\left(1 + i\sum_{j=1}^{32}\left(\frac{|2^j z_i - [2^j z_i]|}{2^j}\right)\right) - \frac{10}{D^2} + f_{\text{pen}}(\boldsymbol{x}) \tag{C.26}$$

ここで, $\boldsymbol{z} = \boldsymbol{Q}\Lambda^{100}\boldsymbol{R}(\boldsymbol{x} - \boldsymbol{x}^{\text{opt}})$

問題性質

- 強い多峰性 (規則的に局所解が存在), 弱い大域的構造, 変数分離不可

得られる知見

- 局所解の規則性が大域的探索に与える影響

C.2.24　f_{24} : Lunacek bi-Rastrigin 関数 [153]

$$f_{24}(\boldsymbol{x}) = \min\left(\sum_{i=1}^{D}(\hat{x}_i - \mu_0)^2, d\,D + s\sum_{i=1}^{D}(\hat{x}_i - \mu_1)^2\right) + 10\left(D - \sum_{i=1}^{D}\cos(2\pi z_i)\right) - \frac{10}{D^2} + 10^4 f_{\mathrm{pen}}(\boldsymbol{x}) \tag{C.27}$$

ここで,

$$\hat{\boldsymbol{x}} = 2\mathrm{sign}(\boldsymbol{x}^{\mathrm{opt}}) \otimes \boldsymbol{x}, \boldsymbol{x}^{\mathrm{opt}} = \mu_0 \mathbf{1}^{+}_{-}, \boldsymbol{z} = \boldsymbol{Q}\Lambda^{100}\boldsymbol{R}(\hat{\boldsymbol{x}} - \mu_0\mathbf{1})$$

$$\mu_0 = 2.5, \mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}}, s = 1 - \frac{1}{2\sqrt{D+20} - 8.2}, d = 1$$

問題性質

- 強い多峰性, double-funnel (サイズの異なる f_{15} を 2 つ並べたような景観), 変数分離不可

得られる知見

- double-funnel が探索に与える影響

C.3　BBOB benchmarks における性能評価指標

本節では, 図 5.3 ∼ 5.4, 図 6.6 ∼ 6.9, 図 6.11, 図 6.14, 6.15 に示した, 累積経験分布関数 $F : \mathbb{R} \to [0, 1]$ (Empirical Cumulative Distribution Functions, ECDF) について述べる.

BBOB benchmarks では, $f_1 \sim f_{24}$ の関数ごとに最適解の位置や最適値を変えることで 15 試行の各試行毎に異なる関数インスタンスを提供しているので, $24 \times 15 = 360$ 個の関数インスタンスが存在する. その内のある関数インスタンスについて, 到達目標となる目的関数値 $f_{\mathrm{target}} = f_{\mathrm{opt}} - \Delta f$ を考える. ここで, f_{opt} は最適解 $\boldsymbol{x}^{\mathrm{opt}}$ の目的関数値, Δf は許容誤差値である. Δf の値は $[10^{-8}, 10^{1.8}]$ の範囲から, $\{10^{1.8}, 10^{1.6}, 10^{1.4}, ..., 10^{-7.6}, 10^{-7.8}, 10^{-8}\}$ のように一様に 50 個選択される. つまり, 関数インスタンスごとに 50 個の到達目標となる解の目的関数値 f_{target} が与えられる.

累積経験分布関数は, 与えられた関数クラス (例えば, 全ての関数 $f_1 \sim f_{24}$ や多峰性関数クラス $f_{15} \sim f_{19}$) の全てのベンチマーク関数において, それぞれ 15 試行づつ行うことで得られ

る．図 5.3 ∼ 5.4, 図 6.6 ∼ 6.9, 図 6.11, 図 6.14, 6.15 において, 縦軸は全ての関数インスタンス (例えば, $f_1 \sim f_{24}$ では $24 \times 15 = 360$ 問, $f_{15} \sim f_{19}$ では $5 \times 15 = 75$ 問) における 50 個全ての Δf について, 実際に対象手法が到達できた, つまり $f_{\mathrm{target}} \leq f(\boldsymbol{x})$ となった割合 $\in [0, 1]$ を示している．なお, 累積経験分布関数は単調増加する．縦軸についての値が高いほど, より多くの関数インスタンスにおいて高い精度の解が得られていることを意味し, 1 の場合は全ての試行において全てのベンチマーク関数の最適解 $\boldsymbol{x}^{\mathrm{opt}}$ が求められていることを意味する．また, 横軸は対応する評価回数を次元数で割った値であり, 対数スケールを取っている．同じ評価回数下において, 縦軸に関しての値が高いほど, 探索性能が優れている手法である．

付録 D

発表文献

査読付き和文論文誌論文

1). 田邊遼司, 福永 Alex:
DE における Exponential Crossover の再評価,
進化計算学会論文誌, vol. 6, no. 1, pp. 42-52
2). 田邊遼司, 福永 Alex:
自動チューナーを用いた異なる最大評価回数における Differential Evolution アルゴリズムのパラメータ設定の調査,
進化計算学会論文誌, vol. 6, no. 2, pp. 67-81

解説

1). 田邊遼司, 串田淳一, 畠中利治:
関数最適化における進化計算,
計測自動制御学会学会誌「計測と制御」, vol. 54, no. 8, pp. 567-572

査読付き国際会議論文

1). Ryoji Tanabe:
A Note on Multi-Funnel Functions for Expensive Optimization Scenario,
Proc. Genetic and Evolutionary Computation Conference (**GECCO-2015**, poster), pp. 1493-1494, Madrid, July, 2015
2). Ryoji Tanabe and Alex Fukunaga:
Tuning Differential Evolution for Cheap, Medium, and Expensive Computational Budgets,
Proc. IEEE Congress on Evolutionary Computation (**CEC-2015**), pp. 2018-2025, Sendai, May, 2015
3). Claus de Castro Aranha, Ryoji Tanabe, Romain Chassagne, Alex Fukunaga:
Optimization of Oil Reservoir Models Using Tuned Evolutionary Algorithms and Adaptive Differential Evolution,
Proc. IEEE Congress on Evolutionary Computation (**CEC-2015**), pp. 877-884, Sendai, May, 2015
4). Ryoji Tanabe and Alex Fukunaga:
Reevaluating Exponential Crossover in Differential Evolution,
Proc. Parallel Problem Solving from Nature (**PPSN-2014**), pp. 201-210, Ljubljana, September, 2014
5). Ryoji Tanabe and Alex Fukunaga:
On the Pathological Behavior of Adaptive Differential Evolution on Hybrid Objective Functions,
Proc. ACM Genetic and Evolutionary Computation Conference (**GECCO-2014**), pp. 1335-1342, Vancouver, July, 2014

6). Ryoji Tanabe and Alex Fukunaga:
Improving the Search Performance of SHADE Using Linear Population Size Reduction,
Proc. IEEE Congress on Evolutionary Computation (**CEC-2014**), pp. 1658-1665, Beijing, July, 2014
7). Ryoji Tanabe and Alex Fukunaga:
Evaluation of a Randomized Parameter Setting Strategy for Island-Model Evolutionary Algorithms,
Proc. IEEE Congress on Evolutionary Computation (**CEC-2013**), pp. 1263-1270, Cancun, June, 2013
8). Ryoji Tanabe and Alex Fukunaga:
Success-History Based Parameter Adaptation for Differential Evolution,
Proc. IEEE Congress on Evolutionary Computation (**CEC-2013**), pp. 71-78, Cancun, June, 2013
9). Ryoji Tanabe and Alex Fukunaga:
Evaluating the performance of SHADE on CEC 2013 benchmark problems,
Proc. IEEE Congress on Evolutionary Computation (**CEC-2013**), pp. 1952-1959, Cancun, June, 2013.

査読無し国内学会発表

1). 田邊遼司, 福永 Alex:
適応 DE の適応メカニズムの解析
第 9 回進化計算学会研究会, 理化学研究所計算科学研究機構, 2015/9
2). 田邊遼司, 福永 Alex:
Expensive Optimization における進化アルゴリズムの Evolvability について
第 29 回人工知能学会全国大会, 公立はこだて未来大学, 2015/5
3). 田邊遼司, 福永 Alex:
Expensive Scenario での多峰性関数における進化アルゴリズムの Evolvability について
第 42 回知能システムシンポジウム, 兵庫県神戸市, 2015/3 Claus de Castro Aranha, Romain Chassagne, Ryoji Tanabe
Application of Auto-Adaptive Evolutionary Algorithms to the History Matching Problem
進化計算シンポジウム 2014, 広島県廿日市市, 2014/12
4). 田邊遼司, 福永 Alex:
解の評価に時間がかかる問題における DE の Parameter Study
広島県廿日市市, 2014/12
5). 田邊遼司, 福永 Alex:
Hybrid 関数における適応 DE の振る舞いについて
計測自動制御学会 システム・情報部門 学術講演会 (SSI), 岡山大学, 2014/11
6). 田邊遼司, 福永 Alex:
DE における Exponential Crossover の再評価
第 7 回進化計算学会研究会, 近畿大学, 2014/8
7). 田邊遼司, 福永 Alex:
Linear Population Size Reduction を用いた SHADE
第 5 回コンピューテーショナル・インテリジェンス研究会, 慶応大学, 2014/7
8). 田邊遼司, 福永 Alex:
Partially Separable 単目的関数について
進化計算シンポジウム 2013, 鹿児島県霧島市, 2013/12
9). 田邊遼司, 福永 Alex:
ランダムに制御パラメタを設定する大規模並列環境におけるパラメタフリー島型進化アルゴリズム
第 27 回人工知能学会全国大会, 富山県富山市, 2013/6,
10). 田邊遼司, 福永 Alex:
離散的なメモリを用いた適応 DE
第 4 回進化計算学会研究会, 防衛大学校, 2013/3
11). 田邊遼司, 福永 Alex:
探索性能のパレートフロントに基づく DE の切り替え戦略
進化計算シンポジウム 2012, 長野県軽井沢, 2012/12