横浜国立大学理工学部

令和元年度学士論文

競争均衡原理に着想を得たメタヒューリスティクスの適応的アルゴリズム調整

学籍番号　1664196

氏名　西原 慧

数物・電子情報系学科　電子情報システム EP

主任指導教員　中田 雅也 准教授

提出日　令和 2 年 2 月 19 日 (水)

概要

今日の工学システムにおいて最適化技術は欠かせないものとなっており、汎用的な技術の代表例としてこれまで様々なメタヒューリスティクスが提案されている。そのアルゴリズムは自身の構成要素として、ハイパーパラメータや遺伝的操作などを持つ。これらを自律適応的に調整されたメタヒューリスティクスは、問題特性と探索状況に適宜特化されるため、問題を効率的に解けることが期待される。

ここに対し既存の研究では、ハイパーパラメータのみを調整する手法や遺伝的操作までを調整対象とする手法が提案されているが、これらの手法は「経験ベース」であると言える。なぜなら、「良い性能を導出したアルゴリズムの設定を再現すれば、今後も良い性能を導出する」という仮定の下、例えば過去に成功したハイパーパラメータから次を予測して調整する等のケースが多いからである。しかしながら、これらの手法はある程度の回数の解評価を行うまでそのアルゴリズムの良し悪しはわからないという難点があり、特に解評価が高コストな問題ではこれは大きな不利益となる。

さらに、遺伝的操作までを調整対象とする場合、例えば遺伝的操作の中で使用するハイパーパラメータ等の影響により、調整するアルゴリズムの挙動の不確実性が増加し、また選択肢が膨大化する困難さから、この調整を実用化するには極めて効率的なアルゴリズム調整技術が必要となる。

これらを受けて本研究では、経済学における競争均衡原理に着想を得た新しい「質ベース」の手法を提案する。具体的には、複数のアルゴリズムを並行して解探索させ競合させることで、より劣ったアルゴリズムらを調整する。これにより追加の解評価なしに、十分な経験を経ずとも質の高い調整を実現できるフレームワークを構築した。また提案手法は、対象とする複数のアルゴリズムは種類を問わず柔軟に構築できるという利点も持つ。

今回はケーススタディとして、複数の差分進化法を用いて調整した実験結果を用いて、複数の単一目的最適化問題における提案手法の有用性を示す。さらに、調整結果の考察や今後の課題も併せて述べる。

目次

第 1 章

序論

1.1 研究背景

ナビゲーションシステムにおけるルート決定から工場におけるスケジューリング、航空機や架橋などの構造物設計まで、諸目的を達成するために今日の様々な実社会工学システムに最適化技術は内包されている。その汎用的な技術の一つにメタヒューリスティクスがあり、中でも生物の遺伝に着想を得たアルゴリズムが多く存在する。

メタヒューリスティクスは、そのアルゴリズムの構成要素であるハイパーパラメータや遺伝的操作について選定方針が不明確である場合が多い。ノーフリーランチ定理 [23] が指摘するように、これらの構成要素を適切に選定することで、そのアルゴリズムを問題に特化させ最適化の性能を高める効果を生むことが期待できる [5, 11]。そこで、ハイパーヒューリスティクスをはじめとして、最適化アルゴリズムの自動設計やアルゴリズムの適応的調整など、問題や探索過程に応じてアルゴリズムを自律的に特化させる研究が盛んに行われている。

これらの方法論は、アルゴリズムを試行錯誤的に特化させながら性能の改善を目指すという点で挑戦的である一方、通常のメタヒューリスティクスよりもはるかに大きく複雑な探索空間を扱う困難さがある。したがって、メタヒューリスティクスと同一の基準 (同じ評価回数) で性能を比較するとき、アルゴリズムを自律的に特化させる方法の性能が著しく劣る場合は極めて多い。そこで、ハイパーヒューリスティクスなどでは、調整用問題を設定し、評価回数の制限をなくした想定でアルゴリズムを自動生成および選定する場合もある。しかしながら、シミュレーションを用いて解を評価する高計算コストな最適化問題では評価回数が制限されるだけでなく、調整用問題 (類似問題) の設計が困難である場合も実社会で多く存在する。したがって、実用的な観点から考えると、通常のメタヒューリ

スティクスが適用される同一の基準のもとで、アルゴリズムを試行錯誤的に特化させながらメタヒューリスティクスを超える性能を導出することが究極的な目標となる。

この目標を達成するために、メタヒューリスティクスのハイパーパラメータの調整に焦点を当てることで、アルゴリズム調整の自由度を小さくする方法が多く存在する。例えばAPSO-VI[24] のように metaEA に総称される進化的な調整方法がある。特に差分進化法 (Differential Evolution, DE)[18] はシンプルにして強力な性能を持つため注目され、以来次のようなバリエーションが次々に提案されている。jDE[2] では、ハイパーパラメータの連続実数値におけるランダムな変更を子個体の性能改善時に採用する。EPSDE[13] では、jDE と類似しているがハイパーパラメータを連続実数値ではなく、事前に用意した離散値を内包するプールからランダムに選出する。JADE[25] では、子個体の性能改善時に正規分布やコーシー分布の平均値を定義式によって調整し、その分布からハイパーパラメータ値をサンプリングする。MDE[10] では、JADE と類似しているが、分布平均値の更新式に乱数係数を用いる。SHADE[20] は JADE に加えメモリ (アーカイブ) を用意することにより、外れ値に対する柔軟な対応をすることに成功している。さらに近年では深層強化学習を用いた方法 [17] も提案されている。これらの方法は、ハイパーパラメータを事前に定義する通常のメタヒューリスティクスよりも優れた最適化性能を導出している。これらの成功を受け、近年ではハイパーパラメータだけでなく、突然変異戦略や交叉戦略などの遺伝的操作も適応的に選択するように、アルゴリズム調整の自由度を高める研究動向がある。特に、解生成に直接的に関連する遺伝的操作は、問題に特化する度合をさらに高めることができ、その適切な調整ができれば最適化の性能がさらに改善することが期待できる。例えば、文法構造に発想を得た Grammatical Evolution (GE)[16, 14, 1] などがあり、[Miranda+, 2018] では、粒子群最適化 (Particle Swarm Optimization, PSO)[12] のアルゴリズムを遺伝的プログラミングで適応的に調整する方法が提案されている。本研究では、この動向を受けて、ハイパーパラメータと遺伝的操作の両方を調整する自由度の高い技術に焦点を当てる。

しかしながら、先に述べた通り、アルゴリズム調整の自由度を高める方針によって期待しうる性能の改善効果は、探索空間の膨大化によってその達成が困難となる。加えて、ハイパーパラメータと遺伝的操作の調整については、その操作で用いるハイパーパラメータの組合せによって挙動が変化するため、ハイパーパラメータのみを調整する場合と比べて、調整の複雑さは格段に増す [14]。既存手法では、この解析困難な調整に関する複雑さに対し、遺伝的プログラミングなどの進化的な調整方法を用いている。しかしながらこのような進化的な調整方法にも、1) アルゴリズムの調整にも解評価を用いるため評価回数が増え、2) 他のアルゴリズムと比較しながら調整しないため効率的にアルゴリズム調整がで

きない、という問題点が依然として残る。

加えて、先程紹介した DE のバリエーションや GE は「経験ベース」という特徴があると言える。なぜなら、「良い性能を導出したアルゴリズムの設定をできるだけ再現すれば、今後も良い性能を導出するアルゴリズムに調整できる」という仮定の下、仮定を満たすように評価回数を費やしながら調整する等のケースが多いからである。しかしながら、これらの手法は 3) ある程度まとまった回数の解評価を行うまでそのアルゴリズムの良し悪しはわからない、という弱点がある。特に先に述べた、シミュレーションベース等の解評価が高コストな問題では、ユーザは 1 試行をできるだけ少ない解評価回数で終えたいため、十分な経験が得られるまでに評価回数や計算コスト、実時間が必要となる「経験ベース」の手法は大きなディスアドバンテージとなる。

1.2 研究目的

本研究の目的は、1.1 で述べた 3 点の問題点に取り組む、従来の進化的な調整法とは異なるフレームワークを構築し、最適化の性能を向上することである。そこで、競争均衡原理 [22] に着想を得た、複数のアルゴリズムを競合させる調整技術を提案する。競争均衡原理とは、「外乱のない一定条件で競争を繰り返すと均衡最適状態に収束する」という経済学における理論である。この調整方法を実現すれば、先述の問題点を次のように解決できる。1) アルゴリズム調整に追加の解評価を要さず、2) 順位が低いアルゴリズムは教師データ (調整の目標となる上位個体) をもとに生成でき、3) 十分な経験を経ずとも質の高い「質ベース」の調整を実現する。言い換えると、従来の進化的な調整方法に対し、解評価をする前に試行錯誤的なアルゴリズム調整が行えるとともに、教師あり学習に近いフレームワークによって妥当性の高いアルゴリズム調整が行えることが期待できる。ここで言う「妥当性の高い」とは、複数のアルゴリズムを扱うため、劣ったアルゴリズムは、優れたアルゴリズムが出した評価値とほぼ同じ評価値を導出できるであろう解を生成できるように、解評価前に調整される。なお、このフレームワークは特定のアルゴリズムに偏らない汎用的なものを目指したことや、複数アルゴリズムのアンサンブル効果も目的の一つとなる。

1.3 研究方法

1.2 を受けて、提案手法におけるアルゴリズム調整の原理は、「ある一定の世代で探索を中断し、新たな解評価を行わずに複数のアルゴリズムを競合させてより適切なアルゴリズ

ムになるよう調整する」というものである。具体的には、複数のアルゴリズムを用意し、それらによる探索を定期的に中断する。続いて得られた個体を競合させそれに順位付けを行い、それらを導出したアルゴリズムについても順位付けを行う。そして、順位が高いアルゴリズムは手を加えず保存し、順位が低いアルゴリズムは上位個体を目標として自身の構成要素を調整する。

提案手法の有効性を評価する検証方法としては、8 つの実数値連続単一目的ベンチマーク関数 [3, 2] を用いて、その性能 (評価値) の比較と有効性の検定を行う。今回はケーススタディとして、複数のアルゴリズムはすべて独立な DE としたので、次のような比較をする。オリジナルの DE と、ハイパーパラメータを適応的に調整する jDE に対し、同一の評価回数において提案手法の性能を比較する。

1.4 論文構成

本論文の構成は次の通りである。第 2 章では今回ソルバーとして機能する DE とそれに対する従来のパラメータ調整方法 (jDE) を要素技術として紹介し、アルゴリズムの構成要素の特徴も同時に述べる。第 3 章では、提案手法の詳細について解説する。第 4 章で実験に用いたベンチマーク問題の紹介と問題性質の説明をし、第 5 章でそこにおける性能結果と検定結果を示し、第 6 章で考察を述べる。最後に、第 7 章で結論と今後の展望を述べる。

第 2 章

要素技術

最適化問題とは、諸目的における最適解を求める問題のことを指す。例えば実数値連続単一目的最小化問題では、決定変数を $\boldsymbol{x} \in \mathbb{R}^D$ とする目的関数 $f(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}$ に対し、D 次元の大域的最適解 $\boldsymbol{x}^* = [x_1, x_2, \cdots, x_D]$ は次のような式で表される。

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

実社会における諸問題は中身の見えないブラックボックスであり目的関数 $f(\boldsymbol{x})$ の形状や勾配といった問題の事前知識は与えられないため、様々な問題に適用可能な最適化技術としてのメタヒューリスティクスはその有効性が認知されている。メタヒューリスティクスの代表として進化アルゴリズム (Evolutionary Algorithm, EA) が挙げられ、EA は複数の解を用い一つ一つを個体とみなして群として扱う。例としては、遺伝的アルゴリズム (Genetic Algorithm, GA)[9]、粒子群最適化 (Particle Swarm Optimization, PSO)[12]、差分進化法 (Differential Evolution, DE)[18]、共分散行列適応進化戦略 (Covariance Matrix Adaptation Evolution Strategy, CMA-ES)[8] などが存在する。第 1 章で述べたように、差分進化法 (DE) はその構成要素の種類が豊富でこれまでに様々な研究がされている手法である。よって今回はこの DE についてのケーススタディを行うため、以下は DE とその構成要素 (突然変異戦略・交叉戦略・ハイパーパラメータ F, CR) の説明に焦点を当てる。また、第 5 章で行う実験で比較手法として用いた、ハイパーパラメータの適応的調整を行う Self-Adaptive DE (jDE)[2] の説明も行う。

2.1 差分進化法 (DE)

個体群を用いたメタヒューリスティクスの一つである DE[18] は、他の EA に比べてシンプルにして強力なその性能 [4] を持つことから、今日に至るまで様々な研究がなされている。

DE は上で例示した他の EA とは異なり、特定の生物の振る舞いを模倣したものではなく、Nelder-Mead 法 [15] に着想を得た手法である。Nelder-Mead 法は勾配情報を利用しない決定的探索アルゴリズムであり、目的関数の次元数 D に対し $D+1$ 個の探索点から構成される個体に反射・拡張・縮小の操作を行い子個体を生成し、次いで比較・置換を行う。同手法は $D \leq 2$ においては State-of-the-Art な EA に対しても比較的優位な性能を持つが、それ以上の次元数や多峰性関数においては性能は格段に落ちる [6]。

対して極端な高次元問題でない限り高性能な DE は、Nelder-Mead 法の「反射」を発展させ、GA のように初期化・突然変異・交叉・淘汰 (選択) からなるアルゴリズムとしている。その詳細は以下に示す通りである。

2.1.1 DE のアルゴリズム

DE の疑似コードを Algorithm 1 に掲載する。

■初期化　まず、*Initialization* とあるように、世代数 $t = 0$ において、個体数 N の数だけ初期解を生成する。具体的には、i 番目の個体 $\boldsymbol{x}_i$ $(i = 1, 2, \cdots, N_P)$ に対し、j 次元目の要素 $x_{i,j} (j = 1, 2, \cdots, D)$ を一様分布乱数 $rand[0, 1]$ を用いて式 (2.2) より生成する。そしてこれを個体群 $\mathcal{P}$ に追加する。ここで、D は問題の次元数である。また、$x_{\max,j}, x_{\min,j}$ は j 次元目の定義域の最大値と最小値である。

$$x_{i,j} = (x_{\max,j} - x_{\min,j}) rand[0, 1] + x_{\min,j} \tag{2.2}$$

■突然変異　次に、t 世代目の i 番目の個体の突然変異個体 $\boldsymbol{v}_i$ を世代 t における個体群 $\mathcal{P}$ から生成する。現在では、表 2.1 にまとめるように代表的な突然変異戦略は複数提案されている。ここで、表 2.1 にまとめた突然変異戦略について、ハイパーパラメータ $F \in [0, 1]$ は差分ベクトルの寄与率を調整するスケールファクターであり、F の値に応じて大域探索と局所探索のバランスを制御する効果がある。本研究では、表 2.1 に示す 7 種類の突然変異戦略を DE の調整時における選定候補と定める。

Algorithm 1 差分進化法 (Differential Evolution, DE)

$t = 0$
for $i = 1$ **to** N **do**
  $\boldsymbol{x}_i \leftarrow$ *Initialization*
  Add $\boldsymbol{x}_i$ *to* $\mathcal{P}$
end for
while $t < t_{\max}$ **do**
  for $i = 1$ **to** N **do**
    $\boldsymbol{v}_i \leftarrow$ *Generate mutant individual from* $\mathcal{P}$
    $\boldsymbol{u}_i \leftarrow$ *Generate solution via crossover*
  end for
  for $i = 1$ **to** N **do**
    if $f(\boldsymbol{u}_i) < f(\boldsymbol{x}_i)$ **then**
      $\boldsymbol{x}_i \leftarrow \boldsymbol{u}_i$
    else
      $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i$
    end if
  end for
  $t = t + 1$
end while

rand/1 や *rand/2*、*current-to-rand/1* は、ランダム性が強いため大域探索を行う効果が強い。一方で、*best/1* や *best/2*、*current-to-best/1* は、世代 t における最良解の周辺を探索するように突然変異が行われる。また、それぞれの突然変異戦略の中でも、末尾の数字が大きいほど加味される差分ベクトルの数が増えるため、*rand/1* や *best/1* よりも *rand/2* や *best/2* の方が生成される解の多様性が向上する傾向があると考えられる。さらに、*current-to-rand/1* や、*current-to-best/1*、*current-to-pbest/1* は、親個体付近に個体を生成する傾向がある。なお、*current-to-pbest/1* で用いるパラメータ p は、局所探索と大域探索を調整可能にするパラメータであり、本論文では簡単のため、広く使われている 1/2 に固定する。

例として、*rand/1* であれば $\boldsymbol{v}_i$ は次の式 2.3 のように求められる。ここで、$\boldsymbol{x}_{r_1}$, $\boldsymbol{x}_{r_2}$, $\boldsymbol{x}_{r_3}$ は現在の個体群 $\mathcal{P}$ より自身 $\boldsymbol{x}_i$ 以外でランダムに選出したものである。

表 2.1　DE における代表的な突然変異戦略

index	名称	定義式
1	*rand/1*	$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3})$
2	*rand/2*	$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}) + F(\boldsymbol{x}_{r_4} - \boldsymbol{x}_{r_5})$
3	*best/1*	$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2})$
4	*best/2*	$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}) + F(\boldsymbol{x}_{r_3} - \boldsymbol{x}_{r_4})$
5	*current-to-rand/1*	$\boldsymbol{v}_i = \boldsymbol{x}_i + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_i) + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3})$
6	*current-to-best/1*	$\boldsymbol{v}_i = \boldsymbol{x}_i + F(\boldsymbol{x}_{best} - \boldsymbol{x}_i) + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2})$
7	*current-to-pbest/1*	$\boldsymbol{v}_i = \boldsymbol{x}_i + F(\boldsymbol{x}_{pbest} - \boldsymbol{x}_i) + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2})$

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}) \tag{2.3}$$

■交叉　交叉戦略も複数あり binomial 交叉 (Algorithm 2) と exponential 交叉 (Algorithm 3) について説明する。突然変異個体 $\boldsymbol{v}_i$ を生成後、次元カウンタ j を走らせながら親個体 $\boldsymbol{x}_i$ と交叉させ、完了後のベクトルを $\boldsymbol{u}_i$ とする。その際、交叉率パラメータ CR を用いるが、全く交叉されない可能性を排除するために $[1, D]$ から整数乱数 j_{rand} を決め、j_{rand} 次元目は必ず交叉させる。なお、CR は突然変異個体の決定変数をどの程度寄与させるかを制御する役割がある。例えば、$CR = 0.0$ では、Algorithm 2 の j_{rand} あるいは Algorithm 3 の初回ループの影響により、親個体に対し決定変数は 1 つのみ変更されるので、1 次元ごとの垂直あるいは水平方向の探索が可能となる。一方、$CR = 1.0$ では垂直あるいは水平方向の探索ではなくなるものの、軸変換に対する回転不変性を持つため、変数分離不可能な関数に適した設定と言える [27]。

高次元問題であるほど binomial より exponential が適しているという報告がなされている [26]。疑似コードを見ればわかるように、交叉率 CR により決定する交叉長 L に対する期待値 $E(L)$ は、二項分布に従う binominal(Algorithm 2, 式 (2.5)) で線形、幾何分布に従う exponential(Algorithm 3, 式 (2.7)) では非線形であり次元数を変数とする指数関数のような形状をしている [27]。従って、後者では問題の次元数に適した CR との組み合わせを構成することで効果を得ることができる。

$$p_m = CR(1 - 1/D) + 1/D \tag{2.4}$$

$$E(L) = Dp_m = (D - 1)CR + 1 \tag{2.5}$$

Algorithm 2 binomial 交叉

$j_{rand} \leftarrow$ *randomly select an index of dimension from* $[1, D]$
for $j = 1$ **to** D **do**
  if $rand[0, 1) < CR$ **or** $j == j_{rand}$ **then**
    $u_{i,j} = v_{i,j}$
  else
    $u_{i,j} = x_{i,j}$
  end if
end for

Algorithm 3 exponential 交叉

$j = randint\ [1, D]$
$k = 1$
$\boldsymbol{u}_i \leftarrow \boldsymbol{x}_i$
repeat
  $u_{i,j} = v_{i,j}$
  $j = (j + 1) \bmod D$
  $k = k + 1$
until $rand\ [0, 1) \geq CR$ **or** $k \geq D$

$$P(L = h) = \begin{cases} (1 - CR)CR^{h-1}, & h < D \\ CR^{D-1} & otherwise \end{cases} \tag{2.6}$$

$$\begin{aligned} E(L) &= \left((1 - CR) \sum_{h=1}^{D} -1(hCR^{h-1}) \right) + D(CR^{D-1}) \\ &= \frac{1 - CR^D}{1 - C} \end{aligned} \tag{2.7}$$

■**選択**　次に、式 (2.8) に示す通り、個体 $\boldsymbol{u}_i$ の評価値が元の個体 $\boldsymbol{x}_i$ よりも小さい場合に $\boldsymbol{x}_i$ を $\boldsymbol{u}_i$ に設定する。そして、突然変異個体の生成に戻り一連の遺伝的操作を繰り返す。この操作を探索終了条件 (例：最大世代数, 最大評価回数) まで行う。

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{u}_i, & f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i) \\ \boldsymbol{x}_i, & otherwise \end{cases} \tag{2.8}$$

2.2 Self-Adaptive DE (jDE)

2.2.1 jDE のアルゴリズム

通常の DE では F, CR が全個体かつ探索中に終始一定であったのに対し、jDE では F, CR のパラメータを適応的に変更する。具体的には、F, CR が各個体 $\boldsymbol{x}_i$ ごとに固有の値 F_i, CR_i を持ち、それぞれの値が突然変異・交叉時に更新される。まず、すべての F_i, CR_i をそれぞれ初期値 $F_i = 0.5, CR_i = 0.9$ に設定し、各個体に対し一定確率 τ_F で F_i を一時的に rand[0.1, 1] に変更した値を F_i^{tmp} とする。なお、変更されない場合はそのまま F_i を引き継ぐことになる。次に、F_i^{tmp} を用いて突然変異個体 $\boldsymbol{v}_i$ を生成する。同様に一定確率 τ_{CR} で CR_i を一時的に rand[0, 1] に変更した CR_i^{tmp} を用いて $\boldsymbol{u}_i$ を生成する。そして、$\boldsymbol{u}_i$ の評価値が親個体 $\boldsymbol{x}_i$ よりも優れた場合に、F_i^{tmp} と CR_i^{tmp} を採用し次世代に用いる。$\boldsymbol{u}_i$ の評価値が改善されない場合は、F_i^{tmp} と CR_i^{tmp} は破棄される。したがって、「問題や探索状況に適したパラメータを用いたことにより優れた子個体の生成が行えた」という仮定のもと、過去の探索において有用であったパラメータ F_i, CR_i を引き継ぎながら、一定確率で新しいパラメータを試行錯誤的に求める。

Algorithm 4 jDE

$t = 0$
for $i = 1$ **to** N **do**
  $\boldsymbol{x}_i \leftarrow$ *Initialization*
  Add $\boldsymbol{x}_i$ *to* $\mathcal{P}$
end for
while $t < t_{\max}$ **do**
  for $i = 1$ **to** N **do**
    if $rand[0,1] \leq \tau_F$ **then**
      $F_i^{tmp} = rand[0.1, 1]$
    else
      $F_i^{tmp} = F_i$
    end if
    if $rand[0,1] \leq \tau_{CR}$ **then**
      $CR_i^{tmp} = rand[0,1]$
    else
      $CR_i^{tmp} = CR_i$
    end if
    $\boldsymbol{v}_i \leftarrow$ *Generate mutant individual from* $\mathcal{P}$ *with* F_i^{tmp}
    $\boldsymbol{u}_i \leftarrow$ *Generate solution via crossover with* CR_i^{tmp}
  end for
  for $i = 1$ **to** N **do**
    if $f(\boldsymbol{u}_i) < f(\boldsymbol{x}_i)$ **then**
      $\boldsymbol{x}_i \leftarrow \boldsymbol{u}_i, F_i = F_i^{tmp}, CR_i = CR_i^{tmp}$
    else
      $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i, F_i = F_i, CR_i = CR_i$
    end if
  end for
  $t = t + 1$
end while

第 3 章

提案手法

この章では、第 1 章で述べた問題に対する解決策としての提案手法の着想のコンセプトと、その詳細メカニズムを示す。

3.1　競争均衡原理の着想

コンセプトイメージを図 3.1 に示す。今回着想を得た競争均衡原理は経済学の一原理であり、一文で表すと「外乱のない一定条件で競争を繰り返すと均衡最適状態に収束する」となる。つまり、複数の経済主体が価格競争を通して、やがてある理想的な均衡状態に収束することを主張するものである。

具体的には、図 3.1 の上部において、A, B, C の 3 つの経済主体を考えることで説明する。この主体は経済的競合関係にあるものであれば、個人・店舗・企業の種や主体の数を問わない。各主体は自身が持つ製品を売ることで利潤の最大化を図るものとする。そのため、生産過程のコストカットや経営体制の見直し、研究開発等を通して、競合に負けず売れるようにすべく自社製品の価格を下げる企業努力をする。しかし実際の市場は、税制や政府介入、為替、国際情勢といった複雑なダイナミクス下にある。これらは原理における「外乱や変化する条件」となるので、原理が求める条件を満たすためにはこれらを一旦度外視する必要がある。つまり、定常状態を仮定する。すると、原理が主張するには、競合関係にある経済主体は一定条件下でやがてある均衡状態に到達するという。言い換えると、理想的な条件下では、その条件に達した時点で劣った (製品価格を下げられなかった) 経済主体が同じ条件で企業努力を重ねることで、複数の経済主体全体がほぼ同じ価格の製品を生み出すことになる。興味深いことに、原理は経済主体同士が同じ戦略をとることは仮定せずとも、価格競争の圧力による均衡は成立する。

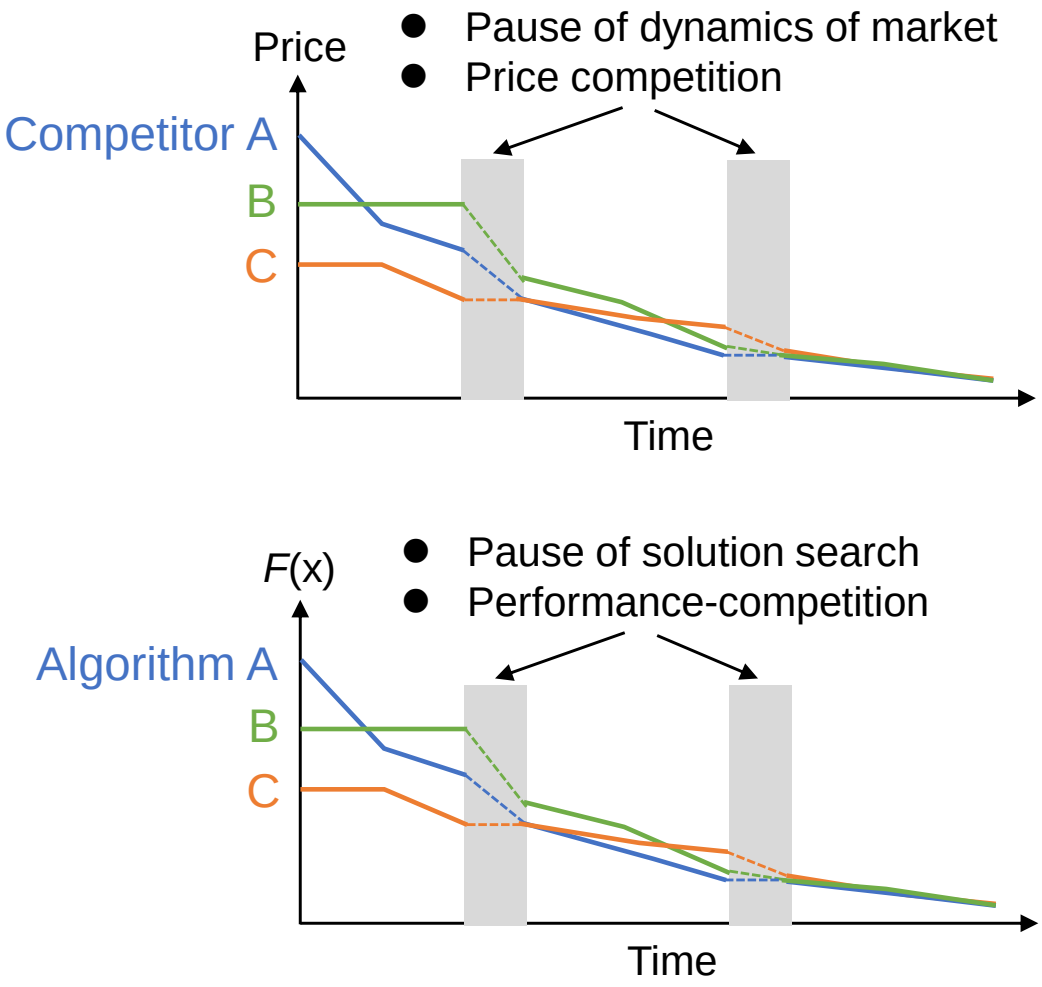


図 3.1　競合しながら適応的にアルゴリズム調整をする提案手法のコンセプト図

これを踏まえ、図 3.1 の下部を用いて、提案手法における原理の応用方法を述べる。まず、A, B, C のように複数のアルゴリズムを用意する。これも原理と同様に種類や個数を問わない。なお、同じアルゴリズムであったとしても、競合をさせるためにその構成要素 (ハイパーパラメータや遺伝的操作等) は異なるものとなるように設定する。一般に、それぞれのアルゴリズムは最小化問題を解くため、解探索というダイナミクス下で複雑な (場合によっては制約付き) 目的関数の評価値の最小化を図る。これは原理における、諸条件下での価格競争に該当する。次に、あるタイミングで解探索を一時停止し、すべてのアルゴリズムが導出した解でアルゴリズムの優劣を判定する。そして優れたアルゴリズムは調整せずに、解探索を再スタートする点まで引き継ぐ。劣ったアルゴリズムは、優れたアルゴリズムが導出した解に類似した解を導出できるように、再スタート時までに調整する。

これを定期的に繰り返すことで、全体としての性能向上かつ均衡的収束が期待できる。

3.2 アルゴリズム調整の準備

提案手法では、複数のアルゴリズム (今回はケーススタディなのですべて DE) を用意しそれらが生成する個体を定期的に競合させる点に着想を得ている。ここで、提案手法はアルゴリズムレベルの粒度で調整をする点で、従来の解個体レベルの粒度で調整する jDE や JADE, MDE, SHADE, SaDE らとは異なる。この競合の原理を活かすことで DE の構成要素であるハイパーパラメータおよび突然変異戦略、交叉戦略を適応的に調整する。具体的には、一定の世代ごとに全アルゴリズムが導出した解個体の順位をつける。そして、順位が高い個体を輩出したアルゴリズムは、評価値の高い解個体が生成できたことを根拠に、それらのアルゴリズムの構成要素が解探索に有効であると判定し調整しない。一方で、順位が低い個体を生成したアルゴリズムはその構成要素を調整することで、解探索に有効なアルゴリズムとなるように構築する。このために、順位が高い (調整の目標となる) 解個体に近い解を現在の解集合から生成できるまで、アルゴリズムの構成要素を調整する。

より厳密に定義をするために、構成ベクトル θ を導入する。今回は DE であるので、θ は以下の 4 要素を持つとする。

- 突然変異戦略のインデックス $x_v = \{1, 2, , \cdots, 7\}$
  カテゴリカル変数
  表 2.1 に対応
- スケール係数 $x_F = F \in [0, 1]$
  実数値
  表 2.1 における突然変異ベクトルの寄与具合を調整
- 交叉戦略のインデックス $x_u = \{0, 1\}$
  バイナリ変数
  0 が binomial 交叉 (Algorithm2) に、1 が exponential 交叉 (Algorithm3) にそれぞれ対応
- 交叉率 $x_{CR} = CR \in [0, 1]$
  実数値
  Algorithm2, 3 における交叉の度合いを調整

なお、アルゴリズム集合 $\mathcal{A}$ についても定義をする。$\mathcal{A}$ は n 個の要素 (アルゴリズ

Algorithm 5 提案手法

for $i = 1$ **to** n **do**
  $\mathcal{P}^i, A^i \leftarrow$ *Initialization*
end for
while TerminationCriteria ==**false do**
  for $l = 1$ **to** I **do**
    for $i = 1$ **to** n **do**
      $\mathcal{P}^i \leftarrow$ *Validation*$(\mathcal{P}, A^i)$
    end for
  end for
  $\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \cdots, \boldsymbol{x}_K^*, \boldsymbol{x}_{K+1}^*, \cdots, \boldsymbol{x}_n^* \leftarrow$ *Competition*$(\mathcal{P})$
  for $i = 1$ **to** $n - K$ **do**
    $A^{K+i}, \mathcal{P}^{K+i} \leftarrow$ *Tuning*$(\mathcal{P})$
  end for
end while

ム)$A^i, i = 1, 2, \cdots, n$ を持ち、$\mathcal{A} = \{A^1, A^2, \cdots A^n\}$ と表記する。今回のケーススタディでは、すべての A^i は DE であるので、A^i に代入される DE は構成ベクトルは $\theta^i = \{x_v^i, x_F^i, x_u^i, x_{CR}^i\}$ となり、$A^i \leftarrow DE(\theta^i)$ と表記できる。言い換えると、DE は θ^i を用いて 1 を実行する。さらに、各 A^i は個体群 $\mathcal{P}^i$ を持ち、$|\mathcal{P}^i| = N$ である。また、各 A^i が持つ j 番目の解個体は $\boldsymbol{x}_j^i \in \mathcal{P}^i$ と表記される。

ここで、構成ベクトルはメタヒューリスティクスによって自由に拡張することができることに注意されたい。次のセクションで説明するように各アルゴリズム A^i は独立に調整されるため、異なるメタヒューリスティクスで $\mathcal{A}$ を構成して、アンサンブル効果を狙うこともできる。しかしながら、今回はフレームワークの基本的な効果を検証するため、この拡張は今後の課題として残しておく。

3.3 メカニズム

提案手法の概要図を図 3.2 に、提案手法の流れを Algorithm 5 に示す。
提案手法は次の 4 つのコンポーネントからなる。

1. 初期化 (*Initialization*)
   各アルゴリズムの個体群 $\mathcal{P}^i$ とアルゴリズム設定 A^i の初期化
2. 検証 (*Validation*)
   すべてのアルゴリズムで一定期間行われる解探索

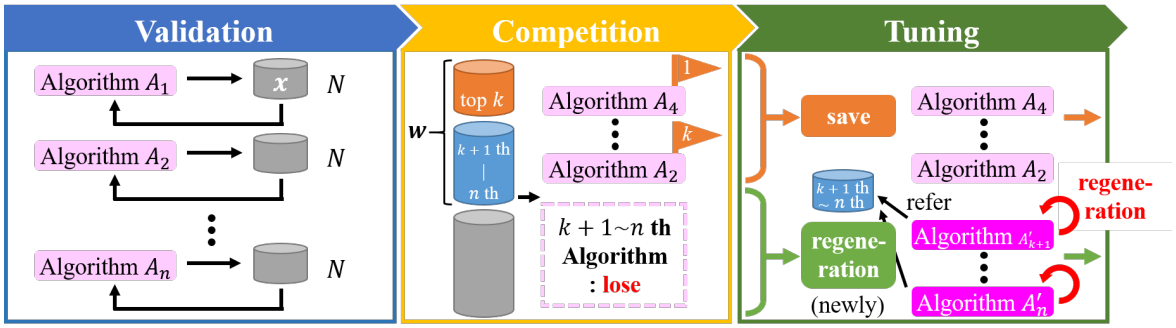


図 3.2　提案手法のアルゴリズム概要図

3. 競合 (*Competition*)
   解探索の結果からアルゴリズムの優劣を決定
4. 調整 (*Tuning*)
   劣ったアルゴリズムを優れたアルゴリズムが見つけた解の類似解を見つけられるように調整

最初に、初期個体群 $\mathcal{P}^i$ と複数のアルゴリズム $\mathcal{A}$ をそれぞれ初期化する *Initialization*。次に、$Validation(\mathcal{P}, A^i)$ として示すように、複数のアルゴリズムを用いて解集合 $\mathcal{P}$ を生成する。この操作を、I 世代だけ繰り返すことで、調整されたアルゴリズムによって評価値が高い解個体が生成できるかを検証する。そして、$Competition(\mathcal{P})$ として示すように、生成された解集合 $\mathcal{P}$ を用いて優良解 $\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \cdots, \boldsymbol{x}_K^*, \boldsymbol{x}_{K+1}^*, \cdots, \boldsymbol{x}_n^*$ を得る。続いて、$Tuning(\mathcal{P})$ として表記されるように、順位が低いアルゴリズムは選定された優良解 $\boldsymbol{x}_{K+1}^*, \cdots, \boldsymbol{x}_n^*$ に近い解を生成できるまで、その構成要素の調整を行う。したがって、I 世代ごとに検証期間としてのインターバルを設け、アルゴリズム調整を行うことになる。そして再び、$Validation(\mathcal{P}, A^i)$ に戻り、調整されたアルゴリズムを用いて解生成を行う。この一連の操作を繰り返すことで、アルゴリズム調整を行いながら、評価値の高い解の生成を目指す。

以下には、それぞれの詳細について述べる。

3.3.1　初期化 (*Initialization*)

初期個体群 $\mathcal{P}^i$ の生成は通常の DE と同様に、式 (2.2) を用いて生成する。

初期アルゴリズム $\mathcal{A}$ の生成にあたり、各アルゴリズム A^i の設定ベクトル θ^i は次のように設定される。ハイパーパラメータ x_F^i, x_{CR}^i に関してはすべての i に共通して、jDE

等の初期値で広く使われている規定値 $(F = 0.5, CR = 0.9)$ に設定する。ただし、突然変異戦略 x_v と交叉戦略 x_u については、アルゴリズムとしての多様性を担保するためにそれぞれ異なる戦略を設定する。そこで、n 個のアルゴリズムに対し、突然変異戦略と交叉戦略の組合せが異なるように初期設定する。

3.3.2 検証 (*Validation*)

このコンポーネントは、通常のメタヒューリスティクスにおける解探索に対応するが、提案手法では、調整されたアルゴリズムが評価値の高い解が生成できるかを検証する役割がある。n 個の各アルゴリズムは調整されたアルゴリズム A^i は、構成ベクトル θ^i に基づく DE に従って、個体数 N だけ解 $\mathcal{P}^i$ を更新する。この操作をハイパーパラメータ I 世代だけ繰り返す。I 世代だけ繰り返した後、全てのアルゴリズムが生成した解を統合した解集合 $\mathcal{P}$ を次に述べる競合で扱う。したがって、提案手法が必要となる全体の解個体数は $|\mathcal{P}| = N \times n$ となる。また、このコンポーネントで必要な総評価回数は $N \times n \times I$ となる。加えて、提案手法では解評価は解探索、つまりより良い解を求めるためのみに使われ、調整では使用されない。

3.3.3 競合 (*Competition*)

生成した解を統合した解集合 $\boldsymbol{x}$ について、評価値の良い順に上位 n 個体 $\boldsymbol{w} = [w_1, w_2, \cdots, w_n]$ を選出し、優良個体として扱う。なお、w の添え字は順位を示している。したがって、w_1 は最も評価が高い解個体を意味する。ここで、各上位個体 w_i は、どのアルゴリズムから生成されたかが記憶される。

その後、アルゴリズム集合 $\boldsymbol{A}$ を空集合に設定し、保存すべきアルゴリズムを追加する。まず、$\boldsymbol{w}$ の中でもさらに上位 $k(0 \leq k \leq n)$ の解個体 w_i を生成したアルゴリズムは、評価の高い解個体が生成できたことを根拠に、それらのアルゴリズムは調整せずに保存される。本稿では $k = 3$ と設定している。したがって、次に試す n 個のアルゴリズムのうち、k 個のアルゴリズムは優良解を生成した既存のアルゴリズムが継承される (エリート保存)。また、上位 k 個の解個体 w_i が同一のアルゴリズムから生成された場合は、重複を許容して保存する。例えば、上位 k の解個体 w_i の全てが同じアルゴリズム A_i から生成されれば、A_i を 3 つに複製しアルゴリズム集合に保存する。この場合、A_i が最も有力なアルゴリズムであることが考えられるため、そのアルゴリズムの解探索をより促進するように、$3 \times (N_P/n)$ に個体数を増やしたアドバンテージを与える。

Algorithm 6 *Algorithm Tuning*()

while $|\mathcal{P}^i_{\epsilon_d}| < \theta_N$ **do**
  $\theta_i \leftarrow$ *set random values of* x^i_v, x^i_F, x^i_u, *and* x^i_{CR}
  for $j = 1$ **to** N **do**
    $\boldsymbol{v}^i_j \leftarrow Mutation(x^i_v, x^i_F, \mathcal{P})$
    $\boldsymbol{u}^i_j \leftarrow Crossover(x^i_u, x^i_{CR})$
    Add $\boldsymbol{u}^i_j$ *to* $\mathcal{P}^i$
  end for
  $\mathcal{P}^i_{\epsilon_d} \leftarrow \{\boldsymbol{u}^i_j \in \mathcal{P}^i \mid ||\boldsymbol{u}^i_j - \boldsymbol{x}^*_{K+i}|| < \epsilon_d\}$
end while
return $DE(\theta^i)$, $\mathcal{P}^i$

一方で、残りの $n-k$ 個のアルゴリズムについては調整を経て新規に生成される。

3.3.4 調整 (*Tuning*)

アルゴリズム調整の詳細について、Algorithm 6 に示す。$n-k$ 個のアルゴリズムを生成するにあたり、調整目標となる優良解 w を設定する。ただし、上位 k の優良個体 $w_i(i = 1, \cdots, k)$ については、それらを生成したアルゴリズムが保存されているため、調整目標と設定しない。これは、調整するアルゴリズムの多様性を担保するためである。実際、事前実験では、アルゴリズムの調整目標を同一 (例えば w_1 のみ) に設定する場合、それぞれのアルゴリズムは同じ決定変数の部分領域を探索する結果、局所解に陥る確率が高くなる。言い換えると、調整目標をアルゴリズムごとに変更し、多様に調整されたアルゴリズムを用いる方が、競合が頻繁に発生し最適化の性能が改善する。そこで、$n-k$ 個のアルゴリズムの調整目標はそれぞれ $w_i(i = k+1, \cdots, n)$ に設定する。したがって、$n-k$ 個のアルゴリズムはそれぞれ異なる調整目標を持つ。ここで、アルゴリズムの調整終了条件は、調整目標 w_i を中心とし半径 d とした超球内に存在する子個体が生成されることである。この条件を達成するため、F, CR、突然変異戦略、交叉戦略をランダムで変更する。本稿では、各次元の決定変数 x_j に共通して $d = 0.1 \times (x_{j,\max} - x_{j,\min})$ と設定する。

第 4 章

問題設定

提案手法の有効性を評価するための実験で用いた問題について説明する。

4.1 ベンチマーク問題

次に示す 8 つの実数値連続単一目的ベンチマーク関数 [3, 2] を用いる。これらの関数は最小値問題であるため、第 2 章冒頭で述べた通り、その評価値 $f(\boldsymbol{x})$ を最小化する解 (大域的最適解)$\boldsymbol{x}^* = [x_1, x_2, \cdots, x_D]$ を探索することが目的となる。なお、8 つの関数の大域的最適解 $\boldsymbol{x}^*$ はすべて $\boldsymbol{x}^* = \mathbf{0}$ であり、その評価値はすべて $f(\boldsymbol{x}^*) = f(\mathbf{0}) = 0$ である。

F1 Sphere
探索領域 $[-100, 100]^D$
$F_1(x) = \sum_{i=1}^{D} x_i^2$

F2 Rosenbrock
探索領域 $[-100, 100]^D$
$F_2(x) = \sum_{i=1}^{D-1} \left(100 \left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right)$

F3 Ackley
探索領域 $[-50, 50]^D$
$F_3(x) = -20 \exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos\left(2\pi x_i\right)\right) + 20 + e$

F4 Rastrgin
探索領域 $[-50, 50]^D$
$F_4(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$

F5 Griewank

探索領域 $[-100, 100]^D$

$F_5(x) = 1 + \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right)$

F6 Weierstrass

探索領域 $[-0.5, 0.5]^D$

$$F_6(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}\left[a^k \cos\left(2\pi b^k\left(x_i + 0.5\right)\right)\right]\right) - D\sum_{k=0}^{k_{\max}}\left[a^k \cos\left(2\pi b^k \cdot 0.5\right)\right],$$

$$a = 0.5, b = 3, k_{\max} = 20$$

F7 Schwefel

探索領域 $[-500, 500]^D$

$F_7(x) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin(\sqrt{|x_i|})$

F8 Schwefel 1.2

探索領域 $[-100, 100]^D$

$F_8(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$

参考までに、上記の 8 関数を次元数 $D = 2$ としたときの概形を図 4.1 に示す。

4.2 関数性質

各関数について、以下の観点から最適化技術が直面する問題性質を述べる。

4.2.1 単峰性・多峰性 (Modality)

説明に先立って、大域的最適解 $\boldsymbol{x}^*$ を再定義する。ここで、S は探索領域を示す。

$$f\left(\boldsymbol{x}^*\right) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in S \tag{4.1}$$

またある解 $\boldsymbol{x}^l \in S$ とその近傍 $S^n\left(\boldsymbol{x}^l\right)$ について次式が成り立つとき、$\boldsymbol{x}^l$ は局所解 (局所的最適解) として次で定義される。ここで、任意の正数 ϵ に対し、$S^n\left(\boldsymbol{x}^l\right) = \left\{\boldsymbol{x} | \epsilon > \left\|\boldsymbol{x}^l - \boldsymbol{x}\right\|\right\}$ とする。

$$f\left(\boldsymbol{x}^l\right) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in S \cap S^n\left(\boldsymbol{x}^l\right) \tag{4.2}$$

つまり、周辺に対してその評価値が小さい点を局所解 $\boldsymbol{x}^l$ としている。したがって、この局所解 $\boldsymbol{x}^l$ の中でも最も評価値が小さいものを最適解 $\boldsymbol{x}^*$ と呼ぶことになる。

本題に戻って、単峰性・多峰性の定義とは、単峰性とは最適解 $\boldsymbol{x}^*$ 以外の局所解 $\boldsymbol{x}^l$ が存在しないものを単峰性とし、そうでないものを多峰性とされる。多峰性関数の中でも、

局所解が多数存在したりその深さが大きいものを定性的に「強い多峰性」や「弱い多峰性」と呼ぶこともある。特に強い多峰性では、局所解にトラップされそこから脱出することが難しいため、比較的単純とされる単峰性関数よりも難しい問題クラスであると言える [27]。

4.2.2 変数分離可能性 (Separable)

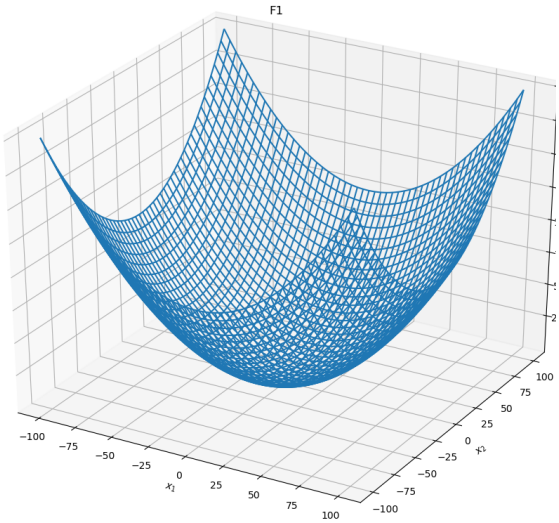
決定変数を各次元 $j \in \{1, 2, \cdots, D\}$ ごとに分離することによって、目的関数も $\sum_{j=1}^{D} f_j(x_j)$ のように分解でき、1 次元ごと、あるいは j 次元のまとまりごとに分けて問題を扱うことができる [27]。この状態のことを変数分離可能という。このようにすることで、次元数増加による計算量増加といった困難さを回避しながら全体として問題を扱うことができる。しかし、F2 や F8 のような別次元の決定変数を同時に用いて組み合わせることで構成される関数は変数分離不可能と呼ばれ、分離ができないため次元ごとの計算が行えないことになる。特に解評価のコストが高い場合では変数分離不可能性は不利な点となる。実問題では、変数分離可能な問題は少ないとされつつも [7]、すべての決定変数が完全に依存している (つまり、部分ごとにさえ分離できない) ケースもまた不自然であるとされている [21]。

4.2.3 関数の性質

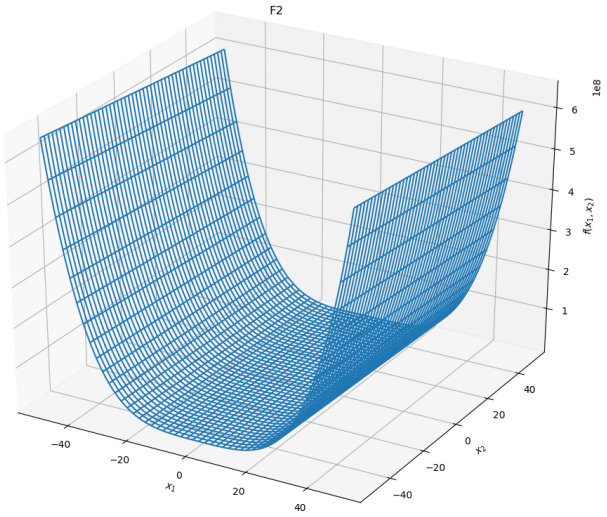
4.2.1, 4.2.2 を踏まえて、今回用いた問題の性質を表 4.1 に示す。

表 4.1　実数値連続単一目的ベンチマーク関数の性質

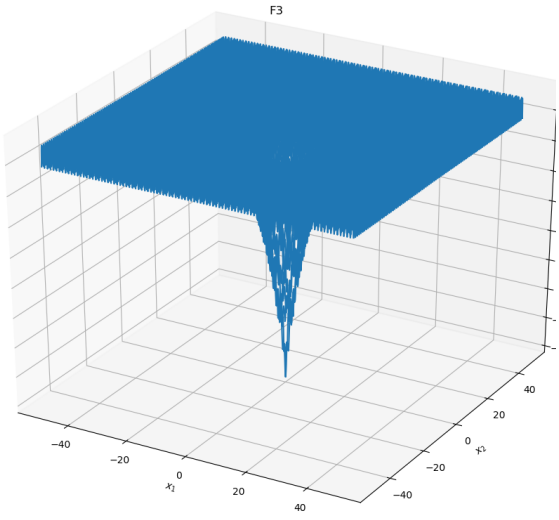
関数	名称	単峰性・多峰性	変数分離可能性
F1	Sphere	単峰性	可能
F2	Rosenbrock	単峰性	不可能
F3	Ackley	多峰性	可能
F4	Rastrgin	多峰性	可能
F5	Griewank	多峰性	可能
F6	Weierstrass	多峰性	可能
F7	Schwefel	多峰性	可能
F8	Schwefel 1.2	単峰性	不可能



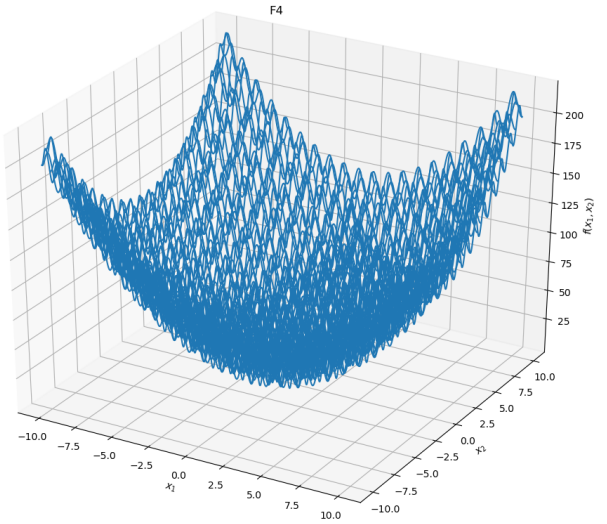
a) F1



b) F2

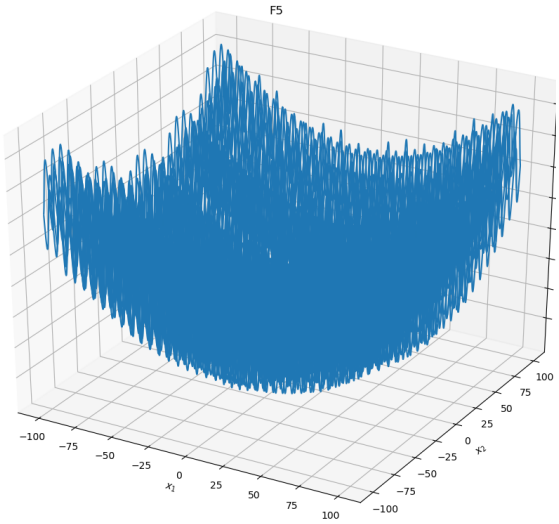


c) F3

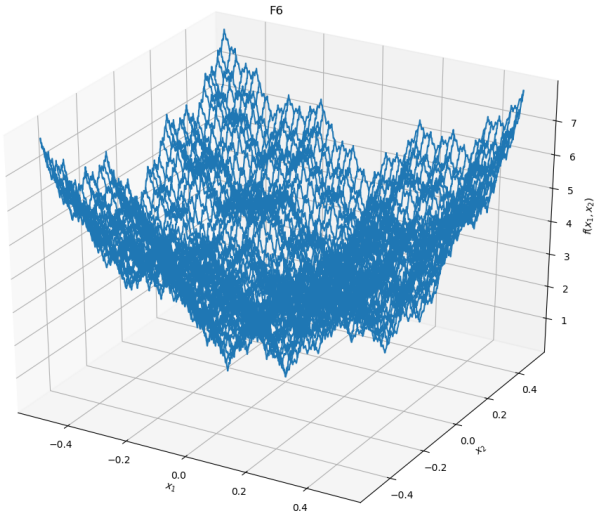


d) F4

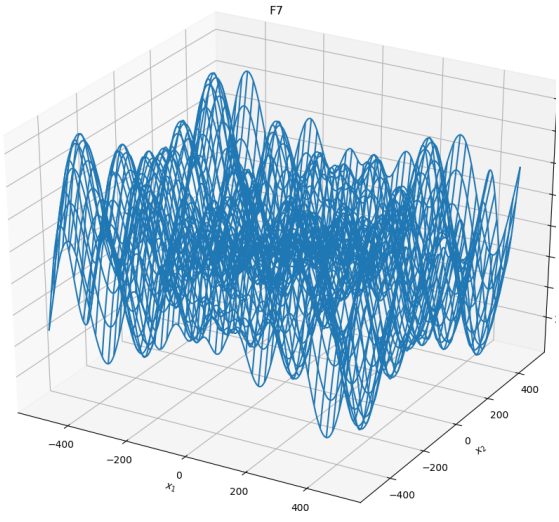
図 4.1　各関数の $D = 2$ での概形



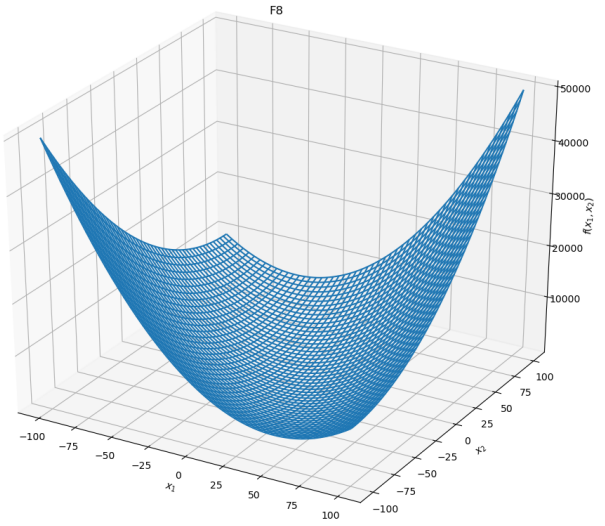
e) F5



f) F6



g) F7



h) F8

図 4.1　各関数の $D = 2$ での概形

第 5 章

実験

5.1　評価方法

第 4 章で示した 8 つの単一目的最小化問題について、最大 100,000 回の評価回数における 30 試行の中央値を比較した。各問題ごとに次元数は $D = \{10, 20, 30\}$ で実験したため、合計 24 ケースについて検討した。また、提案手法の優位性を確かめるためにフリードマン検定とホルム法によるウィルコクソンの符号順位和検定も行った。

5.2　パラメータ設定

提案手法のパラメータ設定を表 5.1 に示す。なお、DE ならびに jDE で用いるパラメータについても、表 5.1 に掲載するパラメータ値に設定する。なお、N_P, F, CR のパラメータについては、jDE の原著 [2] と同じである。また、最大評価回数 $evals_{\max}$ は、DE および jDE も同じ値に設定しており、同一の評価回数のもとで提案手法の性能を比較できる。

各ベンチマーク問題において、乱数のシードが異なる 30 試行の実験を繰り返し、得られた最良解の評価値の中央値を比較する。また、次元数 $D = 10, 20, 30$ と変化させ、提案手法のスケーラビリティを評価する。

5.3　実験結果

表 5.3 に、次元数 $D = 10, 20, 30$ と変化させたときの最良解の評価値を示す。なお、提案手法の収束速度を評価するために、最大評価回数である 100,000 回に加え、20,000 回, 50,000 回の時点で導出した最良解の評価値も示している。

表 5.1　実験パラメータ

パラメータ	説明	数値
D	次元数	10, 20, 30
N_P	個体数	100
F	スケール係数 (初期値)	0.5
CR	交叉率 (初期値)	0.9
$evals_{\max}$	最大評価回数	100,000
n	アルゴリズム数	10
I	検証世代数	5

5.4　実験結果の詳細

$D = 10$ においては、F1, F2, F3, F6, F7 および F8 で提案手法が優れた性能を導出している。特に、$D = 10$ に設定した F1, F2 および F6 では、最良解のオーダーが大きく異なり、提案手法が極めて優れた性能を導出できていることがわかる。また、$D = 20$ では、F1, F2 および F4 で提案手法が優れている。同様に $D = 30$ では、F1, F2, F4 および F8 で提案手法が優位であることがわかる。まとめると、合計 24 個の実験ケースに対し、最大評価回数時点では提案手法が 13 個の実験ケースにおいて DE ならびに jDE よりも優れた性能を導出している。なお、DE は 3 つの実験ケース、jDE は 9 つの実験ケースにおいて最良の性能を導出している。また、評価回数が 20,000 回であるとき、F3($D = 20, 30$), F7($D = 10, 20, 30$), F8($D = 10$) を除く全ての問題ケースにおいて、提案手法が優れた性能を導出していることがわかる。

一方で、$D = 10$ における F4, F5 や $D = 20$ における F5, F6、$D = 30$ における F5, F6 では提案手法が局所解に収束している。また、図 5.1 に示すように、F4 について四分位範囲も合わせた性能変化を見ると、提案手法の性能は安定していないことがわかる。一方で、30 試行中 14 試行において、第一四分位数は jDE より少ない評価回数で最適解を発見している。

以上より、上記の多くの実験ケースにおいて提案手法がハイパーパラメータのみを調整する jDE と比較して優れた性能を導出していることから、ハイパーパラメータと遺伝的操作の両方を調整する意義が確認できる。加えて、F4($D = 10$) では、jDE あるいは DE よりも劣る場合においても少ない試行数ながらも、これらを超える性能を導出している。

表 5.2　Comparison of the best fitness discovered at 1,200-th, 3,000-th, 30,000-th and 100,000-th fitness evaluations, respectively. The median value of all the 30 trials are reported.

		1,200-th			3,000-th		
id	D	DE	jDE	ours	DE	jDE	ours
F1	10	4.32E+03	3.53E+03	**1.96E+03**	7.26E+02	5.83E+02	**2.77E+02**
	20	2.00E+04	2.03E+04	**1.09E+04**	9.86E+03	8.16E+03	**3.96E+03**
	30	4.08E+04	4.29E+04	**2.46E+04**	2.30E+04	2.22E+04	**1.03E+04**
F2	10	1.74E+07	1.52E+07	**4.35E+06**	5.99E+05	4.47E+05	**8.32E+04**
	20	2.78E+08	2.98E+08	**1.08E+08**	7.65E+07	6.37E+07	**1.32E+07**
	30	9.84E+08	9.54E+08	**2.94E+08**	3.22E+08	2.88E+08	**5.63E+07**
F3	10	2.01E+01	2.01E+01	**1.93E+01**	1.99E+01	2.00E+01	**1.63E+01**
	20	2.04E+01	2.04E+01	**2.04E+01**	**2.01E+01**	**2.01E+01**	**2.01E+01**
	30	2.06E+01	2.05E+01	**2.05E+01**	**2.02E+01**	2.03E+01	**2.02E+01**
F4	10	1.04E+03	1.01E+03	**5.85E+02**	2.90E+02	2.56E+02	**1.65E+02**
	20	5.13E+03	5.27E+03	**3.27E+03**	2.56E+03	2.12E+03	**1.22E+03**
	30	1.10E+04	1.08E+04	**6.24E+03**	6.77E+03	5.74E+03	**2.70E+03**
F5	10	1.91E+00	1.90E+00	**1.48E+00**	1.18E+00	1.12E+00	**1.07E+00**
	20	6.03E+00	6.10E+00	**3.75E+00**	3.30E+00	2.94E+00	**1.84E+00**
	30	1.12E+01	1.17E+01	**7.15E+00**	6.75E+00	6.56E+00	**3.57E+00**
F6	10	1.05E+01	1.01E+01	**8.91E+00**	6.99E+00	**5.69E+00**	5.70E+00
	20	2.66E+01	2.66E+01	**2.32E+01**	2.23E+01	2.01E+01	**1.85E+01**
	30	4.34E+01	4.48E+01	**3.87E+01**	3.87E+01	3.64E+01	**3.22E+01**
F7	10	2.27E+03	**2.12E+03**	2.39E+03	2.07E+03	**1.71E+03**	1.97E+03
	20	5.55E+03	**5.40E+03**	5.71E+03	5.23E+03	**4.82E+03**	5.34E+03
	30	9.12E+03	**8.96E+03**	9.33E+03	8.81E+03	**8.17E+03**	8.95E+03
F8	10	5.14E+03	6.39E+03	**3.70E+03**	**1.73E+03**	2.96E+03	1.80E+03
	20	6.56E+04	7.10E+04	**5.94E+04**	3.76E+04	4.58E+04	**3.19E+04**
	30	3.86E+05	3.94E+05	**2.58E+05**	2.20E+05	2.57E+05	**1.76E+05**

jDE や DE に比べ性能が劣る問題が未だ存在するが、アルゴリズム調整の自由度を高める結果、複雑かつ探索空間が増大した場合においても、提案手法によって最適化性能を高めるようにアルゴリズム調整が成功している点に、本研究の意義がある。

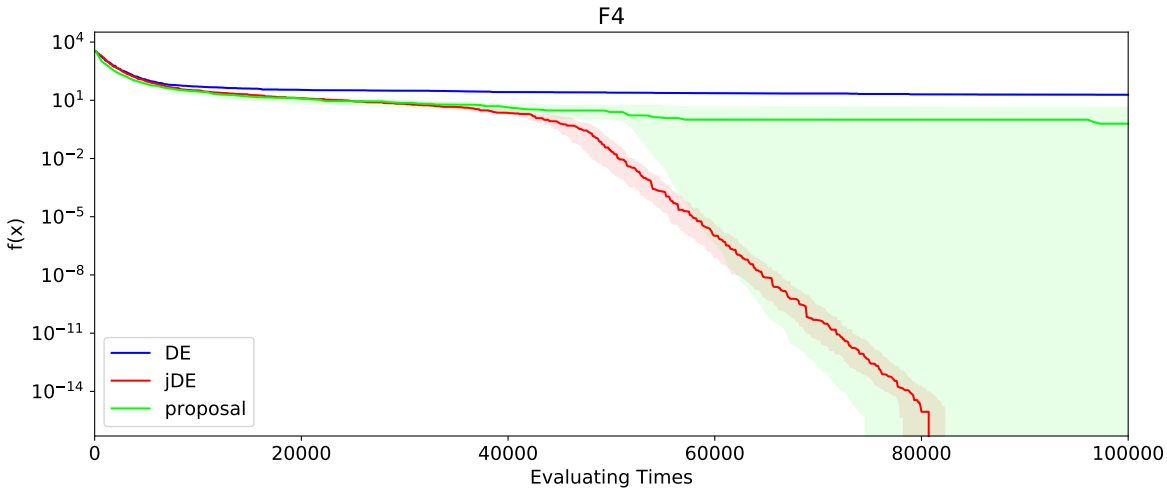


図 5.1　F4($D = 10$) における性能の比較

表 5.3 Comparison of the best fitness discovered at 1,200-th, 3,000-th, 30,000-th and 100,000-th fitness evaluations, respectively. The median value of all the 30 trials are reported.

		30,000-th			100,000-th		
id	D	DE	jDE	ours	DE	jDE	ours
F1	10	1.05E+00	6.91E-01	**4.52E-04**	1.02E-36	8.66E-39	**1.41E-84**
	20	5.50E+02	2.20E+02	**2.85E+00**	3.53E-14	4.14E-18	**4.24E-40**
	30	3.31E+03	1.95E+03	**1.75E+02**	6.17E-08	5.75E-11	**2.92E-19**
F2	10	1.76E+02	3.19E+02	**9.24E+00**	1.14E-11	6.25E-03	**9.51E-18**
	20	3.52E+05	1.62E+05	**8.48E+02**	7.58E+00	1.15E+01	**3.15E+00**
	30	1.09E+07	6.10E+06	**5.00E+04**	2.33E+01	2.42E+01	**2.09E+01**
F3	10	1.87E+01	1.99E+01	**3.08E-01**	4.49E-14	1.11E-14	**4.00E-15**
	20	**1.99E+01**	2.00E+01	2.00E+01	**1.96E+01**	1.99E+01	2.00E+01
	30	**2.00E+01**	**2.00E+01**	**2.00E+01**	**1.99E+01**	2.00E+01	2.00E+01
F4	10	5.13E+01	3.24E+01	**2.73E+01**	1.93E+01	**0.00E+00**	6.20E-01
	20	3.11E+02	2.28E+02	**1.47E+02**	9.68E+01	1.76E+01	**1.64E+01**
	30	1.09E+03	7.63E+02	**3.37E+02**	1.86E+02	6.14E+01	**4.57E+01**
F5	10	5.71E-01	4.07E-01	**3.99E-01**	2.38E-01	**1.41E-13**	2.83E-02
	20	1.14E+00	1.06E+00	**3.49E-01**	2.46E-14	**0.00E+00**	1.97E-02
	30	1.78E+00	1.49E+00	**1.04E+00**	5.01E-09	**4.36E-12**	7.40E-03
F6	10	1.54E+00	5.65E-01	**1.51E-01**	0.00E+00	0.00E+00	0.00E+00
	20	1.27E+01	6.76E+00	**4.20E+00**	1.27E-03	**3.27E-10**	2.52E-01
	30	2.66E+01	1.85E+01	**1.19E+01**	6.41E-02	**4.40E-04**	9.08E-01
F7	10	1.72E+03	**9.83E+02**	1.19E+03	1.17E-01	**1.27E-04**	**1.27E-04**
	20	4.91E+03	**3.71E+03**	4.26E+03	3.66E+03	**2.15E-02**	4.83E+02
	30	8.21E+03	**6.82E+03**	7.58E+03	6.86E+03	**2.85E+03**	3.06E+03
F8	10	**4.72E+01**	8.90E+02	4.74E+01	2.97E-19	1.92E-04	**1.14E-19**
	20	1.36E+04	2.84E+04	**6.24E+03**	**8.95E+00**	6.01E+02	2.00E+01
	30	9.38E+04	1.46E+05	**4.35E+04**	2.68E+03	7.51E+03	**1.86E+03**

第 6 章

考察

本章では、提案手法における競合的なアルゴリズム調整の有用性を分析するとともに、各関数が持つ特徴と照らし合わせた考察をする。また、提案手法で用いたパラメータである検証期間 I と超球の半径 d に対する提案手法の性能変化を分析する。

6.1 各関数に対する詳細

次に、関数に応じた考察をする。F1 は最も単純な単峰性関数であり、対称性だけでなく次元数に対するスケール不変性を持つ。そのため、各グラフはほぼ線形を描き手法同士の逆転はなく、次元数が増加しても提案手法が他の 2 手法に終始優位であった。F2 は変数間相互依存性のため変数分離不可能である。さらに $\boldsymbol{x} = \mathbf{0}$ に集まった後に脱出して長い谷の途中にある $\boldsymbol{x} = \mathbf{1}$ での最適解を求めなくてはならないという特徴を持つ。実際に $D = 10, 20$ では提案手法が $\boldsymbol{x} = \mathbf{0}$ に素早く到達し、そこからの脱却にも成功している。$D = 30$ においては評価回数不足のため判定はできないが、前半の $\boldsymbol{x} = \mathbf{0}$ への到達に関しては同じことが言える。つまり、収束と脱出の両点においてアルゴリズム調整が成功したと言える。F3 は多峰性関数であり、探索領域中心に急勾配の深淵を持つ。そのため一度そこを発見し個体が集中すれば $D = 10, 20$ での提案手法のように評価値は激減する。F4 は格子状に局所解が約 10^D もの多数存在する強い多峰性を持る。また、決定変数ごとに評価値への影響が大きく差が出る悪スケール性を備えるため、影響が小さい個体の探索を十分に行う必要がある [27]。よって、提案手法ではすべての個体に十分な探索をする機会を与えられたと言える。F5 は変数分離不可能で、巨視的に見ると単峰性であるが微視的に見ると強い多峰性であり、比較的大きなステップで探索をする必要があると考えられる。提案手法ではこれに耐えるアルゴリズム調整ができなかった回数が多かったため、四

分位範囲上位では好成績を収めながら中央値では伸び悩んだ。F6 は同じく強い多峰性があり変数分離不可能である上、比較的規則劇に局所解が存在し、さらに最適解が複数存在する特徴を持つ。加えて、微分不可能点が非常に多い。これも $D = 10$ では脱出機能が働いたものの、F5 と同じく高次元では指数関数的に局所解が増加するため、この関数ではある程度の大域探索能力を保持できなかった提案手法は勝利を譲った。F7 も多峰性であり、探索領域の端に最適解を持つという特徴が存在するため自己組織化が必要となる。いずれも jDE が速い収束速度を発揮しているが、$D = 30$ においては提案手法が終盤で挽回をしている。F8 は単峰性でありながらも変数分離不可能性を有しており、一般に DE はこれを不得意としてしているが [19]、特に $D = 10, 30$ でアルゴリズム調整によりこれを解決した。

6.2 調整結果の例

図**??**にアルゴリズムの調整結果の例を示す。

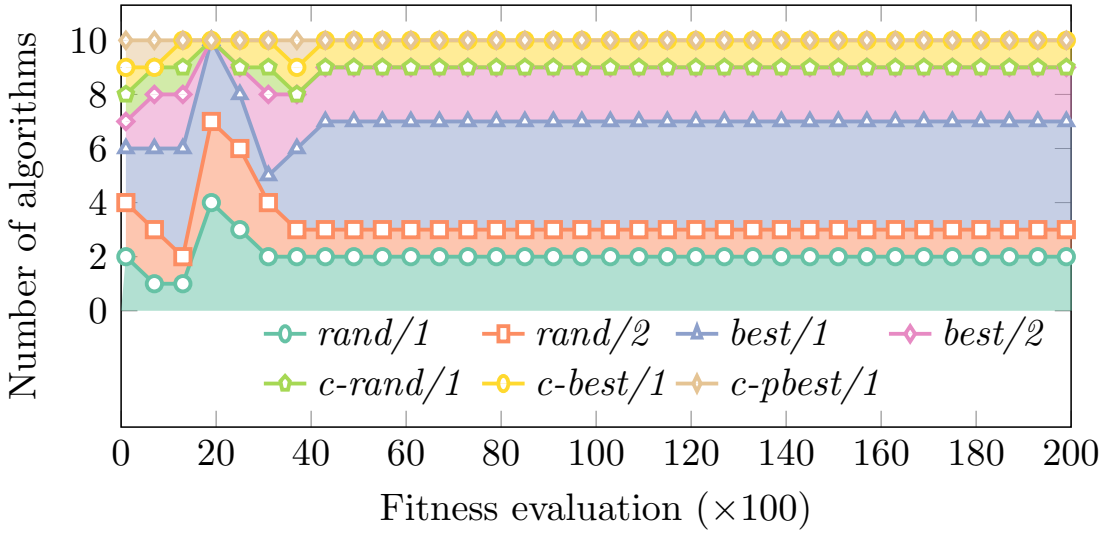
6.3 検証期間 I に対する性能変化

ここでは、調整されたアルゴリズムの検証期間を決定するパラメータ I を 5, 10, 15 と設定し、提案手法の性能変化を明らかにする。図 6.3 に、F4($D = 10$) における各設定に従う提案手法の性能を示す。図より、$I = 10$ では 30 試行中 19 試行で、$I = 15$ では 30 試行中 20 試行でそれぞれ最適解を発見している。一方で、$I = 5$ は、$I = 10, 15$ に比べてその性能が大幅に劣る。前章では、アルゴリズムの調整頻度 (競合頻度) を増やすために $I = 5$ と設定した。しかしながら、図に示すように、アルゴリズムの検証期間は、調整されたアルゴリズムの良し悪しを判断するのに、ある程度の世代数を費やすべきであることがわかる。一方で、検証期間は調整頻度とトレードオフの関係であり、このバランスを適切に設定することが今後の課題となる。

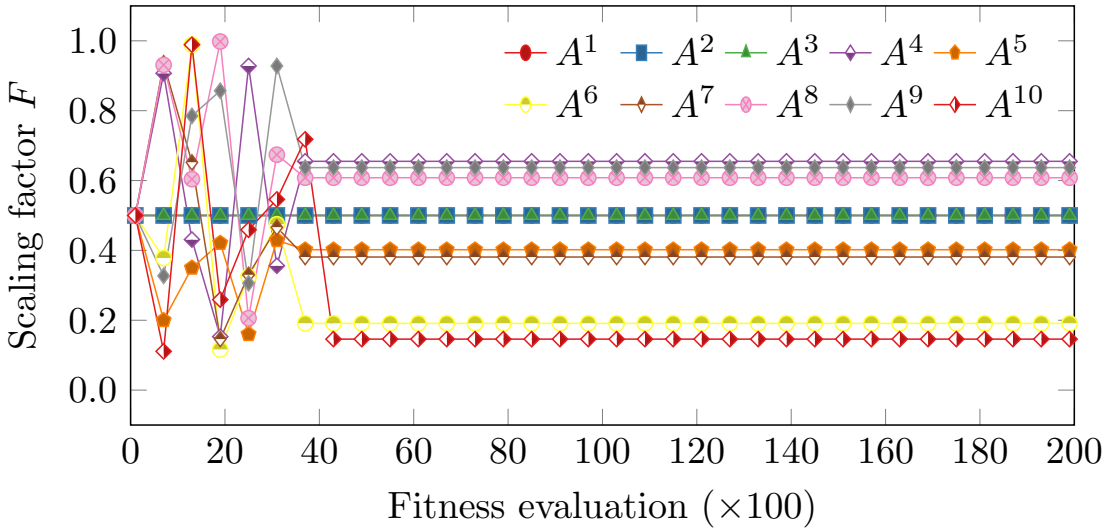
6.4 調整終了判定領域の半径 d に対する性能変化

アルゴリズム調整の終了条件に関するパラメータ d に対する提案手法の性能変化について検証する。 ここで、前章における設定 $d = 0.1 \times (x_{j,\max} - x_{j,\min})$ に対し、$d = 0.05 \times (x_{j,\max} - x_{j,\min})$ と設定したより厳密にアルゴリズム調整を行う場合を検証する。図 6.4 に、F4($D = 10$) における各設定に従う提案手法の性能を示す。図に

示す通り、この問題においては d を小さく保ち、調整目標の近傍に生成するようにアルゴリズム調整を行う場合が、優れた性能を導出することがわかる。言い換えると、$d = 0.05 \times (x_{j,\max} - x_{j,\min})$ と設定した場合は、$d = 0.1 \times (x_{j,\max} - x_{j,\min})$ と比べて調整目標のより近傍を探索するようにアルゴリズム調整がなされる。つまり、提案手法の調整メカニズムでは、調整対象となるアルゴリズムの構成要素とは別に、アルゴリズム調整の仕方によってアルゴリズムの特性に影響を与える可能性がある。したがって、アルゴリズムの調整を通して問題に特化させるという目的を達成するために、パラメータ d を適応的に変化させることで、大域探索および局所探索を明示的に指向可能なアルゴリズム調整が実現できる可能性がある。

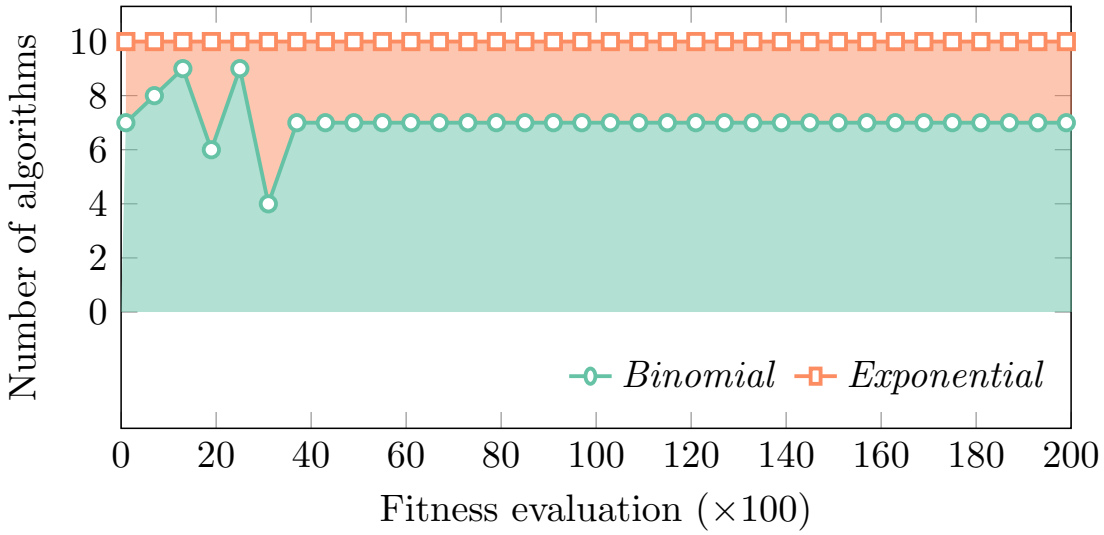


a) Mutation variants x_v

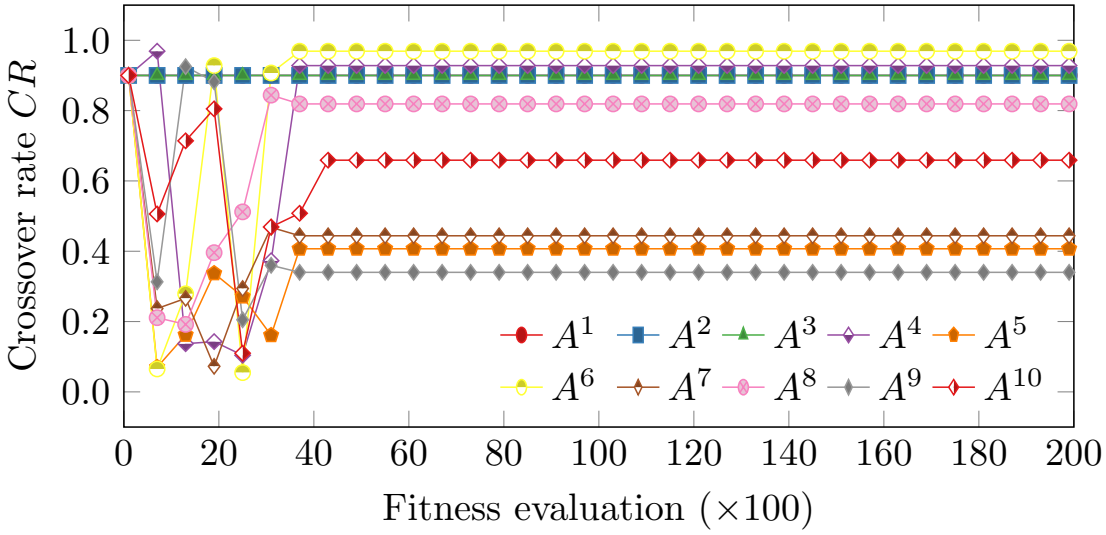


b) Scaling factor x_F

図 6.1　Examples of algorithm-tuning obtained by CAT on the *F1* with $D = 10$. The figures a)-d) report the tuning results of x_F, x_{CR}, x_v and x_u, which are sampled from one trial, respectively. Note that x_v and x_u are reported with stacked curves; each curve indicates the number of algorithms that employ a corresponding mutation (or crossover) variant.

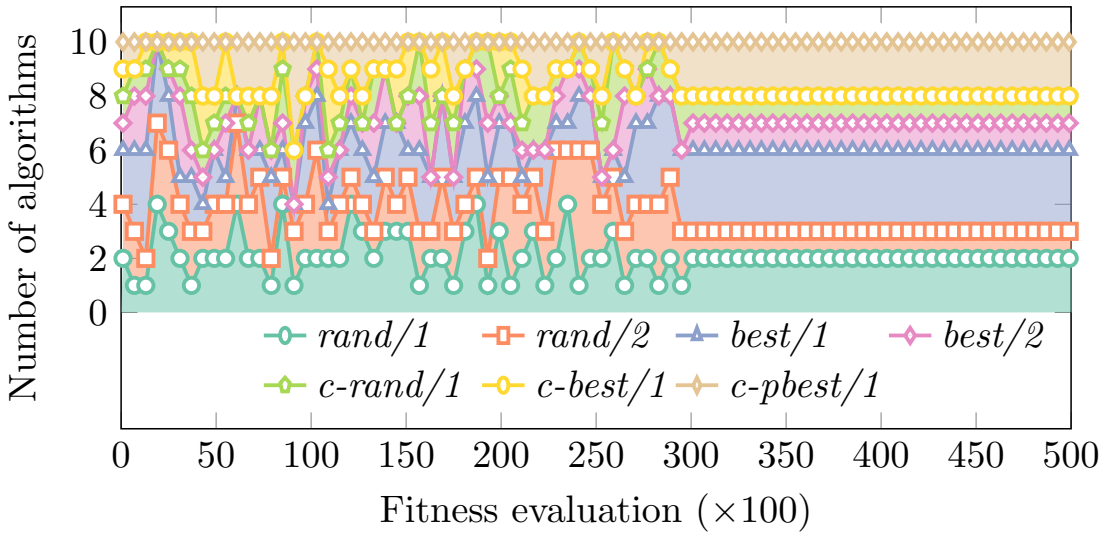


c) Crossover variants x_u

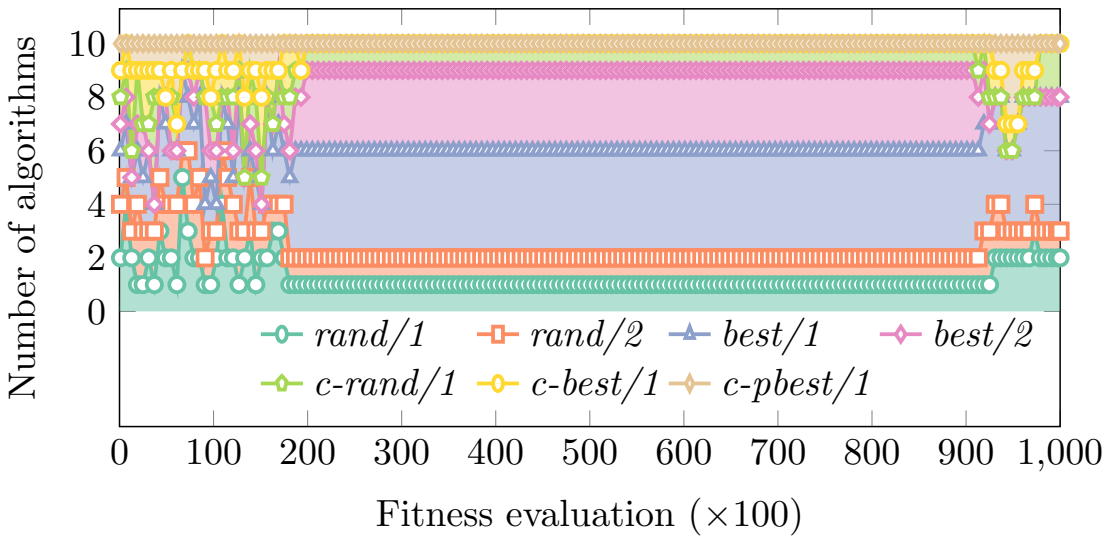


d) Crossover rate x_{CR}

図 6.1 Examples of algorithm-tuning obtained by CAT on the *F1* with $D = 10$. The figures a)-d) report the tuning results of x_F, x_{CR}, x_v and x_u, which are sampled from one trial, respectively. Note that x_v and x_u are reported with stacked curves; each curve indicates the number of algorithms that employ a corresponding mutation (or crossover) variant.



a) *F7* with $D = 10$



b) *F8* with $D = 10$

図 6.2　Examples of algorithm-tuning of mutation variants obtained by CAT on the *F7* and *F8* with $D = 10$.

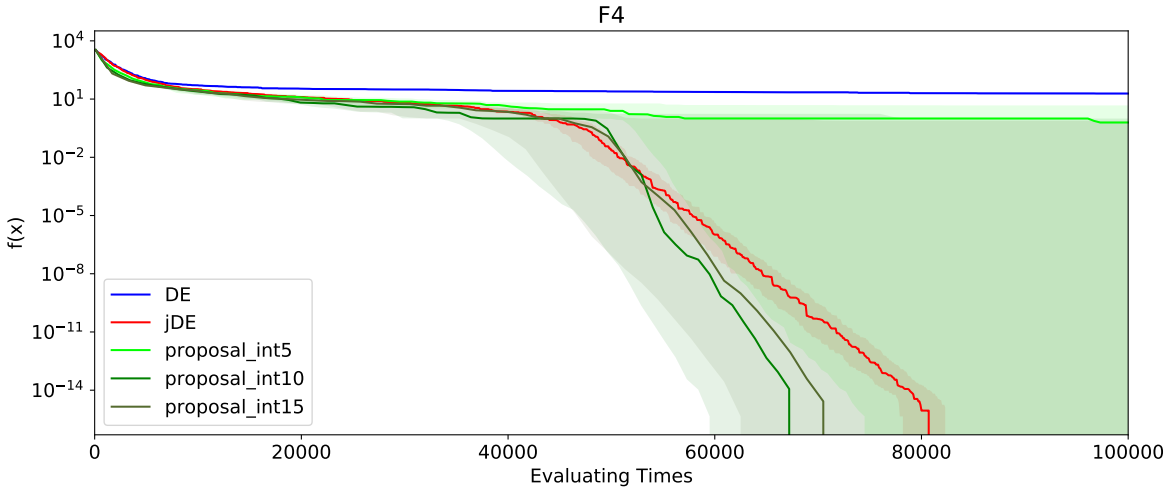


図 6.3　検証期間 I に対する性能の変化

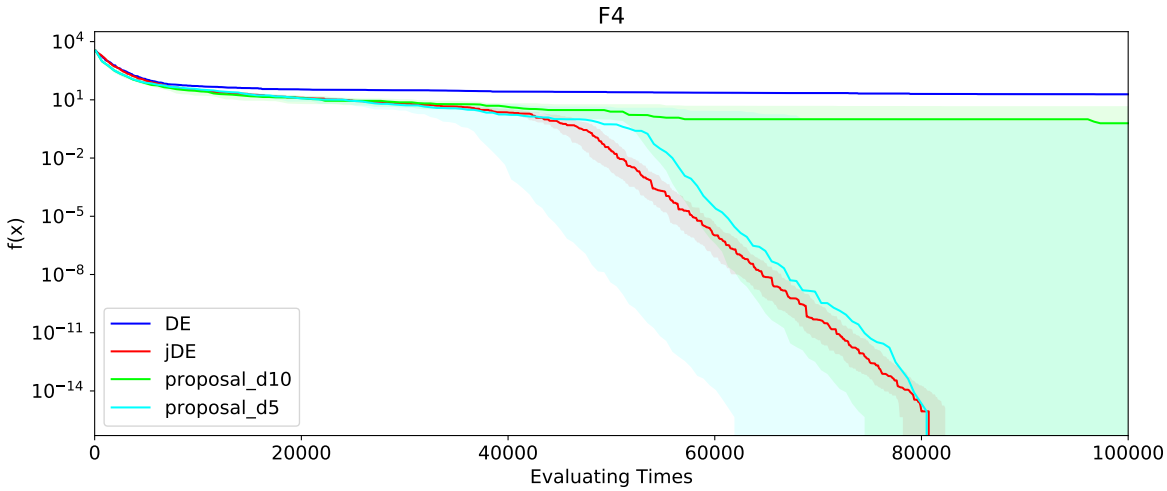


図 6.4　パラメータ d に対する性能の変化

第 7 章

まとめ

7.1　本稿のまとめ

本研究では、競争均衡原理 [22] に着想を得たメタヒューリスティクスの適応的調整技術を提案した。具体的には、差分進化のハイパーパラメータと遺伝的操作を調整対象とし、複数のアルゴリズム (DE) を用意しそれらが生成する個体を定期的に競合させることで、1) アルゴリズム調整に追加の解評価を要さずに、2) 順位が低いアルゴリズムは教師データ (調整の目標となる上位個体) をもとに調整できるフレームワークを構築した。実験結果より、多くの実験ケースにおいて、提案手法は DE やパラメータのみを調整する jDE よりも優れた性能を導出することを明らかにした。これは、ハイパーパラメータと遺伝的操作という高い自由度でアルゴリズム調整を行うことで、問題や探索状況に特化した適切なアルゴリズム調整がなされた結果といえる。これは、遺伝的操作で用いるハイパーパラメータの組合せによって挙動が変化することで調整の複雑さが格段に増す困難さに対し、これに対処可能な提案手法の有用性を示すものである。

7.2　課題

今後の展望としては、調整 (競合) 頻度とアルゴリズムのトレードオフを制御するパラメータ I、および、調整するアルゴリズムの特性を制御できる可能性があるパラメータ d について、実験的な検証をさらに進める。そして、これらのパラメータの適応的な調整法を導入することで、アルゴリズムの調整精度を改善し、本稿で明らかにした課題である提案法の性能安定化を目指す。

謝辞

本論文をまとめるにあたり，終始多大なるご指導とご教示をいただいた主任指導教員の中田雅也准教授に心より感謝の意を表します. また，本論文の研究を進める上で適切な助言を頂いた濱上知樹教授, ならびに研究室の皆様にもこの場を借りて感謝致します.

参考文献

[1] Anna Bogdanova, Jair Pereira Junior, and Claus Aranha. Franken-swarm: grammatical evolution for the automatic generation of swarm-like meta-heuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 411–412. ACM, 2019.

[2] Janez Brest, Sao Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, 10(6):646–657, 2006.

[3] Bingshui Da, Yew-Soon Ong, Liang Feng, A Kai Qin, Abhishek Gupta, Zexuan Zhu, Chuan-Kang Ting, Ke Tang, and Xin Yao. Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results. *arXiv preprint arXiv:1706.03470*, 2017.

[4] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2010.

[5] Agoston E Eiben and Selmar K Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[6] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, pages 1689–1696. ACM, 2010.

[7] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. 2009.

[8] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[9] John H Holland. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. 1975.

[10] Sk Minhazul Islam, Swagatam Das, Saurav Ghosh, Subhrajit Roy, and Ponnuthurai Nagaratnam Suganthan. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):482–500, 2011.

[11] Giorgos Karafotias, Mark Hoogendoorn, and Ágoston E Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2014.

[12] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer, 1995.

[13] Rammohan Mallipeddi, Ponnuthurai N Suganthan, Quan-Ke Pan, and Mehmet Fatih Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied soft computing*, 11(2):1679–1696, 2011.

[14] Péricles BC Miranda and Ricardo BC Prudêncio. A novel context-free grammar for the generation of pso algorithms. *Natural Computing*, pages 1–19, 2018.

[15] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[16] Michael O'Neill and Conor Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, 2001.

[17] Mudita Sharma, Alexandros Komninos, Manuel Lopez Ibanez, and Dimitar Kazakov. Deep reinforcement learning based parameter control in differential evolution. *arXiv preprint arXiv:1905.08006*, 2019.

[18] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[19] Andrew M Sutton, Monte Lunacek, and L Darrell Whitley. Differential evolution and non-separability: using selective pressure to focus search. In *Proceedings of*

the 9th annual conference on Genetic and evolutionary computation, pages 1428–1435. ACM, 2007.

[20] Ryoji Tanabe and Alex Fukunaga. Success-history based parameter adaptation for differential evolution. In *2013 IEEE congress on evolutionary computation*, pages 71–78. IEEE, 2013.

[21] K Tang, X Yao, PN Suganthan, C MacNish, YP Chen, CM Chen, and Z Yang. Benchmark functions for the cec 2010 special session and competition on large scale global optimization. university of science and technology of china (ustc), school of computer science and technology, nature inspired computation and applications laboratory (nical), hefei, anhui. *China. Tech. Rep, Tech. Rep.*, 2010.

[22] Leon Walras. *Elements of pure economics*. Routledge, 2013.

[23] David H Wolpert, William G Macready, et al. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[24] Gang Xu. An adaptive parameter tuning of particle swarm optimization algorithm. *Applied Mathematics and Computation*, 219(9):4560–4569, 2013.

[25] Jingqiao Zhang and Arthur C Sanderson. Jade: adaptive differential evolution with optional external archive. *IEEE Transactions on evolutionary computation*, 13(5):945–958, 2009.

[26] Shi-Zheng Zhao and Ponnuthurai Nagaratnam Suganthan. Empirical investigations into the exponential crossover of differential evolutions. *Swarm and Evolutionary Computation*, 9:27–36, 2013.

[27] 田邊遼司. **関数最適化問題に対する適応型差分進化法の研究**. PhD thesis, University of Tokyo(東京大学), 2016.