情報処理学会論文誌　数理モデル化と応用　Vol.14　No.3　51–67 (Aug. 2021)

自己適応型差分進化法におけるアルゴリズム構成の事前検証フレームワークによる性能の向上

西原 慧[1,a)]　中田 雅也[1]

受付日 2020年11月18日，再受付日 2021年1月8日，
採録日 2021年2月10日

概要：自己適応型差分進化法は，アルゴリズム構成を試行錯誤的に調整するため，少ない解評価回数では性能が十分に改善しない．本論文は，調整されたアルゴリズム構成の事前検証によって，試行錯誤的な調整を削減し，少ない解評価回数で高い性能を実現することを目的とする．また，提案する事前検証フレームワークは高い手法的汎用性があり，スケール係数，交叉率，突然変異・交叉戦略を個体ごとに調整する自己適応型差分進化法に適用できる．ベンチマーク問題を用いた実験では，代表手法である jDE と SaDE，JADE にそれぞれ提案手法を適用した結果，通常よりも少ない数千オーダの解評価回数において，その性能が改善することを示す．これは，自己適応型差分進化法が不得意とする高計算コストな問題において，提案手法がこれに展開できる汎用的な方法論となりうることを示すものである．

キーワード：自己適応型差分進化法，事前検証，高計算コストな最適化問題

Performance Improvement with Prior-validation Framework for Algorithmic Configuration on Self-adaptive Differential Evolution

Kei Nishihara[1,a)]　Masaya Nakata[1]

Received: November 18, 2020, Revised: January 8, 2021,
Accepted: February 10, 2021

Abstract: Self-adaptive differential evolution approaches (self-adaptive DEs) often suffer to boost their performances under a limited number of fitness evaluations, since they heavily rely on the trial-and-error process required to adapt algorithmic configurations. In order to enhance the performance in early generations, this paper presents a generalized prior-validation framework for algorithmic configurations, which can be applicable to major variants of self-adaptive DEs that adapt the scaling factor, the crossover rate, and/or the mutation/crossover strategies for each individual. Experimental results on benchmark problems show that the proposed method successfully boosts the performances of jDE, SaDE, and JADE. Thus, the proposed method reveals a possibility of self-adaptive DEs toward computationally-expensive optimization problems where self-adaptive DEs have had a difficulty.

Keywords: self-adaptive differential evolution, prior validation, computationally-expensive optimization

1. はじめに

実社会の最適化問題は，1 回の解評価に時間がかかる高計算コストな問題となる場合がある．たとえば，航空機の翼形状設計問題では計算量が大きい流体シミュレーションを用いて解を評価する [1]．このような問題では，最適化アルゴリズムの構成を試行錯誤的に調整する余裕はない．しかしながら，実最適化問題に広く用いられる進化アルゴリズムの性能は，ハイパーパラメータや遺伝的オペレータといったアルゴリズム構成に大きく依存することが知られている [2], [3]．そこで，これらのアルゴリズム構成をオンラインで調整する自己適応型進化アルゴリズムの研究がさかんに行われている [4], [5], [6].

[1] 横浜国立大学
Yokohama National University, Yokohama, Kanagawa 240–8501, Japan
[a)] nishihara-kei-jv@ynu.jp

実数値連続最小化問題を対象とした差分進化法（Differential Evolution: DE）[7] は，自己適応技術を組み合わせることで飛躍的に性能が改善することが知られている [8], [9], [10]．この背景には，DE の性能もまたハイパーパラメータ（スケール係数 F，交叉率 CR）や遺伝的オペレータ（突然変異戦略，交叉戦略）といったアルゴリズム構成に大きく左右されることがあげられる [11], [12]．加えて，問題に適切なアルゴリズム構成を事前に決定することが困難である点や，どのアルゴリズム構成を探索中のどのタイミングで適用すべきかが分からない点もあげられる [2], [13], [14]．近年では，アンサンブル最適化 [15], [16], [17]，マルチモーダル最適化 [18], [19]，機械学習を用いたサロゲート型最適化 [20], [21] などに，自己適応型 DE が利用されている．

初期の研究として，アルゴリズム構成をランダム生成する jDE [22]，それらを確率分布を用いて生成する JADE [23] や SaDE [24], [25] がある．これらの手法は，SHADE [26]，jSO [27] や SaNSDE [28] などの発展的な自己適応型 DE につながる重要な洞察を与えた [9], [29]．なお，本段落で述べた自己適応型 DE を含め，一般的なフレームワークでは 1 つの個体に 1 つのアルゴリズム構成を割り当てる個体ベース調整を用いる．個体ベース調整の自己適応型 DE は，アルゴリズム構成の生成方法に違いがあり，調整したアルゴリズム構成をいかに生成方法へフィードバックするかが議論されてきた．

一方で，生成されたアルゴリズム構成をそのまま使用し，その結果を生成方法にフィードバックする点では共通の方法を採用している．つまり，事後検証ベースの方法をもとに試行錯誤的な調整に強く依存することになる．たとえば，ベンチマーク問題では，数万回から数十万回の解評価回数で性能評価を行うことが多く [30]，必ずしも高計算コストな最適化問題を想定してこなかった．実際，自己適応型 DE における試行錯誤的な調整フレームワークは，その最適化性能を改善するために多くの解評価回数を消費するという方法論的な欠点が指摘されている [20]．したがって，高計算コストな最適化問題の展開に向けて，少ない解評価回数でも有効に働く自己適応型 DE の構築が求められる．

そこで本論文では，「アルゴリズム構成をいかに生成するか」よりも，「生成されたアルゴリズム構成のうちどれを利用するか」という観点で，数百から数千の解評価回数でも性能を改善可能な自己適応型 DE の追加フレームワークを検討する．具体的には，生成されたアルゴリズム構成を使用する前に，好適なものを選択する汎用的な事前検証フレームワークを導入する．提案手法では，1 つの個体に対しアルゴリズム構成を選択候補として複数生成し，事前検証を経てこの中から 1 つ選択し出力する操作を行う．事前検証では，候補となるアルゴリズム構成を用いて子個体を仮生成する．そして，その子個体と（すでに発見した）優良解との類似度を計算し，この指標に基づいて好適なアルゴリズム構成を出力する．最後に，出力されたアルゴリズム構成を用いて子個体を改めて本生成する．これは，1）優良解の周辺領域を局所探索するバイアスを高めたアルゴリズム構成によって収束速度を改善すること，一方で 2）子個体の本生成時に用いる突然変異の非決定性によって個体の多様性の著しい低下を抑制し早期収束を防ぐことを意図する．

上記の想定する効果は，粒子群最適化（Particle Swarm Optimization: PSO）[31] における関連研究と，著者らの先行研究の知見を活かしている．具体的には，サロゲート型 PSO である OUPS は，1 つの粒子に対し複数の速度ベクトルを仮生成し，目的関数を近似したサロゲート（機械学習）を用いて近似評価値が最も高い速度ベクトルを利用する [32]．優良解の領域への局所探索のバイアスを高めた速度ベクトルは，本論文で DE のアルゴリズム構成とみなせる．また，著者らが構築したアンサンブル型の自己適応型 DE は，競合する DE が導出した優良解を模倣するようにアルゴリズム構成を競合的に生成することで，収束速度が改善する [33]．したがって，サロゲートを構築しなくとも，優良解の模倣によって優良解の領域への局所探索のバイアスを高めたアルゴリズム構成を生成できる可能性がある．一方で提案手法は，特定の自己適応型 DE に限定せず，個体ベース調整で F，CR，突然変異・交叉戦略を調整する自己適応型 DE に利用できる利点がある．この手法的汎用性を高める理由は，自己適応型 DE の問題依存性 [15], [34] からユーザが用いる自己適応型 DE を選択する余地を残すためや，ユーザが自動調整を所望するアルゴリズム構成の違いに対応するためである．

様々な自己適応型 DE がある一方で，アルゴリズム構成の生成方法および調整するアルゴリズム構成の違いに着目し，基本的な自己適応型 DE に提案手法を適用する．具体的には，jDE と SaDE，JADE に提案手法を組み込む．jDE は，F と CR をランダムで生成し，好適なアルゴリズム構成はそのまま再利用する．これに比べて，SaDE は，F と CR に加えて突然変異戦略と交叉戦略を調整対象にし，好適なアルゴリズム構成から求めた確率分布を用いて生成する．したがって，jDE と SaDE は調整対象にするアルゴリズム構成に加えて，好適なアルゴリズム構成のフィードバック法も異なる．また，F と CR を調整対象とする JADE は，評価値が上位の個体をもとにアルゴリズム構成を調整する点に特徴があり，SaDE のフィードバック方法と異なる．

実験では提案手法を組み込んだ jDE と SaDE，JADE が組み込む前よりも性能が改善することを示す．したがって，本論文で実証する提案手法の有効性は jDE，SaDE ならびに JADE に限定される．この意味で，本論文は他の自己適応型 DE における提案手法の有効性を主張するものではなく，提案手法の手法的汎用性を利点とすることに留意され

たい．また，提案手法を組み込んだ場合の純粋な性能評価を行うために，少ない解評価回数を見越した jDE と SaDE，JADE の微調整（fine-tuning）は行わず，文献にしたがったフレームワークおよびパラメータ設定を用いる（高計算コストな問題を見据え，人手による調整は考慮しない）．

本論文の構成は次のとおりである．2 章では DE のアルゴリズムを紹介し，3 章では既存の自己適応型 DE をアルゴリズム構成の生成方法の観点でまとめる．4 章では提案手法のメカニズムについて述べる．5 章では，提案手法を jDE と SaDE，JADE に組み込み，ベンチマーク問題として IEEE CEC2013 real-parameter single-objective benchmark function suite [30] を用いて，その性能を評価する．6 章で提案手法の分析を行い，最後に 7 章で結論と今後の展望を述べる．なお，以降では個体ベース調整を用いる自己適応型 DE を単に「自己適応型 DE」と呼ぶことにする．

2. 差分進化法（DE）

本論文では，単一目的実数値連続最小化問題を対象とする．同問題では，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，以下で与えられる D 次元の大域的最適解 $\boldsymbol{x}^* = [x_1, x_2, \ldots, x_D]$，あるいはその近似解を求めることが目的である．

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}). \tag{1}$$

DE の各個体 $\boldsymbol{x}_i$（$i = 1, 2, \ldots, N$）も解ベクトル $\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}]$ として表現される．ここで N は解集合サイズ（個体数）であり，各個体は解集合 $\mathcal{P}$ に属する（$\boldsymbol{x} \in \mathcal{P}$）．以下に述べるように，DE のフレームワークでは，初期化を行った後に，突然変異の適用から個体選択までの一連のプロセスを探索の終了条件を満たすまで繰り返す．

2.1 初期化

世代数 $t = 0$ および $\mathcal{P} = \emptyset$ において，個体数 N の数だけ初期解を生成し，解集合 $\mathcal{P}$ に追加する．具体的には，i 番目の個体 $\boldsymbol{x}_i$（$i = 1, 2, \ldots, N$）に対し，j 次元目の決定変数 $x_{i,j} \in [x^l_{i,j}, x^u_{i,j}]$ を一様分布乱数を用いて，定義域からランダムに設定する（$j = 1, 2, \ldots, D$）．ここで，$x^l_{i,j}$ と $x^u_{i,j}$ はそれぞれ $x_{i,j}$ の定義域の最小値および最大値である．

2.2 突然変異

次に，$t \leftarrow t + 1$ として，t 世代目の i 番目の個体の突然変異個体 $\boldsymbol{v}_i$ を解集合 $\mathcal{P}$ から生成する．**表 1** に示すように，これまでに様々な突然変異戦略が提案されている．たとえば *rand/1* は次式で表される．

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}). \tag{2}$$

表 1　DE における代表的な突然変異戦略
Table 1　Popular mutation strategies for DE.

Method	Definition
rand/1	$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3})$
rand/2	$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}) + F(\boldsymbol{x}_{r_4} - \boldsymbol{x}_{r_5})$
best/1	$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2})$
best/2	$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}) + F(\boldsymbol{x}_{r_3} - \boldsymbol{x}_{r_4})$
current-to-rand/1	$\boldsymbol{v}_i = \boldsymbol{x}_i + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_i) + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3})$
current-to-best/1	$\boldsymbol{v}_i = \boldsymbol{x}_i + F(\boldsymbol{x}_{best} - \boldsymbol{x}_i) + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2})$

Algorithm 1 exponential 交叉 (exponential crossover)

```
j = randint [1, D]
k = 1
u_i ← x_i
repeat
    u_{i,j} = v_{i,j}
    j = (j + 1) mod D
    k = k + 1
until rand [0, 1) ≥ CR or k ≥ D
```

ここで，スケール係数 $F \in [0, 1]$ は差分ベクトルの大きさを調整するハイパーパラメータであり，そ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役割がある．上式ならびに表 1 において，$\boldsymbol{x}_{r_1}, \ldots, \boldsymbol{x}_{r_5}$ は現在の解集合 $\mathcal{P}$ より自身 $\boldsymbol{x}_i$ 以外でランダムに選択された個体であり，$\boldsymbol{x}_{best}$ は $\mathcal{P}$ 内の最良個体を意味する．

2.3 交叉

生成した突然変異個体 $\boldsymbol{v}_i$ と親個体 $\boldsymbol{x}_i$ を交叉させ，交叉後の子個体 $\boldsymbol{u}_i$ を生成する．交叉戦略も複数提案されており，代表的な方法として binomial 交叉と exponential 交叉がある．たとえば，binomial 交叉は式 (3) で与えられる．

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } rand[0,1] \le CR \text{ or } j = j_{rand}, \\ x_{i,j} & \text{otherwise.} \end{cases} \tag{3}$$

ここで，交叉率 $CR \in [0, 1]$ は，突然変異個体の決定変数の寄与度合を制御する役割がある．なお，まったく交叉されない場合を避けるために $[1, D]$ から整数乱数 j_{rand} を決め，j_{rand} 次元目は必ず交叉させる．$rand[0, 1]$ は，$[0, 1]$ の範囲から実数の一様分布乱数を返す擬似関数を表す．また，exponential 交叉は Algorithm 1 で与えられる．$randint[1, D]$ は，$[1, D]$ の範囲から整数の一様分布乱数を返す擬似関数を表す．

2.4 個体選択

式 (4) に示すとおり，$\boldsymbol{u}_i$ の評価値が $\boldsymbol{x}_i$ の評価値よりも小さい場合にのみ $\boldsymbol{x}_i$ を $\boldsymbol{u}_i$ に設定する．そして，突然変異

(2.2 節) に戻り一連のプロセスを探索終了条件まで繰り返す．

$$\boldsymbol{x}_i \leftarrow \begin{cases} \boldsymbol{u}_i & \text{if } f(\boldsymbol{u}_i) \le f(\boldsymbol{x}_i), \\ \boldsymbol{x}_i & \text{otherwise.} \end{cases} \tag{4}$$

3. 自己適応型 DE の分類

本章では，まず自己適応型 DE を一般化した擬似アルゴリズムを定義する．この理由は，次章で導入する提案手法のフレームワークを特定の自己適応型 DE に限定せず議論するためである．その後，アルゴリズム構成の生成方法の観点から自己適応型 DE を 2 種類に大別する．また，提案手法を組み込む jDE と SaDE，JADE のメカニズムについても一般化した擬似アルゴリズムに従いながら説明する．なお，jDE や SaDE，JADE のアルゴリズム表記は原著と異なるが，その内容自体は変わらないことに留意されたい．また，DERSF，DETVSF [35] や，sinDE [36] などのスケジューリングを用いた調整は，自己適応的な調整とは異なるため本論文の対象としない．また，DE のハイパーパラメータには F，CR に加えて解集合サイズ N が存在するが，解生成プロセスに直接寄与する F と CR に着目する．

3.1 自己適応型 DE の一般化アルゴリズム

各個体 $\boldsymbol{x}_i$ に割り当てるアルゴリズム構成を $\boldsymbol{\theta}_i = [\theta_{v,i}, \theta_{u,i}, \theta_{F,i}, \theta_{CR,i}]$ と表記する（$i = 1, 2, \ldots, N$）．また，Θ を $\boldsymbol{\theta}$ の集合とすると，$\boldsymbol{\theta} \in \Theta$ であり $|\Theta| = |\mathcal{P}| = N$ である．ここで，$\boldsymbol{\theta}_i$ の各要素 $\theta_{v,i}, \theta_{u,i}, \theta_{F,i}, \theta_{CR,i}$ をそれぞれ以下のように定義する．

θ_v　使用する突然変異戦略の種類に対応させたインデックスを表すカテゴリカル型変数．たとえば，$\theta_v = 1, 2, 3$ がそれぞれ *rand/1*, *best/2*, *current-to-rand/1* に対応すると定義できる．突然変異戦略を自己適応しない場合は，用いる突然変異戦略を永続的に使うという意味で θ_v を 1 に固定する．

θ_u　使用する交叉戦略の種類に対応させたインデックスを表すカテゴリカル型変数．たとえば，$\theta_u = 1, 2$ がそれぞれ binomial 交叉と exponential 交叉に対応すると定義できる．交叉戦略を自己適応しない場合は，用いる交叉戦略を永続的に使うという意味で θ_u を 1 に固定する．

θ_F　スケール係数 F を表す $\theta_F \in [0, 1]$ の実数型変数．スケール係数を自己適応しない場合は，θ_F を規定値に固定する．

θ_{CR}　交叉率 CR を表す $\theta_{CR} \in [0, 1]$ の実数型変数．交叉率を自己適応しない場合は，θ_{CR} を規定値に固定する．

次に，自己適応型 DE のアルゴリズムを一般化した擬似コードを Algorithm 2 に示す．最初に解集合 $\mathcal{P} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ とアルゴリズム構成 $\Theta = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ を初期化する．次にメインループとして，個体 $\boldsymbol{x}_i$ ごとに擬似関数 $Sample(\boldsymbol{\theta}_i^{t-1})$ でアルゴリズム構成 $\boldsymbol{\theta}_i$ を生成（サンプリング）する．具体的には，現在割り当てている $\boldsymbol{\theta}_i$ を $\boldsymbol{\theta}_i^{t-1} = [\theta_{v,i}^{t-1}, \theta_{u,i}^{t-1}, \theta_{F,i}^{t-1}, \theta_{CR,i}^{t-1}]$ として記憶し，必要であれば $\boldsymbol{\theta}_i^{t-1}$ を考慮して $\boldsymbol{\theta}_i$ を生成する．なお，$Sample(\boldsymbol{\theta}_i^{t-1})$ の定義は，自己適応 DE ごとのアルゴリズム構成の生成方法に依存するため擬似関数として表記している．生成された $\boldsymbol{\theta}_i$ を用いて，突然変異個体 $\boldsymbol{v}_i$，次いで交叉個体 $\boldsymbol{u}_i$ を生成し，解評価および個体選択を行う．そして，擬似関数 $Update\ (\boldsymbol{\theta}_i, \boldsymbol{\theta}_i^{t-1})$ を用いて，$\boldsymbol{\theta}_i$ をこのままに設定しておくか，$\boldsymbol{\theta}_i^{t-1}$ に戻すかを決定する．この操作は，自己適応型 DE ごとに異なる．なお，$Sample(\boldsymbol{\theta}_i^{t-1})$ と $Update(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i^{t-1})$ で表記した引数の種類は，自己適応型 DE によって変わるが，最も単純な表記で記載している．

Algorithm 2 一般化した自己適応型 DE
(generalized self-adaptive DE)

$t = 0$
Initialize $\mathcal{P} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$
Initialize $\Theta = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$
while *termination criteria* are not met **do**
  $t = t + 1$
  for $i = 1$ **to** N **do**
    $\boldsymbol{\theta}_i^{t-1} \leftarrow \boldsymbol{\theta}_i$
    $\boldsymbol{\theta}_i = [\theta_{v,i}, \theta_{u,i}, \theta_{F,i}, \theta_{CR,i}] \leftarrow Sample(\boldsymbol{\theta}_i^{t-1})$
    $\boldsymbol{v}_i \leftarrow Mutation(\mathcal{P}, \theta_{v,i}, \theta_{F,i})$
    $\boldsymbol{u}_i \leftarrow Crossover(\boldsymbol{x}_i, \boldsymbol{v}_i, \theta_{u,i}, \theta_{CR,i})$
  for $i = 1$ **to** N **do**
    $\boldsymbol{x}_i \leftarrow \begin{cases} \boldsymbol{u}_i & \text{if } f(\boldsymbol{u}_i) \le f(\boldsymbol{x}_i) \\ \boldsymbol{x}_i & \text{otherwise} \end{cases}$
    $\boldsymbol{\theta}_i \leftarrow Update\ (\boldsymbol{\theta}_i, \boldsymbol{\theta}_i^{t-1})$

3.2 ランダム生成による再利用モデル

このモデルでは，調整するアルゴリズム構成 $\boldsymbol{\theta}_i$ をランダムに選択するため，試行錯誤的な要素が強い方法となる．また，解評価によってその個体を生成したアルゴリズム構成 $\boldsymbol{\theta}_i$ が好適であると判断された場合には，$\boldsymbol{\theta}_i$ を次世代で再利用する形で継承する点に特徴がある．

jDE [22] は解更新に成功（$f(\boldsymbol{u}_i) \le f(\boldsymbol{x}_i)$）したときにその $\theta_{F,i}$ と $\theta_{CR,i}$ を次世代に継承する調整方法を用いる．これは，GA や PSO の適応モデルを流用する傾向が強かった最初期の研究と比較し [38], [39], [40]，DE に特化した方法として位置づけられる [34]．jDE における $Sample(\boldsymbol{\theta}_i^{t-1})$ を $Sample\text{-}jDE(\boldsymbol{\theta}_i^{t-1})$ として Algorithm 3 に示す．

全体のアルゴリズムとしては，最初にすべての個体 $\boldsymbol{x}_i \in \mathcal{P}$ を一様分布で初期化する．また，個体ごとのアルゴリズム構成 $\boldsymbol{\theta}_i$ のうち，$\theta_{F,i}, \theta_{CR,i}$ の値を $\theta_{F,i} = 0.5$，$\theta_{CR,i} = 0.9$ で初期化する．なお，jDE は，突然変異戦略 $\theta_{v,i}$ は *rand/1*，

Algorithm 3 *Sample-jDE*($\boldsymbol{\theta}_i^{t-1}$)

$$\theta_{F,i} = \begin{cases} rand[0.1, 1] & \text{if } rand[0,1] \leq \tau_F \\ \theta_{F,i}^{t-1} & \text{otherwise} \end{cases}$$

$$\theta_{CR,i} = \begin{cases} rand[0,1] & \text{if } rand[0,1] \leq \tau_{CR} \\ \theta_{CR,i}^{t-1} & \text{otherwise} \end{cases}$$

$\theta_{v,i} = 1$ (*rand/1*)

$\theta_{u,i} = 1$ (*binomial crossover*)

return $\boldsymbol{\theta}_i$

交叉戦略 $\theta_{u,i}$ は binomial 交叉を用い調整対象にしない．次に，Algorithm 3 に示したように，jDE のハイパーパラメータ τ_F, τ_{CR} の確率で新しい $\theta_{F,i}, \theta_{CR,i}$ の値を一様分布でサンプリングし，それ以外では前世代で用いた値を引き継ぐ．そして，これらを用いて一連の世代更新を行い，*Update* ($\boldsymbol{\theta}_i, \boldsymbol{\theta}_i^{t-1}$) に相当する操作として，解更新に失敗した場合 ($f(\boldsymbol{u}_i^t) > f(\boldsymbol{x}_i^t)$) に，$\boldsymbol{\theta}_i$ を $\boldsymbol{\theta}_i^{t-1}$ に戻す ($\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i^{t-1}$)．このように，jDE では一定確率 τ_F, τ_{CR} で新しいハイパーパラメータを試しながら，問題や探索状況により好適なハイパーパラメータを継承する．この τ_F, τ_{CR} を動的に変化させた改良手法として FDSADE [41] や ISADE [42] が存在する．

他の例として，EPSDE [43] は，$\theta_{F,i}$ と $\theta_{CR,i}$ に加えて突然変異戦略 $\theta_{v,i}$ も適応的に調整する．それぞれの選択肢を格納したプールを $P_F = \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$，$P_{CR} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$，$P_v = \{$*rand/1*, *best/2*, *current-to-rand/1*$\}$ として事前に定義する．初期化プロセスにおいては，それぞれのプールから $\theta_{F,i}$，$\theta_{CR,i}$，$\theta_{v,i}$ を 1 つずつ一様分布で選択し，個体 $\boldsymbol{x}_i$ に割り当てる．その後の世代更新では前世代で解更新に成功した場合はそれらを継承し，失敗した場合にのみ $\theta_{F,i}$，$\theta_{CR,i}$，$\theta_{v,i}$ のすべてを一様分布でランダムに割り当てる．なお，交叉戦略 $\theta_{u,i}$ は binomial 交叉である．比較的単純なアルゴリズムでありながら，優れた最適化性能を導出することが報告されている [34]．

3.3　確率分布に基づく逐次更新モデル

このモデルでは，正規分布やコーシー分布といった，一様分布以外の確率分布からサンプリングすることで，アルゴリズム構成 $\boldsymbol{\theta}_i$ を適応的に調整する．この方法の特徴は，たとえば過去に解更新に成功した情報から確率分布のメタパラメータを調整するなど，好適なアルゴリズム構成を獲得するために世代を追って逐次的に確率分布を調整する点にある．過去のアルゴリズム構成を再利用する 3.2 節のモデルとは異なり，毎世代調整される確率分布から逐次的にアルゴリズム構成をサンプリングする点も特徴である．

JADE [23] では過去に解更新に成功した情報から有用と予想される範囲のハイパーパラメータを集中的にサンプリングする．まず，$\theta_{F,i}$ と $\theta_{CR,i}$ の値を生成するコーシー分布 $\mathcal{C}$ と正規分布 $\mathcal{N}$ の位置パラメータと平均値として，μ_F と μ_{CR} のメタパラメータを定義する．これらの値は毎回の世代更新終了時に解更新に成功した $\theta_{F,i}$ と $\theta_{CR,i}$ の値に基づいて式 (5)，(6) のように変更される．

$$\mu_F = (1-c)\mu_F + c \cdot mean_L(S_F), \tag{5}$$

$$\mu_{CR} = (1-c)\mu_{CR} + c \cdot mean(S_{CR}). \tag{6}$$

ここで，c は学習率であり，S_F と S_{CR} はその世代で解更新に成功した $\theta_{F,i}$ と $\theta_{CR,i}$ を格納する集合をそれぞれ示す．また，$mean(S_{CR})$ は S_{CR} 内のすべての値の平均値であり，$mean_L(S_F)$ は S_F 内のすべての値の 2 階のレーマー平均 $\left(\sum_{\theta_{F,i} \in S_F} \theta_{F,i}^2 / \sum_{\theta_{F,i} \in S_F} \theta_{F,i}\right)$ である．Algorithm 4 に示すとおり，各個体における $\theta_{F,i}$ と $\theta_{CR,i}$ のサンプリングは，それぞれ逐次的に調整された確率分布 $\mathcal{C}(\mu_F, 0.1)$，$\mathcal{N}(\mu_{CR}, 0.1)$ より行われる．また，交叉戦略 $\theta_{u,i}$ は binomial 交叉であるが，JADE では新たな突然変異戦略 $\theta_{v,i}$ として *current-to-pbest/1* が用いられている．

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F(\boldsymbol{x}_{pbest} - \boldsymbol{x}_i) + F(\boldsymbol{x}_{r_1} - \tilde{\boldsymbol{x}}_{r_2}). \tag{7}$$

ここで，$\boldsymbol{x}_{pbest}$ は $\mathcal{P}$ 内で評価値が $p_i \times N$ 位 ($p_i \in [p_{\min}, p_{\max}]$) の個体からランダムに選ばれた個体であり，$\tilde{\boldsymbol{x}}_{r_2}$ は，あらかじめ用意したアーカイブ $\mathcal{A}$ からランダムに選ばれた個体である．$\mathcal{A}$ には解更新に成功した際の親個体 $\boldsymbol{x}_i$ が毎回保存される．なお，$\mathcal{A}$ のサイズが $\mathcal{P}$ のサイズと一致するように，世代更新の最後に $\mathcal{A}$ 内の個体がランダムに削除される．メタパラメータ μ_F，μ_{CR} とアーカイブ $\mathcal{A}$ の調整をまとめた擬似コードを Algorithm 5 に示す．

また，JADE を改良した手法として SHADE [26] がある．SHADE では，解更新に成功した際のハイパーパラメータ $\theta_{F,i}$ と $\theta_{CR,i}$ の 2 階のレーマー平均を世代ごとに計算し，それぞれ H 個のメモリ $M_{F,h}$ と $M_{CR,h}$ にカウンタ $h = 1, 2, \ldots, H$ を巡回させながらその値を格納する．ハイパーパラメータ $\theta_{F,i}$ と $\theta_{CR,i}$ のサンプリングに，それぞれコーシー分布と正規分布を用いる点は JADE と同じである．一方で，$[1, H]$ よりランダムに選ばれた整数 r_i を

Algorithm 4 *Sample-JADE*($\boldsymbol{\theta}_i^{t-1}$)

repeat

$\theta_{F,i} = \mathcal{C}(\mu_F, 0.1)$

$\theta_{F,i} = 1$ if $\theta_{F,i} > 1$

until $0 \leq \theta_{F,i} \leq 1$

$\theta_{CR,i} = \mathcal{N}(\mu_{CR}, 0.1)$

$$\theta_{CR,i} = \begin{cases} 0 & \text{if } \theta_{CR,i} < 0 \\ 1 & \text{if } \theta_{CR,i} > 0 \end{cases}$$

$\theta_{v,i} = 1$ (*current-to-pbest/1*)

$\theta_{u,i} = 1$ (*binomial crossover*)

return $\boldsymbol{\theta}_i$

Algorithm 5 JADE のパラメータ調整 世代更新後
(adaptation of JADE parameters after generation process)

for $i = 1$ **to** N **do**
  if $f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i)$ **then**
    $S_F \leftarrow$ Add $\theta_{F,i}$, $S_{CR} \leftarrow$ Add $\theta_{CR,i}$, $\mathcal{A} \leftarrow$ Add $\boldsymbol{x}_i$
$\mu_F, \mu_{CR} \leftarrow$ eq. (5), (6)
$S_F = \emptyset, S_{CR} = \emptyset$
Randomly remove $\boldsymbol{x} \in \mathcal{A}$ until $|\mathcal{A}| \leq |\mathcal{P}|$

Algorithm 6 *Sample-SaDE*$(\boldsymbol{\theta}_i^{t-1})$

$\theta_{v,i}, \theta_{u,i} \leftarrow$ Sample a pair of strategies from distribution p
$\theta_{F,i} = \mathcal{N}(0.5, 0.3)$
repeat
  $\theta_{CR,i} = \mathcal{N}(CRm_k, 0.1)$
until $0 \leq \theta_{CR,i} \leq 1$
return $\boldsymbol{\theta}_i$

Algorithm 7 SaDE のパラメータ調整 世代更新前
(adaptation of SaDE parameters before generation process)

if $t > LP$ **then**
  for $k = 1$ **to** K **do**
    $p_{k,t} \leftarrow$ eq. (8)
    Remove $ns_{k,t-LP}, nf_{k,t-LP}$
    $CRm_k = median(CR_{Memory_k})$

Algorithm 8 SaDE のパラメータ調整 世代更新後
(adaptation of SaDE parameters after generation process)

for $i = 1$ **to** N **do**
  if $f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i)$ **then**
    $ns_{k,t} += 1$, $CR_{Memory_k} \leftarrow$ Add $\theta_{CR,i}$
  else
    $nf_{k,t} += 1$

用いて，M_{F,r_i} と M_{F,r_i} をそれぞれ位置パラメータと平均値として，これらの確率分布の調整が行われる．これにより，探索が経過するにつれて $M_{F,h}$ と $M_{CR,h}$ は多様性を保持しながら徐々に問題に適していき，ハイパーパラメータをサンプリングする確率分布も対応して調整されていく．改良手法としては，解集合サイズを線形的に減少させることで大域探索から局所探索へ徐々にシフトする L-SHADE [44]，L-SHADE にガウス分布を用いたローカルサーチと sinDE を参考にしたハイパーパラメータ調整を加えた LSHADE-Epsin [45]，L-SHADE において事前にスケジューリングされたルールで探索フェーズに応じた細かい調整を行う iL-SHADE [46] や jSO [27] などが存在する．

DE のハイパーパラメータに加えて突然変異・交叉戦略も調整する代表的な手法に SaDE [24], [25] がある．SaDE における *Sample*$(\boldsymbol{\theta}_i^{t-1})$ を *Sample-SaDE*$(\boldsymbol{\theta}_i^{t-1})$ として Algorithm 6 に示す．$\theta_{F,i}$ についてはつねに平均値 0.5，標準偏差 0.3 の正規分布 $\mathcal{N}(0.5, 0.3)$ からサンプリングされる．$\theta_{CR,i}$ と突然変異・交叉戦略 $\theta_{v,i}, \theta_{u,i}$ は，ハイパーパラメータ LP を用いて過去 LP 世代での解更新の成功履歴を用いて調整される．また，戦略は突然変異・交叉戦略の組合せを定義し，*rand/1/bin*，*rand/2/bin*，*current-to-rand/1*，*rand-to-best/2/bin* の 4 つが用意される．つまり，この組合せを 1 つ選べば，$\theta_{v,i}$ と $\theta_{u,i}$ が同時に設定される．4 つの組合せのインデックスをそれぞれ $k = 1, 2, 3, 4$（したがってインデックス総数 $K = 4$）とし，これらの組合せごとに過去 LP 世代の解更新の成功率から確率分布 p（歪ルーレット）を定義する．具体的には，Algorithm 8 のように世代更新の終わりの解更新時に戦略 k ごとかつ世代 t ごとの解更新の成功数 $ns_{k,t}$ と失敗数 $nf_{k,t}$ を過去 LP 世代に渡って記録する．そして世代更新の始めに Algorithm 7，式 (9) に示すように戦略 k ごとの成功率 $S_{k,t}$ を求め，これに応じた戦略 k の選択確率 p_k を式 (8) で生成する．ここで式 (9) の ϵ は，すべての $S_{k,t}$ の第一項が 0 であるときに式 (8) で 0 での除算を回避するための定数である．

$$p_{k,t} = \frac{S_{k,t}}{\sum_{k=1}^{K} S_{k,t}}, \tag{8}$$

$$S_{k,t} = \frac{\sum_{g=t-LP}^{t-1} ns_{k,g}}{\sum_{g=t-LP}^{t-1} ns_{k,g} + \sum_{g=t-LP}^{t-1} nf_{k,g}} + \epsilon. \tag{9}$$

次に，Algorithm 6 のように，この確率分布 p を用いて，現在の世代で各個体 $\boldsymbol{x}_i$ が用いる組合せのインデックス k_i を決定する．$\theta_{CR,i}$ については，過去 LP 世代において解更新に成功した値が戦略ごとに CR_{Memory_k} に保存され，その中央値 CRm_k を平均値とした標準偏差 0.1 の正規分布 $\mathcal{N}(CRm_k, 0.1)$ より個体ごとにサンプリングされる．*Update* $(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i^{t-1})$ に相当する操作として，解更新の可否にかかわらず設定した $\boldsymbol{\theta}_i$ をそのまま利用する．なお，SaDE の改良手法としての SaNSDE [28] では，$\theta_{F,i}$ についても適応的に調整している．

4. 提案手法

jDE や SaDE，JADE をはじめとする自己適応型 DE を対象とし，一般化された自己適応型 DE フレームワークに従って提案手法のメカニズムを説明する．

4.1 概要

自己適応型 DE は，解更新に成功した場合にアルゴリズム構成を再利用する方法や逐次的にサンプリング元の確率分布を調整する方法を用いることで，好適なアルゴリズム構成を生成する改良が成されている．一方で，生成するア

ルゴリズム構成は乱数に依存するため，乱数によっては性能改善に寄与しないアルゴリズムが生成される可能性もある．そこで，自己適応型 DE は生成されたアルゴリズム構成をそのまま使用する点では共通の方法を採用していることから，試行錯誤しながら調整する傾向が強くなる．しかしながら，少ない解評価回数を想定した場合は，この試行錯誤的な調整を可能な限り排除することが，最適化性能を改善するうえで重要と考えられる．

そこで提案手法は，アルゴリズム構成候補を複数生成し，その中から好適と考えられるアルゴリズム構成を選択することで，試行錯誤的な調整要素を緩和することを狙う．一般的な自己適応型 DE では，1 つの個体につき 1 つのアルゴリズム構成のみをサンプリングする．一方で，提案手法を組み込むと，1 つの個体につきアルゴリズム構成を一度に複数生成することになる．そして，これらのアルゴリズム構成候補から生成した仮子個体と，すでに発見された優良解とのユークリッド距離をそれぞれ計算し，これが最小となるアルゴリズム構成を出力する．つまり，優良解に最も近づいた仮子個体を生成できたアルゴリズム構成を好適と定義する．これは，1 章で述べたように，優良解の周辺領域を局所探索するバイアスを高めたアルゴリズム構成によって，収束速度を改善することを意図している．加えて，早期収束を抑制する工夫として，出力されたアルゴリズム構成を用いて子個体を改めて生成する．これは，局所探索のバイアスを与えながらも，解の多様性の急速な低下を突然変異の非決定性によって抑制することを意図する．

なお，著者らの分析から，基準個体と仮個体のユークリッド距離と採用されるアルゴリズム構成の間には明確な相関が確認できていない．この理由としては，基準個体に近い個体を生成するアルゴリズム構成は複数種類存在すること，仮個体と子個体は乱数に依存して生成されることがあげられる．したがって，提案手法は，基準個体に近い仮個体を生成したアルゴリズム構成を用いるという簡略的な検証方法を採用し，個体ごとに独立してアルゴリズム構成を調整する．

4.2　メカニズム

4.2.1　基本的な流れ

提案手法は，自己適応型 DE が 1 つの個体に割り当てるアルゴリズム構成を生成するプロセスに適用される．すなわち，一般化された自己適応型 DE フレームワーク（Algorithm 2）において，アルゴリズム構成を生成する $Sample(\boldsymbol{\theta}_i^{t-1})$ を修正する．具体的なメカニズムは以下のとおりとなる．

ある個体 $\boldsymbol{x}_i$ に割り当てるアルゴリズム構成 $\boldsymbol{\theta}_i$ を生成することを考える（$i = 1, 2, \ldots, N$）．まず，事前検証に用いる基準個体 $\boldsymbol{x}_i^*$ を定義する．具体的な定義方法は次項で述べる．そして，C 個のアルゴリズム構成候補 $\boldsymbol{\theta}'_{i,j} \in \Theta'_i$ を生成する（$j = 1, 2, \ldots, C$）．ここで，Θ'_i はアルゴリズム構成候補の集合とする．生成するアルゴリズム構成候補は，利用する自己適応型 DE の生成方法を C 回適用することで得られる．たとえば，jDE では $Sample\text{-}jDE(\boldsymbol{\theta}_i^{t-1})$ を，SaDE では $Sample\text{-}SaDE(\boldsymbol{\theta}_i^{t-1})$ を，JADE では $Sample\text{-}JADE(\boldsymbol{\theta}_i^{t-1})$ をそれぞれ C 回だけ繰り返して実行する．また，$C \in \mathbb{N}$ は提案手法で用いるハイパーパラメータである．

次に，自己適応型 DE の子個体生成プロセスを実行し，$\boldsymbol{\theta}'_{i,j}$ を用いて仮子個体 $\boldsymbol{u}'_{\boldsymbol{\theta}'_{i,j}}$ を生成する．たとえば，jDE では，$\boldsymbol{\theta}'_{i,j}$ で指定される $\theta'_{F,i,j}$ と $\theta'_{CR,i,j}$ を用いて突然変異（*rand/1*）と交叉（binomial 交叉）を適用する．すべての $\boldsymbol{\theta}'_{i,j} \in \Theta'_i$ に対してこの操作を実行し，C 個の仮個体を生成する．ここで，$\boldsymbol{u}'_{\boldsymbol{\theta}'_{i,j}}$ の解評価は行わないことに留意されたい．そして，$\boldsymbol{u}'_{\boldsymbol{\theta}'_{i,j}}$ と $\boldsymbol{x}_i^*$ のユークリッド距離を計算し，その値が最小となるアルゴリズム構成候補を $\boldsymbol{\theta}_i$ と設定する．つまり，$\boldsymbol{\theta}_i$ は以下の式で得られる．

$$\boldsymbol{\theta}_i = \underset{\boldsymbol{\theta}'_{i,j} \in \Theta'_i}{\arg\min} ||\boldsymbol{x}_i^* - \boldsymbol{u}'_{\boldsymbol{\theta}'_{i,j}}||. \tag{10}$$

以上が提案手法の基本的な流れとなる．なお，$\boldsymbol{\theta}_i$ が決定した後は，$\boldsymbol{\theta}_i$ を用いて自己適応型 DE の子個体生成プロセスを適用し，$\boldsymbol{x}_i$ に対する $\boldsymbol{u}_i$ を本生成する．そして，次の個体の $\boldsymbol{\theta}_{i+1}$ も上記のメカニズムを経て決定される．

4.2.2　基準個体の定義

優良解の周辺に対する局所探索のバイアスを生むために，基準個体 $\boldsymbol{x}_i^*$ は現在までに発見した優良解に設定することが好ましいと考える．本論文では，以下の 4 つの戦略を考える．

***rand* 戦略**　現在の解集合 $\mathcal{P}$ からランダムに選択し $\boldsymbol{x}_i^*$ に設定する．

***greedy* 戦略**　現在の解集合 $\mathcal{P}$ における最良個体を選択し $\boldsymbol{x}_i^*$ に設定する．

***p-best* 戦略**　パラメータ $p \in [0, 1]$ を用いて評価値が上位 $p \times N$ 位までの個体からランダムに選択し $\boldsymbol{x}_i^*$ に設定する．

ϵ-*greedy* 戦略　閾値 $\epsilon \in [0, 1]$ を設け，確率 ϵ で *rand* 戦略を，確率 $1 - \epsilon$ で *greedy* 戦略を適用し $\boldsymbol{x}_i^*$ に設定する．

なお，*p-best* 戦略は，突然変異戦略 *current-to-pbest* を参考にしている．*rand* 戦略は，より解の多様性を重視した傾向にあるといえる．*greedy* 戦略では，すべての個体に同じ基準個体が設定され，収束速度を改善する可能性がある．これらに対し，パラメータ p や ϵ の値に依存するが，*p-best* 戦略と *ϵ-greedy* 戦略は，より保守的な戦略である．

これらの戦略を用いた性能比較実験は 6 章で行うが，単純かつ決定的な操作である *greedy* 戦略を既定の戦略として設定する．*greedy* 戦略を用いた提案手法が jDE と SaDE，JADE の性能を安定して改善するならば，追加のパラメータ p や ϵ を設定する必要がなくなる．この場合，すべての

Algorithm 9 *Sample*$(\mathcal{P}, \boldsymbol{\theta}_i, \boldsymbol{x}_i^*)$

Require: $\mathcal{P}, \boldsymbol{\theta}_i, \boldsymbol{x}_i^*$
  if $f(\boldsymbol{u}_i^{t-1}) > f(\boldsymbol{x}_i^{t-1})$ **then**
    $\Theta_i' = \emptyset$
    for $j = 1$ **to** C **do**
      $\Theta_i' \leftarrow$ Add sampled configuration $\boldsymbol{\theta}_{i,j}'$
      $\boldsymbol{v}_{\boldsymbol{\theta}_{i,j}'}' \leftarrow$ *Mutation*$(\mathcal{P}, \theta_{v,i,j}', \theta_{F,i,j}')$
      $\boldsymbol{u}_{\boldsymbol{\theta}_{i,j}'}' \leftarrow$ *Crossover*$(\boldsymbol{x}_i, \boldsymbol{v}_{\boldsymbol{\theta}_{i,j}'}', \theta_{u,i,j}', \theta_{CR,i,j}')$
    $\boldsymbol{\theta}_i \leftarrow \underset{\boldsymbol{\theta}_{i,j}' \in \Theta_i'}{\arg\min} ||\boldsymbol{x}_i^* - \boldsymbol{u}_{\boldsymbol{\theta}_{i,j}'}'||$
  return $\boldsymbol{\theta}_i$

アルゴリズム構成が同一の最良個体をもとに選択されることとなる．しかしながら，提案手法は，基準個体の近傍に必ず解を生成できるアルゴリズム構成を生成することは意図していない．あくまでも自己適応型 DE の生成方法を踏襲しながら 1 つのアルゴリズム構成を選択するものであり，*greedy* 戦略を用いることが，解の多様性の著しい低下を招く原因とはなりえない．

4.3　計算時間の削減

4.2.1 項に示したように，提案手法は実装が容易なメカニズムで実現できるが，このメカニズムを全個体に適用するため計算時間が増加することが懸念される．特に，個体ごとに C 個のアルゴリズム構成候補を生成する繰り返し処理に時間がかかる．高計算コストな最適化問題を想定すると，この個体あたりの計算時間の増加は無視できる可能性がある．しかしながら，解の評価時間が数分などの中程度の計算コストとなる最適化問題も見越し，計算時間を削減する方法を検討する．

具体的には，提案手法のメカニズムにおいて，前世代で解更新に成功した（$f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i)$）個体は，現在のアルゴリズム構成をそのまま出力し再利用することで，計算時間の削減を行う．したがって，前世代で解更新に失敗した（$f(\boldsymbol{u}_i) > f(\boldsymbol{x}_i)$）個体にのみ，アルゴリズム構成候補の生成から選択までを実行する．この計算時間の削減を適用した提案手法として，擬似関数 *Sample*$(\mathcal{P}, \boldsymbol{\theta}_i, \boldsymbol{x}_i^*)$ を Algorithm 9 に示す．

5.　実験

本章では，jDE と SaDE，JADE に提案手法を組み込み，ベンチマーク問題を用いてその性能を比較する．

5.1　実験設定

5.1.1　ベンチマーク問題

The special session and competition on Real-Parameter Single Objective Optimization at IEEE Congress on Evolutionary Computation 2013（CEC 2013）[30] で定義される 28 個の制約なし単一目的実数値連続最適化問題ベンチマーク関数 $F_1, \ldots, F_{28}$ を用いる．これらの関数は，単峰性関数 $F_1 \sim F_5$，多峰性関数 $F_6 \sim F_{20}$ および $F_1 \sim F_{20}$ から選択した関数を合成した関数 $F_{21} \sim F_{28}$ に分類される．問題の次元数は $D = 10, 30, 50, 100$ とし，次元数の増加に対する提案手法のスケーラビリティを評価する．したがって，合計 112（$= 28 \times 4$）の実験ケースを行う．コンペティションの規定どおり，個体 $\boldsymbol{x}$ の定義域は $\boldsymbol{x} \in [-100, 100]^D$ とする [30]．

5.1.2　比較手法と評価方法

提案手法を組み込んだ jDE と SaDE，JADE と組み込む前の jDE と SaDE，JADE を比較する．なお，すべての手法に対して，初期解は一様分布を用いて生成される．提案手法では，C を大きく設定するほど好適なアルゴリズム構成が見つかる可能性は高くなるが，計算時間の増加を避けるために $C = 10$ とする．jDE のパラメータは，$F_{init} = 0.5$，$CR_{init} = 0.9$，$N = 100$，$\tau_F = 0.1$，$\tau_{CR} = 0.1$ とする [22]．SaDE のパラメータは，$p_{k,init} = 0.25$，$CRm_{k,init} = 0.5$，$LP = 50$，$\epsilon = 0.01$ とする [25]．JADE のパラメータは，$\mu_{F,init} = 0.5$，$\mu_{CR,init} = 0.5$，$c = 0.1$，$p_{\min} = 0.05$，$p_{\max} = 0.2$ とする [23]．

解評価回数を最大 10,000 とし，異なる乱数シードを用いた独立した 51 試行で得られた性能（最良値の平均値）を計算する．この平均値の比較は，500，1,000，3,000，10,000 の解評価回数ごとに行う．また，jDE と提案手法を組み込んだ jDE，SaDE と提案手法を組み込んだ SaDE，JADE と提案手法を組み込んだ JADE の 3 組を設定する．加えて，統計的有意差を確認するために次の 2 つのケースの検定を行う．

検定ケース 1　ある次元数 D のベンチマーク関数ごとに各組に対し Wilcoxon の符号順位検定を適用し，有意水準 0.05 の下で有意差を調べる．具体的には，$p \geq 0.05$ であれば有意差なしとして “∼”，$p < 0.05$ で提案手法を組み込んだ手法が優位ならば “+”，$p < 0.05$ で提案手法を組み込まない手法が優位ならば “–” とする．そして，500，1,000，3,000，10,000 の解評価回数ごとにこれらの合計を “+/ – / ∼” として報告する．これにより，すべての試行を通して手法間に有意差が見られるかを関数ごとに確認できる．

検定ケース 2　次元数 D ごとのすべての関数の平均値について，各組に対し Wilcoxon の符号順位検定を適用し，有意水準 0.05 の下で有意差を調べる．すべての関数を通して手法間に有意差が見られるかを p 値で確認できる．

なお，1,000 の解評価回数を規定値とし性能を表にまとめる．一方で，500，3,000，10,000 の解評価回数ごとの詳細な性能の表については，紙面の都合上省略し，検定結果のみを示す．

表 2　jDE と提案手法を適用した jDE の性能比較
Table 2　Comparison of fitness derived from jDE and jDE with the proposed method.

	$D=10$		$D=30$		$D=50$		$D=100$	
	jDE	ours	jDE	ours	jDE	ours	jDE	ours
F1	5.04E+03	4.54E+03 ∼	5.92E+04	5.16E+04 +	1.25E+05	1.08E+05 +	3.28E+05	2.89E+05 +
F2	3.60E+07	3.86E+07 ∼	9.40E+08	9.15E+08 ∼	2.60E+09	2.57E+09 ∼	1.23E+10	1.09E+10 +
F3	1.32E+10	1.27E+10 ∼	5.36E+15	5.53E+15 ∼	2.89E+16	1.04E+16 +	1.30E+24	5.15E+23 +
F4	4.53E+04	4.14E+04 ∼	1.66E+05	1.57E+05 ∼	2.61E+05	2.49E+05 ∼	5.49E+05	5.30E+05 ∼
F5	3.18E+03	2.42E+03 +	4.73E+04	4.58E+04 ∼	1.24E+05	1.10E+05 +	4.15E+05	3.62E+05 +
F6	3.17E+02	3.00E+02 ∼	9.16E+03	8.16E+03 +	1.93E+04	1.51E+04 +	1.03E+05	8.77E+04 +
F7	1.70E+02	1.66E+02 ∼	6.16E+04	4.68E+04 ∼	8.65E+04	5.89E+04 +	3.94E+08	3.33E+08 ∼
F8	2.07E+01	2.07E+01 ∼	2.12E+01	2.12E+01 ∼	2.13E+01	2.13E+01 ∼	2.15E+01	2.15E+01 ∼
F9	1.11E+01	1.14E+01 ∼	4.50E+01	4.49E+01 ∼	7.98E+01	8.03E+01 ∼	1.72E+02	1.73E+02 ∼
F10	6.16E+02	5.71E+02 ∼	8.22E+03	7.54E+03 +	1.72E+04	1.59E+04 +	5.14E+04	4.71E+04 +
F11	1.36E+02	1.26E+02 +	9.48E+02	8.71E+02 +	1.95E+03	1.78E+03 +	5.35E+03	4.86E+03 +
F12	1.36E+02	1.35E+02 ∼	9.94E+02	9.14E+02 +	1.88E+03	1.73E+03 +	5.58E+03	5.00E+03 +
F13	1.36E+02	1.30E+02 ∼	9.67E+02	8.98E+02 +	1.90E+03	1.75E+03 +	5.66E+03	4.94E+03 +
F14	2.19E+03	2.14E+03 +	8.60E+03	8.44E+03 ∼	1.55E+04	1.52E+04 +	3.43E+04	3.39E+04 +
F15	2.19E+03	2.13E+03 ∼	8.97E+03	8.94E+03 ∼	1.64E+04	1.64E+04 ∼	3.43E+04	3.42E+04 ∼
F16	2.52E+00	2.49E+00 ∼	4.53E+00	4.47E+00 ∼	5.51E+00	5.44E+00 ∼	5.70E+00	5.81E+00 ∼
F17	2.25E+02	2.02E+02 +	1.94E+03	1.68E+03 +	4.31E+03	3.42E+03 +	1.04E+04	8.54E+03 +
F18	2.28E+02	2.03E+02 +	1.92E+03	1.66E+03 +	4.22E+03	3.50E+03 +	1.04E+04	8.47E+03 +
F19	3.09E+03	2.74E+03 ∼	1.94E+06	1.47E+06 +	6.65E+06	3.91E+06 +	7.66E+07	5.05E+07 +
F20	4.70E+00	4.66E+00 ∼	1.50E+01	1.50E+01 ∼	2.50E+01	2.50E+01 ∼	5.00E+01	5.00E+01 ∼
F21	7.16E+02	6.98E+02 ∼	4.63E+03	4.26E+03 +	1.02E+04	8.69E+03 +	2.29E+04	1.82E+04 +
F22	2.36E+03	2.34E+03 ∼	9.47E+03	9.31E+03 ∼	1.67E+04	1.66E+04 ∼	3.64E+04	3.60E+04 +
F23	2.47E+03	2.49E+03 ∼	9.55E+03	9.43E+03 ∼	1.73E+04	1.72E+04 ∼	3.60E+04	3.61E+04 ∼
F24	2.33E+02	2.31E+02 +	3.54E+02	3.52E+02 ∼	5.08E+02	5.01E+02 ∼	1.49E+03	1.41E+03 ∼
F25	2.32E+02	2.32E+02 ∼	3.61E+02	3.61E+02 ∼	5.05E+02	5.05E+02 ∼	9.82E+02	9.79E+02 ∼
F26	2.18E+02	2.16E+02 ∼	3.74E+02	3.76E+02 ∼	5.05E+02	5.06E+02 ∼	7.62E+02	7.64E+02 ∼
F27	7.12E+02	7.10E+02 ∼	1.55E+03	1.54E+03 ∼	2.65E+03	2.60E+03 +	5.97E+03	5.72E+03 +
F28	1.21E+03	1.14E+03 +	7.16E+03	7.07E+03 ∼	1.29E+04	1.23E+04 +	3.68E+04	3.48E+04 +
検定ケース 1：+/−/∼		7/0/21		10/0/18		16/0/12		17/0/11
検定ケース 2：p value		0.0006956		0.0003274		0.00003644		0.00003667

5.2　実験結果

提案手法を jDE, SaDE, JADE に適用し，評価回数 1,000 において，次元数を $D=10,30,50,100$ と変化させたときの性能を**表 2**，**表 3**，**表 4** にそれぞれ示す．網掛けした部分は手法がもう一方の手法と同じかより高い性能を導出していることを表す．全体をとおして，提案手法を自己適応型 DE に適用することにより，多くのケースで性能が向上している．例として，jDE に提案手法を適用した場合，表 2 において $D=10,30,50,100$ の順にそれぞれ 28 個のうち 23，23，22，22 個の関数で性能が向上している．検定ケース 1 でも 7，10，16，17 個の関数で “+” となり，提案手法が優位である．特に次元数が増加し問題の難易度が上昇したときに，提案手法の有効性が高くなる．また，表 2 の最後に示したように，検定ケース 2 ですべての次元数で $p<0.05$ となり，関数全体での統計的有意差も見られている．すなわち，28 個の関数における全体的な提案手法の有効性が示されている．一方，$D=10,50,100$ における F_9 や $D=30,50,100$ における F_{26} のように提案手法が性能向上に失敗していることから，提案手法の適用によって性能が低下する場合もある．しかしながら，検定ケース 1 における提案手法を組み込まない jDE が統計的に優位であることを示す “–” は存在せず，検定ケース 2 でも $p<0.05$ であることから，この性能低下に統計的な有意性はない．

SaDE に提案手法を適用した場合（表 3）でも jDE の場合と同様の傾向であり，特に $D=50$ ではすべてのケースで性能を向上または維持している．$D=10,30,50,100$ の順に全 28 個のうち 16，17，18，19 個の関数で “+” となり，有意差をもって提案手法が性能を向上している．同じく SaDE でもすべての関数で，提案手法を用いない場合が優位であることを示す “–” は存在しない．さらに表 3 の最後に示したように，すべての次元数で $p<0.05$ となり関数全体の統計的有意差も見られている．これらの結果から，ハイパーパラメータのみを調整する jDE に対し，SaDE が遺伝的オペレータまで調整することを考えると，提案手法

表 3　SaDE と提案手法を適用した SaDE の性能比較
Table 3　Comparison of fitness derived from SaDE and SaDE with the proposed method.

	$D=10$		$D=30$		$D=50$		$D=100$	
	SaDE	ours	SaDE	ours	SaDE	ours	SaDE	ours
F1	3.68E+03	2.85E+03 +	4.60E+04	3.64E+04 +	9.45E+04	7.28E+04 +	2.53E+05	1.97E+05 +
F2	3.66E+07	3.36E+07 ∼	8.75E+08	7.72E+08 +	2.36E+09	1.91E+09 +	1.05E+10	8.81E+09 +
F3	1.12E+10	9.55E+09 ∼	4.62E+15	9.41E+14 ∼	5.37E+15	2.34E+15 +	3.09E+23	2.28E+22 +
F4	4.71E+04	4.82E+04 ∼	1.50E+05	1.49E+05 ∼	2.33E+05	2.25E+05 ∼	4.98E+05	4.81E+05 ∼
F5	2.60E+03	1.96E+03 +	4.14E+04	3.23E+04 +	8.70E+04	6.47E+04 +	2.92E+05	2.19E+05 +
F6	2.91E+02	2.28E+02 +	7.00E+03	5.26E+03 +	1.23E+04	8.52E+03 +	7.44E+04	5.23E+04 +
F7	1.57E+02	1.45E+02 +	2.58E+04	1.36E+04 +	4.40E+04	2.06E+04 +	2.02E+08	6.37E+07 +
F8	2.07E+01	2.08E+01 ∼	2.12E+01	2.12E+01 ∼	2.13E+01	2.13E+01 ∼	2.15E+01	2.15E+01 ∼
F9	1.13E+01	1.09E+01 +	4.49E+01	4.44E+01 ∼	8.01E+01	8.00E+01 ∼	1.73E+02	1.72E+02 ∼
F10	5.36E+02	4.18E+02 +	6.55E+03	5.66E+03 +	1.41E+04	1.14E+04 +	4.24E+04	3.44E+04 +
F11	1.18E+02	1.08E+02 +	8.12E+02	6.96E+02 +	1.56E+03	1.29E+03 +	4.27E+03	3.48E+03 +
F12	1.27E+02	1.10E+02 +	8.28E+02	7.08E+02 +	1.54E+03	1.32E+03 +	4.44E+03	3.52E+03 +
F13	1.25E+02	1.15E+02 +	8.10E+02	7.25E+02 +	1.54E+03	1.31E+03 +	4.39E+03	3.59E+03 +
F14	2.13E+03	2.06E+03 ∼	8.45E+03	8.47E+03 ∼	1.54E+04	1.53E+04 ∼	3.41E+04	3.41E+04 ∼
F15	2.17E+03	2.12E+03 ∼	8.88E+03	8.91E+03 ∼	1.65E+04	1.63E+04 ∼	3.41E+04	3.42E+04 ∼
F16	2.43E+00	2.42E+00 ∼	4.45E+00	4.56E+00 ∼	5.48E+00	5.43E+00 ∼	5.73E+00	5.65E+00 ∼
F17	1.87E+02	1.48E+02 +	1.44E+03	1.07E+03 +	2.99E+03	2.12E+03 +	7.40E+03	5.42E+03 +
F18	1.93E+02	1.58E+02 +	1.43E+03	1.07E+03 +	2.96E+03	2.15E+03 +	7.41E+03	5.31E+03 +
F19	1.43E+03	6.31E+02 +	9.11E+05	5.36E+05 +	2.55E+06	1.07E+06 +	3.27E+07	1.61E+07 +
F20	4.68E+00	4.62E+00 ∼	1.50E+01	1.50E+01 ∼	2.50E+01	2.50E+01 ∼	5.00E+01	5.00E+01 ∼
F21	6.48E+02	5.75E+02 +	3.75E+03	3.15E+03 +	8.06E+03	6.37E+03 +	1.71E+04	1.36E+04 +
F22	2.43E+03	2.28E+03 +	9.34E+03	9.24E+03 ∼	1.67E+04	1.65E+04 ∼	3.60E+04	3.59E+04 ∼
F23	2.48E+03	2.45E+03 ∼	9.57E+03	9.57E+03 ∼	1.73E+04	1.73E+04 ∼	3.60E+04	3.62E+04 ∼
F24	2.32E+02	2.31E+02 ∼	3.46E+02	3.38E+02 +	4.88E+02	4.67E+02 +	1.20E+03	9.84E+02 +
F25	2.33E+02	2.32E+02 ∼	3.56E+02	3.44E+02 +	5.03E+02	4.54E+02 +	9.41E+02	7.76E+02 +
F26	2.18E+02	2.15E+02 ∼	3.63E+02	3.41E+02 +	5.05E+02	5.04E+02 ∼	7.60E+02	7.51E+02 +
F27	6.89E+02	6.70E+02 +	1.52E+03	1.51E+03 ∼	2.59E+03	2.53E+03 +	5.66E+03	5.39E+03 +
F28	1.11E+03	1.04E+03 +	6.60E+03	6.09E+03 +	1.14E+04	1.03E+04 +	3.22E+04	2.95E+04 +
検定ケース 1：+/−/∼		16/0/12		17/0/11		18/0/10		19/0/9
検定ケース 2：p value		0.00007775		0.00007643		0.00001306		0.00005129

がより大きなアルゴリズム構成の探索空間を扱う場合にも高い有効性があるといえる．

JADE に提案手法を適用した場合（表 4）においても，提案手法は JADE の性能を向上している．具体的には，$D=10,30,50,100$ の順に 14，13，17，16 個の関数で “+” となっている．一方，JADE が優位であることを示す “−” は $D=50,100$ でそれぞれ 1，2 個見られているが，検定ケース 2 ではすべての次元数で $p<0.05$ となり，全 28 個の関数を総合的に見ると提案手法が優位である．

次に，表 2 と表 3，表 4 では解評価回数 1,000 であったが，これを 500，1,000，3,000，10,000 としたときの検定ケース 1 の結果を “+/ − / ∼” で**表 5** に示す．全体として限られた解評価回数であっても性能が統計的に向上していることから提案手法の有効性が確認できる．$D=10,30,50$ では jDE と SaDE が優位であることを示す “−” の数は 0 であり，次元数が $D=100$ と上がってもその数はたかだか 1 個である．特に，解評価回数を 500 に減らした場合においても，jDE，SaDE，JADE に提案手法を用いた場合は “+” の数が多いように，性能の向上が見られる．次元数 $D=100$，解評価回数 10,000 の JADE における実験ケースを除いて，次元数が $D=50,100$ と増加した際は $D=10,30$ のときに比べて “+” の数が増えていることから，問題の難易度が上がったときに提案手法の適用はより一層効果的である．解評価回数 3,000 のときも提案手法を適用した自己適応型 DE が優位である傾向は同様である．解評価回数を 10,000 まで緩和すると，jDE と SaDE に提案手法を用いた場合は各次元数ごとに 28 個中 21∼24 個の関数で，JADE に提案手法を用いた場合は 10∼23 個の関数で “+” となり，その合計数は 112 実験ケース中，jDE では 95 ケース，SaDE では 92 ケース，JADE では 69 ケースである．したがって，提案手法を jDE と SaDE，JADE に組み込むと，解評価回数を 500 と制限しても性能が向上

表 4　JADE と提案手法を適用した JADE の性能比較
Table 4　Comparison of fitness derived from JADE and JADE with the proposed method.

	$D=10$		$D=30$		$D=50$		$D=100$	
	JADE	ours	JADE	ours	JADE	ours	JADE	ours
F1	2.43E+03	1.65E+03 +	2.98E+04	2.58E+04 +	7.00E+04	5.79E+04 +	2.06E+05	1.81E+05 +
F2	3.00E+07	2.38E+07 +	6.11E+08	5.29E+08 +	1.61E+09	1.36E+09 +	6.84E+09	5.94E+09 +
F3	6.47E+09	4.99E+09 +	2.82E+13	1.77E+13 ∼	2.56E+14	4.38E+13 +	7.74E+21	2.69E+21 +
F4	3.98E+04	3.89E+04 ∼	1.36E+05	1.31E+05 ∼	2.10E+05	2.09E+05 ∼	4.55E+05	4.16E+05 +
F5	1.47E+03	1.18E+03 +	2.44E+04	2.04E+04 +	5.69E+04	4.39E+04 +	2.02E+05	1.67E+05 +
F6	1.52E+02	1.35E+02 ∼	3.29E+03	2.80E+03 +	7.30E+03	5.63E+03 +	5.32E+04	4.46E+04 +
F7	1.13E+02	1.06E+02 ∼	3.14E+03	2.59E+03 ∼	7.71E+03	4.85E+03 +	2.93E+07	1.66E+07 +
F8	2.07E+01	2.07E+01 ∼	2.12E+01	2.12E+01 ∼	2.13E+01	2.13E+01 ∼	2.15E+01	2.15E+01 ∼
F9	1.13E+01	1.10E+01 ∼	4.45E+01	4.44E+01 ∼	7.98E+01	7.97E+01 ∼	1.71E+02	1.71E+02 ∼
F10	3.29E+02	2.56E+02 +	4.57E+03	3.74E+03 +	1.04E+04	8.37E+03 +	3.25E+04	2.82E+04 +
F11	9.73E+01	8.69E+01 +	6.15E+02	5.55E+02 +	1.27E+03	1.13E+03 +	3.50E+03	3.14E+03 +
F12	1.00E+02	9.28E+01 +	6.53E+02	5.85E+02 +	1.26E+03	1.11E+03 +	3.59E+03	3.21E+03 +
F13	9.64E+01	9.65E+01 ∼	6.53E+02	5.84E+02 +	1.28E+03	1.11E+03 +	3.71E+03	3.19E+03 +
F14	1.95E+03	1.98E+03 ∼	8.42E+03	8.39E+03 ∼	1.52E+04	1.51E+04 ∼	3.37E+04	3.38E+04 ∼
F15	2.12E+03	2.10E+03 ∼	8.89E+03	8.88E+03 ∼	1.64E+04	1.62E+04 ∼	3.40E+04	3.40E+04 ∼
F16	2.49E+00	2.53E+00 ∼	4.39E+00	4.43E+00 ∼	5.41E+00	5.56E+00 ∼	5.74E+00	5.80E+00 ∼
F17	1.53E+02	1.33E+02 +	1.10E+03	9.65E+02 +	2.46E+03	2.08E+03 +	6.61E+03	5.64E+03 +
F18	1.49E+02	1.34E+02 +	1.11E+03	9.54E+02 +	2.49E+03	2.05E+03 +	6.56E+03	5.57E+03 +
F19	1.79E+02	1.03E+02 +	2.03E+05	1.22E+05 +	9.75E+05	4.65E+05 +	1.65E+07	1.11E+07 +
F20	4.55E+00	4.39E+00 +	1.50E+01	1.50E+01 ∼	2.50E+01	2.50E+01 +	5.00E+01	5.00E+01 ∼
F21	5.71E+02	5.32E+02 +	3.23E+03	2.93E+03 +	6.95E+03	6.14E+03 +	1.57E+04	1.33E+04 +
F22	2.23E+03	2.18E+03 ∼	9.15E+03	9.12E+03 ∼	1.64E+04	1.64E+04 ∼	3.56E+04	3.57E+04 ∼
F23	2.46E+03	2.47E+03 ∼	9.51E+03	9.50E+03 ∼	1.73E+04	1.72E+04 ∼	3.61E+04	3.60E+04 ∼
F24	2.30E+02	2.30E+02 ∼	3.30E+02	3.30E+02 ∼	4.43E+02	4.45E+02 ∼	8.80E+02	9.52E+02 −
F25	2.30E+02	2.30E+02 ∼	3.48E+02	3.48E+02 ∼	4.82E+02	4.89E+02 −	8.91E+02	9.25E+02 −
F26	2.25E+02	2.20E+02 ∼	3.57E+02	3.43E+02 ∼	5.01E+02	4.95E+02 +	7.40E+02	7.40E+02 ∼
F27	6.50E+02	6.39E+02 +	1.46E+03	1.46E+03 ∼	2.45E+03	2.43E+03 ∼	5.13E+03	5.14E+03 ∼
F28	1.02E+03	9.54E+02 +	5.35E+03	5.01E+03 +	9.41E+03	8.67E+03 +	2.84E+04	2.57E+04 +
検定ケース 1：+/−/∼		14/0/14		13/0/15		17/1/10		16/2/10
検定ケース 2：p value		0.0002527		0.00003291		0.00004313		0.0004418

表 5　jDE, SaDE, JADE と提案手法を適用した jDE, SaDE, JADE とのそれぞれの Wilcoxon の符号順位検定の結果（+/ − / ∼）
Table 5　Results of Wilcoxon's signed-rank test between jDE, SaDE, JADE, and jDE, SaDE, and JADE with the proposed method, respectively (+/ − / ∼).

D	10			30			50			100		
解評価回数	vs jDE	vs SaDE	vs JADE	vs jDE	vs SaDE	vs JADE	vs jDE	vs SaDE	vs JADE	vs jDE	vs SaDE	vs JADE
500	1/0/27	6/0/22	7/0/21	7/0/21	12/0/16	7/1/20	12/0/16	12/0/16	10/1/17	12/0/16	14/0/14	10/ 3/15
1,000	7/0/21	16/0/12	14/0/14	10/0/18	17/0/11	13/0/15	16/0/12	18/0/10	17/1/10	17/0/11	19/0/ 9	16/ 2/10
3,000	17/0/11	20/0/ 8	22/0/ 6	18/0/10	22/0/ 6	21/0/ 7	20/0/ 8	21/0/ 7	21/1/ 6	22/1/ 5	21/0/ 7	21/ 0/ 7
10,000	24/0/ 4	24/0/ 4	20/0/ 8	24/0/ 4	24/0/ 4	23/2/ 3	24/0/ 4	23/0/ 5	16/6/ 6	23/0/ 5	21/0/ 7	10/12/ 6

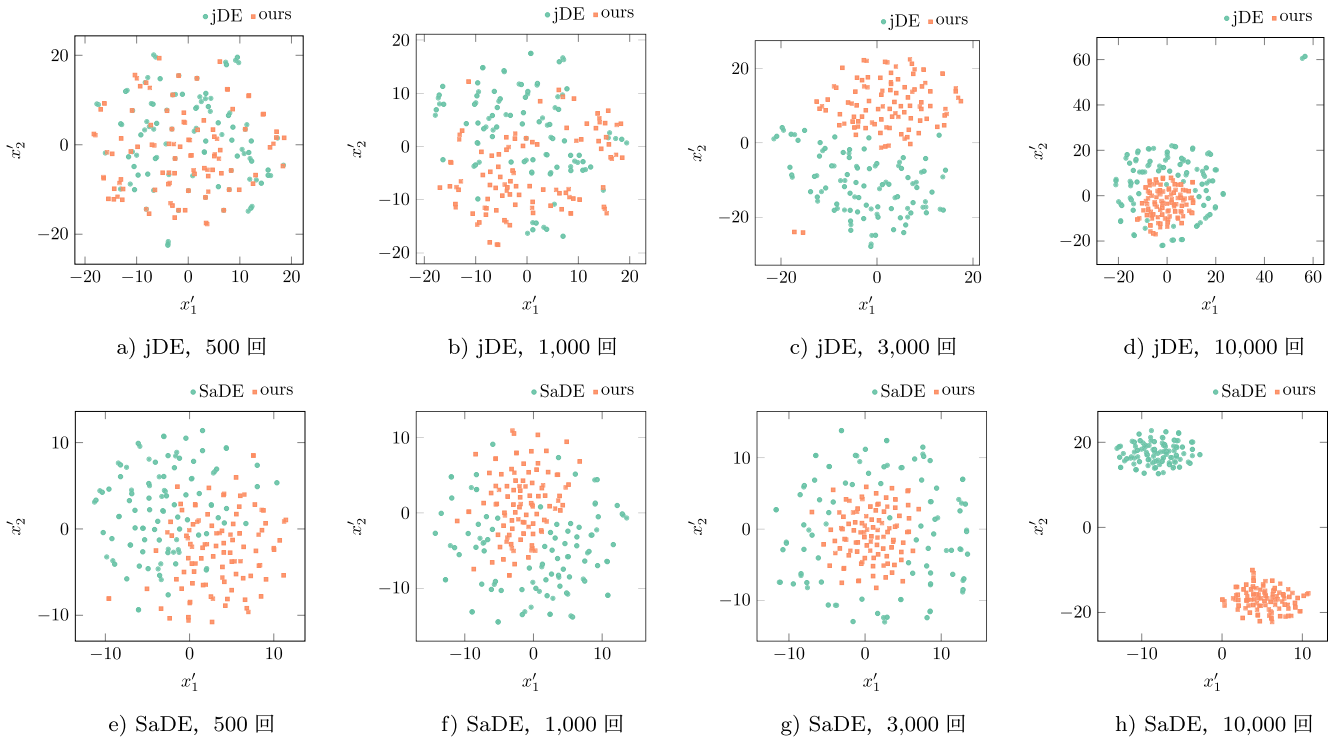
し，10,000 まで緩和すると $D=10,30,50,100$ の多くの次元数で “+” の数が顕著に増加し，提案手法の有効性が確認できる．

なお，すべての関数において最適解の評価値は 0 となる．提案手法ならびに比較手法の性能は最適解の評価値から大きく乖離するが，本論文が設定した評価回数のもとで提案手法の有効性が確認できる．

6. 考察

6.1 解集合の多様性の分析

提案手法では，子個体の本生成メカニズムにより解の多様性の低下を抑制することを意図していた．この効果を



a) jDE，500 回　b) jDE，1,000 回　c) jDE，3,000 回　d) jDE，10,000 回

e) SaDE，500 回　f) SaDE，1,000 回　g) SaDE，3,000 回　h) SaDE，10,000 回

図 1　F1（$D = 100$）での jDE，SaDE と提案手法を適用した jDE，SaDE における解集合の分布

Fig. 1　Scatter diagram of population of jDE, SaDE, and jDE and SaDE with the proposed method on F1 ($D = 100$).

検証するために，本節では jDE，SaDE ならびに提案手法が生成した解集合内の個体分布を解評価回数 500，1,000，3,000，10,000 回ごとに比較する．具体的には，単峰性関数 F1（Sphere Function，$D = 100$）と多峰性の合成関数 F28（Composition Function 8，$D = 100$）において，各手法が生成した個体を t-SNE で 2 次元に写像した分布図をそれぞれ**図 1** と**図 2** に示す．なお，図中の ours は jDE と SaDE に提案手法をそれぞれ組み込んだ手法を示す．

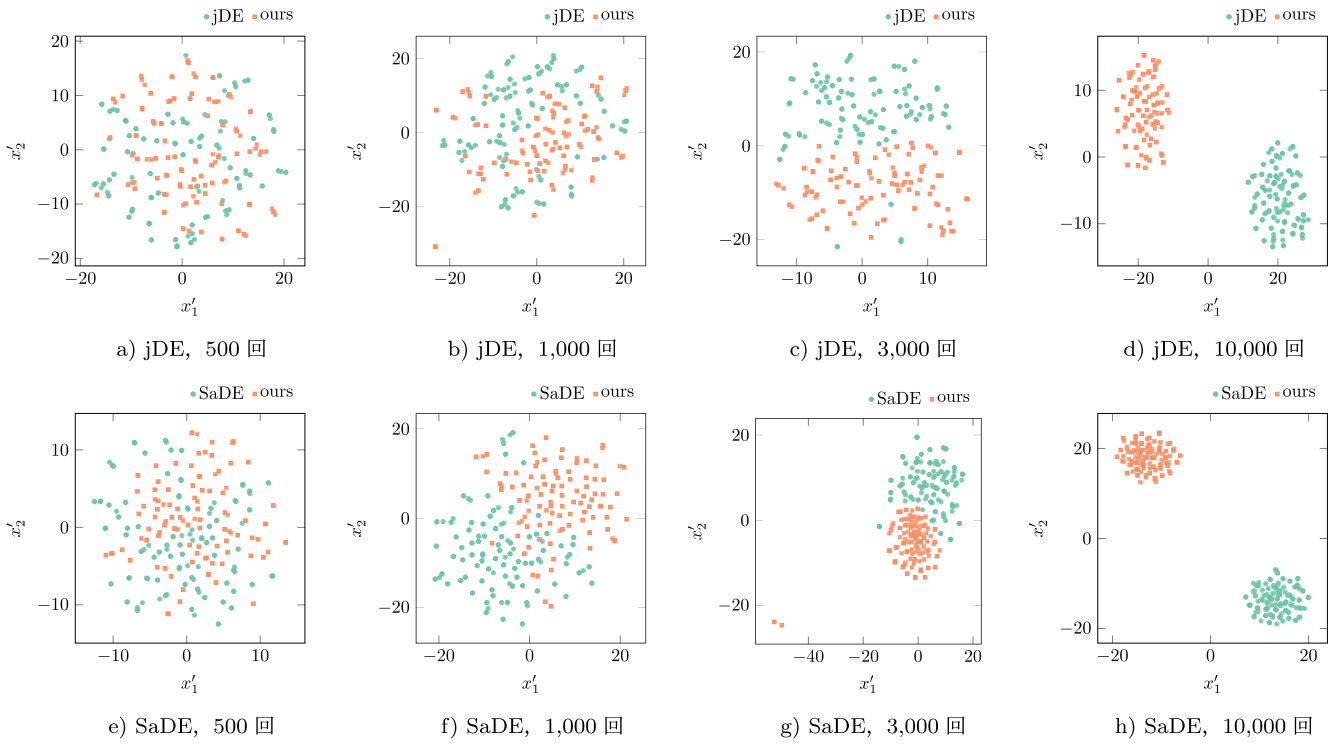
図 1 の d) と g) を除いて，F1 において提案手法が生成した解集合は jDE と SaDE と同程度の分散がある．一方で，同図の d) や g) に示すように，解評価回数が増えると提案手法の解集合の多様性は低下する傾向にある．しかし，単峰性関数である F1 において，この傾向は探索性能を改善するうえで妥当である．また，図 2 のすべての図に示すように，提案手法が生成した解集合は jDE および SaDE と同程度の分散がある．以上より，少ない解評価回数における単峰性関数 F1 と，多峰性の合成関数 F28 において提案手法は解の多様性の低下を抑制できていることが分かる．なお，他の実験ケースにおいても同様の傾向が確認できている．

6.2　計算時間の削減を行わない場合との比較

4.3 節において述べたように，計算時間の削減のため，提案手法では前世代に解更新に失敗した個体のアルゴリズム構成のみを調整した．ここでは，すべての個体についてアルゴリズム構成を調整する方針をとった場合に，性能と計算時間がどのように変化するかを考察する．**表 6** に，有意水準 0.05 の下で Friedman 検定を行った際に得られた，全問題の平均順位を示す．また，Friedman 検定でいずれかの手法間に有意差が見られたときに，事後検定として行う Wilcoxon の符号順位検定結果を Holm 法で調整した後に，有意差が見られた組を**表 7** に示す．ここで，表 6 と表 7 において，ours (F) は 5 章で用いた設定であり，解更新に失敗した個体のアルゴリズム構成を調整する方法を表す．ours (E) はすべての個体のアルゴリズム構成を調整する方法を表す．

jDE では解評価回数 500 など極端に評価回数が限られた場合においては，ours (E) が ours (F) より比較的高い性能を導出しており，なおかつ jDE に対して優位である．これは，全個体でアルゴリズム構成の調整を行う ours (E) が，ours (F) よりも多くの調整を行うため，特に少ない評価回数ではより適切にアルゴリズム構成を調整できているものと考えられる．一方で，解評価回数が増えると ours (F) の再利用モデルとしての機能が働き，たとえば，次元数 $D = 10, 30$ における解評価回数 10,000 では ours (F) が ours (E) に対し優れている．

一方，SaDE では全体として解更新失敗時のみアルゴリズム構成の調整を行う ours (F) が高い性能を導出してい



a) jDE，500 回　b) jDE，1,000 回　c) jDE，3,000 回　d) jDE，10,000 回

e) SaDE，500 回　f) SaDE，1,000 回　g) SaDE，3,000 回　h) SaDE，10,000 回

図 2　F28（$D = 100$）での jDE，SaDE と提案手法を適用した jDE，SaDE における解集合の分布

Fig. 2　Scatter diagram of population of jDE, SaDE, and jDE and SaDE with the proposed method on F28 ($D = 100$).

表 6　jDE，SaDE，JADE と提案手法を適用した jDE，SaDE，JADE においてアルゴリズム構成の調整頻度を変更したときの平均順位比較

Table 6　Comparison of the mean rankings of jDE, SaDE, JADE, and jDE, SaDE, and JADE with the proposed method, changing the frequency of algorithm configuration control.

D	10			30			50			100		
解評価回数	jDE	ours (F)	ours (E)	jDE	ours (F)	ours (E)	jDE	ours (F)	ours (E)	jDE	ours (F)	ours (E)
500	2.286	2.036	1.679	2.536	1.679	1.786	2.643	1.732	1.625	2.786	1.643	1.571
1,000	2.500	1.821	1.679	2.750	1.714	1.536	2.571	1.714	1.714	2.607	1.607	1.786
3,000	2.786	1.571	1.643	2.893	1.679	1.429	2.821	1.536	1.643	2.786	1.643	1.571
10,000	2.893	1.357	1.750	2.857	1.321	1.821	2.786	1.571	1.643	2.893	1.393	1.714
解評価回数	SaDE	ours (F)	ours (E)	SaDE	ours (F)	ours (E)	SaDE	ours (F)	ours (E)	SaDE	ours (F)	ours (E)
500	2.571	1.714	1.714	2.518	1.714	1.768	2.679	1.607	1.714	2.679	1.571	1.750
1,000	2.714	1.679	1.607	2.679	1.571	1.750	2.821	1.321	1.857	2.714	1.393	1.893
3,000	2.786	1.357	1.857	2.857	1.143	2.000	2.821	1.357	1.821	2.750	1.214	2.036
10,000	2.857	1.214	1.929	2.964	1.179	1.857	2.821	1.393	1.786	2.714	1.643	1.643
解評価回数	JADE	ours (F)	ours (E)	JADE	ours (F)	ours (E)	JADE	ours (F)	ours (E)	JADE	ours (F)	ours (E)
500	2.607	1.679	1.714	2.375	1.964	1.661	2.536	1.714	1.750	2.429	1.893	1.679
1,000	2.750	1.500	1.750	2.679	1.536	1.786	2.571	1.607	1.821	2.643	1.607	1.750
3,000	2.714	1.464	1.821	2.786	1.393	1.821	2.679	1.357	1.964	2.643	1.679	1.679
10,000	2.714	1.321	1.964	2.786	1.571	1.643	2.571	1.857	1.571	1.964	2.214	1.821

る．これは，アルゴリズム構成をランダムにサンプリングする jDE のランダム生成による再利用モデルよりも，確率分布を調整してサンプリングする SaDE の確率分布に基づく逐次更新モデルが，解更新失敗時のみアルゴリズム構成を調整する設定に相応しい可能性を示している．この理由は，SaDE は解更新の成功率をもとに確率分布を調整するため，解更新に失敗したアルゴリズム構成を効率的に調整でき，解更新に成功したアルゴリズム構成は長い世代に渡って探索に有効であると考えられるからである．加えて，ours (F) では，擬似関数 $Sample(\mathcal{P}, \boldsymbol{\theta}_i, \boldsymbol{x}_i^*)$ に解評価

表 7　jDE，SaDE，JADE と提案手法を適用した jDE，SaDE，JADE においてアルゴリズム構成の調整頻度を変更したときの比較における有意差検出組

Table 7　Significant difference detection pairs in comparison between jDE, SaDE, JADE, and jDE, SaDE, and JADE with the proposed method, changing the frequency of algorithm configuration control.

解評価回数	$D = 10$	$D = 30$	$D = 50$	$D = 100$
500	(E) - j	(E) - j	(E) - j, (F) - j	(E) - j, (F) - j
1,000	(E) - j	(E) - j, (F) - j	(E) - j, (F) - j	(E) - j, (F) - j
3,000	(E) - j, (F) - j	(E) - j, (F) - j	(E) - j, (F) - j	(E) - j, (F) - j
10,000	(E) - (F), (E) - j, (F) - j	(E) - (F), (E) - j, (F) - j	(E) - j, (F) - j	(E) - j, (F) - j
500	(E) - S, (F) - S	(E) - S, (F) - S	(E) - S, (F) - S	(E) - S, (F) - S
1,000	(E) - S, (F) - S	(E) - S, (F) - S	(E) - (F), (E) - S, (F) - S	(E) - (F), (E) - S, (F) - S
3,000	(E) - (F), (E) - S, (F) - S	(E) - (F), (E) - S, (F) - S	(E) - (F), (E) - S, (F) - S	(E) - (F), (E) - S, (F) - S
10,000	(E) - (F), (E) - S, (F) - S	(E) - (F), (E) - S, (F) - S	(E) - S, (F) - S	(E) - S, (F) - S
500	(E) - J, (F) - J	(E) - J	(E) - J, (F) - J	(E) - J, (F) - J
1,000	(E) - J, (F) - J	(E) - J, (F) - J	(E) - (F), (E) - J, (F) - J	(E) - J, (F) - J
3,000	(E) - (F), (E) - J, (F) - J	(E) - (F), (E) - J, (F) - J	(E) - (F), (E) - J, (F) - J	(E) - J, (F) - J
10,000	(E) - (F), (E) - J, (F) - J	(E) - J, (F) - J	(E) - J	–

辞書順に記載，j: jDE, S: SaDE, J: JADE, (E): ours (E), (F): ours (F)

表 8　jDE，SaDE，JADE と提案手法を適用した jDE，SaDE，JADE の全実験ケースでの平均実行時間比較（単位：秒）

Table 8　Comparison of the execution time of jDE, SaDE, JADE, and jDE, SaDE, and JADE with the proposed method, changing the frequency of algorithm configuration control (in seconds).

D	10			30			50			100		
解評価回数	jDE	ours (F)	ours (E)	jDE	ours (F)	ours (E)	jDE	ours (F)	ours (E)	jDE	ours (F)	ours (E)
500	3.297	2.612	3.278	4.890	5.663	7.045	6.796	8.901	11.27	10.18	15.42	19.69
1,000	6.736	5.861	7.499	9.938	13.04	15.96	13.83	20.14	25.59	20.98	34.52	43.29
3,000	19.21	19.91	23.56	29.39	45.79	49.72	40.95	68.56	78.88	64.04	110.7	131.6
10,000	63.09	72.26	79.10	100.4	163.8	163.5	136.6	248.9	260.9	214.8	411.3	437.7
解評価回数	SaDE	ours (F)	ours (E)	SaDE	ours (F)	ours (E)	SaDE	ours (F)	ours (E)	SaDE	ours (F)	ours (E)
500	1.085	1.908	2.939	2.097	3.543	5.649	3.103	4.888	8.367	5.635	8.639	14.95
1,000	2.236	4.313	7.096	4.370	8.970	13.82	6.478	12.26	20.52	11.77	22.26	36.71
3,000	6.576	13.26	22.81	13.38	29.76	46.16	19.92	38.82	68.33	36.49	76.55	122.6
10,000	22.87	44.01	84.88	44.98	97.83	170.3	66.72	130.8	257.4	122.3	263.2	470.7
解評価回数	JADE	ours (F)	ours (E)	JADE	ours (F)	ours (E)	JADE	ours (F)	ours (E)	JADE	ours (F)	ours (E)
500	0.809	1.219	1.973	1.632	2.396	3.554	2.453	3.552	5.161	4.384	6.489	9.227
1,000	1.683	3.011	4.782	3.342	5.989	8.764	5.011	8.484	12.78	9.075	16.59	22.77
3,000	5.023	11.19	16.12	10.14	18.93	30.08	15.29	25.49	44.11	27.82	53.57	79.15
10,000	16.70	35.71	58.03	33.93	56.53	111.4	51.05	81.14	166.4	93.54	180.7	306.2

が事後検証結果として与えられることになるため，提案手法の事前検証と対になって効率的に働いていることが確認できる．しかしながら，表 7 の検定結果では，ours (F) と ours (E) はどちらも SaDE にすべて統計的に優位であるため，両手法のどちらを用いても性能を向上する．

JADE では，次元数 $D = 100$ かつ解評価回数 10,000 をに設定した場合を除いたすべての実験ケースにおいて，ours (F) と ours (E) のどちらも性能を向上している．なお，表 7 より，次元数 $D = 100$ かつ解評価回数 10,000 の実験ケースにおいて有意差が検出された組はないことから，JADE と ours(F)，ours(E) は競合する性能を導出している．また，表 6 の網掛けした部分を見ると，次元数が $D = 10$ と低いときには ours(F) が高い順位を導出しているのに対し，次元数が増加すると徐々に ours(E) の順位が次第に改善する．これより，優良解の位置情報を利用する JADE に関しては，高次元により問題の難易度が上がった場合，すべての個体について事前検証と調整を行う方が高い性能を導出することが分かる．

次に，ours (E) から ours (F) に変更することによる計算時間の削減効果を確認する．具体的には，**表 8** に示すよ

表 9　jDE に提案手法を適用した際の基準個体の選択戦略による平均順位比較
Table 9　Comparison of the mean rankings of criteria individual selection strategies for jDE with the proposed method.

D	10				30				50				100			
評価回数	*rand*	*greedy*	*p-best*	*ϵ-greedy*	*rand*	*greedy*	*p-best*	*ϵ-greedy*	*rand*	*greedy*	*p-best*	*ϵ-greedy*	*rand*	*greedy*	*p-best*	*ϵ-greedy*
500	2.393	3.000	2.429	2.179	2.875	2.411	2.411	2.304	2.661	2.429	2.554	2.357	2.661	2.554	2.482	2.304
1,000	2.571	2.500	2.393	2.536	2.964	2.036	2.714	2.286	2.304	2.429	2.482	2.786	2.518	2.482	2.554	2.446
3,000	2.929	2.214	2.786	2.071	2.964	2.357	2.679	2.000	3.000	2.071	2.429	2.500	2.875	2.268	2.946	1.911
10,000	3.143	2.036	2.857	1.964	3.393	1.714	2.821	2.071	3.500	1.857	2.607	2.036	3.196	1.732	2.804	2.268

表 10　jDE に提案手法を適用した際の基準個体の選択戦略による比較における有意差検出組
Table 10　Significant difference detection pairs in comparison between criteria individual selection strategies for jDE with the proposed method.

評価回数	$D = 10$	$D = 30$	$D = 50$	$D = 100$
500	e - g	–	–	–
1,000	–	g - r	–	–
3,000	–	e - r	g - r	–
10,000	e - r	e - p, e - r, g - p, g - r	e - r, g - r, p - r	g - p, g - r

辞書順に記載，r: *rand*, g: *greedy*, p: *p-best*, e: *ϵ-greedy*

うに，intel(R) Core(TM) i7-9700（3.0 GHz）の CPU における全 28 関数の平均計算時間を次元数 $D = 10, 30, 50, 100$ ごとに比較する．

表 8 より，提案手法の実行時間は長くなる傾向にある．ours (F) が次元数 $D = 10$ で解評価回数 500，1,000 で逆転している原因としては，ours (F) が解更新失敗時のみ調整を行っているため，一定確率でこれを行う jDE よりも短時間になっていることが考えられる．つまり，解更新に成功したときは前世代のアルゴリズム構成が問題や探索状況に相応しいとし，これを引き継ぐこととすればアルゴリズム構成の事前検証と更新の時間が省かれる．

また，計算時間についても次元数ごとに有意水準 0.05 の下で Friedman 検定を行い，同様の事後検定を行った．検定結果としては，表 8 に掲載したすべての解評価回数において全組で有意差が検出され，表 8 と同じ順位が得られた．したがって，ours (F) と ours (E) を比べたときは実行時間は削減されていることが分かる．一方で，比較的実行時間が短い ours (F) も jDE や SaDE，JADE に比べれば約 2 倍の計算時間がかかっている．しかしながら，提案手法は限られた解評価回数で性能を向上しており，解評価に時間のかかる高計算コストな最適化問題を解く際には，この問題点は緩和されると考えられる．

6.3　基準個体の選択戦略

4.2.2 項において，アルゴリズム構成の事前検証をする際の基準個体 $\boldsymbol{x}_i^*$ は現在の解集合 $\mathcal{P}$ における最良個体とした（*greedy* 戦略）．ここでは，基準個体の選択戦略が提案手法の最適化性能にどのような影響を与えるかを jDE を例に考察する．今回比較する基準個体の選択戦略は 4.2.2 項に示した 4 つである．その他の実験条件は 5 章と同じとし，*p-best* 戦略や *ϵ-greedy* 戦略のパラメータは $p = \epsilon = 0.2$ とする．得られた性能に対しては Friedman 検定を行い，これまでと同様の事後検定を行う．

表 9 に平均順位，**表 10** に Holm 法適用後に有意水準 0.05 で有意差が見られた組を示す．なお，提案手法はすべて統計的有意差をもって jDE より高い平均順位を導出していることから，jDE は表から外した．表 9 について，*rand* 戦略でありながら，$D = 50$，解評価回数 1,000 で高い性能を導出している．この理由として，DE のアルゴリズム自体が非決定的に子個体を生成し，好適な個体が生成されればそれを次世代の親個体とするため，一定確率で高い性能を出す可能性があることが考えられる．全体的には，現在の最良個体の周辺に解生成する能力を持たせる *greedy* 戦略が高い性能を導出する．一方で，*ϵ-greedy* 戦略の順位も高いことから，多峰性関数などでは多様な個体を基準として，個体だけでなくアルゴリズム構成の多様性を担保することも重要になると推測できる．しかしながら，表 10 にも見られるように多くの実験ケースで有意差が見られず，有意差がある組では *greedy* 戦略や *ϵ-greedy* 戦略が優位である．ここで，*p-best* 戦略や *ϵ-greedy* 戦略と *greedy* 戦略を比較したとき，*p-best* 戦略や *ϵ-greedy* 戦略では提案手法のハイパーパラメータ数が 1 つ増加することになる．したがって，jDE からの性能向上という点では基本的にはどの基準個体選択戦略を使用しても大きな差異はないものの，ハイパーパラメータ数の観点からは，*greedy* 戦略が *ϵ-greedy* 戦略よりも良い選択肢になりうる．

7. 結論

本論文は，「アルゴリズム構成をいかに生成すべきか」よりも「生成されたアルゴリズム構成のうちどれを利用すべきか」という観点に着目した．具体的には，自己適応型差分進化法（DE）のアルゴリズム構成を事前検証する，一般的な自己適応型 DE に適用可能なフレームワークを提案した．提案手法は実際に解を生成する前に，自己適応型 DE の方法に則りアルゴリズム構成候補を複数生成する．続いて，優良解への局所探索のバイアスを与えるアルゴリズム構成を 1 つ選択した後に，これを用いて実際の解生成を行う．実験では，提案手法を自己適応型 DE の代表手法である jDE と SaDE，JADE に組み込むことで，特に 500 から 1,000 の解評価回数において，次元数に対するスケーラビリティを実現しつつ最適化性能を向上することを示した．したがって，従来の自己適応型 DE が試行錯誤的調整を行うために解評価回数を大量に消費していたのに対し，提案手法は試行錯誤的な調整要素を緩和して解評価回数を削減できる．これは，自己適応型 DE が不得意とする高計算コストな最適化問題のような問題に対しても，提案手法が展開できる可能性を示すものである．

今後の展望としては，提案手法で新たに導入されたハイパーパラメータ C に関する実験的な検証を行う．また，実応用への展開を図るべく，提案フレームワークを多目的最適化問題や制約付き最適化問題に拡張するとともに，サロゲートを導入してさらに解評価回数を削減することを目指す．

謝辞　本研究は JSPS 科研費 20H04254 の助成を受けた．

参考文献

[1] Nemec, M., Zingg, D.W. and Pulliam, T.H.: Multipoint and multi-objective aerodynamic shape optimization, *AIAA Journal*, Vol.42, No.6, pp.1057–1065 (2004).

[2] Karafotias, G., Hoogendoorn, M. and Eiben, Á.E.: Parameter control in evolutionary algorithms: Trends and challenges, *IEEE Trans. Evolutionary Computation*, Vol.19, No.2, pp.167–187 (2014).

[3] Huang, C., Li, Y. and Yao, X.: A Survey of Automatic Parameter Tuning Methods for Metaheuristics, *IEEE Trans. Evolutionary Computation*, Vol.24, No.2, pp.201–216 (2020).

[4] Smith, J.E.: Self-adaptation in evolutionary algorithms for combinatorial optimisation, *Adaptive and Multilevel Metaheuristics*, pp.31–57, Springer (2008).

[5] Lobo, F., Lima, C.F. and Michalewicz, Z.: Parameter setting in evolutionary algorithms, Vol.54, *Springer Science & Business Media* (2007).

[6] Smith, J.: Self adaptation in evolutionary algorithms, PhD Thesis (2020).

[7] Storn, R. and Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, Vol.11, No.4, pp.341–359 (1997).

[8] Price, K., Storn, R.M. and Lampinen, J.A.: *Differential evolution: A practical approach to global optimization*, Springer Science & Business Media (2006).

[9] Das, S. and Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art, *IEEE Trans. Evolutionary Computation*, Vol.15, No.1, pp.4–31 (2010).

[10] Neri, F. and Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis, *Artificial Intelligence Review*, Vol.33, No.1-2, pp.61–106 (2010).

[11] Gämperle, R., Müller, S.D. and Koumoutsakos, P.: A parameter study for differential evolution, *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, Vol.10, No.10, pp.293–298 (2002).

[12] Mezura-Montes, E., Velázquez-Reyes, J. and Coello Coello, C.A.: A comparative study of differential evolution variants for global optimization, *Proc. 8th Annual Conference on Genetic and Evolutionary Computation*, pp.485–492 (2006).

[13] Sabar, N.R. and Ayob, M.: Examination timetabling using scatter search hyper-heuristic, *2009 2nd Conference on Data Mining and Optimization*, pp.127–131, IEEE (2009).

[14] Hamza, N.M., Essam, D.L. and Sarker, R.A.: Constraint consensus mutation-based differential evolution for constrained optimization, *IEEE Trans. Evolutionary Computation*, Vol.20, No.3, pp.447–459 (2015).

[15] Lynn, N., Mallipeddi, R. and Suganthan, P.N.: Differential Evolution with Two Subpopulations, *International Conference on Swarm, Evolutionary, and Memetic Computing*, pp.1–13, Springer (2014).

[16] Li, G., Lin, Q., Cui, L., Du, Z., Liang, Z., Chen, J., Lu, N. and Ming, Z.: A novel hybrid differential evolution algorithm with modified CoDE and JADE, *Applied Soft Computing*, Vol.47, pp.577–599 (2016).

[17] Wu, G., Shen, X., Li, H., Chen, H., Lin, A. and Suganthan, P.N.: Ensemble of differential evolution variants, *Information Sciences*, Vol.423, pp.172–186 (2018).

[18] Gao, W., Yen, G.G. and Liu, S.: A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, *IEEE Trans. Cybernetics*, Vol.44, No.8, pp.1314–1327 (2013).

[19] Wang, Z.-J., Zhan, Z.-H., Lin, Y., Yu, W.-J., Wang, H., Kwong, S. and Zhang, J.: Automatic niching differential evolution with contour prediction approach for multimodal optimization problems, *IEEE Trans. Evolutionary Computation* (2019).

[20] Lu, X., Tang, K., Sendhoff, B. and Yao, X.: A new self-adaptation scheme for differential evolution, *Neurocomputing*, Vol.146, pp.2–16 (2014).

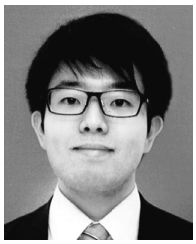
[21] Sharma, M., Komninos, A., Ibanez, M.L. and Kazakov, D.: Deep Reinforcement Learning Based Parameter Control in Differential Evolution, arXiv preprint arXiv:1905.08006 (2019).

[22] Brest, J., Greiner, S., Boskovic, B., Mernik, M. and Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *IEEE Trans. Evolutionary Computation*, Vol.10, No.6, pp.646–657 (2006).

[23] Zhang, J. and Sanderson, A.C.: JADE: Adaptive differential evolution with optional external archive, *IEEE Trans. Evolutionary Computation*, Vol.13, No.5, pp.945–958 (2009).

[24] Qin, A.K. and Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization, *2005 IEEE Congress on Evolutionary Computation*, Vol.2,

pp.1785–1791, IEEE (2005).
[25] Qin, A.K., Huang, V.L. and Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Trans. Evolutionary Computation*, Vol.13, No.2, pp.398–417 (2008).
[26] Tanabe, R. and Fukunaga, A.: Success-history based parameter adaptation for differential evolution, *2013 IEEE Congress on Evolutionary Computation*, pp.71–78, IEEE (2013).
[27] Brest, J., Maučec, M.S. and Bošković, B.: Single objective real-parameter optimization: Algorithm jSO, *2017 IEEE Congress on Evolutionary Computation* (*CEC*), pp.1311–1318, IEEE (2017).
[28] Yang, Z., Tang, K. and Yao, X.: Self-adaptive differential evolution with neighborhood search, *2008 IEEE Congress on Evolutionary Computation* (*IEEE World Congress on Computational Intelligence*), pp.1110–1116, IEEE (2008).
[29] Das, S., Mullick, S.S. and Suganthan, P.N.: Recent advances in differential evolution – An updated survey, *Swarm and Evolutionary Computation*, Vol.27, pp.1–30 (2016).
[30] Liang, J., Qu, B., Suganthan, P. and Hernández-Díaz, A.G.: Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, Vol.201212, No.34, pp.281–295 (2013).
[31] Kennedy, J. and Eberhart, R.: Particle swarm optimization, *Proc. IEEE International Conference on Neural Networks*, Vol.4, pp.1942–1948, Citeseer (1995).
[32] Regis, R.G.: Particle swarm with radial basis function surrogates for expensive black-box optimization, *Journal of Computational Science*, Vol.5, No.1, pp.12–23 (2014).
[33] Nishihara, K. and Nakata, M.: Competitive-Adaptive Algorithm-Tuning of Metaheuristics inspired by the Equilibrium Theory: A Case Study, *2020 IEEE Congress on Evolutionary Computation* (*CEC*), pp.1–8, IEEE (2020).
[34] Tanabe, R. and Fukunaga, A.: Reviewing and benchmarking parameter control methods in differential evolution, *IEEE Trans. Cybernetics*, Vol.50, No.3, pp.1170–1184 (2019).
[35] Das, S., Konar, A. and Chakraborty, U.K.: Two improved differential evolution schemes for faster global search, *Proc. 7th Annual Conference on Genetic and Evolutionary Computation*, pp.991–998 (2005).
[36] Draa, A., Bouzoubia, S. and Boukhalfa, I.: A sinusoidal differential evolution algorithm for numerical optimisation, *Applied Soft Computing*, Vol.27, pp.99–126 (2015).
[37] Mallipeddi, R., Suganthan, P.N., Pan, Q.-K. and Tasgetiren, M.F.: Differential evolution algorithm with ensemble of parameters and mutation strategies, *Applied Soft Computing*, Vol.11, No.2, pp.1679–1696 (2011).
[38] Shi, Y. and Eberhart, R.C.: Empirical study of particle swarm optimization, *Proc. 1999 Congress on Evolutionary Computation-CEC99* (*Cat. No. 99TH8406*), Vol.3, pp.1945–1950, IEEE (1999).
[39] Liu, J. and Lampinen, J.: A fuzzy adaptive differential evolution algorithm, *Soft Computing*, Vol.9, No.6, pp.448–462 (2005).
[40] Kaelo, P. and Ali, M.: A numerical study of some modified differential evolution algorithms, *European Journal of Operational Research*, Vol.169, No.3, pp.1176–1184 (2006).
[41] Tirronen, V. and Neri, F.: Differential evolution with fitness diversity self-adaptation, *Nature-inspired Algorithms for Optimisation*, pp.199–234, Springer (2009).
[42] Jia, L., Gong, W. and Wu, H.: An improved self-adaptive control parameter of differential evolution for global optimization, *International Symposium on Intelligence Computation and Applications*, pp.215–224, Springer (2009).
[43] Mallipeddi, R. and Suganthan, P.N.: Ensemble of constraint handling techniques, *IEEE Trans. Evolutionary Computation*, Vol.14, No.4, pp.561–579 (2010).
[44] Tanabe, R. and Fukunaga, A.S.: Improving the search performance of SHADE using linear population size reduction, *2014 IEEE Congress on Evolutionary Computation* (*CEC*), pp.1658–1665, IEEE (2014).
[45] Awad, N.H., Ali, M.Z., Suganthan, P.N. and Reynolds, R.G.: An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems, *2016 IEEE Congress on Evolutionary Computation* (*CEC*), pp.2958–2965, IEEE (2016).
[46] Brest, J., Maučec, M.S. and Bošković, B.: iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization, *2016 IEEE Congress on Evolutionary Computation* (*CEC*), pp.1188–1195, IEEE (2016).



西原 慧　（学生会員）

1996 年生．2020 年横浜国立大学理工学部数物・電子情報系学科卒業．現在，同大学大学院修士課程．進化計算の研究に従事．学士（工学）．情報処理学会，電気学会，IEEE 各学生会員．

中田 雅也

1988 年生．2015 年電気通信大学大学院情報理工学研究科博士課程修了．2016 年横浜国立大学助教．2019 年より同大学准教授，現在に至る．進化計算，進化的機械学習の研究に従事．博士（工学）．IEEE CIS Japan Chapter Young Researcher Award，システム制御情報学会奨励賞等を受賞．ACM GECCO 2018 Local Program Chair，第 13 回進化計算学会研究会実行委員長．IEEE，進化計算学会各会員．