

卒業論文

勾配情報を考慮した粒子群最適化による 制約付き多目的最適化問題の解の探索

Searching Solutions for Constrained Multi-Objective Optimization
Problems Using Particle Swarm Optimization with gradient
information

富山県立大学 工学部 情報システム工学科

2120019 柴原壮大

指導教員 奥原 浩之 教授

提出年月: 2025年2月

目次

図一覧	ii
表一覧	iii
記号一覧	iv
第1章 はじめに	1
§ 1.1 本研究の背景	1
§ 1.2 本研究の目的	2
§ 1.3 本論文の概要	3
第2章 多目的最適化問題と解法	4
§ 2.1 多目的最適化のパレート解	4
§ 2.2 実践手法とベンチマークの関係	8
§ 2.3 種々のパレート解の導出法	12
第3章 多目的最適化の粒子群最適化	16
§ 3.1 粒子群最適化アルゴリズム	16
§ 3.2 制約がある場合の粒子群最適化	18
§ 3.3 多目的粒子群最適化	21
第4章 提案手法	25
§ 4.1 勾配を考慮した粒子群最適化 (1 目的)	25
§ 4.2 上下制約付き勾配 MOPSO	28
§ 4.3 多目的における勾配の決定法	30
第5章 数値実験並びに考察	34
§ 5.1 数値実験の概要	34
§ 5.2 実験結果と考察	37
第6章 おわりに	39
謝辞	40
参考文献	41

圖一覽

表一覽

記号一覧

以下に本論文において用いられる用語と記号の対応表を示す.

用語	記号
個体	p
ステップ数	m
粒子群最適化における k 番目の粒子の位置	x_p^k
粒子群最適化における k 番目の粒子の速度	v_p^k
粒子群最適化におけるパーソナルベスト	x_{pbest}
粒子群最適化におけるグローバルベスト	x_{gbest}
0～1 のランダム変数	r_1, r_2
粒子群最適化における速度の慣性係数	w
パーソナルベストに向かう力に対する重み	c_1
グローバルベストに向かう力に対する重み	c_2
勾配情報に対する重み	c_3
時刻 k までにおいて, 個体 p が関数 E の値を最も小さくした時刻	$l_p(k)$
時刻 k までにおいて関数 E の値を最も小さくした個体番号とその時刻	$(Q(k), l_o(k))$
個体 p の内部情報量	u_p, v_p
サンプリングパラメータ	ΔT
制約条件	p_i, q_i
勾配情報	∇E

はじめに

§ 1.1 本研究の背景

近年、コンピュータサイエンスの発展は、ハードウェアとソフトウェアの進化により、さまざまな分野で顕著な成果を上げている。特に、処理能力の向上やアルゴリズムの革新によって、大規模な最適化問題に対するアプローチが進展し、実世界での適用範囲が拡大している。これにより、かつて解決不可能とされていたような難解な問題にもアプローチできるようになり、複雑で高次元な問題を効率的に解く技術の需要が急速に高まっている。

中でも、大規模なデータ処理や最適化問題は、企業や研究機関において重要な課題となっており、解決策としての最適化技術の有効性が強調されている。たとえば、ソーシャルネットワークサービス（SNS）の普及により、膨大な量のログデータや通信パスの解析が求められる。これらのデータの中から有用な情報を抽出するためには、非常に高度で効率的な最適化技術が必要とされる。さらに、IoT（モノのインターネット）やビッグデータの普及により、処理対象となるデータ量やネットワーク規模が爆発的に増加しており、従来のアルゴリズムでは計算時間やメモリの制約が大きな課題となることが多い。

また、科学技術計算やエンジニアリング分野においても、大規模なシミュレーションやデータ分析が求められるシーンが増えており、これらに対する効率的な最適化手法の開発は、ますます重要となっている。特に、エネルギー消費の最適化や製造業における生産計画、交通システムの最適化など、実際の現場における最適化問題では、多くの制約条件や競合する目標が存在し、解法の選定が非常に難しくなっている。

最新のコンピュータ技術においても、これらの問題を解決するためには膨大な計算リソースと時間が必要であり、従来のアルゴリズムでは計算時間の長期化や収束性の低下が問題となる場合が多い。特に、これらの問題は大量なデータセットや高次元な問題に対して特に深刻であり、計算量の増加と共に性能が急激に悪化し、現実的な時間内に解を得ることが困難になる場合が頻発する。したがって、計算資源を効率的に活用し、短時間で最適解を見つけるための新たな手法が求められている。

このような背景を踏まえ、複雑で高次元な最適化問題を解決するために、粒子群最適化（PSO）などの群知能に基づくアルゴリズムが注目されており、其有効性は多くの実際の問題において証明されている。しかし、PSOのような進化的手法には、局所的最適解に陥るリスクや、計算の収束速度が遅くなる課題が依然として存在するため、これらの問題を改善するためのさらなる研究が必要とされている。

§ 1.2 本研究の目的

近年、複雑な最適化問題の解決において、従来の最適化手法では解決が難しい新たな課題が浮き彫りになっている。特に、実世界の問題では、複数の目的が相反する関係にあり、これらを同時に最適化する必要が生じる場合が多い。例えば、製造業や物流業におけるコストと効率のバランス、エネルギー管理における環境負荷とコストの折り合いなど、複数の目標を同時に達成することが求められる。このような問題は多目的最適化問題として広く認識されており、その解決には高度で洗練された技術が必要不可欠である。また、従来の最適化手法ではこれらの問題に対応する際、計算コストや時間の面で限界があり、効率的な解法を見つけることがますます困難となっている。

さらに、実際の最適化問題では、単に目標関数を最適化するだけでなく、複雑な制約条件が加わることが一般的である。例えば、製造工程や設計問題においては、リソースの制約、時間制限、安全基準や環境規制といった要素が絡み合い、これらを考慮した上で最適解を求める必要がある。こうした制約条件を満たしつつ最適解を導出することは、単に最適化手法を適用するだけでは実現が難しく、より高度で専門的なアプローチが必要とされる。

本研究の目的は、こうした多目的最適化問題において、数ステップで効率的に最適解を求めることができる新しいハイブリッド動的システムを提案することである。このシステムは、多目的粒子群最適化 (MOPSO) アルゴリズムに勾配情報を統合することにより、探索の精度を高め、収束速度を改善することを目指している。多目的粒子群最適化は、従来より多目的最適化や制約付き最適化問題に有用な手法として広く用いられてきたが、その探索能力や制約条件の処理においては依然として課題が残されている。特に、解が局所的な最適解に収束しやすいという欠点を有しており、また粒子数や世代数が増えることで計算コストが増大するという問題が存在する。この問題を解決するために、本研究では、粒子群最適化に勾配情報を組み込むことにより、粒子が最適解に向かって効率的に移動できるようにし、局所解への陥落を防ぎつつ、より迅速に解を収束させることを目指している。

また、多くの実世界の最適化問題では、目標関数の最適化に加え、さまざまな制約条件が課せられる。例えば、製造業における工程最適化やエネルギー管理における最適化問題では、コストや時間、リソースの使用量、環境負荷など、さまざまな要素を同時に最適化する必要がある。そこで、PSO は遺伝的アルゴリズムと同様に、制約条件を明示的に扱わない手法であるため、機械システムの多目的最適設計問題のような制約を有する問題を効率的に解くための手法が提案されている [1-3]。その初期には、いったん元の位置に戻してやりなおす方法 [4] や制約を逸脱した場合に速度の慣性項をゼロにする操作 [5] が提案された。これらは単純で実装が容易であるが、探索が乱数に依存することになるため、探索効率が悪くなる問題がある。他の手法として、ペナルティ関数が広く用いられている [6]。しかしながら、ペナルティパラメータによっては探索効率が著しく悪化するなど、パラメータ選択に対する問題がある [7]。そこで、制約条件を積極的に取り扱うために、上下限制約領域内に問題の変数を変換して無制約化した新たな変数空間に無制約 PSO モデルを適用した「変数変換モデル」を導入する。

つまりここでは、数ステップでもっとも最適な解が見つかる新しいハイブリッド動的システムを実際の多目的最適化問題に適応する。つまり上下限制約条件付き最適化問題に対し直接適用可能な勾配情報を追加した粒子群最適化アルゴリズムを活用し、それを拡張することで、多目的最適化問題へ組み込みを行う。

§ 1.3 本論文の概要

本論文は次のように構成される.

第1章 本研究の背景と目的について説明する. 背景では, 昨今における, 複雑で高次元な問題を効率的に解く技術の需要について説明している. 目的では, 複雑な最適化問題の解決に向けた最適化手法について述べている.

第2章 多目的最適化問題とパレート解, 実践手法とベンチマークについて説明する.

第3章 粒子群最適化と制約を考慮した粒子群最適化について説明する. また, 多目的粒子群最適化について説明する.

第4章 提案手法として, 勾配情報を考慮した制約付き多目的粒子群最適化について説明する.

第5章 提案手法を, ベンチマーク問題に適応する. そして, 従来手法と比較して本研究の提案手法によって得られた結果が有意であることを説明する.

第6章 本論文における前章までの内容をまとめつつ, 本研究で実現できたことと今後の展望について述べる.

多目的最適化問題と解法

§ 2.1 多目的最適化のパレート解

近年、多目的ダムや多目的ホールなど、“多目的”という言葉がいろいろと使われるようになってきた。これは一つのシステムあるいは機構をいくつかの異なった目的に対して機能させるという意味であり、この言葉が用いられる背景には、社会的ニーズの多様化および資源の効率的利用などの要求があると考えられる。すなわち、単一の目的関数を用いた最適化問題でカバーできないような多目的最適化問題の重要性が高まってきているのである。

数理計画法や最適制御理論は広い意味で効率の良さを求めるための数学的手段であるが、ほとんどの場合、解空間内で一つの目的関数を最小（または最大）にする最適点を決定することが論じられている。多目的最適化問題に対しても、二つ以上の目的関数の和をとって一つの目的関数にする重みパラメータ法や、一つの目的関数のみを残し、これ以外の目的関数を制約条件に置き換える ϵ 制約法など、従来の数理的手法を応用しようとする方法論がいくつか提案されている [8~11]。多目的最適化の本質は複数の目的関数間でのトレードオフをいかにバランスさせるかという点にあり、従来の方法論では不十分である場合が多い。

一方、多目的最適化の理論について見ると、目的関数間でのトレードオフをバランスさせ得る解に関連して、“パレート最適性”が重要な概念として挙げられている [9]。一般に、このパレート最適性を満足する解（パレート最適解）は複数個あり、いかにして二つの解（選好解）を選び出すかが問題となる。従来、この選好解を決定（多目的意思決定）する過程においては、パラメータを対話的に変更しながら、重みパラメータ法や ϵ 制約法を繰り返し適用する SWT（Surrogate Worth Trade-off）法などの対話的手法が提案されている [8]。また、意思決定者の“あいまいな”選好をファジィ（fuzzy）理論を用いて表現し、数理的な手法を適用しようとする報告もなされている [12-13]。しかしながら、パレート最適解のような解候補をあらかじめ集合として求め、これらを提示することができれば、より効率的かつ適切に多目的意思決定を行うことが期待される。ここで、多目的最適化問題とパレート解について解説する

多目的最適化問題

問題 A（多目的最適化問題）： F を \mathbb{R}^n の閉集合とする

$$F \triangleq \{x \mid g_i(x) \leq 0, i = 1, \dots, m\} \quad (2.1)$$

とすると、 $x \in \mathbb{R}^n$ の p 個の関数

$$f_k(x), \quad k = 1, \dots, p \quad (2.2)$$

を、 $x \in F$ の範囲でなるべく小さくせよ.

この多目的最適化問題では、一般にすべての目的関数 $f_k(x), k = 1, \dots, p$ を同時に最小にすることはできない. むしろ、これらの間にトレードオフの関係があることが問題の本質である. したがって、目的関数間での協調を図って各目的関数をできるだけ小さくするようにしなければならない.

パレート最適解

まず、 p 次元定数ベクトル a, b 間の不等関係を、

$$a \leq b \iff a_i \leq b_i \quad (i = 1, \dots, p) \quad (2.3)$$

または

$$a < b \iff a_i < b_i \quad (i = 1, \dots, p) \quad (2.4)$$

で定義する. ただし、 a_i および b_i は、それぞれ a および b の第 i 要素である. ここで、

$$f(x) \triangleq (f_1(x), \dots, f_p(x)) \quad (2.5)$$

とすると、多目的最適化問題における解の優越関係は次のように定義される.

● 定義 1

$x_1, x_2 \in F$ とする.

- $f(x_1) \leq f(x_2)$ のとき、 x_1 は x_2 に優越すると言う.
- $f(x_1) < f(x_2)$ のとき、 x_1 は x_2 に強い意味で優越すると言う.

もし x_1 が x_2 に優越しているならば、 x_1 のほうが x_2 より良い解である. したがって、ほかのいかなる解にも優越されない解を選ぶことが合理的な方法であると言える.

● 定義 2 (パレート最適解)

$x_0 \in F$ とする.

- x_0 に強い意味で優越する $x \in F$ が存在しないとき、 x_0 を弱パレート最適解という.
- x_0 に優越する $x \in F$ が存在しないとき、 x_0 を (強) パレート最適解という.
- 任意の $x \in F$ について $f(x) > f(x_0)$ が成り立つとき、 x_0 を (完全) 最適解という.

目的関数が二つ ($p = 2$) の場合のパレート最適解の例と優越関係の例を図に示す. 図中、太線がパレート最適解を、太い破線が弱パレート最適解をそれぞれ示している.

定義により、最適解があればそれはパレート最適解であり、最適解が存在する時は、それ以外のパレート最適解は存在しない. したがって、パレート最適解は多目的最適化問題に対する最も合理的な解 (の集合) であると言える.

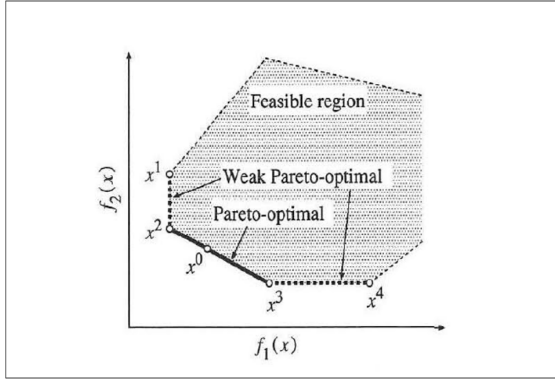


図1 パレート解

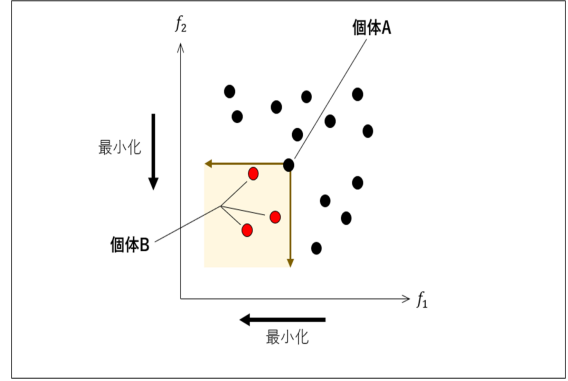


図2 優越関係

また、パレート最適解全体の集合は、一般に「パレート前線 (Pareto Front)」と呼ばれ、最適化問題において、すべての非劣解 (パレート最適解) がこの前線上に位置する。パレート前線は、最適化空間内での複数の目的関数のトレードオフを示す重要な概念であり、各点の一つの最適解を表す。

代表的な多目的最適化問題として多目的ナップザック問題と多目的巡回セールスマン問題を説明する。

多目的ナップザック問題

多目的ナップザック問題は、要素の集合

$$E = \{e_1, e_2, \dots, e_N\}, \quad (2.6)$$

要素の体積の集合

$$V = \{v_1, v_2, \dots, v_N\}, \quad (2.7)$$

および要素の価値の集合

$$C^{(1)} = \{c_1^{(1)}, c_2^{(1)}, \dots, c_N^{(1)}\}, \dots, C^{(p)} = \{c_1^{(p)}, c_2^{(p)}, \dots, c_N^{(p)}\} \quad (2.8)$$

が与えられた場合、次の式 (2.9) および (2.10) を満たす要素の部分集合 $E' \subseteq E$ を求める問題として定式化される。

$$\text{maximize} \quad \begin{cases} f_1(x) = \sum_{i=1}^N c_1^{(i)} x_i \\ \vdots \\ f_p(x) = \sum_{i=1}^N c_p^{(i)} x_i \end{cases} \quad (2.9)$$

$$\text{subject to} \quad \sum_{i=1}^N v_i x_i \leq K \quad (2.10)$$

ここで、 K はナップザックの体積サイズであり、変数 x_i は次のように定義される：

$$x_i = \begin{cases} 1 & \text{if } e_i \in E' \\ 0 & \text{if } e_i \notin E' \end{cases} \quad (2.11)$$

図 3 に，多目的ナップザック問題を簡略に説明したものを示す．

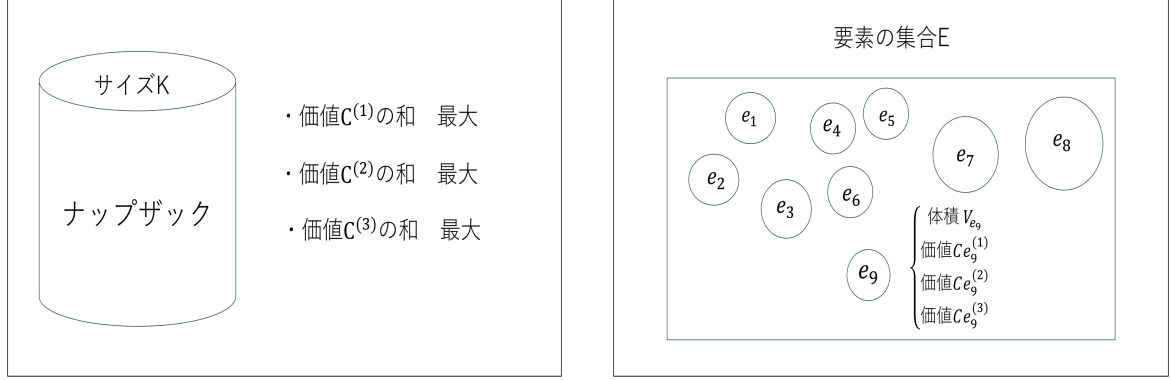


図 3 多目的ナップザック問題

図 4 を用いて多目的ナップザック問題の説明を行う．同図の円柱はナップザックで，その中に詰め込むことのできる要素の総体積を表すサイズは， K で表される．また，円で示した a から i はナップザックに詰め込むべき要素を表しており，各要素には属性として体積と複数の価値が存在する．

多目的ナップザック問題はこれらの要素をある組み合わせでナップザックに詰め込むとき，その体積の和がサイズ K を越えないようにした上で，それぞれの価値の和が最大となるような要素の組み合わせを求める多目的最適化問題といえることができる．

従来よく知られている単目的ナップザック問題と異なる点は，1つの要素に対し2つ以上の価値が存在することである．

多目的巡回セールスマン問題

多目的巡回セールスマン問題は，都市の集合 $U = \{1, 2, \dots, N\}$ とし，都市 (i, j) 間のコストを $c_{ij}^{(1)}, c_{ij}^{(2)}, \dots, c_{ij}^{(p)}$ とすると， N 個の都市すべてをちょうど1度ずつ巡って出発した都市に戻ってくる巡回路の中から，通過した道のコストの総和を最小にする問題とし，以下のように定式化される．

$$\text{minimize} \quad \begin{cases} f_1(x) = \sum_{i \in U} \sum_{j \in U} c_{ij}^{(1)} x_{ij} \\ f_2(x) = \sum_{i \in U} \sum_{j \in U} c_{ij}^{(2)} x_{ij} \\ \vdots \\ f_p(x) = \sum_{i \in U} \sum_{j \in U} c_{ij}^{(p)} x_{ij} \end{cases} \quad (2.12)$$

$$\text{subject to } \begin{cases} \sum_{j \in U} x_{ij} = 1, & \forall i \in U \\ \sum_{i \in U} x_{ij} = 1, & \forall j \in U \\ \sum_{j \in U \setminus V} x_{ij} \geq 1, & \forall V \subset U, \quad V \neq \emptyset, V \neq U \\ x_{ij} \in \{0, 1\}, & \forall i, j \in U \end{cases} \quad (2.13)$$

ここで、 $c_{ii}^{(q)} = +\infty$ ($q = 1, 2, \dots, p$) とした。



図4 多目的巡回セールス問題

図5に多目的巡回セールスマン問題を簡略に説明したものを示す．同図において円で示す都市 A から E は都市 2 を表しており，任意の 2 都市間には複数のコストが存在するものとする．

多目的巡回セールスマン問題は，ある都市を出発し，すべての都市を 1 回ずつまわって最初の都市に戻ってくるような巡回路のうち，その巡回路の複数のコストのそれぞれが最小となるような巡回路を求める多目的最適化問題であるといえることができる．

同図では，都市 A を出発して再び都市 A に戻ってくるような巡回路の例を示した．従来よく知られている巡回セールスマン問題と異なる点は，任意の 2 都市間に 2 つ以上のコストが定義されていることである．

§ 2.2 実践手法とベンチマークの関係

これまで，工学，医療，経済など多くの実問題において多目的最適化アルゴリズムが優れた性能を示すことが報告されている [14]．一方で，目的数が 4 以上の多数目的最適化問題 [15,16]，膨大な決定変数を扱う大規模最適化問題 [17]，複雑な多数の制約を持つ最適化問題 [18-22] といった従来の多目的最適化アルゴリズムが苦手とする問題が存在することも報告されている．そのため，近年の進化計算の分野では，このような困難な特徴を含む実問題を想定した多目的最適化アルゴリズムの開発が高い関心を集めており，多数目的最適化問題を指向した NSGA-III[23] や大規模最適化問題を指向した WOF[24] のように，特定の問題領域において高い性能を発揮するアルゴリズムが提案されている．これらの多目的最適化アルゴリズムの性能評価で主に用いられるのがベンチマーク問題である．特に，近年の激しい多目的最適化アルゴリズムの開発競争では，DTLZ[25] といった一部の有名なベンチマーク問題群による多目的最適化アルゴリズムの評価が繰り返し行われている．しか

し、現実世界には多様な多目的最適化問題が存在するにも拘らず、これらのベンチマーク問題自体の特徴調査はあまり行われていない。そこで本章では、1990年代から2010年代までに提案された既存のベンチマーク問題と既存の多目的最適化アルゴリズムについて解説する。

1990年代

古典的な多目的最適化アルゴリズムの共通の特徴は、パレート優越関係に基づく個体評価と多様性維持である。パレート優越関係では、以下のようなM目的最小化問題に対して、(3.9)の関係が成り立つとき解 \mathbf{a} は解 \mathbf{b} を優越すると定義される。

$$\text{Minimize } f_1(x), f_2(x), \dots, f_M(x) \quad \text{subject to } x \in X, \quad (2.14)$$

ここで、 $f(x)$ はM次元目的ベクトル、 $f_i(x)$ は最小化される i 番目の目的関数、 x は決定変数ベクトル、 X は決定変数空間内での実行可能領域を示す。

$$\forall i, \quad f_i(\mathbf{b}) \geq f_i(\mathbf{a}) \text{ and } \exists i, \quad f_i(\mathbf{b}) > f_i(\mathbf{a}) \quad (2.15)$$

すなわち、解 \mathbf{a} は、いずれの目的においても解 \mathbf{b} に劣っておらず、最低でも一つの目的について解 \mathbf{b} より優れている。このとき、解 \mathbf{a} は解 \mathbf{b} より明らかに優れている。一方で、二つの解の間に上記の関係が成立しない場合、それらは非劣とよばれる。

古典的な多目的最適化アルゴリズムでは、パレート優越関係によりパレートフロントへ解集合を収束させる。しかし、エリート保存戦略が使用されていないため、パレートフロントへの収束が安定しない。ここでは、1990年代に頻繁に使用されたベンチマーク問題の特性が、これらの非エリート保存戦略型 EMO アルゴリズムの性質と密接に関連していることを示す。2つの例を紹介する

• Schaffer

Schaffer のテスト問題は以下の二つの目的関数を最小化する問題である。

$$\text{Minimize } f_1(x) = x^2 \quad \text{and} \quad f_2(x) = (x - 2)^2, \quad (2.16)$$

$$\text{subject to} \quad -20 \leq x \leq 20. \quad (2.17)$$

このテスト問題は、単一の決定変数 x を用いて二つの目的関数の算出を行う単純なベンチマーク問題である。NPGA や NSGA の提案論文など、1990年代に提案された多目的最適化アルゴリズムの探索性能を評価するために頻繁に使用された。

Schaffer のテスト問題ではパレートフロントを発見するための強い収束性能は必要なく、パレートフロント上に多様な解を分布することが重要となる。このような要素は古典的な非エリート保存戦略型多目的最適化アルゴリズムの特性と一致する。

• F3

F3 は次のように定式化される制約付き 2 目的最小化問題である [26].

$$\text{Minimize } f_1(x_1, x_2) = x_1^2 - 2x_1 + x_2^2, \quad f_2(x_1, x_2) = 9x_1^2 - (x_1 - 1)^2, \quad (2.18)$$

$$\text{subject to } -20 \leq x_1 \leq 20, \quad -20 \leq x_2 \leq 20, \quad (2.19)$$

$$x_1^2 + x_2^2 \leq 25, \quad 3x_1 - 10 \leq 0, \quad x_1 - x_2 + 2 \leq 0, \quad (2.20)$$

2000 年代

2000 年頃に提案された SPEA や NSGA-II などのエリート保存戦略を採用した多目的最適化アルゴリズムは、古典的な多目的最適化アルゴリズムに比べ、パレートフロントへの高い収束性を持つことが知られている。これらのアルゴリズムの性能評価に用いられたベンチマーク問題にはパレートフロントへの収束が難しい問題が多く含まれている。ここで、これらのテスト問題の代表例として ZDT テストセットがある。

ZDT テストセットは ZDT1-ZDT6 の 6 種の 2 目的最小化問題から成るテストセットである。各テスト問題は凸状や不連続なパレートフロント、局所最適な疑似パレートフロントの存在といった異なる特徴を持つ。また、決定変数の数 N を自由に変更することが可能であるという特徴を持つ。NSGA-II の提案論文では、決定変数がバイナリ型の ZDT5 を除いた五つのテスト問題がアルゴリズムの性能評価に用いられた。ここでは、ZDT1, ZDT2 について調査を行う。それぞれの問題は以下のように定式化される。

[ZDT1]

Minimize

$$f_1(x) = x_1 \quad (2.21)$$

$$f_2(x) = g(x) \left(1 - \frac{x_1}{g(x)} \right) \quad (2.22)$$

where

$$g(x) = 1 + \frac{9}{N} \sum_{i=2}^N x_i, \quad (2.23)$$

subject to

$$0 \leq x_i \leq 1, \quad \text{for } i = 1, 2, \dots, N. \quad (2.24)$$

[ZDT2]

Minimize

$$f_1(x) = x_1 \quad (2.25)$$

$$f_2(x) = g(x) \left(1 - \frac{x_1}{g(x)} \right) \quad (2.26)$$

where

$$g(x) = 1 + \frac{9}{N} \sum_{i=2}^N x_i, \quad (2.27)$$

subject to

$$0 \leq x_i \leq 1, \quad \text{for } i = 1, 2, \dots, N. \quad (2.28)$$

2010 年代

近年，目的数が 4 以上の多数目的最適化問題で，パレート優越関係ベースの多目的最適化アルゴリズムの探索性能が低下することが報告されている．そのため，目的数を任意に指定可能なテスト問題フレームワークが頻繁にアルゴリズムの性能評価に用いられる．これらは目的数にスケーラブルなテスト問題と呼ばれる．ここでは，スケーラブルなテスト問題である DTLZ テストセットについて調査する．

DTLZ テストセット

DTLZ テストセットは，目的数を自由に指定可能な 7 個のテスト問題（DTLZ1-DTLZ7）によって構成される [?]. これらのテスト問題は，パレートフロントの形状と探索領域を別々に設計する *Bottom-Up* アプローチ と呼ばれる方法で作成される．ここでは，DTLZ1 を例として調査を行う．M 目的の DTLZ1 は以下のように定式化される．

$$\min \left\{ f_1(\mathbf{x}) = (1 + g(\mathbf{x})) \prod_{h=1}^{M-1} \cos(x_h) \prod_{h=M}^d \cos^2(x_h) \right\} \quad (2.29)$$

$$f_2(\mathbf{x}) = (1 + g(\mathbf{x})) \prod_{h=1}^{M-2} \cos(x_h) \prod_{h=M-1}^d \cos^2(x_h) \quad (2.29)$$

⋮

$$f_M(\mathbf{x}) = (1 + g(\mathbf{x})) \cos(x_1) \prod_{h=2}^M \cos^2(x_h) \quad (2.30)$$

ここで， $g(\mathbf{x})$ は以下のように定義される：

$$g(\mathbf{x}) = 100 \sum_{i=M}^d (x_i - 0.5)^2 - \prod_{i=M}^d \cos(20\pi(x_i - 0.5)) \quad (2.31)$$

制約条件：

$$0 \leq x_h \leq 1, \quad \text{for } h = 1, 2, \dots, M-1 \quad (2.32)$$

$$0 \leq x_j \leq 1, \quad \text{for } j = 1, 2, \dots, d \quad (2.33)$$

Bottom-Up アプローチ では，決定変数は位置変数および距離変数と呼ばれる二つのグループに分類される．ここでは，位置変数を x_h ，距離変数を x_d と表す．それぞれの決定変数の数は位置変数が $M-1$ ，距離変数が N_d となる．パレートフロントは位置変数を使用して設計され，探索領域は距離変数を用いて設計される．近年，DTLZ テストセットは，HypE, NSGA-III, MOEA/DD の提案論文や，多数目的最適化に関する調査などで，多目的最適化アルゴリズムの性能評価に頻繁に使用されている．

§ 2.3 種々のパレート解の導出法

パレート解の導出法の中から，いくつかの手法を取り上げ，その概要を紹介する．

重みパラメータ法

目的関数の加重和を最小にする問題は次のように表される：

$$\min_{\mathbf{x}} \sum_{k=1}^p \omega_k f_k(\mathbf{x}) \quad (2.34)$$

ここで， $\mathbf{x} \in \mathcal{F}$ であり，重みパラメータ ω_k は次の条件を満たす：

$$\omega_k \geq 0, \quad \sum_{k=1}^p \omega_k = 1 \quad (2.35)$$

この問題を解くと，もし $f_k(\mathbf{x})$ と $g(\mathbf{x})$ が凸関数であれば，パレート最適解の一つが得られる．

ϵ 制約法

ϵ 制約法では，一つの目的関数 $f_l(\mathbf{x})$ のみを最小化し，他の目的関数 $f_k(\mathbf{x})$ ($k \neq l$) を定数 ϵ_k で制約したスカラー最小化問題として定式化する．

$$\min_{\mathbf{x}} f_l(\mathbf{x}) \quad (2.36)$$

制約条件は次のように表される．

$$\mathbf{x} \in \mathcal{F} \cap \delta, \quad \delta = \{\mathbf{x} \mid f_k(\mathbf{x}) \leq \epsilon_k, k = 1, \dots, l-1, l+1, \dots, p\} \quad (2.37)$$

このようにして， ϵ_k を変化させることにより，パレート最適解の一つを求めることができる．一般に，パレート最適解は唯一解ではなく，集合（以下，パレート最適集合と呼ぶ）を形成する．

計算の効率化と遺伝アルゴリズムの利用

上述の重みパラメータ法および ε 制約法の方法では、パラメータを変えながら繰り返し問題を解くことにより、パレート最適集合を求めることができるが、これには膨大な計算時間が必要である。そこで、進化的アルゴリズムを適用することで、パレート最適集合をより効率的に求める方法を考える。進化的アルゴリズムでは、集団（解候補の集合）を用いて探索が進められるため、パレート最適集合を直接的かつ効率的に求めることが期待される。

進化的アルゴリズム

進化的アルゴリズムは、問題の特定の構造に依存せず、広範囲な探索を通じて近似解を得る方法として、最適化問題の解法に広く用いられている。ここでは、代表的な進化的アルゴリズムであるシミュレーティッド・アニーリング (Simulated Annealing: SA)、遺伝的アルゴリズム (Genetic Algorithm: GA)、免疫的アルゴリズム (Immune Algorithms: IA)、および粒子群最適化 (Particle Swarm Optimization: PSO) について解説する。

シミュレーティッド・アニーリング (Simulated Annealing: SA)

シミュレーティッド・アニーリング (SA) は、物理学におけるアニーリング（焼きなまし）の過程を模倣した最適化手法であり、エネルギーが最小となる状態（最適解）を探索することを目的としている。SA では、解候補間のエネルギー差に基づいて新しい解の受け入れ確率が決まる。具体的には、現在の解を x 、新しい解を x' 、それらのエネルギー差を $\Delta E = E(x') - E(x)$ としたとき、受け入れ確率 $P(\text{accept})$ は次のように表される。

$$P(\text{accept}) = \begin{cases} 1 & \text{if } \Delta E \leq 0, \\ \exp\left(-\frac{\Delta E}{T}\right) & \text{if } \Delta E > 0. \end{cases} \quad (2.38)$$

ここで、 T は温度を表し、温度は徐々に低下させる。温度の更新は以下のように行われる。

$$T_{k+1} = \alpha T_k, \quad (2.39)$$

ここで、 α は冷却率であり、 $0 < \alpha < 1$ の範囲にある。この冷却率により、温度の低下速度が決まる。

遺伝的アルゴリズム (Genetic Algorithm: GA)

遺伝的アルゴリズム (GA) は、自然選択と遺伝の原理を模倣した進化的アルゴリズムであり、個体群（解の集まり）を進化させることによって最適解を探索する。GA の主要な操作は、選択、交叉（交配）、および突然変異である。

選択

個体はその適応度 $f(x)$ に基づいて選択される。選択方法には、ルーレット選択やトーナメ

ント選択などが使用される。

交叉

交叉は、2つの親個体 x_1 と x_2 から新しい個体 x' を生成するために行われる。交叉操作は通常、遺伝子（解の各部分）の一部を交換する形式で実行される。例えば、一点交叉では次のように交叉が行われる。

$$x' = \text{crossover}(x_1, x_2). \quad (2.40)$$

交叉によって得られる子個体 x' は、親の情報を組み合わせた新しい解候補となる。

突然変異

突然変異は、個体の遺伝子をランダムに変更する操作である。この操作により、探索空間の多様性を保ちつつ、新たな解を探索することが可能となる。遺伝子 x_i を突然変異させる操作は次のように表される。

$$x'_i = \text{mutate}(x_i). \quad (2.41)$$

免疫的アルゴリズム (Immune Algorithms: IA)

免疫的アルゴリズム (IA) は、生物の免疫システムを模倣して、最適化問題を解く方法である。IA では、解を「抗体」として表現し、その適応度を評価することで最適解を探索する。

抗体と適応度

解は「抗体」と呼ばれ、その適応度は「抗体の品質」として評価される。抗体の適応度 $A(x)$ は、解 x の適応度関数 $f(x)$ に基づいて次のように定義される。

$$A(x) = f(x). \quad (2.42)$$

免疫の選択と記憶

免疫システムは「記憶細胞」を持ち、これにより良い解を記憶し、後の探索で再利用する。適応度が高い解は選ばれる確率が高くなり、最適解に向かって進化していく。

交叉と突然変異

免疫的アルゴリズムでは、GA と同様に交叉と突然変異が使用される。交叉は解の組み合わせを、突然変異は解の変化を導き、探索の多様性を維持する。

粒子群最適化 (Particle Swarm Optimization: PSO)

粒子群最適化 (PSO) は、群れの個々の粒子が協調的に最適解を探索する手法である。各粒子は位置 x_i と速度 v_i を持ち、その位置と速度は次の式で更新される。

粒子の位置と速度の更新

粒子の位置・速度は次のように更新される。

$$x_p^{k+1} = x_p^k + v_p^{k+1} \quad (2.43)$$

$$v_p^{k+1} = wv_p^k + c_1r_1(x_{pbest} - x_p^k) + c_2r_2(x_{gbest} - x_p^k) \quad (2.44)$$

ここで、 w は慣性重み（過去の速度の影響）、 c_1 と c_2 は学習係数（個体のベスト解と全体のベスト解への影響）、 r_1 と r_2 はランダムな数 ($0 \leq r_1, r_2 \leq 1$)、 p_i は粒子 i の個人ベス

ト位置、 g は全体のベスト位置である。この更新式により、粒子は局所最適解に引き寄せられつつ、全体の最適解に向かって進化していく。

これらのアルゴリズムは、複雑な最適化問題を解決するための基盤を提供しており、実世界のさまざまなシナリオに適応可能である。この中でも、PSOはそのシンプルな構造と高い収束速度から注目されている。さらに、他の最適化手法（例えば、遺伝的アルゴリズムや進化戦略）と組み合わせやすく、ハイブリッド手法を構築することで、さらなる性能向上が期待できる。

これにより、局所解に陥りやすいといったデメリットの改善もしやすいため、PSOは多目的最適化の分野において非常に魅力的な選択肢となっている。

多目的最適化の粒子群最適化

§ 3.1 粒子群最適化アルゴリズム

Swarm Intelligence (群知能) は、鳥や魚、アリのコロニーなどのグループの行動に基づく最適化手法である。この技術の一つである粒子群最適化が開発され、様々な研究に応用されている。しかし、粒子群最適化の収束は根拠がない。本研究では、より良い最適解を求めるための群知能とニューラルネットワークダイナミックスの新しいハイブリッド動的システムを提案した。本節では主な結果として、粒子群最適化と勾配法のメカニズムを理論的にどのように組み合わせるかを示し、今後は提案システムが客観的な環境のグローバルな情報に基づいて補間探索を実現できることを確認する。

粒子群最適化 (Particle Swarm Optimization : PSO) は、群の中の固体 (粒子) が持つ最良の情報とそのグループの最適値から過去の探索から考慮した確率的最適化手法であり、ケネディ [27] が社会的行動に基づいて開発した並列進化計算技術である。社会的方法と計算方法の両方を扱う PSO に関する標準的な研究がある [28]。近年では、PSO の改良版として、適応型粒子群最適化 (Adaptive PSO) や、複数の目的を考慮する多目的粒子群最適化 (Multi-objective PSO) など、さまざまな派生手法が提案されている。これらの改良により、PSO はさらに多様な問題に対応できるようになり、その適用範囲は拡大している。

本節では、粒子群最適化 (PSO) アルゴリズムの基本概念とその応用に関する重要なトピックについて論じる。具体的には、PSO の基本的な原理、制約付き PSO、ならびに多目的最適化におけるパレート解の概念に焦点を当てる。これらの要素は PSO を用いた最適化問題の解決において理論的および実践的な観点から重要な役割を果たすものであり、各トピックが最適化手法の有効性と適用範囲を拡大するための鍵となる。

PSO は群を成して移動する生物の行動を模範したアルゴリズムである。群をなす生物を粒子としてモデル化し、粒子は最適化問題における候補解を示している。PSO は群の中の粒子がもつ最良の情報とその集団の最適値から過去の探索を考慮し、さらにその集団の各粒子の位置および速度を更新することによって計算される。以下に PSO の解説を示す (図 1 参照)。ここで、PSO の探索模式図及び速度と位置の更新式より、各粒子が持つ最良の情報に向かう力、集団の最適値に向かう力、これまでの進行方向へ向かう力の 3 つのベクトルを合成して速度ベクトルを決定し、それを元に次に移動する位置を決定する [29]。

$$x_p^{k+1} = x_p^k + v_p^{k+1} \quad (3.1)$$

$$v_p^{k+1} = wv_p^k + c_1r_1(x_{pbest}^k - x_p^k) + c_2r_2(x_{gbest} - x_p^k) \quad (3.2)$$

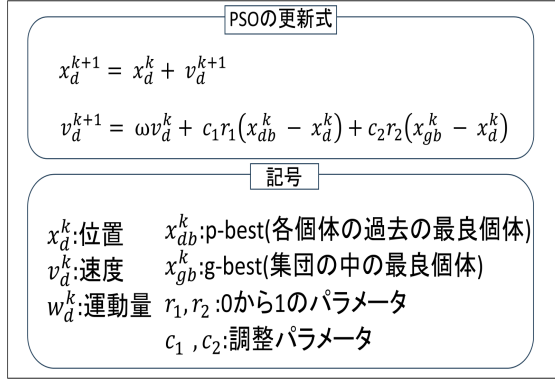


図5 PSOの更新式

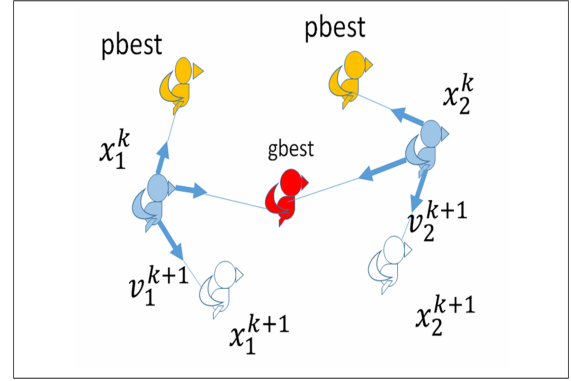


図6 PSOの探索の様子

ここで x_p^k は粒子の位置, v_p^k は速度, w は慣性係数, c_1 と c_2 は学習係数, r_1 と r_2 はランダムな数値である. PSO の速度ベクトルは, $pbest$ に向かうベクトル, $gbest$ に向かうベクトル, そして過去の進行方向に基づくベクトルの合成によって決定される.

ここで, PSO の探索模式図及び速度と位置の更新式より, $pbest$ に向かう $c_1 r_1 (x_p^k - x_p^k)$, $gbest$ に向かう $c_2 r_2 (x_p^k - x_p^k)$, これまでの進行方向へ向かう $w v_p^k$ の3つのベクトルを合成して速度ベクトル v_p^{k+1} を決定し, それを元に次に移動する位置 x_p^{k+1} を決定する. PSO の探索式はランダム要素を含み, 同時に最良解情報である $pbest$ と $gbest$ が探索に伴い変化するという時変性を有している. PSO の探索手順と例を以下に示す.

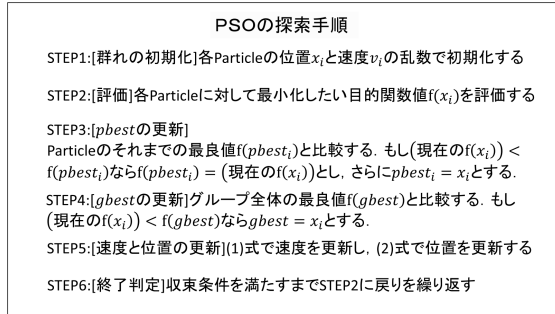


図7 PSOの探索手順

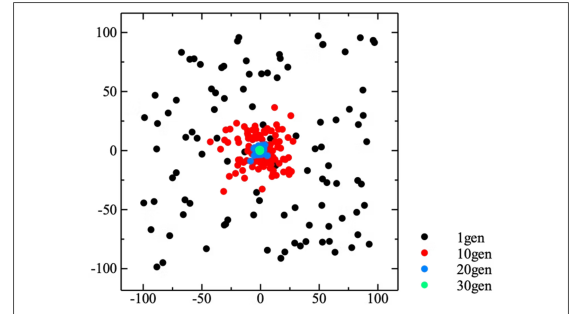


図8 粒子の動き ($x^2 + y^2$)

PSO に関する研究は, この手法自体の特性に関するものから応用研究にいたるまで,すでに数多くなされているが, 一般に, 変数の次元の小さい無制約問題に対しては, 更新式のパラメータを適切に調節すれば, 比較的効率よく大域的最適解が探索可能であるのに対して, 変数次元の大きな問題に対しては, 必ずしも大域的最適化が効率よく求められないことが指摘されている [30]. そこで, 変数次元が比較的大きな問題であっても, ある程度狭い有界な領域を制約条件とする問題であるならば, その領域上の大域的最適解を比較的効率よく探索できる制約条件付最適化問題を直接解くことが可能な PSO について解説する.

また, PSO の更新式を力学系モデルとみなし, その連続化を試みると,

$$\frac{dx_p(t)}{dt} = c \int_0^t e^{-a(t-\tau)} [F_p(x_p(\tau), \tau) + C(x_p(\tau), \tau)] d\tau \quad (3.3)$$

$$\frac{d^2 x_p(t)}{dt^2} + a \frac{dx_p(t)}{dt} = c [F_p(x_p(t), t) + C(x_p(t), t)] \quad (3.4)$$

またそれぞれの関数 F_p と C は以下のようになる。

$$F_p(x_p, k) = c_1 (x_p(l_p(k)) - x_p) \quad (3.5)$$

$$C(x_p, k) = c_2 (x_Q(k)(l_o(k)) - x_p) \quad (3.6)$$

ここで、 $l_p(k)$ や $(Q(k), l_o(k))$ は次のように定義される：

$$l_p(k) = \arg \min_l \{E(x_p(l)) \mid l = 0, 1, \dots, k\} \quad (3.7)$$

$$(Q(k), l_o(k)) = \arg \min_{(q,l)} \{E(x_q(l)) \mid q = 1, 2, \dots, P, l = 0, 1, \dots, k\} \quad (3.8)$$

また、 $l_p(k)$ は時刻 k までにおいて、個体 p が関数 E の値を最も小さくした時刻を表しており、その状態 $\mathbf{x}_p(l_p(k))$ は PSO における p -best に相当する。さらに、 $(Q(k), l_o(k))$ は時刻 k までにおいて、 P 個の個体の中で関数 E の値を最も小さくした個体番号とその時刻を表しており、その状態 $\mathbf{x}_Q(k)(l_o(k))$ は PSO における g -best に相当する。

ここで、 $l_o(k)$ は個体 p に対応して決まる離散時刻であるため、 $Q(k)$ と $l_o(k)$ との間には次の関係が成り立つ：

$$Q(k) = \arg \min_q \{E(x_q(l_q(k))) \mid q = 1, 2, \dots, P\} \quad (3.9)$$

$$l_o(k) = l_{Q(k)}(k) \quad (3.10)$$

また、2 階微分方程式で表される連続時間系モデルの状態変数表現を

$$u_p(t) = x_p(t), \quad v_p(t) = \frac{du_p(t)}{dt} + au_p(t) \quad (3.11)$$

とにおいて導入すると、離散時間系に対応した連続系の内部状態表現モデルは、

$$\frac{du_p(t)}{dt} = -au_p(t) + v_p(t) \quad (3.12)$$

$$\frac{dv_p(t)}{dt} = c [F_p(u_p(t), t) + C(u_p(t), t)] \quad (3.13)$$

このように、PSO を力学系の視点から再解釈し、連続系モデルを活用することで、高次元の問題にもより適応的に対応でき、効率的な最適化を実現する可能性が広がる。

§ 3.2 制約がある場合の粒子群最適化

多くの現実世界の最適化問題は、単純な目的関数の最適化にとどまらず、複数の制約条件を満たす必要がある。制約条件とは、最適解が物理的、技術的、または経済的な制限内で存在することを要求するものであり、これを無視して解を求めることは実行不可能な解

に至る原因となる．例えば，設計問題においては，設計変数が一定の範囲内に収められる必要があり，さらに資源や時間の制約が加味されることが一般的である．

例えば，上下限制約付き最適化問題は次のように定義される：

$$\min E(x) \quad (3.20)$$

subject to

$$p_i \leq x_p \leq q_i, \quad i = 1, \dots, n \quad (3.21)$$

粒子群最適化における制約条件の取り扱い方法として，様々な制約付き粒子群最適化が提案されている．主なアプローチとしては，制約違反に対するペナルティを設ける方法，制約を満たさない粒子を無効とする方法，そして制約満足度に基づいて粒子の評価を行う方法などが挙げられる．これらの方法により，PSO アルゴリズムは制約条件を効果的に考慮しながら最適解を探索することが可能となる．以下に例を示す．

境界上への移動（位置の変更）

粒子が制約条件を越えた場合，その粒子の位置は制約条件の境界に合わせて修正する．たとえば，粒子の位置が上限 q_i を越えた場合，その位置は q_i に設定され，下限 p_i を越えた場合は p_i に設定される．これにより，粒子は制約範囲内に収まるようになる．

上限を越えた場合: 位置を $x_p = q_i$ に設定

下限を越えた場合: 位置を $x_p = p_i$ に設定

粒子の位置修正方法（境界上への移動）には，上限または下限を越えた場合にその値に設定するだけなので，実装が非常にシンプルで理解しやすいという利点がある．一方で，頻繁に境界に触れると，粒子の動きが不自然になり，局所解に閉じ込められるリスクが高まる．**反対方向への速度修正（反転速度）**

粒子が制約条件を越えた場合，速度を反転させることで，粒子を制約範囲内に戻すようにする．

反転速度の修正方法は、次のようになる：

粒子が上限を越えた場合，その粒子の速度 v_p に -1 を乗じ，反対方向に向かわせる．

位置修正後の速度: $v_p = -v_p$ (上限を越えた場合)

同様に，粒子が下限を越えた場合も，速度に -1 を乗じて反対方向に動かす．

位置修正後の速度: $v_p = -v_p$ (下限を越えた場合)

これにより，粒子は境界に「反発」して元の範囲に戻ることができ，探索空間を制約に適合させることができる．この手法の利点としては，粒子が境界に到達した際に速度を反転させることで，制約範囲内に戻る動きが自然で直感的である点が挙げられる．一方で粒子が境界に頻繁に触れると，速度が反転するたびに振動的な動きになり，収束が遅くなる

可能性がある。

ペナルティ関数

制約条件

$$p_i \leq x_p \leq q_i \quad i = 1, \dots, n \quad (3.21)$$

を満たさない場合に対するペナルティを次のように定義できる：

$$P(x) = \sum_{i=1}^n [\max(0, x_i - q_i) + \max(0, p_i - x_i)] \quad (3.33)$$

このペナルティ関数は、各変数 x_i が制約を越えた場合にペナルティを加える。具体的には：

- $x_p > q_i$ の場合、ペナルティ $x_p - q_i$ が加える
- $x_p < p_i$ の場合、ペナルティ $p_i - x_p$ が加える。

制約を満たす場合は、ペナルティはゼロとなる。

ペナルティ法では、最適化すべき目的関数にペナルティ項を加える。これにより、制約を満たさない解にはペナルティが課され、制約を満たす解が優先されるようになる。

修正された目的関数は次のように表される：

$$\min (E(x) + \rho P(x)) \quad (3.34)$$

ここで、 ρ はペナルティ係数で、制約違反に対してどれだけペナルティを課するかを調整するパラメータである。通常、 ρ は正の定数、十分に大きな値に設定されることで、制約違反が最小化されるようになる。ペナルティ法は、制約条件を直接的に扱うことなく目的関数に組み込むことができ、既存の最適化アルゴリズムに簡単に適用できる。このため、複雑な制約条件を持つ問題でも、ペナルティを追加するだけで簡単に制約を考慮した最適化が可能である。しかしながら、ペナルティパラメータによっては探索効率が著しく悪化するなど、パラメータ選択に対する問題がある

非線形変数変換モデル

制約付最適化問題を直接解くために、この上下限制約領域内に問題の変数を変換して無制約化した新たな変数空間に無制約 PSO モデルを適用した「変数変換モデル」を導入する。非線形変数変換モデルを作成するために、

$$x_p = f_i(y_i) = \frac{q_i + p_i \exp(-y_i)}{1 + \exp(-y_i)} \quad (3.38)$$

とおく。ここで、 x_p は制約付き最適化問題の変数であり、 y_i は無制約問題における新しい変数である。変数変換式により、元の変数 x_p は変換後の新しい変数 y_i に置き換えられ、この変換により、制約条件が自然に無制約問題内に組み込まれることとなる。変換後の目的関数は、以下のように記述される。

$$E(f(y)) = E \left(f \left(\frac{q_i + p_i \exp(-y_i)}{1 + \exp(-y_i)} \right) \right) \quad (3.39)$$

このように変換された最適化問題は、制約条件が反映された無制約問題となり、PSO アルゴリズムを適用するための準備が整う。PSO アルゴリズムは、粒子群が最小化すべき目的関数 $E(f(y))$ を探索し、最適解を求める過程で、変換後の変数空間における粒子の位置と速度を更新する。

本手法の主な利点は、変数変換により制約条件を無制約空間内に自然に組み込むことができる点である。これにより、PSO アルゴリズムをそのまま適用でき、制約条件付き最適化問題における探索を効率的に行うことが可能となる。特に、非線形の制約条件が複雑に絡む問題において、変数変換を用いることで問題が簡素化され、無制約 PSO による最適解探索が可能となる。一方で、制約条件が無制約問題の中に「隠れる」形で組み込まれるため、制約を強調した解の探索が難しくなる場合がある。

§ 3.3 多目的粒子群最適化

MOPSO (Multi-Objective Particle Swarm Optimization) は、多目的最適化問題を解くために設計されたアルゴリズムである。これは、単一の目標を達成する従来の Particle Swarm Optimization (PSO) を拡張したものである。MOPSO の目的は、複数の目標を同時に最適化することによって、解空間上にパレート最適解 (Pareto Optimal Solution) と呼ばれる非支配解の集合を得ることである。

PSO は群知能を活用した最適化アルゴリズムであり、粒子と呼ばれる仮想エージェントが解空間内を探索し、最適解を見つける仕組みである。各粒子は自分自身の経験 (ベスト位置) と群全体の経験 (グローバルベスト) をもとに次の位置を決定する。一方、MOPSO では複数の目的関数を考慮する必要があるため、グローバルベストの選択やパレートフロントの管理に特化したメカニズムが追加されている。

MOPSO は PSO と同様に、群状に分布し m 次元の探索空間を移動する探索点 $x_p^k \in \mathbb{R}^m$ が、自身の持つ最良解の位置情報 $x_{\text{pbest}}^k \in \mathbb{R}^m$ と群れで共有するパレート解の位置情報 $x_{\text{gbest}} \in \mathbb{R}^m$ を使って移動ベクトル $v_p^k \in \mathbb{R}^m$ を生成して解を探索し、最終的に残った x_{gbest} の集合をパレート最適解集合とする手法である。

初めに探索点個数 N_I 、反復計算回数 N_K 、保存点の個数を N_R 、保存点個数最大値 N_{max}^R 、はハイパーキューブの分割数 H を決定し、 x_p^k 、 x_{pbest}^k 、archive、 v_p^k の初期設定を行う。ただし、 p は探索点番号を表し、 k は探索点の世代を表す。 x_p ($1 \leq i \leq N_I$) は制限値内で無作為に決定し、 $x_{\text{pbest}}^k = x_p^k$ ($1 \leq i \leq N_I$)、archive = x(i) ($r = i, 1 \leq r \leq N_I$)、 $v(i) = 0$ ($1 \leq i \leq N_I$) とおく。

n 目的最適化問題 ($n > 1$) を扱う場合、探索点 x_p は n 個の目的関数値を持つので、それらの値によって n 次元の目的関数空間内の位置が決まり、各探索点を評価することができる。これらの探索点自身が持つ最良解の位置情報 x_{pbest}^k と群れで共有する archive 内の要素も同様に n 個の目的関数値を持ち、 n 次元の目的関数空間内に存在することになる。次に、pbest, gbest, archive の更新について述べる。

gbest の更新と保存

x_{gbest} の更新は、ハイパーキューブを用いて行われる。ハイパーキューブは、解空間を均等に分けた格子状のセルであり、解空間を複数のハイパーキューブに分割し、各ハイパーキューブ内で解がどれだけ密集しているかを確認することができる。更新の際は、最も多くの解を含んでいるハイパーキューブを選び、その中からランダムに1つの解を gbest として選択する。この方法によって、解空間全体を均等にカバーしつつ、最も多くの解が集まった領域に基づいて最適解を決定することが可能である。ここで、 H_i はハイパーキューブ i の解の集合とする。

x_{pbest}^k とアーカイブの更新と保存

このステップは、以下の (A)～(E) のサブステップから構成されている。

Step(A)

x_p^{k+1} が全ての目的関数値に対して x_{pbest}^k よりも優れている場合は、 x_{pbest}^k を x_p^{k+1} に更新する。

Step(B)

x_p^{k+1} がある目的関数値に対して x_{pbest}^k よりも優れているが、他の目的関数値に対して x_{pbest}^k よりも劣っている場合は、 x_{pbest}^k を x_p^{k+1} に更新するかどうかを無作為に決定する。

Step(C)

全ての目的関数値に対して x_p^{k+1} よりも劣るようなアーカイブの要素が存在する場合は、そのうち1つを x_p^{k+1} に更新する。それ以外は後の処理 (E) で削除される。

Step(D)

x_p^{k+1} が全てのアーカイブの要素に対して少なくとも1つの目的関数値において優れている場合も、あらたなアーカイブの要素として保存する。またこのとき保存点個数が1つ増えるので、保存点個数を $NR + 1$ とする。

Step(E)

アーカイブのサイズが増加すると、計算資源を圧迫するため、アーカイブには最大サイズ (N_{max}^R) が設定されている。アーカイブがこのサイズを超えると、解の選択と削除が行われる。この選択は2つの異なる方法が使われている。

混雑距離に基づく選択:

混雑距離とは、解がパレートフロンティア内でどれほど他の解と異なるかを示す指標である。混雑距離が大きい解は、パレートフロンティア上で多様性を提供するため、優先的に保持される。アーカイブ内の全ての解に対して混雑距離が計算され、距離が大きい解から順に選ばれ、アーカイブが最大サイズに収束するよう調整される。この方法により、多様性が保持され、パレートフロンティアの網羅性が高まる。これは、次式で表現される。

$$d_j = \sum_{m=1}^M \frac{E_{I_m^{j+1}}^m - E_{I_m^{j-1}}^m}{E_{\text{max}}^m - E_{\text{min}}^m} \quad j \in [2, l-1] \quad (3.40)$$

ここで、 d_j は個体 j の混雑距離、 M は目的関数の数、 m は各目的関数を示し、 l はアーカイブ内の個体数、 E^m は目的関数の値を表す。

合計値による選択:

もう一つの方法として、アーカイブ内の解を目的関数の値の合計で評価し、合計値が最大の解を削除する手法がある。この方法では、混雑距離の持つ境界付近の解の処理の不均衡という弱点を補完するために、解の「重要度」を合計値で判断し、より適切な解を保持する。

これらの方法を用いて解を選択し、混雑距離の小さい解や、目的関数の値の合計が大きい解を削除することによって、アーカイブの要素数を調整する。

これらをまとめて、MOPSO アルゴリズムの手順を説明する：

1. 探索母集団を初期化する。
2. 各個体の速度を 0 に初期化する。
3. 各個体を評価する。
4. 非劣解をアーカイブに保存する。
5. 各個体の $pbest$ を初期化する。
6. $gbest$ を選出する。
7. 速度を更新する。
8. 位置を更新する。制約条件を越えた個体は、その位置を境界上に設定し、速度には-1 を乗じて反対方向に向かわせる。
9. 各個体の評価を行う。
10. アーカイブの内容を更新する。
11. 現在の位置が $pbest$ よりも良い場合、 $pbest$ を更新する。
12. ループカウンタに 1 を加えて、手順 6 に戻る。

アルゴリズムの進行において、各粒子は次の目的関数の最適解を探索し、アーカイブに保存された解と比較しながら進化する。これにより、最適解が探索されるとともに、解の多様性を保つことができる。

MOPSO のメリットとデメリット

MOPSO の主なメリットは、群知能に基づくアプローチであるため、他の最適化アルゴリズムと比較して実装が比較的容易である点である。特に、非線形かつ複雑な多目的最適化問題に対して有効であり、解の探索過程において個体群間の相互作用を活かして広範囲にわたる探索を行うため、局所解に陥りにくい特徴を持っている。このような特性から、MOPSO は探索の多様性を保ちながら、高品質なパレート最適解を効率的に求めることが可能である。

また、MOPSO は粒子群の位置と速度を調整することで、非劣解集合（パレートフロンティア）を継続的に進化させ、複数の目標間でのトレードオフを反映した解を提供する。このため、設計や運用における意思決定を支援する有力なツールとして広く使用されています。

一方で、MOPSO にはいくつかのデメリットも存在する。まず、計算負荷が高くなる場合があり、特に粒子数や目的関数の数が増加するにつれて、計算時間が増大し、実行速度に影響を及ぼすことがある。これにより、計算リソースを大量に消費することが課題となる。そのため、大規模な問題においては、計算効率を向上させるための適切なアルゴリズムの改良が求められる。

また、MOPSO の効果的な運用には外部アーカイブの管理が不可欠である。アーカイブ

内の解の多様性を維持することが重要であり、管理が不十分である場合、解空間における探索の偏りが生じ、最適解の探索が不完全になる可能性がある。したがって、アーカイブ更新のアルゴリズムや混雑距離などの評価指標を適切に設定することが、MOPSO の性能を最大化するために必要である。

提案手法

§ 4.1 勾配を考慮した粒子群最適化 (1 目的)

勾配 PSO アルゴリズム

本節では PSO に勾配情報の要素を加えた勾配 PSO について解説する．勾配 PSO は従来の PSO の動きに加えて，目的関数の勾配情報を利用することで，解の探索をより効率的に行う．勾配情報を取り入れることにより，勾配が急激な方向へと粒子が移動するため，収束速度が向上し，局所最適解への収束を防ぐ効果も期待できる．

PSO の力学系としての特徴を捉えるために，探索点が単一の場合を考えると，その更新式は次のように表される：

$$x(k+1) = x(k) + \Delta x(k+1) \quad (4.1)$$

$$\Delta x(k+1) = \lambda \Delta x(k) + c_1 \{x(l(k)) - x(k)\} \quad (4.2)$$

$$l(k) = \arg \min_i \{E(x(i)) \mid i = 0, 1, \dots, k\} \quad (4.3)$$

ここで， $l(k)$ は時刻 k の時点までに関数 E を最も小さくした過去の時刻である．(4.3) 式の右辺第 2 項を次のように定義する：

$$F(x, k) = c_1 \{x(l(k)) - x\} \quad (4.4)$$

これにより，(2b) 式は次のように一般化される：

$$\Delta x(k+1) = \lambda \Delta x(k) + F(x(k), k) \quad (4.5)$$

ここで， F の 2 番目の引数 k は $l(k)$ の k であり，関数 F が時刻 k の変動に直接影響を受けていることを示している．(4.1) 式と (4.2) 式から $\Delta x(k)$ を消去すると，次のような式が得られる：

$$x(k+1) - (1+\lambda)x(k) + \lambda x(k-1) = F(x(k), k) \quad k = 1, 2, \dots \quad (4.6)$$

この式は，特性根として 1 と λ を有する 2 階差分方程式であることがわかる． $\lambda \leq 0$ のとき振動することが確認できる．

さらに，(4.6) 式を解いて，級数形式で書き直すと次のようになる：

$$x(k+1) = x(k) + \sum_{i=0}^k \lambda^{k-i} F(x(i), i) + \lambda^{k+1} \Delta x(0) \quad (4.7)$$

また、 $F(x(0), 0)$ については次のように定義される：

$$F(x(0), 0) = c_1 \{x(0) - x(0)\} = 0 \quad (4.8)$$

この式から明らかに、過去の $F(x(i), i)$, $i = 0, 1, \dots, k-1$ を重み λ^{k-i} で畳み込んで変位する慣性系であることがわかる。

ここで仮に、 k とは無関係に：

$$F(x, k) = -c_3 \nabla E(x) \quad (4.8)$$

とみると、勾配法型慣性系である。勾配法における $-\nabla E(x)$ は、 x において関数 E を局所的に最も減少させる方向であるのに対して、PSO では勾配を用いる代わりに式 (4.4) を用いていることになるが、 $x(l(k))$ が過去の時系列でもっとも E を小さくする状態であることから、式 (4.8) の F も等号を含めた広義の関数 E の減少方向を与えるものであることがわかる。

ここで、勾配を最適解を探索する際に粒子が進むべき方向を決定するために使用する。勾配の計算は、粒子の位置の変化に基づいて行われる。数式で表すと、勾配 ∇E は以下のよう計算される：

$$\nabla E \approx \frac{E(x_k) - E(x_{k-1})}{x_k - x_{k-1}} \quad (4.9)$$

ここで、 x_k と x_{k-1} はそれぞれ現在の粒子位置と前回の粒子位置である。勾配は、目的関数の評価値の変化に基づいて計算される。

速度更新における勾配情報の利用

勾配情報は、粒子がどの方向に移動すべきかを決定するために速度更新に組み込まれる。速度更新式は以下のようになる：

$$v_p^{k+1} = wv_p^k + c_1 r_1 (x_{pbest} - x_p^k) + c_2 r_2 (x_{gbest} - x_p^k) - (c_3 \nabla E) \quad (4.10)$$

勾配を加えることで、粒子は目的関数の傾きに従って効率的に探索を行う。勾配項は、粒子の進行方向を調整し、局所的な最適解から脱出する手助けを行う。

ここで、Griewank 関数と Ackley 関数を用いて従来の PSO と勾配情報を追加した PSO を比較する数値実験を行った。

Griewank 関数

$$f(x_1, \dots, x_n) = 1 + \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) \quad (4.12)$$

条件：

$$-600 \leq x_i \leq 600, \quad f_{\min}(0, \dots, 0) = 0 \quad (4.13)$$

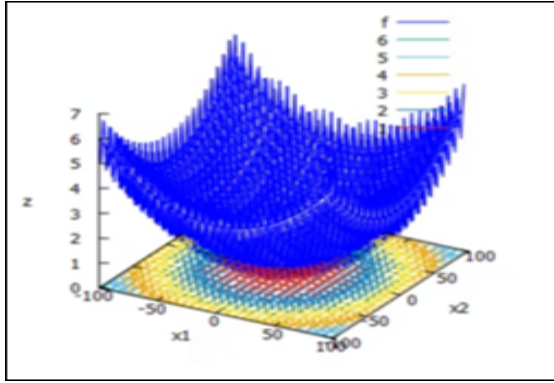


図9 Griewank 関数

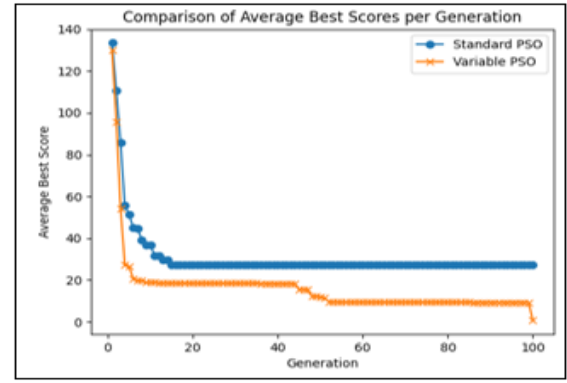


図10 比較

Ackley 関数

$$f(x_1, \dots, x_n) = 20 - 20 \exp \left(\sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) + e - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right) \quad (4.14)$$

条件：

$$-32.768 \leq x_i \leq 32.768, \quad f_{\min}(0, \dots, 0) = 0 \quad (4.15)$$

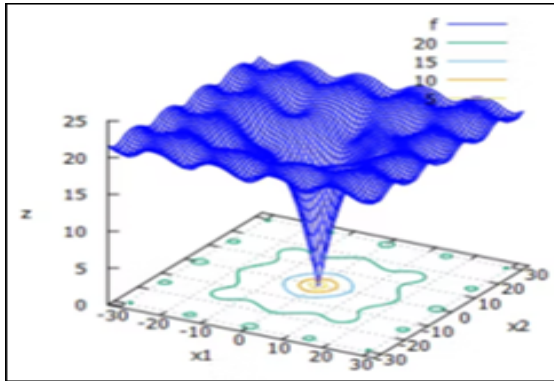


図11 Ackley 関数

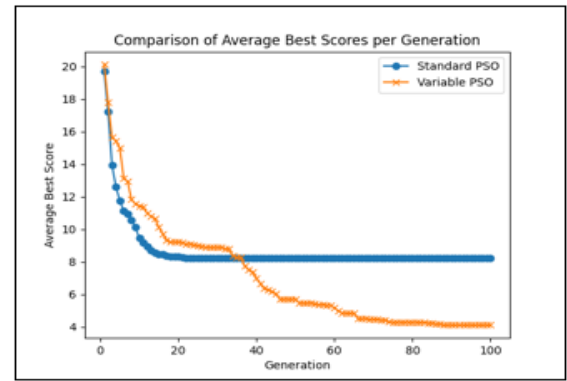


図12 比較

パラメータ：

粒子の数 $N = 10$ $c = 1.0$ $c_1, c_2 = 1.4$ $c_3 = 0.1$ $a = 1.0$ $\Delta T = 0.8$ 世代数: 100

Griewank 関数は、非常に多くの局所解をもつ多峰性関数である。Ackley 関数は、大域的最適解の周辺に多くの局所解をもつ多峰性関数である。結果を考えると、勾配情報を加えた PSO は従来手法よりも最適な値に収束していることがわかる。よって、勾配情報を加えた PSO は多峰性をもつ関数に有効であることが示された。これは、Griewank 関数が非常に多くの局所解をもつという特徴があり、従来手法が局所解に収束してしまう場合が多いことに反して、勾配情報を加えた PSO が Griewank 関数という関数全体を見て、局所解から脱出しているのではないかと考察できる。また、Ackley 関数においても、同様に探索を繰り返す中で局所解を脱出し、大域的最適解に向かっていくと考察できる。

§ 4.2 上下制約付き勾配MOPSO

従来の粒子群最適化 (PSO) アルゴリズムは、主に無制約最適化問題に対して適用されてきた。しかし、実際の最適化問題の多くは、解が特定の範囲内に収束することを求める制約付き問題である。特に、多目的最適化問題においては、複数の目的関数が同時に最適化される必要があり、かつその解に対して上下制約が課されることが多い。したがって、MOPSO における制約条件を考慮した最適化手法の確立は、重要な課題である。本章では、上下制約を有する MOPSO の適用方法について詳細に述べ、制約を考慮した最適化プロセスの実装を行う。

MOPSO は、複数の目的関数を同時に最適化するために粒子群を用いて探索を行い、効率的にパレート最適解を求めるアルゴリズムである。各パーティクルは、目的関数の値およびその探索空間における位置を保持し、探索の過程で最適解に収束するように進化を繰り返す。しかし、MOPSO アルゴリズムにおいても、解が制約を満たすことが求められるため、制約を考慮した探索が不可欠である。

上下制約付き勾配MOPSO の重要な要素

上下制約付き MOPSO の重要な要素については、以下のように整理される。

制約条件の処理 粒子群最適化 (PSO) における制約条件の処理は、探索空間内で粒子が制約を満たすように行われる。具体的には、粒子の位置が制約条件を超えた場合、その位置を適切に修正する手法が用いられる。例えば、上下制約問題が

$$\min E_m(x) = \min\{E_1(x), E_2(x), \dots, E_m(x)\} \quad (4.16)$$

$$p_i \leq x_i \leq q_i, \quad i = 1, 2, \dots, n \quad (4.17)$$

のように与えられた場合、粒子の位置 x_i がこの条件を満たすように修正される。これにより、粒子は解空間内で有効な領域に収束し、制約を満たしつつ最適化が進行する。

次に提案手法であるハイブリッド MOPSO について解説する。MOPSO の応用法であり、勾配情報を加えることにより、精密な探索を行うことを狙いとしている。また、制約条件には非線形変数変換モデルを利用している。

$$\frac{dx_p(t)}{dt} = c \int_0^t e^{-a(t-\tau)} [F_p(x_p(\tau), \tau) + C(x_p(\tau), \tau) - \nabla E(x_p(\tau), \tau)] d\tau \quad (4.18)$$

$$\frac{d^2x_p(t)}{dt^2} + a \frac{dx_p(t)}{dt} = c [F_p(x_p(t), t) + C(x_p(t), t) - \nabla E(x_p(t), t)] \quad (4.19)$$

またそれぞれの関数は以下ようになる。

$$F_p(x_p, t) = c_1(x_{pbest} - x_p) \quad (4.20)$$

$$C_p(x_p, t) = c_2(x_{gbest} - x_p) \quad (4.21)$$

$$\nabla E(x_p, t) = c_3 \frac{\partial E(x_p, t)}{\partial x_p} \quad (4.22)$$

解説したままのモデルでは無制約なので、制約条件に対応したモデルである上下限制約連続時間 PSO モデルを以下の式に示す．上下限制約付最適化問題は式 4.16 と式 4.17 を利用する．

$$\min E_m(x) = \min\{E_1(x), E_2(x), \dots, E_m(x)\} \quad (4.16)$$

$$p_i \leq x_i \leq q_i, \quad i = 1, 2, \dots, n \quad (4.17)$$

これを直接解くために、この上下限制約領域内に問題の変数を変換して無制約化した新たな変数空間に無制約 PSO モデルを適用した「変数変換モデル」を導入する．非線形変数変換モデルを作成するために、

$$x_p = f(y_i) = \left(\frac{q_i + p_i \exp(-y_i)}{1 + \exp(-y_i)} \right) \quad (4.23)$$

とおく．この変換式を制約条件付き問題に代入して変数 x を消去すると、

$$\min E_m(f(y)) \quad (4.24)$$

を得ることができる．よって式 (4.19) に対応させると、

$$\frac{d^2 y_p(t)}{dt^2} + a \frac{dy_p(t)}{dt} = c [F_p(y_p(t), t) + C(y_p(t), t) - \nabla E(y_p(t), t)] \quad (4.25)$$

またそれぞれの関数は以下のようなになる．

$$F_p(y_p; t) = c_1(y_{pbest} - y_p) \quad (4.26)$$

$$C_p(y_p; t) = c_2(y_{gbest} - y_p) \quad (4.27)$$

$$\nabla E(y_p; t) = c_3 \frac{\partial E(y_p; t)}{\partial y_p} \quad (4.28)$$

次にプログラムへの実装を考えた時に、連続式のままではプログラムに実装することが難しいので、オイラー法を用いて連続式を離散化し非線形変数変換モデルの離散化 PSO を作成する．それぞれに対応する式を以下に示す．

$$u_p(k+1) = (1 - a\Delta T)u_p(k) + \Delta T v_p(k) \quad (4.29)$$

$$v_p(k+1) = v_p(k) + c\Delta T \left[F_p(u_p(k), k) + C(u_p(k); k) - \sum_{i=1}^m \nabla E_i(u_p(k), k) \right] \quad (4.30)$$

$$F_p(k, k) = c_1(u_{pbest} - u_p(k)) \quad (4.31)$$

$$C_p(k, k) = c_2(u_{gbest} - u_p(k)) \quad (4.32)$$

$$\nabla E(k, k) = c_3 \frac{\partial E(k, t)}{\partial k} \quad (4.33)$$

$$u_{pbest} = \begin{cases} u_p(k) & \text{if } \forall m, E_m(u_p(k)) < E_m(u_{pbest}) \\ u_p(k) & \text{with probability } random \text{ if } \exists m, E_m(u_p(k)) < any E_m(u_{pbest}) \\ u_{pbest}(k) & \text{otherwise} \end{cases} \quad (4.34)$$

$$u_{gbest} = \text{random}(\arg H_{\max} |H_i|) \quad (4.35)$$

$$x_p^i(k) = f_i(u_p^i(k)) = \left(\frac{q_i + p_i \exp(-u_p^i(k))}{1 + \exp(-u_p^i(k))} \right); i = 1, \dots, n \quad (4.36)$$

このように、粒子の位置更新式（式 4.29）は、現在の位置と速度を基に新しい位置を計算する．ここでは、位置が前回の位置から減衰しつつ、速度に基づいて移動することを示している．次に、速度の更新（式 4.30）では、現在の速度に対して、個人最適解（ u_{pbest} ）およびグローバル最適解（ u_{gbest} ）に向かう引力を加える．また、エネルギー勾配（ ∇E ）を考慮し、粒子の動きを調整する．また、個人最適解（ u_{pbest} ）は、粒子の位置がその粒子自身の最良の解を更新するルール（式 4.34）に従って決定される．もし現在の位置がすべての目的関数値において、個人最適解より優れていれば、新しい位置が最適解として更新される．さらに、ある目的関数値で優れていて、いずれかの目的関数値で劣っている場合も、一定の確率で更新されることがある．このランダム性により、アルゴリズムは局所解に陥ることを避けることができる．さらに、グローバル最適解（ u_{gbest} ）は、最も多くの解を持つハイパーキューブ（探索空間の区分）から選ばれ、そこからランダムに最適解が選ばれる（式 4.35）．この方法で、アルゴリズムは探索空間全体を効率的にカバーする．これらの式を用いて更新を繰り返し、アーカイブに探索過程で得られた解を保存し、支配関係に基づいて最適解を選別する．解がアーカイブ内で支配されていなければ追加され、アーカイブのサイズが上限に達すると、混雑距離や合計値を使って解を選別するパレーフロントは、支配されない解の集合で、最終的な最適解を示す．

§ 4.3 多目的における勾配の決定法

多目的における勾配の決定法について述べる．1 目的最適化問題では、最適化する対象が 1 つの目的関数のみであるため、勾配を利用したアプローチは比較的単純である．よって、PSO の粒子は、目的関数の局所的な変化方向を示す勾配情報を用いて探索を進めることができる．勾配を速度更新に組み込むことで、粒子はより効率的に最適解に収束することが期待できる．

一方、多目的最適化問題では、複数の目的関数が同時に最適化されるため、勾配情報をどのように扱うかは一層複雑になる．各目的関数は相反する場合が多いため、最適解は単

一の解ではなく、パレートフロント（非支配解の集合）として存在する．従って、MOPSOにおいては、勾配情報をどのように活用するかが重要な課題となる．特に、それぞれの目的関数の勾配をどのように統合するかが問題となる．以下に、勾配情報の活用方法をいくつか提案する．

勾配の和を用いる方法 勾配情報を利用した基本的なアプローチの一つとして、各目的関数に対する勾配を単純に加算して粒子の速度を更新する方法がある．この方法では、各目的関数が提供する勾配情報を統合し、それに基づいて粒子の移動方向を決定する．具体的には、任意の目的関数 $E_1(x), E_2(x), \dots, E_m(x)$ に対して、その勾配を次のように合成することにより、粒子の速度更新を行う：

$$v_p^{(k+1)} = wv_p^{(k)} + c_1r_1 \left(x_{\text{pbest}}^{(k)} - x_p^{(k)} \right) + c_2r_2 \left(x_{\text{gbest}}^{(k)} - x_p^{(k)} \right) + c_3r_3 \sum_{i=1}^m \nabla E_i(x_p)$$

ここで、 $v_p^{(k)}$ は粒子 p の速度、 $x_p^{(k)}$ はその位置、 $x_{\text{pbest}}^{(k)}$ と $x_{\text{gbest}}^{(k)}$ はそれぞれ粒子 p のパーソナルベスト位置とグローバルベスト位置を示す．また、 c_1, c_2, c_3 は個別の加速定数、 r_1, r_2, r_3 はランダムな係数である．

この方法の最大の利点は、その単純さと実装の容易さにある．目的関数が複数である場合でも、すべての目的関数の勾配情報を単純に加算することで、粒子の速度更新が行われる．このアプローチにより、粒子は全目的関数の改善を同時に考慮しつつ、探索空間内で最適解を見つけるための移動を行う．また、複数の目的関数の値が最小の位置を見つけるという根本的な目的と合っているという利点がある．

一方で、この方法にはいくつかの課題も存在する．主な問題点は、異なる目的関数が異なるスケールや重要度を持つ場合、単純に勾配を加算するだけでは、特定の目的関数に偏りが生じる可能性がある点である．例えば、ある目的関数の勾配が他の目的関数に比べて非常に大きい場合、その目的関数が支配的となり、他の目的関数の改善が無視されてしまうことがある．このように、勾配の加算だけでは一部の目的関数に対して不適切なバイアスが生じることがあり、最適化の結果が不均衡になることがある

勾配ベクトルの合成と正規化

これは、各目的関数 $E_k(x)$ の勾配ベクトル $\nabla E_k(x)$ を正規化し、合成して粒子の探索方向を決定する方法である．この方法により、勾配の大きさの違いに関係なく、各目的関数を持つ方向情報を均等に反映させることができる．

例えば、任意の目的関数 $E_1(x), E_2(x), \dots, E_m(x)$ に対して、勾配ベクトルを次のように正規化する：

$$\nabla \hat{E}_i(x) = \frac{\nabla E_i(x)}{\|\nabla E_i(x)\|}$$

ここで、 $\nabla E_i(x)$ は目的関数 $E_i(x)$ の勾配ベクトルで、 $\|\nabla E_i(x)\|$ はその大きさである．このように、各目的関数の勾配ベクトルを正規化することで、勾配の方向に基づいた粒子の移動が行われ、スケールの違いが探索に与える影響を排除できる．

粒子の速度更新において、正規化された勾配ベクトルの合成を行うことで、粒子がすべての目的関数の勾配方向を考慮しながら探索を行うことができる．具体的には、次のように速度更新式を記述する：

$$v_p^{(k+1)} = wv_p^{(k)} + c_1r_1(x_{\text{pbest}}^{(k)} - x_p^{(k)}) + c_2r_2(x_{\text{gbest}}^{(k)} - x_p^{(k)}) + c_3r_3 \sum_{i=1}^m \nabla \hat{E}_i(x_p)$$

勾配ベクトルの正規化を行う最大の利点は、目的関数のスケールに依存せず、探索方向を均等に決定できる点である。異なる目的関数が異なるスケールを持っている場合、例えばある目的関数が非常に急峻な勾配を持ち、他の目的関数が緩やかな勾配を持つ場合でも、勾配ベクトルの正規化により、探索が特定の目的関数に偏ることなく行われる。このアプローチは、特に複数の目的関数が非線形で異なる範囲を持つ場合に効果的である。

勾配ベクトルは局所的な情報に基づいており、最適化問題が非線形かつ複雑な場合、勾配ベースのアプローチは局所的な最適解にとどまりやすくなる可能性がある。特に、目的関数が非常に複雑で、局所的な極小や鞍点が多い場合、勾配を使った探索は全体的な解空間を適切に探索できない場合がある。この場合、粒子が局所的な解に収束してしまうことがあり、グローバルな最適解を見つける可能性が低くなる可能性がある。

勾配ベクトルの合成と正規化は、任意の目的関数に対して有効なアプローチである。この手法を利用することで、異なるスケールを持つ目的関数を同時に最適化する際の探索のバランスを取ることができ、粒子が多目的最適化空間において効率的に探索を行うことを可能にする。

勾配の平均を用いる方法

勾配の平均を取る方法は、各目的関数の勾配ベクトルを平均化することで、粒子の速度更新を行い、探索がバランスよく進行するようにする。

このアプローチでは、粒子 i の速度更新において、各目的関数 $E_1(x), E_2(x), \dots, E_m(x)$ の勾配ベクトル $\nabla E_k(x_i)$ を平均して用います。具体的には、次のように粒子の速度を更新する：

$$v_p^{(k+1)} = wv_p^{(k)} + c_1r_1 \left(x_{\text{pbest}}^{(k)} - x_p^{(k)} \right) + c_2r_2 \left(x_{\text{gbest}}^{(k)} - x_p^{(k)} \right) + c_3r_3 \frac{1}{N} \sum_{i=1}^m \nabla E_i(x_p)$$

ここで、 $v_p^{(k)}$ は粒子 p の速度、 $x_p^{(k)}$ はその位置 $x_{\text{pbest}}^{(k)}$ と $x_{\text{gbest}}^{(k)}$ はそれぞれ粒子 p のパーソナルベスト位置とグローバルベスト位置を示す。また、 c_1, c_2, c_3 は個別の加速定数、 r_1, r_2, r_3 はランダムな係数である。また、 N は次元数、 E_i は各次元における勾配の成分である。最終的に、勾配ベクトル全体を平均化した値 gradmean を取得する。

勾配の平均値を各次元に適用します。これにより、全ての次元が同じ平均勾配を持つようになります。最終的なベクトル grad_mean_vector は、各次元が $\nabla \bar{f}$ であるベクトルです。

$$\text{grad_mean_vector} = [\nabla \bar{E}, \nabla \bar{E}, \dots, \nabla \bar{E}]$$

ここで、 grad_mean_vector は、全次元に同じ平均勾配 $\nabla \bar{f}$ を適用した結果です。

勾配の平均を用いる方法のメリットは、目的関数間で勾配の影響が均等になるため、特定の目的関数に偏ることなく、全ての目的関数に対する改善が考慮される点である。このアプローチの最も大きな利点は、目的関数間で勾配の影響が均等になることである。これにより、特定の目的関数に偏ることなく、全ての目的関数に対する改善がバランスよく考慮される。具体的には、複数の目的がある場合でも、各目的関数から得られる情報を等し

く反映させることができ、最適化の過程で特定の目的関数が無視されたり、過度に強調されたりすることが少なくなる。一方で、目的関数が異なるスケールや重要度を持っている場合、単純に勾配を平均化すると、特定の目的関数の影響が強くなりすぎる可能性がある。これは、勾配の和を用いる方法でも現れる同様の問題であり、勾配の平均を用いる方法は、勾配の和を用いる手法の長所と短所を共に減らしていると考えられる。

数値実験並びに考察

§ 5.1 数値実験の概要

本章では，提案した改良型 MOPSO アルゴリズムの性能を評価するために実施した数値実験について解説する．実験の目的は，提案アルゴリズムが従来の MOPSO アルゴリズムに対して，どのようにして優れた性能を発揮するかを確認することである．また，提案手法が多目的最適化問題における解の探索精度や収束特性，制約条件の考慮にどのように寄与するかを評価する．

実験には，ZDT2 問題および ZDT4 問題を選定した．これらは，多目的最適化アルゴリズムの性能を評価するための代表的なベンチマーク問題である．

ZDT2 問題: ZDT2 問題は，は単峰性で非凸型のパレートフロントを有する問題

$$\begin{aligned} f_1(x) &= x_1 \\ g(x) &= 1 + 9(n-1) \sum_{i=2}^n x_i \\ h(f_1, g) &= 1 - \left(\frac{f_1}{g}\right)^2 \\ 0 \leq x_1 &\leq 1, \quad i = 2, \dots, n \end{aligned}$$

Optimum

$$0 \leq x_1^* \leq 1, \quad x_i^* = 0 \quad \text{for } i = 1, \dots, n$$

ZDT4 問題: ZDT4 問題は，多峰性で凸型のパレートフロントを有する問題

$$\begin{aligned} f_1(x) &= x_1 \\ g(x) &= 1 + 10(n-1) + \sum_{i=2}^n (x_i^2 - 10 \cos(4\pi x_i)) \\ h(f_1, g) &= 1 - \sqrt{\frac{f_1}{g}} \\ 0 \leq x_1 &\leq 1, \quad -10 \leq x_i \leq 10, \quad i = 2, \dots, n \end{aligned}$$

Optimum

$$0 \leq x_1^* \leq 1, \quad x_i^* = 0 \quad \text{for } i = 2, \dots, n$$

これらの問題を使用することで、提案手法が異なる特性を持つ多目的最適化問題においても効果的に動作するかを確認することができる。ここで、通常の MOPSO、3 個の異なった方法で勾配情報を考慮した MOPSO の比較を行う。

問題設定

問題: ZDT2

次元数: 10 次元

目的関数数: 2

アルゴリズム設定

粒子数: $N = 300$

最大世代数: $G = 500$

パラメータ

$a = 1.0, \quad T = 0.1, \quad C = 1.0, \quad c_1 = 2.0, \quad c_2 = 2.0, \quad c_3 = 0.2 \quad w = 0.5$

初期条件

初期粒子位置: 各次元ごとに一様分布でランダムに設定

アーカイブの最大サイズ: 1000

ハイパーキューブの分割数: 10

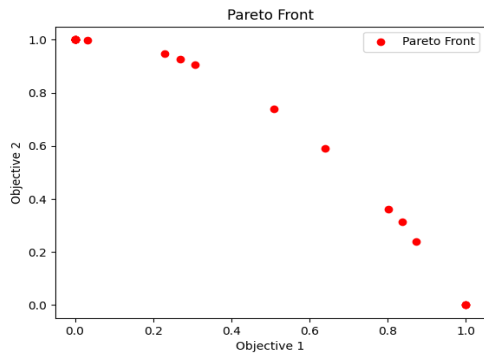


図 13 zdt2-MOPSO

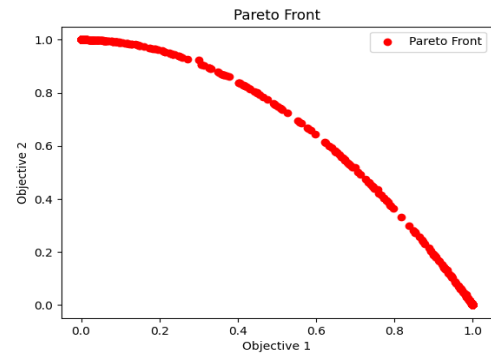


図 14 zdt2-勾配和 MOPSO

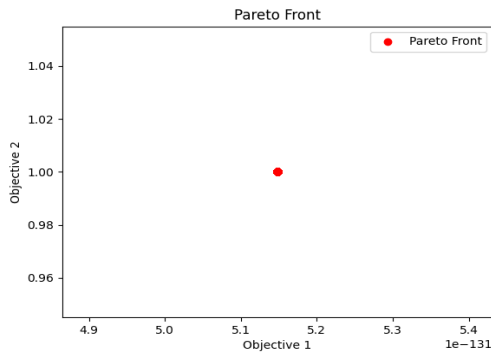


図 15 zdt2-勾配平均 MOPSO

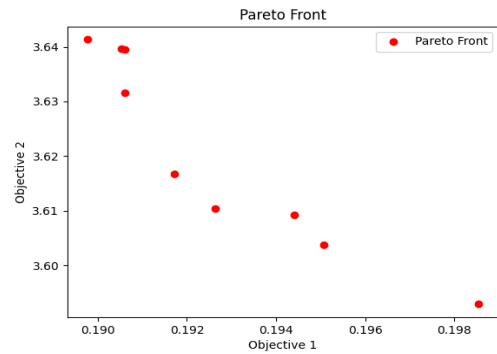


図 16 zdt2-勾配正規化 MOPSO

このように、勾配の和を使用した方法以外はまともに探索が行えていないことがわかった。通常の MOPSO が示している結果が探索自体は行えているものの明らかに探索をし切れておらず勾配を正しく活用することで粒子の節約につながり、より精密な探索ができることが考察できる。

問題設定

問題: ZDT4

次元数: 10 次元

目的関数数: 2

アルゴリズム設定

粒子数: $N = 300$

最大世代数: $G = 500$

パラメータ

$a = 1.0$, $T = 0.3$, $C = 1.0$, $c_1 = 2.0$, $c_2 = 2.0$, $c_3 = 0.2$ $w = 0.5$

初期条件

初期粒子位置: 各次元ごとに一様分布でランダムに設定

アーカイブの最大サイズ: 1000

ハイパーキューブの分割数: 10

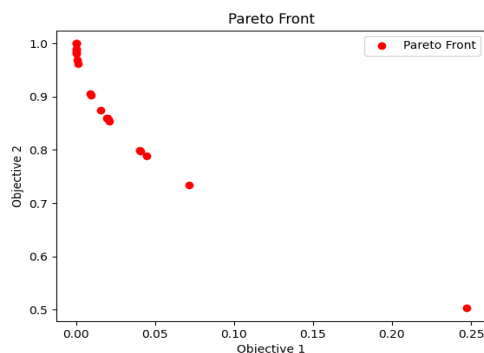


図 17 zdt4-MOPSO

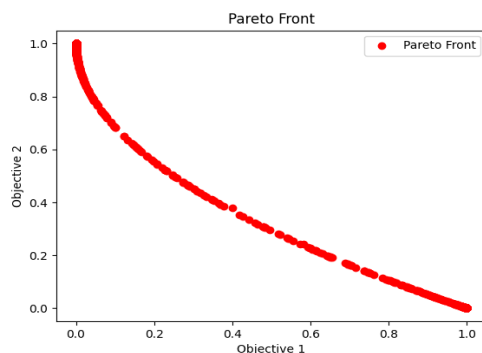


図 18 zdt4-勾配和 MOPSO

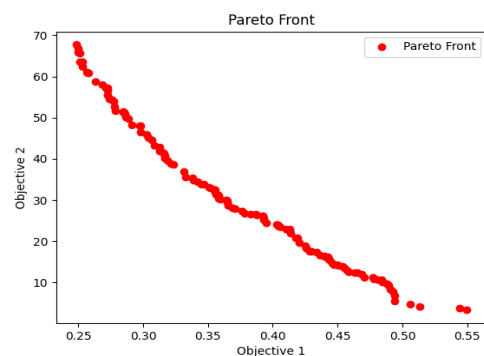


図 19 zdt4-勾配平均 MOPSO

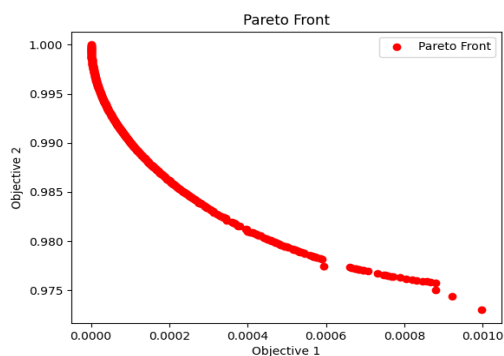


図 20 zdt4-正規化 MOPSO

zdt4 の場合も同様に、勾配の和を利用した方法が最も探索を行っている。

§ 5.2 実験結果と考察

ここで、5.1 章の実験結果について考察する．勾配の和を利用した方法がうまくいった理由について考察すると、まず一つ目に挙げられるのは「勾配情報の一貫性」である．勾配の和を使用することで、複数の目的関数の勾配方向を統合し、それぞれの目的間の関連性を反映することができる．つまり、各目的に対して一貫して最小値の探索を促進するため、pbest, gbest とつながる新たな 3 要素目として、最適化過程でバランスの取れた解を導きやすくなると考えられる．

次に、勾配の平均を活用する方法が失敗した理由として、勾配を平均化することで各次元の特性や挙動の違いが無視されてしまうことが考えられる．多くの最適化問題では、各次元が異なるスケールや重要性を持っており、これを平均化で一律に扱うことは適切ではなく、この結果、粒子が誤った方向に更新され、最適解に到達できないと考えられる．また、勾配の方向性が失われることも問題である．勾配は目的関数の最適化方向を示すものであり、各次元で異なる方向に改善が必要な場合でも、その情報が平均化によってぼやけてしまう．これにより、粒子が進むべき適切な方向を見失い、最適解を探索する能力が低下したと考えられる．

また、勾配の正規化が失敗した理由として、情報の損失が問題となる場合が考えられる．正規化によって目的関数間で勾配を同等に扱おうとする過程で、各目的の重要な情報が失われる可能性がある．例えば、ある目的関数の勾配が非常に急である場合、その情報が正規化処理で圧縮され、最適化過程において重要な変化を捉えられないことが考えられる．同様に、数値的な不安定性も考えられる．勾配の正規化では、勾配の大きさを調整するためのスケーリングが行われますが、これが数値的に不安定になることがある．特に、非常に小さな勾配を正規化する際に計算誤差が増幅されることがあり、この不安定性が最適化過程に悪影響を与えられていると考えられる．

最後に、MOPSO と多目的最適化手法として最も一般的な手法の一つである Nondominated sorting genetic algorithm-II(NSGA-II) を比較して数値実験を行う．扱うベンチマーク問題は同様に ZDT 2 および ZDT4 問題を利用する．なお次元数は 10、最大世代数は 500 とする．粒子数 500 の場合と 300 の場合を考える．

以下に ZDT2 の結果を示す．

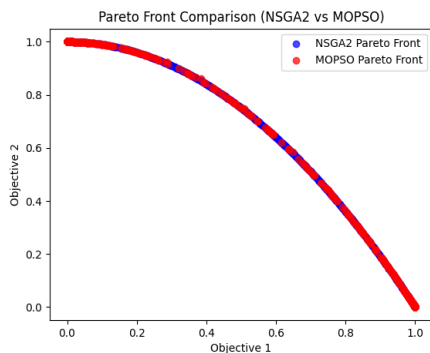


図 1
粒子 500

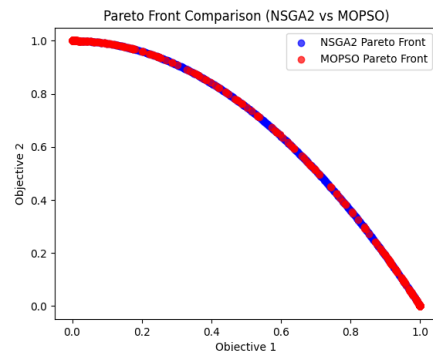


図 2 粒子数 300

以下に ZDT4 の結果を示す。

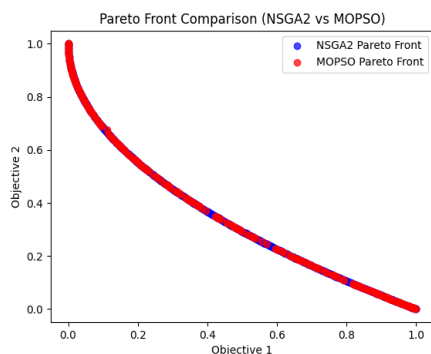


図 1

粒子数 500

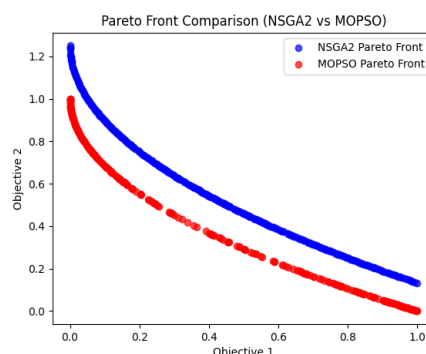


図 2 粒子数 300

結果を見ると zdt2 ではほとんど結果に差がなかったものの、多峰性で凸型のパレートフロントを有する問題である zdt4 においては、NSGA-II よりも提案手法が少ない粒子数で収束している。それぞれの特徴について解説し、考察する。MOPSO は、探索空間を効率的に探索する特性を持つ一方で、早い段階で全体的な収束が進みすぎるという特徴がある。これは、粒子が個体ベストやグローバルベストに強く引き寄せられるため、解の多様性を維持するためには、探索過程で多目的の指標を適切に調整する必要があり、調整がうまくいかないと、特定の目的に偏った解に収束することが懸念されるということである。一方、NSGA-II は、非支配ソートと混雑距離を利用することで、収束と多様性のバランスをうまく取ることができる。混雑距離によって密度の低い領域に解を配置し、多様性を保ちながら収束を進めることができるため、NSGA-II は収束性と多様性の両方を高める優れた特性を持っている。これらのことから、これらのことから、ZDT2 のような比較的単純で凸型のパレートフロントを持つ問題では、MOPSO と NSGA-II の性能差がほとんど見られなかった理由が理解できる。ZDT2 では解空間が単純で、目的関数間の相互作用が比較的少ないため、どちらのアルゴリズムも効率的に収束し、最適解に達することができるこのような場合、MOPSO のように探索空間を効率的に探索する特性と NSGA-II の収束性と多様性の両方を高める優れた特性が共に働き、両者の結果に大きな差が見られなかったと考えられる。

一方で、ZDT4 のように多峰性で凸型のパレートフロントを有する問題では、解空間が複雑で目的関数間に強い相互作用があるため、解の多様性を維持しながら効率的に収束を進めることが難しくなる。特に、MOPSO は早い段階で全体的な収束が進み過ぎる傾向があり、粒子が個体ベストやグローバルベストに強く引き寄せられるため、解の多様性が失われやすく、特定の目的に偏った解に収束するリスクが高くなる。しかし、提案手法で扱われている勾配情報が、MOPSO の収束速度の速さを担保しつつ、多様性の維持や、局所解の収束といった弱点を補ったため、少ない粒子でパレートフロントを見つけ出せたと考察する。

おわりに

本研究では、上下制約付き多目的最適化問題に対して、勾配情報を追加した改良型 MOPSO (多目的粒子群最適化) アルゴリズムを提案した。従来の MOPSO アルゴリズムでは、問題の次元数が多くなると探索が困難であるという問題がある。しかし、勾配情報を加えることで、制約付き最適化問題における探索効率を向上させ、より高精度な解を得ることができを確認した。本手法は、特に ZDT4 問題において良好な結果を示し、制約を満たしつつも探索空間を広くカバーできる可能性を示唆している。

一方で、本研究にはいくつかの課題が残されている。例えば、勾配計算に関して、数値的なアプローチを使用しているため、計算精度に限界があり、計算コストが増大し、計算資源に対する負担が大きくなることである。これを解決するために、より精度の高い勾配計算手法を導入することが求められる。

謝辞

本研究を遂行するにあたり，多大なご指導と終始懇切丁寧なご鞭撻を賜った富山県立大学工学部電子・情報工学科情報基盤工学講座の奥原浩之教授，António Oliveira Nzinga René 講師に深甚な謝意を表します．また，システム開発および数値実験にあたり，ご助力いただいた富山県立大学電子・情報工学科3年生の島部達哉氏に感謝の意を表します．最後になりましたが，多大な協力をしていただいた研究室の同輩諸氏に感謝致します．

2022 年 2 月

長瀬 永遠

参考文献

- [1] C. A. Coello Coello, “Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art,” *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 12, pp. 1245–1287, 2002.
- [2] J. Sienz and M. S. Innocente, “Particle Swarm Optimization: Fundamental Study and its Application to Optimization and to Jetty Scheduling Problems,” Chap. 6, *Trends in Engineering Computational Technology*, B. H. V. Topping and M. Papadrakakis, Eds., pp. 103–126, Saxe-Coburg Publications, 2008.
- [3] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema, “Constraint Handling in Multiobjective Evolutionary Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 514–525, 2009.
- [4] X. Hu and R. Eberhart, “Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization,” *Proceedings on 6th World Multiconference on Systems, Cybernetics and Informatics*, pp. 203–206, 2002.
- [5] G. Venter and J. Sobieszcanski-Sobieski, “Particle Swarm Optimization,” *AIAA Journal*, vol. 41, no. 8, pp. 1583–1589, 2003.
- [6] 北山哲士, 荒川雅生, 山崎光悦, “非劣解の多様性を考慮した多目的 Particle Swarm Optimization,” 日本機械学会論文集 C 編, vol. 74, no. 742, pp. 1575–1583, 2008.
- [7] S. He, E. Prempan, and Q. H. Wu, “An Improved Particle Swarm Optimizer for Mechanical Design Optimization Problems,” *Journal of Engineering Optimization*, vol. 36, no. 5, pp. 585–605, 2004.
- [8] 市川惇信 編, 多目的意思決定の理論と方法, 計測自動制御学会, 1980.
- [9] 西川, 三宮, 茨木, 最適化, 4 章, 岩波書店, 1982.
- [10] 伊理, 今野 編, 数理計画法の応用〈理論編〉, 5 章, 産業図書, 1982.
- [11] 坂和正敏, 非線形システムの最適化〈一目的から多目的へ〉, 7 章, 森北出版, 1986.
- [12] M. Sakawa, *Fuzzy Sets and Interactive Multiobjective Optimization*, Plenum Press, 1993.
- [13] 坂和, 乾口, 砂田, 澤田, 改良型遺伝的アルゴリズムによるファジィ多目的組合せ最適化, 日本ファジィ学会誌, 6-1, 177/186, 1994.
- [14] C. A. C. Coello and G. B. Lamont, *Applications of Multi-objective Evolutionary Algorithms*, vol. 1, World Scientific, 2004.

- [15] R. C. Purshouse and P. J. Fleming, “Evolutionary many-objective optimisation: An exploratory analysis,” In *Proc. of 2003 IEEE Congress on Evolutionary Computation*, pp. 2066–2073, 2003.
- [16] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, “Evolutionary many-objective optimization: A short review,” In *Proc. of 2008 IEEE Congress on Evolutionary Computation*, pp. 2419–2426, 2008.
- [17] M. Sakawa, “Optimal reliability-design of a series-parallel system by a large-scale multiobjective optimization method,” *IEEE Trans. on Reliability*, vol. 30, no. 2, pp. 173–174, 1981.
- [18] R. R. de Lucena, B. S. L. P. de Lima, B. P. Jacob, and D. M. Rocha, “Optimization of pipeline routes using an AIS/adaptive penalty method,” In *Proc. of Eighth International Conference on Engineering Computational Technology*, Paper 66, 2012.
- [19] M. Liao, Y. Zhou, Y. Su, Z. Lian, and H. Jiang, “Dynamic analysis and multi-objective optimization of an offshore drilling tube system with pipe-in-pipe structure,” *Applied Ocean Research*, vol. 75, pp. 85–99, 2018.
- [20] G. K. W. Kenway and J. R. R. A. Martins, “Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration,” *Journal of Aircraft*, vol. 51, no. 1, pp. 144–160, 2014.
- [21] K. S. Zhang, Z. H. Han, Z. J. Gao, and Y. Wang, “Constraint aggregation for large number of constraints in wing surrogate-based optimization,” *Structural and Multidisciplinary Optimization*, 18 pages, 2018.
- [22] T. Kohira, H. Kemmotsu, A. Oyama, and T. Tatsukawa, “Proposal of benchmark problem based on real-world car structure design optimization,” In *Companion of the Genetic and Evolutionary Computation Conference (GECCO) 2018*, pp. 183–184, 2018.
- [23] K. Deb and H. Jain, “An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part I: Solving problems with box constraints,” *IEEE Trans. on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.
- [24] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, “A framework for large-scale multiobjective optimization based on problem transformation,” *IEEE Trans. on Evolutionary Computation*, vol. 22, no. 2, pp. 260–275, 2018.
- [25] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable multi-objective optimization test problems,” In *Proc. of 2002 IEEE Congress on Evolutionary Computation*, pp. 825–830, 2002.

- [26] N. Srinivas and K. Deb, “Multiobjective optimization using nondominated sorting in genetic algorithms,” *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1995.
- [27] J. Kennedy and R.C. Eberhart, “Particle swarm optimization,” *IEEE Conf. on Neural Networks*, vol. IV, Piscataway, NJ, pp. 1942-1948, 1995.
- [28] J. Kennedy, R.C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann Publishers, San Francisco, CA, pp. 1942-1948, 2001.
- [29] 石亀敦司, 安田恵一郎, “群れの知能： Particle Swarm Optimization,” 知能と情報（日本知能情報ファジィ学会誌）, vol. 20, no. 6, pp. 829-839, 2008.
- [30] 岩崎信弘・安田恵一郎・井出東, “Particle Swarm Optimization の高次元問題への適用に関する検討,” 電学研資, 産業計測制御研究会, IIC-04-47～66, pp. 87–92, 2004.

