# 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: 令和7年 (2025年)2月

# 目 次

# 図一覧

# 表一覧

# 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 | $\Delta T$ |
| 制約条件 | $p_i, q_i$ |
| 勾配情報 | $\nabla E$ |

# 第1章

# 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，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 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，これらの問題は大規模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（Particle Swarm Optimization: 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（Multi Objective Particle Swarm Optimization: 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] [2] [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が提案された [5]．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 [6]．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．

# 第2章

# 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，社社会的ニーズの高度な多様化と限られた資源の効率的利用という，現代の複雑な課題への対応が求められる時代の要請がある．

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 $\epsilon$ 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] [9] [10] [11]．これらの手法は，基本的に従来の数理的アプローチを多目的最適化に適用しようとするものだが，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は複数個あり，いかにして選好解を選び出すかが問題となる．これまでに，この選好解を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 $\epsilon$ 制約法を繰り返し適用する SWT 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されている [12] [13].

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

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

とするとき、$x \in \mathbb{R}^n$ の $p$ 個の関数

$$f_k(x),\ k = 1, \ldots, p \tag{2.2}$$

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 $f_k(x), k = 1, \ldots, p$ 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，

$$a \leq b \iff a_i \leq b_i \ (i = 1, \ldots, p) \tag{2.3}$$

または

$$a < b \iff a_i < b_i \ (i = 1, \ldots, p) \tag{2.4}$$

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，

$$f(x) \triangleq (f_1(x), \ldots, f_p(x)) \tag{2.5}$$

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.
  a. $f(x_1) \leq f(x_2)$ のとき，$x_1$ は $x_2$ に優越すると言う.
  b. $f(x_1) < f(x_2)$ 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.
  a. $x_0$ 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.
  b. $x_0$ 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.
  c. 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 2.1 と図 2.2 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.

また，パレート最適解全体の集合は，一般に「パレートフロント」と呼ばれ，最適化問題において，すべてのパレート最適解がこの前線上に位置する．パレートフロントは，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，各点は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を説明する.

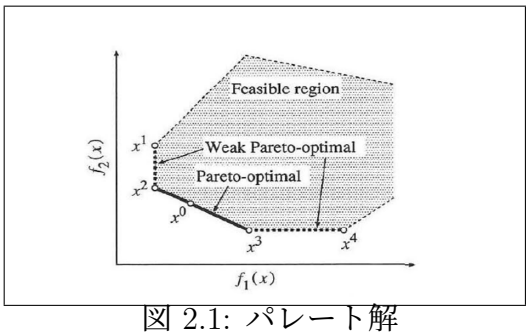


図 2.1: パレート解

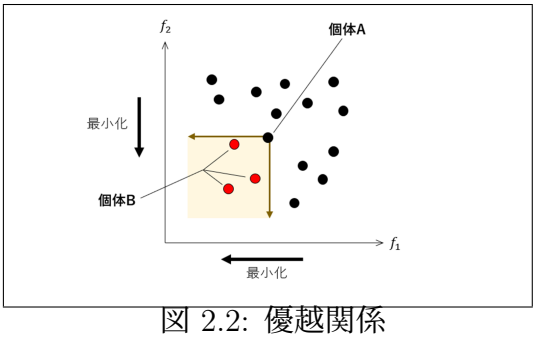


図 2.2: 優越関係

**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

$$E = \{e_1, e_2, \ldots, e_N\}, \tag{2.6}$$

要素の体積の集合

$$V = \{v_1, v_2, \ldots, v_N\}, \tag{2.7}$$

および要素の価値の集合

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

が与えられた場合、次の式 (2.9) および式 (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．

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

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

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：

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

ここでは，ナップザック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，式 (2.6) 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うことができる．

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．

## § 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] [19] [20] [21] [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．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 年代に活用されていた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，式 (2.13) の関係が成り立つとき解 **a** は解 **b** を優越すると定義される．

$$\min\ f_1(x), f_2(x), \ldots, f_M(x) \quad \text{subject to} \quad x \in X, \tag{2.12}$$

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．

$$\forall i, \quad f_i(\mathbf{b}) \geq f_i(\mathbf{a}) \text{ and } \exists i, \quad f_i(\mathbf{b}) > f_i(\mathbf{a}) \tag{2.13}$$

すなわち，解 **a** は、いずれの目的においても解 **b** に劣っておらず，最低でも一つの目的について解 **b** より優れている．このとき，解 **a** は解 **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多目的最適化アルゴリズムの性質と密接に関連していることを示す．二つの例を紹介する

**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．

$$\min\ f_1(x) = x^2 \qquad f_2(x) = (x-2)^2, \tag{2.14}$$

$$\text{subject to} \quad -20 \leq x \leq 20. \tag{2.15}$$

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].

$$\min\ f_1(x_1, x_2) = x_1^2 - 2x_1 + x_2^2, \quad f_2(x_1, x_2) = 9x_1^2 - (x_1 - 1)^2, \tag{2.16}$$

$$\text{subject to} \quad -20 \leq x_1 \leq 20, \quad -20 \leq x_2 \leq 20, \tag{2.17}$$

$$x_1^2 + x_2^2 \leq 25, \quad 3x_1 - 10 \leq 0, \quad x_1 - x_2 + 2 \leq 0, \tag{2.18}$$

次に，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 から 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**

min

$$f_1(x) = x_1 \tag{2.19}$$

$$f_2(x) = g(x)\left(1 - \frac{x_1}{g(x)}\right) \tag{2.20}$$

where

$$g(x) = 1 + \frac{9}{N}\sum_{i=2}^{N} x_i, \tag{2.21}$$

subject to

$$0 \leq x_i \leq 1, \quad \text{for } i = 1, 2, \ldots, N. \tag{2.22}$$

**ZDT1**

min

$$f_1(x) = x_1 \tag{2.23}$$

$$f_2(x) = g(x)\left(1 - \frac{x_1}{g(x)}\right) \tag{2.24}$$

where

$$g(x) = 1 + \frac{9}{N}\sum_{i=2}^{N} x_i, \tag{2.25}$$

subject to

$$0 \le x_i \le 1, \quad \text{for } i = 1, 2, \ldots, N. \tag{2.26}$$

2010 年代では，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 個のテスト問題である DTLZ1 から 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．

$$\min \left\{ f_1(\mathbf{x}) = (1 + g(\mathbf{x})) \prod_{h=1}^{M-1} \cos(x_h) \prod_{h=M}^{d} \cos^2(x_h) \right\} \tag{2.27}$$

$$f_2(\mathbf{x}) = (1 + g(\mathbf{x})) \prod_{h=1}^{M-2} \cos(x_h) \prod_{h=M-1}^{d} \cos^2(x_h) \tag{2.28}$$

$$\vdots \tag{2.29}$$

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

ここで、$g(\mathbf{x})$ は以下のように定義される：

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

**制約条件**：

$$0 \le x_h \le 1, \quad \text{for } h = 1, 2, \ldots, M-1 \tag{2.32}$$

$$0 \le x_j \le 1, \quad \text{for } j = 1, 2, \ldots, d \tag{2.33}$$

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．以下で紹介する手法にはそれぞれが持つ特徴や適用範囲によって，最適化問題において最適解を導出するための異なるアプローチを提供する．多目的最適化問題を解決するためには，問題の特性や求める解の性質を十分に理解し，適切な手法を選択することが重要である．

**重みパラメータ法**

重みパラメータ法は，複数の目的関数を単一の目的関数に変換して最適化を行う方法である．具体的には，各目的関数に重みを付けて，それらを加重和としてまとめ，最小化または最大化するというアプローチである．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：

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

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

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

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 制約法は，複数の目的関数を一つの目的関数に変換するのではなく，ある目的関数を最適化し，他の目的関数を制約条件として扱う方法である．ここでは，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 $\epsilon_k$ で制約したスカラ最小化問題として定式化する．

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

制約条件は次のように表される．

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

このようにして，$\epsilon_k$ 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,，パレート最適集合と呼ぶを形成する．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．例えば，進化的アルゴリズムでは、集団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．これらのアルゴリズムは，自然界の進化過程に触発されたもので，解候補を集団として扱い，遺伝的操作を通じて新たな解を生成し，最適化を行う．進化的アルゴリズムの特徴は，局所解にとどまらず，探索空間全体を効率的に探索できる点である．

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），蟻コロニー最適化（Ant Colony Optimization: ACO），および粒子群最適化（Particle Swarm Optimization: PSO）について解説する．

#### SA

SA は，物理学における焼きなましの過程を模倣した最適化手法であり，温度という概念を導入し，その温度をうまく制御することにより，生成した初期解を改悪も許しながら徐々に改善し，最適解または最適解に近い最良解を導く．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 は次のように表される．

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

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．

$$T_{k+1} = \alpha T_k, \tag{2.39}$$

ここで，$\alpha$ 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．連続したいくつかの温度で受理される回数が規定値より少なければ解は凍結したものと考え，その解を最適解とみなし操作を終了する．

#### GA

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．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．

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

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．また，突然変異を起こす確率は他の操作と比べるととても小さい．なぜなら頻繁に突然変異を起こしているとランダムに次世代の遺伝子を生み出しているだけであり，最適な解を得られないからである．通常 0.1 から 1 %で設定される

$$x_i' = \mathrm{mutate}(x_i). \tag{2.41}$$

**ACO**

ACO は，実際のアリが餌を採集する際の経路形成の方法にヒントを得た探索手法であり，実際のアリが，発見した餌場から巣までを短い距離で往復する経路を見つけるのにフェロモンを介した協調行動を用いている．フェロモンの濃度は次式で定義される.。

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{2.42}$$

ここで，$\tau_{ij}$ は都市 $i$ から都市 $j$ までの道筋に残るフェロモン、$\rho$ はフェロモンの蒸発率，$m$ は探索を行うアリの数を表している．フェロモンの更新式 $\Delta\tau_{ij}^{k}(t)$ は次式で計算する．

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{1}{L_k(t)} & \text{if } i,j \in L_k, \\ 0 & \text{otherwise.} \end{cases} \tag{2.43}$$

このように，ランダムな出発点に配置された蟻たちが，フェロモンという物質を使って自分の通った経路をマーキングする．他の蟻はそのフェロモンの濃度を感知し，フェロモンが多い道を選びやすくなる．このような反復を重ねることで最適解に収束することを期待する．

**PSO**

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．

$$x_p^{k+1} = x_p^k + v_p^{k+1} \tag{2.44}$$

$$v_p^{k+1} = wv_p^k + c_1 r_1 (x_{pbest}^k - x_p^k) + c_2 r_2 (x_{gbest} - x_p^k) \tag{2.45}$$

ここで，$w$ は慣性重み，$c_1$ と $c_2$ は学習係数，$r_1$ 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．

# 第3章

# 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

群知能は，自然界における集団の協調的な行動を模倣した最適化手法である．特に，鳥の群れ，魚の群れ，アリのコロニーなど，個々の個体が互いに影響を与え合いながら，全体として複雑な行動や問題解決能力を発揮する現象に基づいている．このアプローチは，分散型で自己組織的なシステムを活用して，問題解決を行うことが特徴である．個々のエージェントは，単純なルールに従って動きながら，集団全体として非常に高度な行動を実現する．そのため，個々のエージェントが直接的なコミュニケーションを取らなくとも，協力し合って問題解決を達成できる．この自己組織的な動きが，群知能の最も魅力的な点であり，特に複雑で動的な環境において強力な探索能力を発揮する．群知能を活用した最適化手法は，特に探索空間の大きい複雑な問題に対して有効であり，局所的な解に陥るリスクを避けつつ，効率的に最適解を探索する能力を持っている．その中でも PSO は，群知能を基盤にした最適化アルゴリズムの代表的な手法として広く認知されている．

PSO は，群の中の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，[27] 社会的行動に基づいて開発され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や，複数の目的を考慮する多目的粒子群最適化（Multi-objective Particle Swarm Optimization: MOPSO）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，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．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 3.1．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].

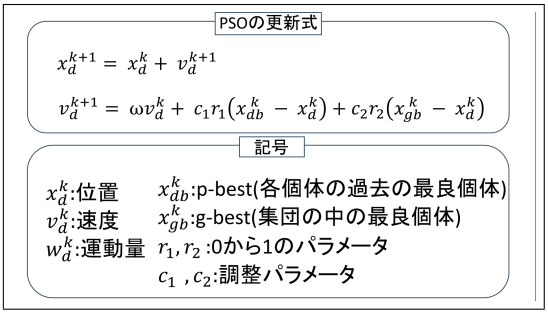


図 3.1: PSO の更新式

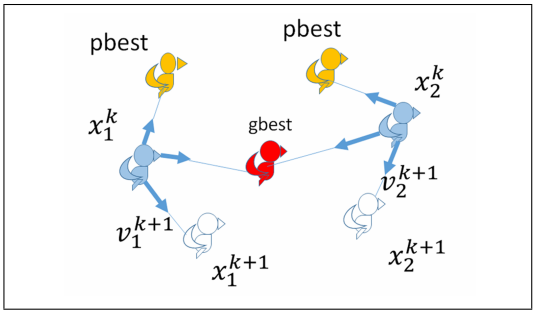


図 3.2: PSO の探索の様子

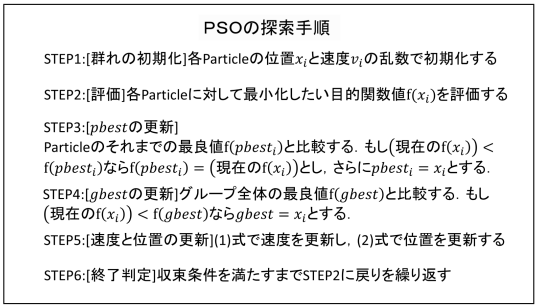


図 3.3: PSO の探索手順

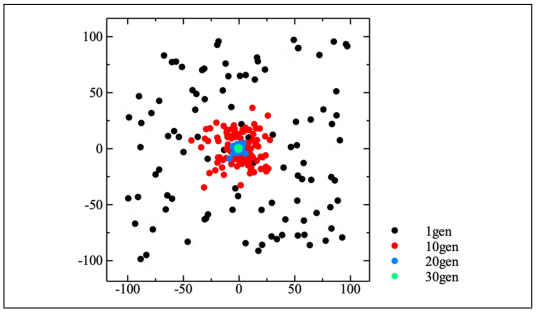


図 3.4: 粒子の動き $(x^2 + y^2)$

$$x_p^{k+1} = x_p^k + v_p^{k+1} \tag{3.1}$$

$$v_p^{k+1} = wv_p^k + c_1 r_1 (x_{pbest}^k - x_p^k) + c_2 r_2 (x_{gbest} - x_p^k) \tag{3.2}$$

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，$p_{\mathrm{best}}$ に向かう $c_1 r_1 (x_p^k - x_p^k)$，$g_{\mathrm{best}}$ 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 $wv_p^k$ 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 $p_{\mathrm{best}}$ と $g_{\mathrm{best}}$ 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．

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，

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

なるモデルを導入することができる．両辺微分することにより，明らかに

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

またそれぞれの関数 $F_p$ と $C$ は以下のようになる．

$$F_p(x_p,t) = c_1\left(x_p(T_p(t)) - x_p\right) \tag{3.5}$$

$$C(x_p,t) = c_2\left(x_Q(t)(T_0(t)) - x_p\right) \tag{3.6}$$

ここで、$T_p(t)$ や $(Q(t),T_0(t))$ はは次のように定義される：

$$T_p(t) = \arg\min_{\tau}\left\{E(x_p(\tau))|0 \le \tau \le t\right\} \tag{3.7}$$

$$(Q(t),T_0(t)) = \arg\min_{(q,\tau)}\left\{E(x_q(\tau))|q = 1,\ldots,P\right\} \tag{3.8}$$

$T_p(t)$ は時間が連続化したことにより，初期時刻 $\tau = 0$ から現時刻 $\tau = t$ の連続時間区間 $[0,t]$ 上の個体 $p$ の軌道 $x_p(\cdot)$ に対して関数 $E$ を評価し，現時刻 $t$ までの間に、その最小値を与える時刻 $T_p(t)$ を意味している．現時刻 $\tau = t$ はこれ自体時間変数として時々刻々変化するので，$E$ の最小値を与える時刻 $T_p$ もこの $t$ の関数として $T_p(t)$ と記してある．

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

$$u_p(t) = x_p(t), v_p(t) = \frac{du_p(t)}{dt} + au_p(t) \tag{3.9}$$

とおいて導入すると，離散時間系に対応した連続系の内部状態表現モデルは，

$$\frac{du_p(t)}{dt} = -au_p(t) + v_p(t) \tag{3.10}$$

$$\frac{dv_p(t)}{dt} = c\left[F_p(u_p(t),t) + C(u_p(t),t)\right] \tag{3.11}$$

が得られる．なお，状態変数 $v_p$ は

$$v_p(t) = \frac{du_p(t)}{dt} \tag{3.12}$$

と一般的においた内部状態変数モデルも考えられるが，式 (3.9) のようにおいた場合，多峰性関数の最適化のベンチマークにおいて大域的最適解の発見率が高かったため，式 (3.9) を採用されている．

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．このままでは，無制約であるが制約の適応については後の章で解説する．

## § 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．具体的には，製品のサイズや重さ，使用される材料の量，製造プロセスにかかる時間やコストなどが制約となり，これらを超えないように設計変数を最適化しなければならない．

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：

$$\min E(x) \tag{3.13}$$

subject to

$$p_i \leq x_p \leq q_i, \quad i = 1, \ldots, n \tag{3.14}$$

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

粒子が制約条件を超えた場合，粒子の位置を制約条件の境界に合わせて修正することで，探索空間を制約領域内に強制的に収束させる．この修正を修正操作や境界修正と呼ぶ．例えば，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．

$$\text{上限を越えた場合: 位置を } x_p = q_i \text{ に設定} \tag{3.15}$$

$$\text{下限を越えた場合: 位置を } x_p = p_i \text{ に設定} \tag{3.16}$$

粒子の位置修正方法は，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，頻繁に粒子が制約の境界に触れる場合，粒子は境界に張り付いたまま動かなくなり，最適解を探索する動きが不自然になる．特に，制約条件が厳しい場合や，粒子が境界に近い場所に集中してしまう場合，次第に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

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．

$$\text{位置修正後の速度: } v_p = -v_p\text{(上限を越えた場合)} \tag{3.17}$$

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．

$$\text{位置修正後の速度: } v_p = -v_p\text{(下限を越えた場合)} \tag{3.18}$$

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．さらに，この反転速度修正に加えて，速度のダンピングや適応的な速度調整を行うことで，粒子の動きをより滑らかにし，制約境界付近での過剰な振動や不安定な動きを抑えることができる．そのため，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，収束が遅くなるというデメリットを調整できる可能性がある．

**ペナルティ関数**

ペナルティ法は，制約違反に対してペナルティを課す方法である．粒子が制約を満たさない場合，そのペナルティ値を目的関数に加算することで，制約違反の解を探索空間から罰する形になる．これにより，最適化は制約を満たす解に近づくように導かれる．ペナルティは通常，制約を越えた分だけ増加し，ペナルティ項の重みを調整することで，制約違反の程度をコントロールできる．

制約条件

$$p_i \leq x_p \leq q_i \qquad i = 1, \ldots, n \tag{3.19}$$

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：

$$P(x) = \sum_{i=1}^{n} [\max(0, x_i - q_i) + \max(0, p_i - x_i)] \tag{3.20}$$

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．

上記のとおり，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：

$$\min\left(E(x) + \rho P(x)\right) \tag{3.21}$$

ここで，$\rho$ 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かを調整するパラメータである．通常，$\rho$ 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**

制約付き最適化問題を直接解くために，与えられた上下限制約領域内での問題の変数を適切な非線形変換を行い，これにより無制約の新たな変数空間にマッピングする．この変換によって，制約条件が自然に取り込まれ，最適化問題が無制約問題として再定義される．新たな無制約変数空間では，PSO アルゴリズムを適用することが可能となり，より効率的な最適解を探索することができる．そのため，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」を考える．非線形変数変換モデルを作成するために，

$$x_p = f_i(y_i) = \frac{q_i + p_i \exp(-y_i)}{1 + \exp(-y_i)} \tag{3.22}$$

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．

$$E(f(y)) = E\left(f\left(\frac{q_i + p_i \exp(-y_i)}{1 + \exp(-y_i)}\right)\right) \tag{3.23}$$

また，$p_i = 0$， $q_i = 1$ の場合，(3.9) は、ニューラルネットワークの出力関数であるいわゆるシグモイド関数であり，$f_i : [-\infty, \infty] \to (0, 1)$ である．

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 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 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フロントと呼ばれる非支配解の集合を得ることである．また，探索途中の優良な解である非劣解を保存するために，アーカイブと呼ばれるレポジトリを有する．そして個体群の中での最良の解 gbest をアーカイブ中の非劣解から選出する．

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と群全体の経験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 は効率良く多目的最適解集合つまりパレート解集合を求めることができる．しかし PSO と同じ難点として，設計変数が多くなると指数関数的に計算時間が増大し，最適化が困難になることが経験的にわかっている以下にアルゴリズムを示す．

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 $x_{\mathrm{pbest}}^k \in \mathbb{R}^m$ と群れで共有するパレート解の位置情報 $x_{\mathrm{gbest}} \in \mathbb{R}^m$ 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 $x_{\mathrm{gbest}}$ 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 $N_P$、反復計算回数 $N_K$、保存点個数最大値 $N_{\max}^R$ を決定し，$x_p^k$，$x_{\mathrm{pbest}}^k$，$v_p^k$ の初期設定を行う．ただし，$p$ は探索点番号、$r$ は保存点番号を表す．$x_p\ (1 \leq i \leq N_I)$ は制限値内で無作為に決定し，$x_{\mathrm{pbest}}^k = x_p^k\ (1 \leq i \leq N_I)$，$g_{\mathrm{best}}(r) = x(i)\ (r = i, 1 \leq r \leq N_I)$，$v(p) = 0\ (1 \leq i \leq N_I)$ とおく．$g_{\mathrm{best}}(r)\ (N_I + 1 \leq r \leq N_{\max}^R)$ については初期値を持たず，保存点個数を $N_R$ とすると初期設定では $N_R = N_I$ である．

$n$ 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 $x_{\mathrm{pbest}}^k$ と群れで共有するパレート解の位置情報 $g_{\mathrm{best}}(r)$ 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．MOPSO では，全ての $g_{\mathrm{best}}(r)$ が存在する $n$ 次元の目的関数空間を任意の数だけ分割するようにハイパーキューブと呼ばれる $n$ 次元立方体を生成する．次に，pbest，gbest，$g_{\mathrm{best}}(r)$ の更新について述べる．

#### gbest の更新と保存

移動ベクトル $v(p)$ を生成する際に必要となる $x_{\mathrm{gbest}}$ を選択する手順は以下の通りである。目的関数空間において、少なくとも 1 つの $g_{\mathrm{best}}(r)$ を含む全てのハイパーキューブに着目し、各ハイパーキューブに属する $g_{\mathrm{best}}(r)$ の個数が最も多いハイパーキューブを選びハイパーキューブを 1 つ指定し，ハイパーキューブ $h$ とする。ハイパーキューブ $h$ の中から $g_{\mathrm{best}}(r)$ を 1 つ無作為に選択し，選択された $g_{\mathrm{best}}(r)$ を $x_{\mathrm{gbest}}$ とする．このように $g_{\mathrm{best}}(h)$ を選択することで，探索点を $g_{\mathrm{best}}(r)$ の密度の低い領域へと引き寄せ，広範囲の探索を行うことができるという効果がある．

#### 新たな探索点の生成

$k+1$ 回目の探索において，探索空間内では $p$ 番目の探索点 $x^k_{\mathrm{pbest}}$ が，式 (3.24) で記述される移動ベクトル $v^k_p$ に従って，式 (3.24) で示される新たな位置へと移動する．

$$v^{k+1}_p = w \cdot v^{k+1}_p + \mathrm{rand1}() \cdot (x^k_{\mathrm{pbest}} - x^k_p) + \mathrm{rand2}() \cdot (x_{\mathrm{gbest}} - x^k_{\mathrm{pbest}}) \tag{3.24}$$

$$x^{k+1}_p = x^k_p + v^{k+1}_p \tag{3.25}$$

式 (3.24) に関して，$w$ は慣性値，rand1()，rand2() は 0 から 1 までの一様乱数を表す．右辺第 1 項は前回移動した方向への慣性を表すベクトルである．右辺第 2 項は探索点を自身の持つ最良解の位置へ引き寄せるベクトル，右辺第 3 項は探索点を $x_{\mathrm{gbest}}$ の位置へ引き寄せるベクトルである．

### $x^k_{\mathbf{pbest}}$ と $g_{\mathbf{best}}(r)$ の更新

このステップは、以下の (A)〜(F) のサブステップから構成されている．

**Step(A)**
$x^{k+1}_p$ が全ての目的関数値に対して $x^k_{\mathrm{pbest}}$ よりも優れている場合は、$x^k_{\mathrm{pbest}}$ を $x^{k+1}_p$ に更新する．

**Step(B)**
$x^{k+1}_p$ がある目的関数値に対して $x^k_{\mathrm{pbest}}$ よりも優れているが，他の目的関数値に対して $x^k_{\mathrm{pbest}}$ よりも劣っている場合は、$x^k_{\mathrm{pbest}}$ を $x^{k+1}_p$ 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 $x^{k+1}_p$ 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 $x^{k+1}_p$ に更新する．それ以外は後の処理 (E) で削除される．

**Step(D)**
$x^{k+1}_p$ が全ての $g_{\mathrm{best}}(r)$ 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 $g_{\mathrm{best}}(r)$ 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)**
$g_{\mathrm{best}}(r)$ が増加すると，計算資源を圧迫するため， $g_{\mathrm{best}}(r)$ は最大サイズ（$N^R_{\max}$ ）が設定されている． $g_{\mathrm{best}}(r)$ 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．

$$d_j = \sum_{m=1}^{M} \frac{E^m_{I^{j+1}_m} - E^m_{I^{j-1}_m}}{E^m_{\max} - E^m_{\min}} \quad j \in [2, l-1]$$

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 $g_{\mathrm{best}}(r)$，$E^m$ 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.

**Step (F)**

指定された反復計算回数 $N_K$ に到達するまでは Step 1–2 $\sim$ Step 1–6 の手順を繰り返す．計算回数が $N_K$ に到達すると探索を終了し、最終的に残った $g_{\mathrm{best}}(r)$ の集合をパレート最適解集合とする.

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.

# 第4章

# 提案手法

## § 4.1　勾配を考慮した粒子群最適化

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：

$$x(k+1) = x(k) + \Delta x(k+1) \tag{4.1}$$

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

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

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． 式 (4.3) の右辺第 2 項を次のように定義する：

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

これにより，(4.2) 式は次のように一般化される：

$$\Delta x(k+1) = \lambda \Delta x(k) + F(x(k), k) \tag{4.5}$$

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．式 (4.1) と式 (4.2) から $\Delta x(k)$ を消去すると、次のような式が得られる：

$$x(k+1) - (1+\lambda)x(k) + \lambda x(k-1) = F(x(k), k) \quad k = 1, 2, \ldots\ldots \tag{4.6}$$

この式は，特性根として 1 と $\lambda$ 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，式 (4.6) を解いて、級数形式で書き直すと次のようになる：

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

また，$F(x(0),0)$ については次のように定義される：

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

この式から明らかに，過去の $F(x(i),i),\ i = 0,1,\cdots,k-1$ を重み $\lambda^{k-i}$ 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：

$$F(x,k) = -c_3\nabla E(x) \tag{4.9}$$

とおくと，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 (4.9) 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 $\nabla E$ は以下のように計算される：

$$\nabla E \approx \frac{E(x_k) - E(x_{k-1})}{x_k - x_{k-1}} \tag{4.10}$$

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：

$$v_p^{k+1} = wv_p^k + c_1r_1(x_{pbest} - x_p^k) + c_2r_2\left(x_{gbest} - x_p^k\right) - \left(c_3\nabla E\right) \tag{4.11}$$

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**

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

条件：

$$-600 \le x_i \le 600, \quad f_{\min}(0,\ldots,0) = 0 \tag{4.13}$$

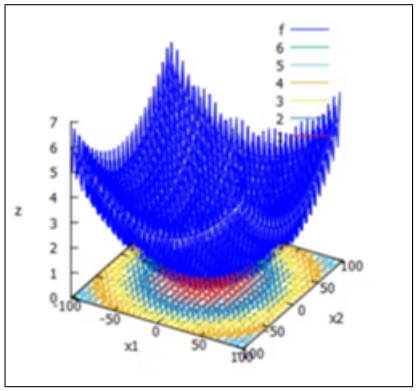


図 4.1: Griewank 関数

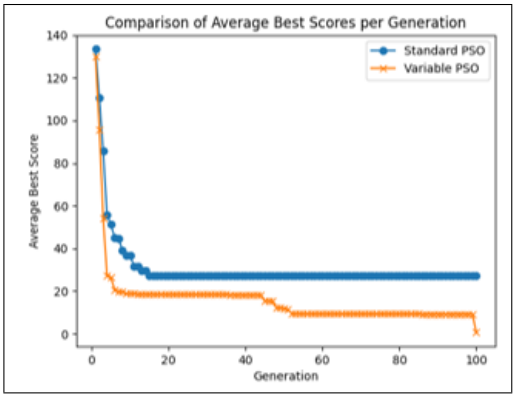


図 4.2: 比較

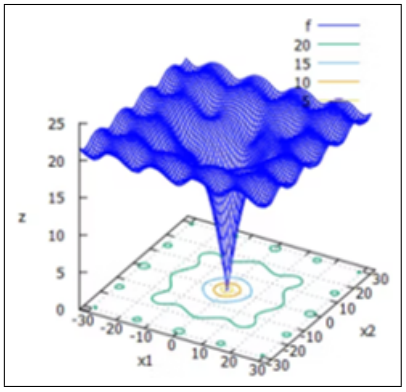


図 4.3: Ackley 関数

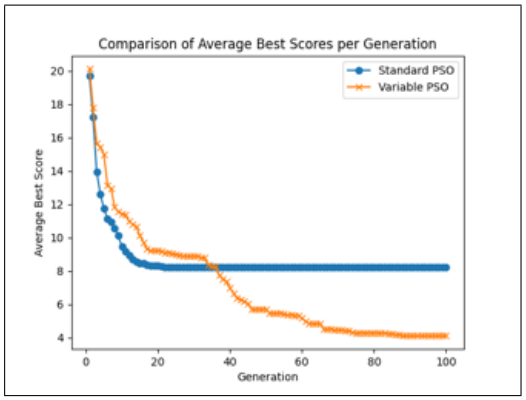


図 4.4: 比較

**Ackley 関数**

$$f(x_1,\ldots,x_n) = 20 - 20\exp\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) + e - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) \tag{4.12}$$

条件：

$$-32.768 \le x_i \le 32.768, \quad f_{\min}(0,\ldots,0) = 0 \tag{4.13}$$

**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ると考察できる．

## § 4.2　上下制約付き勾配MOPSO

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．s 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．さらに，前節で紹介した従来の 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 PSO を MOPSO に適応し，上下制約を有する勾配 MOPSO について詳細に述べ，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．さらに，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ことを目指す．

### 上下制約付き勾配 MOPSO の重要な要素

上下制約付き勾配 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

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

$$\min E_m(x) = \min\{E_1(x), E_2(x), \ldots, E_m(x)\} \tag{4.14}$$

$$p_i \leq x_i \leq q_i, \quad i = 1, 2, \ldots, n \tag{4.15}$$

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，最適化が進行させる．

まず，提案手法である上下制約付き勾配 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．PSO の更新式を力学系モデルとみなした連続 PSO モデルを考えると，

$$\frac{dx_p(t)}{dt} = c\int_0^t e^{-a(t-\tau)}\left[F_p(x_p(\tau),\tau) + C(x_p(\tau),\tau) - \sum_{i=1}^{m}\nabla E(x_p(\tau),\tau)\right]d\tau \tag{4.16}$$

$$\frac{d^2x_p(t)}{dt^2} + a\frac{dx_p(t)}{dt} = c\left[F_p(x_p(t),t) + C(x_p(t),t) - \sum_{i=1}^{m}\nabla E(x_p(t),t)\right] \tag{4.17}$$

であり，またそれぞれの関数は以下のようになる．

$$F_p(x_p, t) = c_1(x_{pbest} - x_p) \tag{4.18}$$

$$C_p(x_p, t) = c_2(x_{gbest} - x_p)) \tag{4.19}$$

$$\nabla E(x_p, t) = c_3 \frac{\partial E(x_p, t)}{\partial x_p} \tag{4.20}$$

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 (4.14) と式 (4.15) を利用する．

$$\min E_m(x) = \min\{E_1(x), E_2(x), \ldots, E_m(x)\} \tag{4.14}$$

$$p_i \leq x_i \leq q_i, \quad i = 1, 2, \ldots, n \tag{4.15}$$

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，

$$x_p = f(y_i) = \left( \frac{q_i + p_i \exp(-y_i)}{1 + \exp(-y_i)} \right) \tag{4.21}$$

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，

$$\min E_m(f(y)) \tag{4.22}$$

を得ることができる．よって式 (4.17) とに対応させると，

$$\frac{d^2 y_p(t)}{dt^2} + a \frac{dy_p(t)}{dt} = c \left[ F_p(y_p(t), t) + C(y_p(t), t) - \nabla E(y_p(t), t) \right] \tag{4.23}$$

またそれぞれの関数は以下のようになる．

$$F_p(y_p; t) = c_1(y_{pbest} - y_p) \tag{4.24}$$

$$C_p(y_p; t) = c_2(y_{gbest} - y_p)) \tag{4.25}$$

$$\nabla E(y_p; t) = c_3 \frac{\partial E(y_p; t)}{\partial y_p} \tag{4.26}$$

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．

$$u_p(k+1) = (1 - a\Delta T)u_p(k) + \Delta T v_p(k) \tag{4.27}$$

$$v_p(k+1) = v_p(k) + c\Delta T \left[ F_p(u_p(k), k) + C(u_p(k); k) - \sum_{i=1}^{m} \nabla E_i(u_p(k), k) \right] \tag{4.28}$$

$$F_p(k, k) = c_1(u_{pbest} - u_p(k)) \tag{4.29}$$

$$C_p(k, k) = c_2(u_{gbest} - u_p(k)) \tag{4.30}$$

$$\nabla E(k, k) = c_3 \frac{\partial E(k, t)}{\partial k} \tag{4.31}$$

$$u_{pbest} = \begin{cases} u_p(k) & \text{if } \forall m, E_m(u_p(k)) < E_m(u_{\text{pbest}}) \\ u_p(k) & \text{with probability } ramdom \text{ if } \exists m, E_m(u_p(k)) < anyE_m(u_{\text{pbest}}) \\ u_{\text{pbest}}(k) & \text{otherwise} \end{cases} \tag{4.32}$$

$$u_{gbest} = \text{random}(\arg H_{\max} |H_i|) \tag{4.33}$$

$$x_p^i(k) = f_i(u_p^i(k)) = \left( \frac{q_i + p_i \exp(-u_p^i(k))}{1 + \exp(-u_p^i(k))} \right); i = 1, \ldots, n \tag{4.34}$$

このように，粒子の位置更新式 (4.27) 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 (4.28) では，現在の速度に対して $u_{pbest}$ 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，勾配を考慮し，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の式 (4.32) 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から選ばれ，そこからランダムに最適解が選ばれる式 (4.33) ．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

多目的最適化問題は，単一の目的関数ではなく，複数の目的関数が同時に最適化される問題である．これにより，勾配を計算する方法が単目的最適化問題とは大きく異なる．単目的最適化では，問題が 1 つの目的関数に収束するため，最適解に向かうための方向を示す勾配ベクトルは一意に定義できる．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可能であり，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た．

しかし，多目的最適化においては，複数の目的関数が同時に存在するため，勾配情報が複数次元となり，それぞれの目的関数に対応する勾配が独立して計算される．そのため，多目的最適化問題では，単一の勾配を定義することができず，目的関数間でどのようにこれらの勾配を統合・組み合わせるかが大きな課題となる．また，各目的関数が互いに異なる方向性を持ち，最適化の進行においてトレードオフが生じることにある．例えば，ある目的関数を改善するために変数の方向を変更すると，その結果として他の目的関数の値が悪化する場合がある．従って，MOPSO においては，勾配情報を単独で使用するのではなく，各目的関数の勾配をどのように扱うかが非常に重要である．複数の勾配をどのように統合するかは，多目的最適化問題の解法における重要な要素となり，探索空間内で最適解を見つけるための有効な手段を提供する鍵となる．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．このアプローチでは，複数の目的関数から得られ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，各目的関数が提供する勾配情報を加算することで，粒子が最適化すべき方向を示す総合的な勾配情報を求め，これを使用して粒子の速度を更新する．具体的には，任意の目的関数 $E_1(x), E_2(x), \ldots, E_m(x)$ 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：

$$v_p^{(k+1)} = w v_p^{(k)} + c_1 r_1 \left( x_{\text{pbest}}^{(k)} - x_p^{(k)} \right) + c_2 r_2 \left( x_{\text{gbest}}^{(k)} - x_p^{(k)} \right) + c_3 r_3 \sum_{i=1}^{m} \nabla E_i(x_p) \tag{4.35}$$

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，各目的関数が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 $E_1(x), E_2(x), \ldots, E_m(x)$ に対して、勾配ベクトルを次のように正規化する：

$$\nabla \hat{E}_i(x) = \frac{\nabla E_i(x)}{\|\nabla E_i(x)\|} \tag{4.36}$$

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：

$$v_p^{(k+1)} = w v_p^{(k)} + c_1 r_1 (x_{\text{pbest}}^{(k)} - x_p^{(k)}) + c_2 r_2 (x_{\text{gbest}}^{(k)} - x_p^{(k)}) + c_3 r_3 \sum_{i=1}^{m} \nabla \hat{E}_i(x_p) \tag{4.37}$$

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ことがある．

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 $E_1(x), E_2(x), \ldots, E_m(x)$ 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：

$$v_p^{(k+1)} = wv_p^{(k)} + c_1 r_1 \left( x_{\text{pbest}}^{(k)} - x_p^{(k)} \right) + c_2 r_2 \left( x_{\text{gbest}}^{(k)} - x_p^{(k)} \right) + c_3 r_3 \frac{1}{N} \sum_{i=1}^{m} \nabla E_i(x_p) \quad (4.38)$$

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を取得する．

勾配の平均値を各次元に適用する．これにより全ての次元が同じ平均勾配を持つようになる．最終的なベクトル grad_mean_vector は、各次元が $\nabla \bar{E}$ であるベクトルである．

$$\text{grad\_mean\_vector} = \left[ \nabla \bar{E}, \nabla \bar{E}, \ldots, \nabla \bar{E} \right] \quad (4.39)$$

ここで、grad_mean_vector は、全次元に同じ平均勾配 $\nabla \bar{E}$ を適用した結果である．

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．

今回は，複数の勾配情報の扱いについて提案したが，多目的最適化問題に対する一つの主要な解法としてスカラー化手法がある．これは，あるパラメータを用いて，もとの多目的な問題を単一目的関数をもつ最適化問題に変換し解を得る方法である [31] [32]．代表的なスカラー化手法として，目的関数の非負線形結合を最適化する加重平均法という方法があるが，問題の形式やパラメータによっては，非有界な問題に変換されてしまう．そのため，線形加重和の代わりに劣線形関数を用いた劣線形スカラー化手法などが提案されている [33] [34]．このように，多目的最適化問題に勾配情報を利用したい場合は，目的関数自体を単一に変換する方法を考えられる．

しかし，意思決定の際に決めなければならないパラメータの適正な値は事前にわからない．その欠点を克服するため，多目的最適化問題に対するスカラー化しない手法も多い．

.

# 第5章

# 数値実験並びに考察

## § 5.1　数値実験の概要

本章では，提案手法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，提案手法が従来の 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明確に示すことである．特に，提案手法が解空間内での探索精度や収束特性においてどのように改善をもたらすのかを評価する．

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， 単峰性で非凸型のパレートフロントを有する問題

$$f_1(x) = x_1$$

$$g(x) = 1 + 9(n-1)\sum_{i=2}^{n} x_i$$

$$h(f_1, g) = 1 - (\frac{f_1}{g})^2$$

$$0 \leq x_1 \leq 1, \quad i = 2, \ldots, n$$

**Optimum**

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

**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

$$f_1(x) = x_1$$

$$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i^2 - 10\cos(4\pi x_i)\right)$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$$

$$0 \leq x_1 \leq 1, \quad -10 \leq x_i \leq 10, \quad i = 2, \ldots, n$$

**Optimum**

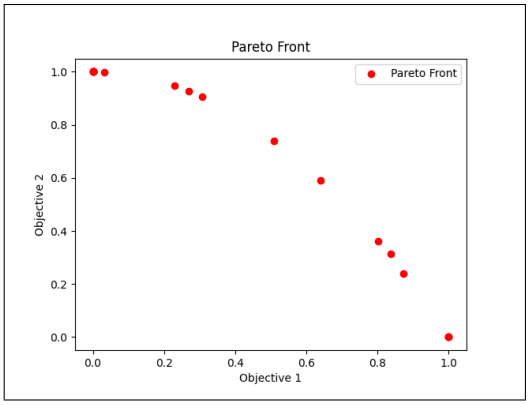


図 5.1: zdt2-MOPSO

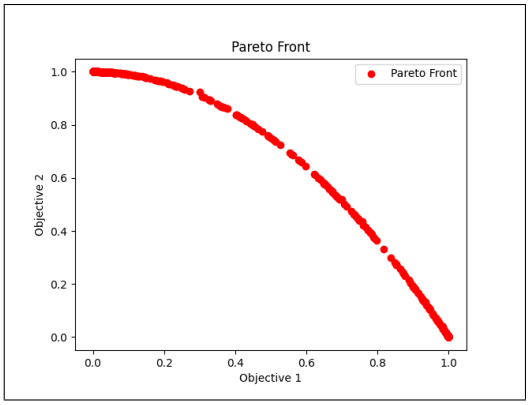


図 5.2: zdt2-勾配和 MOPSO

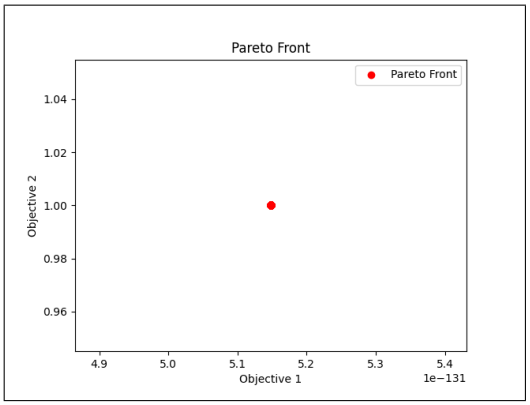


図 5.3: zdt2-勾配平均 MOPSO

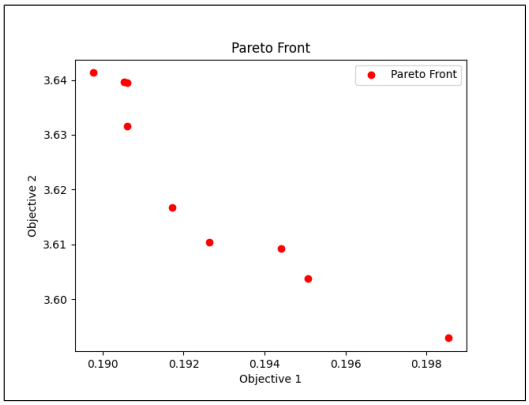


図 5.4: zdt2-勾配正規化 MOPSO

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

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: $G = 500$　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

**パラメータ**

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

パラメータについては，実装の際に用いた式 (4.27) から式 (4.32) を参照している．

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．

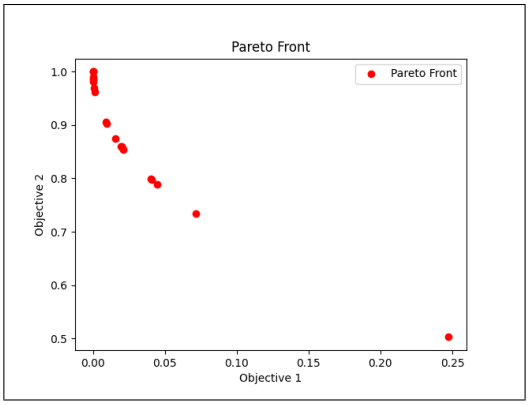


図 5.5: zdt4-MOPSO

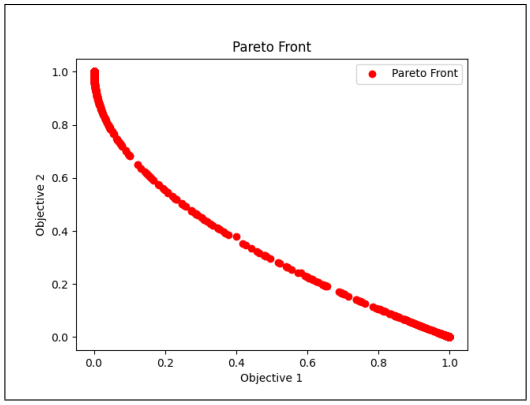


図 5.6: zdt4-勾配和 MOPSO

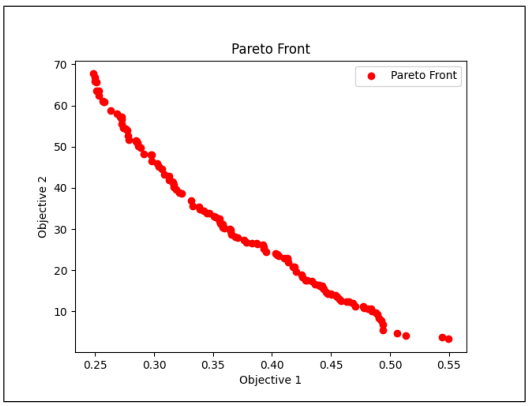


図 5.7: zdt4-勾配平均 MOPSO

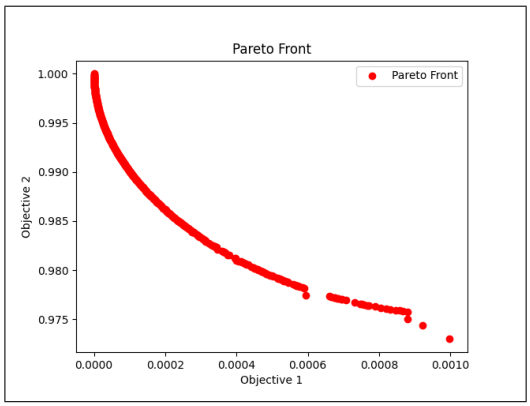


図 5.8: zdt4-正規化 MOPSO

### 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: $G = 500$　　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

### パラメータ

$a = 1.0,\quad T = 0.3,\quad C = 0.2,\quad c_1 = 2.0,\quad c_2 = 2.0,\quad c_3 = 0.2\quad \mathrm{w} = 0.5$

パラメータについては zdt4 と同様に，実装の際に用いた式 (4.27) から式 (4.32) を参照している．

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えている．勾配の正規化を利用した方法では，一見してパレートフロントを取れているように，見えるが一方の目的関数において，探索がし切れておらず zdt-4 のパレートフロントとしては間違ったものが出力されている．通常の MOPSO と勾配の平均を活用した MOPSO も同様に探索が行えていない．

## § 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，最適化過程においてバランスの取れた解を導きやすくなる．このようにして，最適解に収束するまでの過程がスムーズに進行し，優れた探索能力を発揮することが可能となると考えられる．

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，平均化により，特定の次元における急激な変化が平滑化され，粒子が誤った方向に進むこととなるため，最適解に到達する能力が低下することが考えられる．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．

このように，実験では，勾配の和を利用したアプローチが効果的であることが確認された．一方で，勾配の平均化や正規化によって次元間の特性が無視され，粒子の進行方向が誤ってしまう可能性があることが示唆された．これは，多目的最適化問題において異なるスケールや重要性を持つ次元を適切に反映するためには，単純な平均化では不十分であることを意味している．勾配の和を用いた方法は，効果的ではあるもののそれぞれの目的関数の勾配の特徴を完全に網羅できているわけではない．そのため，勾配の特徴を残しつつ粒子の計算に組み込む方法の提案がさらに必要だと考えられる．

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 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. 次元数は 10、最大世代数は 500 に設定する．また，粒子数は 500 の場合と 300 の場合の両方を考慮する．これにより、粒子数の影響を比較し、最適化の効率を評価する．さらに，問題の次元数を 2 から 10 まで変化させ，計算時間の変化を比較することで，次元数が増加した際の計算負荷の違いを明確にする．次元数が増えることで，MOPSO や NSGA-II それぞれの手法がどのように反応するか，またどちらの手法が次元数の増加に対してより効率的に動作するかを検証する．パラメータについては，5.1 節と同様のものを使用している．

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

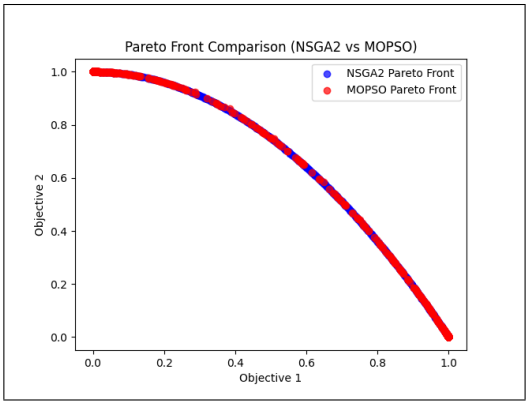


図 5.9: zdt2-粒子 500-比較

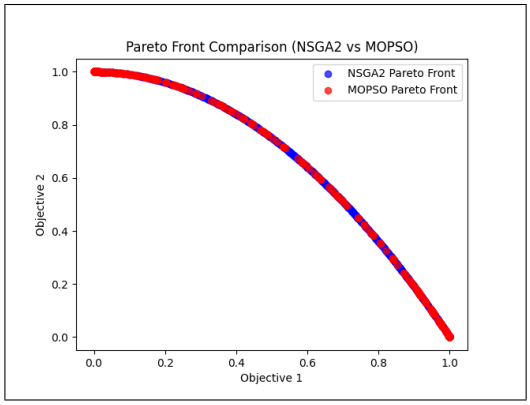


図 5.10: zdt2-粒子 300-比較

表 5.1: zdt2 の次元数ごとの計測時間

| 次元数 | 2 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|
| MOPSO | 46.03$s$ | 74.20$s$ | 200.50$s$ | 259.81$s$ | 426.88$s$ |
| NSGA | 41.68$s$ | 43.40$s$ | 43.91$s$ | 44.48$s$ | 44.35$s$ |

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で，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ういうことである．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．これらの事から，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．

計算時間において zdt2 の次元数が小さい場合は，MOPSO と NSGA-II の差はあまりないが，NSGA-II の計算時間が，次元数にあまり依存していないのに対して，MOPSO は次元が増えるごとに計算時間が肥大していっている．これは，MOPSO の性質として，次元数の増加に伴って計算量が急増することに起因している．次元が増えると，粒子が探索する範囲が広がり，それに伴って粒子群の相互作用や個々の粒子の位置更新に必要な計算量が増加する．一方，NSGA-II は，集団ベースのアプローチであり，個々の個体に対する計算は次元数の影響を直接受けるものの，次元数が大きくなっても集団サイズの影響の方が大きいため、次元数の増加による計算時間の急激な増加は見られにくいと考えられる．

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

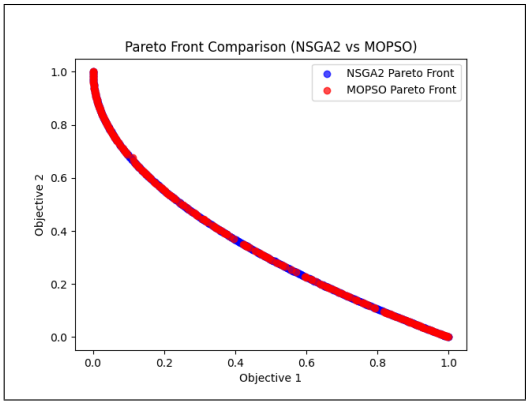


図 5.11: zdt4-粒子 500-比較

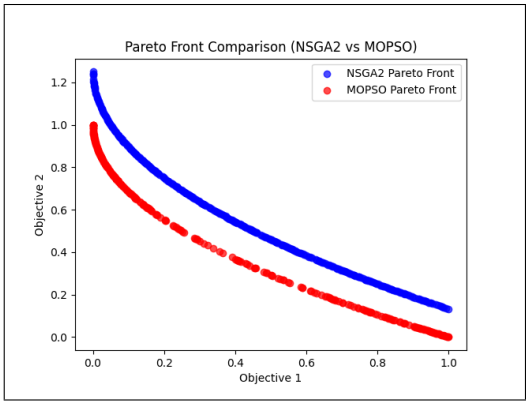


図 5.12: zdt4-粒子 300-比較

表 5.2: zdt4 の次元数ごとの計測時間

| 次元数 | 2 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|
| MOPSO | 219.04$s$ | 231.11$s$ | 492.68$s$ | 506.06$s$ | 517.97$s$ |
| NSGA | 41.45$s$ | 43.55$s$ | 44.82$s$ | 47.18$s$ | 47.23$s$ |

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．また，zdt2 と比べて zdt4 に対して MOPSO の計算時間が大きくなっている理由に関しては，単純に zdt と比べ zdt4 の問題がより複雑な構造を持っていることと zdt2 が単峰性の関数であり，勾配情報をよりうまく活用できているのではないかと考えられる．

# 第6章

# おわりに

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上下制約を有する勾配MOPSO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ることを確認した．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．

本研究で提案するシステムでは、多目的最適化問題において勾配情報を活用する手法として，各目的関数の勾配の和を用いましたこの方法により，粒子は各目的関数の改善を同時に考慮し，最適解に向けて効率よく探索を行うことができる．このアプローチは，目的関数が複数であっても，全体として最適化された解を導く能力を有しており，多目的問題における複雑な相互作用を考慮した解探索が可能となる．しかしながら，いくつかの課題も浮き彫りとなっており，異なる目的関数が異なるスケールや重要性を持つ場合，勾配の和を使用することで特定の目的関数に偏りが生じる可能性があり，この点に関しては今後の改善が求められる．今後の研究では、多目的最適化問題における勾配情報の取り扱いをさらに精緻化し，最適化過程におけるバランスを取る方法を検討することが重要である．

さらに，本研究にはいくつかの未解決の課題も存在している．具体的には，MOPSOアルゴリズムの次元数が増加するにつれて計算コストが増大するという問題が依然として残っている．単純なベンチマーク関数においては一定の効果を確認したが，実際の応用問題においては，勾配の活用方法やアーカイブの更新方法，粒子の選択方法に関するさらなる改善が求められる．特に，大規模な問題に対する計算負荷を低減し，より高速かつ高精度な探索を実現するための工夫が必要である．

また，勾配MOPSOアルゴリズムの性能をさらに向上させるためには，粒子の位置更新や移動規則，さらには適応的なアルゴリズムの調整が求められる．今後は，実際の応用に向けた検証を行い，複雑な多目的最適化問題に対しても高い性能を発揮できるような改良を進めることが重要となる．さらに，他の最適化アルゴリズムとの組み合わせや，異なる問題設定における効果的な手法の開発を目指して，研究を深めていく必要がある．

総じて，本研究は勾配情報を活用した多目的最適化手法における新たな道筋を示すものであり，今後の研究や実務において、非常に有望な技術基盤を提供するものと考えられる．最適化問題の解決に向けた新たな視点とアプローチを提供し，より高精度で効率的な最適解を得るための手段を開拓することができた．

# 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．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．

2025 年 2 月

柴原壮大

# 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 Systemics, 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. Prempain, 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.

[31] J. Jahn, ”Scalarization in vector optimization,” *Mathematical Programming*, Vol. 29, No. 2, pp. 203–218, 1984.

[32] 中山，谷野：多目的計画法の理論と応用，計測自動制御学会, 1994.

[33] D. T. Luc, *Theory of Vector Optimization*, Springer, 1989.

[34] Y. Ogata, Y. Saito, T. Tanaka and S. Yamada, ”Sublinear scalarization methods for sets with respect to set-relations,” *Linear and Nonlinear Analysis*, Vol. 3, No. 1, pp. 121–132, 2017.