

# 多目的マルチタスクナップサック問題の提案

大阪府立大学 工学域

電気電子系学類 情報工学課程 計算知能工学研究グループ

1161201192 瓜田 俊貴

## 1. はじめに

複数の評価基準を同時に考慮して最適化する問題は、多目的最適化問題(Multiobjective Optimization Problem: MOP)と呼ばれる[1]。一般に、MOP では、ある評価基準を改善すると、他の評価基準が改悪されるトレードオフの関係が成り立つ。この場合、すべての評価基準を同時に最適化する単一の最適解は存在せず、実行可能なすべての解に優越されないパレート最適解が複数存在することが多い。しかし、すべてのパレート最適解集合を獲得することは計算コストの観点から非効率的である。そのため、MOP における解法の目標はパレート最適解の集合を近似する解の獲得となる。

近年、複数の異なるMOP(以下、タスクと呼ぶ)を、進化計算を利用して一度の試行で同時に最適化する、進化型多目的マルチタスク最適化手法(Evolutionary Multiobjective Multitasking: EMOMT)が盛んに研究されている[2]-[4]。EMOMT の特徴は、明示的あるいは黙示的に関連性のあるタスクで情報共有を行い、探索の効率化を図ることである。EMOMT では、一つの個体群の中に異なるタスクを解く個体を共存させることにより、タスク間での情報共有を可能にしている。情報共有を適切に行うことで、EMOMT が独立探索手法より優れた探索性能を示すことが明らかにされている[2], [3]。

一般に、EMOMT の探索性能はベンチマーク問題によって調査される。そこで、適切にEMOMT の探索性能を調査するためには、問題の性質が調査された様々なベンチマーク問題が必要である。そのようなベンチマーク問題として、Yuan らによって、決定変数が実数値であるベンチマーク問題が提案されている[4]。しかし、問題の性質が調査された、決定変数が離散値であるベンチマーク問題は著者の知る限り提案されていない。決定変数が離散値の問題では、決定変数が連続値の問題で用いられる交叉操作や突然変異操作とは異なる操作が用いられる。そのため、決定変数が離散値の問題では、EMOMT の挙動は決定変数が連続値の問題と異なると考えられる。したがって、決定変数が離散値の問題に対するEMOMT の探索性能が調査できるベンチマーク問題が必要である。

そこで本研究では、決定変数が離散値であるMOPの多目的ナップサック問題(Multiobjective Knapsack Problem: MOKP)をマルチタスク問題に拡張した、新

たなベンチマーク問題を提案する。提案問題では、元のMOKP に対して変更を加えた、明示的な関係のあるMOKP を組み合わせて、シングルタスクのMOKP をマルチタスク問題に拡張する。また、元のMOKP に加える変更量をパラメータで決定することで、各タスクの関係性が調整可能な問題に設計する。そして、提案問題の各タスクの最適解の類似度およびパラメータの変更による影響を調査した後、提案問題を用いた数値実験より、EMOMT の探索性能を調査する。

提案問題によって、決定変数が離散値の問題に対する、EMOMT の探索性能の調査が新たに可能になる。さらに、提案問題では各タスクの類似度がパラメータで調整可能なため、任意の類似度を持つタスクに対するEMOMT の探索性能を調査することが可能になる。そのため、決定変数が離散値の問題に対して優れた探索性能を示すEMOMT の開発が可能になる。また、提案問題を既存のEMOMT で解くことにより、既存のEMOMT が持つ今まで明らかにされなかった特徴を発見することが期待される。

本論文の構成は以下の通りである。まず2章で多目的最適化問題を説明する。次に、3章でMOKP を説明する。4章では、提案するベンチマーク問題を定義し、5章でベンチマーク問題の最適解の類似性を調査する。そして、6章で数値実験により提案問題に対するEMOMT の探索性能を調査する。最後に、7章で本論文の結論と今後の課題について述べる。

## 2. 多目的最適化問題

$k$  個の目的からなるMOP は、次式で表される。

$$\text{Maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})), \quad (1)$$

$$\text{subject to } \mathbf{x} \in \mathbf{X}, \quad (2)$$

ここで、 $\mathbf{f}(\mathbf{x})$  は  $k$  次元目的ベクトル、 $f_i(\mathbf{x})$  は最大化されるべき  $i$  番目の目的関数、 $\mathbf{x}$  は決定変数ベクトル、 $\mathbf{X}$  は決定変数空間内の実行可能領域を表す。

MOP では複数の目的関数が存在するため、二つの解を比較した際にどちらの解が優れているか判断できない場合が存在する。たとえば、解  $\mathbf{x}, \mathbf{y}$  の任意の目的関数  $i, j$  が式(3)の関係にある場合、一般にどちらの解が優れているという優劣をつけることはできない。このような関係を非劣と呼ぶ。

$$f_i(\mathbf{x}) < f_i(\mathbf{y}) \text{ and } f_j(\mathbf{x}) > f_j(\mathbf{y}). \quad (3)$$

一方、二つの解が式(4)の関係を満たす場合、解  $y$  は解  $x$  よりも優れた解とみなすことができる。このような関係を  $y$  は  $x$  を優越すると呼ぶ。

$$\forall i, f_i(x) \leq f_i(y) \text{ and } \exists j, f_j(x) < f_j(y). \quad (4)$$

また、解  $x$  が他のすべての実行可能解に優越されないとき、解  $x$  をパレート最適解と呼ぶ。

### 3. 多目的ナップサック問題

MOKP は、Zitzler らによって提案された決定変数が 0 または 1 の離散値で表現される MOP である[5]。多目的ナップサック問題の定義式を式(5)-(7)に示す。

$$\text{Maximize } f(x) = (f_1(x), f_2(x), \dots, f_m(x)), \quad (5)$$

$$\text{subject to } \sum_{j=1}^n w_{ij}x_j \leq c_i, i = 1, 2, \dots, m, \quad (6)$$

$$\text{where } f_i(x) = \sum_{j=1}^n p_{ij}x_j, i = 1, 2, \dots, m, \quad (7)$$

ここで、 $m$  は目的数、 $n$  はアイテム数、 $x$  は決定変数ベクトル、 $c_i$  は  $i$  番目のナップサックの容量である。また、 $w_{ij}$ 、 $p_{ij}$  はそれぞれ  $i$  番目のナップサックに対応する  $i$  番目のアイテム集合(以下、アイテム集合  $i$  と表記)における  $j$  番目のアイテムの重さおよび価値である。

文献[5]では、式(6)の制約を満たさない実行不可能解を実行可能解に修復する操作として、Greedy Repair が提案されている。Greedy Repair は、利得率の低いアイテムから順番に取り除くことにより、実行不可能解を実行可能解に修復する操作である。 $j$  番目のアイテムの利得率  $q_j$  を式(8)に示す。

$$q_j = \max_{i=1}^m \left( \frac{p_{ij}}{w_{ij}} \right). \quad (8)$$

本問題の目標は、すべてのナップサックの容量を超えず、各ナップサック内のアイテムの価値の総和を最大化することである。

MOKP は、アイテムの価値・重さ・ナップサックの容量の値のみで問題が定義される。加えて、MOKP の評価関数は、式(7)より各決定変数の線形和で表される。そのため、MOKP は単純な最適化問題であるといえる。しかし MOKP の各目的は、計算量理論において、NP-hard と呼ばれる問題のクラスに属するため、最適解を求めるのが非常に困難とされている。したがって、MOKP のパレート最適解の獲得も非常に困難であると考えられる。

### 4. 提案するベンチマーク問題

本研究では、実装および問題の解析の容易さを考慮して、各タスクに MOKP を設定したマルチタスクベンチマーク問題を提案する。さらに、タスク間に明示的な関係を与えて問題の特徴づけるため、一方の MOKP は他方の MOKP に対し変更を加えた

MOKP とする。具体的には、3 章で述べた文献[5]の MOKP をタスク 1 とし、タスク 1 のナップサックの容量やアイテム集合に変更を加えた問題をタスク 2 として設定する。

本章では、タスク 1 のナップサックの容量を変更してタスク 2 を作成する Capacity Scaling 問題、およびタスク 1 の一部のアイテムの価値を変更してタスク 2 を作成する Profit Inversion 問題を提案する。また、タスク 1 に加える変更量はパラメータによって調整する。

#### 4.1 Capacity Scaling 問題

Capacity Scaling 問題は、ナップサックの最大容量が異なる MOKP を同時に解くマルチタスク問題である。ナップサックの最大容量が各タスクで異なるとき、各タスクで選択されるアイテム数が異なる。そのため、本問題で各タスクが異なるタスクの解を利用するには、他タスクの解に対し追加でアイテムを選択する操作、あるいは複数のアイテムを選択しないようにする操作が必要である。

タスク 1 のナップサックの最大容量を  $c_i^{T1}$  として、タスク 2 の各ナップサックの最大容量  $c_i^{T2}$  を式(9)で定義する。

$$c_i^{T2} = \alpha c_i^{T1}, i = 1, 2, \dots, m, \quad (9)$$

ここで、 $\alpha$  はナップサックの容量の変更率を決定する (0.0, 2.0) のハイパーパラメータである。容量倍率  $\alpha$  が 1 から離れるほど、タスク間でナップサックの容量の差が大きくなり、選択されるアイテム数の差が大きくなる。よって、容量倍率  $\alpha$  が 1 から離れるほどタスク間の類似度が小さくなると予想される。

#### 4.2 Profit Inversion 問題

Profit Inversion 問題は、一部のアイテムの価値が異なる MOKP を同時に解くマルチタスク問題である。一部のアイテムの価値を変更すると、各タスクで特有のアイテムが生まれる。そのため、本問題で各タスクが異なるタスクの解を利用するには、各タスクで共通するアイテムの情報のみを利用する操作が必要である。

本問題では、タスク 1 の特定のアイテム集合に対して、一部のアイテムに価値反転操作を適用した問題をタスク 2 とする。価値反転操作の概略図を図 1 に示す。ただし、 $p_{ij}^{\text{old}}$  は変更前のアイテム  $j$  の価値、

$p_{ij}^{\text{new}}$  は変更後のアイテム  $j$  の価値、 $p_{\max}$  および  $p_{\min}$  はそれぞれ価値の取り得る最大値、最小値である。この操作によって、価値の高い(低い)アイテムが、価値の低い(高い)アイテムになる。

タスク 1 のアイテム集合  $i$  に価値反転操作を適用してタスク 2 を作成するとき、タスク 1 のアイテム集合  $i$  のアイテム  $j$  の価値を  $p_{ij}^{T1}$  として、タスク 2 のアイテム集合  $i$  のアイテム  $j$  の価値  $p_{ij}^{T2}$  を式(10)で定

義する.

$$p_{ij}^{T_2} = \begin{cases} (p_{\max} + p_{\min}) - p_{ij}^{T_1}, & \text{if } j=1, 2, \dots, \lfloor n\beta \rfloor \\ p_{ij}^{T_1} & \text{otherwise} \end{cases}, \quad (10)$$

ここで,  $\beta$ はタスク間で価値の異なるアイテムの割合を決定する(0.0, 1.0]のハイパーパラメータである. 価値変更割合 $\beta$ が1に近いほど, 各タスクで共通のアイテム数が減少し, 各タスクで情報を共有できるアイテム数が減少する. そのため, 価値変更割合 $\beta$ が大きくなるほど, タスク間の類似度が大きくなると予想される.



図 1: アイテムの価値反転操作の概略図

## 5. ベンチマーク問題の性質調査

EMOMT の探索性能は, 各タスクの最適解の類似度に大きく影響を受けることが報告されている[3]. したがって, 提案問題が EMOMT の探索性能に与える影響を明確にするため, 各タスクの最適解の類似度を明らかにする必要がある.

そこで, 本章では提案するベンチマーク問題における, 各タスクの最適解の類似性を調査する. また, パラメータの変更によって, 最適解の類似性がどのように変化するか調査する.

### 5.1 最適解の類似性の調査方法

本節では, 各タスクの最適解の類似性として, タスク 1 およびタスク 2 の最適解が選択するアイテムの差異を調査する. すなわち, 最適解が選択するアイテムの差異を, 各タスクの最適解の類似度として考える.

本調査は, 独立探索手法の MOEA/D[6]により得られた近似最適解(以降, 近似解と表記)を用いて行う. 選択するアイテムの差異は, タスク 2 の各近似解と最近傍であるタスク 1 の近似解(以下, タスク 1 の最近傍解と表記)で調査する. 具体的には, タスク 2 の近似解が選択しておりタスク 1 の最近傍解が選択していないアイテムの数, タスク 1 の最近傍解が選択しておりタスク 2 の近似解が選択していないアイテムの数を調査する. なお, 近似解間の距離を計算する距離関数はハミング距離を用いる.

本節で調査するベンチマーク問題の設定を表 1, 独立探索手法の設定を表 2 に示す. Profit Inversion 問題では, 価値反転操作をタスク 1 のアイテム集合 2 に適用してタスク 2 を作成する. なお, 各タスクの MOKP のアイテム集合およびナップサックの容量の設定は, MOKP の提案論文[5]に準拠して, アイテムの価値と重さは[10, 100]のランダムな整数値に, 各ナップサックの容量は対応するアイテム集合の重さの総和の 1/2 に設定する. また, 式(6)の制約を

満たさない実行不可能解は, 式(8)の利得率に基づく Greedy Repair によって実行可能解に修復する.

表 1: ベンチマーク問題設定

目的数 $m$	2
アイテム数 $n$	500
容量倍率 $\alpha$	{1.05, 1.10, 1.15, 1.20, 1.25}
価値変更割合 $\beta$	{0.05, 0.10, 0.15, 0.20, 0.25}

表 2: 独立探索手法設定

スカラー化関数	Tchebycheff 関数
個体群サイズ	200
近傍サイズ	(個体群サイズ)/10
交叉手法	Uniform Crossover (適用確率: 0.9)
突然変異手法	Bitflip Mutation (適用確率: 1/n)
評価回数	100,000 回

### 5.2 Capacity Scaling 問題の最適解の類似性

Capacity Scaling 問題における, タスク 2 の近似解とタスク 1 の最近傍解との差異を図 2 に示す. 図 2 では, タスク 2 の各近似解がタスク 1 の最近傍解と異なるアイテム数をプロットしている. つまり, 原点から離れた点はタスク 2 の近似解とタスク 1 の最近傍解の類似度が低いことを示す. さらに, 横軸から離れた点ほどタスク 2 の近似解でのみ選択されるアイテム数が増加, 縦軸から離れた点ほどタスク 1 の最近傍解でのみ選択されるアイテム数が増加することを示す.

図 2 より, Capacity Scaling 問題では, タスク 2 の近似解は, タスク 1 の最近傍解が選択するアイテムをほとんど選択しており, タスク 2 の近似解のみが選択するアイテムがあることがわかる. 加えて, 同じ容量倍率 $\alpha$ におけるタスク 2 の各近似解は, 原点からほぼ等距離に分布しているため, タスク 1 の最近傍解に対して同様の類似度であることがわかる. また, 容量倍率 $\alpha$ の増加に伴い, タスク 2 の近似解でのみ選択されるアイテム数が増加する.

以上より, Capacity Scaling 問題では, 各タスクの近似解で選択されるアイテム数が異なることが明らかとなった. また, ナップサックの容量が大きいタスク 2 の近似解は, ナップサックの容量が小さいタスク 1 の近似解で選択するアイテムをすべて選択している. これは, ナップサックの容量の変更により選択されるアイテム数は増加するが, 各アイテムの価値と重さは各タスクで共通であり, 選好されるアイテムは変わらないためと考えられる.

また, Capacity Scaling 問題では, 容量倍率 $\alpha$ によってタスク 2 の近似解でのみ選択されるアイテム数が変化した. したがって, 容量倍率 $\alpha$ を調整することで, 各タスクの近似解の類似度が調整可能であることが確認された.

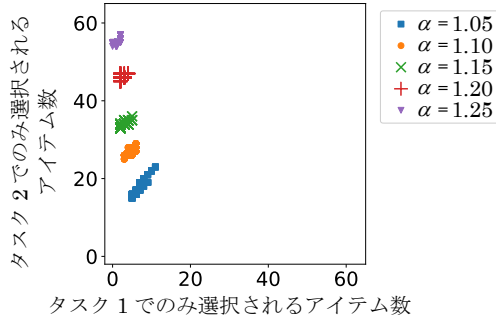


図 2: Capacity Scaling 問題におけるタスク 2 の最適解の類似度

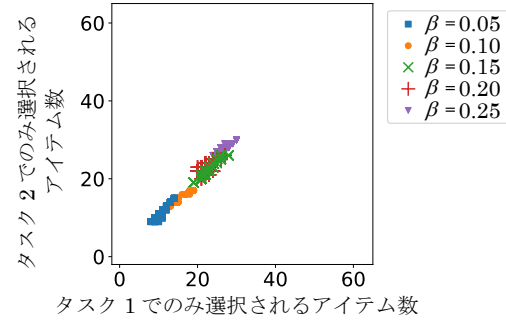


図 3: Profit Inversion 問題におけるタスク 2 の最適解の類似度

### 5.3 Profit Inversion 問題の最適解の類似性

Profit Inversion 問題における、タスク 2 の近似解とタスク 1 の最近傍解との差異を図 3 に示す。

図 3 より、Profit Inversion 問題では、タスク 2 の近似解のみが選択するアイテムがあることがわかる。また、Capacity Scaling 問題とは異なり、タスク 1 の最近傍解のみが選択するアイテムもあることがわかる。加えて、同じ価値変更割合  $\beta$  において、タスク 2 の各近似解は、原点からの距離が同じ位置に分布していない。したがって、タスク 2 の各近似解のタスク 1 の最近傍解に対する類似度は、各近似解で異なることがわかる。また、価値変更割合  $\beta$  の増加に伴い、タスク 1 の最近傍解でのみ選択されるアイテム数および、タスク 2 の近似解でのみ選択されるアイテム数が増加する。

以上より、Profit Inversion 問題では、各タスクの近似解で選択されるアイテムの種類が一部異なることが明らかとなった。これは、価値反転操作の適用により、アイテムの選好が変化したことが理由に挙げられる。価値反転操作の適用により、各タスクで特有のアイテムが生成される。そのため、選択されるアイテムの選好が変化し、選択されるアイテムの種類が一部異なったと考えられる。

また、Profit Inversion 問題では、価値変更割合  $\beta$  によって各タスクの近似解で異なるアイテム数が変化した。したがって、価値変更割合  $\beta$  を調整することで近似解の類似度が調整可能であることが確認された。

## 6. 数値実験

### 6.1 数値実験設定

数値実験では、提案したベンチマーク問題を用いて EMOMT の探索性能を評価する。EMOMT には情報共有操作としてタスク間交叉を利用する MO-MFEA[2]、情報共有操作として移住操作を利用する島モデル[3]を用いる。また、独立探索手法には NSGA-II[1]を用いて EMOMT と比較する。各アルゴリズムの設定を表 3 に示す。交叉操作ならびに突然変異操作は、決定変数が離散値の最適化問題で用い

られる Uniform Crossover および Bitflip Mutation を使用する。ベンチマーク問題のパラメータは、容量倍率  $\alpha$  を  $\{1.10, 1.20\}$  に、価値変更割合  $\beta$  を  $\{0.10, 0.20\}$  に設定する。目的数およびアイテム数は、5 章と同様の設定とする。実験の試行回数は 31 回に設定し、アルゴリズムの性能は、 $(0.0, 0.0)$  を参照点とする Hypervolume[5]で評価する。なお Hypervolume は、目的関数空間における獲得した解の最適解への収束性、獲得した解の多様性を同時に評価できる評価指標である。Hypervolume が大きいほど、獲得した解が最適解に収束しており、多様であることを示す。

表 3: 数値実験設定

共通実験設定	
個体群サイズ	100 (各タスク)
交叉操作	Uniform Crossover (適用確率: 0.9)
突然変異操作	Bitflip Mutation (適用確率: $1/n$ )
試行回数	31 回
評価回数	100,000 回 (各タスク)
目的数 $m$	2
アイテム数 $n$	500
容量倍率 $\alpha$	$\{1.1, 1.2\}$
価値変更割合 $\beta$	$\{0.1, 0.2\}$
MO-MFEA のパラメータ設定	
$rmp$	0.9
島モデルのパラメータ設定	
移住個体数	10 個体
移住間隔	5 世代

### 6.2 数値実験結果

Capacity Scaling 問題において、各アルゴリズムで得られた個体群の Hypervolume 値の中央値を表 4 に、Profit Inversion 問題において、各アルゴリズムで得られた個体群の Hypervolume 値の中央値を表 5 に示す。有意水準 0.05 の Wilcoxon の順位和検定を行い、NSGA-II に対して統計的有意差があり優れている結果を赤字、統計的有意差があり劣っている結果を青字で示す。



表 4: Capacity Scaling 問題における  
獲得した個体群の Hypervolume 値の中央値

アルゴリズム名	$\alpha = 1.05$		$\alpha = 1.10$		$\alpha = 1.15$		$\alpha = 1.20$		$\alpha = 1.25$	
	タスク 1	タスク 2	タスク 1	タスク 2	タスク 1	タスク 2	タスク 1	タスク 2	タスク 1	タスク 2
NSGA-II	3.75E+8	3.98E+8	3.75E+8	4.22E+8	3.75E+8	4.45E+8	3.75E+8	4.70E+8	3.75E+8	4.93E+8
MO-MFEA	3.75E+8	3.97E+8	3.72E+8	4.17E+8	3.71E+8	4.40E+8	3.70E+8	4.17E+8	3.70E+8	4.87E+8
島モデル	3.71E+8	3.88E+8	3.71E+8	4.10E+8	3.71E+8	4.34E+8	3.71E+8	4.10E+8	3.71E+8	4.82E+8

表 5: Profit Inversion 問題における  
獲得した個体群の Hypervolume 値の中央値

アルゴリズム名	$\beta = 0.05$		$\beta = 0.10$		$\beta = 0.15$		$\beta = 0.20$		$\beta = 0.25$	
	タスク 1	タスク 2	タスク 1	タスク 2	タスク 1	タスク 2	タスク 1	タスク 2	タスク 1	タスク 2
NSGA-II	3.75E+8	3.73E+8	3.75E+8	3.77E+8	3.75E+8	3.79E+8	3.75E+8	3.76E+8	3.75E+8	3.75E+8
MO-MFEA	3.76E+8	3.77E+8	3.81E+8	3.82E+8	3.83E+8	3.84E+8	3.80E+8	3.81E+8	3.78E+8	3.79E+8
島モデル	3.73E+8	3.73E+8	3.77E+8	3.78E+8	3.79E+8	3.80E+8	3.75E+8	3.78E+8	3.74E+8	3.76E+8

表 4 より, Capacity Scaling 問題では独立探索手法である NSGA-II が EMOMT より探索性能が優れていることがわかる. 一方 Profit Inversion 問題では, 表 5 より EMOMT が NSGA-II より探索性能が優れている. また, パラメータを変更した場合においても同様に, Capacity Scaling 問題では NSGA-II が, Profit Inversion 問題では EMOMT が優れた探索性能を示した. したがって, Capacity Scaling 問題では情報共有操作によって探索性能が低下, Profit Inversion 問題では情報共有操作によって探索性能が向上することが明らかとなった.

### 6.3 考察

本節では, 提案問題において情報共有操作により EMOMT の探索性能が向上, または低下した要因を考察する. そこで, 容量倍率  $\alpha$  が 1.10 の Capacity Scaling 問題, 価値変更割合  $\beta$  が 0.10 の Profit Inversion 問題において, 5 章で獲得した近似解に対してタスク間交叉および移住操作を行った際の影響を調査する. タスク間交叉による影響は, 各タスクで得られた近似解を利用して子個体を 10,000 個体生成し, 生成した子個体を各タスクで評価して調査する. 移住操作による影響は, 各タスクで得られた近似解を別タスクに移住した際の, 移住先の目的関数空間の分布により調査する.

#### ・ Capacity Scaling 問題でのタスク間交叉の影響

容量倍率  $\alpha$  が 1.10 の Capacity Scaling 問題において, タスク間交叉により生成された子個体の分布を図 4 に示す. 図 4(a) より, タスク 1 で子個体を評価した場合, 近似解と非劣あるいは近似解を優越するような解がわずかに存在することがわかる. しかし, 図 4(b) より, タスク 2 で子個体を評価した場合, 生成されたすべて子個体は近似解に優越されることがわかる.

したがって, Capacity Scaling 問題ではタスク間交叉によって優れた子個体がほとんど生成されない. そのため, MO-MFEA の探索性能がタスク間交叉に

よって低下し, NSGA-II より劣った探索性能を示したと考えられる.

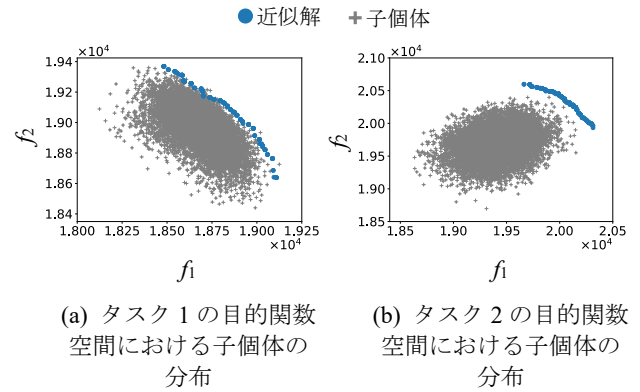


図 4: Capacity Scaling 問題で, タスク間交叉で生成された子個体の目的関数空間の分布

#### ・ Profit Inversion 問題でのタスク間交叉の影響

価値変更割合  $\beta$  が 0.10 の Profit Inversion 問題において, タスク間交叉により生成された子個体の分布を図 5 に示す. 図 5(a) より, Profit Inversion 問題ではタスク 1 で子個体を評価した場合, 近似解に優越されない個体が存在することがわかる. また, 図 5(b) よりタスク 2 で子個体を評価した場合, ごく一部の個体を除き近似解に優越されることがわかる. また, タスク間交叉によって  $f_1$  の値が高く,  $f_2$  の値が低い子個体が多く生成されることがわかる.

したがって, Profit Inversion 問題ではタスク間交叉によって優れた子個体が生成されることがある. そのため, MO-MFEA の探索性能がタスク間交叉によって向上し, NSGA-II より優れた探索性能を示したと考えられる.

#### ・ Capacity Scaling 問題での移住操作の影響

容量倍率  $\alpha$  が 1.10 の Capacity Scaling 問題において, 異なるタスクに近似解を移住した際の目的関数空間における分布を図 6 に示す. 図 6 より, 両方の

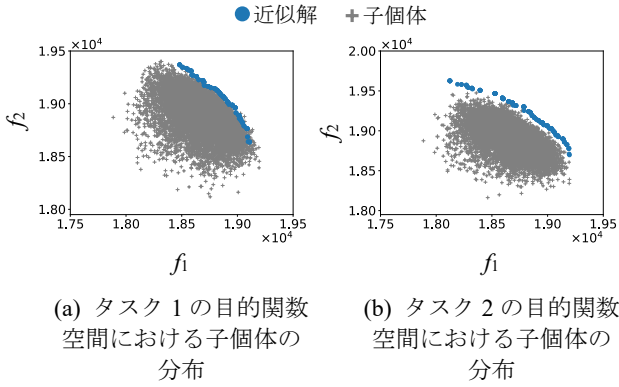


図 5: Profit Inversion 問題で、タスク間交叉で生成された子個体の目的関数空間の分布

タスクにおいて、移住したすべての近似解は移住先のいずれかの近似解に優越されることがわかる。

したがって、Capacity Scaling 問題では移住操作によって優れた解が獲得できない。そのため、島モデルの探索性能が移住操作により低下し、NSGA-II より劣った探索性能を示したと考えられる。

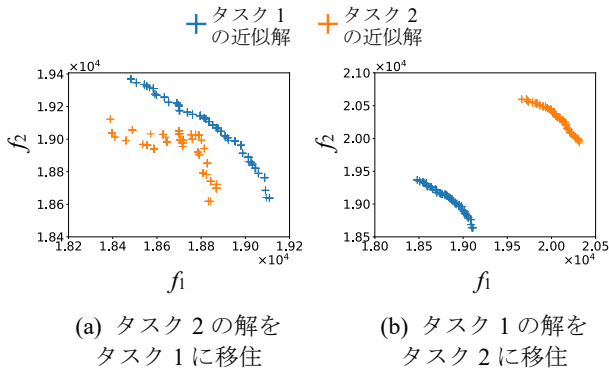


図 6: Capacity Scaling 問題で、異なるタスクに最適解を移住した際の目的関数空間の分布

#### ・ Profit Inversion 問題での移住操作の影響

価値変更割合 $\beta$ が 0.10 の Profit Inversion 問題において、異なるタスクに近似解を移住した際の目的関数空間における分布を図 7 に示す。図 7(a) より、タスク 2 の近似解をタスク 1 に移住した場合、タスク 1 の近似解に優越されない解が存在することがわかる。一方、図 7(b) よりタスク 1 の近似解をタスク 2 に移住した場合、移住した解はタスク 2 のいずれかの近似解に優越されることがわかる。

したがって、Profit Inversion 問題では移住操作に

よって優れた解が獲得できることがある。そのため、島モデルの探索性能が移住操作によって向上し、NSGA-II より優れた探索性能を示したと考えられる。

以上より、5 章および本節の調査によって示された提案問題の特徴を表 6 に示す。提案問題は、タスク 2 の作成方法を変えることにより、各タスクで変更可能な類似度が異なる問題になっている。

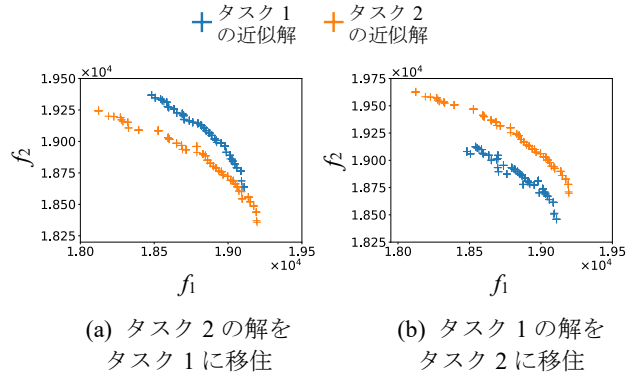


図 7: Profit Inversion 問題で、異なるタスクに最適解を移住した際の目的関数空間の分布

## 7. おわりに

本研究では、EMOMT に対して、決定変数が離散値である新たなマルチタスクベンチマーク問題を提案した。提案問題は、異なる設定の MOKP を組み合わせた 2 種類の問題を提案し、各タスクの類似度をパラメータで変更可能にした。また、各パラメータにおける、提案問題の各タスクの最適解の類似度を調査した。数値実験では、提案問題に対する EMOMT の探索性能が、情報共有が有効な場合は向上、有効でない場合は低下することを示した。

今後の課題として、タスク間で探索難易度が大きく異なる問題や、特定のタスクにのみ情報共有が有効な問題の提案が挙げられる。また、提案問題の性質を利用して最適化を行う EMOMT の提案が求められる。

## 8. 謝辞

本研究ならびに卒業論文の作成にあたって、熱心なご指導およびご支援を頂いた本学大学院工学研究科 電気・情報系専攻 知能情報工学分野の能島裕介准教授、増山直輝助教ならびに Yiping Liu 特認助教に深く感謝の意を表します。

また、平素より有益な助言を頂いた南方科技大学

表 6: 提案問題の特徴

問題名	タスク間の相違	変更可能な類似度	EMOMT の探索性能	
			MO-MFEA	島モデル
Capacity Scaling 問題	ナップサックの容量	タスク 2 の最適解にのみ含まれるアイテム数	独立探索より劣る	独立探索より劣る
Profit Inversion 問題	特定のアイテム集合のアイテムの価値	各タスクの最適解で共通のアイテム数	独立探索より優れる	独立探索より優れる

の石淵久生教授ならびに計算知能工学研究室の皆様  
様に厚くお礼を申し上げます。

---

#### 参考文献

---

- [1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [2] A. Gupta, Y. S. Ong, L. Feng, and K. C. Tan, “Multiobjective multifactorial optimization in evolutionary multitasking,” *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1652–1665, 2017.
- [3] R. Hashimoto, N. Masuyama, Y. Nojima, and H. Ishibuchi, “Effect of solution information sharing between tasks on the search ability of evolutionary multiobjective multitasking algorithms,” in *Proc. of IEEE Symposium Series on Computational Intelligence*, pp. 2671–2678, 2019.
- [4] Y. Yuan, Y. S. Ong, L. Feng, A. K. Qin, A. Gupta, B. Da, Q. Zhang, K.C. Chen, Y. Jin, and H. Ishibuchi, “Evolutionary multitasking for multiobjective continuous optimization : Benchmark problems, performance metrics and baseline results,” arXiv: 1706.02766v1[cs.NE], 2017.
- [5] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach,” *IEEE Transactions on Evolutionary Computation*, pp. 257–271, 1999.
- [6] Q. Zhang and H. Li, “MOEA/D: A Multiobjective evolutionary algorithm based on decomposition,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.