情報処理学会論文誌　数理モデル化と応用　Vol.14　No.3　11–26 (Aug. 2021)

Cartesian Genetic Programmingを用いた転用可能な積み付けアルゴリズムの自動生成

蛭田 悠介[1,a)]　西原 慧[1]　小熊 祐司[2]　藤井 正和[1,2]　中田 雅也[1]

受付日 2020年11月19日，再受付日 2021年1月8日，
採録日 2021年1月20日

概要：本論文では，人手による積み付けパターン設計の自動化を目的とし，Cartesian Genetic Programming（CGP）を用いた転用可能な積み付けアルゴリズムの自動生成技術を提案する．提案法は，積み方の種類と積み付け位置を同時に考慮した選択ルールを定義し，CGP で同ルールの実行順を最適化することで，少ない評価回数で積み付けアルゴリズムを生成できる．数値実験では，提案法が 200 回の評価回数でベースラインと競合する積み付けアルゴリズムを生成可能であり，導出した積み付けアルゴリズムが段ボール箱の種類と個数が異なる問題に転用可能であることを示した．

キーワード：遺伝的プログラミング，アルゴリズム自動生成，積み付け最適化

Automated Construction of Transferable Loading Algorithm with Cartesian Genetic Programming

Yusuke Hiruta[1,a)]　Kei Nishihara[1]　Yuji Koguma[2]　Masakazu Fujii[1,2]　Masaya Nakata[1]

Received: November 19, 2020, Revised: January 8, 2021,
Accepted: January 20, 2021

Abstract: This paper proposes an automatic construction technique of transferable loading algorithms based on Cartesian Genetic Programming. The proposed method aims to construct the loading algorithm with a few hundred fitness evaluations by optimizing the execution order of rules to decide a type of multiple cardboard boxes and their loadable positions simultaneously. Experimental results show that auto-constructed loading algorithms can derive competitive performances to defined baselines under two hundred fitness evaluations on similar problems without any additional fitness evaluation.

Keywords: genetic programming, automatic algorithm construction, loading optimization problem

1. 序論

我が国の物流業界においては労働力不足が深刻化しており，特に現場の定型作業に要する人材の不足が指摘されている [1]．近年では物流の自動化・省人化に向けた製品開発が活発であり，段ボール箱のパレットへの積み付け・積み下ろしや個別商品のピッキングなどの定型作業を自動化するロボットシステムが市場投入されている．一方で，積み付け作業では，段ボール箱の種類や個数に応じて，段ボール箱の積み方（積み付けパターン）を人手で毎回設計する必要がある．積み付けパターン設計においては，輸送コストの抑制を目的とした積載率向上や使用パレット数の削減に加え，段ボール箱の安定性や検品性が同時に配慮される必要がある．現状では，熟練した作業者の経験則を頼りに積み付けパターンが設計されており，これにともなう人的かつ時間的な負担を軽減するためには，作業者の経験則を反映した積み付けパターン設計の自動化が重要となる．

そこで著者らは，上述した人的・時間的な負担を軽減するために，作業者の経験則を反映し，かつ転用可能な積み付けアルゴリズムを自動生成する技術構築を目指している．ここで転用可能とは，アルゴリズム生成に使用した問題とは異なる問題に対して，作業者の経験則を反映した積み付けパターンが得られる性質を指す．具体的な方法論とし

[1] 横浜国立大学
Yokohama National University, Yokohama, Kanagawa 240–8501, Japan
[2] 株式会社 IHI
IHI Corporation, Yokohama, Kanagawa 235–8501, Japan
[a)] hiruta-yusuke-vn@ynu.jp

て，本論文で扱う積み付け問題の特徴を活用し，評価者の感性を反映して最適化可能な対話型進化計算（Interactive Evolutionary Computation: IEC）[2], [3] とアルゴリズムを自動生成可能な Hyper Heuristics（HH）[4] を融合した方法を検討する．本論文で扱う問題の特徴を以下に示す．

(1) 積み付けパターンの良否は定式化困難であるが，作業者は経験則からその良否を判断できる．
(2) 段ボール箱の種類や個数が異なる場合，好適な積み付けパターンも変わる可能性がある．
(3) 積載率が向上する棒積みや荷崩れを防ぐピンホール積みを用いて，同種類の段ボール箱をまとめて積み付けることが推奨される．

特徴 (1) が IEC を用いる理由である．また，段ボール箱の種類や個数の違いごとに IEC を適用する必要がなくなる点で，特徴 (2) が HH により転用可能な積み付けアルゴリズムを導出する動機である．特徴 (3) は本論文の提案法で活用する特徴であり，以降で詳述する．

IEC と HH を融合する場合，アルゴリズムを評価するユーザの評価負担は評価回数に比例して増加する．したがって，ユーザの評価負担を軽減するためには，少ない評価回数でアルゴリズムを自動生成可能な要素技術を構築する必要がある．しかし，2 章で詳述するように，一般的な HH はアルゴリズムの探索空間が膨大化するため数千以上の評価回数を要する．

そこで本論文では，転用可能な積み付けアルゴリズムを少ない評価回数で自動生成可能な技術を提案する．具体的に提案法は，以下の工夫を取り入れる．

(i) 先述した特徴 (3) を活用して，同種類の段ボール箱を棒積みやピンホール積みを用いてまとめ，このまとまりの単位で積み付けを行う．さらに，積み付ける段ボール箱のまとまりの種類，積み付け位置の両基準をあらかじめ定義された複数のルールで選択する．このルールによって，両基準を同時に決定可能な積み付けアルゴリズムを生成でき，両基準を個々に選択する方法と比較してアルゴリズムの探索空間を縮小できる．
(ii) Cartesian Genetic Programming（CGP）[5] を用いて上述した複数のルールの実行順を最適化し，所望の積み付けパターンを出力する積み付けアルゴリズムを生成する．一般的に CGP は進化戦略である $(1+4)$-ES を用いることから [6]，世代ごとに 4 つの子個体を評価するだけでよい．また，ユーザに掲示する積み付けアルゴリズムも 1 世代ごとに 5 つ（1 親個体と 4 子個体）となり，ユーザの評価負担の観点から IEC との親和性が高い．

本論文の構成は次のとおりである．2 章では，積み付け問題およびアルゴリズムの自動構築技術に関する関連研究を述べる．3 章で CGP の概要を導入し，4 章では本論文で扱う積み付け問題の設定を説明した後，問題のモデル化についての諸概念を整理する．なお，本論文では同種類の段ボール箱のまとまりをブロックと呼び，この定義についても 4 章で述べる．5 章で提案法である CGP を用いた積み付けアルゴリズムの自動生成技術について述べる．6 章では，自動生成した積み付けアルゴリズムの性能を評価する．最後に 7 章で本論文の結論を述べる．

2. 関連研究

本章では，積み付け問題およびアルゴリズムの自動構築技術に関する関連研究を述べる．

一般化された積み付け問題として，使用コンテナ数の最少化を目的とするビンパッキング問題がある．コンテナの物理的な次元に応じて 1，2，3 次元ビンパッキング問題に区別される [7], [8], [9]．ビンパッキング問題は NP 困難であり，メタヒューリスティクスを含む近似解法が有効である [10], [11]．たとえば，アントコロニー最適化 [12]，粒子群最適化 [13]，遺伝的アルゴリズム [14], [15]，カッコーサーチ [16] が用いられる．ほかにも，整数計画法 [17] やビンパッキング問題に特化したヒューリスティクスが研究されている [18]．一方で，実応用を想定した問題設定においても研究が行われている．たとえば，積み込みおよび積み下ろし順序 [19]，キークレーン間干渉 [20]，区画コンテナ [21]，コンテナの種類 [22] を考慮した問題が研究されている．

上述した研究は問題ごとに所与の解法アルゴリズムを用いて最適化を行うが，HH は類似問題に転用可能な解法アルゴリズム自体を生成する．近年では，アルゴリズム設計問題に適した遺伝的プログラミングを用いる HH（Genetic Programming-based HH: GPHH）がさかんに研究されている [23], [24], [25], [26]．HH は，定義した解法を選択的に実行するアルゴリズムを生成する方法（Selective HH）とアルゴリズムを一から生成する方法（Generative HH）に大別できる．提案法は GPHH を用いた Selective HH となる．Selective HH は，所与の解法を選択的に組み合わせるため，Generative HH と比較して少ない評価回数で転用可能なアルゴリズムを生成可能である．一方で，Selective HH の欠点として，問題に適した解法を事前定義する必要がある．

HH の分野において，ビンパッキング問題はベンチマーク問題として位置付けられる [27]．Selective HH では，積み付け方の実行手順を求める方法 [28], [29]（[Ross, 03]）や，積み付け状態に対し適応的に解法を選択する方法 [23]（[Nguyen, 11]）がある．これらの研究は提案法に最も関連するが，提案法は積み付ける段ボール箱のまとまりの種類と積み付け位置を同時に決定する点で異なる．Generative HH では，進化的ルール学習や遺伝的アルゴリズムに基づく方法 [30] に加え，特に GPHH に基づく方法がさかんに研究されている [25], [26], [31]．たとえば，積み荷の

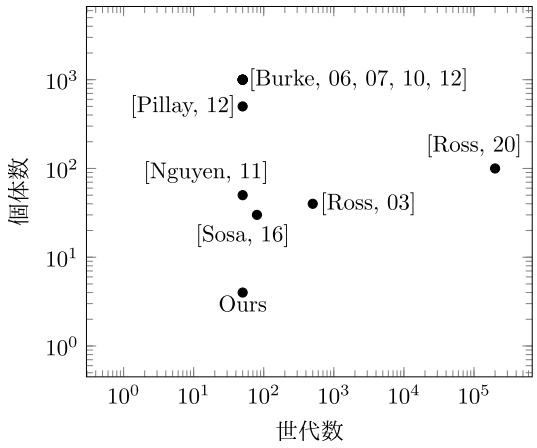


図 1　Hyper Heuristics をビンパッキング問題に適用した関連研究および本論文が用いた世代数と個体数

Fig. 1　The population size and the number of generations used in this paper and existing works regarding to Hyper Heuristics applied to bin packing problems.

種類 [32], [33]（[Burke, 06, 07]）や積み付け位置 [34], [35]（[Burke, 10, 12]）を自動的に決定するために GPHH が用いられる．また，アルゴリズムの探索空間を縮小する方法としては，島モデル型の GPHH [36] や文法構造を持つ進化計算に基づく GPHH [37]（[Sosa, 16]）があり，最適化手法に改良を加えている．ほかにも，積み荷と積み付け位置の選択アルゴリズムを結合する方法 [38]（[Pillay, 12]）や，アルゴリズムの有効性を推定し適用する類似問題を決定する方法 [39]，事前知識を組み込んだ GPHH [40]（[Ross, 20]）も存在する．著者らの調査の限りでは，CGP に基づく GPHH を巡回セールスマン問題へ適用した事例 [41] は存在するが，ビンパッキング問題あるいは積み付け問題に適用した事例はない．

図 1 に HH をビンパッキング問題に適用した関連研究および本論文が用いた世代数と個体数を示す．同図は，著者らが文献より確認できた情報に限定して示しており，Ours は本論文の実験設定（世代数 50，個体数 4）を指す．本論文の問題設定は関連研究と異なるため一概に比較できないが，本論文の実験設定は比較的少ない評価回数（世代数 × 個体数）に設定している．1 章で述べたように，本論文の実験設定のように少ない評価回数を用いることが，IEC におけるユーザの評価負担の軽減に重要となる．

HH と異なるアルゴリズム生成技術として，深層学習を用いる方法がある．Hu らは，深層強化学習を用いて積み付けパターンを出力する方法を提案した [42]．また，Duan らは，積み荷にかかる圧力を最小化する問題を扱い，再帰型ニューラルネットワークを用いて積み荷のパターンを出力する方法を構築した [43]．これらの研究は，評価回数に相当する学習データ数を 150,000 個に設定しており，本論文の実験設定と比べきわめて多くの評価回数を要する．

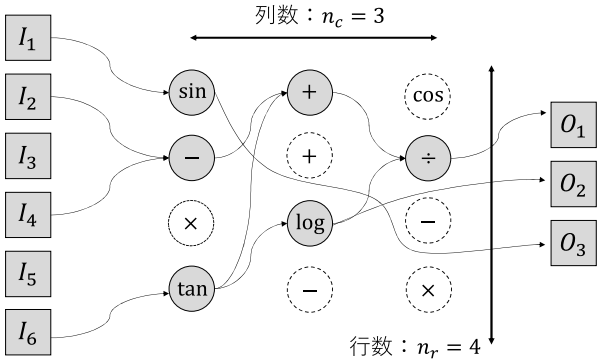


図 2　CGP で用いる遺伝子モデルの表現型の例

Fig. 2　An example of the gene-phenotype in CGP.

3. Cartesian Genetic Programming

本章では，提案法の要素技術として用いる CGP の概要を説明する．

CGP で用いる遺伝子モデル（個体）は，有向非巡回グラフに相当するフィードフォワード型のネットワークで表現される．また，ネットワーク内のノードは再利用可能であり，通常の遺伝的プログラミングで生じやすいブロート問題（決定木の構造が複雑になる問題）が緩和できる利点がある．**図 2** に個体表現を例示する．同図では，6 つの入力 $I_1, ..., I_6 \in \mathbb{R}$ から，3 つの出力 $O_1, O_2, O_3 \in \mathbb{R}$ を求めるネットワークを示している．入出力ノードの間に存在するノードは，定義した数学記号が割り当てられた演算ノードであり，行数 n_r，列数 n_c のグリッド上に配置される．なお，同図では出力に関与しない演算ノードを破線で示している．近年では，$n_r = 1$ とし n_c を増加させる設定が推奨されている [6]．この理由は，$n_r = 1$ に設定することで出力に関与しない演算ノードが削減され CGP の性能が改善すること，n_c を増加させることで個体のネットワークが示す関数の表現能力が向上することがあげられる．出力ノードの値は，入力ノードを含む前段以前のノードから演算された値に設定される．たとえば，図 2 における出力 O_1 と O_2 は，それぞれ $O_1 = \frac{(I_2 - I_4) + \tan(I_6)}{\log(\tan(I_6))}$，$O_2 = \log(\tan(I_6))$ となる．

一般的な CGP は，進化戦略（Evolutionary Strategy: ES）の中でも $(1 + 4)$-ES を用いて個体を最適化する [6], [44]．最初に，初期個体を生成し目的関数を用いて適合度を評価する．そして，適合度が最良値を持つ個体を親個体に設定する．次に，世代数を 1 つ増加させ，親個体に確率 μ で突然変異を適用し，4 つの子個体を生成する．この操作を最大世代数 G まで繰り返す．親個体と子個体の適合度が等しい場合，子個体を次世代の親個体に設定し，探索範囲の重複を抑制する．

4. 問題の設定とモデル化

本章では，本論文で扱う積み付け問題の概要を説明した

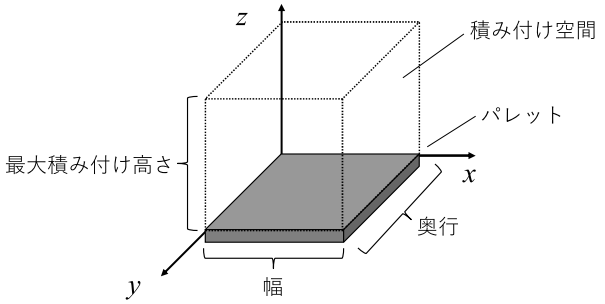


図 3　積み付け空間と座標系の概念図

Fig. 3　The conceptual diagram of the loadable space and the coordinate system.

後，同問題のモデル化として段ボール箱を積み付ける空間と座標系，およびブロックについて定義を述べる．

4.1　問題の概要

本論文で扱う積み付け問題は，異なる種類の段ボール箱を複数のパレットに積み付ける混載積み付け問題である．具体的には，段ボール箱の形状，個数が与えられたもとで，輸送コストの抑制，積み荷の安定性，検品性を総合的に考慮した合理的な積み付けパターンを求めることが目的である．3 次元ビンパッキング問題とは異なり，輸送コストの抑制が唯一の目的とはならない．

輸送コストは，各パレットの積載率の向上および使用パレット数の削減により抑制される．積み荷の安定性は，積み付け高さの抑制と平準化によって向上する．また，積み荷の検品性は，可能な限り同種類の段ボール箱をまとめて積み付けることで向上する．4.2.2 項で詳述するように，棒積みとピンホール積みを用いることでも，積載率と積み荷の安定性がそれぞれ向上することから，実際の積み付け作業では棒積みとピンホール積みが多用される．

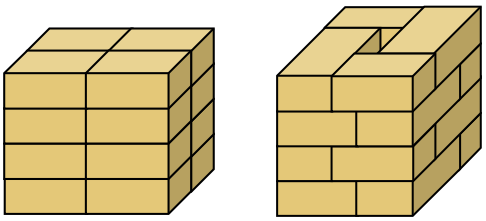
4.2　問題のモデル化

4.2.1　積み付け空間と座標系

段ボール箱を積み付け可能な 3 次元の空間を積み付け空間と呼び，同空間はパレットの幅（x 軸），奥行（y 軸），最大積み付け高さ（z 軸）で規定される．パレットの幅と奥行は，典型的な平面寸法（T11 型）[45] である 1,100 mm，1,100 mm にそれぞれ設定する．また，段ボール箱の最大積み付け高さを 1,800 mm に設定する．**図 3** に示すように，積み付け空間の座標系は，正面からみてパレットの左，奥，下の隅を原点とし，原点から右方向，手前方向，上方向をそれぞれ x, y, z 軸方向とする左手系を採用する．z 軸の原点はパレットの表平面に設定し，パレットの厚みは加味しない．各パレットは同一の積み付け空間を持つ．

4.2.2　アイテムとブロック

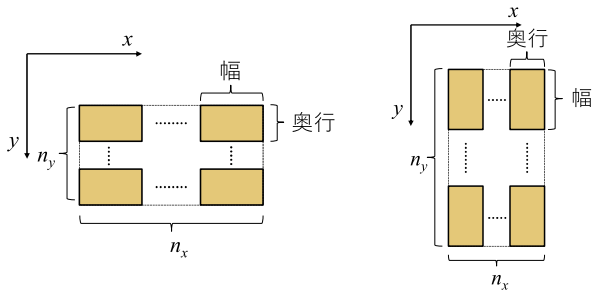
段ボール箱を計算上の概念としてアイテムと呼ぶ．アイテムは複数種類から成り，種類ごとの幅，奥行，高さが既知



(a) 棒積みの例　　(b) ピンホール積みの例

図 4　ブロックの構築例

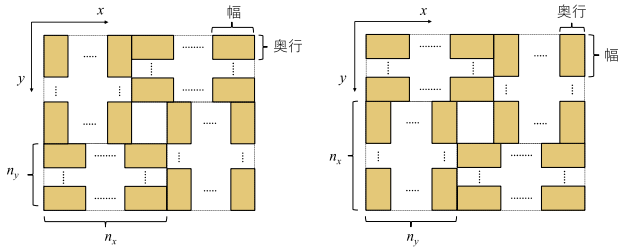
Fig. 4　Examples of block construction.



(a) 元の向きで配置　　(b) 90° 回転して配置

図 5　棒積みブロックにおける水平方向のアイテム配置

Fig. 5　Horizontal item-placements in a parallel block.



(a) 奇数段　　(b) 偶数段

図 6　ピンホール積みブロックにおける水平方向のアイテム配置

Fig. 6　Horizontal item-placements in a pinhole block.

である直方体形状を持つ．ブロックは同種類のアイテムを複数積み付けたまとまりであり，棒積みあるいはピンホール積みを施した直方体形状を持つ．

図 4 (a) と**図 5** に示すように，棒積みはアイテムの向きを揃える積み方であり，各段で x 軸方向に n_x 個，y 軸方向に n_y 個のアイテムを積み付ける．なお，アイテムを水平方向に 90° 回転させた積み方も可能である（図 5 (b)）．棒積みは積載率の向上に寄与するが，水平方向の揺れに対して不安定である．図 4 (b) に示すピンホール積みは各アイテムが奇数段・偶数段で互いに噛み合う積み方であり，積み荷の安定性に優れる．具体的には，**図 6** に示すように，各段で x 軸方向に n_x 個，y 軸方向に n_y 個同じ向きで並べたものを 4 つ風車状に組み，かつ奇数段と偶数段で組み方を逆にして積み付ける．

5.　提案法

本章では，積み付けアルゴリズムが積み付けパターンを

決定する手順を最初に説明する．そして，この決定手順を構成するメカニズムを詳述し，最後に CGP による積み付けアルゴリズムの最適化方法を述べる．

5.1　積み付けアルゴリズムの概要

積み付けアルゴリズムは，現在の積み付け状態に積み付け可能なブロックをすべて列挙し，ブロックの種類と積み付け位置に関するルール（選択ルール）に基づいて好適なブロックを選択し積み付ける過程を繰り返す．列挙されるブロックは後述する積み付け可能位置ごとに複数生成される．そして，複数の選択ルールを順に適用し積み付けるブロックを選択する．なお，選択ルールの実行順は，CGP の遺伝子モデル（CGP モデル）によって決定される．また，選択されたブロックは（そのブロックの生成に用いられた）積み付け可能位置に積み付けられる．

最適化された CGP モデルを用いた積み付けアルゴリズムは，以下の手順で積み付けパターンを決定する．

入力　アイテムの種類ごとの個数と形状（幅・奥行・高さ），使用可能パレットの数，積み付け空間の規定値（幅・奥行・最大積み付け高さ）

出力　積み付けパターンもしくは計算失敗ステータス

Step 1（初期化）
未積み付けアイテムリストに全アイテムを登録する．

Step 2（終了判定）
未積み付けアイテムリストが空であれば，現時点での積み付けパターンを出力する．

Step 3（積み付け可能なブロック群の生成）
現在の積み付け状態において積み付け可能位置をすべて求め，それらの位置に積み付け可能なブロックを生成しブロック群に追加する．積み付け可能なブロックがない場合は計算失敗ステータスを出力する．

Step 4（積み付けブロックの選択）
CGP モデルが出力した実行順で選択ルールを適用し，積み付け可能なブロック群からブロックを 1 つ選択する．

Step 5（積み付け実行）
選択したブロックを（そのブロックの導出に用いられた）積み付け可能位置に積み付ける．選択したブロックを構成するアイテムを未積み付けアイテムリストから除外する．Step 2 に戻る．

以降では，Step 3 と Step 4 のメカニズムおよび CGP モデルの最適化方法について詳述する．

5.2　積み付け可能なブロック群の生成方法

まず，**図 7** に示すように，ブロックの積み付け可能位置をパレットの直上，あるいはすでに積み付けたブロックの直上・手前・右に設定する．なお，同図は 2 つのパレット（$pallet_1$，$pallet_2$）に積み付ける場合を図示している．また，積み付け空間における余剰な空隙部分を削減するために，ブロックは座標原点に寄せて積み付けられる．ここで，生成するブロックは以下の制約条件を満たす必要がある．

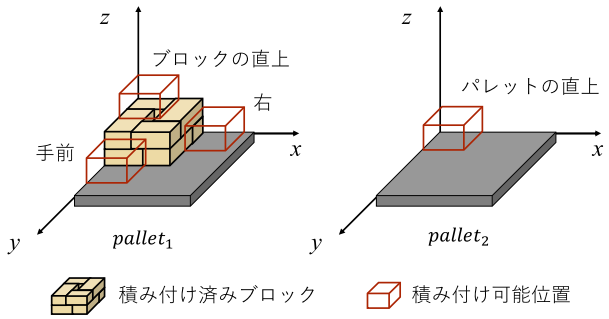


図 7　積み付け可能位置の概念図
Fig. 7　The conceptual diagram of loadable positions.

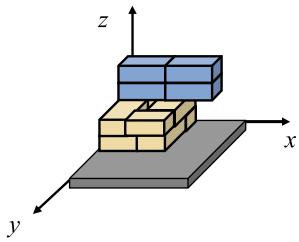


図 8　制約条件 (3) を満たさないブロック
Fig. 8　An example of a block violating the constraint (3).

(1) すべてのブロックはいずれかのパレットの積み付け空間に含まれること．
(2) 互いに異なる任意の 2 つのブロックが，積み付け空間上で正の体積の共通部分をもたないこと．
(3) 積み付けたブロックは，他のブロックあるいはパレットにより全底面を支持されていること．

上述の制約条件 (3) より，ブロックをはみ出して積み付けることは禁止される（**図 8**）．ただし，ピンホール積みされたブロックの空隙部分をまたぐブロックは許容される．

次に，積み付け可能なブロック群の生成手順を説明する．最初に，現在の積み付け状態における積み付け可能位置をすべて求める．次に，未積み付けアイテムリストに存在するアイテムの種類の数だけ，積み付け可能位置ごとに制約条件 (1)〜(3) を満足するブロックを求める．ここで，同じ積み付け可能位置に対して，同種類のアイテムからブロックを生成する場合でも，x, y, z 軸方向でそれぞれアイテム数が異なる複数のブロックが存在する．一方で，検品性の向上には，可能な限り多くのアイテムから構成されるブロックを用いる方が良い．このために，次に定義する内包関係において，内包関係にないブロックのみを積み付け可能なブロック群に加える．具体的には，ブロック A がブロック B を内包する条件を以下に定義する．

$$\text{A が B を内包する} := ((B_x \le A_x) \wedge (B_y \le A_y) \wedge (B_z \le A_z))$$

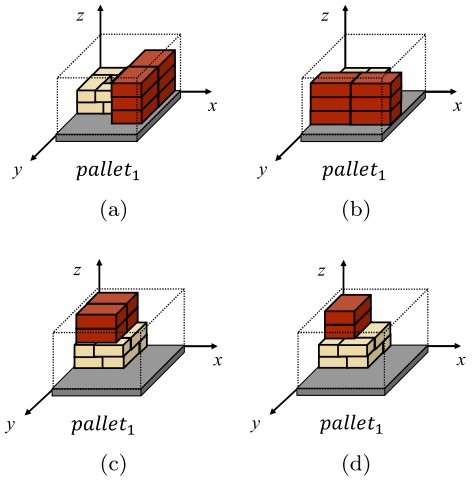


図 9　$pallet_1$ へ積み付け可能なブロックの例
Fig. 9　Examples of loadable blocks to $pallet_1$.

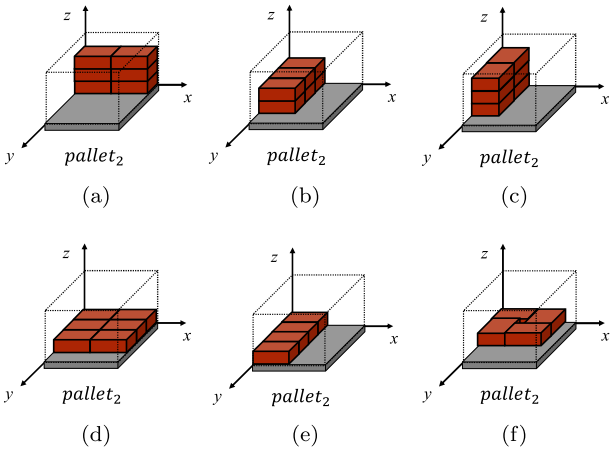


図 10　$pallet_2$ へ積み付け可能なブロックの例
Fig. 10　Examples of loadable blocks to $pallet_2$.

$$\wedge(\neg((B_x = A_x) \wedge (B_y = A_y) \wedge (B_z = A_z)))$$

ただし，A_x, A_y, A_z と B_x, B_y, B_z は，それぞれ A と B の x, y, z 軸方向のアイテム数を表す．内包関係にないブロックは，ある積み付け可能位置において x, y, z 軸方向のいずれかで最大アイテム数を持つ．たとえば，最大 6 個のアイテムを棒積みするブロックのうち，ブロック A ($A_x = 2$, $A_y = 1$, $A_z = 3$) は B ($B_x = 2$, $B_y = 1$, $B_z = 2$) を内包するが，A はブロック C ($C_x = 1$, $C_y = 3$, $C_z = 2$) を内包しない．また，図 7 に示す積み付け状況において（内包関係にない）積み付け可能なブロックの例を**図 9** と**図 10** に示す．同図では積み付け可能なブロックを赤色で示しており，図 9 と図 10 は，それぞれ図 7 の $pallet_1$ と $pallet_2$ にブロックを積み付けた場合に対応する．これらの図に示すとおり，積み付け可能位置ごとに複数種類のブロックが存在し，次節で述べる選択ルールによって積み付け位置を考慮しながら 1 つのブロックが選択される．

表 1　CGP に用いる入力の定義
Table 1　The definition of inputs used in the CGP models.

ID	定義
I_1	使用パレット数
I_2	パレットごとのブロック数の最小値
I_3	パレットごとのブロック数の最大値
I_4	パレットごとのブロック数の平均値
I_5	パレットごとのアイテム数の最小値
I_6	パレットごとのアイテム数の最大値
I_7	パレットごとのアイテム数の平均値
I_8	パレットごとの積載済みブロック総体積の最小値 [mm^3]
I_9	パレットごとの積載済みブロック総体積の最大値 [mm^3]
I_{10}	パレットごとの積載済みブロック総体積の平均値 [mm^3]
I_{11}	パレットごとの積載率の最小値
I_{12}	パレットごとの積載率の最大値
I_{13}	パレットごとの積載率の平均値
I_{14}	パレットごとの頭頂部 z 座標の最小値 [mm]
I_{15}	パレットごとの頭頂部 z 座標の最大値 [mm]
I_{16}	パレットごとの頭頂部 z 座標の平均値 [mm]
I_{17}	未積み付けアイテム数
I_{18}	積み付け済みアイテム数
I_{19}	積み付け済みブロック数
I_{20}	積み付け済み棒積みパターンブロック数
I_{21}	積み付け済みピンホールパターンブロック数
I_{22}	積み付け済みブロック総体積 [mm^3]
I_{23}	積み付け済み棒積みパターンブロック総体積 [mm^3]
I_{24}	積み付け済みピンホールパターンブロック総体積 [mm^3]
I_{25}	積み付け済みブロック総体積の全パレット容積に対する積載率
I_{26}	積み付け済み棒積みパターンブロック総体積の全パレット容積に対する積載率
I_{27}	積み付け済みピンホールパターンブロック総体積の全パレット容積に対する積載率
I_{28}	積み付け済みブロック幅の最小値 [mm]
I_{29}	積み付け済みブロック幅の最大値 [mm]
I_{30}	積み付け済みブロック幅の平均値 [mm]
I_{31}	積み付け済みブロック奥行の最小値 [mm]
I_{32}	積み付け済みブロック奥行の最大値 [mm]
I_{33}	積み付け済みブロック奥行の平均値 [mm]
I_{34}	積み付け済みブロック高さの最小値 [mm]
I_{35}	積み付け済みブロック高さの最大値 [mm]
I_{36}	積み付け済みブロック高さの平均値 [mm]
I_{37}	積み付け済みブロック体積の最小値 [mm^3]
I_{38}	積み付け済みブロック体積の最大値 [mm^3]
I_{39}	積み付け済みブロック体積の平均値 [mm^3]
I_{40}	積み付け済みブロック上面積の最小値 [mm^2]
I_{41}	積み付け済みブロック上面積の最大値 [mm^2]
I_{42}	積み付け済みブロック上面積の平均値 [mm^2]
I_{43}	積み付け済みブロック構成アイテム数の最小値
I_{44}	積み付け済みブロック構成アイテム数の最大値
I_{45}	積み付け済みブロック構成アイテム数の平均値

5.3　CGP モデルによる積み付けブロックの選択

5.3.1　CGP モデルの設定と選択ルール

CGP モデルの入力は，現在の積み付け状態から計算可

表 2　選択ルール
Table 2　Selection rules.

ID	定義
$rule_1$	パレットのインデックスが最小となるブロックを選択する
$rule_2$	構成するアイテム数が最大となるブロックを選択する
$rule_3$	積み付けた際に頭頂部 z 座標が最小となるブロックを選択する
$rule_4$	ブロック単体の高さが最小となるブロックを選択する
$rule_5$	ブロック単体の上面積が最大となるブロックを選択する
$rule_6$	ブロック単体の体積が最大となるブロックを選択する
$rule_7$	棒積みのブロックを選択する
$rule_8$	ピンホール積みのブロックを選択する

能な 45 個の特徴量に設定される（**表 1**）．作業者が着目する特徴量が解析困難であることから全特徴量を入力するが，CGP モデルの最適化を通して必要な特徴量が取捨選択されることになる．なお，表 1 において最小値，平均値および最大値を導出する際，ブロックが未積み付けであるパレットは計算対象から除外する．

提案法では**表 2** に示す 8 個の選択ルール（$rule_1$〜$rule_8$）を用い，選択するブロックの種類と積み付け可能位置に関する基準を設定する．同表に示すように，$rule_1$ と $rule_3$ は積み付け可能位置に関連する基準である．たとえば，$rule_3$ を適用すると，ブロックを（そのブロックの生成に用いた）積み付け可能位置に配置した場合に，全体の積み付け高さが最小となるブロックが選択される．したがって，パレット直上に配置されるブロックが選択される傾向がある．その他の選択ルールは，形状を含むブロックの種類のみを考慮して選択される．また，CGP モデルの出力ノード $O_i \in \mathbb{R}$ の値は，対応する i 番目の選択ルール（$rule_i$）の選択重要度に相当する．したがって，CGP モデルは 8 つの出力ノード O_1〜O_8 を持つ．

最後に，CGP モデルの演算ノードは**表 3** に示す数学記号を用いる．つまり，CGP モデルは，入力ノードを含む前段以前のノードから演算される出力値（選択重要度）を算出する関数を意味する．なお，定義した特徴量と選択ルールは積み付け状態やアイテムの種類・個数に依存せず使用できる．

5.3.2　積み付けブロックの選択

5.1 節の Step 4 の手順を述べる．まず，現在の積み付け状態から算出した特徴量を CGP モデルに入力し O_1〜O_8 の値を計算する．そして，選択重要度の降順に従って選択ルールの実行順を決定する．

次に，5.2 節の手順で生成した積み付け可能なブロック群から，実行順 1 位の選択ルールを満たすブロックを集約し部分群を抽出する．終了条件としてブロックが 1 つに定まれば，そのブロックを選択し 5.1 節の Step 5 を実行する．終了条件を充足しない場合は，次の実行順の選択ルールを満たすブロックを部分群から集約し部分群を更新する．この手順を実行順 8 位まで繰り返す．実行順 8 位の適用後に終了条件を充足しない場合は，部分群に格納された先頭のブロックを出力する例外処理を行う．この例外処理は，水平方向における回転体など体積や高さが同じブロックが生成された場合にのみ適用され，適合度の低下に影響しないことを確認している．

表 3　CGP に用いる演算子
Table 3　Operators used in the CGP models.

記号	ノード入力数	定義
$+$	2	$x_1 + x_2$
$-$	2	$x_1 - x_2$
$\times$	2	$x_1 \times x_2$
$\div$	2	$x_1 \div x_2$
mod	2	$x_1 \bmod x_2$
$\lvert\,\rvert$	1	$\lvert x_1 \rvert$
sum	$1 \le n \le 45$	$\sum_{i=1}^{n} x_i$
0.0	0	Const. 0.0
0.1	0	Const. 0.1
0.2	0	Const. 0.2
0.3	0	Const. 0.3
0.4	0	Const. 0.4
0.5	0	Const. 0.5
0.6	0	Const. 0.6
0.7	0	Const. 0.7
0.8	0	Const. 0.8
0.9	0	Const. 0.9
1.0	0	Const. 1.0
10.0	0	Const. 10.0
-1.0	0	Const. -1.0

5.4　CGP モデルの最適化

提案法では，CGP モデルが出力する選択ルールの実行順を最適化することで，目的関数で指定される所望の積み付けパターンを出力するように積み付けアルゴリズムを調整する．**図 11** に図示するように，提案法は 3 章で述べた CGP の最適化手順に従って CGP モデルの最適化を行う．具体的には，1 つの親 CGP モデルを選択し 4 つの子 CGP モデルを生成する (1 + 4)-ES の最適化手順に従う．ここで，CGP モデルの適合度は次の手順で計算される．まず，CGP モデルを 5.1 節で述べた積み付けアルゴリズムに組み込み，最終的な積み付けパターンを出力する．次に，出力した積み付けパターンを目的関数で評価し，得られた評価値を CGP モデルの適合度に設定する．そして，適合度が最良値を持つ CGP モデルを次世代の親 CGP モデルに選択し，次世代の子 CGP モデルを生成する．この操作を最大世代数 G まで繰り返す．

6.　数値実験

本章では，提案法の性能を評価するために，訓練用問題と検証用問題を設定し数値実験を行う．以降では，実験方

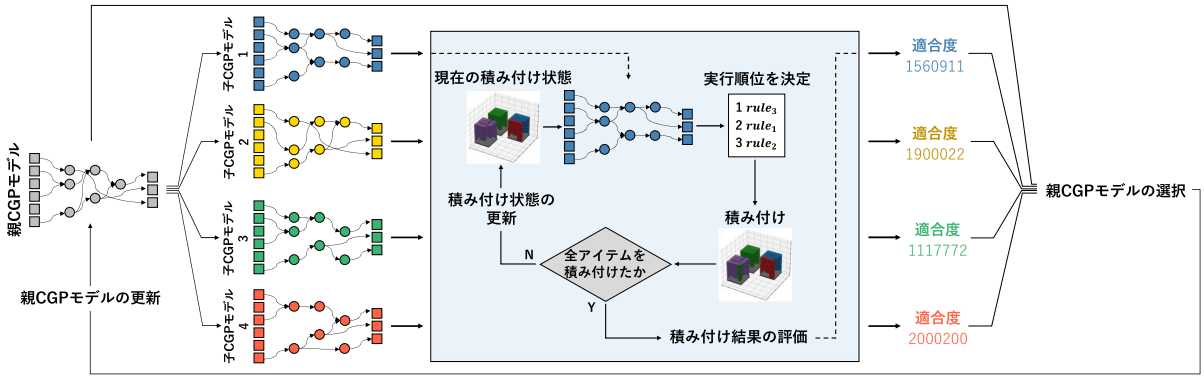


図 11　提案法における CGP モデルの最適化

Fig. 11　The CGP framework used in the proposed method.

表 4　アイテムのサイズ [mm]

Table 4　Size of load items [mm].

ID	幅	奥行	高さ
$item_1$	300	200	300
$item_2$	300	400	200
$item_3$	250	200	200
$item_4$	200	300	200
$item_5$	400	250	200
$item_6$	400	400	400
$item_7$	400	300	250
$item_8$	200	250	350
$item_9$	300	350	250
$item_{10}$	250	200	300
$item_{11}$	300	400	250
$item_{12}$	250	250	200
$item_{13}$	200	350	200
$item_{14}$	400	300	200
$item_{15}$	400	400	350
$item_{16}$	300	300	250
$item_{17}$	400	250	350
$item_{18}$	300	350	200

表 5　訓練用問題

Table 5　Training problems.

$train_1$	$train_2$	$train_3$
90 個	100 個	100 個
130 個	120 個	120 個
150 個	140 個	140 個
-	110 個	110 個
-	100 個	100 個
-	60 個	60 個
-	-	70 個
-	-	100 個
-	-	90 個
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-
-	-	-
計 370 個	計 630 個	計 890 個

表 6　検証用問題

Table 6　Test problems.

$test_1$	$test_2$	$test_3$	$test_4$
110 個	130 個	-	-
80 個	100 個	-	-
160 個	180 個	-	-
-	70 個	-	-
-	150 個	-	-
-	80 個	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	140 個	120 個
-	-	90 個	110 個
-	-	170 個	130 個
-	-	90 個	100 個
-	-	160 個	120 個
-	-	90 個	70 個
-	-	-	90 個
-	-	-	130 個
-	-	-	80 個
計 350 個	計 710 個	計 740 個	計 950 個

法と実験設定を詳述した後，実験結果を示す．なお，評価回数 200 回を規定値とするが，本章の最後に 800 回に増加させた場合の追加分析も行う．

6.1　実験方法

定義した目的関数 $f(\boldsymbol{x})$ の最小化を目的とし，CGP モデルを用いた積み付けアルゴリズムを単一の訓練用問題で最適化する．その後，最大世代数 G の終了時点で得られた最良の評価値を持つ積み付けアルゴリズムを複数の検証用問題に転用し，同一の目的関数から算出した評価値を評価する．これを 1 つの実験ケースとする．

具体的には，**表 4** に示す 18 種類のアイテム（$item_1$〜$item_{18}$）を用いて，3 つの訓練用問題（$train_1$〜$train_3$）と 4 つの検証用問題（$test_1$〜$test_4$）を設定する．**表 5** と**表 6** に訓練用問題と検証用問題の設定を示す．また，使用可能なパレットの数を $\{train_1, train_2, train_3\}$ について $\{12, 16, 20\}$，$\{test_1, test_2, test_3, test_4\}$ について $\{12, 20, 20, 25\}$ にそれぞれ設定する．アイテムの種類数，個数および使用可能なパレット数の増加にともない考えうる積み付けパターン数が増加することから，$train_1, train_2, train_3$ および $test_1, test_2, test_3, test_4$ の順でそれぞれ問題の難易度が高くなる．また，以下に定義する 2 つの目的関数 $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$ を用いる．

$$f_1(\boldsymbol{x}) = 10^5 g_1(\boldsymbol{x}) + g_2(\boldsymbol{x}) - g_3(\boldsymbol{x}) \quad (1)$$

$$f_2(\boldsymbol{x}) = -10^5 g_1(\boldsymbol{x}) + g_2(\boldsymbol{x}) + 100 g_4(\boldsymbol{x}) \quad (2)$$

ここで，$f_1(\boldsymbol{x}), f_2(\boldsymbol{x}) \in \mathbb{R}$ であり，$\boldsymbol{x}$ は積み付けアルゴリズムが出力した積み付けパターンを示すベクトルである．また，$g_1(\boldsymbol{x})$〜$g_4(\boldsymbol{x})$ は，それぞれ $\boldsymbol{x}$ から計算可能な以下の特徴量である．

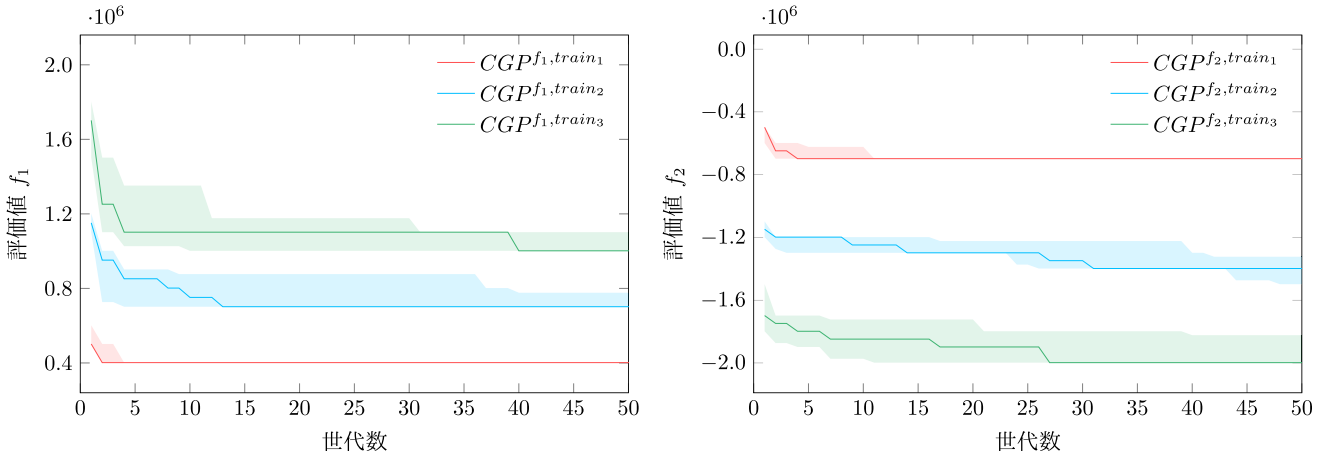


図 12　f_1 と f_2 に設定した訓練用問題における各評価値の遷移

Fig. 12　The best objective values of f_1 and f_2 over generations on the training problems.

- $g_1(\boldsymbol{x})$：使用パレット数
- $g_2(\boldsymbol{x})$：パレットごとの頭頂部 z 座標の最大値
- $g_3(\boldsymbol{x})$：パレットごとの積載率の平均値
- $g_4(\boldsymbol{x})$：パレットごとの積載率の最大値

$f_1(\boldsymbol{x})$ は，可能な限り少ないパレット数で各パレットへの積載率を高くしつつ，積み付けパターン全体の高さを抑制した積み付けパターンが好適となる．一方で $f_2(\boldsymbol{x})$ は，使用パレット数の増加および各パレットの積載率の低下を目的とし，アイテムを分散して積み付ける積み付けパターンが好適となる．

6.2　比較基準とパラメータ設定

実験ケースごとの表記を簡略化するため，f_i に設定した $train_j$ で最適化するプロセスを $CGP^{f_i,train_j}$ と表記する．また，$CGP^{f_i,train_j}$ で獲得した積み付けアルゴリズムを f_k に設定した $test_l$ に転用するプロセスを $CGP^{f_i,train_j}_{f_k,test_l}$ と表記する．ただし，$i=\{1,2\}$，$j=\{1,2,3\}$，$k=\{1,2\}$，$l=\{1,2,3,4\}$ である．なお，提案法では，同一の目的関数 f_i のもとで転用するため $CGP^{f_i,train_j}_{f_i,test_l}$ と表記される．

提案法が導出した積み付けアルゴリズムの転用時（$CGP^{f_i,train_j}_{f_i,test_l}$）における性能は，$test_l$ において積み付けアルゴリズムが導出した積み付けパターンに対する f_i の評価値として評価される．以降では単に提案法の転用性能と呼び，この転用性能と次の 2 つのベースラインを比較する．

- **ベースライン 1**：$CGP^{f_i,test_l}$ での f_i の最良値
  すなわち，提案法の転用時に用いた検証用問題で積み付けアルゴリズムを最適化（$CGP^{f_i,test_l}$）し，最良個体となる積み付けアルゴリズムが導出した積み付けパターンに対する f_i の評価値をベースラインとする．
- **ベースライン 2**：$CGP^{f_k,train_j}_{f_i,test_l}$ での評価値（$i \neq k$）
  すなわち，提案法の最適化プロセスとは異なる目的関数 f_k を用いて最適化した積み付けアルゴリズムを構築し，それを転用したとき（$CGP^{f_k,train_j}_{f_i,test_l}$）の性能（$test_l$ における f_i の評価値）をベースラインとする．たとえば，$CGP^{f_1,train_1}_{f_1,test_1}$ に対して $CGP^{f_2,train_1}_{f_1,test_1}$ からこのベースラインが算出される．

提案法の転用性能とベースライン 1 の差が小さく，かつ，ベースライン 2 との差が大きいほど，提案法が導出した積み付けアルゴリズムは，目的関数で指定される所望の積み付けパターンを生成できることが評価できる．各目的関数の最適解を用いた絶対評価や既存 GPHH と公平な比較が困難であることから上述のベースラインを設定する[*1]．

本論文で用いるパラメータ設定として，最大世代数 $G=50$，CGP で用いるノードの行数 $n_r=1$，ノードの列数 $n_c=800$，(1+4)-ES で用いる突然変異確率を $\mu=0.006$ とする．(1 + 4)-ES は 1 世代につき 4 個体を評価するため，$G=50$ によって評価回数は 200 回となる．実験ケースごとに乱数シードが異なる独立した 10 試行の実験を行う．提案法の転用性能およびベースライン 1，2 は 10 試行の中央値[*2]として報告する．

6.3　実験結果

まず，訓練用問題における評価値の推移を**図 12** に示す．各グラフの実線と着色部の領域は，それぞれ評価値の中央値と四分位範囲である．同図より，全実験ケースに対して評価値が改善しており，提案法が導出した積み付けアルゴリズムが目的関数で指定される所望の積み付けパターンを出力するように調整されている．付録内の A.1 には提案法が導出した積み付けパターンを例示している．

次に，提案法の転用性能を**図 13** と**図 14** に示す．図中

[*1] 各目的関数の最適解は，とりうる積み付けパターンの数が膨大になり特定できていない．また，本論文は一般的なビンパッキング問題でないことに加え，選択ルールを求める独自点から既存 GPHH と解法も異なることが理由である．

[*2] すべての実験ケースにおいて観測した評価値が正規分布と乖離することから中央値を用いている．

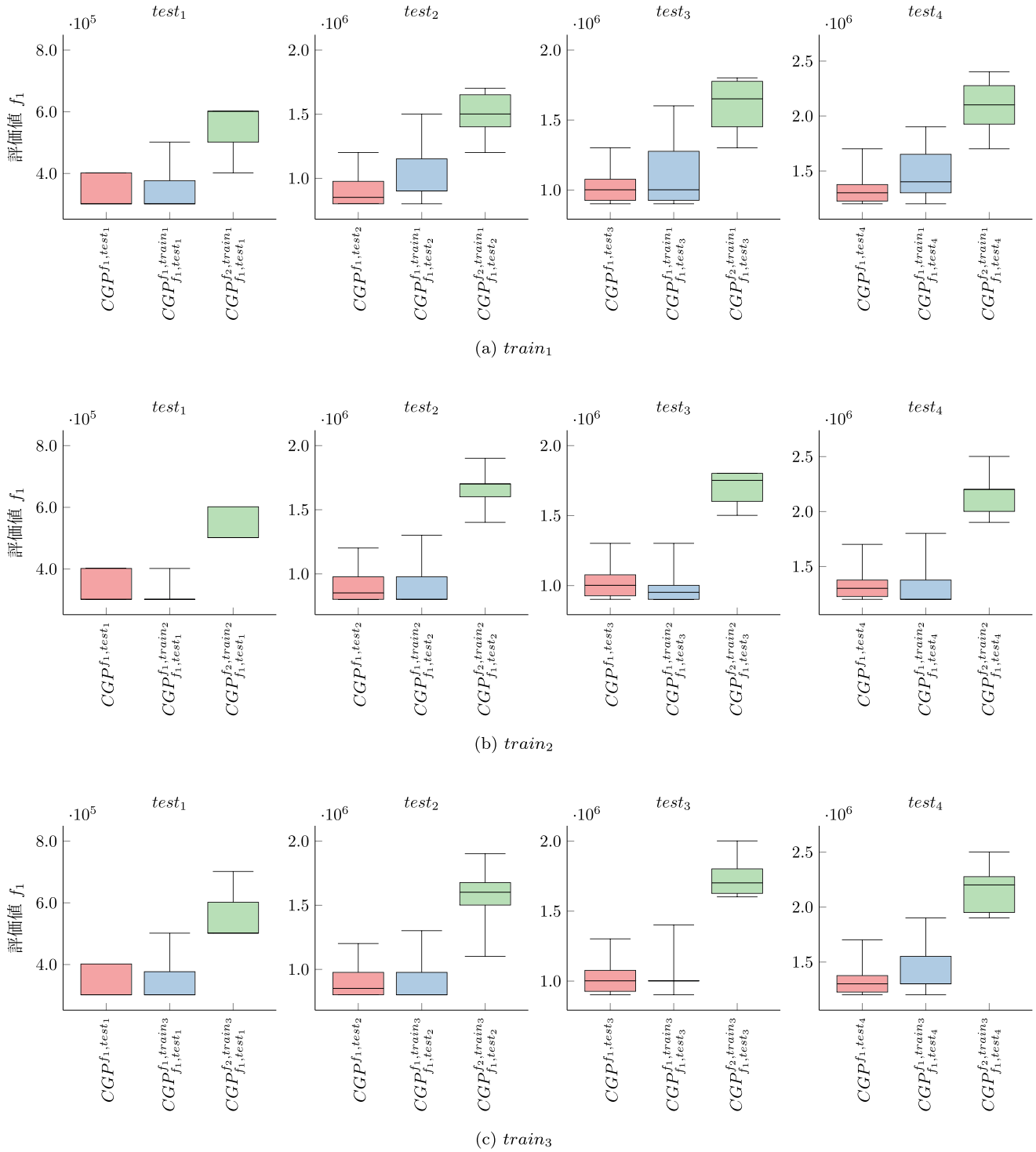


図 13　f_1 に設定した各訓練用問題から生成した積み付けアルゴリズムの転用性能の五数要約 ($G = 50$)

Fig. 13　The five number summary of the performance of the constructed loading algorithms on the trainig problems with f_1 ($G = 50$).

の箱ひげ図は 10 試行で得られた性能の五数要約であり，提案法の転用性能を青色，ベースライン 1 と 2 をそれぞれ赤色と緑色で示す．全実験ケースにおいて全アイテムの積み付けに成功している．図 13 と図 14 より，全実験ケースにおいて中央値に着目すると，提案法の転用性能はベースライン 2 よりもベースライン 1 との差が小さいことから，提案法は各目的関数に調整された積み付けアルゴリズムを生成できている．また，難易度が最も低い $train_1$ では，検証用問題の難易度が増加するにつれて，提案法の転用性能の四分位範囲が大きくなる．一方で，訓練用問題の難易度が増加するにつれて，提案法の転用性能の四分位範囲が小さくなる．これは，アイテムの種類数や個数が多い問題で

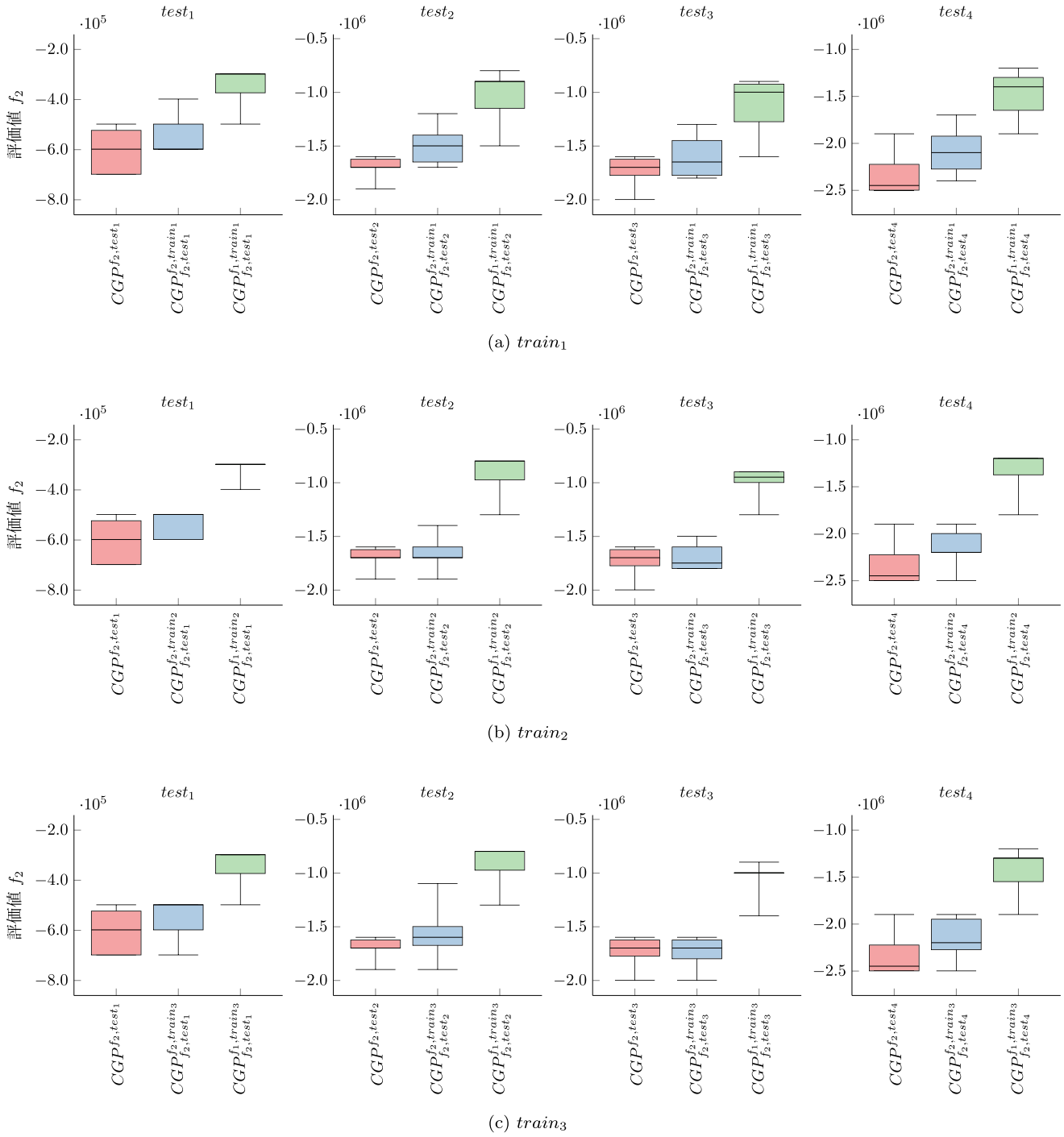


図 14　f_2 に設定した各訓練用問題から生成した積み付けアルゴリズムの転用性能の五数要約 ($G = 50$)

Fig. 14　The five number summary of the performance of the constructed loading algorithms on the trainig problems with f_2 ($G = 50$).

積み付けアルゴリズムが過剰に調整されていないことを意味する．したがって，難易度が高い訓練用問題を用いることが，難易度の低い問題においても転用可能な積み付けアルゴリズムの生成に寄与する．ベースライン 1，2 および提案法の平均順位は f_1 において 1.33，3.00，1.67，f_2 において 1.17，3.00，1.83 である[*3]．また，フリードマン検定および Holm 法を適用したウィルコクソンの符号順位検定を適用した結果，すべての手法間に有意水準 5% 未満の有意差が存在することを確認している．

以上より，提案法は類似問題で転用可能な積み付けアルゴリズムを生成可能であり，検証用問題で改めて積み付けアルゴリズムを最適化せずとも，所望の積み付けパターンを導出できることが分かる．上述した実験結果に限定すれば，積み付けアルゴリズムを検証用問題に転用することで，

[*3] 平均順位の導出に用いた図 13 と図 14 の詳細な値を付録 A.2 に示す．

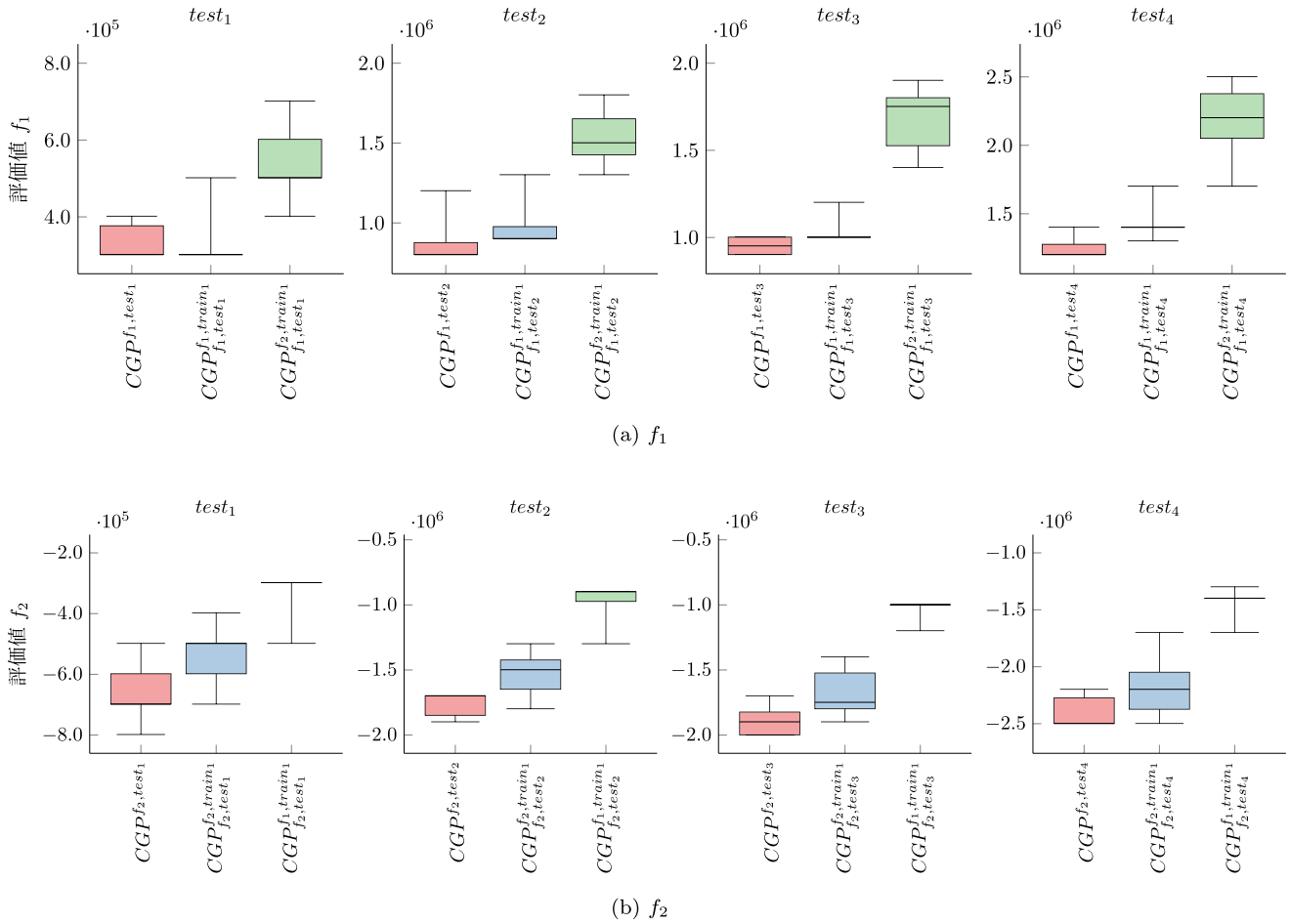


図 15　f_1，f_2 に設定した $train_1$ から生成した積み付けアルゴリズムの転用性能の五数要約（$G = 200$）

Fig. 15　The five number summary of the performance of the constructed loading algorithms on $train_1$ with f_1, f_2 ($G = 200$).

200 回の評価回数を削減して目的関数を満たす積み付けパターンが生成できることになる．

6.4　追加分析

実験結果で示したとおり，$train_1$ における提案法の転用性能の四分位範囲は，検証用問題の難易度が増加するにつれて大きくなる．この結果は，訓練用問題よりも検証用問題の難易度が高い場合，試行によっては転用性能が劣る積み付けアルゴリズムが生成されることを意味する．したがって，訓練用問題の設定を上回るアイテムの種類数や個数に対応可能な積み付けアルゴリズムを導出することが困難であるという提案法の問題が考えられる．一方で，この問題は評価回数を増やすことで解消できる可能性がある．この可能性を明らかにするために，本節では最大世代数を $G = 200$（評価回数 800 回）に増加させ提案法の転用性能を評価する．最大世代数を除いて前節と同一の実験設定を用いる．

図 15 に $train_1$ で訓練した提案法の転用性能を示す．また，同図では，200 世代時点でのベースライン 1 と 2 も示している．まず f_1 について，50 世代での転用性能（図 13 (a)）と比較すると，200 世代まで増加させることで提案法の転用性能の四分位範囲が小さくなる．しかし，f_2 では，50 世代での転用性能（図 14 (a)）と比べて，200 世代まで増加させても転用性能の四分位範囲は改善していない．この理由として，50 世代の時点で局所解に陥った可能性が考えられる．

以上より，評価回数を増やすことで四分位範囲が小さくなり，試行ごとに積み付けアルゴリズムの再現性が改善することから，積み付けアルゴリズムの転用性能が改善する可能性がある．一方で，評価回数を削減しながら局所解への早期収束を抑制するためには，CGP の最適化方法を改良する必要がある．たとえば，目的関数を近似する機械学習を用いたサロゲート進化計算 [46] を CGP に用いる方法が考えられる．

7.　結論

本論文では，転用可能な積み付けアルゴリズムを少ない評価回数で自動生成可能な技術を提案した．提案法の特徴は，ブロック（アイテムのまとまり）の種類および積み付

け位置を同時に決定する選択ルールを導入することで，積み付けアルゴリズムの探索空間の膨大化を抑制する点にある．そして，選択ルールの実行順を出力する CGP モデルを最適化することで，所与の目的関数で指定される好適な積み付けパターンを出力可能な積み付けアルゴリズムが生成できる．実験結果では，アイテムの種類数と個数が異なる類似問題において，提案法が導出した積み付けアルゴリズムがベースラインと競合する性能で転用可能であることを示した．

今後の予定として，サロゲート進化計算を用いた CGP を構築し提案法に組み込むことで転用性能を改善する．また，提案法と IEC を融合し，作業者の経験則を反映した積み付けアルゴリズムの自動生成技術に取り組む．提案法と IEC を融合する際，積み付け状況ごとの作業者の経験則を詳細に反映するために，複数の訓練用問題を用いて積み付けアルゴリズムを生成する方法が考えられる．このために，進化的ルール学習を提案法に組み込むことで，状況ごとに適用する積み付けアルゴリズムを細分化して生成可能な方法を検討する．この拡張手法を用いて，想定する状況や所望の積み付けパターンが異なる複数の作業者が評価を行えば，様々な状況に対して転用可能な積み付けアルゴリズムの構築につながる．また，本論文が取り組んだように，問題特徴を活用したヒューリスティクスが設計できれば，積み付け問題において CGP を用いた GPHH が評価回数の削減に有効な方法となる可能性がある．この点も調査を進める予定である．

参考文献

[1] 厚生労働省：令和元年版労働経済の分析—人手不足の下での「働き方」をめぐる課題について—(2019).

[2] Takagi, H.: Interactive evolutionary computation, *Proc. Int. Conf. Soft Comput. Inf./Intell. Syst*, pp.41–50 (1998).

[3] Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation, *Proc. IEEE*, Vol.89, No.9, pp.1275–1296 (2001).

[4] Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E. and Woodward, J.R.: A classification of hyper-heuristic approaches: revisited, *Handbook of Metaheuristics*, pp.453–477, Springer (2019).

[5] Miller, J.F. and Harding, S.L.: Cartesian genetic programming, *Proc. 10th Annual Conference Companion on Genetic and Evolutionary Computation*, pp.2701–2726 (2008).

[6] Miller, J. and Turner, A.: Cartesian Genetic Programming, *Proc. Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO Companion '15*, pp.179–198, Association for Computing Machinery (online), DOI: 10.1145/2739482.2756571 (2015).

[7] Martello, S., Pisinger, D. and Vigo, D.: The three-dimensional bin packing problem, *Operations Research*, Vol.48, No.2, pp.256–267 (2000).

[8] Lodi, A., Martello, S. and Vigo, D.: Recent advances on two-dimensional bin packing problems, *Discrete Applied Mathematics*, Vol.123, No.1-3, pp.379–396 (2002).

[9] Lee, C.C. and Lee, D.-T.: A simple on-line bin-packing algorithm, *Journal of the ACM* (*JACM*), Vol.32, No.3, pp.562–572 (1985).

[10] Coffman, E.G.Jr., Garey, M. and Johnson, D.: Approximation algorithms for bin packing: A survey, *Approximation Algorithms for NP-hard Problems*, pp.46–93 (1996).

[11] Silva, E., Oliveira, J.F. and Waescher, G.: The pallet loading problem: A review of solution methods and computational experiments, *International Trans. Operational Research*, Vol.23, No.1-2, pp.147–172 (2016).

[12] Levine, J. and Ducatelle, F.: Ant colony optimization and local search for bin packing and cutting stock problems, *Journal of the Operational Research Society*, Vol.55, No.7, pp.705–716 (2004).

[13] Liu, D., Tan, K.C., Huang, S., Goh, C.K. and Ho, W.K.: On solving multiobjective bin packing problems using evolutionary particle swarm optimization, *European Journal of Operational Research*, Vol.190, No.2, pp.357–382 (2008).

[14] Reeves, C.: Hybrid genetic algorithms for bin-packing and related problems, *Annals of Operations Research*, Vol.63, No.3, pp.371–396 (1996).

[15] Stawowy, A.: Evolutionary based heuristic for bin packing problem, *Computers & Industrial Engineering*, Vol.55, No.2, pp.465–474 (2008).

[16] Zendaoui, Z. and Layeb, A.: Adaptive cuckoo search algorithm for the bin packing problem, *Modelling and Implementation of Complex Systems*, pp.107–120, Springer (2016).

[17] De Carvalho, J.V.: LP models for bin packing and cutting stock problems, *European Journal of Operational Research*, Vol.141, No.2, pp.253–273 (2002).

[18] Haouari, M. and Serairi, M.: Heuristics for the variable sized bin-packing problem, *Computers & Operations Research*, Vol.36, No.10, pp.2877–2884 (2009).

[19] 小畑尚輝，太田秀典，中森眞理雄ほか：積み込み，積み下ろし順序を考慮した 3 次元パッキング問題に関する研究，研究報告数理モデル化と問題解決（MPS），Vol.2011, No.17, pp.1–7 (2011).

[20] 宮崎晃年，鮭川矩義，高野祐一，水野眞治ほか：キークレーン間干渉を考慮したコンテナ事前配列問題，情報科学研究，Vol.37, pp.1–11 (2017).

[21] Júnior, R.R., Yanasse, H.H., Morabito, R. and Junqueira, L.: A hybrid approach for a multi-compartment container loading problem, *Expert Systems with Applications*, Vol.137, pp.471–492 (2019).

[22] Xiang, X., Yu, C., Xu, H. and Zhu, S.X.: Optimization of heterogeneous container loading problem with adaptive genetic algorithm, *Complexity*, Vol.2018 (2018).

[23] Nguyen, S., Zhang, M. and Johnston, M.: A genetic programming based hyper-heuristic approach for combinatorial optimisation, *Proc. 13th Annual Conference on Genetic and Evolutionary Computation*, pp.1299–1306 (2011).

[24] Park, J., Mei, Y., Nguyen, S., Chen, G., Johnston, M. and Zhang, M.: Genetic programming based hyper-heuristics for dynamic job shop scheduling: Cooperative coevolutionary approaches, *European Conference on Genetic Programming*, pp.115–132, Springer (2016).

[25] Glanville, R., Griffiths, D., Baron, P., Drake, J.H., Hyde, M., Ibrahim, K. and Ozcan, E.: A genetic programming hyper-heuristic for the multidimensional knapsack prob-

lem, *Kybernetes* (2014).
[26] Tan, B., Ma, H., Mei, Y. and Zhang, M.: A Cooperative Coevolution Genetic Programming Hyper-Heuristic Approach for On-line Resource Allocation in Container-based Clouds, *IEEE Trans. Cloud Computing* (2020).
[27] Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. and Qu, R.: Hyper-heuristics: A survey of the state of the art, *Journal of the Operational Research Society*, Vol.64, No.12, pp.1695–1724 (2013).
[28] Gomez, J.C. and Terashima-Marín, H.: Evolutionary hyper-heuristics for tackling bi-objective 2D bin packing problems, *Genetic Programming and Evolvable Machines*, Vol.19, No.1-2, pp.151–181 (2018).
[29] Ross, P., Marín-Blázquez, J.G., Schulenburg, S. and Hart, E.: Learning a procedure that can solve hard bin-packing problems: A new GA-based approach to hyper-heuristics, *Genetic and Evolutionary Computation Conference*, pp.1295–1306, Springer (2003).
[30] Terashima-Marín, H., Farias Zarate, C., Ross, P. and Valenzuela-Rendón, M.: Comparing two models to generate hyper-heuristics for the 2D-regular bin-packing problem, *Proc. 9th Annual Conference on Genetic and Evolutionary Computation*, pp.2182–2189 (2007).
[31] Allen, S., Burke, E.K., Hyde, M. and Kendall, G.: Evolving reusable 3d packing heuristics with genetic programming, *Proc. 11th Annual Conference on Genetic and Evolutionary Computation*, pp.931–938 (2009).
[32] Burke, E.K., Hyde, M.R. and Kendall, G.: Evolving bin packing heuristics with genetic programming, *Parallel Problem Solving from Nature-PPSN IX*, pp.860–869, Springer (2006).
[33] Burke, E.K., Hyde, M.R., Kendall, G. and Woodward, J.: Automatic heuristic generation with genetic programming: Evolving a jack-of-all-trades or a master of one, *Proc. 9th Annual Conference on Genetic and Evolutionary Computation*, pp.1559–1565 (2007).
[34] Burke, E.K., Hyde, M., Kendall, G. and Woodward, J.: A genetic programming hyper-heuristic approach for evolving 2-D strip packing heuristics, *IEEE Trans. Evolutionary Computation*, Vol.14, No.6, pp.942–958 (2010).
[35] Burke, E.K., Hyde, M.R., Kendall, G. and Woodward, J.: Automating the packing heuristic design process with genetic programming, *Evolutionary Computation*, Vol.20, No.1, pp.63–89 (2012).
[36] Sim, K. and Hart, E.: Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model, *Proc. 15th Annual Conference on Genetic and Evolutionary Computation*, pp.1549–1556 (2013).
[37] Sosa-Ascencio, A., Terashima-Marín, H., Ortiz-Bayliss, J.C. and Conant-Pablos, S.E.: Grammar-based selection hyper-heuristics for solving irregular bin packing problems, *Proc. 2016 on Genetic and Evolutionary Computation Conference Companion*, pp.111–112 (2016).
[38] Pillay, N.: A study of evolutionary algorithm selection hyper-heuristics for the one-dimensional bin-packing problem, *South African Computer Journal*, Vol.48, No.1, pp.31–40 (2012).
[39] Silva-Gálvez, A., Lara-Cárdenas, E., Amaya, I., Cruz-Duarte, J. and Ortiz-Bayliss, J.: A Preliminary Study on Score-Based Hyper-heuristics for Solving the Bin Packing Problem, *Mexican Conference on Pattern Recognition*, pp.318–327, Springer (2020).
[40] Ross, N., Keedwell, E. and Savic, D.: Human-Derived Heuristic Enhancement of an Evolutionary Algorithm for the 2D Bin-Packing Problem, *International Conference on Parallel Problem Solving from Nature*, pp.413–427, Springer (2020).
[41] Ryser-Welch, P., Miller, J.F., Swan, J. and Trefzer, M.A.: Iterative Cartesian genetic programming: Creating general algorithms for solving travelling salesman problems, *European Conference on Genetic Programming*, pp.294–310, Springer (2016).
[42] Hu, H., Zhang, X., Yan, X., Wang, L. and Xu, Y.: Solving a new 3D bin packing problem with deep reinforcement learning method, arXiv preprint arXiv:1708.05930 (2017).
[43] Duan, L., Hu, H., Qian, Y., Gong, Y., Zhang, X., Wei, J. and Xu, Y.: A Multi-Task Selected Learning Approach for Solving 3D Flexible Bin Packing Problem, *Proc. 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19*, pp.1386–1394, International Foundation for Autonomous Agents and Multiagent Systems (2019).
[44] Miller, J.F.: *Cartesian Genetic Programming*, pp.17–34, Springer Berlin Heidelberg (2011).
[45] 日本工業規格：JIS Z 0105：2015 包装貨物–包装モジュール寸法 (2015).
[46] Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges, *Swarm and Evolutionary Computation*, Vol.1, No.2, pp.61–70 (2011).

付　　録

A.1　最適化された積み付けパターンの例

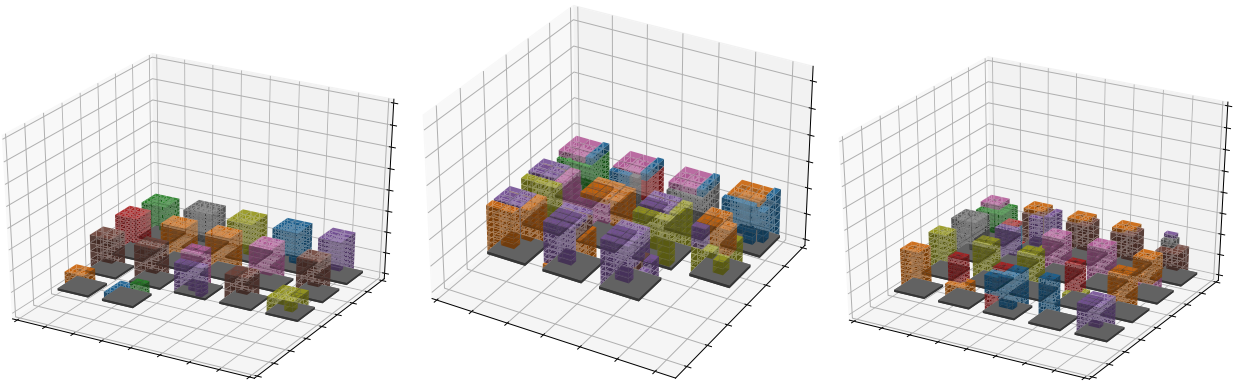
図 **A·1** (a)，(b)，(c) に，初期の積み付けアルゴリズム（1 世代目の初期個体）と f_1 に設定した $train_3$ における最良の同アルゴリズム（50 世代目の最良個体），および，f_2 に設定した $train_3$ における最良の同アルゴリズムがそれぞれ出力した積み付けパターンを図示する[*4]．図 A·1 (a)

表 A·1　f_1 における転用性能とベースライン 1，2 の中央値および平均順位

Table A·1　The median values and the average ranks of the baseline 1, the baseline 2, and the proposed method on f_1.

問題設定 (j,l)	$CGP^{f_1,test_j}$	$CGP^{f_1,train_j}_{f_1,test_l}$	$CGP^{f_2,train_j}_{f_1,test_l}$
$train_1-test_1$ $(1,1)$	301799.15	301799.15	601799.57
$train_1-test_2$ $(1,2)$	851799.16	901799.21	1501799.52
$train_1-test_3$ $(1,3)$	1001799.22	1001799.21	1651799.52
$train_1-test_4$ $(1,4)$	1301799.18	1401799.24	2101799.49
$train_2-test_1$ $(2,1)$	301799.15	301799.15	501799.49
$train_2-test_2$ $(2,2)$	801799.11	801799.11	1401799.49
$train_2-test_3$ $(2,3)$	901799.13	901799.13	1501799.48
$train_2-test_4$ $(2,4)$	1201799.11	1201799.11	1901799.44
$train_3-test_1$ $(3,1)$	301799.15	301799.15	501799.49
$train_3-test_2$ $(3,2)$	851799.16	801799.11	1601799.55
$train_3-test_3$ $(3,3)$	1001799.21	1001799.22	1701799.54
$train_3-test_4$ $(3,4)$	1301799.18	1301799.18	2201799.52
平均順位	1.33	1.67	3.00

[*4] パレット（灰色）がアイテムに対し上書きで表示されているが，実際はアイテムがパレット上に配置されている．



(a) 初期（f_1, f_2，1 世代）　(b) 最良（f_1，50 世代）　(c) 最良（f_2，50 世代）

図 A·1　f_1 および f_2 に設定した $train_3$ で生成した初期・最良積み付けアルゴリズムの積み付けパターン

Fig. A·1　Loading patterns obtained by the initial and best loading algorithms on $train_3$ with f_1, f_2, respectively.

表 A·2　f_2 における転用性能とベースライン 1，2 の中央値および平均順位

Table A·2　The median values and the average ranks of the baseline 1, the baseline 2, and the proposed method on f_2.

問題設定 (j,l)	$CGP^{f_2,test_j}$	$CGP^{f_2,train_j}_{f_2,test_l}$	$CGP^{f_1,train_j}_{f_2,test_l}$
$train_1-test_1$ $(1,1)$	−598128.51	−598121.49	−298104.18
$train_1-test_2$ $(1,2)$	−1698117.36	−1498119.19	−898106.34
$train_1-test_3$ $(1,3)$	−1698117.36	−1648117.36	−998105.45
$train_1-test_4$ $(1,4)$	−2448119.47	−2098120.46	−1398109.45
$train_2-test_1$ $(2,1)$	−598128.51	−498129.8	−298102.94
$train_2-test_2$ $(2,2)$	−1698117.36	−1698117.36	−798101.65
$train_2-test_3$ $(2,3)$	−1698117.36	−1748117.36	−948102.89
$train_2-test_4$ $(2,4)$	−2448119.47	−2198117.36	−1198102.71
$train_3-test_1$ $(3,1)$	−598128.51	−498126.08	−298103.4
$train_3-test_2$ $(3,2)$	−1698117.36	−1598117.36	−798102.75
$train_3-test_3$ $(3,3)$	−1698117.36	−1698117.36	−998107.19
$train_3-test_4$ $(3,4)$	−2448119.47	−2198117.36	−1298102.71
平均順位	1.17	1.83	3.00

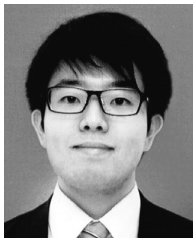
と図 A·1 (b) より，50 世代目の積み付けパターンは，使用パレット数が削減され積載率も向上している（f_1 の目的）．図 A·1 (a) と図 A·1 (c) より，50 世代目の積み付けパターンは，パレットの使用数が 20 個に増え各パレットの積載率も低下している（f_2 の目的）．以上より，提案法は各目的関数に調整された積み付けアルゴリズムを生成できている．

A.2　実験結果の中央値と平均順位

図 13 と図 14 に示した中央値と平均順位をそれぞれ**表 A·1** と**表 A·2** にまとめる．

蛭田 悠介

1998 年生．2021 年横浜国立大学理工学部数物・電子情報系学科卒業．現在，同大学大学院修士課程．進化計算の研究に従事．学士（工学）．



西原 慧　（学生会員）

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

小熊 祐司

1985 年生．2009 年慶應義塾大学大学院理工学研究科基礎理工学専攻修士課程修了．同年株式会社テプコシステムズ入社，2013 年株式会社 IHI 入社し現在に至る．2015 年慶應義塾大学大学院理工学研究科基礎理工学専攻博士課程修了．博士（工学）．数理最適化や進化計算に関する研究に従事．電気学会優秀論文発表賞等を受賞．電気学会，日本 OR 学会各会員．

藤井 正和

1978 年生．2002 年大阪大学大学院基礎工学研究科システム人間系専攻博士前期課程修了．同年石川島播磨重工業株式会社（現，株式会社 IHI）に入社し現在に至る．2019 年 7 月横浜国大環境情報研究院客員准教授（非常勤）兼務．制御工学やロボティクス，AI 関連技術の研究に従事．日本ロボット学会実用化技術賞等受賞，計測自動制御学会，日本機械学会各会員．

中田 雅也

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