Vol.2020-MPS-131 No.10
2020/12/17

Cartesian genetic programmingを用いた転移利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]

概要：本論文では，定義された目的関数のもとで，Cartesian genetic programming（CGP）を用いた転移利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．提案法では，積み付け候補から選択する基準として選択ルールを複数定義し，その実行順位を出力するモデルを CGP で構築する．類似問題へ転移利用するシミュレーション実験では，提案法がベースラインと同等の性能を導出できることを示す．これは，自動生成された積み付けアルゴリズムが，ヒトによる追加評価なしで，類似問題へ転移利用できる可能性を示す意義がある．

Automated construction of Transferable loading algorithm with Cartesian genetic programming

1. はじめに

混載積み付け作業では，積み付けるべき段ボール箱の種類や個数は毎回異なるため，作業の都度積み付けパターンの検討が必要となる．その際，積載率向上・使用パレット数の抑制を優先しつつも，段ボール箱の安定性や製品健全性，検品性にも同時に配慮する必要がある．現状では，これらの観点を考慮した積み付けは熟練した作業者の経験則を頼りに行われており，これに伴う人的・時間的な負担を削減するためには，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．

本論文で扱う積み付け問題は次の特徴がある．(1) 積み付けパターンの良否は定式化できるほど明確でないが，作業者は経験則から積み付けパターンの良否を判断できる．(2) 一般的に，積み付けるべき段ボール箱の種類や数が異なれば，好適な積み付けパターンも大きく変わりうる．(3) 積載率を向上させるために棒積みを採用する，荷崩れを防ぐためにピンホール積みを採用するなど，部分的な積み付けパターンは共通することが多い．

特徴 (1) より，主観にもとづく解同士の相対評価は可能であることから，本問題を対話型最適化進化計算（IEC）[1], [2] を用いて解くアプローチが考えられる．IEC とは，進化計算における解の評価を人間が代替する最適化手法であり，解に対する定量的な評価指標がなくても評価者の感性を反映した解を得ることができる．しかしながら，特徴 (2) より，状況が異なれば好適な積み付けパターンは変わるため，個別の問題に対して IEC の適用が必要となり，作業者にとって解の評価が大きな負担となるほか，積み付けパターン決定の自動化も果たせないという問題が残る．

この問題点を解決するものとして，著者らは，個別の問題に対して都度 IEC を適用し積み付けパターンを得るのではなく，IEC とアルゴリズム自動構築技術を融合し，転移利用可能な，熟練者の経験則を反映した積み付けアルゴリズム自体を自動生成することを目指している．アルゴリズム自動構築技術は，ハイパーヒューリスティクス（Hyper heuristics: HH）[3] ともよばれ，個別の問題の解ではなく，所定の問題クラスに対して有効なヒューリスティクス自体を求めるものである．近年では，遺伝的プログラミングを用いた方法（Genetic programming-based HH: GPHH）が，複雑なヒューリスティクスの表現が可能であることから盛んに研究されている [4], [5], [6]．HH におけるヒューリスティクスの最適化には進化計算アルゴリズムが使われることから上述した IEC との親和性も高い．一方で，一般的には HH は探索空間が膨大となり多くの解評価回数を要

[1] 横浜国立大学
Yokohama National University
[2] 株式会社 IHI
IHI Corporation
[a)] hiruta-yusuke-vn@ynu.jp

するため，HH と IEC を組合せる場合，上述した解評価に係る負荷はより深刻なものとなる．

そこで本論文では，少ない解評価回数でも有効に働く，転移利用可能な積み付けアルゴリズムの自動構築技術を提案する．ただし本論文では，著者らの最終目標に向けた要素技術の位置づけとして，対話型最適化計算を用いた作業者による解評価ではなく，事前定義した目的関数にもとづく解評価を採用する．本論文では一般的に HH では数千以上に設定される解評価回数を数百に制限した状況を想定する．少ない解評価回数でも妥当な積み付けアルゴリズムを構築するために，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，積み付けアルゴリズムの探索空間の膨大化を抑制する．棒積みあるいはピンホール積みを施した部分的な積み付けパターン群を列挙し，そこから好適なものを選択する積み付けアルゴリズムを構築する．つまり，「考えられる部分的な積み付けパターンのうち，どのような性質を持つ部分的パターンを選ぶべきか」という逐次的選択問題に帰着させ，探索空間の縮小を狙う．

(ii) 1 世代あたり少ない解個体数で有効に駆動する Cartesian genetic programming（CGP）[7] を用いて，ルール選択時に用いるルールの重要度を算出する数理モデルを自動構築する．CGP は進化戦略である $(1+4)$-ES を用いることが推奨されており [8]，これは本論文も採用するように 1 世代あたり 4 個体の解評価で済む．(i) と合わせ解個体数と世代数の削減が期待できる．

本論文の構成は以下のとおりである．2 章と 3 章で本論文が対象とする積み付け問題の定義と CGP の概要をそれぞれ説明した後，4 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構築技術について述べる．5 章では，自動構築されたアルゴリズムの転移利用時の性能を評価し，最後に 6 章で本論文の結論を述べる．

2. 問題設定

2.1 概要

段ボール箱の形状はすべて直方体であると仮定する．積み付けるべき段ボール箱の種類は一般に複数あり，種類ごとの形状（幅，奥行，高さ）は既知とする．段ボール箱はパレットとよばれる輸送容器に積み付けられる．パレットは典型的には平面寸法が 1100 mm × 1100 mm（T11 型）[9] の木製あるいは硬質プラスチック製の台である．

積み付けパターンを決定するにあたり，輸送コストを削減するためには使用するパレット数を抑えることが肝要であり，そのためには各パレットの積載率を上げることが求められる．積み荷の安定性の観点からは，積み付け高さを抑えかつ平準化することが望ましい．また 2.3 節に後述するとおり，段ボール箱の積み方も安定性に大きく寄与する．検品性の観点からは，同じ種類の段ボール箱の数を目視で容易に確認できることが望ましい．同じ段ボール箱を分散させて積み付けると検品性が低下するため，なるべく同じ種類の段ボール箱はまとめて積み付けることが望ましい．

2.2 積み付け空間と座標系

段ボール箱は，パレットの幅，奥行，最大積み付け高さで規定される直方体形状の空間（積み付け空間）の内部に積み付けていくものとする．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．以降で述べる「幅」「奥行」「高さ」は，それぞれ x, y, z 軸方向に対応する向きとする．

2.3 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」とよぶこととする．アイテムは複数の種類からなり，種類ごとに形状（幅，奥行，高さ）の情報が既知である．また，同一種類のアイテムを複数まとめて大きな直方体形状としたものを「ブロック」とよぶ．本論文では，積み付けパターンの計算に際してブロック単位の積み付けを考える．これにより，同じアイテムがまとめられることで，検品性に優れた積み付けパターンが得やすくなる．ブロックの作り方として，本論文では「棒積み」と「ピンホール積み」を考える．

3. Cartesian genetic programming

CGP で用いる遺伝子は，フィードフォワード型のネットワークモデルで表現され，演算ノードを行数 n_r，列数 n_c のグリッド上に配置する点が特徴であり，$N = n_r \times n_c$ 個の演算ノードを用いることなる．また，CGP では進化戦略を用いて目的関数を最小化するように遺伝子を最適化する．特に，$(1+4)$-ES を用いることが推奨されており [8], [10]，本論文もこの設定を用いる．具体的には，まず初期解を生成し，これらの適合度を目的関数で評価する．そして，適合度が最良値を持つ 1 つの個体を親個体として選択する．つぎに，世代数を 1 つ増加させ，この親個体に確率 μ で突然変異を適用し，4 つの子個体を新たに生成する．4 つの子個体と親個体の中から，最良値を持つ 1 つの個体を親個体と設定する．この操作を設定した最大世代数 G まで繰り返すことで最適化を進める．

4. 提案法

本章では，まず CGP をもとにした積み付けアルゴリズムの概要について述べ，つぎに，これを構成する各コンポーネントについて説明する．

4.1 積み付けアルゴリズムの概要

1 章で述べたように，提案法で構築される積み付けアルゴリズムは，積み付け可能なブロック群を列挙し，そこか

ら好適なものを選択する方式をとる．ここで積み付け可能なブロックとは，積み付け位置に積み付けることのできる棒積みあるいはピンホール積みを施したブロックである．このとき，4.2 節に示すように，考えられるブロックは複数存在する．そして，たとえばパレットごとの積載率など，これらのブロックの性質も異なる．そこで，本論文では「どの性質を持つブロックを選択すれば，最終的な積み付けパターンが所望の指標を満たすか」という逐次的選択問題を考え，CGP を用いてこれを選択するモデルを構築する．具体的には，「定めた性質を持つブロックと積み付け位置の組合せを選択する」という選択ルールを複数定義し，これらの選択ルールの実行順位を決定するモデルを CGP で生成する．まだ積み付けていない初期状態から，この操作を逐次的に繰り返すことで，全ての段ボール箱を積み付けた（最終的な）積み付けパターンを出力する．要約すると，以下の手順で積み付けパターンを出力する．

入力　積み付けるべきアイテムの種類ごとの個数，種類ごとの形状（幅・奥行・高さ），使用可能パレットの数，パレットの形状（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（Item list）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で積み付け可能な位置をすべて求め，この積み付け位置ごとに積み付け可能なブロックを列挙する．もし積み付け位置・ブロックの組合せが存在しない場合，計算失敗ステータスを出力する．

Step 4（積み付け位置とブロックの選択）　Step 3 で列挙した積み付け位置・ブロックの組合せから，CGP が出力した実行順で選択ルールを適用し，ひとつ選択する．

Step 5（積み付け実行）　Step 4 で選択された積み付け位置・ブロックの組合せをもとに積み付けを行う．Step 2 に戻る．

4.2　積み付け可能なブロック群の生成

ブロックの積み付け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．また，ブロックは座標原点に寄せるかたちで積み付けることで積み付け空間を効率的に利用する．ここで生成されるブロックは次の制約条件を満たすものとする．(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．具体的には，まず現在の積み付け状態をもとに積み付け可能な位置を求め，つぎに，積み付け位置と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組合せに対し，制約条件（1）～（3）を満足するブロックをすべて列挙する．ただし，同一アイテムでの各積み付け位置に対するブロックの候補において，積み方（棒積み・ピンホール積み）が同じであれば，検品性を考慮し，つぎに定める内包関係に対して，内包されないブロックの候補のみを積み付け可能なブロック群に加える．

A はB を内包する

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

と定義する．なお，A_x や B_y はそれぞれ，ブロック A の x（幅）方向のブロックの数，ブロック B の y（奥行）方向のブロックの数を表している．

4.3　CGP モデルの構築方法

CGP モデルに用いる入力として，現在の積み付け状態から計算できる 45 個の特徴量を用いる（**表 1**）．これらの特徴量は，積み付け状態を示す考えうる性質を計算している．また，積み付け可能なブロック群からブロックを選択する基準として，**表 2** に示す 8 個の選択ルールを定義する．ルールの選択重要度となる CGP の出力ノードは $O_1 \sim O_8$ の計 8 個であり，出力ノード O のインデックスがルールのインデックスに対応する．たとえば，O_1 は *Rule1* の選択重要度に相当する．表 2 において，*Rule1* と *Rule3* が積み付け可能位置を考慮した選択ルールである．また，**表 3** に CGP の演算ノードで用いる演算子をまとめる．以上より，提案法で用いる CGP は，45 個の特徴量から演算ノードで示される演算手順を用いて，8 個のルールの選択重要度をそれぞれ出力する関数を表現するモデルである．なお，ここで定義した特徴量と選択ルールは，積み付け状態や段ボール箱の種類・個数に依存せず，どのような状況でも適用できる．

上述の CGP モデルを 3 章で示した CGP のフレームワークで最適化する．CGP モデルの適合度を計算するために，その CGP モデルを 4.1 節で説明し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．そして，この積み付けパターンを目的関数で評価し，得られた目的関数値を CGP モデルの適合度として設定する．ここでは，特に最終的な積み付けパターンを生成する過程において，CGP モデルが出力した選択重要度を用いた，選択ルールの実行方法について述べる．

まず，4.1 節で述べた Step 3 で現在の積み付け状態に対する積み付け可能なブロック群を列挙した後を考える．現在の積み付け状態から 45 個の特徴量を計算し CGP モデルに入力し，$O_1 \sim O_8$ の出力値を計算する．そして，出力

表 1: CGP に用いる入力の定義
Table 1 Defition of inputs used in 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

表 2: 選択ルール
Table 2 List of selection-rules.

ID	定義
Rule1	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
Rule2	構成するアイテム数が最大となるブロックを選択する
Rule3	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
Rule4	ブロック単体の高さが最小となるブロックを選択する
Rule5	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
Rule6	ブロック単体の体積が最大となるブロックを選択する
Rule7	棒積みのブロックを選択する
Rule8	ピンホール積みのブロックを選択する

表 3: CGP に用いる演算子
Table 3 Operator list.

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	$1 \leq n \leq 45$	$\sum_{i=0}^{n} x_i$
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

値を選択重要度とみなし，重要度が高い順から選択ルールの実行順位を決定する．実行順序 1 位の選択ルールを列挙されたブロック群に適用し，そのルールに該当するブロックをまとめたブロックの部分群を形成する．ブロックの部分群の大きさが 1 であれば，そのブロックの利用を決定し Step 5 に進む．一方で，この大きさが 2 以上あれば，実行順序 2 位の選択ルールを形成した部分群に再度適用し，1 つに定まれば Step 5 に進む．実行順序 8 位を上限として，ブロックが 1 つに定まるまでこの操作を繰り返し適用する．また，実行順序 8 位の選択ルールを適用した時点で，ブロックの部分群の大きさが 2 以上であれば，ブロックの部分群に格納された最初のブロックを出力する例外処理を行う．

5. シミュレーション実験

5.1 実験方法

以下の 2 つの手順を行うことで，提案法が導出した積み付けアルゴリズムの転移利用時の性能を評価する．まず，定義した目的関数 $f(\boldsymbol{x})$ のもとで，1 つの訓練用問題を通して提案法で積み付けアルゴリズムを最適化する．つぎに，得られた積み付けアルゴリズムを複数の検証用問題に適用し，同一の目的関数でその適合度を評価する．

ここでは，3 つの訓練用問題 *train1, train2, train3* ，4 つの検証用問題 *test1, test2, test3, test4* および 2 つの目的関数 $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$ を定義する．たとえば，$f_1(\boldsymbol{x})$ のもとで *train1* を用いて積み付けアルゴリズムを導出し，*test1* ～ *test4* に適用することが 1 つの実験ケースであり，1 つの目的関数に対し訓練用問題の数だけ実験ケースが存在する．本論文で用いる訓練用問題および検証用問題は，**表 4** に示す積み荷をもとに，それぞれ**表 5** と**表 6** に示すように積み荷の種類と個数が異なる．積み荷の種類数ならびに個数の観点からは，*train1, train2, train3* および *test1, test2, test3, test4* の順で問題が難しくなる．また，本論文で用いる 2 つの目的関数を以下のように定義する．

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

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

ここで，$\boldsymbol{x}$ は積み付けアルゴリズムを実行することで得られた積み付けパターンである．また，$g_1(\boldsymbol{x}) \sim g_4(\boldsymbol{x})$ は，$\boldsymbol{x}$ から計算される以下の特徴量である．

表 4: 積み荷のサイズ [mm]
Table 4 Size of Load items [mm].

ID	幅	奥行	高さ
Item1	300	200	300
Item2	300	400	200
Item3	250	200	200
Item4	200	300	200
Item5	400	250	200
Item6	400	400	400
Item7	400	300	250
Item8	200	250	350
Item9	300	350	250
Item10	250	200	300
Item11	300	400	250
Item12	250	250	200
Item13	200	350	200
Item14	400	300	200
Item15	400	400	350
Item16	300	300	250
Item17	400	250	350
Item18	300	350	200

表 5: 訓練用問題
Table 5 Training problems.

train1	*train2*	*train3*
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.

test1	*test2*	*test3*	*test4*
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

- $g_1(\boldsymbol{x})$: 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

5.2 比較基準とパラメータ設定

実験プロセスの表記を簡略化するために，たとえば目的関数 f_1 のもとで *train1* を通して積み付けアルゴリズムを最適化するプロセスを $CGP_{f_1}^{train1}$ と表す．また，これで獲得したアルゴリズムを目的関数 f_1 のもとで *test1* に転移利用するプロセスを $CGP_{f_1\to f_1}^{train1}$ と表記する．

つぎに，提案法を評価する 2 つのベースラインについて述べる．例として，$CGP_{f_1\to f_1}^{train1}$ での性能（目的関数値）を評価することを考える．第 1 のベースラインは，転移利用時の性能を評価することを目的としたものであり，目的関数 f_1 のもとで *test1* に提案法を適用した場合（$CGP_{f_1}^{test1}$）に得られた積み付けアルゴリズムの性能と比較するものである．なお，*test1* に特化した $CGP_{f_1}^{test1}$ での積み付けアルゴリズムの方が良好な性能を導出することが期待できる一方で，$CGP_{f_1\to f_1}^{train1}$ での性能が *test1* に特化した積み付けアルゴリズムの性能に対し，どの程度再現できるかを評価する．第 2 のベースラインは，導出された積み付けアルゴリズムの妥当性を評価することを目的として，目的関数 f_2 のもとで *train1* を通して得られた積み付けアルゴリズムを，*test1* に転移させた場合（$CGP_{f_2\to f_1}^{train1}$）の性能と比較する．

パラメータ設定は次のとおりである．CGP の設定としては，ノードの行数 $n_r = 1$ ，ノードの列数 $n_c = 800$，突然変異確率 $\mu = 0.006$ とする [8], [10]．実験ケースごとに乱数のシードが異なる独立した 10 試行の実験を繰り返し，性能として設定した適合度の中央値を報告する．

5.3 実験結果

まず，目的関数 f_1 に設定した 3 つの訓練用問題における目的関数値の推移を**図 1** に示す．グラフの実線は中央値を表し，その実線にまたがる着色部の領域は四分位範囲である．なお，全ての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．これにより，CGP モデルの性能が順当に改善されていることがわかる．なお，考えうる積み付けパターンが膨大になることから，各目的関数を満たす最適解は特定できていない．世代数を 50 と制限していることもあり，最適解の性能と乖離があることが想定できるが，目的関数（ユーザ評価）を満たすように CGP モデルが適切に調整されていると判断できる．

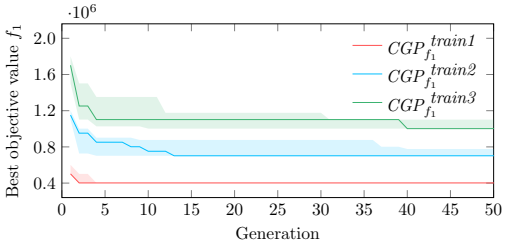
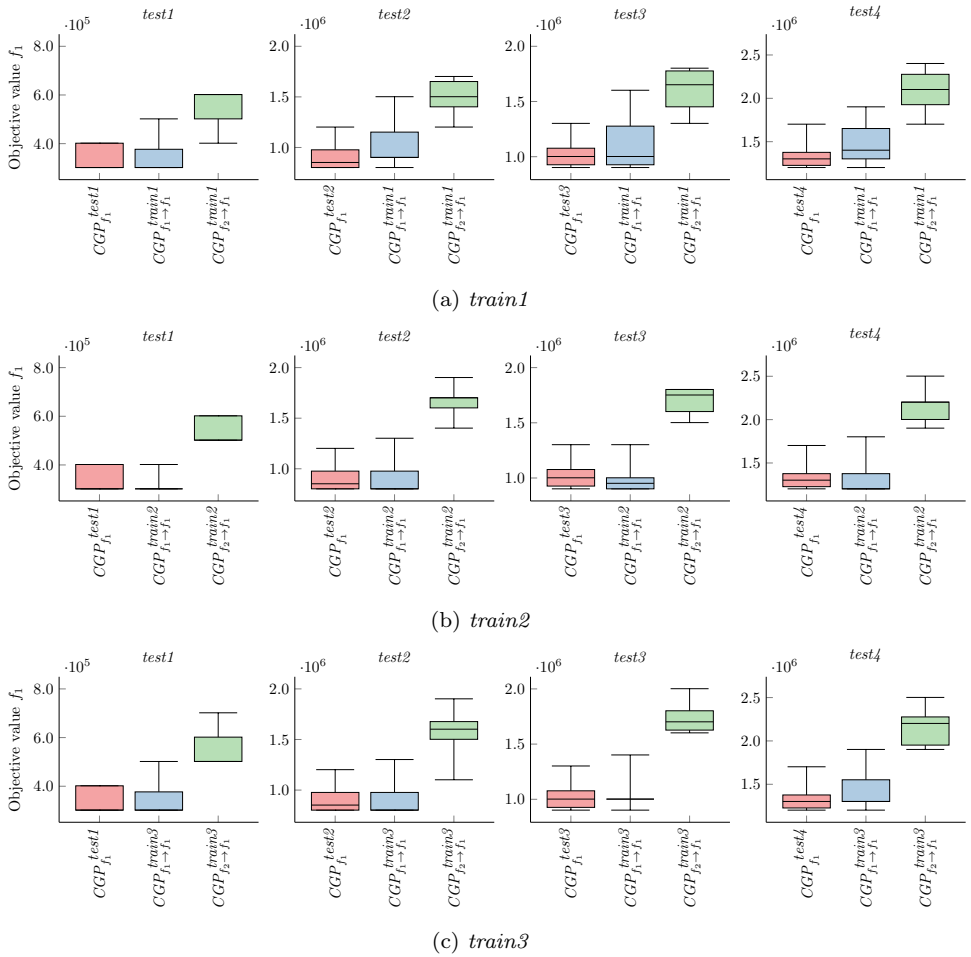


図 1: f_1 の調整プロセスにおける評価値の遷移
Fig. 1 Best objective value of f_1 over generations.

つぎに，訓練用問題より導出した CGP モデルを転移利用した場合の性能について議論する．目的関数 f_1 のもとで最適化した CGP モデルを 4 つの検証用問題に転移利用したときの結果を**図 2** に示す．図中の箱ひげ図は，10 試行で得られた性能の五数要約（最小値，第一四分位，中央値，第三四分位および最大値）を示している．図 2 より，転移利用した CGP モデル $CGP_{f_1\to f_1}^{train\ i}$ $(i = 1, 2, 3)$ は，概ね第 1 のベースラインである $CGP_{f_1}^{test\ j}$ $(j = 1, 2, 3, 4)$ に近い性能を導出していることがわかる．具体的には，$CGP_{f_1\to f_1}^{train1}$ の性能は，検証用問題が難しくなるにつれて $CGP_{f_1}^{test\ j}$ との乖離が大きくなる．一方で，$CGP_{f_1\to f_1}^{train2}$ および $CGP_{f_1\to f_1}^{train3}$ の性能は，4 つの検証用問題において第 1 のベースラインと近い性能を導出している．この傾向は，各訓練用問題の総アイテム数と関連している．これは，総アイテム数が多く複雑な問題である *train2, 3* が訓練用問題として効果的



(a) *train1*

(b) *train2*

(c) *train3*

図 2: f_1 における転移利用した場合の性能の五数要約
Fig. 2 Five number summary of objective function f_1.

に働いた結果と考えられる．一方で，第 2 のベースライン $CGP_{f_2 \to f_1}^{train\ i}$ $(i = 1, 2, 3)$ と比較すると， $CGP_{f_1 \to f_1}^{train\ i}$ の性能は改善している．すなわち，訓練用問題に対し過剰に調整されることなく，転移利用時も所定の目的関数 f_1 を小さくする CGP モデルを獲得できていることがわかる．

6. おわりに

本論文では，少ない解評価回数でも有効に働く，CGP を用いた転移利用可能な積み付けアルゴリズムの自動構築技術を提案した．実験結果は，獲得した CGP モデルを転移利用することで，ベースラインと競合する性能を導出でき，所望の目的関数を小さくする妥当な積み付けアルゴリズムを導出できることを示した．今後は，対話型最適化フレームワークとの融合を行う．

参考文献

[1] Takagi, H.: Interactive evolutionary computation, *Proc. Int. Conf. Soft Comput. Inf./Intell. Syst*, pp. 41–50 (1998).
[2] Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation, *Proceedings of the IEEE*, Vol. 89, No. 9, pp. 1275–1296 (2001).
[3] 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*, Springer, pp. 453–477 (2019).
[4] Nguyen, S., Zhang, M. and Johnston, M.: A genetic programming based hyper-heuristic approach for combinatorial optimisation, *Proceedings of the GECCO 2011*, pp. 1299–1306 (2011).
[5] 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 problem, *Kybernetes* (2014).
[6] 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. on Cloud Computing* (2020).
[7] Miller, J. F. and Harding, S. L.: Cartesian genetic programming, *Proceedings of the GECCO 2008*, pp. 2701–2726 (2008).
[8] Miller, J. and Turner, A.: Cartesian Genetic Programming, *Proceedings of the GECCO 2015 Companion*, p. 179–198 (online), DOI: 10.1145/2739482.2756571 (2015).
[9] 日本工業規格：JIS Z 0105：2015 包装貨物－包装モジュール寸法.
[10] Miller, J. F.: *Cartesian Genetic Programming*, pp. 17–34, Springer Berlin Heidelberg (2011).