Automatic Construction of Loading Algorithms With Interactive Genetic Programming

YUSUKE HIRUTA, KEI NISHIHARA, YUJI KOGUMA, MASAKAZU FUJII, AND MASAYA NAKATA

ABSTRACT
The design of a freight loading pattern is often conducted by skilled workers, who handle unquantifiable objectives and/or preferences. Our previous study presented an automatic construction technique for loading algorithms using genetic programming-based hyper-heuristics; however, this technique is only applicable to fully quantifiable loading problems. Thus, the approach described in this paper integrates an interactive framework with users into our previous technique to automatically construct algorithms that derive loading patterns adapted to user objectives and/or preferences. Thus, once a loading algorithm has been derived with user interactions, it can be reused to obtain the preferred loading patterns on other problems without any additional interactions. Experimental results show that the proposed algorithm can produce loading algorithms adapted for user preferences under a limit of 50 human interactions. Further, we also show that the derived loading algorithms can be applicable to different loading situations without any additional user interactions. Thus, these observations suggest the benefit of our approaches in reducing the burden placed on skilled workers for practical LPD tasks.

INDEX TERMS
Genetic programming, automatic algorithm construction, interactive evolutionary computation, loading optimization problem.

I. INTRODUCTION
In the logistics industry, the automation of warehouse operations, such as automated storage/retrieval systems and automatic guided vehicles, has become increasingly important [1], [2], [3]. A loading pattern design (LPD) is a major warehouse operation [4], [5], and its goal is to determine the freight loading positions on containers while satisfying certain objectives and conditions. Typically, multiple objectives are considered in the design of a loading pattern, for instance, a reduction in the transportation cost and an improvement in the inspection efficiency of loaded freight [6], [7]. These objectives may be in conflict with each other, and thus, their priorities may change depending on the status of the warehouse and/or loading.

LPD has been actively studied, particularly in the field of optimization, which includes bin-packing problems (BPPs) and container loading problems (CLPs) [8], [9], [10]. Along with their explicit objectives, these problems are often defined such that they can be solved computationally. For instance, BPPs intend to find the loading positions with the fewest bins, and CLPs intend to find a feasible solution in which all freight can be loaded into defined containers. However, in reality, LPD tasks often rely on the tacit knowledge of skilled workers to handle complex objectives and/or conditions, for instance, to manage the opposing objectives and specialize the objectives for dealing with uncommon conditions [11], [12], [13]. Thus, skilled workers play an important role in the LPD in practice.
However, in reality, LPD operations place a non-negligible burden on skilled workers because they must design the loading patterns for all loading states, which continuously change (freight is frequently carried in/out from the warehouse). Thus, the automation of the LPD process is crucial for reducing the burden of skilled workers.

For the automation of the LPD process, a challenge is to construct loading algorithms that are capable of automatically deriving patterns adapted to the knowledge of skilled workers. This is expected to reduce the burden of skilled workers during the LPD process, and once such a loading algorithm is constructed, it can be used under different loading states without the tacit feedback of the workers. For this challenge, the following two core approaches are useful (see Section II for a detailed review):

- Interactive optimization [14], such as interactive evolutionary computation (IEC) [15], is a promising approach to optimizing solutions satisfying user preferences. Its core idea is to integrate an optimization technique with a decision-making process, wherein users participate in a solution-evaluation process as an alternative to objective functions. Certain approaches have employed IEC in BPPs and CLPs to adapt loading patterns to user preferences [16], [17], [18].
- Hyper-heuristics (HHs) [19] are heuristic search techniques that automatically design a heuristic for a given problem. For instance, selective HHs intend to generate a high-level heuristic, which adaptively selects pre-defined heuristics dependent on the search status. Many studies have applied HHs to BPPs, as well as CLPs [20], [21], [22] and have demonstrated that HHs succeed in automatically deriving a powerful heuristic to optimize explicit objectives.

Based on these observations, the hybridization of IEC and HHs, namely interactive HHs, can be a possible approach to realize the automated design of loading algorithms embedded with the insights of skilled workers. However, to the best of our knowledge, there are few examples of interactive HHs applied to LPD tasks. In 2019, an interactive HH was introduced to solve standard BPPs, where an objective is to minimize the number of bins [16]. In brief, this study is aimed at improving the solving process of BPPs by utilizing user knowledge, rather than optimizing it for the user objectives/preferences. With this recent insight as a starting point, further exploration is needed for automating more realistic LPD processes conducted by skilled workers.

As a difficulty behind the hybridization of IEC and HHs for LPD tasks, HHs typically require a large number of iterations to obtain a good heuristic adapted for an objective function because the search space of the heuristic typically becomes complex and large [23]. However, a popular issue in the IEC field is the need to reduce the number of human interactions and thereby reduce the burden of users, wherein the number of interactions typically corresponds to that of iterations consumed by HH. Thus, toward the hybridization of IEC and HHs, an efficient approach to achieving interactive HHs is required to obtain good heuristics (i.e., a loading algorithm) while reducing the number of human interactions.

This paper presents an interactive HH method for LPD tasks, wherein the desired loading patterns depend on user preferences owing to multiple explicit/implicit objectives. The proposed method intends to automatically construct a loading algorithm capable of deriving the loading patterns adapted for user preferences. We previously developed a non-interactive HH approach for LPD tasks, called automatic construction technique of loading algorithms with cartesian genetic programming (ACLA-CGP) [24], which is used to construct a loading algorithm adapted for a given objective function such as a minimization of the number of pallets used. Although ACLA-CGP has not been designed for an interactive framework, it can construct a well-adapted loading algorithm for a given objective with a small number of user interactions. Thus, we expect that making ACLA-CGP interactive will be an effective approach toward tackling the possible difficulty of the aforementioned interactive HHs. Accordingly, in this study, an interactive framework is integrated with ACLA-CGP, the result of which is called i-ACLA-CGP. Because most components of the ACLA-CGP framework have a sufficient affinity toward an interactive optimization (see Section IV), we simply replace its solution-evaluation process to handle user evaluations through a GUI-based interaction scheme. The contributions of this paper are as follows.

- To the best of our knowledge, this study marks the first attempt at automatically constructing a loading algorithm adapted for human preferences. We demonstrated that integrating an interactive framework into our previous algorithm (ACLA-CGP), namely, i-ACLA-CGP, is a possible approach to realize this achievement.
- We have further shown that i-ACLA-CGP enables the construction of a preference-based loading algorithm under the limited resources of 50 human interactions. This suggests that our approach, that is, making ACLA-CGP interactive, can provide a practical benefit in decreasing the user burden regarding the interactions.

Note that powerful modern optimizers exist that may further boost the performance of i-ACLA-CGP, including the adaptation, ensemble, and voting mechanism of evolutionary algorithms [25], [26], [27], [28], [29], [30], [31], as well as surrogate-assisted approaches [32], [33], [34]. However, this paper focuses on demonstrating the feasibility of the automatic generation of loading algorithms through interactive optimization frameworks, and we leave possible improvements of i-ACLA-CGP as a topic of future study (see also Section VII).

This paper is organized as follows. Section II provides a brief review of HHs applied to LPD tasks. Section III describes our LPD task and the framework of ACLA-CGP. The presented algorithm of i-ACLA-CGP is described in Section IV. In Section V, experimental results with human subjects are reported to validate the performance of i-ACLA-CGP, and the loading patterns obtained are analyzed.
II. RELATED WORK

BPPs and CLPs have been extremely popular problems in the optimization field, and 1D, 2D, and 3D-BPPs/CLPs can be defined depending on the physical dimensions of the freight [35], [36], [37]. Because both problems are NP-hard, meta-heuristics, such as ant colony optimization [38], particle swarm optimization [39], and genetic algorithms [40], have been frequently employed. Certain studies have attempted to solve more complex BPPs and CLPs under practical conditions, such as crane time management [41] and different container types [42], [43].

Apart from solving the problems directly, BPPs and CLPs have also been frequently used as benchmark problems of HHs, as briefly summarized below.

A. NON-INTERACTIVE HYPER-HEURISTICS

Existing HHs can be classified as either selective HHs or generative HHs. Selective HHs are aimed at the construction of a heuristic used to select other pre-defined heuristics, whereas generative HHs aim at directly generating a heuristic.

For BPPs and CLPs, various selective HHs have been considered. Typically, HHs are designed to select certain freight and/or a bin as the selected freight is loaded. The consideration of both selections may be effective in boosting the performance of the selective HHs [44]. Further, genetic programming (GP) has shown potential as an optimizer of selective HHs, and it can be used to optimize a more complicated selection/combination procedure of the defined heuristics [45]. In [20] and [46], pre-defined heuristics were adaptively selected based on the feature values observed from the loading status. A score-based HH for BPPs was recently considered [47], where predefined heuristics are determined based on their estimated performance.

For generative HHs, GP-based HHs (GPHHs) have been a major approach to BPPs and CLPs [22], [23]. In [21], a GP model was used to calculate the freight scores when loading them under the current status, and the derived heuristic loaded the freight based on the freight scores. A possible challenge to this approach is an increase in the search space of the heuristics. To avoid this issue, a grammatical evolution [48] and island modeling [49] have been considered.

B. INTERACTIVE HYPER-HEURISTICS

As mentioned in the previous section, there are very few examples of interactive HHs applied to LPD tasks or even to the logistic industry field. In [16], users participated in a refinement process of the problem-solving BPP procedure, where a gamification scheme was employed as an interactive framework. A similar approach was attempted in [17] and [50]. As another example in the logistics industry, interactive HHs have been employed to a task assignment problem. Specifically, a genetic programming-based optimization system was developed, which optimizes a task assignment plan while considering the knowledge and preferences of the users [51].

Note that, apart from the few cases of interactive HHs, interaction optimization approaches have been actively employed in the logistics industry field, including the route design [52], [53] and warehouse layout [54], as well as various BPPs [18].

III. PRELIMINARIES

This section describes the LPD task considered in this paper. Subsequently, the ACLA-CGP framework is provided.

A. LPD TASK

Our LPD task is a variant of a 3D BPPs, and except for the objectives to be optimized, is similarly defined. Note that the mathematical modelling of 3D-BPPs can be found in [35]. The standard goal of a BPP is to find a loading pattern with the fewest bins, whereas, in essence, our problem allows users to define their preferred objectives. In fact, to quantitatively evaluate the algorithm performance certain objectives along with some instructions must be prepared for the users. However, users can manage the balance of these objectives based on their intuition, and they can consider additional objectives if necessary. The details of the experimental design are described in Section V. This paper supposes that freight are loaded onto pallets (rather than into bins) in line with the warehouse operations.

Specifically, as in standard BPPs, given \( N_p \) feasible pallets and \( N_f \) cuboid freight, a goal is to find a loading pattern optimized for the objectives, where all freight must be loaded onto any one of \( N_p \) pallets. Different types of freight can be defined. Figure 1 illustrates an example of loading patterns with four types of freight, which are colored differently. As shown in Figure 2, a pallet has an independent loadable...
Y. Hiruta et al.: Automatic Construction of Loading Algorithms With Interactive Genetic Programming

TABLE 1. Features used as inputs of the heuristic selector.

| ID  | Definition                                                                 |
|-----|---------------------------------------------------------------------------|
| x1  | Number of used pallets.                                                  |
| x2  | Minimum amount of freight per pallet.                                    |
| x3  | Maximum amount of freight per pallet.                                    |
| x4  | Average amount of freight per pallet.                                     |
| x5  | Minimum volume of loaded blocks per pallet [mm³]                         |
| x6  | Maximum volume of loaded blocks per pallet [mm³]                         |
| x7  | Average volume of loaded blocks per pallet [mm³]                         |
| x8  | Minimum loading ratio per pallet.                                        |
| x9  | Maximum loading ratio per pallet.                                        |
| x10 | Average loading ratio per pallet.                                        |
| x11 | Minimum value of z coordinate per pallet [mm]                            |
| x12 | Maximum value of z coordinate per pallet [mm]                            |
| x13 | Average value of z coordinate per pallet [mm]                            |
| x14 | Minimum number of composition freight types per pallet.                  |
| x15 | Maximum number of composition freight types per pallet.                  |
| x16 | Average number of composition freight types per pallet.                  |
| x17 | Amount of unloaded freight.                                              |
| x18 | Volume of loaded blocks [mm³]                                            |
| x19 | Loading ratio of total loaded volume to total pallet volume.              |
| x20 | Loading ratio of total volume of loaded bar-pattern blocks to total pallet volume. |
| x21 | Loading ratio of total volume of loaded pinwheel-pattern blocks to total pallet volume. |

TABLE 2. Pre-defined heuristics.

| ID  | Definition                                                                 |
|-----|---------------------------------------------------------------------------|
| h1  | Select a block loaded using minimum pallet index.                         |
| h2  | Select a block contains with the largest number of freight.               |
| h3  | Select a block with the shortest z coordinate when loaded.               |
| h4  | Select a block with the smallest height.                                  |
| h5  | Select a block with the largest top area.                                 |
| h6  | Select a block with the largest volume.                                  |
| h7  | Select a block with bar-pattern loading.                                 |
| h8  | Select a block with pinwheel-pattern loading.                            |

space of the same size as the others, and all freight must be loaded within this space. Once a loading pattern, that is, the positions of all freight, are determined, the pattern is evaluated using objective functions and/or user evaluations.

Typically, a problem-solving procedure of a BPP is designed to select an individual piece of freight and determine its position based on the current loading status, that is, a one-by-one loading strategy is used. In reality, a block-wise loading strategy has generally been the default for improving the freight inspection efficiency [55]. Specifically, considering that a block is defined as consisting of the same type of freight, its size is proportional to the amount of freight. Then, as exemplified in Figure 1, a practical loading pattern is often designed for loading as large a block as possible onto the same pallet. Aside from this, two major block designs have been employed, namely, a bar-pattern block and a pinwheel-pattern block, to improve the stability of the loading pattern (see Figure 3). These loading strategies can be an efficient heuristic used to relax the complexity of a BPP and is employed in ACLA-CGP.

B. ACLA-CGP

ACLA-CGP is a selective GPHH adapted for an LPD task, and is intended to derive a loading algorithm with a low computational cost by utilizing a block-wise loading strategy. A block is designed with either a bar or pinwheel pattern. Its basic idea is to optimize a heuristic selector embedded in the loading algorithm for selecting a proper block for each loading status, where this status is a temporal loading pattern until all freight has been loaded.

The rest of this section describes the design of the heuristic selector and the loading algorithm. Subsequently, we describe the optimization process of the heuristic selectors using a cartesian genetic programming (CGP) [56] approach.

1) HEURISTIC SELECTOR

A heuristic selector is utilized to determine the execution order of the pre-defined heuristics for each loading status. Specifically, a heuristic selector, $s$, is a mathematical function, $s : \mathbb{R}^D \to \mathbb{R}^H$, wherein an input, $x \in \mathbb{R}^D$, is a feature vector calculated from the current loading status, and an output, $y = s(x) \in \mathbb{R}^H$, represents the priority scores of $H$ pre-defined heuristics. The descending order of these scores is treated as the executing order of the pre-defined heuristics.

Specifically, 22 feature values and 8 pre-defined heuristics are employed, as summarized in Tables 1 and 2, respectively; thus, $D = 22$ and $H = 8$. Each feature value can be calculated from the loading status. As noted in Table 2, each heuristic defines the selection criterion of the blocks. Here, ACLA-CGP recursively executes multiple heuristics to identify a suitable block among the candidates (see Section III-B2).

As exemplified in Figure 4, the heuristic selector is represented using the expression of a CGP, that is, an $n_r \times n_c$ grid-like feedforward network. A mathematical operator is assigned to each node, and we use six operators and two constant values, as summarized in Table 3. Each node is
Algorithm 1 GetLoadingPattern (s)
1: Initialize the current loading status $p_c$
2: Set all freight to an unloaded list $\mathcal{L}$
3: while $\mathcal{L} \neq \emptyset$ do
4:   $x \leftarrow$ Set a feature vector calculated from $p_c$
5:   $y \leftarrow$ Set a score vector calculated from $s(x)$
6:   $B \leftarrow$ Set all possible blocks with their positions loadable onto $p_c$
7:   for $i = 1$ to $H$ do
8:      $h \leftarrow$ Set the heuristic having the $i$-th largest priority score $y$ in $y$
9:      $\hat{B} \leftarrow$ Set blocks in $B$ selected by $h^n$
10:     if $|\hat{B}| \neq 0$ then
11:        $B \leftarrow \hat{B}$
12:     end if
13:   end for
14:   $b^* \leftarrow$ Set a block from $B$
15:   Update $p_c$ by loading $b^*$ to $p_c$
16:   Update $\mathcal{L}$ by eliminating the freight composed in $b^*$
17: end while

TABLE 3. Pre-defined mathematical operators used in CGP.

| Symbol | Arity | Definition |
|--------|-------|------------|
| +      | 2     | $x_1 + x_2$ |
| −      | 2     | $x_1 - x_2$ |
| $\times$ | 2     | $x_1 \times x_2$ |
| $\div$ | 2     | $x_1 / x_2$ |
| mod   | 2     | $x_1 \mod x_2$ |
| $\sqrt{\cdot}$ | 1     | $\sqrt{x_1}$ |
| 0.1    | 0     | Const. 0.1 |
| 1.0    | 0     | Const. 1.0 |

responsible for outputting a result of the assigned operation with inputs obtained from connected predecessor nodes. Thus, a complete network represents a function by recursively concatenating the operator of each node.

2) LOADING ALGORITHM

Algorithm 1 describes a loading algorithm controlled by a heuristic selector $s$, denoted as GetLoadingPattern ($s$). For the initialization phase, the current loading status, $p_c$, is set to the initial status with no freight loaded, and all freight is registered to an unloaded list, $\mathcal{L}$.

For the main loop, a feature vector, $x$, and a score vector, $y$, are calculated from $p_c$ and $s(x)$, respectively. Subsequently, ACLA-CGP builds a set of candidate blocks, $B$, which consist of all possible blocks loadable onto the current loading status. A candidate block is generated with its loading position for a certain pallet, which can be variously generated in terms of the following perspectives: pallets, loading positions within a pallet, freight type, block patterns (bar/pinwheel-patterns), and block size. However, to reduce the number of their combinations, a loading position is restricted to the top-left corner if no block has been loaded onto a pallet; otherwise, it can be the top, front, or right corner of an existing block near the origin (see Figure 5). In addition, for each loading position, only the candidate block composed of the maximum amount of same-type freight is set as a candidate, providing artificial pressure to improve the inspection efficiency.

After $B$ has been constructed, ACLA-CGP determines a block to be loaded by recursively executing the heuristics in descending order in terms of the priority scores of $y$. Specifically, the top priority heuristic, $h$, having the largest priority score is executed and a subset of $B$, $\hat{B}$, comprising candidate blocks that satisfy the selection criterion of $h$, is built. Then, $B$ may be updated with $\hat{B}$ if multiple candidates exist in $\hat{B}$; in addition, the next heuristic having the second largest score (more generally, the $i$-th largest score, as in line 8) is applied to update $\hat{B}$. This recursive cycle is terminated if $\hat{B}$ has only one candidate. Next, ACLA-CGP selects a block from $B$, and updates $p_c$ by loading the selected block onto the current loading status. Finally, $\mathcal{L}$ is updated by eliminating the freight comprising the selected block.

3) OPTIMIZATION OF HEURISTIC SELECTOR

The refinement process of the loading algorithms corresponds to the optimization of the heuristic selectors. Further, the evaluation of a heuristic selector is substituted with that of the loading pattern obtained from GetLoadingPattern ($s$). Specifically, Algorithm 2 describes the optimization procedure of the heuristic selectors, where a (1+4) evolutionary strategy (ES) is employed as in the CGP framework.
Algorithm 2 ACLA-CGP

1: $s^* \leftarrow$ Initialize a heuristic selector  
2: $p^* \leftarrow$ GetLoadingPattern ($s^*$)  
3: $g \leftarrow 0$  
4: while $g < g_{\text{max}}$ do  
5: \hspace{1em} $\mathcal{S} \leftarrow \{s_1, \cdots, s_{\lambda}\}$  
6: \hspace{1em} for each $s \in \mathcal{S}$ do  
7: \hspace{2em} $p \leftarrow$ GetLoadingPattern ($s$)  
8: \hspace{2em} if $f(p)$ is better than $f(p^*)$ then  
9: \hspace{3em} $s^* \leftarrow s$  
10: \hspace{2em} end if  
11: \hspace{1em} end for  
12: $g \leftarrow g + 1$  
13: end while  
14: return $s^*$

\[\[\]

FIGURE 6. Framework of i-ACLA-CGP.

Initially, a heuristic selector, $s^*$, is randomly generated, and its loading pattern, $p^*$, is then obtained by executing GetLoadingPattern ($s^*$). Next, in the main loop, ACLA-CGP generates four new heuristic selectors as candidates by applying a mutation operator to $s^*$. For each candidate $s$, its loading pattern $p$ is derived and evaluated with a given objective function $f$. Finally, $s^*$ and $p^*$ may be updated as $s$ and $p$, respectively, if $f(p)$ is better than $f(p^*)$. The above main loop corresponds to one generation count, $g$, and is repeated until $g$ reaches the maximum generation, $g_{\text{max}}$.

IV. PROPOSAL

A graphical overview of i-ACLA-CGP is provided in Figure 6. In i-ACLA-CGP, a user participates in the evaluation process of the heuristic selectors. Thus, i-ACLA-CGP intends to optimize a heuristic selector and derive a loading pattern adapted for the user preferences.

A. CHARACTERISTICS

A main challenge to an interactive HH is reducing the burden on users during the optimization process. The ACLA-CGP framework already includes efficient mechanisms in this regard. Specifically, i-ACLA-CGP has the following characteristics.

1) Inheriting the search efficiency of ACLA-CGP.

i-ACLA-CGP is designed with almost the same framework as ACLA-CGP, with the same defined features and heuristics. The intention is to inherit its search efficiency, aiming at adapting a loading algorithm with as few user interactions as possible. ACLA-CGP can relax the complexity of LPD tasks through a block-wise loading strategy and the reduction in the number of candidate blocks. We validate this insight under a restricted setting of 50 iterations in Section V. Moreover, the user interaction scheme can be simplified by integrating the interactive framework directly to ACLA-CGP, as detailed below.

2) Simplification of user interaction.

Recalling that (1+4)-ES evaluates the candidates and identifies the fittest one at each generation, this process can be simplified under the interactive framework. Specifically, i-ACLA-CGP requires the user to only select the preferred loading pattern among the candidates rather than grading the candidates individually. Thus, i-ACLA-CGP utilizes a binary evaluation scheme, which is expected to reduce the burden on users in comparison with other popular schemes, for instance, ranking and grading schemes [57], [58]. Optimizing all possible preferences (corresponding to objective functions) may be unsuitable because various user preferences can exist, wherein some preferences may be extremely personalized and/or difficult to quantify. Thus, we employ a single objective CGP framework rather than multi-objective optimization frameworks.
Algorithm 3 i-ACLA-CGP

1: $s^* \leftarrow \text{Initialize a heuristic selector}$
2: $g \leftarrow 0$
3: \textbf{while } $g < g_{\text{max}}$ \textbf{do}
4: \hspace{1em} $S \leftarrow \{s^*, s_1, \ldots, s_4\}$ \hspace{1em} \text{Generate four heuristic selectors mutated from } s^*, \text{ and set them to a candidate set } S$
5: \hspace{1em} $P \leftarrow \{p^*, p_1, \ldots, p_4\}$ \hspace{1em} \text{Set loading patterns, } p, \text{ derived from } \text{GetLoadingPattern}(s) \text{ for each } s \text{ in } S$
6: \hspace{1em} \text{Display all loading patterns in } P \text{ on the GUI}$
7: \hspace{1em} $s^* \leftarrow \text{Set a heuristic selector that derives the loading pattern selected by the user}$
8: \hspace{1em} $g \leftarrow g + 1$
9: \textbf{end while}
10: \textbf{return } s^*$

FIGURE 7. GUI design.

B. PROCEDURE

As described in Algorithm 3, i-ACLA-CGP follows the same procedure as in ACLA-CGP up to the calculation of the loading patterns of the heuristic selectors (i.e., line 4).

Specifically, although our proposed framework optimizes the heuristic selectors (i.e., the CGP individuals), their quality (i.e., fitness) is evaluated based on whether the heuristic selectors can derive a loading pattern adapted for user preferences. As with ACLA-CGP, i-ACLA-CGP generates new candidates of the heuristic selectors based on the parent individual under the (1+4)-ES framework, as noted in line 4 of Algorithm 3. Next, to evaluate the quality of the heuristic selectors, the following two steps are conducted. First, i-ACLA-CGP calculates the loading patterns by executing the heuristic selectors, as described in line 5 of Algorithm 3, and those loading patterns are then displayed on the GUI (see Figure 7) in line 6. Subsequently, the user selects the most preferred loading pattern among those displayed in line 7. After the evaluation process based on use is completed, the heuristic selector that has derived the loading pattern selected by the user is set to the parent individual for the next generation.

This process is repeated until $g$ reaches $g_{\text{max}}$, and thus $g_{\text{max}}$ user interactions are conducted. Accordingly, the role of the user is only to select a loading pattern displayed on the GUI. Thus, with our approach, the user preference acts as an implicit objective function that defines the preferred loading patterns with binary decisions. Further, the preference can be considered as a black-box function because it might involve implicit knowledge, which is difficult to model.

Note that analyzing the computational complexity of the proposed algorithm is non-trivial because the (1+4)-ES of a CGP has not been revealed mathematically. To the best of our knowledge, only the time complexity of a simpler case, (1+1)-ES, has been discussed [59].

V. EXPERIMENT

This section validates the performance of i-ACLA-CGP in terms of the following perspectives:

- Under some instructions to users, we validate whether i-ACLA-CGP can derive a desired loading algorithm through the user interactions on a training problem.
- We validate whether a derived loading algorithm can derive a desired loading pattern even on untrained problem settings under different loading states.

Note that, although the users are not skilled, the main interest of this study is validating whether i-ACLA-CGP can derive a loading algorithm adapted for user preferences, regardless of their expertise. Also note that all users (as human subjects) gave written informed consent. The proposed method including the GUI interface and experiments are implemented using in-house codes with Python.

A. EXPERIMENTAL DESIGN

1) INSTRUCTIONS TO USERS

As mentioned in Section III, to quantitatively evaluate the performances of the algorithms using a sufficient number of examples, it is necessary to provide users common goals. Accordingly, we designed user instructions allowing users to consider secondary objectives in addition to a common goal if preferred.

Specifically, we define two frequently considered practical objectives:

- $f_1$ Minimize the number of pallets used in a loading pattern.
TABLE 4. Settings of training/test problems.

| Freight type | Width [mm] | Depth [mm] | Height [mm] | train | test1 | test2 | test3 | test4 | test5 |
|--------------|------------|------------|-------------|-------|-------|-------|-------|-------|-------|
| t1           | 300        | 200        | 300         | 100   | 120   | 110   | -     | -     | -     |
| t2           | 300        | 200        | 200         | 100   | 95    | 100   | -     | -     | -     |
| t3           | 250        | 200        | 200         | 160   | 150   | 130   | -     | -     | -     |
| t4           | 200        | 300        | 200         | 130   | 140   | 120   | -     | -     | -     |
| t5           | 400        | 250        | 200         | 80    | 70    | 90    | -     | -     | -     |
| t6           | 300        | 300        | 300         | 60    | 40    | 70    | -     | -     | -     |
| t7           | 400        | 300        | 250         | 50    | 70    | -     | -     | -     | -     |
| t8           | 200        | 250        | 350         | 120   | -     | 70    | -     | -     | -     |
| t9           | 300        | 350        | 250         | 110   | -     | 100   | -     | -     | -     |
| t10          | 300        | 300        | 200         | -     | -     | 110   | 100   | -     | -     |
| t11          | 400        | 200        | 300         | -     | -     | 140   | 110   | 80    | -     |
| t12          | 200        | 200        | 250         | -     | -     | 170   | 140   | 160   | -     |
| t13          | 200        | 250        | 300         | -     | -     | 100   | 110   | 120   | -     |
| t14          | 200        | 400        | 250         | -     | -     | 100   | 120   | 110   | -     |
| t15          | 250        | 250        | 250         | -     | -     | 60    | 50    | 80    | -     |
| t16          | 400        | 250        | 300         | -     | -     | -     | 60    | 40    | -     |
| t17          | 200        | 350        | 250         | -     | -     | -     | 110   | 100   | -     |
| t18          | 250        | 300        | 350         | -     | -     | -     | 120   | 120   | -     |
| t19          | 300        | 250        | 250         | -     | -     | -     | -     | 100   | -     |
| t20          | 250        | 250        | 350         | -     | -     | -     | -     | 90    | -     |
| t21          | 200        | 350        | 200         | -     | -     | -     | -     | 130   | -     |

| Amount of freight $N_f$ | 910 | 610 | 900 | 620 | 920 | 1,250 |
| Number of feasible pallets $N_p$ | 20  | 15  | 20  | 15  | 20  | 25    |

- $f_2$: Minimize the average height of loaded freight on pallets used in a loading pattern.

Here, $f_1$ is commonly used in BPPs, and $f_2$ improves the stability of the loaded freight. These objectives conflict with each other because minimizing the average height of the loaded freight corresponds to the increasing number of pallets used. Thus, adequate heuristics can also differ. For instance, the heuristic of $h_1$ produces bias in loading freight to the same pallets, improving $f_1$, whereas $h_3$ and $h_4$ produce bias in loading the freight to reduce $f_2$ (see Table 2).

In addition, we define the following three preferences:

- **Preference 1**: Moderately improve $f_1$ compared to $f_2$
- **Preference 2**: Balance both $f_1$ and $f_2$
- **Preference 3**: Moderately improve $f_2$ compared to $f_1$

A human subject is instructed to participate in the evaluation-process of i-ACLA-CGP while following an assigned preference. However, users are still permitted to consider additional objectives while considering the assigned preference. The additional objectives are reported using a questionnaire given out at the end of the experiments.

2) EXPERIMENT SCHEME

We employed 10 human subjects (male students in their 20s) from Yokohama National University, which are denoted as Subject A, ..., J. For each human subject, three experiments were conducted with three different preferences. Only the first five subjects, Subject A, ..., E, were allowed to consider additional objectives, and the last five subjects followed an assigned preference. This was aimed at validating the stability of the i-ACLA-CGP performance against the complexity of additional objectives.

Each experiment is composed of training and test phases. During the training phase, i-ACLA-CGP is executed with a subject, that is, it optimizes the loading algorithms through user interactions on a single training problem (train). The maximum generation, $g_{\text{max}}$, is set to 50, and thus the user must participate in 50 solution-evaluation steps. During the test phase, a derived loading algorithm is applied to five test problems with different loading status ($\text{test}_1$, ..., $\text{test}_5$). The training and test problem settings are summarized in Table 4, where the amount of freight to be loaded and the number of feasible pallets are described at the bottom of the table. For instance, for the training problem, 910 pieces of freight from among 9 different types must be loaded onto a maximum of 20 pallets.

3) EVALUATION SCHEME

The performance of i-ACLA-CGP was evaluated using the objective values of a loading pattern derived from the best heuristic selector, obtained at the 50-th generation for each problem. For visibility, each objective function is normalized using the possible maximum value, that is, the maximum number of pallets $N_p$ and the maximum loadable height (1,800 mm) for $f_1$ and $f_2$, respectively. For the additional objectives of Subject A, ..., E, we conducted a further analysis, as described in the next section. We used the following parameter settings: $g_{\text{max}} = 50$, $n_r = 1$, $n_c = 800$, and $\mu = 0.08$ [56].

As aforementioned, this study aims to validate whether i-ACLA-CGP can produce a loading algorithm that returns a loading pattern satisfying the user preferences. Typically, meta-heuristic algorithms are evaluated with multiple runs and different random seeds. However, in our case, conducting multiple runs for the same subject may over-evaluate the proposed algorithm, and the human subjects may determine the shortest way to obtain the preferred results through multiple trials. Thus, we designed the experiments such that all
10 human subjects followed the same preference, wherein
each subject corresponded to one trial, and we therefore tested
i-ACLA-CGP using 10 trials.

B. RESULT
Figure 8 reports the normalized objective values of a derived
loading pattern for each subject. The horizontal and vertical
axes indicate the normalized values of $f_1$ and $f_2$, respec-
tively. The circle, square, and triangle symbols correspond
to the objective values obtained under Preference 1, 2,
and 3, respectively. Note that the additional objectives defined by
Subject A, . . . , E are reported in the next section.

As an interpretation of this figure, when the curve
is downward-sloping, we can see that i-ACLA-CGP suc-
cessfully optimized the loading algorithms, satisfying the
assigned preferences. In the figure, this tendency can be
observed for all subjects except for Subject E, F, and J.
Note that, as discussed in Section V-A3, to test i-ACLA-CGP
with multiple trials for the same preference, we designed the
experiments such that 10 human subjects follow the same
preference, and thus the effectiveness of i-ACLA-CGP can be
confirmed if similar landscapes of the curve can be observed
for each human subject. Specifically, the following observa-
tions are obtained from Figure 8.

- For Preference 1 (moderately improving $f_1$ compared to
  $f_2$), the loading algorithms obtained have been success-
  fully adapted for this preference because the $f_1$ values
  obtained for Preference 1 tend to be smaller than that for
  Preference 2 and 3, and this tendency can be commonly
  observed for all human subjects.
- For Preference 3 (moderately improving $f_2$ compared to
  $f_1$), a similar tendency can be observed for all subjects
  except for Subject E, F, and J; in addition, the $f_2$ values
  (obtained for Preference 1) are smaller than those for
  Preference 1 and 2. Thus, this result indicates that for
  most of the human subjects (i.e., trials), i-ACLA-CGP
derived the loading algorithms adapted for Preference 3.
- For Preference 2 (balancing both $f_1$ and $f_2$), the
  $f_1$ and $f_2$ values obtained, except for Subject E, F,
  and J, are balanced compared to those obtained for
  Preference 1 and 3.
- The above results have been obtained under the limited
  resources of 50 human interactions. Further, the above
  insights can also be observed for the test problems.
  Thus, the derived loading algorithms are still capable of
deriving the preferred loading patterns under different
loading situations without additional user interactions.

Note that for Subject E, F, and J, the $f_2$ values
for Preference 3 are not always smaller than those for
Preference 2. We suspect that this is due to an inconsistency
in the evaluation process. These users might struggle to bal-
cance the objectives unless grasping the possible degree of
improvements of the objectives.

Supplementarily, Figure 9 shows the curves of the nor-
malized objective values over the generations during the
training process (Subject G). Note that the left, center, and
right figures show the curves for Preference 1, 2, and 3, respectively. The figures indicate that i-ACLA-CGP successfully improves the loading algorithms and satisfies each preference even during the early generations. For instance, for Preference 1 (moderately improving $f_2$ compared to $f_1$), the objective values of $f_1$ obtained are larger than those for Preference 1 and 3. Further, for Preference 2, the $f_2$ value gradually improves over the generations while maintaining the minimum value of $f_1$ discovered during a run. This is consistent with Preference 2, that is, balancing $f_1$ and $f_2$. Note that the execution times (including the evaluation time by the users) of the algorithm for each subject are $[80, 232, 145, 145, 85, 164, 64, 161, 112, 144]$ min for Subject A, B, $\cdots$, and J, respectively. The execution time greatly depends on the subject because each subject required a different evaluation time, i.e., some of the subjects spending time on other activities (e.g., going to the restroom and eating lunch).

The overall results empirically suggest that i-ACLA-CGP can produce loading algorithms adapted to the defined preferences under a limit of 50 human interactions. Further, the results on the test problems indicate the possibility that, once interacting with the user, i-ACLA-CGP can construct a ‘stand-alone’ loading algorithm applicable to different loading situations. These observations highlight the important benefit of our approaches in reducing the burden of skilled workers in practical LPD tasks. The skilled workers should design loading patterns for each situation.

### VI. CASE STUDY

This section provides a case-study for the additional objectives defined by Subject A, $\ldots$, E. The additional objectives defined by the subjects are summarized in Table 5. Subject A, B, and C defined essentially the same objective, i.e., minimizing the freight types per pallet, which is denoted as $f_3$; in addition, Subject B defined two objectives. Subject D considered maximizing the number of bar-pattern blocks for all defined preferences, which is denoted as $f_4$. By contrast, Subject E considered the stability of the loading patterns, which is difficult to evaluate quantitatively.

These additional objectives were not sufficiently considered under certain experimental cases. Specifically, Subject A and B reported that the objective of $f_3$ was considered only for Preference 1 and 3, respectively. Subject B and C failed to reflect the second objective for all preferences. Accordingly, they pointed out that their additional objective could not be considered because they prioritized the assigned preferences and/or the first additional objective. The objective of Subject E is difficult to evaluate quantitatively.

Accordingly, we further evaluated the loading algorithms derived from Subject A, B, and C in terms of $f_3$, $f_4$, and $f_4$, respectively.

### A. SUBJECT A

To quantify the objective of $f_3$, we define its objective values as the average number of freight types per pallet, calculated from the loading patterns (as evaluated in the previous section). Note that the pre-defined heuristic of $h_2$ produces bias to improve $f_3$ because it selects blocks containing as much freight (of the same type) as possible. Recall that Subject A considered $f_3$ only for Preference 1; in addition, the objective value of $f_3$ for each training/test problem with Preference 1 is reported in Figure 10. Note that $f_3$ should be minimized to satisfy the objective, and Average indicates the objective value of $f_3$ averaged for all subjects.

As shown in this figure, the objective value of $f_3$ of Subject A is better than that of Average for all training/test problems, except for test1. This indicates that, on average, the loading algorithm of Subject A has been adapted to reduce the

### TABLE 5. User-defined objectives.

| Subject ID | User-defined objective                                      |
|------------|-------------------------------------------------------------|
| A          | Minimize the freight type per pallet, $f_3$ (valid only for Preference 1) |
| B          | Minimize the freight type per pallet, $f_3$ (valid only for Preference 3) Select pawl-wheel-pattern block if the block is composed of more than two tiers (invalid for all Preference 1, 2, and 3) |
| C          | Minimize the maximum number of types of freight loaded on a pallet (invalid for all Preference 1, 2, and 3) |
| D          | Select a loading pattern consisting of as many bar-pattern blocks as possible from candidate patterns, $f_4$ (valid for Preference 1, 2, and 3) |
| E          | Select a stable loading pattern |

### FIGURE 9. Curves of normalized objective values over generations during the training process (Subject G). The left, center, and right figures show the objective values for Preference 1, 2, and 3, respectively.
number of freight types per pallet, in addition to the defined Preference 1. In addition to the above empirical observations, the reasonability of the derived loading algorithm can be further confirmed in terms of the execution order of the pre-defined heuristics. Table 6-a summarizes the execution order obtained when the trained loading algorithm (with Preference 1) was applied to test 2. As summarized in the table, the loading algorithm utilized a total of 27 steps to complete the loading of all freight defined in test 2. Each row indicates the execution order of the heuristics determined by the heuristic selector at the corresponding step. The heuristics of $h_1$, $\{h_3, h_4\}$, $h_2$ are colored in red, blue, and green, indicating that they improve $f_1$, $f_2$, and $f_3$, respectively. From the table, $h_1$ and $h_2$ tend to be executed with a higher priority than other heuristics; in addition, the priority of $\{h_3, h_4\}$ is lower than the other heuristics. This is consistent with the preferences of Subject A, improving $f_1$ and $f_3$ rather than $f_2$ for Preference 1. In particular, $h_2$ is stably prioritized compared to that of Subject F for instance (Table 6-b).

### B. SUBJECT B

Similar to the case of Subject A, we also compared the objective of $f_3$ for Preference 3. Figure 11 reports this comparison, and Table 7 summarizes the execution order obtained when the trained loading algorithm (with Preference 3) was applied to test 2.

From the figure and table, the objective value of $f_3$ of Subject B is better than that of Average for all training/test problems. Moreover, as shown in Table 7-a, the loading algorithm derived from Subject B has been successfully adapted to give priority to $\{h_3, h_4\}$, $h_2$ over $h_1$, improving $f_2$ and $f_3$ rather than $f_1$ under Preference 3. Note that the priority of $\{h_3, h_4\}$ is relatively low compared to that of Subject G for instance (see Table 7-b). However, this is due to the individual differences in strength when considering $f_2$. 

### TABLE 6. Execution order of pre-defined heuristics in test 2 (Subject A, Preference 1).

```
| Step | 1st  | 2nd  | 3rd  | 4th  | 5th  | 6th  | 7th  | 8th  |
|------|------|------|------|------|------|------|------|------|
| 1    | $h_8$ | $h_7$ | $h_1$ | $h_5$ | $h_4$ | $h_3$ | $h_2$ |
| 2    | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 3    | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 4    | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 5    | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 6-19 | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 20-21| $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 22   | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 23-26| $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 27   | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
```

### TABLE 7. Execution order of pre-defined heuristics in test 2 (Subject B, G, Preference 3).

```
| Step | 1st  | 2nd  | 3rd  | 4th  | 5th  | 6th  | 7th  | 8th  |
|------|------|------|------|------|------|------|------|------|
| 1    | $h_8$ | $h_7$ | $h_5$ | $h_1$ | $h_4$ | $h_3$ | $h_2$ |
| 2    | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 3-4  | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 5    | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 6-19 | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 10   | $h_6$ | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 11   | $h_6$ | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 12   | $h_6$ | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
| 23-24| $h_6$ | $h_7$ | $h_5$ | $h_1$ | $h_8$ | $h_4$ | $h_3$ | $h_2$ |
```
Finally, we quantify $f_4$ as the ratio of bar-pattern blocks to all blocks contained in a loading pattern, where $f_4$ should be maximized as intended by Subject D. Note that $h_7$ can be an effective heuristic to improve $f_4$ (see Table 2).

Figure 12 shows the objective values of $f_4$ for Preference 1, 2, and 3. Table 8 summarizes the execution order obtained when the trained loading algorithm is applied to tests.

From the figure and table, the objective value of $f_4$ of Subject D is better than that of Average for all training/test problems for Preference 1, and the execution of $h_7$ is relatively prioritized. Thus, the loading algorithm derived from Subject D has been successfully adapted to the preferences of Subject D. However, for Preference 2 and 3, the objective value of $f_4$ is less than average on certain problems, for instance, test5. This tendency can be observed in terms of the execution order of the heuristics. Specifically, as shown in Table 8, a relatively high priority is assigned to $h_7$ for Preference 1 and 2, but a low priority is assigned for Preference 3.

### VII. CONCLUSION

This paper presented an interactive HH approach to automatically construct a loading algorithm adapted for user preferences. Despite the complexity of this attempt, to the best of our knowledge, we successfully demonstrated its feasibility on the first attempt and achieved the following insight. First, the proposed algorithm can derive loading algorithms satisfying the defined preferences through user interactions under a restriction of 50 iterations. We showed that the loading algorithm can be adapted for two categories of preferences, i.e., managing opposing objectives and adding new objectives. Further, the derived loading algorithms can be reused under different loading states without any additional user interactions.

Apart from these contributions made in this study, a further area of interest is how to boost the performance of i-ACLA-CGP and obtain a more personalized loading algorithm under limited interactions with users. Accordingly, an immediate future study will be to conduct comparative and complexity analyses for the proposed framework.

### TABLE 8. Execution order of pre-defined heuristics in test5 (Subject D).

| Step | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| a) Preference 1 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 2-3 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 4 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 5-14 | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 15 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 1-24 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 25-28 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 29 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 30-42 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 43-45 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| b) Preference 2 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 1 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 2 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 3 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 4-5 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 6-7 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 8 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 9-16 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 17-18 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 19-24 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| c) Preference 3 | $h_7$ | $h_7$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 1 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 2 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 3 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 4 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 5-7 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 8-12 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 13 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 14-16 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 17-23 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
| 24 | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ | $h_8$ |
algorithm changes when increasing the mathematical operators used in the CGP nodes.

The results have also suggested that the proposed algorithm may potentially underperform owing to inconsistent evaluations, which is a common difficulty faced in interactive optimization. Thus, our future research will focus on integrating surrogate-assisted approaches into the i-ACLA-CGP framework, wherein surrogates are utilized as a recommendation system while learning the user preferences. Moreover, based on recent insight regarding a voting approach to evolutionary algorithms and constraint handling techniques, we may further boost our approach by utilizing such methods to capture more complex user preferences. These techniques may be useful if we suppose the user preferences as constraints that the loading algorithms must satisfy. Further, inspired by recent advances in adaptive evolutionary algorithms, the adaptive tuning of the CGP hyper-parameters may boost its performance and obtain loading algorithms more personalized for the user preferences.

REFERENCES

[1] A. Dekhne, G. Hastings, J. Murnane, and F. Neuhauß, “Automation in logistics: Big opportunity, bigger uncertainty.” McKinsey Q. Tech. Rep., Apr. 2019, pp. 1–12. [Online]. Available: https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/automation-in-logistics-big-opportunity-bigger-uncertainty

[2] M. Hompel and T. Schmidt, Warehouse Management: Automation and Organization of Warehouse and Order Picking Systems. Berlin, Germany: Springer, 2006.

[3] W. Echelmeyer, A. Kirchheim, and E. Wellbrock, “Robotics-logistics: Challenges for automation of logistic processes,” in Proc. IEEE Int. Conf. Autom. Logistics, Sep. 2008, pp. 2099–2103.

[4] C.-F. Chien, C.-Y. Lee, Y.-C. Huang, and W.-T. Wu, “An efficient computational procedure for determining the container-loading pattern,” Comput. Ind. Eng., vol. 56, no. 3, pp. 965–978, Apr. 2009.

[5] R. Morabito, S. R. Morales, and J. A. Widmer, “Loading optimization of palletized products on trucks,” Transp. Res. E, Logistics Transp. Rev., vol. 36, no. 4, pp. 285–296, Dec. 2000.

[6] S.-T. Jung and U. Caroline University, “A study on the effect of packaging design considering SCM aspects on logistics efficiency (focusing on the core of domestic a company),” KOREAN J. Packag. Sci. Technol., vol. 26, no. 1, pp. 11–17, Apr. 2020.

[7] M. Gajda, A. Trivella, R. Mansini, and D. Pisinger, “Evolving bin packing heuristics with genetic programming,” in Proc. 9th Annu. Conf. Genet. Evol. Comput. Evol. Comput. (GECCO), 2009, pp. 931–938.

[8] K. Burke, M. Hyde, G. Kendall, and J. Woodward, “Automatic heuristic generation with genetic programming: Evolving a jack-of-all-trades or a master of one,” in Proc. Int. Conf. Genet. Evol. Comput., 2007, pp. 1559–1566.

[9] E. K. Burke, M. Hyde, and G. Kendall, “Evolving bin packing heuristics with genetic programming,” in Parallel Problem Solving from Nature-PPSN IX. Berlin, Germany: Springer, 2006, pp. 860–869.

[10] Y. Hiruta, K. Nishihara, Y. Koguma, M. Fujii, and M. Nakata, “Automated construction of transferable loading algorithm with Cartesian genetic programming,” Trans. Math. Model. Appl., vol. 14, no. 3, pp. 11–26, 2021.

[11] G. Wu, X. Wen, L. Wang, W. Pedrycz, and P. N. Suganthan, “A voting-mechanism-based ensemble framework for constraint handling techniques,” IEEE Trans. Evol. Comput., vol. 26, no. 4, pp. 646–660, Aug. 2021.

[12] W. Qi, J. Zhu, G. Wu, H. Chen, W. Pedrycz, and P. N. Suganthan, “Ensemble many-objective optimization algorithm based on voting mechanism,” IEEE Trans. Syst., Man, Cybern. Syst., vol. 52, no. 3, pp. 1716–1730, Mar. 2020.

[13] S. S. Al-Gharaibeh, M. Z. Ali, M. I. Daoud, R. Alazrai, H. Abdel-Nabi, S. Hrizek, and P. N. Suganthan, “Real-parameter constrained optimization using enhanced quality-based cultural algorithm with novel influence and selection schemes,” Inf. Sci., vol. 576, pp. 224–273, Oct. 2021.

[14] M. Z. Ali, N. H. Awad, R. G. Reynolds, and P. N. Suganthan, “A balanced fuzzy cultural algorithm with a modified levy flight search for real parameter optimization,” Inf. Sci., vol. 447, pp. 12–35, Jun. 2018.

[15] A. W. Mohamed and P. N. Suganthan, “Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation,” Soft Comput., vol. 22, no. 10, pp. 3215–3235, 2018.

[16] V. Palakonda, R. Mallipeddi, and P. N. Suganthan, “An ensemble approach with external archive for multi- and many-objective optimization with adaptive mating mechanism and two-level environmental selection,” Inf. Sci., vol. 555, pp. 164–197, May 2021.

[17] N. A. Khan, M. Sulaiman, A. J. Aljohani, M. A. Bakar, and Miftahuddin, “Mathematical modeling of Electrohydrodynamic flow in a circular cylindrical conduit by using neuro evolutionary technique,” Energies, vol. 14, no. 22, p. 7774, Nov. 2021.

[18] N. A. Khan, M. Sulaiman, C. A. T. Romero, and F. S. Alshammari, “Analysis of nanofluid particles in a duct with thermal radiation by using an efficient Metaheuristic-driven approach,” Nanomaterials, vol. 12, no. 4, p. 637, Feb. 2022.
J. Levine and F. Ducatelle, “Ant colony optimization and local search for bin packing and cutting stock problems,” J. Oper. Res. Soc., vol. 55, no. 7, pp. 705–716, Jul. 2004.

D. S. Liu, K. C. Tan, S. Y. Huang, C. K. Goh, and W. K. Ho, “On solving multiobjective bin packing problems using evolutionary particle swarm optimization,” Eur. J. Oper. Res., vol. 190, no. 2, pp. 357–382, Oct. 2008.

A. Stawowy, “Evolutionary based heuristic for bin packing problem,” Comput. Ind. Eng., vol. 55, no. 2, pp. 465–474, Sep. 2008.

A. Lodi, S. Martello, and D. Vigo, “Recent advances on two-dimensional bin packing and cutting stock problems,” J. Oper. Res. Soc., vol. 55, no. 9, pp. 705–716, Jul. 2004.

A. Silva-Gálvez, E. Lara-Cárdenas, I. Amaya, J. Cruz-Duarte, and J. Ortiz-Bayliss, “A preliminary study on score-based hyper-heuristics for solving the bin packing problem,” Expert Syst. Appl., vol. 137, pp. 471–492, Dec. 2019.

N. Pillay, “A study of evolutionary algorithm selection hyper-heuristics for the one-dimensional bin-packing problem,” South Afr. Comput. J., vol. 48, pp. 31–40, Jun. 2012.

S. Nguyen, M. Zhang, and M. Johnston, “A genetic programming based hyper-heuristic approach for combinatorial optimisation,” in Proc. 13th Ann. Conf. Genetic Evol. Comput. (GECCO), 2011, pp. 1299–1306.

K. Sim and E. Hart, “Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model,” in Proc. 15th Ann. Conf. Genetic Evol. Comput. Conf. Companion, 2016, pp. 111–112.

P. Ross, J. G. Marín-Blázquez, S. Schulenburg, and E. Hart, “Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics,” in Proc. Genetic Evol. Comput. Conf., 2003, pp. 1295–1306.

A. Sosa-Ascencio, H. Terasshima-Marín, J. C. Ortiz-Bayliss, and S. E. Conant-Pablos, “Grammar-based selection hyper-heuristics for solving irregular bin packing problems,” in Proc. Genetic Evol. Comput. Conf. Companion, 2016, pp. 111–112.

N. Ross, E. Keedwell, and D. Savic, “Human-derived heuristic enhancement of an evolutionary algorithm for the 2D bin-packing problem,” in Proc. Int. Conf. Parallel Problem Solving Nature, 2020, pp. 413–427.

S. Nguyen, M. Zhang, D. Alahakoon, and K. C. Tan, “People-centric evolutionary system for dynamic production scheduling,” IEEE Trans. Cybern., vol. 51, no. 3, pp. 1403–1416, Apr. 2019.

D. Anderson, E. Anderson, N. Lesh, J. Marks, K. Perlin, D. Rataczak, and K. Ryall, “Human-guided simple search: Combining information visualization and heuristic search,” in Proc. Workshop New Paradigms Inf. Visualization. Manipulation Conject 8th ACM Int. Conf. Inf. Knowl. Manage., 1999, pp. 21–25.

D. E. Weldon, “Human-assisted logistics optimization (HALO): Support for timely logistics decisions-making,” in Proc. Int. Symp. Aviation Psychol., 2009, pp. 557–562.

R. Furmann, B. Furmannova, and D. Więcek, “Interactive design of reconfigurable logistics systems,” Proc. Eng., vol. 192, pp. 207–212, Jan. 2017.

J. F. Launderie, “Unitizing goods on pallets and slipsheets,” U.S. USDA Forest Service, Forest Products Lab., USA, Tech. Rep., 1986, vol. 52.

J. F. Miller, Cartesian Genetic Programming, Berlin, Germany: Springer, 2011, pp. 17–34.

C. Parreño-Torres, R. Alvarez-Valdes, R. Ruiz, and K. Tierney, “Minimizing crane times in pre-marshalling problems,” Transp. Rev. E, Logistics Transp. Rev., vol. 137, May 2020, Art. no. 101917.

X. Xiang, C. Yu, H. Xu, and S. X. Zhu, “Optimization of heterogeneous container loading problem with adaptive genetic algorithm,” Complexity, vol. 2018, pp. 1–12, Nov. 2018.

R. R. Júnior, H. H. Yanasse, R. Morabito, and L. Junqueira, “A hybrid approach for a multi-department container loading problem,” Expert Syst. Appl., vol. 137, pp. 471–492, Dec. 2019.

S. Nguyen, M. Zhang, and D. Alahakoon, “A study of evolutionary algorithm selection hyper-heuristics for the one-dimensional bin-packing problem,” South Afr. Comput. J., vol. 48, pp. 31–40, Jun. 2012.

P. Ross, J. G. Marín-Blázquez, S. Schulenburg, and E. Hart, “Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics,” in Proc. Genetic Evol. Comput. Conf., 2003, pp. 1295–1306.