

Solving an Optimization Problem of Image View Layout with Priority using Heuristic Approach

Lely Hiryanto¹, Andhika Putra Wirawan², Viciano Lee³

lelyh@fti.untar.ac.id¹, andhika.535210010@stu.untar.ac.id², viciano.lee@city.ac.uk³

^{1,2} Department of Informatics Engineering, Tarumanagara University, Banten, Indonesia

³ Department of Mathematics and Computer Science, City University of London, UK

ABSTRACT

The image view layout with priority (IVLP) problem focuses on efficiently arranging picture cards of uniform height but varying widths into the minimum number of 2D frames or display sets and prioritizing images with higher priority to be placed at the earlier displays. We mathematically modeled IVLP using integer linear programming. To approximate IVLP solutions, we introduce a greedy-based heuristic, Best-Fit-IVLP (BFI), and a swarm optimization algorithm, Ant Colony Optimization (ACO). BFI allocates picture cards in descending order of priority and width for each display line, seeking another card that can optimally fill the remaining space on each line. In contrast, ACO randomly arranges cards from high to low priority within every line. Experimental results using different numbers of SVG images indicate that BFI and ACO generate solutions close to optimal. BFI demonstrates superior practicality, executing significantly faster than ACO; for 160 images, BFI runs in 0.00044 seconds compared to ACO's 117.93 seconds. Both BFI and ACO achieve space utility rates ranging from 0.578 to 0.8. While BFI consistently produces the same card arrangement, ACO offers diverse arrangements for identical optimal display set counts and space utilization.

Keywords: Greedy-based heuristic; Image View Layout with Priority; Swarm Optimization; User Interface;

Article Info

Received : 11-02-2025

This is an open-access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

Revised : 21-04-2025

Accepted : 26-06-2025



Correspondence Author:

Lely Hiryanto
Department of Informatics Engineering,
Tarumanagara University,
Jl. S. Parman No.1, Grogol, Jakarta Barat, 11440
Email: lelyh@fti.untar.ac.id

1. INTRODUCTION

Bad experiences due to poor user interface (UI) layout can cause a distressful feeling for users, and they should be avoided when dealing with cognitively disabled users, such as people with autism spectrum disorder (ASD) [1][2]. One aspect of UI layout which is the focus of this paper is the arrangement of a set of two-dimensional (2D) elements in a UI frame. Here, each 2D element is a card holding different components together (grouping together heterogeneous items) such as images, text, action buttons, and icons. For our work, the card includes only an image with a certain width and height and a text label. From this point forward, we call the card a picture card. In general, the size of all cards is scaled to a fixed width and height when they are being arranged in a UI frame for the sake of tidiness. However, scaling in such a way makes an image with a wider size look smaller than those that are slimmer. This inconsistency may distract autistic people because, in general, they can be easily attracted to detail rather than the whole object [1]. Furthermore, minimizing scrolling so that users do not assume what information might be on the next page or display can make an application such as a website or a mobile app more accessible for autistic individuals [1]. In addition, packing

cards optimally and efficiently improves app loading times and responsiveness, which is particularly important for users who may have limited patience or attention spans [1].

This paper studies a special case of UI elements layout for an augmented and alternative communication (AAC) application for users with non-verbal communication. In particular, the picture card collection feature, which is a display of the cards in a predefined size of a UI frame. Fig. 1 shows examples of display layout of picture cards in three different mobile-based AAC applications: (a) AAC Cboard [3], (b) Leeloo [4], and (c) VICARA [5][6]. Note that any tapping action on each picture card will generate speech according to the text label on the card. In this way, users with speech difficulties can communicate their feelings and needs to any person close to them.



Figure 1. Picture card layout in Cboard, Leelo and VICARA 2.0 apps

As can be seen in Fig. 1, the space to display the picture cards is bounded by the device screen. Therefore, the available picture cards cannot be displayed at once. Moreover, to improve the accessibility of autistic users, some picture cards that are frequently used or are compulsory for learning or important for their speech therapy should be displayed at the earliest displays [1]. In this case, those cards should be prioritized. This study seeks to maximize the use of the display frame space to fit in as many images as possible in such a way that the number of users swiping from the first set to the last set of images, termed "display sets," is minimal, and at the same time ensure that images with higher priority are placed at the earlier display sets.

This work extends our previous work in [7] that introduces the novel optimization problem called image view layout (IVL) in a 2D-UI container or frame. IVL aims to minimize the number of display sets subject to a fixed width and height of the 2D frame, with the picture cards inside the frame and not overlapping with each other, and there is no empty (no picture card) display between its former and/or later display containing picture cards. Our work adds one more constraint to the IVL problem, which is a priority for each picture card. This constraint must guarantee that picture cards with higher priority are displayed earlier than those with lower priority. Adding this constraint increases the complexity of IVL. We called our optimization problem IVL with Priority or IVLP.

Similar to IVL, IVLP is a special case of a well-known two-dimensional bin packing problem, or 2DBPP [8]. Given a collection of small rectangles (items) with different widths and heights, 2DBPP optimizes their allocation in a minimum number of identical rectangles (bins) in width and height without having them overlap. Note that 2DBPP assumes all items have the same priority. Therefore, the solution of 2DBPP might place a set of picture cards with higher priority in the later display instead of in the earlier display filled with cards of lower priority. On the other hand, IVLP ensures the earlier displays contain cards with higher priority. The 2D bin packing problem [8][9][10] is a combinatorial problem that has been proven to be non-polynomial (NP)-hard. Adding a priority constraint to the problem increases its hardness.

In [9], some proposed exact techniques for 2DBPP have been demonstrated to solve only up to 100 rectangular pieces, and occasionally they are unable to solve as few as 20 items. However, compared to exact approaches, approximation methods such as greedy algorithms [11][12][13], meta-heuristics [14][15][16], reinforcement learning [17], and deep learning [18] can offer an approximate solution for a large number of items with a substantially faster runtime.

Chen *et al.* [15] addresses a user interface layout problem that minimizes the unused space of a large rectangular UI container by allocating as many as possible a set of small and non-overlapped rectangular display components that vary in their width and height. They regard their problem as 2DBPP. The authors [15] approach the optimal solution using the Firefly algorithm. A study in [18] addresses an optimization of mobile UI layout, which is the arrangement of UI components in a mobile device in such a way that it minimizes the task completion time and error rate when users interact with the components. They use a deep learning approach called gradient descent to find its near-optimal solution. Another work in [19] addresses a problem of arranging items in a number of identical 2D bins robotically, as minimally as possible, where the items are weakly heterogeneous, meaning they have very few types. The authors in [19] proposed a novel approximate algorithm called neural-driven constructive heuristic.

Our work is closely related to work by Chen *et al.* [15]. While the authors arrange UI elements with various sizes in a 2D-UI frame with predefined width and height, our IVLP problem allocates a set of picture cards in the frame where every image has the same height but different widths, and thus, the wider images will be displayed better in terms of their proportion to the width. Furthermore, every card varies in priority, where the card with the highest priority is ensured to be allocated in the earliest frame, i.e., displayed as early as possible, compared to the other cards with the lower priority.

Some possible real applications of IVLP include image symbol optimization in AAC apps, image gallery optimization, and sprite sheet generation for game assets. AAC apps use frames containing picture cards. The optimal number of picture cards and their arrangement can be prioritized based on the user's visual acuity, motor skills, cognitive load, and the frequency or importance of specific image-based vocabulary [20]. The optimization of image galleries focuses on displaying a collection of images or photos, especially when they have different sizes and priorities based on time, category, favorites, and other personal references, in a visually appealing and space-efficient manner. Sprite sheet generation for game assets [21] is a single large image that contains multiple smaller images, or sprites, used in 2D game development. Its optimization deals with reducing draw calls and network overhead by minimizing the overall sheet size. Furthermore, some assets could be prioritized to be on the most easily accessible parts of the sprite sheet.

In comparison with 2DBPP, IVLP introduces three main differences. First, each 2D item or picture card in IVLP has a priority that can be varied with the other items. On the contrary, items in 2DBPP have no priority. Second, IVLP places a set of picture cards with higher priority in the earlier displays, while 2DBPP does not include that requirement. Third, IVLP considers each item to have the same height but different widths, while 2DBPP considers items that vary in their height and width. By fixing the height, IVLP ensures the arrangement of the picture cards looks neat.

Our IVLP is a general version of our previous optimization, IVL, proposed in [7]. IVLP introduces one constraint, which is some picture cards may have different priority than the others. Cards with the smallest priority value must be displayed at the earliest, i.e., allocated in the earlier UI frames, than other cards with the larger priority. To the best of our knowledge, IVLP is a new optimization problem extending 2DBPP by including the priority constraint of the items allocated in the bin. Furthermore, our work is the first to use ant colony optimization to approach a close-to-optimal solution of IVLP.

The remainder of this paper is organized as follows. Section 2 discusses steps taken in optimization consisting of problem, parameters, and variables definition; mathematical formulation in Integer Linear Programming (ILP) for the IVLP problem; and two approximation solutions for the problem. We evaluate and discuss the two approximate solutions and their performance comparison by using the optimal solution produced by ILP in Section 3. Section 4 concludes the paper.

2. RESEARCH METHOD

We follow the six general steps of operations research methodology as described in [22] to perform the optimization study. They are (i) problem description, (ii) parameters and variables definition, (iii) mathematical modeling, (iv) optimal solution, (v) interpretation, and (vi) validation. This section discusses the first four steps, while the last three steps are described in Section IV.

2.1. Problem Description

The objective of IVLP is to minimize the number of users swiping to different sets of picture cards, where the cards are allocated in 2D UI frames with a fixed width and height, and those with the higher priority must be displayed as early as possible compared to cards with lower priority. IVLP is a special case of 2DBPP, where the item is the picture card with a fixed height, while the bin is the display set, i.e., a UI frame containing a set of picture cards. Furthermore, IVLP guarantees that each picture card must be packed into exactly one display set. Moreover, IVLP considers each card might have a different priority value. We assume priorities are discrete and ordered, such that a higher numerical value indicates higher priority. To view picture cards with higher priority as early as possible, we ensure the total priority value of the cards allocated in the earlier display set is larger than or equal to the later display sets. Therefore, we set the priority value of the higher priority significantly larger than the priority value of the lower priority. Finally, IVLP ensures that every card lies inside the frame. If IVLP does not fix the card's height and the priority value for all cards is the same, then IVLP is 2DBPP, which is an NP-hard problem.

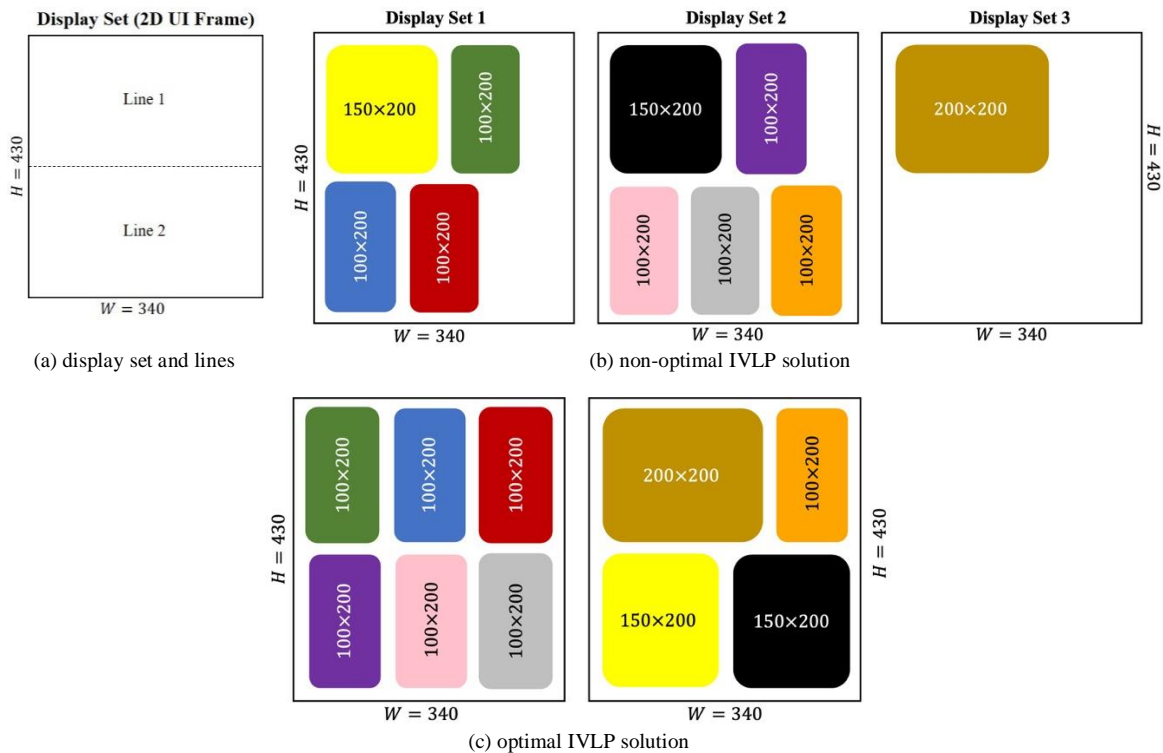


Figure 2. Problem Illustration of Image View Layout with Priority

Fig. 2 provides an illustration to our IVLP problem. Given a UI Frame with its width and height size ($W \times H$), e.g., in pixels, is 340×430 . There are ten picture cards having their respective size ($w \times h$) are as follows: 150 x 200 (yellow), 100 x 200 (green), 100 x 200 (blue), 100 x 200 (red), 150 x 200 (black), 200 x 200 (brown), 100 x 200 (purple), 100 x 200 (pink), 100 x 200 (gray), and 100 x 200 (orange). For better display, each card will have an extra padding space, e.g., 5 pixels, for its four margins, i.e., top, bottom, left

and right. We divide the UI frame into L lines, where $L = \lfloor H/h \rfloor$. Fig. 1(a) shows the sample of the display set or 2D UI frame. With $H = 430$ and the item's fixed height $h = 200$ pixels, $L = \lfloor H/h \rfloor = \lfloor 430/200 \rfloor = 2$ lines, i.e., Line 1 and Line 2 in Fig. 2(a). Except for the black and brown cards that have a priority value of 1, meaning they have the lowest priority, the other cards have a priority value of 10, which is the highest priority.

2.2. Parameters and Variables Definition

Table 1 shows the notations used as parameters and decision variables. Let n denote the number of picture cards to be arranged in m display sets. Each display set is an 2D UI frame with given width W , height H , and a maximum number of display sets m . Every card has its own width w_i and height h_i as well as priority value p_i , where $i \in \{1, 2, \dots, n\}$. Note that the card with the smaller value for its priority has the lower priority. For the experiment in Section 3, we use three types of priority, which are high, medium, and low. We assign the three priority types in a uniform random fashion to a given set of picture cards. Adding them increases the complexity in finding feasible solution and hence prolong the running time. Section 2.5 discusses in detail the impact of adding the priority constraint to the processing time and the number of display sets.

Following Liu *et al.* [10], we define six decision variables: (a) z_k is a binary variable set to 1 if a display set k contains at least one card, (b) s_{ik} is a binary variable set to 1 if card i is allocated to display set k , (c) l_{ij} and u_{ij} are binary variables set to 1 if card i is located to the left or under card j , respectively, for $i \neq j$, and (d) x_i and y_i are positive integer variables, including 0 and denoted by $Z_{(i \geq 0)}$, that indicates the location of card i , which its bottom and left corner, respectively.

Table 1. Summary of Notations

Notation	Description
n	The number of picture cards.
m	The maximum number of display sets.
W (H)	The width (height) of 2D UI Frame of each display set.
w_i (h_i)	The width (height) of picture card i .
p_i	The priority value of picture card i .
z_k	The binary variable set to 1 (0) if a display set k contains (does not contain) a minimum one card.
s_{ik}	The binary variable set to 1 (0) if card i is allocated (is not allocated) to display set k .
l_{ij}	The binary variable set to 1 (0) if card i is located (is not located) to the left of card j .
u_{ij}	The binary variable set to 1 (0) if card i is located (is not located) under card j .
x_i (y_i)	The integer variable indicating the location of card i which is its bottom (left) corner
$Z_{(i \geq 0)}$	A set of positive integers, i.e., $\{0, 1, 2, \dots\}$.
o	the minimum number of display sets containing at least one card.

2.3. Mathematical Modeling

Integer Linear Programming (ILP) in Formula 1 presents the mathematical model of IVLP that aims to minimize the number of display sets, formulated in 1a. The objective in formula 1a is subject to a set of constraints in formula 1(b) to formula 1(p).

Constraint (1b) guarantees that each display set is marked as used if it contains at least one picture card, while Constraint (1c) ensures that each card is allocated into exactly one display set. In constraints (1d) and (1e), every card must lie inside its bin. Then, constraints (1f), (1g) and (1h) ensure that all cards in the same display set do not overlap. Next, constraints (1i) and (1j) aid in lessening the symmetry issue, where all display sets with cards must have their index value in order. In other words, the three constraints make sure that the current display set with index k can be filled with the remaining picture cards if all the lower-index (previous) display sets, i.e., display set 1 to $(k - 1)$ have been filled with cards. Lastly, the lower bound o , which is the minimum number of display sets with cards, is applied by constraint (1k). Additionally, Constraint (1l) guarantees cards with higher priority are placed in the earlier display sets. Therefore, the total priority values for the cards allocated in a display set k must be smaller than or equal to those in the previous display set $k - 1$, for the priority value of the higher priority should be as large as possible than the priority value of the lower priority. Finally, constraints (1m) to (1p) define the data types of every decision variable.

$$\begin{aligned}
& \min \sum_{k=1}^m z_k && \text{1a} \\
& \text{s.t.} \\
& \sum_{i=1}^n s_{ik} \leq n z_k, && \text{1b} \\
& \sum_{k=1}^m s_{ik} = 1, && \text{1c} \\
& x_i + w_i \leq W, && \text{1d} \\
& y_i + h_i \leq H, && \text{1e} \\
& x_i + w_i \leq x_j + W(1 - l_{ij}) \quad i \neq j, && \text{1f} \\
& y_i + h_i \leq y_j + H(1 - u_{ij}) \quad i \neq j, && \text{1g} \\
& s_{ik} + s_{jk} - 1 \leq l_{ij} + l_{ji} + u_{ij} + u_{ji} \quad i < j, && \text{1h} \\
& z_k \leq z_{k-1} \quad k > 1, && \text{1i} \\
& \sum_{k=1}^i s_{ik} = 1 \quad i < m, && \text{1j} \\
& \sum_{k=1}^m z_k \geq 0, && \text{1k} \\
& \sum_{i=1}^n p_i s_{ik-1} \geq \sum_{i=1}^n p_i s_{ik} \quad k > 1, && \text{1l} \\
& z_k \in \{0,1\}, && \text{1m} \\
& s_{ik} \in \{0,1\}, && \text{1n} \\
& l_{ij}, u_{ij} \in \{0,1\} \quad i < j, && \text{1o} \\
& x_i, y_i \in Z_{(i \geq 0)}. && \text{1p}
\end{aligned}$$

2.4. Complexity Impact of Priority Constraints

Solving ILP like the formula (1) is an NP hard in general [23]. The complexity of solving ILP model is determined by the number of decision variables and number of constraints, with their number depends on the number of picture cards n and the maximum number of display sets m .

ILP has six types of decision variables: (a) m binary variables of z_k , (b) $m \cdot n$ binary variables of s_{ik} , (c) n^2 binary variables of l_{ij} , (d) n^2 binary variables of u_{ij} , (e) n positive integer variables x_i , and (f) n positive integer variables y_i . Overall, ILP in (1) has $(m(1+n) + 2n(n+1))$ decision variables, where the number of decision variables referring to picture cards dominates the complexity.

Constraints (1b), (1i), (1k), (1l) and (1m) of ILP in (1) are repeated for each display set k , where $\forall k \in \{1, 2, \dots, m\}$, while constraints (1c) to (1e), (1j), and (1p) are for each picture card $\forall i \in \{1, 2, \dots, n\}$. Next, constraints (1f), (1g), (1j), and (1o) apply to all pairs of items (i, j) with $\forall i, j \in \{1, 2, \dots, n\}$. Finally, constraint (1h) exists for each bin k and each pair of items (i, j) , where $\forall k \in \{1, 2, \dots, m\}$ and $\forall i, j \in \{1, 2, \dots, n\}$ and constraint (1n) is for all display sets m and cards n . Therefore, the total number of constraints for ILP (1) is $(5m + 5n + 4n^2 + mn^2 + mn)$.

The exponential nature of ILP comes from the combination of possible values for each decision variable for the overall number of variables and constraints. This causes a combinatorial explosion for ILP and hence results in NP-hard complexity. To sum up, the ILP has $O(n^2(1+m))$ binary variables and constraints; solving time grows exponentially and becomes intractable beyond ≈ 20 cards on our hardware.

If we omit constraint (1l) from formula (1), then the formula becomes an ILP model for 2DBPP. The introduction of priority constraint into the 2DBPP significantly impacts its complexity, generally making an already NP-hard problem even harder because it increases the problem difficulty. Adding priority rules expands the search space and requires more complex feasibility checks. Here, the evaluation of a potential arrangement of the items in each bin does not consider only the geometric overlapping of the items and the bin capacity but also ensures all priority rules are fulfilled. Furthermore, the priority constraint creates dependencies among

items that might otherwise be independent. For example, if an item i with higher priority must be laid before the item j with lower priority, this influences the available space for item j and potentially for other items. This cascading effect makes local decisions harder to make without sacrificing the overall solution.

2.5. Solutions

In general, a solution for an optimization problem can be approached using an exact method and an approximate method. Solving a large number of instances using an exact method is time-consuming and does not provide certainty of a sufficiently good convergence to a global optimum [9]. Many prefer approximate method [24], in particular heuristic methods [25][26][27] to approach a near-optimal solution. In this paper, we propose a greedy-based heuristic and swarm-based optimization to provide approximate solutions for a large number of picture cards. We provide details of the two proposed approaches as follows:

a. Greedy-based Heuristic Solution

We slightly modified the greedy algorithm proposed in [7], the Best-Fit-IVL, so that it can consider the priority of picture cards when allocating them into display sets. We call the modified version of the algorithm **Best-Fit-IVLP (BFI)**. Appendix A provides the pseudocode of BFI.

In step 1, all picture cards are sorted in decreasing order by both their priority and width. Here, cards with the same priority value are ordered by descending width to minimize line gaps. Therefore, the card with the highest priority and the widest size will be selected as the first item to put in the display frame. Starting from the top line section of the UI frame, step 3a of BFI allocates picture cards in descending order of their priority value and width. For the next card, BFI considers the following cases and takes different actions accordingly when a current card has a width larger than the remaining width space for the current line section:

- i. Step 3b obtains *the best-fit* card j in collection C with its width size taking up most of the remaining width space of line l .
- ii. Step 3b finds out that there is no card j in collection C that fits the remaining width space. Consequently, step 3c put the current card i in the next line of the current display set, if possible. If it is not possible because the current line is the last line of the display, then step 3d allocates card i in the new display set.

We now illustrate how BFI works using a set of ten cards (see Fig. 2b) and display frame with size (340×430) discussed in Section 2.1. Note that the black and brown cards have the same priority value of 1, meaning they have the lowest priority, and the other cards, i.e., yellow, green, blue, red, purple, pink, gray, and orange, have a priority value of 10, which is the highest priority. BFI sorts the cards in descending order of their priority and width, resulting the card sequence of yellow, green, blue, red, purple, pink, gray, orange, brown and black. BFI then put those cards one by one starting from the first line of the frame and continue to the next line until all cards are in their respective display set. Fig. 3(a) shows the results of arranging the cards into three display sets using the BFI strategy. Notice that the first line of display set 1 and the second line of display set 2 have a gap, meaning an empty space. BFI cannot fill the gap because there is no card that fit the gap.

Let consider swapping the priority value of the green card with that of the black card; black has higher priority than green. The card sequence changes to yellow, black, blue, red, purple, pink, gray, orange, brown, and green. In this case, BFI can arrange those cards into only two display sets. This second illustration shows the impact of priority on the number of display sets.

The time complexity of BFI is $(n \log n + n^2)$ times or $O(n^2)$. The sorting complexity using an $O(n \log n)$ sorting algorithm is less than the complexity of finding the best-fit card for both case (i) and case (ii) that, overall, during the course of allocating card, can take $O(n^2)$.

b. Swarm-based Optimization

IVLP arranges a collection of picture cards into a minimum number of display sets. To achieve a minimum number of display sets while ensuring that cards with higher priority must be laid in earlier display sets, we need to use as much as possible the space of the UI frame of each display set. In this case, a selection of the first card followed by the selection of the second card, and so on, in such a way that the former display set contains more prioritized cards than the latter display set and the frame area of all display sets is used as much as possible. Notice that this approach can be applied using one of the swarm-based methods, which is **Ant Colony Optimization (ACO)** [28][29].

ACO is commonly used for solving combinatorial optimization problems like the Traveling Salesperson Problem (TSP) and network routing [28][29]. It follows ants' behavior, for which most of the communication among individuals, or between individuals and the environment, is based on the use of chemicals called pheromones produced by the ants. As more ants pass through the shorter path, the pheromone concentrations also increase. Due to evaporation, the pheromone concentration in the longer path reduces, decreasing the probability of selection of this path in further stages. The whole colony gradually uses the shorter path in higher probabilities. So, path optimization is attained.

Appendix B provides the adaptation of ACO to solve ILP. The shortest possible path that ACO can find, with respect to IVLP, is a sequence of cards generated from a random selection of cards starting from the largest to the lowest priority. If there is a set of cards with the same priority, then ACO randomly selects one of them. When those cards are placed within the first display frame, then into the second display frame, and so on, the number of the displays containing the cards must be as minimal as possible.

Given a collection of cards C , step 1 sets the initial pheromone $\tau(i, j)$ of all pairs of cards i and j to 1. In the second step, each ant generates the solution, which is the allocation of all cards in C . In steps 2a to 2c, each ant randomly selects card i with the highest priority value and allocates the card in the frame. We use this strategy to speed up ACO in obtaining an optimal solution. Next, steps 2d to 2h put each selected card j in the frame, line per line, and move to the next display set $(k + 1)$ if the remaining width space of the last line is smaller than the card width size.

In step 2d, every unallocated card has a weighted probability to be selected for the next card based on its pheromone and attractiveness values when pairing with the previous allocated card i . Formula (2) provides the calculation of $p_k(i, j)$. Heuristic information or attractiveness $\eta(i, j)$ assists in selecting j as the next card after i . We use each card's priority value for the attractiveness. Parameter α and β control the importance of the pheromone and attractiveness to the probability of the next card selection. Set $N_k(i)$ contains all non-allocated cards. As shown clearly in Formula (2), cards with the higher priority value have a higher probability of being selected. Step 2d randomly selects one of the unallocated cards with the same highest probability. In this way, the step ensures earlier views of cards with the higher priority.

$$p_k(i, j) = \begin{cases} \frac{\tau(i, j)^\alpha \eta(i, j)^\beta}{\sum_{j \in N_k(i)} \tau(i, j)^\alpha \eta(i, j)^\beta} & j \in N_a(i) \\ 0 & j \notin N_a(i) \end{cases} \quad (2)$$

After all ants have their own solution, step 3 updates pheromones using formula (2). First, the pheromone value of each pair of cards is decreased by an evaporation level of ρ . The pheromone of each pair of cards can cumulatively increase for every ant that includes the pair. Next, step 4 notes the best solution from all ants that does not violate Constraint (1d) and has the smallest fitness value, which is the smallest number of display sets. Then, steps 2 to 4 are repeated for a maximum iteration of MAX_ITER. We use the maximum iteration for the termination criteria to avoid the algorithm falling into a local optimum, i.e., there is no improvement in its best solution for a certain number of consecutive iterations before finding a significantly better solution later [30].

$$\tau(i, j) = \rho \cdot \tau(i, j) + \sum_{k=1}^K \Delta\tau_k(i, j) \quad (3)$$

$$\Delta\tau_k(i, j) = \begin{cases} \frac{1}{L_k} & (i, j) \text{ included in the solution of ant } k \\ 0 & (i, j) \text{ not included in the solution of ant } k \end{cases}$$

To illustrate how ACO arranges a given set of cards, we refer back to the ten cards used for IVLP illustration in Section 2.1. For simplicity, we use only two ants, $\alpha = \beta = 1$, and a maximum of one iteration to show how ACO works.

For the first ant, ACO starts by randomly selecting one card from the eight cards with the highest priority, i.e., yellow, green, blue, red, purple, pink, gray, and orange. Let green be the selected card. ACO then calculates the probability value $p_k(i, j)$ for the other nine cards using formula (2). Here, each of the seven cards with the highest priority, i.e., yellow, blue, red, purple, pink, gray, and orange, has a probability value of $\frac{1^1 \times 10^1}{(7 \text{ cards} \times 10 + 2 \text{ cards} \times 1)} = 0.139$, while the probability of brown card and black card is 0.0139. ACO then randomly select the next card from the seven cards with higher probability. Note that ACO uses uniform distribution when randomly selecting one card among the other cards with the same priority. Meaning each of the cards has the same chance to be selected. For the second card, let assume ACO chooses the orange card. The selections of the third card until the last card follow the same steps. Fig. 3(c) shows the final card arrangement for the first ant that requires two display sets.

Let yellow card be the first selected by ACO for its second ant. In this case, the second card can be one of the remaining seven cards, which are green, blue, red, purple, pink, gray, and orange, having the highest probability of 0.139. Let assume the sequence of the selected cards by ACO for the second ant is yellow, green, blue, red, purple, pink, gray, orange, brown and black. In this case, the card arrangement produced by the second ant is exactly the same as shown in Fig. 3(a), which results in three display sets. Therefore, ACO decides the first ant as the best ant.

In ACO, step 1 requires $O(n^2)$ to assign initial pheromone value. In each iteration, step 2 in total takes $O(Kn^2)$ because its sub steps 2a, 2c, 2d and 2f need $O(n^2)$, while other substeps are constant. Next, step 3 and step 4, respectively, have a time complexity of $O(n^2)$ to update the pheromone of each pair of the cards and $O(K)$ to define the best card arrangement solution that provides minimum number of display sets. Note that steps 2 to 4 are repeated in MAX_ITER. Therefore, the overall time complexity of ACO is $O(K \cdot \text{MAX_ITER} \cdot n^2)$. The time complexity indicates that as the value of n , K and MAX_ITER increases, so does the time required to obtain an optimum result from ACO.

3. RESULTS AND DISCUSSION

We solve the ILP and use its optimal solutions as the benchmark to evaluate the performance of the two proposed approximate-based algorithms. The performance measurement involves the number of display sets and running time of ILP, BFI, and ACO. We have implemented ILP, BFI, and ACO in Python and used Gurobi for Python [31] to solve ILP in Formula (1). All experiments were conducted on a 64-bit Apple M1 machine with 8 processor cores and 16 GB of memory.

Table 2 provides the summary of all experimental results that consist of the frame size, the number of images, the number of display sets, running time and space utilization rate produced by each algorithm. We calculate the space utilization rate by totaling the area used to place all images in all display sets over the total area of all display sets. As shown in the table, there are six experimental scenarios, which are defined by the number of images and the size of the UI frame. The first scenario used images in Fig. 1. The other scenarios used 13, 15, 20, 40, 80, and 160 images in the scalable vector graphic (SVG) format taken from VICARA apps [32]. Each image is turned into a picture card by allocating it in a frame with a fixed height of 145 pixels and a white space (gap between two cards) of 23 pixels.

Table 2. Summary of Experimental Results

Size of UI Frame ($W \times H$)	Number of Images	Number of Display Sets				Running Time (second)			Space Utilization Rate		
		ILP	BFI	ACO		ILP	BFI	ACO	ILP	BFI	ACO
				min	max						
(340 × 430)	10	2	3	2	3	0.03	0.00000744	0.396	0.889	0.593	0.889
(300 × 500)	13	3	3	3	4	6.01	0.0000352	0.55	0.654	0.654	0.654
(300 × 500)	15	4	4	4	5	9555.33	0.0000230	1.355	0.578	0.578	0.578
(375 × 667)*	20	N/A	4	4	4	>24h	0.0000236	1.966	N/A	0.664	0.664
(412 × 915)**	40	N/A	5	5	6	>24h	0.0000622	11.051	N/A	0.683	0.683
(412 × 914)***	80	N/A	10	11	11	>24h	0.0000777	33.273	N/A	0.709	0.644
(768 × 1024)****	160	N/A	8	8	9	>24h	0.000441	117.93	N/A	0.8	0.8
* iPhone 12 Pro	** Pixel 7	*** Samsung Galaxy A51/71				**** iPad Mini					

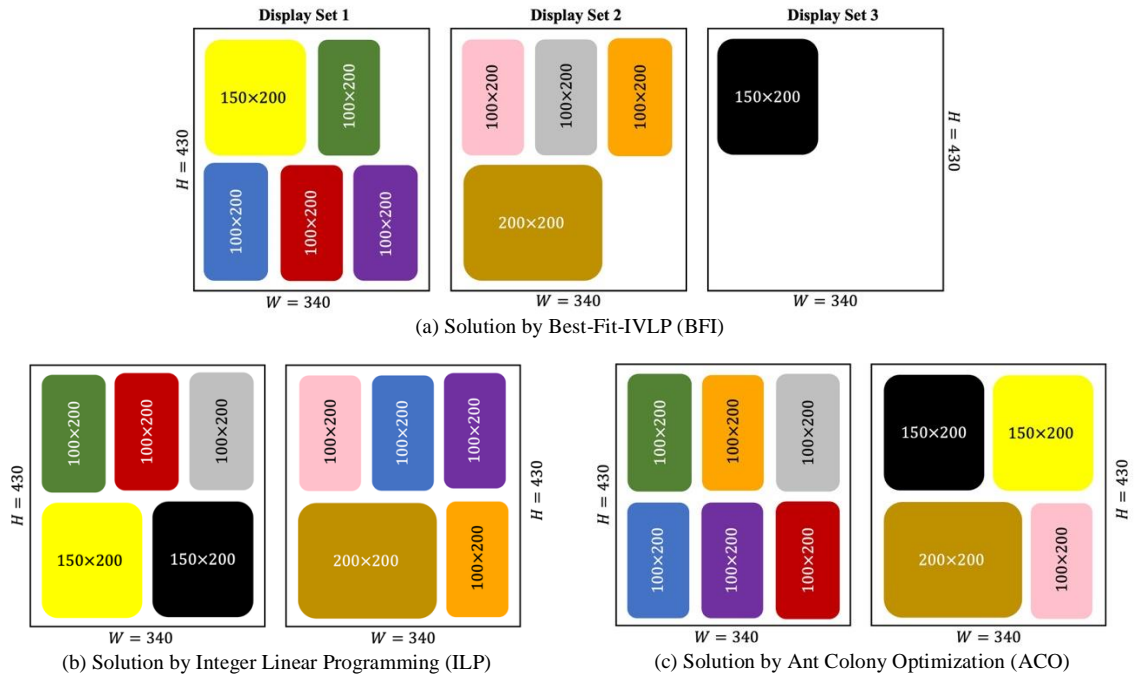


Figure 3. Experiment Result Using 10 Images Arranged in UI Frame with Size (340 × 430)

Since ACO involves random arrangement of the picture cards, we rerun ACO ten times for each scenario and note the lowest and the highest number of display sets in columns min and max. We have performed an experiment in using different values for the parameters of ACO. On average, we found that the values for parameters $\alpha = \beta = 1$, $\rho = 0.8$, MAX_ITER = 300 iterations allow ACO to have the best probability to produce the minimum number of display sets in ten runtimes. For the number of ants, we found that using $N = 20$ ants for the experiment that uses more than 20 images is sufficient. Using more than 20 ants only increases the running time and does not improve the result. For experiments with a smaller number of images, we used a smaller number of ants; 5 ants for 10 and 13 images and 10 ants for 15 and 20 images. Utilizing a smaller number of ants results in the same number of displays as if we used 20 ants.

We defined UI frame of size (340 × 430), (300 × 500), (375 × 667), (412 × 915), (412 × 914), and (768 × 1024) for which the last four are screen sizes of iPhone 12 Pro, Pixel 7, Samsung Galaxy A51/71, and iPad Mini, respectively. Note that the height size of each frame is only 67% of the whole screen size of each device. We cut the part of the screen size by 220 pixels to make space to put some UI components, such as UI action icons to view the previous and the next display set. Moreover, we used a smaller number of images for smaller screen sizes. We used three types of priority, the highest priority with value of 300, the medium priority with value of 80, and the lowest priority of 1 in value.

For the first experiment, we used the ten picture cards illustrated in Fig. 2 for a screen size of (340 × 430). The results in Table 2 shows that the optimum total display sets produced by ILP, which is 2 displays, BFI requires 3 displays, and ACO in 10 runs results in a 10% chance to require a minimum of 2 displays and a 90% chance to produce a maximum of 3 displays. Fig. 3 provides the illustration of the solutions produced by ILP, BFI and ACO using its minimum number of displays. Notice that the first display set produced by BFI and ACO consistently contains picture cards with the highest priority value, while ILP includes one card with the lowest priority value. Due to BFI performing allocation of cards that have been sorted in decreasing order of the priority and width of the picture cards, BFI does not explore other cards that may provide better arrangement such that it results in 2 instead of 3 display sets. Since BFI uses one more display set, its space utilization rate is only 0.593, which is smaller than ILP and ACO, which can reach 0.889. Note that a higher space utility rate is achievable by packing images in the smallest number of display sets possible.

For the second and third experiments, we took 13 and 15 out of the 160 SVGs and used a UI frame with size (300 × 500) pixels. As shown in Table 2, ACO has a 90% and 10% chance to use one more display set than ILP and BFI for 13 and 15 images, respectively. Using the minimum number of display sets for arranging 13 and 15 images, ACO utilizes up to 65.4% and 57.8%, respectively, of the space of all used displays as ILP and BFI.

In the fourth to seventh experiments, we fail to obtain any solution from ILP after running it for 24 hours. For indication of this failure, we put not available notation, i.e., N/A, for the number of display sets and space

utilization rate. In addition, we use symbol “>24h” to denote the overall time of our attempts to find the optimal solution from ILP. In this case, we only evaluate the results of BFI and ACO for the last four experiments.

Similar to the two previous experiments, we took 20, 40, and 80 out of the 160 SVGs to be arranged in a frame with a size of (375×667) , (412×915) , (412×914) , and (768×1024) , respectively. As shown in Table 2, ACO requires one more display set for arranging 80 images than BFI, resulting in its space usage being 9% smaller than BFI. However, ACO has a probability of 100%, 60%, and 40% in 10 runtimes to produce the same number of display sets as BFI when arranging 20, 40, and 160 images, respectively.

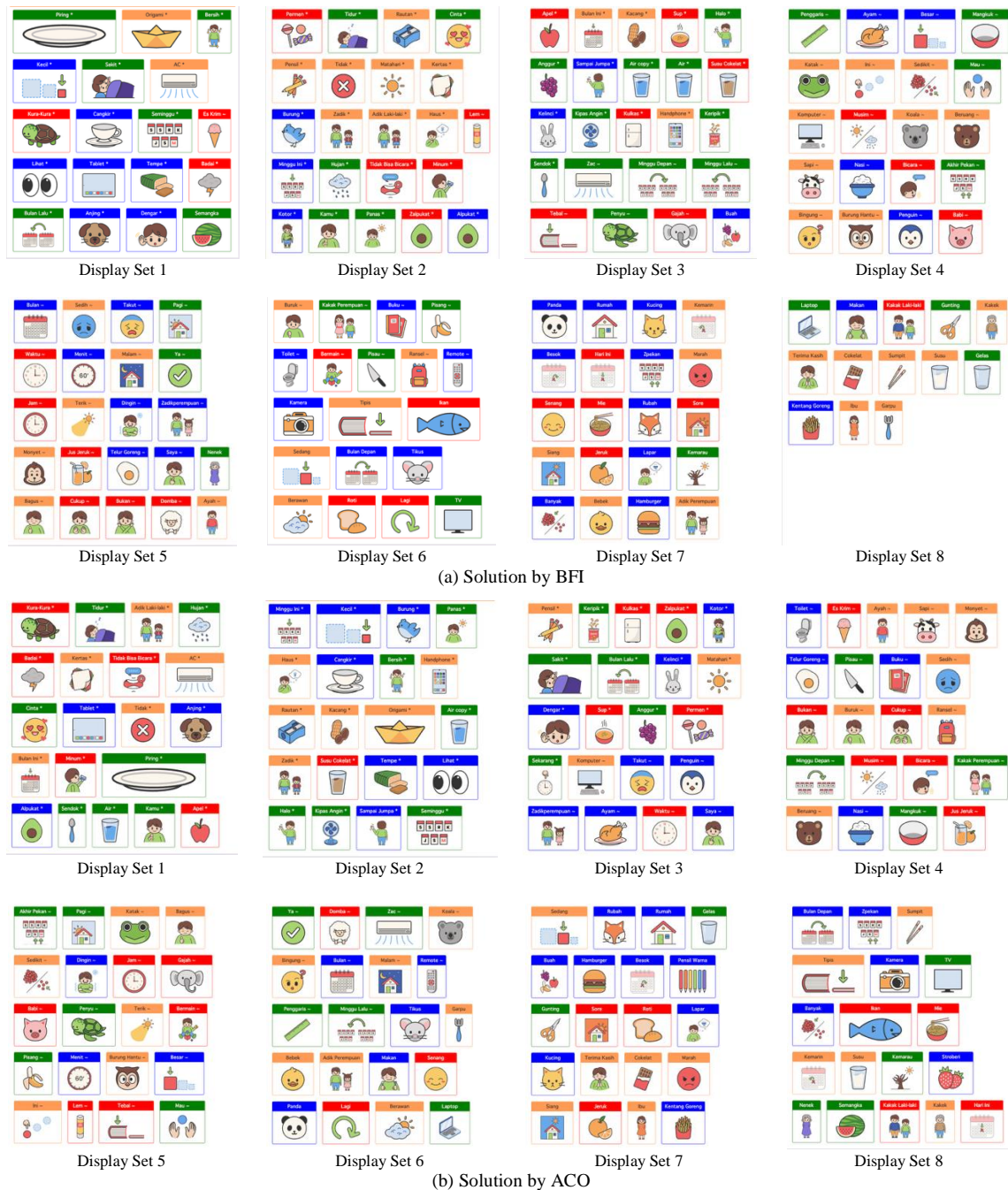


Figure 4. Experiment Result Using 160 Images Arranged in UI Frame with Size (768×1024)

Fig. 4 shows the arrangement of 160 cards produced by BFI and ACO in a screen size of an iPad mini. As can be seen from the figure, BFI and ACO produce different card arrangements, but they are able to allocate all prioritized cards in earlier display sets. Notice that BFI utilizes more of the line space of the first seven display sets than ACO, which causes its last display set to have fewer cards. In contrast, ACO evenly distributes the cards, with a few frame lines of display sets 4 and 8 having empty spaces that can be filled with other cards

allocated in the later display and frame line within the display set, respectively. Overall, BFI and ACO require the same space utilization rate, which is 0.8, for their 8 display sets.

Referring to the algorithm in section 2.4.b, ACO works by randomly selecting picture cards based on the order of their higher priority and putting them in every line of the frame one by one. In this way, ACO can have different arrangements for each run. ACO moves to the next line or the first line of the next display set if the remaining width space of the line is smaller than the width size of the next selected card. In BFI, as shown by its algorithm in section 2.4.a, if the remaining width space in a line of a frame is less than the width of the selected card, BFI will look for another unallocated card that fits into the remaining width space. Consequently, it is possible that, in some cases, BFI requires a smaller number of display sets than ACO. Note that BFI does not explore other arrangements for the cards that may result in a smaller number of display sets. Therefore, BFI may not always provide the optimal solution for a larger number of images.

The average running time for all experiments in Table 2 shows that BFI provides a significantly faster time than ILP and ACO, with ILP taking a significantly longer time. Referring to section 2.4, ILP combines the possible values for $O(n^2 + n^2m)$ binary variables and constraints to get the optimum result. For that reason, solving time grows exponentially and becomes intractable beyond ≈ 20 cards. In this case, for viewing a larger number of images, BFI is more practical than ILP.

However, if viewing images requires dynamicity, ACO is preferable because it can provide other possible arrangements for viewing the cards on the same number of display sets. Note that different arrangements of cards offered by ACO offer several benefits for real cases such as e-commerce and a photographer's portfolio. They include showcasing a variety of subjects and styles, creating visual interest, and allowing consumers or viewers to connect with different aspects of the product collection or the photographer's work.

4. CONCLUSION

The optimization problem of image view layout with priority (IVLP) focuses on arranging a set of picture cards, with the same height but different widths, in a minimum number of display sets. A display set is a 2D frame that holds a collection of non-overlapping picture cards. Moreover, IVLP considers each card to have a different priority. Our study aims to optimize the display space by packing as many cards as possible into each frame while utilizing the fewest number of display sets. Since IVLP considers picture cards of the same height, we view it as a special case of 2DBPP. We use Integer Linear Programming (ILP) to model IVLP and propose one greedy-based heuristic algorithm called Best-Fit-IVLP (BFI) and a swarm optimization algorithm called Ant Colony Optimization (ACO).

Using different number of SVG based images put in a card frame with a fixed height but varying width, our experimental results show that BFI and ACO can generate close to optimal solutions produced by ILP. We found that BFI is more practical to use because it has a running time significantly faster than ACO and ILP. For 160 images, BFI runs in less than one second, i.e., 0.00044 seconds, while ACO takes 117.93 seconds. Overall, BFI and ACO can provide an optimal number of display sets to pack a given number of images with space utility rates ranging from 0.578 to 0.889. However, BFI always provides the same picture card arrangement. ACO, on the other hand, can provide different arrangements for viewing the picture cards but with the same number of display sets and space utilization rate.

BFI has a weakness in providing different arrangements of picture cards in each display set, while ACO in some cases requires one more display set than BFI. For future work, we are planning to combine BFI and ACO to get the strength of both algorithms and, hence, remove their weaknesses. The idea is to apply the BFI strategy in ACO when selecting the last card that occupies most of the remaining width space for each frame line and does not violate the priority constraint. Furthermore, we consider relaxing our mathematical model, i.e., ILP, and combining it with BFI, ACO, or their combination to improve their scalability in finding the optimal solution for a larger number of images. In addition, we would like to explore other approximate methods, such as machine learning, to improve the solution and running time. Finally, we plan to extend the IVLP problem not only for use in augmentative and alternative communication apps for people with autism and communication difficulties but also for common applications that require arranging images of any size neatly in a minimum number of displays.

ACKNOWLEDGEMENT

The authors would like to thank Oxana Agatha Guijaya, S.Sn. and Ruby Chrissandy, S.Sn., M.Ds. for their picture cards used as the experimental dataset in this paper.

REFERENCES

- [1] M. Rezae, N. Chen, D. McMeekin, T. Tan, A. Krishna, and H. Lee, 'The evaluation of a mobile user interface for people on the autism spectrum: An eye movement study', *Int J Hum Comput Stud*, vol. 142, p. 102462, Oct. 2020, doi: 10.1016/j.ijhcs.2020.102462.
- [2] M. Alzahrani, A. L. Uitdenbogerd, and M. Spichkova, 'Human-Computer Interaction: Influences on Autistic Users', *Procedia Comput Sci*, vol. 192, pp. 4691–4700, 2021, doi: 10.1016/j.procs.2021.09.247.
- [3] Cboard, 'Cboard: Communication for Everyone'. Accessed: Mar. 19, 2024. [Online]. Available: <https://www.cboard.io/>
- [4] Dream Oriented, 'Leeloo AAC - Austim Speech App'. Accessed: Jul. 01, 2023. [Online]. Available: <https://play.google.com/store/apps/details?id=org.dreamoriented.leeloo&hl=en-ID&pli=1>
- [5] H. Hersinta, C. R. A. Bangun, and O. D. Hutagaol, 'Developing VICARA 2.0: Exploring the potential use of augmentative and alternative communication (AAC) apps for the parents and teachers of autistic students', in *AIP Conference Proceedings*, AIP Publishing., 2023, doi: 10.1063/5.0127048.
- [6] 'Vicara 2', Google Play. Accessed: Jul. 02, 2025. [Online]. Available: <https://play.google.com/store/apps/details?id=com.vicara.vicara2>
- [7] L. Hiryanto, A. Putra Wirawan, and Tony, 'Greedy Approach for Optimizing Image View Layout on Various Sizes of 2D UI Container', in *2024 International Conference on Electrical Engineering and Computer Science (ICECOS)*, IEEE, Sep. 2024, pp. 298–303, doi: 10.1109/ICECOS63900.2024.10791081.
- [8] M. Iori, V. L. de Lima, S. Martello, F. K. Miyazawa, and M. Monaci, 'Exact solution techniques for two-dimensional cutting and packing', *Eur J Oper Res*, vol. 289, no. 2, pp. 399–415, Mar. 2021, doi: 10.1016/j.ejor.2020.06.050.
- [9] S. Polyakovskiy and R. M'Hallah, 'Just-in-time two-dimensional bin packing', *Omega (Westport)*, vol. 102, p. 102311, Jul. 2021, doi: 10.1016/j.omega.2020.102311.
- [10] C. Liu, K. Smith-Miles, T. Wauters, and A. M. Costa, 'Instance space analysis for 2D bin packing mathematical models', *Eur J Oper Res*, vol. 315, no. 2, pp. 484–498, Jun. 2024, doi: 10.1016/j.ejor.2023.12.008.
- [11] A. M. Chwatal and S. Pirkwieser, 'Solving the Two-Dimensional Bin-Packing Problem with Variable Bin Sizes by Greedy Randomized Adaptive Search Procedures and Variable Neighborhood Search', 2012, pp. 456–463, doi: 10.1007/978-3-642-27549-4_58.
- [12] H. Zhang, Q. Liu, L. Wei, J. Zeng, J. Leng, and D. Yan, 'An iteratively doubling local search for the two-dimensional irregular bin packing problem with limited rotations', *Comput Oper Res*, vol. 137, p. 105550, Jan. 2022, doi: 10.1016/j.cor.2021.105550.
- [13] Y. Yuan, K. Tole, F. Ni, K. He, Z. Xiong, and J. Liu, 'Adaptive simulated annealing with greedy search for the circle bin packing problem', *Comput Oper Res*, vol. 144, p. 105826, Aug. 2022, doi: 10.1016/j.cor.2022.105826.
- [14] S. Kosari, M. Hosseini Shirvani, N. Khaledian, and D. Javaheri, 'A Hybrid Discrete Grey Wolf Optimization Algorithm Imbalance-ness Aware for Solving Two-dimensional Bin-packing Problems', *J Grid Comput*, vol. 22, no. 2, p. 49, Jun. 2024, doi: 10.1007/s10723-024-09761-7.
- [15] W. Chen, H. Yu, X. Li, L. Qu, and Z. Mi, 'Layout Design with a Firefly Algorithm for User Interfaces in Vehicle System', in *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, IEEE, Sep. 2020, pp. 110–113, doi: 10.1109/ICISCAE51034.2020.9236921.
- [16] X. Zhang, M. Shan, and J. Zeng, 'Parallel Batch Processing Machine Scheduling Under Two-Dimensional Bin-Packing Constraints', *IEEE Trans Reliab*, vol. 72, no. 3, pp. 1265–1275, Sep. 2023, doi: 10.1109/TR.2022.3201333.
- [17] K. Zhu, N. Ji, and X. D. Li, 'Hybrid Heuristic Algorithm Based On Improved Rules & Reinforcement Learning for 2D Strip Packing Problem', *IEEE Access*, vol. 8, pp. 226784–226796, 2020, doi: 10.1109/ACCESS.2020.3045905.
- [18] P. Duan, C. Wierzynski, and L. Nachman, 'Optimizing User Interface Layouts via Gradient Descent', in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA: ACM, Apr. 2020, pp. 1–12, doi: 10.1145/3313831.3376589.
- [19] M. Kaleta and T. Śliwiński, 'Neural-Driven Constructive Heuristic for 2D Robotic Bin Packing Problem', *Electronics (Basel)*, vol. 14, no. 10, p. 1956, May 2025, doi: 10.3390/electronics14101956.
- [20] M. Bertuccio and T. D. Sanger, 'A Model to Estimate the Optimal Layout for Assistive Communication Touchscreen Devices in Children With Dyskinetic Cerebral Palsy', *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 7, pp. 1371–1380, Jul. 2018, doi: 10.1109/TNSRE.2018.2840445.
- [21] K. Fukaya, D. Daylamani-Zad, and H. Agius, 'Intelligent Generation of Graphical Game Assets: A Conceptual Framework and Systematic Review of the State of the Art', *ACM Comput Surv*, vol. 57, no. 5, pp. 1–38, May 2025, doi: 10.1145/3708499.
- [22] Z. Fan, B. Ghaddar, X. Wang, L. Xing, Y. Zhang, and Z. Zhou, 'Artificial Intelligence for Operations Research: Revolutionizing the Operations Research Process', Jan. 2024.
- [23] D. Knop, M. Pilipczuk, and M. Wrochna, 'Tight Complexity Lower Bounds for Integer Linear Programming with Few Constraints', *ACM Transactions on Computation Theory*, vol. 12, no. 3, pp. 1–19, Sep. 2020, doi: 10.1145/3397484.
- [24] B. Guo *et al.*, 'Two-dimensional irregular packing problems: A review', *Front Mech Eng*, vol. 8, Aug. 2022, doi: 10.3389/fmech.2022.966691.
- [25] J. Zeng and X. Zhang, 'An Adaptive Large Neighborhood Search for Single-Machine Batch Processing Scheduling With 2-D Rectangular Bin-Packing Constraints', *IEEE Trans Reliab*, vol. 71, no. 1, pp. 139–148, Mar. 2022, doi: 10.1109/TR.2021.3128167.
- [26] H. Zhao, C. Zhu, X. Xu, H. Huang, and K. Xu, 'Learning practically feasible policies for online 3D bin packing', *Science China Information Sciences*, vol. 65, no. 1, p. 112105, Jan. 2022, doi: 10.1007/s11432-021-3348-6.
- [27] M. Witteman, Q. Deng, and B. F. Santos, 'A bin packing approach to solve the aircraft maintenance task allocation problem', *Eur J Oper Res*, vol. 294, no. 1, pp. 365–376, Oct. 2021, doi: 10.1016/j.ejor.2021.01.027.
- [28] M. Dorigo and T. Stützle, 'Ant Colony Optimization: Overview and Recent Advances', 2019, pp. 311–351, doi: 10.1007/978-3-319-91086-4_10.
- [29] R. Priyadarshi and R. R. Kumar, 'Evolution of Swarm Intelligence: A Systematic Review of Particle Swarm and Ant Colony Optimization Approaches in Modern Research', *Archives of Computational Methods in Engineering*, Mar. 2025, doi: 10.1007/s11831-025-10247-2.
- [30] G. Li, C. Liu, L. Wu, and W. Xiao, 'A mixing algorithm of ACO and ABC for solving path planning of mobile robot', *Appl Soft Comput*, vol. 148, p. 110868, Nov. 2023, doi: 10.1016/j.asoc.2023.110868.
- [31] Jerry Yurchisin, 'Getting Started with Mathematical Optimization in Python', [gurobi.com](https://www.gurobi.com/resources/mathematical-optimization-in-python-how-to-get-started/). Accessed: Jul. 02, 2025. [Online]. Available: <https://www.gurobi.com/resources/mathematical-optimization-in-python-how-to-get-started/>
- [32] 'Vicara 3', [play.google.com](https://play.google.com/store/apps/details?id=com.vicaraxebp.vicara3&hl=id). Accessed: Jul. 02, 2025. [Online]. Available: <https://play.google.com/store/apps/details?id=com.vicaraxebp.vicara3&hl=id>

APPENDIX A

Best-Fit-IVLP Algorithm

Best-Fit-IVLP (BFI)

Given C as the collection of picture cards, W and H as the size of the UI frame or display set, and number of lines $L = \lfloor W/h \rfloor$ where $l \in \{1, 2, \dots, L\}$

1. Sort cards in C in descending order of their priority value and width; cards with equal priority are sorted by descending width
 2. Start with display set $k = 1$ and line $l = 1$
 3. for card in C :
 - a. if line l still have space for card i , put card i in line l
 - b. else if a card j in C best fits the remaining space, put card j in line l
 - c. else if $(l+1 \leq L)$, put card i in the next line $l = l+1$
 - d. else put card i in line $l=1$ of the next display set $k = k+1$
 4. Halt
-

APPENDIX B




Ant Colony Optimization Algorithm

Ant Colony Optimization (ACO)




Given C as the collection of picture cards, W and H as the size of the UI frame or display set, number of lines $L = \lfloor W/h \rfloor$ where $l \in \{1, 2, \dots, L\}$, K is the number of ants, $\tau(i, j)$ is the pheromone level of laying card i and card j side by side in the same line, and maximum iteration MAX_ITER

1. Initialize pheromone level $\tau(i, j)$ for $\forall i, \forall j \in \{1, 2, \dots, n\}$ and $i \neq j$
 2. For each ant $a \in \{1, 2, \dots, K\}$:
 - a. Randomly select card i in C with the smallest priority value
 - b. Start with display set $k = 1$ and line $l = 1$
 - c. $C' = C - \text{card } i$
 - d. Randomly select card j in C' with highest probability $p_k(i, j)$ of formula (2)
 - e. if line l still have space for card j , put card j in line l
 else if $(l+1 \leq L)$, put card j in the next line $l = l+1$
 else put card j in line $l = 1$ of the next display set $k = k+1$
 - f. $C' = C' - \text{card } j$
 - g. $i = j$
 - h. Repeat steps d to g until C' is empty
 3. Update pheromone values for each pair of (i, j) using formula (3)
 4. best = ant $a \in \{1, 2, \dots, K\}$ that complies with constraint (11) and has the smallest number of display sets
 5. Repeat steps 2 to 3 for MAX_ITER
 6. Halt
-




BIOGRAPHIES OF AUTHORS

Lely Hiryanto, ST., M.Sc., Ph.D.    received her bachelor's degree in computer science from Tarumanagara University, Jakarta, Indonesia, in 2001. She received her postgraduate diploma, Master of Science, and PhD in Computer Science from Curtin University in 2015, 2016, and 2022, respectively. She is a senior lecturer in the Faculty of Information Technology, Tarumanagara University, Jakarta, Indonesia. Her research interests include decision optimization, data mining, and network optimization. He can be contacted at email: lelyh@fti.untar.ac.id



Andhika Putra Irawan    received his bachelor's degree in computer science from Tarumanagara University, Jakarta, Indonesia, in 2025. His research interests include decision optimization and networking. He can be contacted at email: andhika.535210010@stu.untar.ac.id



Dr. Viciano Lee, B.B.A., M.Sc., Ph.D.    is a Lecturer in Mathematics and Computer Science at City, University of London. He is a Fellow of the Higher Education Accreditation (HEA). In 2021, He received his PhD in Operations Research, with applications to the area of Search Games (Game Theory) and Social Choice Theory from The University of Warwick. Recently, He is also interested in pedagogic research, particularly in areas around abstraction development and computing education. Prior to joining City, He was a Teaching Fellow at Warwick University. He can be contacted at email: viciano.lee@city.ac.uk