Multi-objective optimization of layout with functional constraints

Xiaoxiao Song, Emilie Poirson, Yannick Ravaut, Fouad Bennis

Optimization and Engineering manuscript No.
(will be inserted by the editor)

Multi-objective optimization of layout with functional constraints

Xiaoxiao Song[1] · **Emilie Poirson**[1] · **Yannick Ravaut**[2] · **Fouad Bennis**[1]

Received: date / Accepted: date

Abstract The conventional layout problem is concerned with finding the arrangements of components inside the container to optimize objectives under geometrical constraints, i.e., no component overlap and no container protrusion. In this paper, the multi-objective optimization for layout balance and component activity requirements with functional constraints is developed. Integrating the accessibility of components as functional constraints ensures components maintenance or proper operation. However, addressing the functional constraints increase the complexity of the layout optimization. A novel multi-objective optimization algorithm is proposed using the constructive placement and the simulated annealing to search for compromised solutions between the two objectives. Thereafter, a similarity indicator is defined to evaluate how similar optimized layout designs are. The experiments indicate that the proposed optimization approach performs well in ensuring accessibility and efficiently finding high-qualified solutions, where the constructive placement largely contributes to the search for alternatives satisfying constraints.

Keywords Layout problem · Multi-objective optimization · Accessibility integration · Constructive placement · Similarity indicator

1 Introduction

The layout problems are inherently multidisciplinary tasks. The applications can be the space radiator design (Cuco et al., 2011), the chip layout design

✉ Xiaoxiao Song
E-mail: Xiaoxiao.song@ec-nantes.fr

[1] Laboratoire des sciences du numérique de Nantes (LS2N), ECN, UMR 6004, 44321 Nantes, France.

[2] Thales Communications, 49300 Cholet, France.

(Gao et al., 2019), the vehicle layout design (Fossati et al., 2019), the architecture layout design (Zawidzki and Szklarski, 2020) in Fig. 1(a), the manufacturing systems layout design (Niroomand et al., 2015) in Fig. 1(b) and so on. Excellent layout design can effectively improve the system performance. The

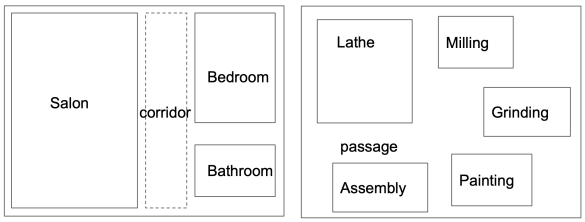


Fig. 1 Layout problem examples (a) Architecture layout, (b) Manufacturing system.

problem is generally solved as an optimization problem. It concerned with finding the optimal arrangements of components(i.e., equipment, machines) inside the container(i.e., workshop, plant) to optimize the objectives and respect geometrical and functional constraints. The most encountered components are represented by rectangles with determined sizes (Ou-Yang and Utamima, 2013) or determined area (Saraswat et al., 2015). No component overlap and no container protrusion are the common geometrical constraints, while orientation or alignment is to define functional relationships between components (Cagan et al., 2002). The functional constraints specify the requirements to ensure the system's proper functioning. A majority of studies optimize, for example the mass distribution related to mobile spacecraft layout (Cuco et al., 2014), or the adjacency requirement (Ghassemi Tari and Neghabi, 2015) and the material handling cost (Lin and Yingjie, 2019) in facility layouts.

The constraint satisfied region, the non-linear and non-convex objective of layout formulation make the optimization complex in nature. The constraints satisfied solutions can be obtained by penalizing the constraints violations in the objective function or generated from the feasible designs domains (Szykman and Cagan, 1997). The most commonly encountered layout problems have multiple objectives that need to be optimized. In fact, multi-objective problems can be solved by single-objective optimization or multi-objective optimization techniques. The former case transforms multiple objectives into an aggregation function using predefined weights, so there is a corresponding single solution. In the latter approach, a multi-objective optimizer considers multiple objectives simultaneously and aims to find a set of compromised solutions, known as the Pareto front. In this paper, multi-objective layout optimization is employed and the optimal solution could be subjectively decided by the designer after optimization.

Analyzing the similarities between optimized layout designs can reduce the work load of designer in the selection phase. Usually, similarity computa-

tion is the difference in layout designs. Layout i and layout j are geometrically different if any component is moved a certain distance from the layout configuration (Bénabès et al., 2011). Nevertheless, evaluating the similarities globally is more generic than individual component comparison. A similarity indicator defines how similar two layout designs are based on the element-wise difference in Song et al. (2021b). However, it can not differentiate the symmetrical configuration, and it becomes time consuming when the size of the layout area increases. The main contributions of this paper are the following:

1. It develops a multi-objective layout model with functional constraints using the new component representation. The proposed model optimizes the balance and activity requirements between components, and considers the accessibility of components from the container's entry.
2. An efficient multi-objective optimizer that integrates accessibility analysis within simulated annealing method is proposed to conduct the optimization in the feasible alternatives that respect all constraints.
3. The proposed algorithm allows efficient optimization of complex layout problems, enabling a truly interactive design process. A modified similarity indicator based on the relative position is proposed. The user could select the most favorable solution based on the hierarchical similarity visualization.

The paper is organized as follows. Section 2 introduces the overview of the related work. Section 3 formally defines the problem model. Then the proposed optimization framework is described in Section 4. Section 5 presents the results analysis. The conclusions and perspectives are discussed in Section 6.

2 Related work

2.1 Layout optimization

The layout problem defined by the representation of components, formulation of objectives, and evaluation of constraints maybe different but it is non-deterministic polynomial-time hard in general. Finding optimal layouts is beyond the reach of exact optimization techniques (Xie and Sahinidis, 2008). Stochastic algorithms have been proposed to provide sub-optimal solutions in reasonable computational time, such as extended pattern search designed for wind farm layout (Du Pont and Cagan, 2012), genetic algorithm applied for vehicle layout (Guarneri et al., 2013), particle swarm optimization of operating theater layout (Chraibi et al., 2016) and simulated annealing used for plans layout (Zheng and Ren, 2020). The optimal configuration can be obtained through design techniques. Cagan and Mitchell (1993) developed shape annealing by integrating shape grammars into simulated annealing. The shape grammar defines a transformation rule for an existing shape(component) and is applied incrementally to generate a set of configurations. It is worth noting that layout problems involve components placements and have a range of variability.

Current layout optimization techniques can be divided into mathematical models or heuristic methods. In the continuous formulation, the position and/or orientation of components are variables. Mixed integer programming approach takes the orientation as integer variable while a modified genetic algorithm with rotation operator can avoid the mixed integer form (Hasda et al., 2016). Various placement algorithms and encoding schemes are designed for efficient layout optimization (Tiwari et al., 2006). Miao et al. (2008) developed a packing genetic algorithm with customized operators and the encoding scheme for an army vehicle. Cagan et al. (1998) developed a simulated annealing algorithm using translation and rotation related movement for complex geometrical shapes layout problems. What's more, the layout problem can be formulated as a quadratic assignment problem and solved as a combinatorial problem. Simulated annealing demonstrated the robustness in combinatorial layout optimization (Baykasoğlu and Gindy, 2001; Singh and Sharma, 2008; Şahin, 2011; Moradi and Shadrokh, 2019).

The mathematical model can easily incorporate description of the problem and perfectly reflect reality, while using heuristic methods can reduce the search complexity. A flexible bay structure (Mazinani et al., 2012; Garcia-Hernandez et al., 2015) and a slicing tree (Shayan and Chittilappilly, 2004; Ripon et al., 2013; Kang and Chae, 2017) divide the container into horizontal and vertical directions, and the components are restricted to be located inside follow a specific logical order. Whereas the constructive heuristic generates a complete layout that others cannot represent by sequentially placing components anywhere inside the container (Hosseini nasab et al., 2018).

The placement approach applied with meta-heuristic proved to be an efficient algorithm for layout optimization (Halawa et al., 2021a). A genetic algorithm encodes the packing sequences and a placement algorithm determines the component placements for components packing in Li et al. (2014). Gonçalves and Resende (2015) proposed one hybrid approach to minimize the distances between components. A biased random-key genetic algorithm determines the order and a placement positions facilities, finally, a linear programming model is applied for local optimization. In Huo et al. (2021), an NSGAII algorithm encodes the placement sequence to optimize multi-objective layout problems, a local optimization algorithm based on simulated annealing is invoked if the similarity among individuals extends the threshold. In parallel, several layout optimization methods combined constructive approach and simulated annealing can be found in recent layout problems studies. Xiao et al. (2013) proposed a two-step heuristic algorithm using discrete modeling, first, the feasible layout is constructed using the placement order of simulated annealing. Then, local optimization is followed to improve the solution in each zone. Allahyari and Azab (2018) developed a simulated annealing with swap operator and displaced operator to optimize the solution generated by the placement. Simulated annealing has fewer parameters and the simpler structure compared to genetic algorithm optimizations.

2.2 Accessibility integration

In most real-world applications, the main constraints are the geometrical constraints between components. More importantly, the components are functionally connected. The accessibility is one particular layout functional constraint that expresses the maintainability, inspection as well as reachability to components. For instances, leave free space around the facility to ensure components function properly (Bénabès et al., 2010), keep enough whitespace in cell design to ensure pins reachability (Seo et al., 2017) and bundle multiple wires as a harness for easy inspection (Masoudi and Fadel, 2021). Accessibility is a functional constraint in layout design, but can be formulated as a geometrical expression. Considering the accessibility requirement, the designer can insert the expertise in problem descriptions or integrate the accessibility constraint in problem formulations. Michalek et al. (2002) applied the intersection constraint to force the room interaction to ensure access and RazaviAlavi and AbouRizk (2017) integrated minimum distance constraint for accessibility between facilities. More recently, Halawa et al. (2021b) formed identical rooms into one or two rows to ensure a corridor can access the rooms in pods design. The intuition-inspired way limits innovation of finding solutions. Grignon and Fadel (2004) implemented necessary access volumes as static components and maximized the number of intersecting components. Bénabès et al. (2010) characterized the accessibility as one distance-based objective in layout optimization. However, introducing the objective may increase the difficulty and computational cost. As far as we know, the accessibility analysis in layout optimizations has not been developed maturely yet.

The reviewed studies have greatly enriched the layout knowledge base and applications. However, most research work is tested using pre-defined problems and remains theoretical. Also, accessibility is not completely considered. And the optimization may generate many infeasible solutions, particularly in the dense layout problem. Besides, the existing constructive method is applied in parallel with meta-heuristic optimization, where the component placement is determined by estimating the objective iteratively. In other words, the objective function evaluation is based on the accumulating process, assuming that the objective function values between component are determined independently. However, the assumption is not true in reality.

3 Layout model

The problem is to locate the rectangular components in a rectangular container. Based on the different functional properties of components, the novel component, including the solid and virtual part is defined. To formulate the model, we define the following notations in Table 1.

Table 1 Model notations

c_i	Component i
s_i	Solid component i
v_{ij}	Virtual component j of s_i
n	Number of component
n_i	Number of virtual component attached to s_i
(x_i, y_i)	Coordinates of s_i
(w_i, h_i)	Size of s_i
$(x_{L_i}, y_{B_i}, x_{R_i}, y_{T_i})$	Left, Bottom, Right, Top side location of s_i
(x_{ij}, y_{ij})	Coordinates of v_{ij}
(w_{ij}, h_{ij})	Size of v_{ij}
β_s	Solid components density
β_v	Virtual components density
β_c	Components capacity
a	Available space
(x_a, y_a)	Coordinates of available space a
(w_a, h_a)	Size of available space a
$(x_{L_a}, y_{B_a}, x_{R_a}, y_{T_a})$	Left, Bottom, Right, Top side location of a
m_i	Mass of s_i
(x_{c_i}, y_{c_i})	Gravity center of s_i
(X_{gra}, Y_{gra})	Gravity center of all solid components
(X'_{gra}, Y'_{gra})	Geometry center of container
(W, H)	Size of container
ω_{ij}	Activity factor between c_i and c_j
d_{ij}	Distance from center of s_i to s_j

3.1 Component definition

The layout problem aims to find the optimal arrangement of components in a container. If the components are solid, meaning that the overlap is forbidden. However, in reality, there are virtual components without mass where the overlap among them is acceptable. For example, the space of the cabinet for door opening and closing in Fig. 2(a), the space of the drawer to pull it out in Fig. 2(b), and the space of desk to allow the user sit down as shown in Fig. 2(c) etc. Moreover, a collapsible object is designed to be folded flat when

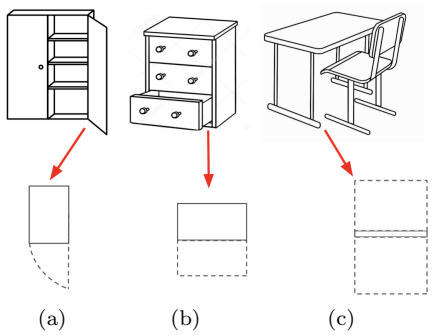


Fig. 2 Component examples (a) Cabinet, (b) Drawer, (c) Desk.

it is not being used, such as a collapsible chair or desk in the office. The collapsible component is stored easily and its foldable character saves more space. Besides, the foldable, expandable, retractable, inflatable, and stackable components also allow multi-task.

Based on the analysis above, the different components are summarized for the layout problem formulation:

1. Virtual components could overlap with virtual components and have no mass.
2. Solid components could not overlap with solid or virtual components and have mass. A solid component may become temporary solid if it is collapsible and can overlap with virtual and solid components.

Thus, the component $c_i = (s_i, v_{ij}), i = 1, ..., n, j = 1, ..., n_i$, can have the solid component s_i (solid rectangle) and the virtual component v_{ij} (dotted rectangle), as shown in Fig. 3. Each solid component s_i is represented by coordinates (x_i, y_i) and rectangle size (w_i, h_i). There are n_i accessibility spaces, namely virtual components v_{ij}, attached to s_i. The virtual component v_{ij} is defined by coordinates (x_{ij}, y_{ij}) and dotted rectangle size (w_{ij}, h_{ij}) in the local frame of s_i. The solid and virtual components can be denoted as $s_i = (x_i, y_i, w_i, h_i)$ and $v_{ij} = (x_{ij}, y_{ij}, w_{ij}, h_{ij})$, respectively.

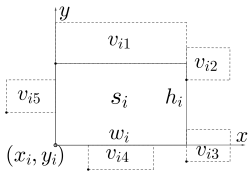
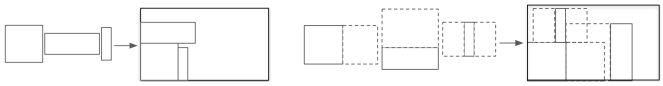


Fig. 3 Component c_i representation.

3.2 Capacity definition

Before applying an optimization algorithm, it is necessary to analyze the feasible complexity of the problem. The feasible complexity analysis aims to estimate the space capacity, which is the most desirable question in a layout design. For the layout description, the area occupied by components should be less than the container area. One example of three solid components packing is shown in Fig. 4(a). The density represents the container area occupied by the components. The density of solid components $\beta_s = \frac{\sum_{i=1}^{n}(w_i \times h_i)}{W \times H}$, and the density of virtual components $\beta_v = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n_i}(w_{ij} \times h_{ij})}{W \times H}$, W, H are the size of the container space. However, the overlap among virtual components is acceptable. The sum of the dimensions of the components will exaggerate the feasibility complexity and incorrectly indicates that the problem can not be



(a) (b)

Fig. 4 Component packing (a) Solid components, (b) Solid and virtual components.

solved (i.e., $\beta_s + \beta_v > 1$). Therefore, a capacity index β_c is defined to measure the minimum occupied space of a given number of solid and virtual components. Bénabès et al. (2012) estimated the capacity using the intersection matrix with no geometry included. Song et al. (2021a) maximized the overlap of virtual spaces and found the minimum occupied space of the components. It turns out that the latter method is more accurate. One example of three components packing is shown in Fig. 4(b). We can deduce the relationship between the density and capacity as:

$$\beta_s \leq \beta_c \leq \beta_s + \beta_v \tag{1}$$

In the early problem design, space capacity is essential to the designer. If the layout problem is feasible, the capacity should be less than 1. The larger the value, the more difficult it is to find feasible solutions. And the capacity is more precise than the density.

3.3 Problem formulation

In most layout problems, the problem formulation comprises the various components, the geometrical and functional constraints, the qualitative and quantitative objectives. The functional constraints specify the functional requirements of components. Some components like external windows and doors are not orientation-free and need a specific direction to connect to the wall. The edge constraint is designed to force the component against the edge of the space because of a window or door. To simplify the formulation, we introduce the side location. The side location of component s_i can be extracted from two extreme corners' coordinates $(x_i, y_i, x_i + w_i, y_i + h_i)$. Each component s_i has four sides, defined as $(x_{L_i}, y_{B_i}, x_{R_i}, y_{T_i})$:

1. Left side location $x_{L_i} = x_i$
2. Bottom side location $y_{B_i} = y_i$
3. Right side location $x_{R_i} = x_i + w_i$
4. Top side location $y_{T_i} = y_i + h_i$

In order to force a component s_i to the side of a rectangular space $a = (x_a, y_a, w_a, h_a)$, defined by the bottom left coordinates and sizes, the component side locations $(x_{L_i}, y_{B_i}, x_{R_i}, y_{T_i})$ and available space side locations $(x_{L_a}, y_{B_a}, x_{R_a}, y_{T_a})$ should satisfy:

$$\min\{(x_{L_a} - x_{L_i})^2, (y_{B_i} - y_{B_a})^2, (x_{R_i} - x_{R_a})^2, (y_{T_i} - y_{T_a})^2\} = 0 \tag{2}$$

The layout problem is a multi-objective problem. The first objective aims to balance the mass distribution. It is calculated as the minimization of the Euclidean distance between gravity center of all solid components and geometry center of the container:

$$f_1 = \sqrt{\left(X_{gra} - X'_{gra}\right)^2 + \left(Y_{gra} - Y'_{gra}\right)^2} \tag{3}$$

$$X_{gra} = \frac{\sum_{i=1}^{n}\left(x_{c_i} \times m_i\right)}{\sum_{i=1}^{n} m_i}, Y_{gra} = \frac{\sum_{i=1}^{n}\left(y_{c_i} \times m_i\right)}{\sum_{i=1}^{n} m_i} \tag{4}$$

where x_{c_i}, y_{c_i} are the gravity center and m_i is the mass of s_i. X_{gra} and Y_{gra} are the gravity center of all solid components that can be obtained according to the sizes and coordinates. $X^{'}_{gra}, Y^{'}_{gra}$ represent the geometry center of the container.

Another objective f_2 optimizes the functional distance of components. In order to quantitatively describe the activity relationship between components, an activity factor is designed according to expert judgment to define the relationship between components. For instance, there is less circulation between energy network and ventilation, and the activity factor may be zero to reduce the distance effects. In contrast, it is important to limit interactions between the energy network and the electrical network, so the circulation distance should be maximized and the activity coefficient between them can be taken -1. The formulation can be expressed as:

$$f_2 = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij} \times \omega_{ij} \tag{5}$$

$$d_{ij} = \sqrt{\left(x_{c_i} - x_{c_j}\right)^2 + \left(y_{c_i} - y_{c_j}\right)^2} \tag{6}$$

where ω_{ij} represents activity factor and d_{ij} measures the Euclidean distance between component c_i and c_j.

The multi-objective layout optimization aims to find the arrangement (location and dimension) of components $\boldsymbol{c} = \{c_1, c_2, ..., c_n\}$, optimize objectives f_1, f_2 and satisfy geometrical (non-overlap and non-protrusion) and functional constraints (accessibility, edge).

4 Optimization approach

In the problem modelling, we introduce the virtual components connected to the solid component to deal with local accessibility. Indeed, the virtual space may be inaccessible from the entrance if there is no path to access it. The integration of virtual spaces is necessary but is not suffice for the component accessibility. The layout model is composed by a set of rectangular components. Therefore, the proposed method uses rectangle as the accessible space required by the user, shown in Fig. 5, where rectangles represent the path taken by the user inside the layout, in order to reach to the component

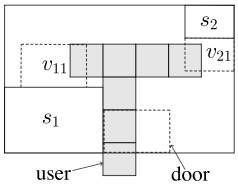


Fig. 5 Accessibility representation.

from the entrance. In our work, we characterizes component accessibility as a constraint during the optimization process. The more constraints, the more the design space is divided into separate zones. Also, the lager the capacity index, the harder it is to find feasible configurations. The layout problem is then more complex and effective constraint evaluation must be implemented to reduce the computational time. Thus, the optimization based on constructive approach is proposed addressing the accessibility optimization difficulties.

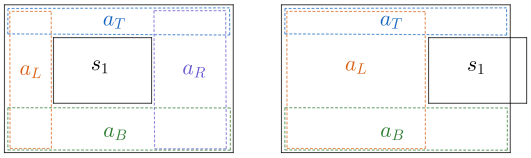
Algorithm 1 Optimization framework

While stop condition not met
  Given current state $X = (\boldsymbol{c}, \boldsymbol{p})$
  For each component $c_i = (s_i, v_{ij})$ with configuration p_i, $c_i \in \boldsymbol{c}$, $p_i \in \boldsymbol{p}$
    Space selection based on *strategy*
    Component placement subject to *accessibility analysis*
    Space generation
  } Constructive placement
  Evaluate objective functions $F = (f_1(X), f_2(X))$
  Neighbor generation $X_{new} \longleftarrow$ SA
Similarity analysis

This section explains the optimization framework, including the main elements of the constructive placement, simulated annealing and similarity analysis, described in Algorithm 1: a constructive placement algorithm, to place component $c_i \in \boldsymbol{c}$ following placement sequence $\boldsymbol{c} = (c_1, ..., c_n)$: firstly, selects the available space for component placement, then places the component within the area according to configuration sequence $\boldsymbol{p} = (p_1, ..., p_n)$ and determine the configuration among the accessibility satisfied candidates, finally updates space generation; a simulated annealing algorithm, to evaluate two layout designs constructed from two sequences $F = (f_1(X), f_2(X))$ and $F = (f_1(X_{new}), f_2(X_{new}))$, determines which combination is better, denoted as current state $X = (\boldsymbol{c}, \boldsymbol{p})$, then generates new sequences X_{new} improving objectives; a similarity analysis is followed after optimization to evaluate how similar the layout designs are and help designer to select the most favorable solution. The optimization algorithm circumvents the difficulty arising from the designed constraints. In particular, the constructive placement is introduced to decrease the computational complexity in the optimization.

4.1 Space generation

Constructive placement algorithm is inspired from Lai and Chan (1997). The former algorithm was developed for the cutting problem and the virtual components are not considered. To account for virtual spaces, we propose a novel constructive placement algorithm. It places the components with respect to the constraints and ensures the search for feasible solutions. The space around the placed components will be divided into available spaces. The available rectangular space is defined by the coordinates of lower left corner, the dimensions along the axes where $a = (x_a, y_a, w_a, h_a)$. The complete space generation between the component space and available space generates four candidate available spaces, named $a_L = (x_{a_L}, y_{a_L}, w_{a_L}, h_{a_L})$, $a_R = (x_{a_R}, y_{a_R}, w_{a_R}, h_{a_R})$, $a_T = (x_{a_T}, y_{a_T}, w_{a_T}, h_{a_T})$ and $a_B = (x_{a_B}, y_{a_B}, w_{a_B}, h_{a_B})$, as shown in Fig. 6(a). In contrast, if the component space and available space partially intersect, some candidate available space may not exist, for example in Fig. 6(b), the right side location x_{R_1} of component s_1 is not included in the available space, so the a_R does not exist. As mentioned before, the overlap of solid components is



(a) (b)

Fig. 6 Space generation (a) Complete included case, (b) Partial intersected case.

forbidden while the overlap of virtual components is allowed. To place components in the feasible regions, $\boldsymbol{a}$ tracks the available space generation of placed solid components while $\boldsymbol{a}'$ records the available space generation of placed solid and virtual components. And we have $\boldsymbol{a} = \{a_1, ..., a_m\}, \boldsymbol{a}' = \{a_1, ..., a_k\}$. The relationship between available spaces can be formulated as:

$$\forall i \in [1, k], \exists j \in [1, m], \quad a_i \sqsubseteq a_j \tag{7}$$

m and k represent the number of spaces in $\boldsymbol{a}$ and $\boldsymbol{a}'$. New virtual components will be placed in $\boldsymbol{a}$ to benefit overlap between virtual components, while new solid components will be placed in $\boldsymbol{a}'$ to guarantee non-overlap of solid components. After a component is placed, the space generation replaces the available space that intersects the component space with candidate available spaces. In addition, before adding candidate available spaces to the space list, it should remove the available space if it is included in any candidate available space, and filter out the candidate available space if it is included in any available space. The update aims to release storage space.

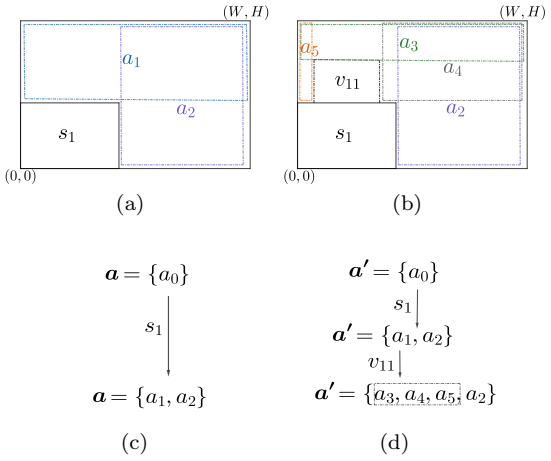


Fig. 7 Component placement and space generation (a) Space generation of $\boldsymbol{a}$, (b) Space generation of $\boldsymbol{a}'$, (c) Tree representation of $\boldsymbol{a}$, (d) Tree representation of $\boldsymbol{a}'$.

Fig. 7 illustrates one space generation example. At first, the available space in $\boldsymbol{a}$ and $\boldsymbol{a}'$ is initialized to the container size, $a_0 = [0, 0, W, H]$. Once the solid component s_1 is placed, a_0 will be divided into new available spaces $\{a_1, a_2\}$. The space generation is shown using a slicing tree representation in Fig. 7(c). Besides, the virtual component v_{11} placed in a_1 generates new available spaces $\{a_3, a_4, a_5\}$ in Fig. 7(d). The novel space generation integrates the available space generation of the placed solid and virtual components. Placing the new components in available spaces ensures the search for feasible solutions that satisfy the geometrical constraints.

4.2 Space selection

To place a component c_i, it should decide which available space will be used. The component configuration is decided by the selected space in lists $\boldsymbol{a}$ and $\boldsymbol{a}'$. The successive placement process can be treated as a combination problem. Thus, an effective space selection rule is essential for a constructive placement. In this paper, three space selection strategies are proposed and compared in experimental tests.

1. Strategy 1: Check all the combinations of spaces (a_i, a_j), $a_i \in \boldsymbol{a}'$ and $a_j \in \boldsymbol{a}$.
2. Strategy 2: Select one combination of spaces (a_i, a_j), $a_i \in \boldsymbol{a}'$, $a_j \in \boldsymbol{a}$ satisfying Eq. 7, a_i is the smallest sized space. The selection aims to successfully finish the construction process with less computational effort, namely space-filling strategy.
3. Strategy 3: Select one combination of spaces (a_i, a_j), $a_i \in \boldsymbol{a}'$, $a_j \in \boldsymbol{a}$ satisfying Eq. 7, a_i is the largest sized space.

4.3 Component placement

For a component c_i, it has four rotation configurations. The placement is performed only for available space in which the component fits, and there will be two possibilities according to the selection of the available space in $\boldsymbol{a}'$ and $\boldsymbol{a}$. If the selected space $a_i \in \boldsymbol{a}'$ and $a_j \in \boldsymbol{a}$ are coincide, then the component will be placed in the corners of selected space with four rotations. It ensures less margin space is generated and the non-overlap constraint is satisfied automatically. The feasible configurations are numbered from 1 to 16 as shown in Fig. 8, and we have the configuration sequence $p_i = (1, 2, ..., 16)$. Otherwise, the solid component will be placed in the corners of a_i, and the configurations in Fig. 8(b), (c) becomes the placements as shown in Fig. 9, where certain configurations will be adjusted according to the selected space a_j. One example is given in Fig. 10, instead of placing c_2 in the corner in Fig. 10(a), the position is refined to avoid overlapping with c_1 in Fig. 10(b).

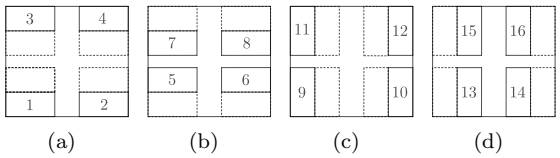


Fig. 8 Placement examples a_i and a_j are coincide.

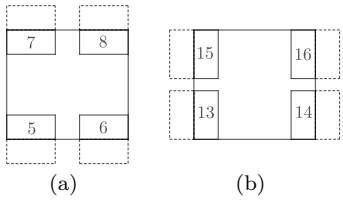


Fig. 9 Placement examples a_i and a_j are not coincide.

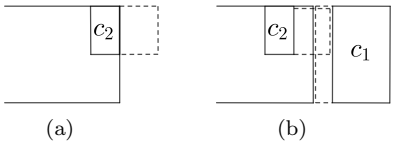


Fig. 10 Placement adjustment.

Assume that the current available spaces are kept in $\boldsymbol{a}$ and $\boldsymbol{a}'$. To place component c_i, the accessibility analysis in applied as summarized in Algorithm 2. The accessibility analysis builds the connection tree using spaces in

Algorithm 2 Accessibility analysis

1: Initialize the door space a_d.
2: Generate connection tree of available space list $\boldsymbol{a} = \{a_1, ..., a_i, ..., a_j, .., a_m\}$.
3: Find path for each placed virtual component.

$\boldsymbol{a}$ that generated by solid components. The root is a_d and the nodes are the connected space a_i, a_j in $\boldsymbol{a}$. The connection is measured by intersection space:

$$max(0, min(x_{R_{a_i}}, x_{R_{a_j}}) - max(x_{L_{a_i}}, x_{L_{a_j}})) \geq w_r \tag{8}$$

$$max(0, min(y_{T_{a_i}}, y_{T_{a_j}}) - max(y_{B_{a_i}}, y_{B_{a_j}})) \geq h_r \tag{9}$$

where the rectangle size (w_r, h_r) represents the accessible space required by the user. The connection is evaluated at each level of the tree. Once the tree generation is finished, check if there is one path for each placed virtual component (accessible from the door through available space; ignore the indirect connection between components). Assuming component $c_i = (s_i, v_{ij})$ is accessible from the entrance, there is at least one path for the human to reach the component. The path starts from the door space a_d and ends at the virtual space of the component v_{ij}. For the placed virtual component v_{ij}, find the corresponding space a_v where it placed inside. If there is a path start=a_d, end=a_v, path=[a_d, ..., a_v], then the component is accessible; otherwise, the component's configuration is not acceptable. One example is presented in Fig. 11, the placement of v_{11} occupies a_1, and there is one path=[a_d, a_2, a_1].

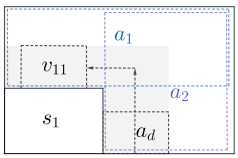


Fig. 11 Connection path [a_d, a_2, a_1].

Fig. 12 illustrates the procedure of accessibility analysis. The placements of (c_1, c_2, c_3, c_4) in Fig. 12(a) generate available spaces $\boldsymbol{a} = \{a_1, a_2, a_3, a_4, a_5\}$ in Fig. 12(b) in colors. The connection tree in Fig. 12(c) is generated based on the Eq. 8 and Eq. 9, where tree=$\{a_d : [a_1], a_1 : [a_2, a_3, a_5], a_2 : [a_4], a_4 : [], a_3 : [], a_5 : []\}$. And there exists at least one connection path for the placed virtual components:

– v_{11}: path=[a_d, a_1, a_2]

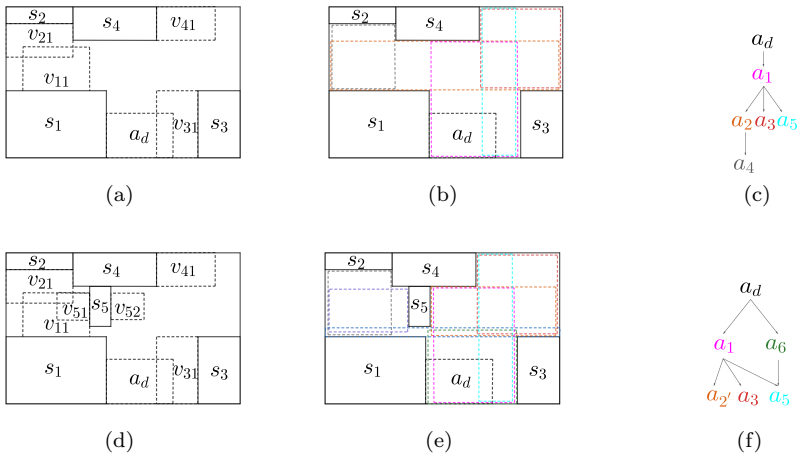


Fig. 12 Accessibility analysis (a) Placement of components $c_i = (s_i, v_{ij}), i \in (1, 2, 3, 4)$, (b) Space generation $\boldsymbol{a}$ of (s_1, s_2, s_3, s_4), (c) Connection tree $\boldsymbol{a}$ generated by (s_1, s_2, s_3, s_4), (d) Placement of components $c_i = (s_i, v_{ij}), i \in (1, 2, 3, 4, 5)$, (e) Space generation $\boldsymbol{a}$ of $(s_1, s_2, s_3, s_4, s_5)$, (f) Connection tree $\boldsymbol{a}$ generated by $(s_1, s_2, s_3, s_4, s_5)$.

- v_{21}: path=$[a_d, a_1, a_2, a_4]$
- v_{31}: path=$[a_d, a_1]$
- v_{41}: path=$[a_d, a_1, a_3]$

So the placed components are accessible and the current layout configuration is feasible. If the placement continues and the component c_5 is placed as shown in Fig. 12(d), then the space generation updates as in Fig. 12(e) and the connection tree becomes tree=$\{a_d : [a_1, a_6], a_1 : [a_{2'}, a_3, a_5], a_6 : [a_5], a_{2'} : [], a_3 : [], a_5 : []\}$ as shown in Fig. 12(f). There is a connection path for solid components s_3, s_4 while s_1, s_2 can not be accessible anymore. So the configuration does not satisfy the accessible requirement. The placement of c_5 will not be accepted as a feasible solution.

Indeed, the placement of component c_i may have more than one feasible configurations satisfying the geometrical and functional constraints. We need the criteria to select which configuration is used for the space generation. For the high-capacity layout problem, maximizing the space utilization to find feasible designs always has the highest priority. However, we notice that maximizing the overlap of virtual spaces sometimes conflicts with accessibility requirements. If the overlap maximization is too aggressive than other objectives, then the final solutions will converge to part of the feasible region. To balance the feasibility and the diversity, we classify the configurations based on the container boundary then select configuration according to the overlap maximization rule. A detailed explanation of component placement is summarized in Algorithm 3. In step 1, we select space and place the component to have the geometrically feasible configurations p_{ig}. Then the component placement

is determined by the boundary classification in step 2, accessibility verification in step 3 and overlap maximization in step 4.

Algorithm 3 Component placement

1: Place component c_i following the placement sequence $\boldsymbol{c}$. With the selected available space $(a_i, a_j), a_i \in \boldsymbol{a}', a_j \in \boldsymbol{a}$ according to the space selection strategy, go through the configuration sequence p_i to find all the feasible configurations that satisfy the geometric constraints, denoted as $p_{ig} = (i_1, i_2, ..., i_r), r \leq 16$.
2: Classify the configurations in p_{ig}. If the configuration is on the boundary, keep it in the first level p_{ib_1}, otherwise, keep it in the list p_{ib_2}. And the feasible configurations becomes $p_{ib} = (\underbrace{j_1, ..., j_l}_{p_{ib_1}}, \underbrace{j_{l+1}, ..., j_r}_{p_{ib_2}})$.
3: Check the accessibility requirement of the obtained configurations and filter out the unsatisfied candidates. The final feasible configurations list is $p_{ia} = (\underbrace{k_1, ..., k_h}_{p_{ia_1}}, \underbrace{k_{h+1}, ..., k_q}_{p_{ia_2}}), q \leq r \leq 16$.
4: If there are more than one feasible configurations in p_{ia_1}, sort the configuration list by computing the available space area in descending order. The first with maximum available space in p_{ia_1} will be selected as the prior choice; otherwise, select the first configuration with maximum available space in p_{ia_2}.

4.4 Simulated annealing algorithm

With the placement sequences $\boldsymbol{c}$ and configuration sequences $\boldsymbol{p}$, the placement algorithm can constructively build a layout. Moreover, the proposed algorithm uses the discrete formulation and the complexity is computed according to the combination possibilities $\mathcal{N}$. The complexity $\mathcal{N}$ relates to the number of spaces in $\boldsymbol{a}, \boldsymbol{a}'$. For the selected available space $(a_i, a_j), i \in [1, k], j \in [1, m]$, there are at most 16 feasible solutions $p_{ia} = (k_1, ..., k_q), q \leq 16$. If we check all combinations of available spaces in strategy 1, the combination complexity for one component equals $\mathcal{N} = q * m * k$. In strategy 2 and 3, check one selected available space, then the complexity becomes $\mathcal{N} = q$. The complexity of strategy 1 can be highly increased if the number of available spaces is quite large. Furthermore, as the number of components increases, the computational time to explore the sequence space using an exhaustive search approach increases exponentially. Hence, it is necessary to develop a meta-heuristic method to effectively search the feasible space.

Simulated annealing is a stochastic neighborhood search approach for global optimization and has been widely implemented in various combinatorial problems. The combinatorial optimization aims to find the optimal sequencing or permutation of multiple discrete terms. It originated from the concept in physics explaining the annealing of a solid until finding the minimal energy. Similar to the physical process, the annealing process generates a new solution in the neighborhood at each iteration. It allows to replace the current solution

with a worse neighborhood solution. The probability decreases along with the temperature, enabling hill-climbing.

In previous work, we proposed a nondomination-based simulated annealing to solve multi-objective problems and proved its good performance (Song et al., 2021a). So it is used to improve the placement and configuration sequences here. An external archive is used to keep non-dominated solutions during the optimization. And new sequences are generated by a swap operator. In the swap procedure of the placement sequence, σ is related to temperature t:

$$\sigma = n * exp^{(-1/t)} \tag{10}$$

The integer parameter $\sigma \in [1, n], n$ is the number of components, determines the process of neighbor generation. With high temperature and big σ, any two elements of the sequence can be exchanged; with low temperature and small σ, only the last few elements could be exchanged. The mechanism is the same in the configuration sequence. Given a state X, a layout with $F(X) = (f_1(X), f_2(X))$ is generated using the constructive placement. Multi-objective optimization aims at finding multiple non-dominated points. In SA, we consider the new state X_{new} as a better solution based on the nondomination relationship between $F(X_{new})$ and $F(X)$. Assuming that all objective functions are minimized, the domination can be expressed as:

$$\begin{cases} \forall i \in [1,2] & f_i(X_{new}) \leq f_i(X) \\ \exists j \in [1,2] & f_j(X_{new}) < f_j(X) \end{cases} \tag{11}$$

In fact, $F(X_{new})$ dominates $F(X)$ if $F(X_{new})$ is no worse than $F(X)$ for all objectives and $F(X_{new})$ is better than $F(X)$ for at least one objective. Then definitely X_{new} is a better solution. However, accepting a poor solution enables uphill moves sometimes, which is one of the cores of simulated annealing algorithm. The new state X_{new} will replace the current state X if one of the conditions is satisfied:

1. $F(X_{new})$ dominates $F(X)$.
2. $F(X)$ dominates $F(X_{new})$ and $rand(0,1) < exp^{(-(F(X_{new})-F(X))/t)}$
3. $F(X_{new})$ and $F(X)$ are non-dominated solutions, and $F(X_{new})$ is not dominated by any solution in the archive; or $F(X_{new})$ is not far from the obtained Pareto-front.

The annealing process determines how many temperature decreases are performed in the outer loop and the iterations per temperature. New solutions will be generated and compared in the inner loop. The temperature t is initialized as $t = t_s$ and reduced with the cooling rate r in the outer loop, $t = t * r$. The optimization searches for the better solution until it reaches the maximum iteration number L.

4.5 Similarity analysis

The proposed optimization addresses the difficulties of constraints and searches for feasible solutions effectively. In traditional multi-objective optimization,

optimization techniques can be evaluated by the convergence and diversity, that is, the comparison of the desired and obtained solutions. However, the layout problem optimization is special:

- Some requirements are difficult to model mathematically, therefore, they are typically simplified or ignored in optimization models.
- The multi-objective optimizer searches for trade-off solutions in both objectives. The optimal solution is subjectively selected by the designer.
- The final decision-making evaluates not only the performance in the objective space, but also the quality in the design space. However, there are fewer performance indicators for diversity evaluation in the design space.

Thus, layout optimization aims to search for diverse solutions with good objective values. The designer can choose among solutions to achieve the best compromise between optimization objectives. Therefore, we will keep all the feasible solutions in memory and evaluate similarities among them. Maintaining the diversity of solutions is important to guarantee interaction after optimization.

The similarity analysis is performed on the obtained solutions. In general, two layout designs are similar if they have similar configurations of certain components. To simplify a layout design, the relative position scheme is introduced to replace the original layout with a n-by-n matrix M in Eq. 12. Each binary element is a pairwise comparison of components (c_i, c_j). The binary variable defines the relative position of the components and ensures symmetrical configuration detection.

$$M = \begin{pmatrix} 00 & 01 & \cdots & 10 \\ 10 & 00 & \cdots & 01 \\ \vdots & \vdots & \vdots & \vdots \\ 11 & \cdots & \cdots & 00 \end{pmatrix} \tag{12}$$

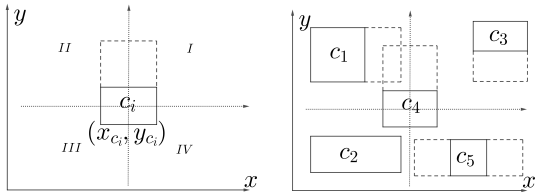
In a pairwise comparison of components (c_i, c_j), there are four possible relative positions I, II, III, IV of component c_j with respect to the reference component c_i, as shown in Fig. 13(a), it is determined according to the location of centroid, expressed as:

$$\alpha_{x_{ij}} = x_{c_i} - x_{c_j} \tag{13}$$

$$\alpha_{y_{ij}} = y_{c_i} - y_{c_j} \tag{14}$$

If $i = j$, we use the container as the reference component. The comparison is determined as follows:

1. I: $\alpha_{x_{ij}} \leq 0$ and $\alpha_{y_{ij}} \leq 0$, $M_{ij} = 00$. c_3 is in the region I of reference component c_4 in Fig. 13(b), $M_{34} = 00$
2. II: $\alpha_{x_{ij}} > 0$ and $\alpha_{y_{ij}} \leq 0$, $M_{ij} = 10$. c_1 is in the region II of reference component c_4 in Fig. 13(b), $M_{14} = 10$
3. III: $\alpha_{x_{ij}} > 0$ and $\alpha_{y_{ij}} > 0$, $M_{ij} = 11$. c_2 is in the region III of reference component c_4 in Fig. 13(b), $M_{24} = 11$



(a) (b)

Fig. 13 Relative position (a) Definition, (b) Representation.

4. IV: $\alpha_{x_{ij}} \leq 0$ and $\alpha_{y_{ij}} > 0$, $M_{ij} = 01$. c_5 is in the region IV of reference component c_4 in Fig. 13(b), $M_{54} = 01$

To evaluate the similarities among the relative position schemes, calculate an element-wise difference for each pair of the matrices M. The similarity value is expressed as the percentage of all elements that are the same. The similarity is between 0 to 1. The larger the value, the higher the similarity. For symmetrical configuration detection, convert $\alpha_{x_{ij}} = 1 - \alpha_{x_{ij}}$ to check the bilateral symmetry, and $\alpha_{y_{ij}} = 1 - \alpha_{y_{ij}}$ to check the longitudinal symmetry.

5 Experiment result

In this section, first, three different layout examples are formulated to assess the developed constructive placement. Then the proposed optimization approach and the comparative algorithms are applied to solve the practical shelter problem provided by Thales. The algorithms were coded using Object-Oriented Programming language in Python.

5.1 Constructive placement strategy comparison

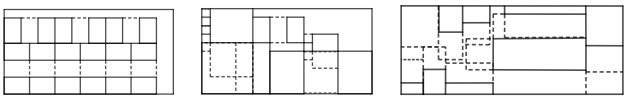
Constructive placement is designed to circumvent the difficulty arising from constraints. With a given number of iterations, the more feasible solutions it finds, the better the performance. Here, we use three layout examples to test the different strategies in the constructive placement. The strategy is proposed to place component in appropriate space with respect to the constraints. The strategy involves two aspects of configuration sequence and space selection. The test examples properties are summarized in Table 2, including the number, the density and capacity, the size of the components, and the functional constraints.

5.1.1 Test 1 – Equal-sized component in Fig. 14(a)

The equal-sized component can eliminate the effect of different placement sequences. The problem is concerned with placing 18 equal-sized components

Table 2 Properties of test examples.

	Test 1	Test 2	Test 3
Number of components	18	11	9
Density of solid components	0.54	0.47	0.38
Density of virtual components	0.54	0.66	0.65
Density of solid and virtual components	1.08	1.13	1.03
Capacity	0.81	0.75	0.77
Equal size	Yes	No	No
Accessibility	No	Yes	Yes
Number of components edge on the wall	0	0	2



(a) Test 1 (b) Test 2 (c) Test 3

Fig. 14 Test examples.

into a container, width is 4000 mm, and height is 2000 mm, respecting geometrical constraints. The virtual component, symbolized by the dotted rectangle, has the same size as the solid component. And the dimensions are given in Table 3.

Table 3 Data in Test 1.

Item	Dim/w (mm)	Dim/h (mm)
1-18.type 1 components	600	400

5.1.2 Test 2 – Unequal-sized component in Fig. 14(b)

The unequal-sized component is more common and realistic. The problem involves accessibility requirements and geometrical constraints. It has four types of components: type 1, type 2, type 4 components, each with 1-equal size virtual component, and type 3 component with 2-equal size virtual components. The door, fixed to the lower left corner of the container, is modeled as the virtual component. The container size is the same as in Test 1. The detailed dimensions of components are given in Table 4.

5.1.3 Test 3 – Big-sized component in Fig. 14(c)

The big-sized component introduces the size difference issue into test instances. The container is rectangular with a width of 5945 mm and a height of 2286 mm. The big-sized component, namely type 1 component, has three virtual

Table 4 Data in Test 2.

Item	Dim/w (mm)	Dim/h (mm)
1-3.type 1 components	600	400
4-6.type 2 components	1000	800
7-9.type 3 components	200	200
10.type 4 components	1200	400
11.door	1200	1200

components: two virtual components with a width of 2469 mm and a height of 600 mm, one virtual component with a width of 800 mm and a height of 841 mm, and occupies almost half of the container space as shown in Fig. 15. Except for geometrical constraints, additional constraints include edge on the

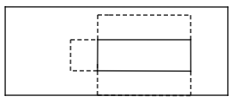


Fig. 15 Big-sized component representation.

wall, alignment, and accessibility of components. The alignment specifies that type 2 component must attach to the right side of type 1 component. In addition, type 6 and type 7 components must place against one wall of the container, this requirement can be found in the air conditioner or other window like component. The door, fixed to the upper left corner of the container, is modeled as the virtual component. The other components, each with a virtual component attached, have the same width as the solid component and the height of 600 mm. The dimensions are given in Table 5.

Table 5 Data in Test 3.

Item	Dim/w (mm)	Dim/h (mm)
1.type 1 component	2469	841
2.type 2 component	860	1100
3.type 3 component	650	650
4.type 4 component	600	600
5.type 5 component	2320	350
6.type 6 component	800	406
7.type 7 component	1330	283
8.type 8 component	600	300
9.door	1060	1060

The strategy comparison results are summarized in Table 6 where the number of solutions is obtained with a given number of iterations. Three space selection strategies are compared with/without configuration permutations. If we do not permute the configuration sequence, then the configuration sequence is fixed. In Test 1, if there is no configuration permutation, the constructive

Table 6 Component placement strategy comparison.

Number of solutions		Test 1	Test 2	Test 3
Fix configuration sequence	Space selection strategy 1	1/100	2/100	24/600
	Space selection strategy 2	1/100	37/100	21/600
	Space selection strategy 3	1/100	5/100	2/600
Permute configuration sequence	Space selection strategy 1	22/100	5/100	51/600
	Space selection strategy 2	26/100	40/100	50/600
	Space selection strategy 3	13/100	6/100	18/600

placement will find one feasible solution with poor diversity. In order to overcome the limitation, the configuration permutation is included in the constructive process, and we can see that the number of solutions is improved dozens of times. In Test 2, both geometrical and functional constraints are considered. Because of the unequal-sized component, three strategies with fixed configuration sequence can find more than one feasible solution. It is worth noting that it results from the placement sequence but not the configuration permutation. Besides, it turns out that the performance of strategy 1 will decrease if there is accessibility requirement. Furthermore, strategy 2 starts with the smallest size of the available space and fills the container space gradually. It helps the placement to finish the constructive process. In Test 3, the number of feasible solutions in strategy 1 and 2 are at the same level. However, since strategy 1 goes through all of the combinations of the available space, it is tedious compared to strategy 2. After the comparison, we can conclude that

- The permutation of configuration sequence is necessary for diversity in the design space, especially in the case of equal-sized component.
- Considering the search ability under constraints, strategy 2 is much better than strategy 3. The edge constraints can improve the performance of strategy 1 but it is time-consuming. Strategy 2 conducts the placement effectively and achieves similar or better results compared to the others.

5.2 Practical example

Based on the above analysis, strategy 2 with configuration permutation is effective in generating feasible solutions and proved to be the best placement strategy for developing the multi-objective optimization algorithm. The proposed layout model represents an unexplored layout feature, namely accessibility, compared to current research. Thus, none of the existing layout optimization methods are applicable. However, we would like to demonstrate the feasible difficulty of the layout optimization using the Sequential Least Squares Programming (SLSQP) and the non-dominated sorting genetic algorithm (NSGAII) in a practical shelter layout example. In contrast, the proposed optimization algorithm resolves the layout problem efficiently and achieves highly preferable results.

The layout problem studied here, as shown in Fig. 16, is a three-dimensional shelter with three different spaces, named storage zone, technical zone and

operator zone. Light and mobile shelter with on-board equipment provides complete protection for personnel and against battlefield aggression. Its versatility means a variety of armed forces can use the shelter. For our first study,

Fig. 16 3D representation.

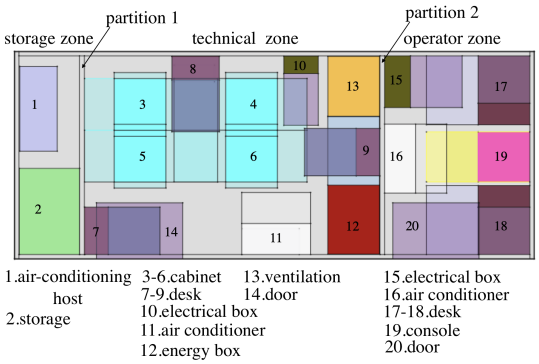


Fig. 17 2D configuration model.

we simplify the shelter into two-dimensional because the components are full height, as shown in Fig. 17. Consequently, the evaluation of constraints is more easily realized. The data of the container is given in Table 7.

Table 7 Data of container in 2D model.

Item	Dim/W (mm)	Dim/H (mm)
container	5930	2306
storage zone	703	2306
technical zone	3400	2306
operator zone	1717	2306

Moreover, each component is represented by a rectangle. A set of virtual components attached to components (light color) represent accessibility spaces.

Table 8 Data of components in storage zone.

Item	Dim/w (mm)	Dim/h (mm)	Mass/m (kg)
1.air-conditioning host	435	983	180
2.storage	700	1000	150

Table 9 Data of components in technical zone.

Item	Dim/w (mm)	Dim/h (mm)	Mass/m (kg)
3.cabinet	600	600	420
4.cabinet	600	600	420
5.cabinet	600	600	420
6.cabinet	600	600	274
7.desk	277	550	10
8.desk	550	277	34
9.desk	277	550	10
10.electrical box	400	203	48
11.air conditioner	795	353	70
12.energy box	600	800	500
13.ventilation	575	680	72
14.door	1000	5	120

Table 10 Data of components in operator zone.

Item	Dim/w (mm)	Dim/h (mm)	Mass/m (kg)
15.electrical box	300	600	54
16.air conditioner	353	795	54
17.desk	600	800	54
18.desk	600	800	54
19.console	600	580	420
20.door	1000	5	120

For example, virtual spaces of the cabinet guarantee interaction and correct usability. Dimensions described in Table 8, Table 9 and Table 10 match the configuration in Fig. 17. The size of the virtual component, either equal to the size of the attached solid component, or set to 600 mm, represents the size of the accessible space. Except for the accessibility constraint, there are other functional constraints of the application:

1. The cabinets are placed in an allowed space, the 70 mm virtual space is used to avoid full attachment to the wall and is also dedicated the shock absorbers freedom.
2. The desk 8 is grouped with a cloison, the cloison is a window-like component that has to be attached to the external wall of the shelter.
3. The desk 7 and desk 9 can be fully folded and can overlap with all virtual spaces. However, the overlap of desks is forbidden considering the possibility of two people working simultaneously. So they are temporary solid components.

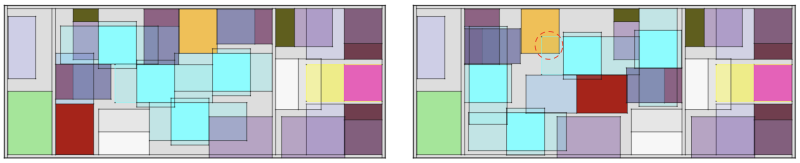
4. The electrical boxes and the air conditioners have to be placed against the wall.
5. The ventilation maintains from the outside, therefore has to be on the back wall and no rotation is allowed.
6. The doors are accessible from the exterior and the virtual space is used for a door opening from outside.

Here we consider the technical zone in the shelter problem as the single container optimization problem. The technical zone's capacity equals 0.82, which is the most complex layout compared to the test examples with functional constraints. The distance between cabinet 3, cabinet 4, cabinet 5 and energy box 12 should be maximized. For simplicity, we list the activity relationship of these four components in Table 11. So the two objectives are defined as: minimizing layout balance (objective 1) and maximizing distance (objective 2).

Table 11 Activity factor.

Item	3.cabinet	4.cabinet	5.cabinet	12.energy box
3.cabinet	0	0	0	-1
4.cabinet	0	0	0	-1
5.cabinet	0	0	0	-1
12.energy box	-1	-1	-1	0

– Optimization using SLSQP. The continuous optimization is based on Sequential Least Squares Programming (SLSQP), which can handle equality and inequality constraints. The two objectives of the layout problem are combined into a single objective with the same weight to the local optimization. We have implemented this algorithm using *minimize* function available in Python with 10000 iterations. Since it is local optimization, it is fast to find the solution as shown in Fig. 18(a). However, it is difficult to jump out of the current search region and get stuck in the infeasible region as shown in Fig. 18(b). In other words, the final solution highly depends on the initial solution. Setting different initial configurations through interaction seems to work, but it is not realistic.



(a) (b)

Fig. 18 SLSQP optimization result.

– Optimization using NSGAII. Genetic algorithm is the most popular nature-inspired evolutionary algorithm. It improves the solution through crossover and mutation operators and survival selection. NSGAII is implemented based on non-dominated sorting technique in multi-objective optimization. We have implemented this algorithm using the Pymoo available in Python. The population size is taken as 100, the number of generation equals to 100. The simulated binary crossover probability is 0.9 and polynomial mutation probability is $1/n$, where n is the number of decision variables. Since it is a population-based algorithm, the evolution evolved with generation. However, the layout problem studied here is quite complex with high capacity. So the algorithm may generate many infeasible solutions where the non-overlap constraint cannot be satisfied. Even if different operators are used for new individuals generation, the non-overlap satisfied individuals are still very sparse. Finally, the optimization has a high possibility to converge into a small niche of the solution space. This phenomenon can be seen in Fig. 19, and the Pareto-front is generated by the local movement of components in one type of layout design.

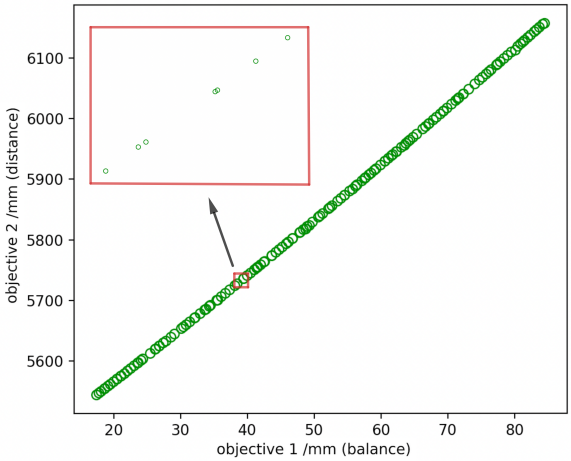


Fig. 19 NSGAII result.

Based on the above analysis, it proves that applying the continuous optimization algorithm has limited performance. Moreover, there is no method that is much better than the other. The constructive placement makes the search for feasible solutions easier but the combination complexity remains high. Therefore, it is better to guide the search for constructive placement. As far as we know, simulated annealing is more lightweight on a large scale or a large number of iterations than genetic algorithm. So we use simulated annealing to optimize the configuration sequences and the placement sequences, and apply

constructive placement generate feasible solutions. The initial temperature t_s is initialized according to the algorithm in Ben-Ameur (2004) where $t_s = 100$. Set the cooling rate $r = 0.9$ and the total number of iterations $L = 4000$ to perform the annealing process.

The optimization algorithm outputs the configurations of the layout under the technical zone area, which is the most complex area in this shelter, through limited iterations. The algorithm searches for solutions by considering the geometrical and functional constraints and objectives of the problem formulation. The optimization algorithm generated 337 feasible solutions, as shown in Fig. 20. We can see that the exploration region distributed in objec-

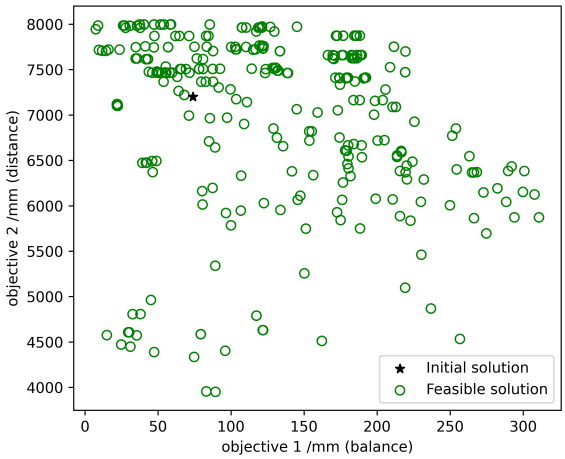


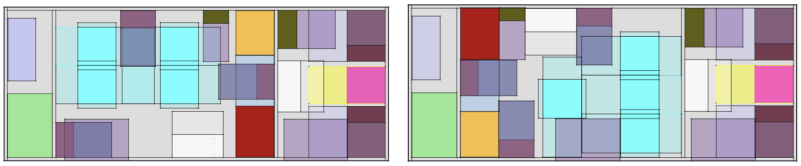
Fig. 20 Display of solutions in objective space.

tive space is no longer a niche. Besides, the proposed algorithm can generate more choices for designers. At the same time, it can find better solutions in both objectives compared to the initial solution. For example, the solution realizes the best compromise between optimization objectives. Fig. 21 presents the initial solution , and an optimal solution that realizes the maximum value of objective 2. It can be seen that there is a significant difference between the optimal layout solution and the initial solution. Table 12 illustrates that the

Table 12 Optimal solution and initial solution.

Objective	Initial solution	Optimal solution
objective 1/mm(minimization)	74.3	47.4
objective 2/mm(maximization)	7164.6	8003.9

optimal solution can achieve much better objective values compared to the



(a) Initial solution (b) Optimal solution

Fig. 21 Solution in design space.

initial one. Indeed, the initial configuration created by the engineers of Thales was generated from geometrical aspects. The experimental results prove that the proposed algorithm is effective in solving the layout problems under functional constraints.

5.3 Similarity analysis

It is proved that the optimization based on constructive approach can generate high-qualified solutions that are well-distributed in objective space. However, the diversity in design space is also important. Therefore, we now analyze the similarities among the obtained feasible solutions. To simplify the demonstration, we first apply non-dominated sorting to select solutions in the rank range $[1, 13]$ and the number of solutions reduced to 130. Then we randomly select six solutions as shown in Fig. 22. The corresponding configurations are given in Fig. 23 and each solution is a new variant compared to the other (at least one different component configuration).

The similarity indicators for paired layout designs formulate a similarity symmetric matrix, represented in Fig. 24(a). By comparison, designs 4 and 5 have higher similarity values compared to design 0. Considering the different variants, it is necessary to identify the difference and cluster similar sets. Considering the undetermined number of clusters, the similarity matrix is analyzed using hierarchy cluster algorithm Müllner (2011). The algorithm uses a distance matrix to merge similar solutions consecutively and builds nested clusters until there is only one cluster left. The hierarchical similarity relationship of the selected designs is presented in Fig. 24(b). It can detect the geometrical differences between configurations and identify similar groups. For example, the similar designs 4 and 5 are assigned into one group, whereas designs 0 and 2 are grouped into another cluster. The visualization tool can provide information on the hierarchical similarity of designs, helping users to quickly select the preferred solution.

After preliminary experiments, we now apply the similarity analysis to ranked solutions, and clustering results are shown in Fig. 25. If we set the threshold to define the clusters, we can have the different grouped solutions, as shown in Fig. 26, the same grouped solutions have the same color. It is proved that, close points in objective space can have different configurations

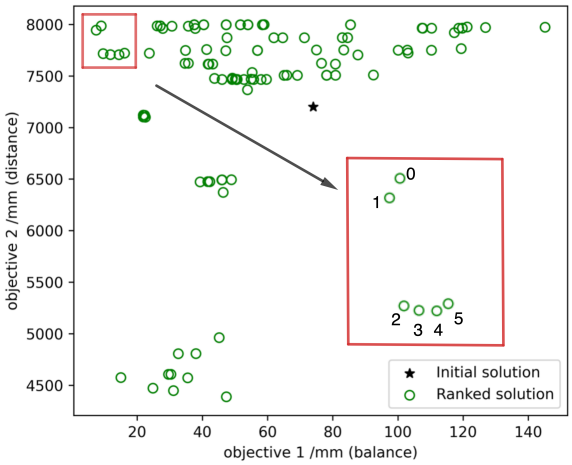


Fig. 22 Display of ranked solutions.

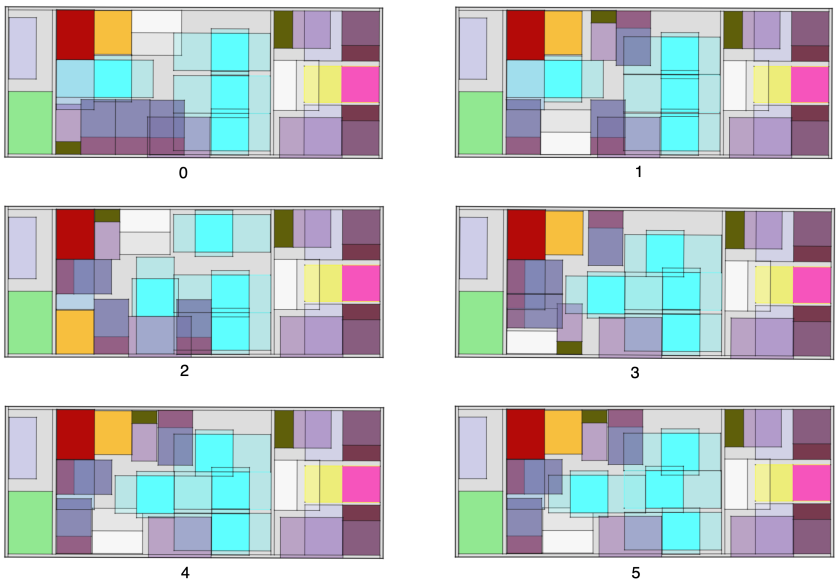
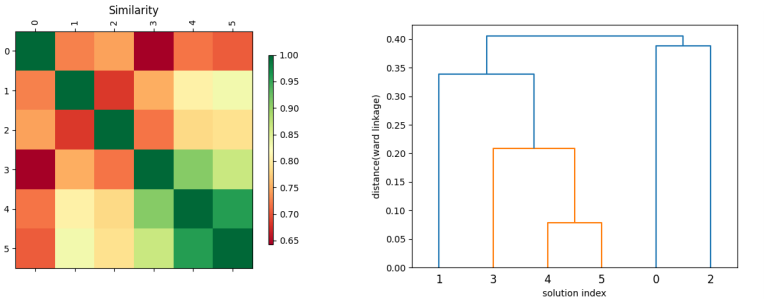


Fig. 23 Display of selected configurations.

in design space and vice versa. In the layout optimization, configurations affect the system performance directly. The similarity analysis makes the solution selection more reliable.



(a) Similarity matrix (b) Hierarchical cluster dendrogram

Fig. 24 Similarity analysis.

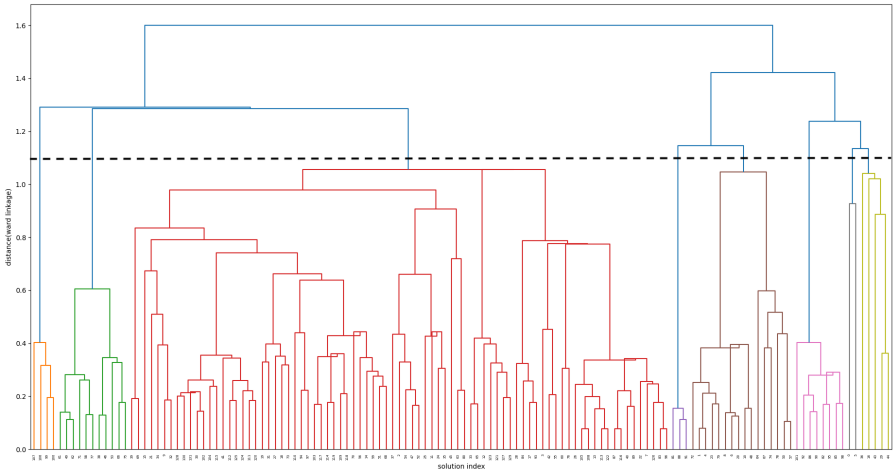


Fig. 25 Display of cluster dendrogram.

6 Conclusion

This paper presents an optimization based on constructive approach to solve the novel multi-objective layout problem model. The model, taking into account the virtual components and the accessibility to components, requires an effective optimization algorithm for addressing the feasible difficulty. Two objectives, namely layout balance and activities, are considered. The continuous optimizer may get stuck and fail to jump out of (in-)feasible search region. Instead, a multi-objective optimization method integrates accessibility analysis with simulated annealing is developed.

The accessibility analysis is conducted by the constructive placement. The placement, not only integrates space generation of solid and virtual components guaranteeing non-overlap of components, but also introduces the idea of connection path ensuring accessibility of components. Layout solution is

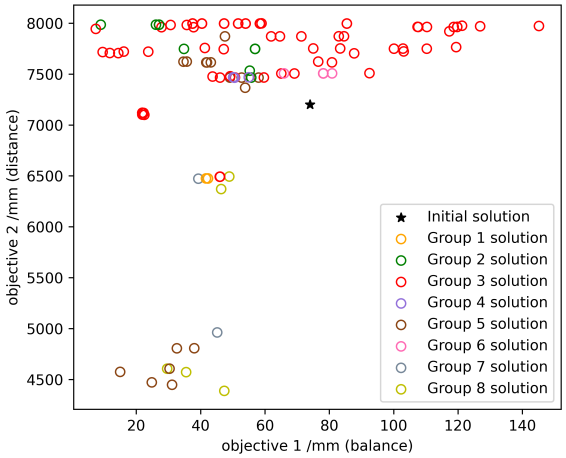


Fig. 26 Display of cluster in objective space.

constructed sequentially, which is a combination process. Simulated annealing search technique explores combinations of component configurations and optimizes both objectives simultaneously. The strategy comparisons confirm that the space-filling strategy can effectively generate feasible solutions and reduce computational efforts. The experimentation proves that the proposed optimization is effective in ensuring accessibility and finding high-qualified solutions compared to the existing algorithms. Moreover, the similarity analysis demonstrates good diversity of the obtained layout set, which can be applied as an interactive tool.

The developed optimization method uses the space generation of rectangular shapes. Further research could adapt the available space generation to the free-form component. Besides, the proposed layout problem model assumes having one container space. However, it could be interesting if the space division, i.e., the partition in the shelter problem, is formulated as one variable to ensure the automatic layout design during the optimization process. Afterward, we will investigate the multi-container layout problems.

Acknowledgements This work was supported by China Scholarship Council. The authors would like to acknowledge Thales for the application study.

Conflict of interest

The authors declare that they have no conflict of interest.

References

Allahyari MZ, Azab A (2018) Mathematical modeling and multi-start search simulated annealing for unequal-area facility layout problem. Expert Syst Appl 91:46–62, DOI https://doi.org/10.1016/j.eswa.2017.07.049

Baykasoğlu A, Gindy NN (2001) A simulated annealing algorithm for dynamic layout problem. Comput & Oper Res 28(14):1403–1426, DOI https://doi.org/10.1016/S0305-0548(00)00049-6

Ben-Ameur W (2004) Computing the initial temperature of simulated annealing. Comput Optim Appl 29:369–385, DOI https://doi.org/10.1023/B:COAP.0000044187.23143.bd

Bénabès J, Bennis F, Poirson E, Ravaut Y (2010) Accessibility in Layout Optimization. In: 2nd International Conference On Engineering Optimization, Lisbonne, Portugal

Bénabès J, Poirson E, Bennis F, Ravaut Y (2011) Interactive modular optimization strategy for layout problems. In: Proceedings of the ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Washington, DC, United States, pp 553–562, DOI http://doi.org/10.1115/DETC2011-47925

Bénabès J, Guédas B, Poirson E, Bennis F (2012) Indicator of feasibility for layout problems. In: Proceedings of the ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, Illinois, United States, pp 727–734, DOI http://doi.org/10.1115/DETC2012-70640

Cagan J, Mitchell WJ (1993) Optimally directed shape generation by shape annealing. Environ Plan B: Plan Des 20(1):5–12, DOI https://doi.org/10.1068/b200005

Cagan J, Degentesh D, Yin S (1998) A simulated annealing-based algorithm using hierarchical models for general three-dimensional component layout. Computer-Aided Des 30(10):781–790, DOI https://doi.org/10.1016/S0010-4485(98)00036-0

Cagan J, Shimada K, Yin S (2002) A survey of computational approaches to three-dimensional layout problems. Computer-Aided Des 34(8):597–611, DOI https://doi.org/10.1016/S0010-4485(01)00109-9

Chraibi A, Kharraja S, Osman IH, El-Beqqali O (2016) A particle swarm algorithm for solving the multi-objective operating theater layout problem. IFAC-PapersOnLine 49(12):1169–1174, DOI https://doi.org/10.1016/j.ifacol.2016.07.663

Cuco A, Sousa F, Vlassov V, Silva Neto A (2011) Multi-objective design optimization of a new space radiator. Optim Eng 12:393–406, DOI https://doi.org/10.1007/s11081-011-9142-6

Cuco A, Sousa F, Silva Neto A (2014) A multi-objective methodology for spacecraft equipment layouts. Optim Eng 16, DOI https://doi.org/10.1007/s11081-014-9252-z

Du Pont BL, Cagan J (2012) An Extended Pattern Search Approach to Wind Farm Layout Optimization. J Mech Des 134(8), DOI https://doi.org/10.

1115/1.4006997
Fossati G, Miguel L, Paucar Casas W (2019) Multi-objective optimization of the suspension system parameters of a full vehicle model. Optim Eng 20, DOI https://doi.org/10.1007/s11081-018-9403-8
Gao X, Hu X, Feng X, Feng W, Hu Y, Tang X (2019) Layout optimization design of power iot chips. In: 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), pp 1620–1624, DOI https://doi.org/10.1109/IAEAC47372.2019.8998045
Garcia-Hernandez L, Palomo-Romero J, Salas-Morera L, Arauzo-Azofra A, Pierreval H (2015) A novel hybrid evolutionary approach for capturing decision maker knowledge into the unequal area facility layout problem. Expert Syst Appl 42(10):4697–4708, DOI https://doi.org/10.1016/j.eswa.2015.01.037
Ghassemi Tari F, Neghabi H (2015) A new linear adjacency approach for facility layout problem with unequal area departments. J Manuf Syst 37:93–103, DOI https://doi.org/10.1016/j.jmsy.2015.09.003
Gonçalves JF, Resende MG (2015) A biased random-key genetic algorithm for the unequal area facility layout problem. Eur J Oper Res 246:86–107, DOI https://doi.org/10.1016/j.ejor.2015.04.029
Grignon PM, Fadel GM (2004) A GA Based Configuration Design Optimization Method . J Mech Des 126(1):6–15, DOI https://doi.org/10.1115/1.1637656
Guarneri P, Dandurand BC, Fadel GM, Wiecek MM (2013) Bilevel multiobjective optimization of vehicle layout. In: Proceedings of the 10th World Congress on Structural and Multidisciplinary Optimization, p 19–24
Halawa F, Chalil Madathil S, Khasawneh MT (2021a) Integrated framework of process mining and simulation–optimization for pod structured clinical layout design. Expert Syst Appl 185:115696, DOI https://doi.org/10.1016/j.eswa.2021.115696
Halawa F, Madathil SC, Khasawneh MT (2021b) Multi-objective unequal area pod-structured healthcare facility layout problem with daylight requirements. Comput Ind Eng 162:107722, DOI https://doi.org/10.1016/j.cie.2021.107722
Hasda R, Bhattacharjya R, Bennis F (2016) Modified genetic algorithms for solving facility layout problems. Int J Interact Des Manuf 11:713–725, DOI https://doi.org/10.1007/s12008-016-0362-z
Hosseini nasab H, Fereidouni S, Ghomi S, Fakhrzad M (2018) Classification of facility layout problems: a review study. Int J Adv Manuf Technol 94:957–977, DOI https://doi.org/10.1007/s00170-017-0895-8
Huo J, Liu J, Gao H (2021) An nsga-ii algorithm with adaptive local search for a new double-row model solution to a multi-floor hospital facility layout problem. Appl Sci 11(4):1758, DOI https://doi.org/10.3390/app11041758
Kang S, Chae J (2017) Harmony search for the layout design of an unequal area facility. Expert Syst Appl 79:269–281, DOI https://doi.org/10.1016/j.eswa.2017.02.047

Lai K, Chan JW (1997) Developing a simulated annealing algorithm for the cutting stock problem. Comput Ind Eng 32(1):115–127, DOI https://doi.org/10.1016/S0360-8352(96)00205-7

Li X, Zhao Z, Zhang K (2014) A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins. In: Industrial and Systems Engineering Research Conference, pp 2039–2048

Lin Z, Yingjie Z (2019) Solving the facility layout problem with genetic algorithm. In: 2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA), pp 164–168, DOI https://doi.org/10.1109/IEA.2019.8715148

Masoudi N, Fadel G (2021) An Optimization Framework for the Design of Cable Harness Layouts in Planar Interconnected Systems. J Mech Des 144(1), DOI https://doi.org/10.1115/1.4051685

Mazinani M, Abedzadeh M, Mohebali N (2012) Dynamic facility layout problem based on flexible bay structure and solving by genetic algorithm. Int J Adv Manuf Technol 65:929–943, DOI https://doi.org/10.1007/s00170-012-4229-6

Miao Y, Fadel G, Gantovnik V (2008) Vehicle configuration design with a packing genetic algorithm. Int J Heavy Vehicle Sys 15:433–448, DOI https://doi.org/10.1504/IJHVS.2008.022252

Michalek J, Choudhary R, Papalambros P (2002) Architectural layout design optimization. Eng Optim 34:461–484, DOI https://doi.org/10.1080/03052150214016

Moradi N, Shadrokh S (2019) A simulated annealing optimization algorithm for equal and un-equal area construction site layout problem. Int J Res p 89–104, DOI https://doi.org/10.22105/RIEJ.2019.169867.1073

Müllner D (2011) Modern hierarchical, agglomerative clustering algorithms. 1109.2378

Niroomand S, Hadi-Vencheh A, Şahin R, Vizvári B (2015) Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems. Expert Syst Appl 42(19):6586–6597, DOI https://doi.org/10.1016/j.eswa.2015.04.040

Ou-Yang C, Utamima A (2013) Hybrid estimation of distribution algorithm for solving single row facility layout problem. Comput Ind Eng 66:95–103, DOI https://doi.org/10.1016/j.cie.2013.05.018

RazaviAlavi S, AbouRizk S (2017) Site layout and construction plan optimization using an integrated genetic algorithm simulation framework. J Comput Civil Eng 31:04017011, DOI https://doi.org/10.1061/(ASCE)CP.1943-5487.0000653

Ripon KSN, Glette K, Khan KN, Hovin M, Torresen J (2013) Adaptive variable neighborhood search for solving multi-objective facility layout problems with unequal area facilities. Swarm Evolut Comput 8:1–12, DOI https://doi.org/10.1016/j.swevo.2012.07.003

Saraswat A, Venkatadri U, Castillo I (2015) A framework for multi-objective facility layout design. Comput Ind Eng 90:167–176, DOI https://doi.org/10.1016/j.cie.2015.09.006

Seo J, Jung J, Kim S, Shin Y (2017) Pin accessibility-driven cell layout redesign and placement optimization. In: Proceedings of the 54th Annual Design Automation Conference, pp 1–6, DOI https://doi.org/10.1145/3061639.3062302

Shayan E, Chittilappilly A (2004) Genetic algorithm for facilities layout problems based on slicing tree structure. Int J Prod Res 42:4055–4067, DOI https://doi.org/10.1080/00207540410001716471

Singh SP, Sharma R (2008) Two-level modified simulated annealing based approach for solving facility layout problem. INT J PROD RES 46:3563–3582, DOI https://doi.org/10.1080/00207540601178557

Song X, Poirson E, Ravaut Y, Bennis F (2021a) Efficient multi-objective simulated annealing algorithm for interactive layout problems. Int J Interact Des 15:441–451, DOI https://doi.org/10.1007/s12008-021-00773-1

Song X, Poirson E, Ravaut Y, Bennis F (2021b) Interactive design optimization of layout problems. In: Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems, pp 387–395, DOI https://doi.org/10.1007/978-3-030-85914-5_41

Szykman S, Cagan J (1997) Constrained Three-Dimensional Component Layout Using Simulated Annealing. J Mech Des 119(1):28–35, DOI https://doi.org/10.1115/1.2828785

Tiwari S, Fadel G, Gantovnik V (2006) A survey of various encoding schemes and associated placement algorithms applied to packing and layout problems. In: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol Volume 1: 32nd Design Automation Conference, Parts A and B, pp 609–618, DOI https://doi.org/10.1115/DETC2006-99271

Xiao Y, Seo Y, Seo M (2013) A two-step heuristic algorithm for layout design of unequal-sized facilities with input/output points. Int J Prod Res 51(14):4200–4222, DOI https://doi.org/10.1080/00207543.2012.752589

Xie W, Sahinidis NV (2008) A branch-and-bound algorithm for the continuous facility layout problem. Comput Chem Eng 32(4):1016–1028, DOI https://doi.org/10.1016/j.compchemeng.2007.05.003

Zawidzki M, Szklarski J (2020) Multi-objective optimization of the floor plan of a single story family house considering position and orientation. Adv Eng Softw 141:102766, DOI https://doi.org/10.1016/j.advengsoft.2019.102766

Zheng H, Ren Y (2020) Architectural layout design through simulated annealing algorithm. In: Proceedings of the 25th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), pp 275–284, DOI https://doi.org/10.52842/conf.caadria.2020.1.275

Şahin R (2011) A simulated annealing algorithm for solving the bi-objective facility layout problem. Expert Syst Appl 38(4):4460–4465, DOI https://doi.org/10.1016/j.eswa.2010.09.117