International Journal on Interactive Design and Manufacturing manuscript No.
(will be inserted by the editor)

Efficient multi-objective simulated annealing algorithm for interactive layout problems

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

Received: date / Accepted: date

Abstract This paper presents an efficient simulated annealing algorithm for solving multi-objective layout problems where several rectangular components are placed, respecting non-overlap and non-protrusion constraints in the given space. Resolving layout problems can be very hard in some industrial cases because problems are over-constrained and computing feasible optimal layout designs are time consuming. In most practical problems, both real and virtual components exist. The virtual components represent the required accessible space allowing the user to access to the component. The virtual components can overlap with each other, while the overlap is not allowed for the real components. Considering the limited layout design space, the capacity of the layout problem is analyzed using constructive placing techniques. To explore the feasible layout space, a hybrid simulated annealing is proposed to determine the order of placement; then, a constructive placing strategy based on empty maximal space is developed. What's more, an interactive visualization environment is introduced between the optimization and expert. The proposed algorithm is the first attempt to search feasible space of multi-objective layout design by constructive placing method.

Keywords Layout problem · Capacity · Simulated annealing · Constructive placing strategy · Interactive optimization

* This work was supported by China Scholarship Council
[1] ✉ X. Song · F.Bennis · E. Poirson
Laboratoire des sciences du numérique de Nantes (LS2N), ECN, UMR 6004, 44321 Nantes, France.
E-mail: Firstname.Lastname@ec-nantes.fr
[2] Y. Ravaut
Thales Communications, 49300 Cholet, France.
E-mail: yannick.ravaut@thalesgroup.com

1 Introduction

The layout problems (LPs) concern finding the optimal arrangements of a given number of components with known dimensions within a given container. The component can be the equipment, device, cabinet or building, work-space depending on the application. LPs are generally considered as optimization problems. Solving optimization problems consists of finding one or several solutions that optimizes the objectives and respects a set of constraints. Generally, each placement problem presents non-overlap and non-protrusion constraints. These constraints express the fact that there is no collision between components and each component must stay inside the container.

For most complex LPs, there is no prior information about the capacity. However, solving LPs are time consuming and designers know whether an optimal solution is accessible until the end of the optimization. Solving LPs without capacity analyzed in advance may be in vain. Many studies use density of all components as an indicator of capacity [21]. But it is not suitable when there are virtual components without mass. In [1], the author comes up with an innovative indicator to assess capacity. Actually, it's an estimation of density of all components including real and virtual items without considering their geometry. In this paper, we use a more accurate method to find the minimum space occupied by all components and provide prior information to assess the capacity of LPs.

The formulation of LPs uses either single-objective or multi-objective optimization. In single-objective optimization, finding the optimal solution corresponds to the minimum or maximum value of the objective function. On the contrary, multi-objective optimization usually solves multi-objective simultaneously without con-

verting them into single one. Therefore there is no single optimal solution but a set of compromised solutions, widely known as Pareto front [5]. For convex problems, traditional single-objective optimization can be used through weighted sum method to find the Pareto front in many individual runs if the weights are consistently modified for each run. However, non-dominance based is a generic way to find well distributed Pareto front in a single run. Moreover, the interaction with optimization processes helps the expert find the optimal solutions and select the final solution among the set of compromised alternatives.

Various methods have been developed to solve LPs. Exact approaches are widely used to solve small-sized LPs, while it's not applicable for large-sized LPs with high computational efforts and extensive memory capabilities [8]. Motivated by this, constructive heuristic and meta-heuristic methods that can provide sub-optimal solutions have been proposed. Constructive heuristic procedures build a layout from scratch by successively selecting and placing facilities until a completed layout is obtained [13]. Meta-heuristic is also an effective stochastic optimization technique. For example, Simulated Annealing (SA) [14] is single-objective optimization with provably convergence. The efficiency in solving complex combinatorial problems making it interesting for extension to multi-objective optimization [19]. Moreover, more and more researchers now work on hybrid methods by combining the global and local optimization strategies [12, 17, 18, 20].

In the practical LP applications, the conventional optimization usually takes a couple of hours or days to solve the problem which is computationally expensive. People are looking for the hybridization optimization that takes less computational efforts to find the high-quality layout designs. Constructive heuristics are widely used in parallel with other meta-heuristics to solve LPs. A genetic algorithm (GA) coupled with constructive optimization is designed [16] to do rectangular components packing where the packing sequence are encoded in a chromosome. One hybrid approach [10] combines a BRKGA, to determine the order of placement and the dimensions of each component, a constructive placement strategy, to position each facility, and a linear programming model, to fine-tune the solutions is proposed to solve the single-objective LP. In fact, GA and SA are the most popular meta-heuristics. SA was first proposed as a method for solving combinatorial optimization problems such as traveling salesman problem and knapsack problem [7]. One example that combines constructive heuristic and SA can be found in [11]. Compared to other heuristic optimizations, SA has global search ability and simpler structure with less parameters.

The LP studied here involves the real and virtual rectangular components. The space of accessibility associated with the real component is virtual, which allows the user to access the real component in reality, such as facility maintenance. Moreover, most of the LPs are NP hard. The great complexity of LPs increase the difficulty in finding a feasible layout design in a reasonable time. To resolve these problems, we contribute an efficient SA based algorithm coupled with a constructive placing strategy to solve the innovative multi-objective LPs that formulated by the real and virtual components. First, using SA based optimization to determine the placement order of components, an acceptable order is non-dominated in the corresponding objective space. Second, to explore the feasible space and guarantee non-overlap constraints, a constructive placing strategy is applied to benefit overlap of virtual components while keeping the maximal free space. It proves the algorithm can generate high quality solutions in relatively small computing efforts. In addition, the user can make the final decision through interactive visualization of the final Pareto front.

This paper is organized as follows. Sect. 2 presents the practical LP formulation. Then the assessment of the capacity is described in Sect. 3. In Sect. 4, the hybrid optimization scheme is proposed and applied to the practical LP in Sect. 5. Conclusion is given in Sect. 6.

2 Formulation of the problem

The chosen LP is taken from [4]. This case study is a shelter with four cabinets, two desks and two electrical boxes as illustrated in Fig. 1 [4]. In order to simplify the optimization problem, there are two assumptions:

1. Components have the same height and no superposition.
2. Components are cuboids.

Consequently, the configuration of the problem is defined in two-dimensions, shown in Fig. 2. We use rectangle to represent each component. Considering the accessibility, a virtual component (dotted line) is attached to each cabinet and desk. For example, the virtual space of cabinet allows interaction between human and itself or insert some materials into the cabinet. The hatched rectangle is a virtual space below air-conditioner but no cabinet can be placed. Besides, one virtual corridor located in the middle is connected to the door of the entry, which guarantees all real components should be accessible. The detailed geometry parameters of the problem can be found in [3].

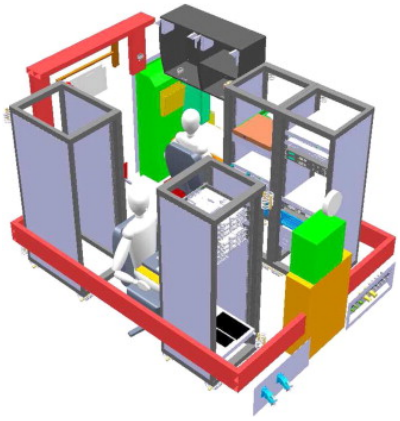


Fig. 1: Overview of CAD model [4]

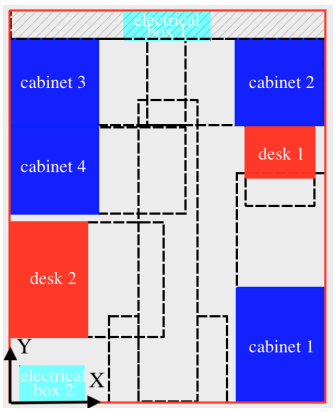


Fig. 2: 2D configuration model

The purpose of the case study is to place properly the components in the feasible space to achieve given objectives. The overall formulation of the problem including variables, constraints and objectives.

2.1 Variables

Each component $C_i = \{x_i, y_i, w_i, h_i\}, i \in n$, n is the number of components, is defined by the bottom left coordinates (x_i, y_i) and the width and height of the rectangle (w_i, h_i), shown in Fig.3. The accessibility space attached to the real component is named virtual component. This accessibility space can be considered as a set of rectangles allowing the user to access to the component or for occasional use or action of the real component. So for each component C_i, it has a list of associate n_i accessibility spaces named virtual components $v_{ij} = \{x_{v_{ij}}, y_{v_{ij}}, w_{v_{ij}}, h_{v_{ij}}\}, j \in n_i$. The virtual component v_{ij} is defined by its bottom left coordinates $(x_{v_{ij}}, y_{v_{ij}})$ in the local frame of real component and the size $(w_{v_{ij}}, h_{v_{ij}})$. Each virtual component is fixed to the real component and can be deduced in the same way by the relatives coordinates to the associates the real component. If there is rotation, the coordinates and size will update in the corresponding local frame as shown in Fig.3 (b).

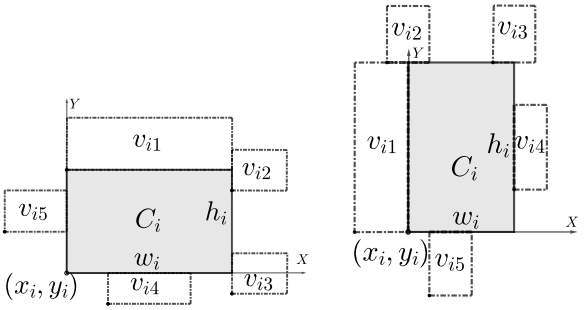


Fig. 3: Component representation

2.2 Constraints

The design constraints of the LP are non-overlap and non-protrusion constraints, including:

1. Non-overlap between real components.
2. Non-overlap between real and virtual components.
3. Non-protrusion between container and components.

Besides, there are also implicit remarks of the LP:

1. The cabinets are restricted in the allowed spaces, where non-overlap between cabinets and the free hatched space.
2. The second electrical box has to be placed against the wall near the door or the hatched space, where no rotation is allowed.

Actually, SA is usually used in unconstrained optimization. One way to extend SA for the constrained case is to use penalty method [9], where a large cost is added to the objective function for the solutions that violate constraints. However, the multiple geometric constraints makes it harder to find the feasible layout solutions through traditional optimization methods.

2.3 Objectives

There are two objective functions. The first one is to balance the mass inside the layout design by minimizing the Euclidean distance between the geometry

centre of the container and the centre of gravity of all real components calculated as follows:

$$f_1 = \sqrt{\left(X_{gra} - X'_{gra}\right)^2 + \left(Y_{gra} - Y'_{gra}\right)^2} \quad (1)$$

$$X_{gra} = \frac{\sum_{i=1}^{n}(x_{ci} \times m_i)}{\sum_{i=1}^{n} m_i}, Y_{gra} = \frac{\sum_{i=1}^{n}(y_{ci} \times m_i)}{\sum_{i=1}^{n} m_i} \quad (2)$$

where X'_{gra} and Y'_{gra} are the geometry center of the container. m_i represents the mass and (x_{ci}, y_{ci}) is the centre of gravity of the component C_i and can be deduced from the coordinates and size easily. The second objective f_2 is maximization of the Euclidean distance between the cabinet 1 (energy work), cabinet 3 and cabinet 4 and electrical box 2, in order to limit interactions between electric and energy network. A cost table is designed to measure the activity relationship between the components.

3 Capacity of layout problem

Before applying optimization algorithm to find feasible solutions of the LP, it's necessary to analyze the capacity β of the container. Based on the given dimensions of container and components, the density of the real components can be calculated as:

$$\beta = \frac{\sum_{i=1}^{n}(w_i \times h_i)}{W \times H} \quad (3)$$

Where W and H are the dimension along x-axis and y-axis of container. Since the real components cannot overlap with others, the total occupied space is equivalent to the sum of area of real components. Here, $\beta = 0.4$. However, if the virtual components (6 spaces of accessibility, 1 corridor, 1 door and 1 free space) are included and overlap between them are ignored, then $\beta = 1.05$, which means there is no feasible solution. In [1], the author use intersection matrix to calculate the capacity $\beta = 0.66$. However, the value is based on estimation and no geometry included. Hence, we extend the constructive placing method to find the minimum occupied space by packing the components.

For the placing problem, we need to generate a placing order and strategy. Suppose we have 8 real components, the number of permutations equals $8! = 40320$. Exploring all possibilities is time consuming. Besides, the main idea of placing is to maximize space utilization, in other word, to place all components as compact as possible. So we apply SA to optimize placing order and use constructive placing strategy to determine the position of components that minimizes the occupied space.

3.1 Simulated annealing

The algorithm optimizes the order X in which the components are placed into the container. X is represented as permutation of integer values in the interval $[1, n]$. In each iteration of inner loop, a new order X' will be generated by swapping elements. During the swap procedure, two components could be selected randomly based on step parameter σ. To control the performance of the swap, we define σ relating to the temperature t. When the temperature is high, σ is large, any two components can be swapped; when the temperature closes to the final temperature, only the last few components could be selected. In other words, the placing order may widely change at beginning but will converge to the optimal solution X^* finally. The optimization will stop if it reaches to the final temperature or it is up to the number of iterations. The overall idea is described in Algorithm 1.

Algorithm 1: Simulated annealing

```
Input: X, X*=X;
Output: X*;
β(X) = Place (X);
while stop condition not met do
    while iteration in inner loop do
        X'=X(σ);
        # Randomly swap two elements of X;
        β(X') = Place (X');
        δβ = β(X')-β(X);
        if δβ < 0 or Accept(δβ) then
            X = X';
            β(X)=β(X');
            if β(X') < β(X*) then
                X* = X';
            end
        end
    end
    Decrease temperature t and step σ;
end
```

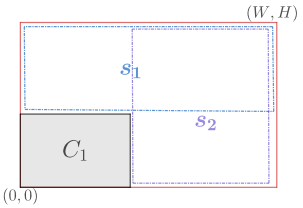
3.2 Placing strategy

The placing strategy is based on the difference process of *empty maximal spaces* (EMSs) [15]. EMSs represent the list of largest rectangle empty space in the container. In the LP, there are both real and virtual components. Considering their different properties, we use two lists of EMS for real components, real and virtual components separately, named S, S' respectively. The list S' is used for real components and guarantees non-overlap between real and virtual components while the list S is used for virtual components and benefits

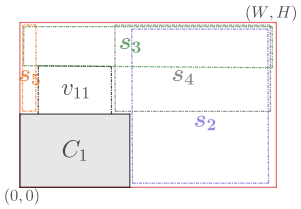
overlap between virtual components. In the following sections, we describe the space generation and the main idea of the placing strategy.

3.2.1 Space generation

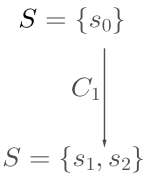
The EMS is a rectangle and defined by its vertices with lower bound and width and height along x-axis and y-axis where $s = [x_s, y_s, w_s, h_s]$. A simple example of space generation is shown in Fig. 4. At beginning, since there is no component inside the container, there is one empty space $s_0 = [0, 0, W, H]$ in S and S'. After placing the real component C_1, new empty spaces are generated. So we update the lists S and S' as $\{s_1, s_2\}$. A slicing tree illustrates the space generation in Fig.4 (c),(d). If there is a virtual component attached to the real component, for example v_{11}, then we update the list S' as $\{s_3, s_4, s_5, s_2\}$. With these two lists, we can keep all the empty spaces generated by the new components during the placing procedure.



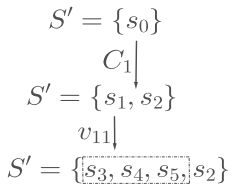
4.(a) Space generation of S



4.(b) Space generation of S'



4.(c) Tree representation of S



4.(d) Tree representation of S'

Fig. 4: Example of space generation

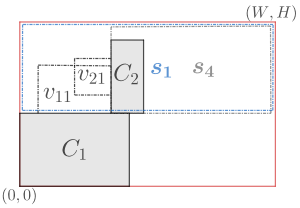
3.2.2 Placing procedure

During the previous step, there will be many available empty spaces that can be used for the new component. The constructive placing strategy and the selection of the optimal empty spaces are necessary. In order to pack the components compactly, the placement needs satisfy:

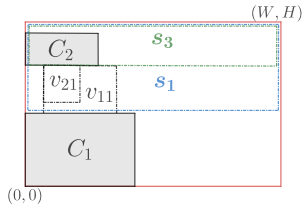
– The new real component closes to these already placed real components.

– The overlap between the new virtual component and placed virtual components should be maximized.

To place a new component into the container, there are two things to measure: the size of the empty space and the overlap of virtual space. No other constraints are taken into account except the non-protrusion. Fig.5 illustrates an example of placing two components. The detailed steps of constructive placing are described as follows:



5.(a)



5.(b)

Fig. 5: Example of placing component

Step 0 : Place the first component C_1 according to the placing order into the bottom left corner of the container and calculate all the coordinates and size of the attached virtual component v_{11} in the local frame. Update the lists S and S' (see Fig.4).

Step 1 : Place the second component C_2 and virtual components v_{21}. Enumerate all permutations of elements in the lists S' and S, for each pair of space $(s', s), s' \in S', s \in S$, check four rotations of the component and filter some pairs that not satisfy the size requirement such as $(s_5, *)$. The illustrative example in Fig.5(a) uses (s_4, s_1). Suppose the component C_2 is placed to space of s_4 that closes to the component C_1. Then check if the coordinates (x_i, y_i) are inside the space by computing the relation of the pair of space (s_4, s_1) and the size of virtual component v_{21} through Eq.4. If the pair of space (s_4, s_1) satisfies Eq.4, then we compute the placement that closes to C_1 in the space (s_4, s_1) by Eq.5 and deduce the relative coordinates of v_{21} with this configuration.

$$\begin{cases} x_{s_4} \leq x_{s_1} + max(w_{v_{21}}, x_{s_4} - x_{s_1}) + w_2 \leq x_{s_4} + w_{s_4} \\ max(0, min(y_{s_4} + h_{s_4}, y_{s_1} + h_{s_1}) - max(y_{s_4}, y_{s_1})) \geq h_{v_{21}} \end{cases} \tag{4}$$

$$\begin{cases} x_2 = x_{s_1} + max(w_{v_{21}}, x_{s_4} - x_{s_1}) \\ y_2 = max(y_{s_4}, y_{s_1}) \end{cases} \tag{5}$$

Another example with pair of space (s_3, s_1) is shown in Fig.5(b), the placement has a 90° rotation compared to

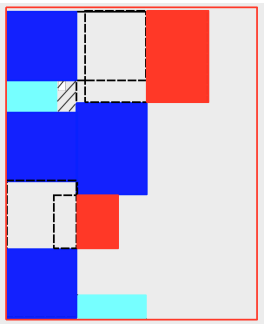


Fig. 6: The compact configuration, $\beta = 0.55$

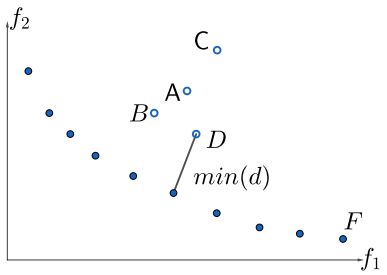


Fig. 7: Criteria to select offspring

Fig.5(a). Determining the placement is rather straightforward by swapping the size of component C_2 and following the same idea of Eq.4 and Eq.5.

In this way, the placement maximizes the overlap area between virtual space and satisfies no overlap between real and virtual components. If there is feasible solution, record the temporary placement and orientation of components and compute the occupied space of components. Select the placement that generates the largest free space, here, the placement in Fig.5(b) is prior.

Step 2 : Update the lists S and S'. Repeat placing new components until a complete layout is finished. Otherwise, marked the placing order as unfeasible.

3.3 Evaluation of capacity

For the LP that consists of real and virtual components, we define the index of capacity as a function of the empty space that evaluated by:

$$\beta = 1 - \frac{\sum S'}{W \times H} \tag{6}$$

By applying SA and placing strategy with 1000 iterations, it takes 12 minutes to find the minimum occupied space of the case study where β =0.55, the corresponding layout design is shown in Fig.6. And it is smaller than 0.66 that computed according to the intersection matrix. Besides, when we place the components sequentially, it may generate some small spaces, which are empty, but no components can be placed (hatched area in Fig.6). Indeed, these small spaces should be identified as unfeasible spaces. After obtaining the prior feasible information, we now apply optimization algorithm to find the set of feasible solutions of the LP.

4 Solving layout problem

To solve the LP, the main challenge is the multiple geometric constraints. From Sect. 3, it proves that constructive placing method can be used to separate the components, which transforms the constrained problem into unconstrained case. In this section, we propose a hybrid optimization algorithm: on the one hand, we extend SA to multi-objective optimization (MSA) to guide the search direction. The population P has N_{pop} individuals and each individual $X_k, k \in N_{pop}$ is the permutation between $[1, n]$; on the other hand, we adopt the constructive placing method to search the feasible region and find the corresponding value $\boldsymbol{f} = [f_1, f_2]$ in the objective space. The configuration of the components is determined successively by constructive placement according to the optimized order. The framework of MSA that coupled with constructive placement is shown in Algorithm 2.

4.1 Premise

In order to simplify the problem while satisfying multiple geometric constraints, we define:

1. For the hatched free space, the corridor and the door, they are all virtual components, and they are fixed during the optimization process.
2. Considering the non-overlap constraint between the cabinets and hatched free space, we assign specific constraints that restrict all cabinets below the space of hatched rectangle. While other components are constrained by the container boundary, in other word, non-protrusion.

4.2 MSA

The proposed MSA is based on non-domination criteria. During the optimization process, non-dominated points are kept in the archive F. Crowding distance

strategy [6] will be used if the size of the archive exceeds the limited number. Overall, it is similar to the original SA with significant changes in its acceptance mechanism. But, unlike in SA, there is an additional annealing loop of solutions in archive F to find more points on Pareto front. These two ideas are described as follows:

1) Acceptance mechanism :

Accepting the worse solutions allows more extensive search for the global optimal solution. For a new solution X', it has three possibilities (B, C, D) compared to the current solution $X = A$, as shown in Fig.7.

- Case B: point B dominates point A, then the new solution is better than the current solution. Update X with X'.
- Case C: point A dominates point C meaning the new solution is worse than the current solution. The probability of accepting the worse solution is specified by an acceptance probability p

$$p = e^{-(f(X')-f(X))/t} \tag{7}$$

  where t is the current temperature. The probability decreases exponentially with t. Therefore, at beginning inferior solutions are likely to be accepted but the probability will decrease as the temperature decreases.
- Case D: the new solution D and the current solution A are non-dominated to each other. It is not easy to tell which one is better according to the domination criteria only. Here, we compare them in two steps:

  (a) : Considering the F, if the new solution is not dominated by any point in F, then we consider it as a better solution.

  (b) : If it is dominated by any existing point, then we compute the Euclidean distance d between the new solution and solutions in F. Taking the minimum distance as the transition energy, we accept the new solution when it closes to the optimal Pareto front.

2) Additional loop :

In the first loop, population size $N_{pop} = 1$, we evaluate one individual at each iteration and keep all non-dominated solutions in the archive F. To enlarge the number of the final points along Pareto front, a second annealing loop is followed. Taking current non-dominated solutions as the input population $P = F$ and applying a small perturbation to the search region near current non-dominated solutions. This helps to find more alternatives.

To analyze the performance of the proposed MSA, we also apply a basic MSA with simple acceptance mechanism that mixes case B and case D together and do not use the last additional loop of the archive F. Then we apply MSA to benchmark unconstrained multi-objective functions ZDT1 and KUR. ZDT1 is a 30-variable problem with a convex Pareto-optimal set, while 3-variable problem KUR is discontinuous, asymmetric and non-convex in nature. The performance of obtained Pareto front of the basic MSA, improved MSA and comparative algorithm NSGAII are carried out using these two indicators [6]:

1. Generational distance γ:

$$\gamma = \frac{1}{N}\sqrt{\sum_{i=1}^{N} d_i^2} \tag{8}$$

   where N is the number of non-dominated points, d_i measures distance between each point and the nearest point in Pareto front. It measures the convergence of the algorithm. The smaller the value of γ, the better convergence.
2. Diversity metric Δ:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left|d_i - \bar{d}\right|}{d_f + d_l + (N-1)\bar{d}} \tag{9}$$

   where d_f and d_l measure how far that boundary solutions away from extreme solutions. $\bar{d}$ is the average of all distance d_i between consecutive points. It reveals the diversity among the optimal solutions. The smaller the value of Δ, the better uniformity of the distribution.

The parameters γ and Δ are computed over five independent runs for MSA and NSGAII. The results obtained in the form of mean and standard deviation (SD) are shown in Table 1. Basic MSA is able to converge to the convex function ZDT1 but not in the non-convex function KUR, where the improved MSA found much better convergence. In general, KUR is difficult for solving because very small changes in the variables can cause big differences in the objective space. By comparing the results, it proves that the additional loop and new acceptance mechanism improve the performance of obtained solutions. It is also observed that in both cases, the improved MSA has better convergence while the population based algorithm NSGAII has superior diversity.

Table 1: Comparative results of proposed MSA and NSGAII

Function	Indicator	NSGAII		Basic MSA		Improved MSA	
		Mean	SD	Mean	SD	Mean	SD
ZDT1	γ	0.00578	0.00043	0.00503	0.00025	0.00492	0.00035
	Δ	0.35813	0.02482	0.37962	0.03508	0.37341	0.01133
KUR	γ	0.01002	0.00089	0.04161	0.00051	0.00876	0.00019
	Δ	0.40488	0.00871	0.66883	0.04108	0.43079	0.01421

Algorithm 2: Multi-objective Simulated annealing

```
Input: P = X_1, X_2, ...X_{N_pop};
Output: F;
F ∪ P #Initialize archive;
f = Place(P);
while stop condition not met do
    while iteration in inner loop do
        P'=P(σ);
        #Generate new population based on step
          parameter σ;
        F := Pareto(F ∪ P');
        f' = Place(P');
        while i < N_pop do
            if f_i dominates f'_i then
                δ = f'_i − f_i;
                if rand < e^{−δ/t} then
                    X_i = X'_i, f_i = f'_i;
                end
            else
                if f'_i dominates f_i then
                    X_i = X'_i, f_i = f'_i;
                else
                    d = dis(F, f'_i) ;
                    if rand < e^{−min(d)/t} then
                        X_i = X'_i, f_i = f'_i;
                    end
                end
            end
        end
    end
    Decrease temperature t and step σ;
end
```

4.3 Constructive placing

The placement is determined by applying a convention and a selection criteria during the placing process. To place the component properly, we enumerate the elements in the lists S and S'. Then we apply the boundary convention to determine the configuration of the component and select the one with maximal empty space. The constructive placing is formulated as follows:

Step 0 : Place the first component into one corner of the container then update S and S'.

Step 1 : Place the new component sequentially according to the placing order. If the pair of space (s', s) satisfies the size requirement for the component C_i, then we go through 4-way orientations of C_i to find all the feasible configurations by Eq.10 to Eq.13. $\{x_{a_i}, y_{a_i}, w_{a_i}, h_{a_i}\}$ defines a rectangle of the allowed space of C_i. By default, the placement of the component always closes to the boundary of the selected empty space $min(s', s)$ or $max(s', s)$. The placement follows the boundary convention and will generate new empty spaces with less margin.

$$\begin{aligned} max(0, min(min(x_{s'} + w_{s'}, x_{a_i} + w_{a_i}) - w_i, \\ x_s + w_s) - max(min(x_{s'} + w_{s'}, \\ x_{a_i} + w_{a_i}) - w_i - w_{v_{ij}}, x_s)) \geq w_{v_{ij}} \end{aligned} \tag{10}$$

$$\begin{aligned} max(0, min(max(x_{s'}, x_{a_i}) + w_i + w_{v_{ij}}, x_s + w_s) \\ - max(max(x_{s'}, x_{a_i}) + w_i, x_s)) \geq w_{v_{ij}} \end{aligned} \tag{11}$$

$$\begin{aligned} max(0, min(min(y_{s'} + h_{s'}, y_{a_i} + h_{a_i}) - h_i, \\ y_s + h_s) - max(min(y_{s'} + h_{s'}, \\ y_{a_i} + h_{a_i}) - h_i - h_{v_{ij}}, y_s)) \geq h_{v_{ij}} \end{aligned} \tag{12}$$

$$\begin{aligned} max(0, min(max(y_{s'}, y_{a_i}) + h_i + h_{v_{ij}}, y_s + h_s) \\ - max(max(y_{s'}, y_{a_i}) + h_i, y_s)) \geq h_{v_{ij}} \end{aligned} \tag{13}$$

If the component C_i and virtual components v_{ij} satisfy one of the above conditions under a certain configuration in the space pair (s', s), then the space is applicable. If there is feasible solution, record the temporary placement and orientation of components. If one component has several feasible placements, the one with maximal empty space will be selected as the prior placement.

Step 2 : Update the lists S and S'. Repeat placing new components until a complete layout is finished. Otherwise, marked the placing order as unfeasible.

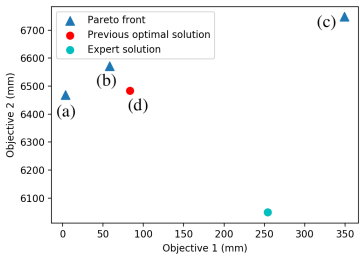


Fig. 8: Display of Pareto front

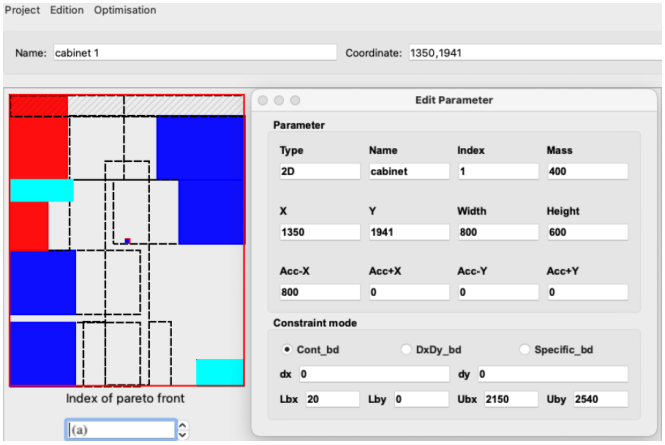
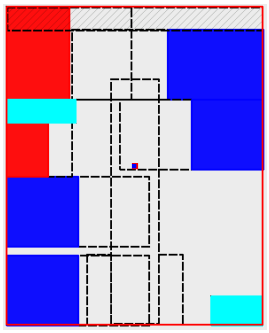
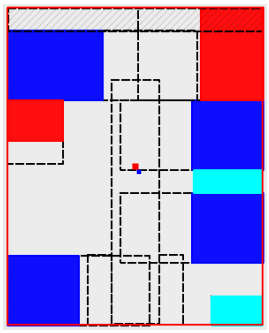


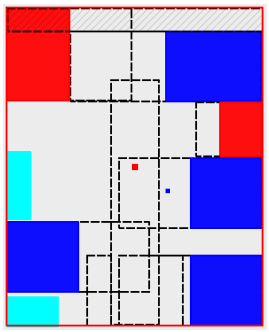
Fig. 9: Interactive display of Pareto-optimal designs



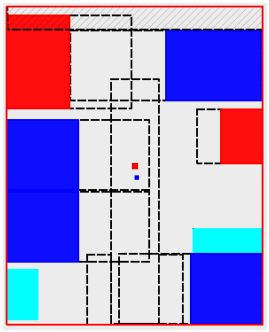
10.(a)



10.(b)



10.(c)



10.(d)

Fig. 10: Display of Pareto-optimal designs

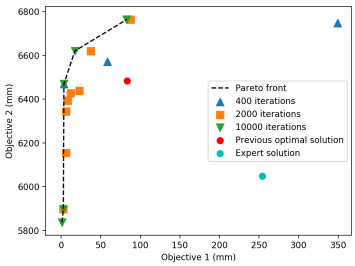


Fig. 11: Evolution of Pareto front

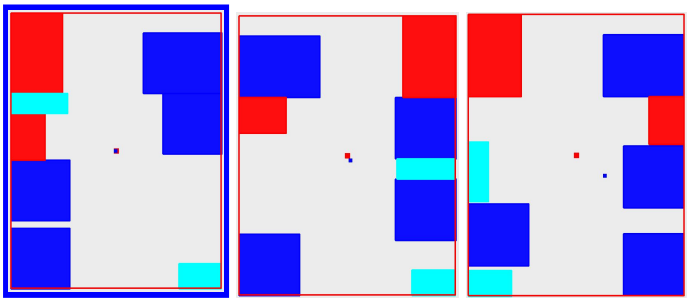
5 Application

Based on the previous analysis, now we test the proposed algorithm in the industrial application. The hybrid optimization algorithm is programmed as a Python class and the interactive environments are developed based on the PyQt5 package. The obtained Pareto front is shown in Fig.8. It takes 5 minutes to finish 400 iterations, which is very efficient compared to algorithms that take hours or days. The interactive display of designs in Fig.9 helps the user evaluate performance of the designs by locally manipulating or modifying the information of the component and guarantees the interaction in time. Besides, in order to help the user perform the fine optimization manually, different constraint modes are properly defined:

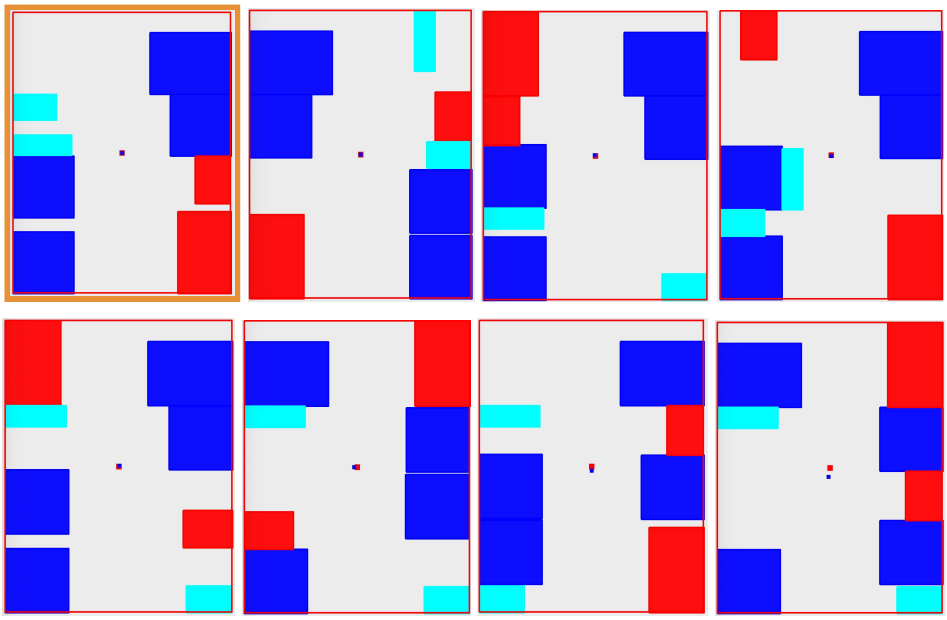
1. Cont_bd represents the container non-protrusion constraint where the component can be anywhere inside the container.
2. DxDy_bd defines the distance that the component can be moved from the current position.
3. Specific_bd stands for the rectangle space in which the component can be located.

The corresponding layout designs are displayed in Fig.10 (a) to (c). The red and blue points in each layout configuration represent the geometry center of container and centre of gravity of all components. The configuration of the expert solution is described in Fig.2. And the previous optimal solution based on interactive modular optimization is presented in [2], the corresponding layout is shown in Fig.10 (d). All layout designs satisfy the non-overlap, non-protrusion and the additional user defined constraints. And the obtained non-dominated solutions are better compared to the initial and previous optimal solutions.

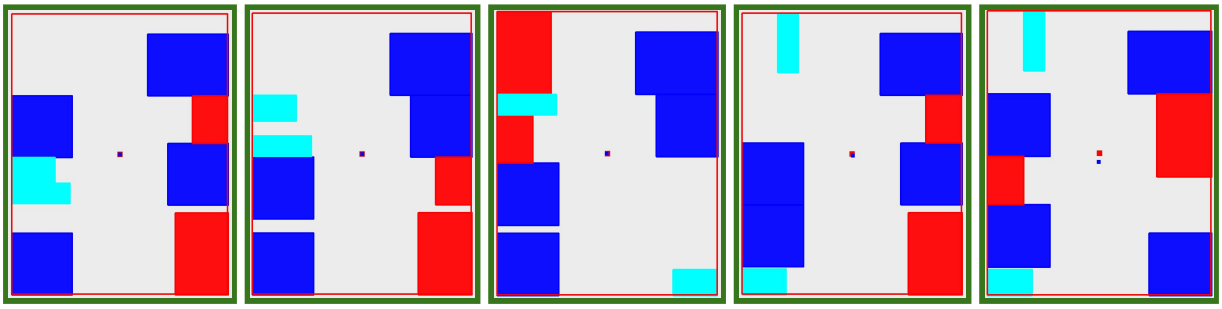
Fig.11 illustrates the evolution of Pareto front obtained by the optimization algorithm and the corresponding layout designs are shown in Fig.12. It can be observed that in the first 400 iterations , the algorithm could find some feasible solutions efficiently. By display-



Number of iterations = 400



Number of iterations = 2000



Number of iterations = 10000

Fig. 12: Evolution of Pareto-optimal designs

ing the obtained Pareto front, the expert could interact with the solutions and make the final choice. The constructive placing strategy splits the space to evaluate the overlap and transforms the design variables into discrete, here, the number of variables equals 8, which highly reduces the optimization complexity of MSA.

6 Conclusion

LPs arise more and more attention in the industrial field. The research shows that optimal layout design will improve production efficiency. However, the complexity of the LP is related to the problem formulation, including the number of variables, constraints and objectives.

Regarding the virtual components of LP, first of all, we define a new indicator β to formulate the capability of the LP. The proposed indicator is based on the empty space computation. The minimum packing area of all components provides a prior information about the density. If β is larger than 1, the LP could not be solved properly.

The application study in this paper containing both real and virtual components that increases the complexity. Hence, a hybrid multi-objective simulated annealing algorithm is presented for solving complex LPs. To explore the feasible layout space, simulated annealing is proposed to determine the order of placement; then, a constructive placing strategy based on maximal space is developed. The proposed algorithm is the first attempt to directly search the feasible space of multi-objective layout design. The results prove the interactive optimization efficiency and performance of the proposed algorithm.

Acknowledgements The authors would like to acknowledge THALES for the application study.

References

1. Bénabès, J., Guédas, B., Poirson, E., Bennis, F.: Indicator of feasibility for layout problems. In: Int.Des.Eng.Tech.Conf.(IDETC) and Comput.Information.Eng.Conf. (CIE), pp. 727–734. Chicago, Illinois, United States (2012)
2. Bénabès, J., Poirson, E., Bennis, F., Ravaut, Y.: Interactive modular optimization strategy for layout problems. In: Int.Des.Eng.Tech.Conf.(IDETC) and Comput.Information.Eng.Conf. (CIE), pp. 553–562. Washington, DC, United States (2011)
3. Bénabès, J., Bennis, F., Poirson, E., Ravaut, Y.: Interactive optimization strategies for layout problems. Int.J.Interact.Des. Manuf. **4**, 181–190 (2010). DOI 10.1007/s12008-010-0100-x
4. Bénabès, J., Poirson, E., Bennis, F.: Integrated and interactive method for solving layout optimization problems. Expert Systems with Applications **40**, 5796 – 5803 (2013). DOI 10.1016/j.eswa.2013.03.045
5. Chankong, V., Haimes, Y.Y.: Multiobjective decision making: theory and methodology. Courier Dover Publications (2008)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE.Transactions.Evolut.Comput. **6**, 182–197 (2002). DOI 10.1109/4235.996017
7. Delahaye, D., Chaimatanan, S., Mongeau, M.: Simulated annealing: From basics to applications. pp. 1–35.ISBN 978–3–319–91085–7. Springer (2019). DOI 10.1007/978-3-319-91086-4_1
8. Foulds, L.R., Hamacher, H.W.: Facilities layout problems, pp. 975–979. Springer US, Boston, MA (2009). DOI 10.1007/978-0-387-74759-0_171
9. Freund, R.M.: Penalty and barrier methods for constrained optimization. Lecture Notes, Massachusetts Institute of Technology (2004)
10. Gonçalves, J.F., Resende, M.G.: A biased random-key genetic algorithm for the unequal area facility layout problem. Eur.J.Ope.Res. **246**, 86–107 (2015)
11. Hanafi, R., Kozan, E.: A hybrid constructive heuristic and simulated annealing for railway crew scheduling. Comput.Ind.Eng **70** (2014). DOI 10.1016/j.cie.2014.01.002
12. Hasda, R., Bhattacharjya, R., Bennis, F.: Modified genetic algorithms for solving facility layout problems. Int.J.Interact.Des. Manuf. **11**, 713–725 (2016). DOI 10.1007/s12008-016-0362-z
13. Hosseini nasab, H., Fereidouni, S., Ghomi, S., Fakhrzad, M.: Classification of facility layout problems: a review study. Int.J.Adv.Manuf.Technol. **94**, 957–977 (2018). DOI 10.1007/s00170-017-0895-8
14. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Sci. **220**, 671–680 (1983)
15. Lai, K., Chan, J.W.: Developing a simulated annealing algorithm for the cutting stock problem. Comput.Ind.Eng. **32**, 115–127 (1997)
16. Li, X., Zhao, Z., Zhang, K.: A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins. IIE.Annual.Conf. and Expo pp. 2039–2048 (2014)
17. Mariem, B., Marc, Z., AFFONSO, R., Masmoudi, F., Haddar, M.: A methodology for solving facility layout problem considering barriers – genetic algorithm coupled with A* search. J.Intell.Manuf. **31**, 615–640 (2019). DOI 10.1007/s10845-019-01468-x
18. Méndez, M., Rossit, D.A., González, B., Frutos, M.: Proposal and comparative study of evolutionary algorithms for optimum design of a gear system. IEEE.Access. **8**, 3482–3497 (2020)
19. Pllana, S., Memeti, S., Kolodziej, J.: Customizing pareto simulated annealing for multi-objective optimization of control cabinet layout. 22nd International Conference on Control Systems and Computer Science (CSCS) pp. 78–85 (2019). DOI 10.1109/CSCS.2019.00021
20. Ripon, K.S.N., Glette, K., Khan, K.N., Hovin, M., Torresen, J.: Adaptive variable neighborhood search for solving multi-objective facility layout problems with unequal area facilities. Swarm.Evolut.Comput. **8**, 1–12 (2013). DOI 10.1016/j.swevo.2012.07.003
21. Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems. Eur.J.Oper.Res. **183**, 1109 – 1130 (2007). DOI 10.1016/j.ejor.2005.12.047