Layout optimization based on multi-objective interactive approach

Xiaoxiao Song

THÈSE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES

ÉCOLE DOCTORALE N° 602
Sciences pour l'Ingénieur
Spécialité : *Robotique-Mécanique*

Par
Xiaoxiao SONG

Layout optimization based on multi-objective interactive approach

Thèse présentée et soutenue à Ecole Centrale de Nantes, le 22 Novembre 2022
Unité de recherche : UMR 6004, Laboratoire des Sciences du Numérique de Nantes (LS2N)

Rapporteurs avant soutenance :

Xavier FISCHER	Professeur, ESTIA, Bidart
Bernard YANNOU	Professeur des universités, CentraleSupélec Université Paris-Saclay

Composition du Jury :

Président :	Eduardo SOUZA DE CURSI	Professeur des universités, INSA Rouen Normandie
Examinateur :	Yannick RAVAUT	Expert intégration et modélisation mécanique, Thales SIX France, Cholet
Dir. de thèse :	Fouad BENNIS	Professeur des universités, École Centrale de Nantes
Co-dir. de thèse :	Emilie POIRSON	Professeure des universités, École Centrale de Nantes

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciation to my supervisor, Professor Fouad Bennis, for his guidance, encouragement, constructive critiques, and patience throughout this research. He has been a great mentor and guide to me. This thesis would not have been possible without him.

I would also like to sincerely thank my thesis co-supervisor Emilie Poirson. I am very grateful for her valuable comments, insights, enthusiasm and the involvement throughout these three years of doctoral studies.

This PhD work would not have been so concrete without industrial application cases. These case studies, proposed by Yannick Ravaut, engineers from the company Thales SIX France in Cholet, greatly helped me when it was necessary to materialize and validate the different concepts developed in this thesis.

A big thank you to the colleagues that I was able to meet during this doctorate, whether in my research activities within the REV team of the LS2N, in my teaching activities at the Ecole Centrale de Nantes.

I would like to thank the members of the jury too. Your remarks, criticisms and advice have been very useful to me for the outcome of this thesis.

Above all, I warmly thank my friends and my parents who encouraged me in these years. A big thanks to my boyfriend, with whom I shed sweat, blood and tears in the trenches of sciences, thanks for always being there.

听我说，谢谢你，因为有你，温暖了四季 ....

Acronymes

SA	Simulated Annealing
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
KKT	Karush-Kuhn-Tucker
MOGA	Multi-objective genetic algorithm
MOSA	Multi-objective simulated annealing
FPA	Flower pollination algorithm
SPEA	Strength Pareto evolutionary algorithm
MOPSO	Multi-objective particle swarm optimization
MSSA	Multi-objective salp swarm algorithm
NSGA II	Non-dominated sorting genetic algorithm II
SOS	Symbiotic organisms search
VEGA	Vector evaluated genetic algorithm
MOEA/D	Multi-objective evolutionary algorithm based on decomposition
NSGA III	Non-dominated sorting genetic algorithm III
MOEA/DD	Multi-objective evolutionary algorithm based on dominance and decomposition
IBEA	Indicator-based evolutionary algorithm
SMS-EMOA	s-metric selection evolutionary multi-objective optimization algorithm
DWU	Dominance-weighted uniformity
GD	Generational distance
GD-MOEA	Generational distance-based multi-objective evolutionary algorithm
CDAS	Controlling dominance area of solutions
SDR	Strengthened dominance relation
SQA	Simplified quadratic approximation
QAP	Quadratic assignment problem
MIP	Mixed integer programming
LP	Linear programming
NLP	Nonlinear programming
IP	Integer programming

MINLP	Mixed integer nonlinear programming
FLP	Facility layout problem
DSO	Dynamic space ordering
DFLP	Dynamic facility layout problem
DEA	Data envelopment analysis
PSA	Pareto simulated annealing
BRKGA	Biased random-key genetic algorithm
SCHN1	Schaffer function N1
SCHN2	Schaffer function N2
POL	Poloni function
QUAD	Quadratic function
FON	Fonseca function
CT	Computational time
SD	Standard deviation
MOKP	Multi-objective knapsack problem

NOTATIONS

$\boldsymbol{x}$	Variables
$\boldsymbol{f}$	Objectives
$\boldsymbol{g}$	Inequality constraints
$\boldsymbol{h}$	Equality constraints
P	Current population
N	Population size
P'	New population
(LB, UB)	Global boundaries
(lb, ub)	Local boundaries
τ	Interval parameter
$\mathcal{A}$	Archive
CD	Crowing distance
M	Archive size
t	Temperature
R	Rank of current population
R'	Rank of new population
t_f	Final temperature
t_0	Initial temperature
Δ	Diversity metric
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_{sv}	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
a_{ik}	Overlap between s_i and s_k
a_{kj}	Overlap between s_k and v_{ij}
a_{ij}	Overlap between s_i and v_{kj}
A_{ik}	Overlap between c_i and c_k
$\boldsymbol{a}$	Available Space of virtual components
$\boldsymbol{a'}$	Available Space of solid components
$\boldsymbol{c}$	Component placement order
$\alpha_{x_{ij}}$	Relative position of x-coordinate between component c_i and c_j
$\alpha_{y_{ij}}$	Relative position of y-coordinate between component c_i and c_j
(w_r, h_r)	Accessible space required by the user
$\boldsymbol{p}$	Component configuration sequence
I	Infeasible violation
z	Container area
l	Number of sub-containers
r	Assignments of components
$\hat{\beta}_c$	Minimum occupied space

List of Figures

List of Tables

Table of Contents

Part I

English version

INTRODUCTION

Problem statement

The layout problems are inherently multidisciplinary tasks. The applications can be the space radiator design [1], the chip layout design [2], the vehicle layout design [3], the architecture layout design [4], the manufacturing systems layout design [5] and so on. Excellent layout design can effectively improve the system performance. The problems are generally 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 [6] or determined area [7]. No component overlap and no container protrusion are the common geometrical constraints, while orientation or alignment is to define functional relationships between components [8]. 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 [9], or the adjacency requirement [10] and the material handling cost [11] in facility layouts.

Without optimization techniques, the layout problem of the industrial environment cannot be solved successfully. However, 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 [12]. 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. Moreover, the optimization process can be used as decision support for the designer. When faced with multiple choices under risk and uncertainty, the decision-maker

may select the layout design to achieve the best compromise of system performance.

Thesis objectives

The thesis objectives are described below:

1. Understand the main difficulties confronted with layout optimization.
2. Design a generic layout model and interactive environment.
3. Develop an effective multi-objective layout optimization approach.
4. Extend the application to more complex layout problems.

Organization and contribution of the thesis

The following is a summary of our main contributions:

1. A population-based simulated annealing algorithm for multi-objective problem without using an external archive is proposed. The dynamic selection preserves the non-domination and distributed solutions in the population. It is a simple-structured algorithm with the advantage of ease of implementation.
2. A new layout model addressing accessibility is presented. The novel layout model consists of novel components definition where virtual spaces associated with solid components represents the accessibility of component. Accessibility is an important functional requirement in the field of layout design.
3. Two indicators are defined for layout optimization: capacity index measures the optimization difficulty and provides a priori information on the layout optimization feasibility; similarity indicator evaluates the similarity of obtained layout alternatives and helps the expert select the final decision.
4. An efficient placement algorithm for the layout configuration construction is developed. The placement, not only guarantees non-overlap of components, but also introduces the idea of connection path ensuring accessibility of components. It circumvents the difficulty arising from the designed constraints and generates more alternatives than comparative optimizers.
5. A multi-objective layout optimizer that integrates accessibility analysis within population-based simulated annealing method is proposed to conduct the opti-

mization in the feasible alternatives that respect all constraints. It demonstrates the effectiveness and portability of the proposed layout optimization algorithm.

The organization of the thesis is as follows:

Chapter 1 Provide a literature review of multi-objective optimization algorithms and optimization of layout problems. It describes the fundamental concepts, the classical optimization approaches and the archive effects, followed by details on strategies that try to improve the convergence and diversity in multi-objective optimization. The following sections introduce the layout problem, including the representation, the formulation, and the optimization approaches.

Chapter 2 Study the population-based simulation annealing algorithm considering the external archive for the multi-objective problem. We explained the general framework and investigated the performances of archive-based and archive-free cases. The comparison is carried out on continuous and combinatorial benchmark instances with the known multi-objective optimization method. In particular, the convergence resistance and improvement is further studied.

Chapter 3 Present a new layout model that takes into account component accessibility and designed interaction tools for layout problems. First of all, the component definition, design constraints and objectives are detailed. Then a capacity indicator is proposed to evaluate the layout optimization difficulty. Finally, an interactive environment which allows integrating mathematical optimization is presented, with a focus on similarity analysis applied to obtained layout alternatives.

Chapter 4 Describe our contribution to multi-objective layout optimization that integrates accessibility analysis within simulated annealing method. The accessibility analysis is conducted by the constructive placement. The placement procedure and the placement strategy comparisons are described in detail. After that, the ideas of multi-objective layout optimization with the proposed multi-objective simulated annealing and placement is presented whereas the optimization of multi-container layout is then detailed.

Chapter 5 Apply the proposed approach to several industrial layout problems and discuss the optimization results. For each application, all the steps of the method are

introduced: the problem description, the capacity analysis, the resolution of the problem, as well as the similarity analysis for later interaction. A general conclusion can be deduced on the optimization performance that efficient placement satisfying constraints will lead to an effective optimizer and enable a truly interactive optimization process.

Chapter 1

Literature review

1.1 Introduction

Mathematically speaking, optimization aims to find the solution, i.e. one or multiple decision variables, by maximizing or minimizing the given function(s) while respecting a number of constraints. Thus, two kinds of optimization problems are classified in the literature: single-objective and multi-objective optimization problems. In the past, deterministic optimization, for example, gradient-based, was the most commonly used technique. Gradient-based methods start with an initial point and search near the solution space based on the gradient information. It proves that gradient-based methods converge faster compared to stochastic approaches but cannot easily solve non-convex cases. Stochastic approaches are suitable for global search because they are able to explore and find promising solutions with reasonable computational time in the search region. These algorithms demonstrate high performance for practical optimization applications in different areas e.g. layout problems.

The Cutting & Packing (C & P) problem is common in the industry. A cutting problem aims to maximize the number of placed components inside the container and a packing problem refers to minimizing the number of used containers to place all components. In a layout problem, the components representation, the objectives, and the constraints may be different. For example, the components are geometrically connected in a C & P problem whereas the components placement functional connected in a layout problem. Layout problems refer to finding optimal arrangements of several components in the container area. The main procedure of a layout problem usually starts with the problem representation, followed by problem formulation, then problem optimization. The designer defines the problem description (data and requirements). The information will be translated into problem formulation and output as a model solved by optimization techniques. Various methods have been developed to solve layout problems, which can be divided into two broad categories i.e. exact and meta-heuristic approaches. This chapter

provides a comprehensive review of completed and under-explored research on the layout problems and optimization.

1.2 Multi-objective optimization

1.2.1 From single- to multi- objective problem

In practical optimization applications, usually a single objective or multiple objectives need to be optimized. In the single-objective problem, only one objective function is taken into account (assuming minimization). It could be expressed as follows:

$$\begin{cases} variable & \boldsymbol{x} = (x_1, x_2 ... x_n)^T \\ min & f(\boldsymbol{x}) \\ s.t. & \boldsymbol{g}(\boldsymbol{x}) \leq 0, \boldsymbol{h}(\boldsymbol{x}) = 0 \end{cases} \tag{1.1}$$

where the objective f is a function of n-dimensional variable $\boldsymbol{x}$, $\boldsymbol{g}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x})$ are inequality and equality constraints. A global optimal solution achieves the minimum objective value, as shown in Fig. 1.1.

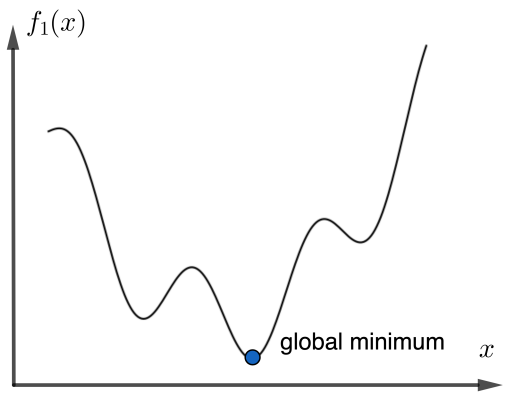


Figure 1.1 – Single-objective single variable minimization $f_1(x)$.

The multi-objective problems involve more than one goals, which are usually conflict-

ing. The multi-objective problem is defined by:

$$\begin{cases} variable & \boldsymbol{x} = (x_1, x_2 ... x_n)^T \\ min & \boldsymbol{f}(\boldsymbol{x}) = f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ... f_k(\boldsymbol{x}) \\ s.t. & \boldsymbol{g}(\boldsymbol{x}) \leq 0, \boldsymbol{h}(\boldsymbol{x}) = 0 \end{cases} \tag{1.2}$$

where $\boldsymbol{f}$ is the vector of k objective functions to minimize (or maximize). Single-objective optimization aims to find the best solution that achieves the maximum or minimum objective value. On the contrary, multi-objective optimization refers to finding multiple compromised solutions rather than a single optimal solution, widely known as Pareto front. Fig.1.2 illustrates a multi-objective problem of bi-dimensional variables and two objective functions, where the feasible design space is projected onto the corresponding objective space. Similarly, if one of the objectives aims to maximize, Fig. 1.3 presented the possible shape of the corresponding Pareto front.

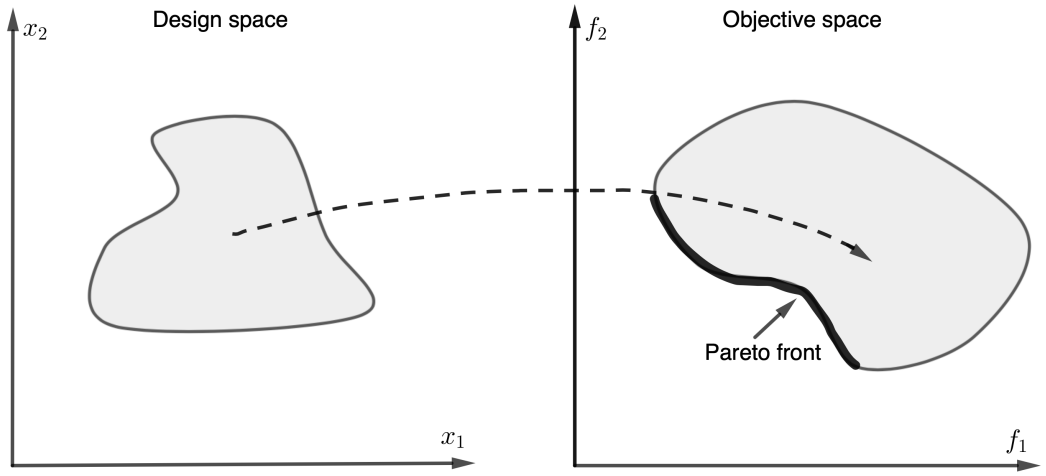


Figure 1.2 – Multi-objective bi-dimensional variable (left) minimization $(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ (right).

1.2.2 Resolution methods

In single-objective optimization, different optimization methods, including deterministic and stochastic, are available in the literature.

- Deterministic methods have no random behaviour and find the same solutions under the same conditions, for example, gradient-based methods. They are reliable

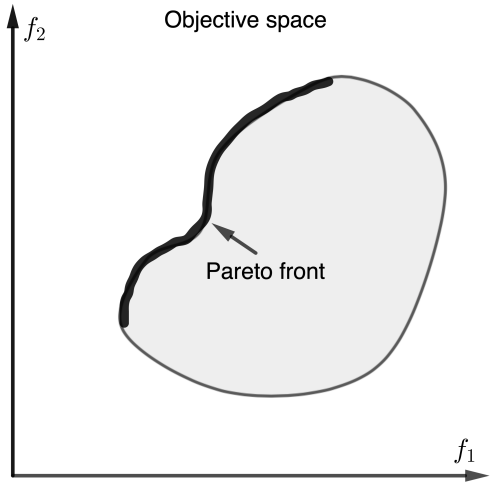


Figure 1.3 – Multi-objective bi-dimensional variable minimization $(f_1(\boldsymbol{x}), -f_2(\boldsymbol{x}))$.

but may converge to a local optimum (depending on the start point), as shown in Fig. 1.4.

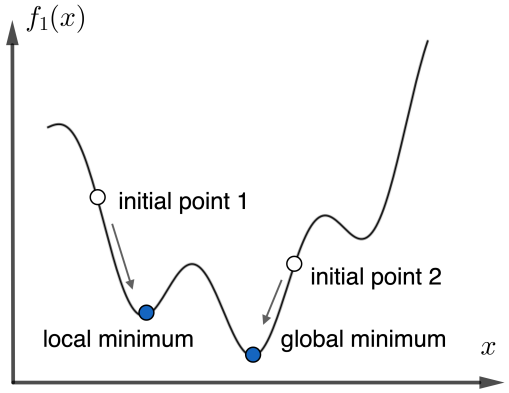
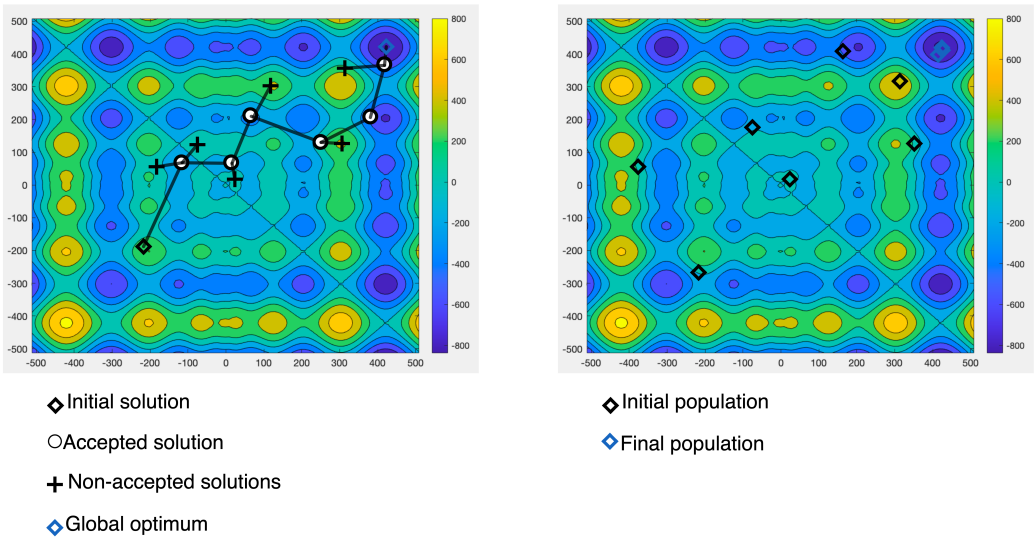


Figure 1.4 – Gradient-based optimization of single-objective single variable.

- Stochastic methods benefit from random behaviour and find different solutions, even if they start from the same start point. The stochastic operators i.e. crossover and mutation of genetic algorithms facilitate avoiding local optima. These methods include, among others, Simulated Annealing, Genetic Algorithm and Particle Swarm Optimization:

  Simulated Annealing: SA originates from the phenomenon of crystallization

and is the representative of individual-based approach [13]. It begins with a random solution and generates the new neighborhood solution. Accept solutions with decreasing cost, while accepting solutions with increasing cost with probability defined by the cost and temperature, which enables uphill moves. Fig. 1.5(a) illustrates the principle of SA optimization process.



(a) SA optimization (b) GA optimization

Figure 1.5 – SA and GA optimization principle.

Genetic Algorithm: GA is one popular population-based evolutionary method [14]. Based on the concept of the best survival, it imitates the Darwinian evolution mechanism to achieve sub-optimal solutions through crossover and mutation operators. Fig. 1.5(b) illustrates the theory of GA optimization process.

Particle Swarm Optimization: PSO is a swarm-based algorithm that simulates the collective behaviour of birds in navigating and hunting [15]. It randomly initializes multiple particles over the search space and dynamically adjusts particles motions based on itself and other particles to find the global best position after a number of iterations.

Regardless of the differences between evolutionary and swarm-based techniques, they both improve a set of solutions during optimization process. If an algorithm

improves only single solution, it is called an individual-based algorithm. If a set of solutions is improved, it is referred as a population-based algorithm. Individual-based algorithms are advantageous due to the small number of function evaluations and the simplicity of the overall optimization process. However, the probability of local optima is very high. Population-based algorithms are capable of avoiding local solutions and exchanging information about the search space.

The multi-objective problems could be resolved via single-objective optimization techniques and multi-objective optimization techniques. Ideally, they aim to find multiple solutions that are well-distributed and close to the theoretical Pareto front, namely diversity and convergence. To achieve these goals, various approaches have been developed that could be classified into the followings groups:

- Based on scalarization
- Based on Pareto domination
- Based on decomposition
- Based on indicator
- Based on interaction

Based on scalarization

The scalarization transforms a multi-objective problem into an equivalent single-objective problem. The weighted sum approach is one of the most intuitive ways to perform the transformation. The formulation is given below:

$$\begin{cases} \min & \sum_{i=1}^{k} w_i f_i(\boldsymbol{x}) \\ s.t. & \sum_{i=1}^{k} w_i = 1 \end{cases} \tag{1.3}$$

where w_i is the weight assigned to each objective function $f_i, i = 1, 2, ..., k$, which reflects the designer's preference. Varying the set of w_i will generate different problem formulation, that is, corresponding to different optimal solution. Considering the scale difference among objective functions, it is necessary to use a non-dimensional transformation:

$$\begin{cases} \min & \sum_{i=1}^{k} w_i f_i'(\boldsymbol{x}) \\ s.t. & \sum_{i=1}^{k} w_i = 1 \quad f_i'(\boldsymbol{x}) = \frac{f_i(\boldsymbol{x}) - f_i^{\min}}{f_i^{\max} - f_i^{\min}} \end{cases} \tag{1.4}$$

where $f_i^{\min}$ and $f_i^{\max}$ are the minimum and maximum values of i-th objective function. The normalization transforms the objective function into an interval between zero and one. With enough weight combinations, it is possible to approximate the true Pareto front with consistent weights modifications in each run. The equivalent single-objective can be solved by any single-objective optimization algorithms. Some of the literature that uses the weighted sum method is reviewed below.

Gradient method uses the negative gradient direction of the objective function as the search direction, also known as the steepest descending method. Similarly for multi-objective optimality, a necessary condition is defined using the objective gradients, widely known as the Karush-Kuhn-Tucker (KKT) condition. A gradient-based method was developed to efficiently track the Pareto front in bi-objective problems [16]. There are two steps: the first is named go-to-Pareto, which optimizes the weighted sum objective function to find one start point on the Pareto front; the second is called move-on-Pareto, which moves a point on Pareto front to an infeasible solution reducing one objective while maintaining the other objectives, and then minimizes the distance using the predicted point that satisfy Karush–Kuhn–Tucker (KKT) condition as a good initial point to find a new solution on Pareto front. In gradient-based optimization, there is high possibility that stuck in a local optima because the gradient at any local optimal is zero.

A multi-objective genetic algorithm (MOGA) uses the weighted sum of objective function values to select individuals [17]. For each selection, the weights assignment to the objective functions are random rather than constant. It is relatively simple and efficient, but the performance is highly dependent on the weight distribution.

Similarly, a multi-objective simulated annealing (MOSA) was developed which transforms the objectives into a scalar function [18]. The scalar function, defined by the weighted sum objective function value, is minimized by the scalar SA optimizer.

The recently developed flower pollination algorithm (FPA) mimics the evolution of flower pollination. It has been extended to solve multi-objective optimization problems [19]. The proposed method is used to solve a set of multi-objective test functions, and the weighted sum with random weights shows that the FPA with fast convergence is effective.

Nevertheless, a uniform distribution of weights rarely results in a good distribution of solutions. For non-convex problems, continuously varying the weights may not necessarily lead to a uniform distribution of points along Pareto front [20]. One multi-objective problem is presented in Fig. 1.6. The set of compromised points forms the Pareto front where the objective minimizes $(f_1(\boldsymbol{x}), -f_2(\boldsymbol{x}))$. And it may fail in finding the non-convex

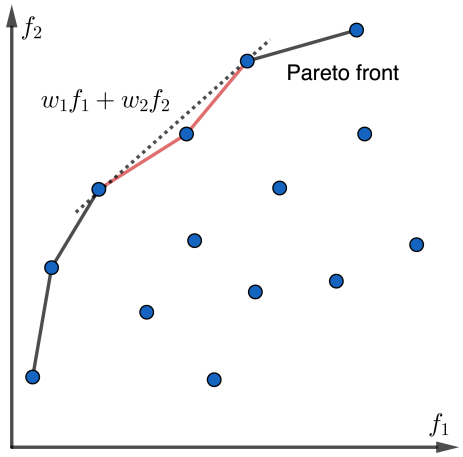


Figure 1.6 – Example of weighted sum approach $(f_1(\boldsymbol{x}), -f_2(\boldsymbol{x}))$.

Pareto front (in red) in the objective space.

ε- constraint method was presented as another single-objective optimization by considering the rest objective functions as constraints [21]. The general formulation is:

$$\begin{cases} \min & f_i(\boldsymbol{x}) \\ s.t. & f_j(\boldsymbol{x}) \leq \varepsilon_j, \quad j = 1, 2, ..., k, \quad \forall j \neq i \end{cases} \tag{1.5}$$

where ε_j is a pre-defined parameter. It expresses the fact that the j-th objective function value should be smaller than ε_j during the optimization process. Fig. 1.7 shows a two-dimensional representation of the ε-constraint method. The combination of selected objective and constraints will give different optimal solutions. It can be used for both convex and non-convex problems. The Pareto front can be obtained by systematic variation of ε_j. However, improper ε_j can result in a problem formulation with no feasible solution.

Thus, an alternative approach divided the objectives into two groups: control group contains only one selected function, performance group collects the rest [22]. The proposed method is based on a priori information, i.e. which objective should be selected as a control function. To apply the proposed method, the performance functions, is replaced by the KKT necessary condition. The single-objective problem optimizes the control function over the Pareto set, which results from the optimization problem consisting of the performance functions group.

Many researchers implemented scalarization methods to solve multi-objective prob-

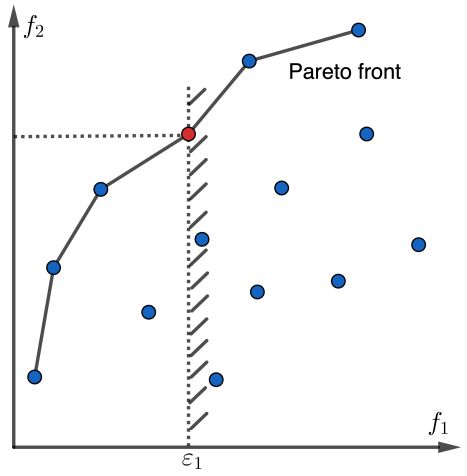


Figure 1.7 – Example of $\varepsilon-$constraint method $\min(-f_2(\boldsymbol{x})$, s.t., $f_1(\boldsymbol{x}) \leq \varepsilon_1)$.

lems. Since these approaches integrate a priori information on the objective function, the obtained solutions are highly preference-dependent. An incorrect formula may lead to an invalid search region. Multi-objective problems require multi-objective optimization techniques, even though it is challenging to accurately obtain high-quality Pareto front.

Based on the Pareto domination

To overcome the difficulties arise from the scalarization methods, researches turned to developing alternative approaches. One classic method is based on the Pareto domination, where a set of solutions is moved to the Pareto front. Assuming that all objective functions are minimizations, the domination can be expressed as:

$$\begin{cases} \forall i = 1, 2, ..., k, & f_i(\boldsymbol{x}') \leq f_i(\boldsymbol{x}) \\ \exists j = 1, 2, ..., k, & f_i(\boldsymbol{x}') < f_j(\boldsymbol{x}) \end{cases} \tag{1.6}$$

In fact, $\boldsymbol{f}(\boldsymbol{x}')$ dominates $\boldsymbol{f}(\boldsymbol{x})$ if $\boldsymbol{f}(\boldsymbol{x}')$ is no worse than $\boldsymbol{f}(\boldsymbol{x})$ for all objectives and $\boldsymbol{f}(\boldsymbol{x}')$ is better than $\boldsymbol{f}(\boldsymbol{x})$ for at least one objective. For instance, a multi-objective minimization considers two objectives $(f_1, -f_2)$ in Fig. 1.8. It is assumed that there is no prior information on objective importance. Point B dominates point C where point B is better than point C in both objectives. And point A and B are non-dominated to each other where point B is better than point A in f_2 but is worse in f_1. All these non-dominated solutions are known as Pareto front. Pareto domination optimization methods approach the entire Pareto front by evaluating the multiple objectives directly.

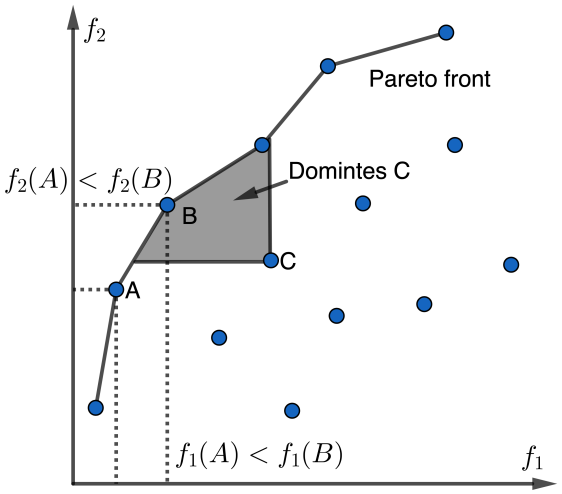


Figure 1.8 – Multi-objective minimization $(f_1(\boldsymbol{x}), -f_2(\boldsymbol{x}))$.

Strength Pareto evolutionary algorithm (SPEA) stored the non-dominated solutions externally [23]. A clustering method maintains the diversity and ensures that the number does not exceed the archive size. The fitness assignment is determined by the number of dominated solutions and ensures that the search direction is towards the non-dominated regions.

Multi-objective particle swarm optimization (MOPSO) used the external repository to keep non-dominated solutions [24]. The theoretical basis of this method is the same as the PSO method, with the addition of repository and dominance concepts. In the swarm, each particle has a corresponding position and velocity. Each particle changes its position according to the velocity deduced from the best particle position and the leader of the repository member.

Multi-objective salp swarm algorithm (MSSA) implemented a mathematical model to update the position of leading and following salps [25]. A repository was designed and applied to preserve non-dominated solutions obtained so far and the food source was chosen from the non-dominated solutions with the least number of neighborhood solutions.

A multi-objective simulated annealing (MOSA) was developed using domination inside state difference probabilities [26]. The algorithm compares the domination between the current solution and new solution, the new one is accepted if the new solution dominates the current solution. This is similar to the acceptance mechanism of a smaller objective function value in single-objective SA. A number of independent individuals are used to

search for Pareto front without exchanging of information between them.

One presented a multi-objective simulated annealing based on dominance amount [27]. During optimization process, an archive is applied to keep all solutions discovered as yet. For any new solution, compare it to the archive members. If some members in the archive are dominated by the new solution, replace them with the new solution. Thus, the size of archive is varying while the population size of GA is constant. Besides, the energy of a solution is measured using the amount of solutions that dominate itself. However, if the archive has few members, the comparison to the archive becomes less useful. To overcome the issue, an attainment surface is created by adding some temporary states to the archive. An attainment surface is defined in the objective space containing the points dominated by the elements of the archive members. The domination amount makes the search gradually proceed to the region where the global optimal solutions exist, and ensures that it is likely to jump out of local search.

Non-dominated sorting genetic algorithm-II(NSGA-II) was implemented based on fast non-dominated sorting and elitist selection techniques in multi-objective optimization [28]. The fast non-dominated sorting strategy is designed to assess the domination of individual in the population. Based on the individual's dominance, the population is divided into different subgroups where all individuals are ordered in accordance with the degree of non-domination: the rank of the first non-dominated solution is equal to 1, then the rank of all non-dominated solutions in the remaining solutions is equal to 2, the process is repeated until all individuals are assigned to a rank, as shown in Fig. 1.9. It is effective to assess the quality of each individual. In the population-based algorithm, the individuals in population evolved with generations. Additionally, when it comes to selection within the same ranked solutions, the NSGA-II computes the crowding distance on the relevant front to measure the solution distribution. Solutions with higher crowding distances values are preferred because they are in less visited areas.

Another nature inspired evolutionary algorithm, on the basis of the relationship between different organisms, lives and survives together, named symbiotic organisms search (SOS) [29]. It is parameterless and requires initialization of population and number of generation. It has three stages of mutualism, commensalism and parasitism. The recently developed SOS has used fast non-dominated sorting to solve multi-objective problems [30].

Several cooperative co-evolutionary frameworks have also been involved. A two-archive evolutionary algorithm C-TAEA adopts two collaborative archives at the same time in the constrained multi-objective problem [31]: one, called the convergence-oriented archive

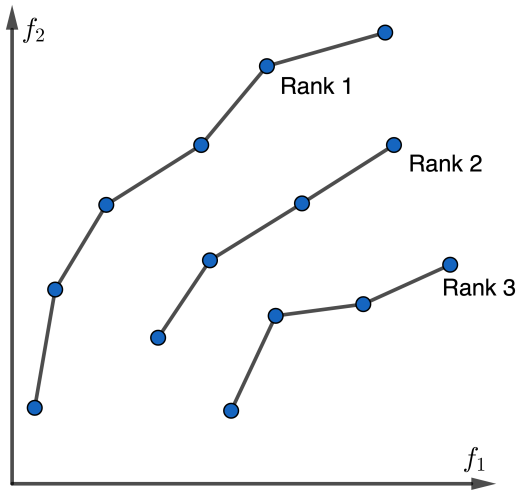


Figure 1.9 – Multi-objective minimization $(f_1(\boldsymbol{x}), -f_2(\boldsymbol{x}))$.

(CA), mainly designed to push the population to the Pareto front using the fast non-dominated sorting; the other, named the diversity-oriented archive (DA), primarily inclined to preserve the population diversity and explore the under-exploited area.

Multi-objective optimizations based on Pareto domination are domination-oriented fitness assignments rather than objective-oriented fitness assignments. They have demonstrated high performance on problems of two or three objectives. Nevertheless, if the problems have more than three objectives to optimize, their performance degrades.

Based on decomposition

The multi-objective problem can be divided into multiple scalar optimization problems and optimized collaboratively.

A vector evaluated genetic algorithm (VEGA), which decomposes the population into disjoint sub-populations governed by each objective function, then mixes the sub-population together, followed by traditional crossover and mutation operators [32]. It is able to find the extreme solutions (in red) but fails to generate the all points in the Pareto front, as shown in Fig.1.10.

Multi-objective evolutionary algorithm based on decomposition (MOEA/D) divides the problem into several scalar sub-problems [33]. In each generation, it creates the new solution based on several of its neighboring sub-problems and uses the best solutions found so far for each sub-problem to form a new population. This is the main difference between

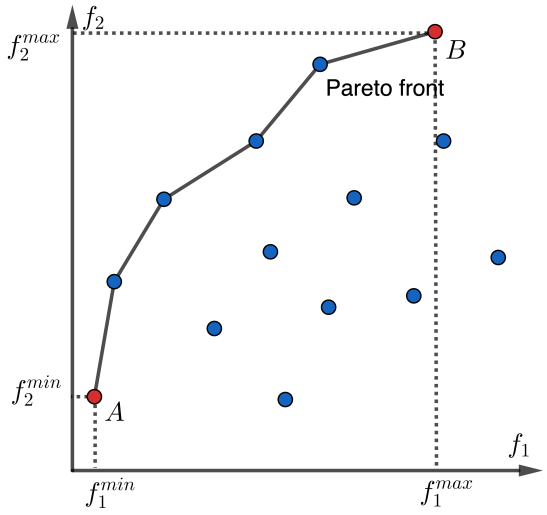


Figure 1.10 – Example of VEGA minimization $(f_1(\boldsymbol{x}), -f_2(\boldsymbol{x}))$.

MOEA/D and a pure scalarization approach. Among them, the most popular aggregation methods are weighted sum (is not applicable for non-convex), Tchebycheff (works for any shape but has poor distribution), and penalty-based boundary intersection approaches (balances convergence and diversity, but relies on weight vectors). The neighborhood relationship between the sub-problems is measured using the distance between the weight vectors. And non-dominated solutions found so far are preserved in an external population (EP).

A number of MOEA/D based approaches have been studied, for example, the NSGA-III evaluates the solution distribution using uniformly allocated reference points [34]. Solutions are linked with the closest reference point, and solutions associated with the less visited reference points are preferred. Considering the convergence and diversity, the augmented penalty boundary intersection is applied on each non-dominated fronts during the selection phase [35].

MOEA/D does not use the concept of Pareto dominance to select solutions, where a solution quality is evaluated by the predefined scalarization function. NSGA-III relies on the decomposition idea to preserve population diversity, where the convergence is governed by Pareto dominance. What's more, a multi-objective evolutionary algorithm based on dominance and decomposition (MOEA/DD) was developed [36] to enhance the performance of MOEA/D by combining dominance relationships and decompositions.

Multi-objective optimization based on decomposition approaches transform the multi-objective problem into several single-objective problems. The single-objective problem is

designed for a different part of the Pareto front. They are applicable for any scalarization methods because of the flexible algorithm framework. However, they need a priori information on where the Pareto front is located in objective space.

Based on indicator

Unlike the conventional Pareto domination multi-objective optimization, performance indicator-based algorithms are theoretically well-supported alternatives. They adopt the performance indicator(s) to evaluate the convergence and diversity of solutions and directly assign fitness value to each individual.

Various indicators have been studied in previous researches. Hypervolume [37] is a widely used performance indicator. It quantifies the convergence and distribution by calculating the area/volume dominated by the provided set of solutions about the reference point. Typically, the indicator assigns better (higher) values to approximations of the Pareto front that dominate more objective vectors than to approximations that dominate fewer objective vectors. Furthermore, it is shown that given a reference point and a limited search space, finding the Pareto optimal set is equivalent to maximizing hypervolume value.

A hypervolume indicator maximization method was developed by searching along its gradient ascent direction relative to design variables [38]. The Pareto front is the geometric location of the point where the gradients of the objective function are collinear and opposite. To substitute zero sub-gradient of the dominated point, five methods were designed to guide these candidates to the Pareto front and the points turn into non-dominated. Thus, hypervolume maximization can approach the Pareto front and increase the diversity of points.

A general indicator-based evolutionary algorithm (IBEA) is proposed that uses the hypervolume indicator for individuals comparisons and corresponding fitness assignments [39]. The fitness of a solution is designed using the sum of the indicator values relative to pairwise comparisons with all other solutions in the population and should be maximized. It is confirmed that fitness proposal conforms to the Pareto dominance relation.

A s-metric selection evolutionary multi-objective optimization algorithm (SMS-EMOA) aims to cover the largest hypervolume with limited number of points [40]. The ranking is performed using the non-dominated sorting strategy. In addition, the hypervolume is used to select individuals in the population. Individuals who contribute the least hypervolume to the worst-ranked front should be abandoned.

In addition to the hypervolume-based optimization, a dominance-weighted uniformity (DWU) heuristic was designed, which considers the population distribution in the design space and maintains the population convergence in the objective space [41]. The DWU can be integrated into any multi-objective optimization approach as a selection criterion.

Generational distance (GD) is a classical convergence indicator [28]. For each point of obtained solutions, GD quantifies distance between the point and the closest point in the true Pareto front. Recently, the authors developed a generational distance-based multi-objective evolutionary algorithm (GD-MOEA) [42]. In each generation, GD-MOEA uses the non-dominated solution set as a reference set and computes the fitness of the dominated individuals.

A novel multi-objective evolutionary algorithm used two-archive (TriMOEA-TA&R), i.e. diversity archive and convergence archive [43]. Decision variable analysis leads the search direction, and a niche-based clearing technique promotes diversity of the decision space. At the same time, the diversity archive uses a clustering technique to ensure diversity of the objective space. The convergence index is computed by summing the objective values rather than Pareto domination, boosting the selection pressure towards Pareto front. In the end, recombining solutions in the diversity and the convergence archives to have a larger number of Pareto optimal set.

The indicator-based methods employ the performance indicator to optimize the obtained solution for desired properties. It has been demonstrated promising results in multi-objective optimization, especially with quite large numbers of functions.

Based on interaction

The methods described above are based on a priori preferences or a posteriori preferences. The former relies on the pre-defined parameters to convert the original problem into single-objective problem. While the latter aims to approach the entire Pareto front. Once the optimization finishes the optimization process, the expert could select the final solution among the obtained solutions. Apart from that, an expert could modify problem formulation and optimization criteria whenever needed for interactive optimization. Ultimately, the interaction aims to find more preferable optimal solutions using the expert intelligence.

In order to identify promising areas, the Pareto navigator was designed to help the decision maker to explore the Pareto front [44]. The interactive learning-oriented method creates a polyhedral approximation of the Pareto front using a set of optimal solutions.

Instead of computing the entire Pareto optimal solutions, it explores the Pareto front interactively. Starting from the current Pareto optimal solution, the decision maker can decide which objective is more important than the other in the design, augmenting that objective at the expense of sacrificing currently less important objectives.

In some complex applications, it is challenging to seek the entire Pareto front with good convergence and distribution during optimization without interaction. Instead of specifying the preferences at the beginning of the problem description, the expert knowledge helps the search procedure follow up their preferences. The interaction can take place after a complete run or during the process.

An interactive optimization strategy based on GA combined with separation algorithm was proposed to solve multi-objective problem [45]. It may be not easy to express objectives explicitly using numerical values. So the interaction involves two steps: the first is the interaction of the expert to select feasible solutions in the initial population of GA; the second is that designer can interact in the environment and modify the obtained solutions locally to improve their performances.

A multi-objective evolutionary optimization algorithm using interactive preference, which applies three steps of 'partitioning- updating-tracking' to interactively express preferences and consistently control interested area according to human cognition process [46]. The decision maker's preference region is expressed and used to lessen the angle-defined feasible objective space before the evolution of nadir point. Then, the preferences of decision-maker are traced and updated dynamically based on satisfied alternatives in the process of evolution. At last, individuals are selected and assessed according to the preference regions in the population.

The hybrid approach developed in the study includes an interactive GA which combines two different niche approaches to enable interactions between the expert and the algorithm [47]. Incorporating niching methods into the approach preserves diversity and avoids offering designers similar solutions. However, this approach requires designers to directly intervene in the optimization algorithm, conducting the search direction to suit their preferences. To avoid fatigue, the designer evaluates each of the representative alternatives in the population.

Moreover, an interactive method was proposed for combining mathematical optimization with expert in the architectural floorplan layout [48]. Interaction enables the designer to dynamically change the optimization formulation by modifying, adding and removing elements, objectives and constraints. The preference can be assessed by solutions com-

parisons. Pairwise comparisons of solutions can be applied to tell whether one solution is better than another, or if they are irrelevant or incomparable. Another approach is to cluster a set of solutions into groups where solutions in each cluster are irrelevant or incomparable. Comparing solutions needs relatively less cognitive burden. However, the burden on experts may increase with the number of solutions.

These interactive approaches are able to search for more preferable solutions without necessarily expressing explicitly of the actual preference. Supposing that there are multiple criteria to be optimized, the interactive techniques can greatly reduce the computational efforts because it is not significantly affected by the Pareto set dimension.

1.2.3 Archive analysis of multi-objective optimization

Unlike the single-objective optimizer, the multi-objective optimization process should move towards the true Pareto front by finding a set of compromised solutions. Thus, a multi-objective optimizer could either preserve the 'good' solutions in an archive, or evolve the 'good' solutions in the population. In summary, the reviewed multi-objective optimization approaches could be classified into two categories: individual-based algorithms (e.g. SA) with a single individual; population-based algorithms (e.g. swarm intelligence (PSO, SSA); evolutionary algorithms (e.g. FPA, GA, SOS)) with multiple individuals in a population. Population-based algorithms are exploration-oriented and can achieve better diversification across the search space, and they are more suitable for multi-objective applications.

It is worth noting that an external archive is usually employed to preserve elite candidates found during the multi-objective optimization process. For examples, the pareto domination based, such as SPEA, MOPSO, MSSA, MOSA, MOEA/D, and many others, are adopted with external archives. These researches have demonstrated that such elite mechanism could guarantee the monotonically non-degenerate performance. It has also been proved that create new solutions with the help of the external archive can improve the diversity of population. Various archiving approaches have been developed to improve the performance of a multi-objective algorithm but with some drawbacks. One main issue is related to the time. To save an alternative into the external archive, it is necessary to check the dominance quality relative to all the archived solutions. The processing time will grow exponentially as the archive size increases, which leads to serious computational efficiency issue. To overcome the shortcoming, various strategies are proposed that allow efficient searching for elite solutions of the entire archive, e.g., the dominated

tree [49]. However, as the archive size increases, the time complexity of these methods remains prohibitively high. Therefore, the archive-free algorithms such as SOS, NSGA-II, etc, are commonly applied to address these issues. They find the final population as an approximation of the Pareto front.

1.3 Handling convergence and diversity

The main goals of multi-objective optimization are to be as close as possible to the real solution and as distributed as possible, that is, convergence and diversity. This section summarizes the existing strategies in multi-objective optimization algorithms to improve the convergence based on the dominance relation and to preserve the diversity of the solution set.

1.3.1 Convergence enhancement

In multi-objective optimization, the number of non-dominated points is small in the beginning, namely convergence stage. When non-dominated solutions accumulate and continuously move on the true Pareto front, the diversity phase is invoked where the solution selection becomes strongly dependent on the diversity measurement. In this case, the approximate Pareto front is well distributed in objective space, but convergence of Pareto front to the true Pareto front becomes progressively worse. A lot of research has been done o improve the convergence by modifying the Pareto dominance relation and developing new selection criteria.

— Fuzzy- based Pareto optimality quantifies the degree of domination [50]. The number of objectives that one alternative is smaller or larger than the other is often considered as the criterion for determining the dominance relationship.

— Controlling dominance area of solutions (CDAS) increases the selection pressure by expanding the area of dominance [51]. CDAS has difficulties in approaching the entire Pareto front, thus self-CDAS is proposed to adjust the expansion angle of different solutions [52].

— Grid-dominance such as ϵ-dominance [53], is a relaxed form of the Pareto dominance because it makes it easier for one individual to dominate another.

— θ-dominance [54], defines a set of weight vectors and leads the solutions along the weight vectors to converge and preserve a uniform distribution.

— Strengthened dominance relation (SDR) [55] is based on niching strategy. It keeps the alternative with the best convergence measurement in each niche to balance the convergence and diversity.

In addition, there are algorithms that combine the local and global optimizers, such as MOEA/D-SQA [56]. To improve the convergence of the global optimizer MOEA/D, a simplified quadratic approximation (SQA) is adopted as the local search operator.

1.3.2 Diversity maintenance

Diversity maintenance is invoked to promote the spread of solutions while approaching to the points near the Pareto front. This can be done using clustering or distance criterion.

— Neighborhood cluster gathers similar solutions together, and only the solutions within the same cluster have the competitive relationship [57]. The neighborhood of a solution is defined by the set of points within a neighborhood distance centered on that solution at a given neighborhood distance.

— Fitness sharing is a classic niching method for punishing the crowded solutions [58]. In fitness sharing approaches, the number of individuals in the same neighborhood affects the fitness of individual. The more individuals in the neighborhood, the smaller the fitness assignment, thereby degrading the number of individuals within crowded niche.

— Crowding distance is a distance-based selection scheme [28]. In crowding methods, one individual competes with its close neighbors. The solutions are firstly sorted in each objective dimension. Then, for each dimension, measure the distance between the solution and the two closest neighbors, and assign the normalized sum of the resulting value as its crowding distance value. At the same fitness level, the individual in the sparse region is preferred.

— Reference point is a new diversity-maintaining operator, which constructs a hyperplane in each generation [34]. It is developed using the perpendicular distance from the solution to several predefined reference lines. The multiple reference points are applied to guide the population well-distributed among each other during the evolution. The niche count of each reference point is relative to the number of its associated individuals in the population .

Multi-objective optimization algorithms aim to find a good approximation of the optimal compromise among the several objectives. A majority of studies only research on the performance of solutions in the objective space. However, from the decision process

perspective, a good distribution of solutions in the decision space has important implications. On the one hand, some multi-objective problems have different Pareto sets with the same objective values, namely multimodel multi-objective problems. On the other hand, points that are very close in the objective space may have large difference solutions in the decision space. In practical applications, finding a set of well-distributed solutions may be more important than finding a set of similar optimal solutions. Thus, it is necessary to enhance the decision space convergence that is difficult to achieve in the existing optimization algorithms.

1.4 Layout problem definition and classification

There are countless applications for optimization. Every problem has a potential to be optimized. In recent years, research on layout problems studies has been intensified. Current research falls into several application areas, including manufacturing systems (assembly lines, fabrications) and service industries (hospitals, airports). The most commonly encountered layout problems are static, where the layout does not change over time; while dynamic layout problem has to be optimized over multiple epochs [59].

1.4.1 Layout problem representation

Typically, the layout problem is related to the location of components (e.g., equipments, machines, departments) in the container (e.g., workshop, plant). The layout problem representation could be classified according to the configurations. The layout could be one-, two-, or three-dimensional problem.

1. The one-dimensional configuration leads to row layout problem. Row layout problem concerns with the arrangement of a given number of components (the assignment of components to rows, and the locations of components within each row), each of a given length of components next to each other along multiple rows, such that the total weighted center-to-center distances is minimized [60].
2. The two-dimensional configuration involves in the planner space, for example, the rectangular environment [11]. In this case, not only the position of each component are optimized, but also the dimension of each component. The placement should satisfy no component overlap and no container protrusion.

3. The three-dimensional configuration is related to the placement on the x-y-z axis. The architecture designs are usually represented by a set of cuboids, and the additional dimension makes the constraints evaluation more difficult. Thus, the three-dimensional configuration is usually simplified as two-dimensional problem (height restrictions) [61] or n-two-dimension problem (staked two-dimension) [62]. Indeed, the three-dimensional configuration is rarely studied in scientific literature.

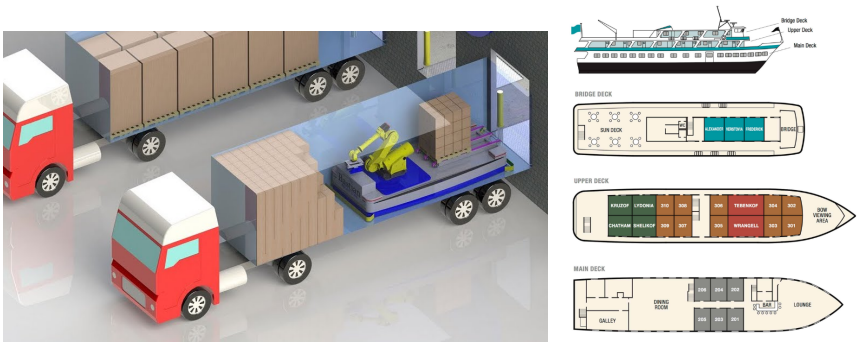
We present a wide range classification of layout problems according to the component and container representations in the followings.

Components A component can be a machine tool, a work centre, a warehouse etc. The components may have regular shapes (e.g., rectangle, circle) and irregular shapes (polygons). To simplify the problem formulation, the components are usually represented by equal-area rectangles with determined sizes [6] or unequal-area rectangles with undetermined dimensions [7]. An undetermined dimension component may be also represented by its area, or its aspect ratio, named unequal-area component [63]. If the aspect ratio is fixed, it corresponds to the fixed shape block. Component Properties include: component number, dimension, mass and so on. One van design example is shown in Fig. 1.11, it consists of a set of equipments such as desks, beds, cabinets and so on.

Figure 1.11 – Van layout design representation.

Containers Depending on the number of containers, layout problems can be divided into single-container and multi-container groups. Most research has focused on placing components on a single container. Nevertheless, the layout design with multiple containers

has always been a hot and challenging point. Container loading problem gives rise to the real-world problem in which components are to be placed on two or more containers, as shown in Fig. 1.12(a). Multiple container problems have a variety of applications that can be distinguished as: multi-row problems concern with assigning and placing components in the design of application-specific integrated circuits [64]; multi-compartment problems deal with components that are placed into the space but have to be separated from each other in the engine layout design [65] and the naval ship design [66]; multi-floor problems consist of horizontal and vertical component movement in the architectural design [67], as shown in Fig. 1.12(b). In these cases, the container(compartment) size is specified.



(a) Packing (b) Ship design

Figure 1.12 – Multi-container problem representation.

Nevertheless, if the sizes can vary, then determining them is also part of the layout design, i.e. the adaptive internal compartment design of container ship [68].

1.4.2 Layout problem formulation

The variables, the objective functions and the constraints form a complete problem. The formulation could be handled through either mathematical programming models or other heuristic methods.

1. Mathematical programming models include quadratic assignment problem (QAP), mixed integer programmings (MIP), Linear programming (LP), nonlinear programming (NLP), integer programming (IP), mixed integer nonlinear programming (MINLP) etc.
2. Heuristic methods include flexible bay structure, slicing tree structure, sequence-pair representation, location/shape pair representation, graph-pair representation

and so on.

In the layout problem, the variables can be either discrete or continuous. In the discrete formulation, the space is divided into identical grids. And the continuous allocation problem can be handled through mathematical programming model, that is QAP [59]. In QAP formulation the components are assigned to a location and at least one component to a particular location [69]. The name was so given because the objective function is a second degree function of the variables and the constraints are linear. And constructing irregular shapes is possible here.

Discrete layouts are not suitable for representing the exact position of components in the plant and cannot model specific constraints as the orientation of components. In such case, a continuous representation is found to be more appropriate. Continuous formulation takes the dimensions and orientation of components as variables such that only regular shapes could be achieved. The component placement is evaluated continuously and can be formulated as a MIP problem. A MIP model consists of the objective function of a mixture of integer and non-integer decision variables that are constrained by a number of equalities and inequalities. One of the first MIP formulations of the layout on the continuous plane was presented [70]: in the proposed model, binary variables specify the relative location of each component pair and ensure that they do not overlap. In addition to the QAP and the MIP, there are other programming formulations for the layout problem. An IP model is a mathematical optimization in which all of the variables are integers [71]. LP is a technique for optimizing a linear objective function, which is subject to linear constraints [72]. On the contrary, the problem is called an NLP if the objective function is non-linear and/or the feasible region is determined by nonlinear constraints [73], and the similar idea in MINLP [74].

In mathematical programming based methods, the dimensions and/or position of the components are represented by variables. The mathematical programming methods can easily incorporate specific description of the problem and perfectly reflect the real-life situations. Layout problems are continuous in nature while using discrete formulation can reduce the search complexity. A flexible bay structure is a continuous layout where the components are located in parallel bays with flexible widths. This special case of the layout problem occurs in manufacturing facilities [75]. The bay structure is similar to the row structure in row facility layout problem (FLPs). Whereas in row FLPs, the rows and components are of equal and fixed height, the width of each bay depends on the total area of the components in that bay. The slicing tree structure organizes a lay-

out into a tree structure which is an encoding representation. It consecutively divides the space into horizontal and vertical directions. The encoding vector usually contains the component sequence, the slicing sequence and the component orientation [76]. The sequence-pair structure is used to represent the relative positions of departments in a block layout [77], which determines the relative locations of the components in an ordered sequence of indices of components. It naturally eliminates inconsistent assignments of binary decision variables used to represent relative components locations. Unlike the sequence-pair representation, the location/shape representation specifies the binary decision variables and facilitates to integrate the unequal-area component shape and relative position information [78]. A graph-pair representation encodes the relative locations of components, one of the graphs represents the horizontal separation of the components, and the other represents the vertical separation [79].

The formulation of problem uses either single-objective or multi-objective. Indeed, a majority of studies work on single-objective problem, for example minimizes material handling cost [80] or maximize adjacency requirement [10]. However, in reality, there are multiple requirements, either quantitative or qualitative objectives, at the same time. Generally, quantitative objectives aim to minimize the sum of material flow between facilities based on the distance functions, activity costs, space utilization and mass distribution. Qualitative objectives aim at placing facilities that utilize common materials, or utilities adjacent to each another, while keeping facilities separate for safety, noise, or cleanliness [81]. Thus, layout problems fall into the category of multi-objective problems. One example can be found in minimizing the occupied room area and the material handling cost [82]. A GA based approach was used to minimize departmental material handling cost and also to maximize closeness rating in a multi-floor layout problem [83].

In most real-world applications, the main constraints are the geometrical constraints between components. No component overlap and no container protrusion are the common geometrical constraints, while orientation or alignment is to define functional relationships between components [8]. The functional constraints specify the requirements to ensure the system's proper functioning [84]. The accessibility to components is one particular layout functional constraint. The constraint exists in many real-world applications while the mathematical expression is still under research. A functional part should be accessible for maintenance in mechanical assemblies design. Keep enough whitespace in cell design to ensure pins accessibility [85]. In facility layout design, space located around the facility guarantees the facility is accessible from the entrance. The accessibility is

the required available space attached to the component and ensures that the component functions properly [86]. Considering the accessibility requirement, the designer can insert the expertise in problem descriptions or integrate the accessibility constraint in problem formulations. Identical rooms are formed into one or two rows to ensure a corridor can access the rooms in pods design [87]. The intersection constraint is applied to force the room interaction and ensure access [67]. The intuition-inspired way limits innovation of finding solutions. Therefore, one can characterize the accessibility as one objective in layout optimization [86]. However, introducing the objective may increase the difficulty and computational cost. As far as we know, the accessibility analysis in layout problems has not been developed maturely yet.

1.5 Layout optimization approaches

The discontinuous constraint satisfied region, the non-linear and non-convex objective of layout formulation make the optimization complex in nature. Finding optimal layouts is beyond the reach of exact optimization techniques, except for small-sized cases [88, 89, 90]. When the problem scale is large, it is not applicable due to computation and memory capability. Thus, meta-heuristic and construction algorithms have been proposed to provide sub-optimal solutions [91, 92, 93, 94].

1.5.1 Exact approaches

The exact algorithm considers the entire solution space and guarantees the optimality of the final layout solution. Exact methods can find optimal solutions for small-sized layout problems. Dynamic programming is a method for solving an optimization problem by solving smaller sub-problems recursively, but it has an exponential complexity. Branch-and-bound algorithm is widely used for highly constrained problems.

An integer formulation with branch-and-bound approach was proposed for space planning [95, 96]. In the first step, the dynamic space ordering (DSO) algorithm is applied to enumerate feasible topological solutions according to the degree of constraints, and then in the second step, the global optimum of each topological solution is determined by the branch-and-bound method. The more constrained the problem, the faster the branch-and-bound method.

Radar search pattern optimization is to cover a set of grids with a minimum number

of available covers. A branch-and-bound method was proposed for radar search patterns [97]. The space of sub-collection of covers can be represented as a finite binary tree, where each node represents the value choice of an integer variable. It is computationally infeasible to explore the entire tree in reasonable time. However, by solving its linear relaxation, the lower bound of the node sub-tree best solution can be estimated at each node. Knowing their lower bound makes it possible to avoid exploring certain subsets.

A branch-and-bound algorithm is developed to minimize the material handling cost in continuous facility layout problem [88]. The branch-and-bound algorithm implicitly enumerates all facility layouts via a permutation tree and prunes inferior combinations of sequence-pairs early in the search. The sequence-pair representation defines the relative position in a complete layout. It dramatically reduces the search space before obtaining the globally optimal layout.

The biggest limitation of exact approaches is that they cannot optimally solve large layout problems due to the computational intractability of the problem. It is not applicable for resolving layout problems with more than 20 facilities in reasonable time [98].

1.5.2 Meta-heuristic approaches

Since exact approaches are not suitable for large-scale problems, numerous researchers have relied on meta-heuristic methods. Those methods are representatives of layout optimization, searching through the huge search space based on objectives values rather than higher order information, such as gradient and Hessian. Unlike exact approaches that can find optimal results, meta-heuristic algorithms can provide good suboptimal solutions. This kind of methods include, tabu search, PSO, SA and GA and so on. The followings are dedicated to those algorithms designed for solving layout problems.

The previous study optimized dynamic facility layout problem (DFLP) with equal departments by applying data envelopment analysis (DEA) with consideration of multiple specific objectives which are cost, adjacency, and distance requested [99]. In the proposed algorithm, the initial layout is first created by arranging departments into multiple periods, and then its neighborhood is generated by the pair-swap and reverse strategy. The most ideal efficient layout is obtained by applying tabu search heuristic of diversification strategy including “frequency-based memory,”“penalty function,” and “dynamic tabu list size” to the DEA model.

A multi-objective PSO is applied to the layout of wireless sensor networks and to find the sensor locations to optimize the coverage and lifetime [100]. The coordinates of the

sensor nodes are considered as design variables. It maintains an elite archive and uses the archive members to dynamically guide the particle swarm to find more and better non-dominated solutions.

Pareto simulated annealing (PSA) is designed for intelligent exploration instead of evaluating all discrete search space of cabinet configurations [101]. It aims to determine the optimal arrangements of cabinet components with consideration of the total wire length and heat convection in the control cabinet. The probability of accepting a poor solution is defined by the difference of the scalar objective function value and the temperature. During the iterative search process, the non-dominated solutions are stored in an archive, and the weights of objectives are varied to explore the entire set of non-dominated solutions.

Meta-heuristic algorithms are widely used to solve layout problems. However, they converge slowly to the Pareto front. Thus, a hybrid algorithm that integrates local search is usually developed to improve performance.

A modified GA was applied to optimize the material handling cost of the layout, where the position and rotation of components are encoded into the chromosome [102]. If the rotation of the block is considered, the problem is formulated as a mixed-integer problem. However, the algorithm proposed a rotation operator that transformed the mixed-integer case into a non-integer problem. This also reduced the number of variables in the optimization problem by a third. Besides, the local optimal solution is further improved using the gradient based local search techniques.

A hybrid GA and LP approach uses the relative location/shape encoding scheme to optimize the material flow in unequal-area components on a continuous plane [78]. The location/shape encoding scheme converts the MIP into LP, where GA determines the relative position and LP optimizes the actual position and shape.

An adaptive variable neighborhood search and GA optimizes the material handling cost and closeness rating score simultaneously [81]. A layout is created using the slicing sequence in the chromosome. If the similarity exceeds the pre-defined threshold, the genetic algorithm will use a local search of variable neighborhood search. The similarity coefficient is defined by the difference between the two chromosomes.

An interactive GA was proposed to allow direct intervention between the algorithm and the expert [47]. The layout is represented by flexible bay structure, with the bay divisions and facility sequences encoded into the chromosome. The expert evaluate the score of each solution in order to perform interaction by continuously including the expert

preferences. In addition, a clustering method is introduced to group the similar solutions and show the representative of each group to the expert. A niching method is adopted to maintain the diversity of solutions at the same time.

1.5.3 Construction and meta-heuristic hybrid approaches

Current layout optimization techniques can be divided into two groups according to the type of variables, either discrete or continuous. In continuous optimization, the dimensions and orientation of components are variables, and the progress evolved by applying different operators [102]. LPs are continuous in nature while using discrete formulation can reduce the search complexity. A flexible bay structure [103] and a slicing tree [104] divide the container into horizontal and vertical directions, and the components are restricted to be located inside follow a specific logical order. Whereas the construction algorithm can generate a complete layout that others cannot represent by sequentially placing components [105, 106].

Construction procedures build a layout from scratch by successively selecting and placing facilities until a completed layout is obtained. By automating the selection rules for a set of component configurations, the designer's intelligent thought process can be used to define the construction algorithm at each stage. The simplicity of construction algorithm is often associated with feasible solutions and can be used in parallel with meta-heuristic algorithms.

Construction algorithm combines the component configuration to generate a complete layout. In combinatorial problems, a general neighbor generation method is to perform permutation operations or bit flip operations at random positions. The search space for all possible combinations of components in a layout design problem is so large that a brute force approach is impractical. Moreover, the discrete search space makes the gradient or downhill methods inapplicable. Indeed, the construction algorithm applied with meta-heuristic proved to be an efficient algorithm for layout optimization [107]. For instance, GA coupled with construction algorithm and SA algorithm hybrid with construction algorithm.

Construction and GA optimization GA works with a set of solutions, that is population, and improves the solutions through crossover and mutation operators and survival selection. It proves that a large crossover rate may lead to premature convergence. Besides, a small mutation rate may lead to genetic drift, while a large mutation rate may lead

to loss of good solutions unless elitist selection is used. The most important component of GA is the solution representation of the problem, also known as individual or chromosome. These properties resulted in the development of multi-objective genetic algorithms with different structures. A GA encodes the packing sequences and container loading sequence in a chromosome, a placement algorithm determines the component placements considering the compactness [108]. One hybrid approach minimizes the distances between unequal-area components: a biased random-key genetic algorithm (BRKGA) determines the facility sequence, facility aspect ratios and the coordinates of the first facility to be placed, and a placement positions facilities inside one of the empty maximal space considering the objective function, finally, a linear programming model is applied for local optimization [109]. A NSGAII algorithm encodes the placement sequence to optimize the multi-objective two-row layout problem [110]. The layout construction procedure places departments according to the order represented by each individual and starts with placing departments in the lower row in the first floor, like packing a set of items into containers. A local optimization algorithm based on SA is invoked if the similarity among individuals extends the threshold.

Construction and SA optimization SA is an effective stochastic optimization technique known for its high performance in solving combinatorial problems and attracts our attention because of the capability to solve large and complex layout problems. SA is a single-objective optimization technique which is provably convergent, making it an interesting technique for extension to multi-objective optimization. Considering the fact that many single-objective combinatorial optimization problems are NP-hard, it is reasonable to expect that the corresponding multi-objective versions are usually harder to solve. SA was firstly came up to solve combinatorial problems and has also been widely used in practical applications: traveling salesman problem [111], knapsack problem [112] and railway crew scheduling [113]. Several layout optimization methods combined construction algorithm and SA can be found in recent layout problems studies. SA with swap and displaced operators was developed to optimize the initial solution created by the placement, which first divides a given area into four equally sized quadrants. Afterwards, all facilities are placed radially [114]. A two-step heuristic algorithm using discrete modelling was proposed, first, the feasible layout is constructed using the placement order of SA, in each zone, the position of the facility is determined with the objective measurement between this facility and the previously placed ones. Then, local optimization is followed

to improve the solution in each zone [115]. SA has fewer parameters and the simpler structure compared to GA optimizations.

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. And the optimization may generate many infeasible solutions, particularly in the dense layout problem. Besides, the existing construction 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.

1.5.4 Multi-container layout optimization

Depending on the number of containers, layout problems can be divided into two groups. Most research on layout problems has focused on placing components on a single container. Nevertheless, the multi-container layout design has always been a hot and challenging point.

Multi-container layout can be formulated as a continuous space allocation problem. One possible simplification is to transform empty spaces into grids and then solve the quadratic assignment problem [4]. Considering the real-life scenario, a new multi-floor departments arrangement problem was formulated as a MINLP model and solved by an exact ϵ-constraint algorithm [82]. The same floor is divided into several blocks, one block contains more than one department, each department consists of multiple fixed rooms and the same room must be adjacent.

Indeed, the assignment of components to containers can be either flexible or fixed. The former decides to assign components to container during the optimization process [116]. Given the complexity of the problem, thus, some approaches decompose the main problem solving into two steps: first distributing the components over the containers, and then independently optimizing the layout of each container [117]. While the latter predetermines to assign specific components to specific containers. If the components are arranged in well-defined container, then the multiple container layout design is rendered as the single container layout problem [62].

It is worth noting that the container size may be either determined or undetermined. The layout optimization under bounded region (i.e. container) is more common in the industrial engineering, for example, a naval ship compartment design with fixed inner

wall structure [118]. Considering the construction cost, thus leading to the design of more compact plants. However, as the size and construction cost of layout designs increase with more space, the trend toward undetermined space layout design becomes more important. Space layout design has become a much more important consideration in efficiently allocating all compartments within the limited region. One study has determined the optimal rectangular shape of land area using the MIP model [119].

1.6 Conclusion

In this chapter, firstly, we described the formulation of single-objective and multi-objective problems. We thus presented the main differences between the "optimal" solutions in these two cases and reviewed the resolution methods. Deterministic and stochastic methods in single-objective optimization can be extended to multi-objective optimization. The scalarization-based, domination-based, decomposition-based, indicator-based as well as interaction-based approaches have been distinguished. As discussed, a critical consensus has been appeared that using the external archive along the optimization process can improve the performance at the cost of computation efficiency. For multi-objective optimization algorithms, one of the key features is the ability to find a good approximation to the optimal trade-off among the multiple objectives. However, two issues are still open. First of all, the solution selection becomes strongly dependent on the diversity measurement such that convergence of Pareto front towards the true Pareto front is gradually deteriorated. A second issue is related to the decision space. A majority of algorithms focus only on the distribution of solutions in the objective space. Nevertheless, a good representation of solutions in the decision space is also important from the point of view of the decision-making process.

The second main part of this chapter is the layout problem. We provided a review of the layout classification in terms of the representation and formulation. From the literature review, it can be concluded that it is still open and active area for research. This motivates the author to work in the layout optimization. For this purpose, the previous researches in the filed have been analyzed taking into account the formulation and resolution approaches. In the present literature, there are trends of the optimization techniques with combining effort of construction and meta-heuristic method to solve the layout problem. The concept of construction approach has gained a lot of attention from researchers due to its feasibility. Furthermore, we closed this chapter by presenting one

hotspot layout problem, namely multi-container layout problem. We noticed that the research on flexible container size is a relatively under-discovered area.

Chapter 2

Population-based simulated annealing for multi-objective problem

2.1 Introduction

Over the past decade, meta-heurisitc algorithms have been recognized to be very successful in solving multi-objective optimization problems. Simulated annealing has an advantage over the other meta-heuristic algorithms in terms of the ease of implementation and gives reasonably good solutions for many practical problems. It starts from an initial point and searches for new nearby points to find the global optimal solution. There have been some attempts to extend SA to multi-objective optimization. In most early attempts, a single objective function was constructed by combining the different objectives into one objective function using a weighted sum method. The problem is how to choose the weights in advance. There are also some alternatives used in this regard.

In order to develop an efficient optimizer with a simple structure, two multi-objective simulated annealing algorithms are proposed: one is archive-based SA and the other is archive-free SA. Both algorithms integrate the idea of Pareto domination. Compared with the well-known NSGA II algorithm, the proposed approaches are validated using various multi-objective problems. Furthermore, we analyze the convergence resistance and improvement methodology.

2.2 Multi-objective simulated annealing algorithm

Unlike the single-objective optimizer, the multi-objective optimization process should move towards the true Pareto front by finding a set of compromised solutions. Thus, a

multi-objective SA could either preserve the 'good' solutions in an archive, or evolve the 'good' solutions in the population.

2.2.1 Scalar simulated annealing algorithm

SA is a stochastic neighborhood search approach for global optimization and has been widely implemented in various combination problems. It originated from the concept in physics explaining the annealing of a solid until finding the minimal energy. Similar to the physical process, SA 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. The flowchart of a typical scalar SA algorithm is presented in Fig 2.1.

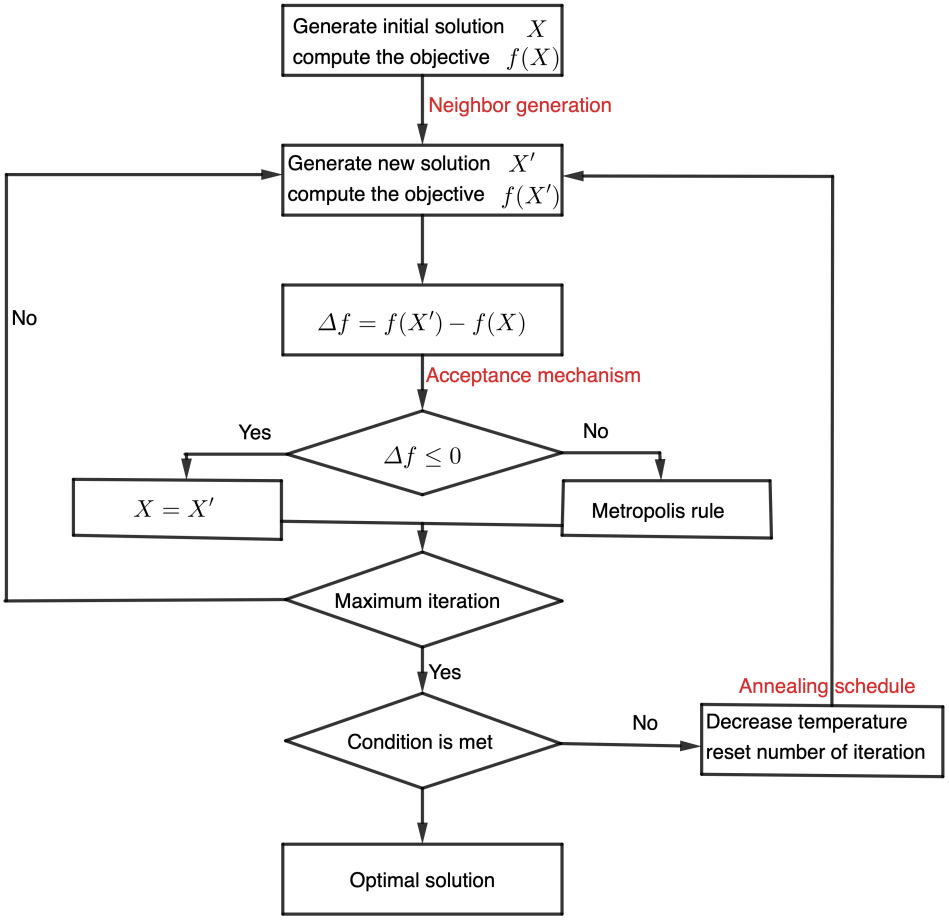


Figure 2.1 – Flowchart of scalar SA.

The main features of the SA method are:

— Neighbor generation: new solution far away from the current solution allows coarse search whereas the new solution nearby leads to fine search.
— Acceptance mechanism: the condition in which a solution is accepted. A worse point is accepted probabilistically where the likelihood of accepting a solution worse than the current solution is a function of the temperature of the search and how much worse the solution is than the current solution, i.e, the metropolis acceptance criterion: $p(\Delta f) = (f(X') - f(X))/\text{temperature}$, which allows the algorithm to escape from local minima when the temperature is high.
— Annealing schedule: which is the function of temperature that controls the probability $p(\Delta f)$ of accepting a cost-increasing interchange. The annealing schedule can be a function of the iteration number or a constant decrease rate.

Compared to other meta-heuristic optimizations, SA is individual-based optimization with global search ability and has simple structure with less parameters.

2.2.2 Archive-based simulated annealing

The proposed archive-based SA is based on Pareto domination criteria. The population P has N individuals $X_i, i \in N$ with the corresponding objective values $\boldsymbol{f}_i$. The framework of archive-based SA is presented in Algorithm 1. In each iteration, the new population will be generated as $P' = lb + rand * (ub - lb)$ in line 5, where $lb = max(P - \tau * (Ub - Lb), Lb)$ and $ub = min(P + \tau * (Ub - Lb), Ub)$. Lb, Ub are the global boundaries and lb, ub are the local boundaries defined by the interval parameter $\tau \in (0, 1)$. During the optimization process, the new population is compared with current archive to form an overall non-dominated set of solutions, which updates the status of archive $\mathcal{A}$ in line 7. If the size of the archive $|\mathcal{A}|$ exceeds the limited number M, crowding distance strategy [28] will be used in line 9. Compared to most of the state-of-the-art diversity maintenance strategy, crowding distance is simple and efficient. The computation of the crowding distance based on the normalized values of objectives, is given below:

$$CD_i = \sum_{j=1}^{m} d_{i_j} / \Delta f_j \tag{2.1}$$

where Δf_j is the interval of the j-th objective function. The sum of individual crowding distance values for each objective gives the overall crowding distance value using the nearest neighbors, as depicted in Fig. 2.2. The individuals with small value of CD will

Algorithm 1 Archive-based Simulated annealing

Input: $P = X_1, X_2, ..., X_N$
Output: $\mathcal{A}$

1: $\mathcal{A} := \mathcal{A} \cup P$
2: $\boldsymbol{f} = \boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_N$ =objective(P)
3: **while** stop condition not met **do**
4: **while** *iteration* in inner loop **do**
5: P'=*neighbor generation*(P)
6: $\boldsymbol{f'}$ =objective(P')
7: $\mathcal{A} :=$ *update domination status*($\mathcal{A} \cup P'$)
8: **if** $|\mathcal{A}| > M$ **then**
9: $CD =$ *crowding distance* ($\mathcal{A}$)
10: $\mathcal{A} =$ *crowded solution removement*($\mathcal{A}, CD$)
11: **end if**
12: **while** $i < N$ **do**
13: $X_i, \boldsymbol{f}_i$=*individual acceptance mechanism*($X_i, X'_i, \boldsymbol{f}_i, \boldsymbol{f}'_i$)
14: **end while**
15: **end while**
16: decrease temperature t
17: **end while**

be removed until the archive size reaches the limited number in line 10.

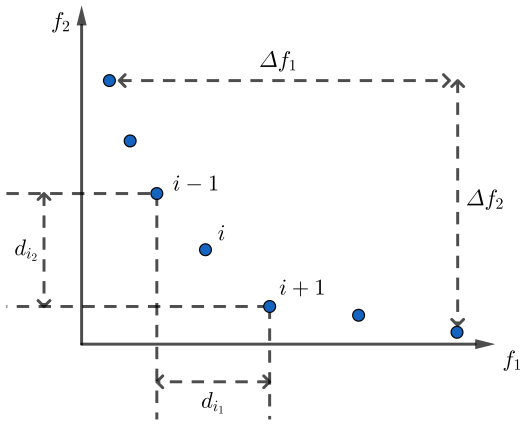


Figure 2.2 – Crowding distance of individual i.

As explained before, the scalar SA explores the space through an individual, and the final result of the individual-based optimizer strongly relies on the initial solution. However, finding reasonable solutions in high-dimensional and multimodal problems may take a long computational time. Therefore, archive-based SA adopts the second loop using

the archive $\mathcal{A}$ as the initial population, which is similar to the social behavior. Overall, archive-based SA is similar to the scalar SA, but differs significantly in the acceptance mechanism and the additional annealing loop of the archive $\mathcal{A}$. These two ideas are described as follows:

Individual acceptance mechanism Accepting the worse solutions allows more extensive search for the global optimal solution. The comparison is individual to individual. For the *i-th* new individual X_i' with objective value $\boldsymbol{f}_i'$, it has three possibilities compared to the current *i-th* individual X_i with objective value $\boldsymbol{f}_i$, as shown in Fig.2.3.

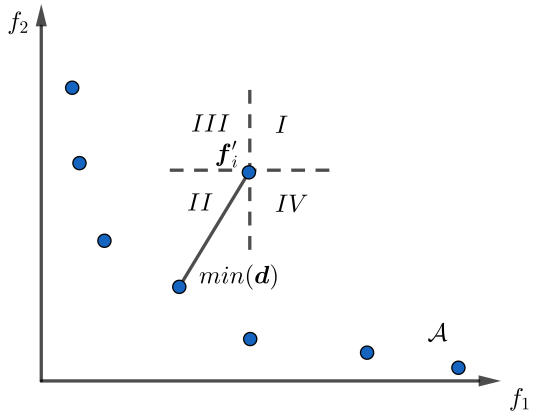


Figure 2.3 – Criteria to select offspring.

— Case I: $\boldsymbol{f}_i$ is dominated by $\boldsymbol{f}_i'$, then the new solution is better than the current solution. Update X_i with X_i', $\boldsymbol{f}_i$ with $\boldsymbol{f}_i'$.

— Case II: $\boldsymbol{f}_i$ dominates $\boldsymbol{f}_i'$, the new solution is worse than the current solution. The probability of accepting the worse solution is specified by an acceptance probability p

$$p = e^{-(\boldsymbol{f}_i' - \boldsymbol{f}_i)/t} \tag{2.2}$$

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 III, IV: $\boldsymbol{f}_i'$ and $\boldsymbol{f}_i$ are non-dominated to each other. It is not easy to tell which one is better according to the domination criteria only. Therefore, we compare the two cases considering the archive $\mathcal{A}$:

1. If $\boldsymbol{f}'_i$ is not dominated by any point in $\mathcal{A}$, then we consider it as a better solution. Update X_i with X'_i, $\boldsymbol{f}_i$ with $\boldsymbol{f}'_i$.
2. If $\boldsymbol{f}'_i$ is dominated by any existing point, then we compute the Euclidean distance $\boldsymbol{d}$ between $\boldsymbol{f}'_i$ and solutions in $\mathcal{A}$. Taking the minimum distance as the transition energy,

$$p = e^{-min(\boldsymbol{d})/t} \tag{2.3}$$

we accept the new solution when it closes to the optimal Pareto front.

Additional loop In the first loop, with population size $N = 1$, we evaluate an individual at each iteration and keep all non-dominated solutions in the archive $\mathcal{A}$. Since each iteration of SA provides just one alternative, the algorithm attempts to find a set of Pareto optimal solutions by using multiple iterations. To enlarge the number of the final points along the Pareto front, a second annealing loop is followed. Take the current non-dominated solutions as the initial population $P = \mathcal{A}$, and apply a small perturbation to the search region near the current non-dominated solutions.

2.2.3 Archive-free simulated annealing

Algorithm 2 Archive-free Simulated annealing

Input: $P = X_1, X_2, ..., X_N$
Output: P

$\boldsymbol{f} = \boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_N$ =objective(P)
while stop condition not met **do**
  while *iteration* in inner loop **do**
    P'=*neighbor generation*(P)
    $\boldsymbol{f}'$ =objective(P')
    $(R, R') = fast\ nondominated\ sort(\boldsymbol{f} \cup \boldsymbol{f}')$
    $(CD, CD') = crowding\ distance(\boldsymbol{f} \cup \boldsymbol{f}', R, R')$
    $P = dynamic\ selection(P, P', R, R', CD, CD')$
  end while
  decrease temperature t
end while

Even the use of an external archive, purely for storage purposes, can bring substantial benefits for multi-objective optimization, but it introduces additional computation efforts. To overcome the potential drawbacks, a population-based archive-free simulated annealing Algorithm 2 is proposed. It incorporates the population's ability to search different

regions and collect information from different individuals. On the one hand, it performs as scalar SA by accepting the worse solutions in the population; on the other hand, it tries to preserve the better solutions in the population. It's worth noting that the proposed population-based SA is an archive-free optimizer, which reduces the computation efforts by keeping the well-distributed non-dominated solutions in the population rather than in the external archive. In order to enlarge the search space while accelerating the convergence speed, dynamic selection based on fast non-domination sort and crowding distance criteria is integrated into the optimization. The comparisons are made on the combination of current and new populations. At the end of the operation, the first non-dominated solutions are assigned a rank equals to 1, the second non-dominated solutions are assigned a rank equals to 2, and so on. And we have the rank $R = (R_1, ..., R_N)$ of the current population and the rank $R' = (R'_1, ..., R'_N)$ of the new population in line 6; the CD and the CD' contain the crowding distance of the individual in each rank level in line 7. Then *dynamic selection* is followed in line 8 to select the next population that maintains the population convergence and distribution.

Algorithm 3 *dynamic selection*

Input: P, P', R, R', CD, CD'
Output: P

1: $P_{next} = \emptyset$
2: **for** $j \leftarrow 1$ to N **do**
3: (R_i, CD_i, X_i) = individual i with minimum rank and maximum distance in P
4: $x_{param} = [R_i, -CD_i]$
5: $y_{param} = [R'_j, -CD'_j]$
6: **if** y_{param} dominates x_{param} **then**
7: $P_{next} = P_{next} \cup X'_j$
8: **else**
9: $\delta = y_{param} - x_{param}$
10: **if** rand $< e^{-\delta/t}$ **then**
11: $P_{next} = P_{next} \cup X'_j$
12: **else**
13: $P_{next} = P_{next} \cup X_i$
14: $P = P \setminus (X_i)$
15: **end if**
16: **end if**
17: **end for**
18: $P := P_{next}$

dynamic selection It is inspired by the Metropolis–Hastings algorithm and applied according to the rank R and R' and the crowding distance CD and CD'. The pseudo-code is given in Algorithm 3. First initialize the population P_{next} to be empty in line 1. To select the individual, in general, the main idea is to compare the individual X'_j in population P' with the best individual X_i of P with minimum rank and maximum distance in line 3. The acceptance of X'_j occurs if X'_j is better than X_i in the rank and the distance in line 6, or it is still a promising solution in line 10. Otherwise, fill the P_{next} with X_i in line 13 and remove X_i from P in line 14. *dynamic selection* continues until there are N individuals in P_{next}.

2.3 Algorithm assessment

The performance of our proposed methods are validated and compared with the well known NSGA II algorithm on a number of continuous multi-objective problems, a set of combinatorial multi-objective 0-1 knapsack problems. The main reasons for choosing NSGA II for comparison are that it is based on the population evolution and outperforms than many popular algorithms. The test problems are chosen in such a way so as to systematically investigate various aspects of an algorithm: convergence, precision, effectiveness and so on. The algorithms were coded in Python and the experiments were conducted on a computer with Intel Core i7 - 6770 3.4 gigahertz CPU and 31.2 gigabyte memory running the Linux operating system.

2.3.1 Continuous benchmarks and performance evaluations

We use five widely used multi-objective benchmark functions to compare the proposed algorithms with NSGA II. All of these instances are minimization of the objectives, and the details about these test functions are outlined here.

— Test Function-1: Schaffer function N1 (SCHN1) comprises of two objectives with one variable and convex in nature. Mathematically given by

$$\text{Minimize } = \begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \end{cases} \tag{2.4}$$

where $x \in [-10, 10]$

— Test Function-2: Schaffer function N2 (SCHN2) consists of two objectives with one

variable and discontinuous given by

$$\text{Minimize} \begin{cases} f_1(x) = \begin{cases} -x & \text{if} \quad x \leq 1 \\ x-2 & \text{if} \quad 1 < x \leq 3 \\ 4-x & \text{if} \quad 3 < x \leq 4 \\ x-4 & \text{if} \quad x > 4 \end{cases} \\ f_2(x) = (x-5)^2 \end{cases} \tag{2.5}$$

where $x \in [-5, 10]$

— Test Function-3: Poloni function (POL) consists of two objectives with two variables. It is non-convex and disconnected.

$$\begin{aligned} \text{Minimize} \ &= \begin{cases} f_1(x,y) = \left[1 + (A_1 - B_1(x,y))^2 + (A_2 - B_2(x,y))^2\right] \\ f_2(x,y) = (x+3)^2 + (y+1)^2 \end{cases} \\ \text{where} \ &= \begin{cases} A_1 = 0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2) \\ A_2 = 1.5\sin(1) - \cos(1) + 2\sin(2) - 0.5\cos(2) \\ B_1(x,y) = 0.5\sin(x) - 2\cos(x) + \sin(y) - 1.5\cos(y) \\ B_2(x,y) = 1.5\sin(x) - \cos(x) + 2\sin(y) - 0.5\cos(y) \end{cases} \end{aligned} \tag{2.6}$$

where $-\pi \leq x, y \leq \pi$

— Test Function-4: Quadratic function (QUAD) consists of two objectives with two variables. It has sharp peak and low tail in nature.

$$\text{Minimize} \ = \begin{cases} f_1(x,y) = (2x - y + 15)^2 + (-x + 4y)^2 \\ f_2(x,y) = (-2x - y + 15)^2 + (-x - 2y)^2 \end{cases} \tag{2.7}$$

where $-10 \leq x \leq 10$ and $-10 \leq y \leq 10$

— Test Function-5: Fonseca function (FON) consists of two objectives with three variables. It is non-convex in nature.

$$\text{Minimize} \begin{cases} f_1(x) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2\right) \\ f_2(x) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i + \frac{1}{\sqrt{3}}\right)^2\right) \end{cases} \tag{2.8}$$

where $x_i \in [-4, 4]$ and $1 \leq i \leq 3$

The operator parameters for NSGA II are set following the practice in [120]. For archive-free SA and NSGA II, the population size N is taken as 100 and number of generations

is taken as 250, while for archive-based SA, the maximum iterations is set to 15000 with $N = 1$ in the first annealing loop and the maximum iterations is set to 100 in the second annealing loop, the archive size M is 100. Therefore, these comparative algorithms maintain the same number of function evaluations. For archive-based SA algorithm, the initial temperature $t_0 = 100$ and the final temperature $t_f = 10^{-5}$, while for archive-free SA algorithm, $t_0 = 1$ and $t_f = 10^{-7}$. The initial temperature values were determined following the idea of acceptance ratio [121]. The cooling rate is kept equal to 0.9 and the neighbor interval τ is taken as 0.2. Each algorithm is run twenty times and quantitative results are calculated using three metrics below.

1. Generational distance (GD): this performance measurement determines how far is the distance deviation between the optimal solutions achieved by the proposed algorithm and true Pareto front.

$$GD = \frac{1}{n}\sqrt{\sum_{i=1}^{n} d_i^2} \tag{2.9}$$

   where n is the number of non-dominated points, d_i measures distance between each point and nearest point in Pareto front. Therefore lower the value of GD reveals the goodness of a multi-objective algorithm. GD = 0 represents that all the solutions generated are on the true Pareto front.

2. Diversity metric (Δ): the diversity metric measures the uniformity in the distribution of non-dominated solutions.

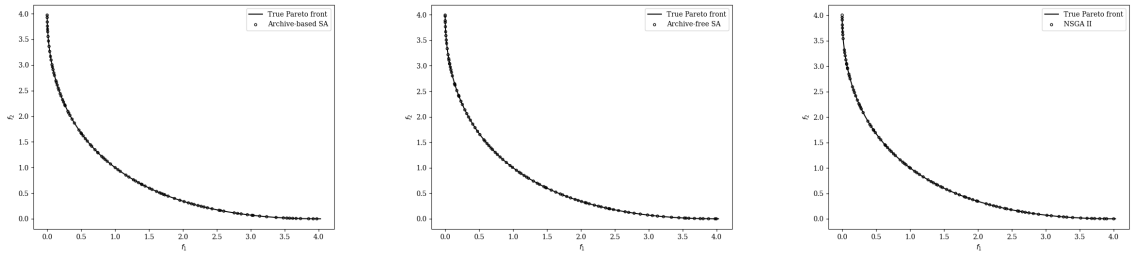
$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (n-1)\bar{d}} \tag{2.10}$$

   where d_f and d_l measures how far that boundary solutions away from extreme solutions. $\bar{d}$ is the average value of the set of d_i. Smaller value of metric Δ the better diversity of non-dominated solutions set. $\Delta = 0$ represents the most widely and uniformly distributed set of non-dominated solutions.

3. Computational time (CT): the run time of algorithms is evaluated which is a performance measurement to show the computational efficiency of the algorithms.

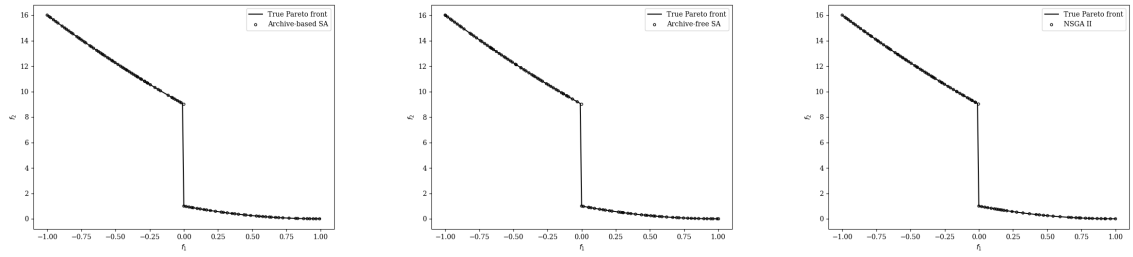
The Pareto fronts obtained for the benchmark functions SCHN1, SCHN2, POL, QUAD and FON using the proposed archive-based and archive-free multi-objective SA and comparative algorithms NSGA II are shown in Fig. 2.4 - Fig. 2.8. It is observed

that in SCHN1 the algorithms are accurately converge to the true Pareto front. The SCHN2 and POL both functions are discontinuous in nature. The obtained solutions are approximately on the true Pareto front but the diversity among the solutions are better in archive-free SA over the comparative algorithms. It is observed that in QUAD the NSGA II convergence and diversity to the true Pareto front is inferior than the other algorithms. In case of FON all the algorithms Pareto fronts slightly deviate from the true front. However the deviation is lower in case of archive-free SA compared to the rest algorithms.



(a) (b) (c)

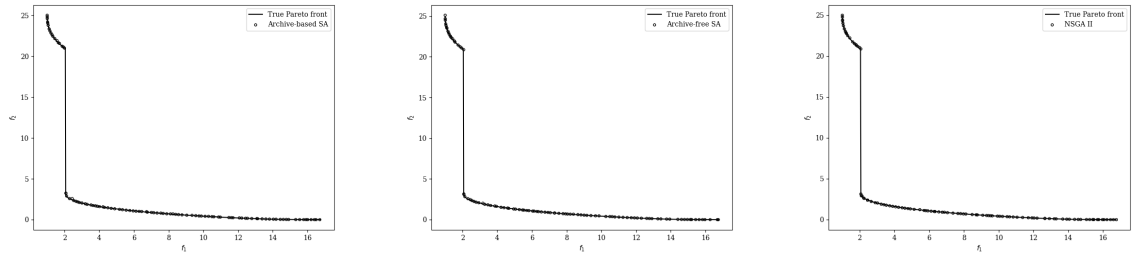
Figure 2.4 – Pareto front determined by (a) archive-based SA, (b) archive-free SA, (c) NSGA II on SCHN1.



(a) (b) (c)

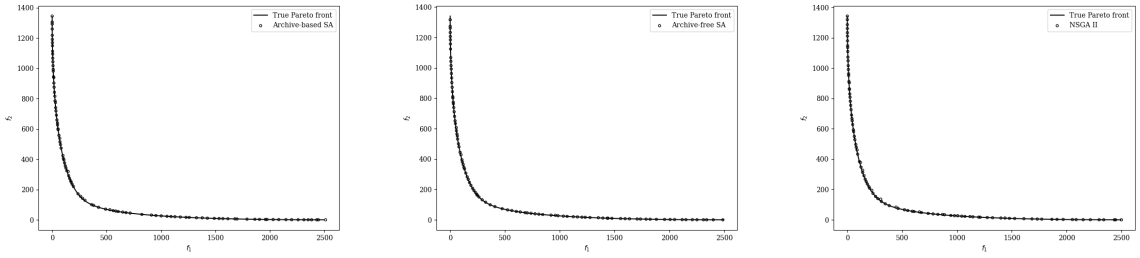
Figure 2.5 – Pareto front determined by (a) archive-based SA, (b) archive-free SA, (c) NSGA II on SCHN2.

The parameters GD, Δ and CT are obtained in the form of mean, standard deviation (SD), best and worst values over twenty independent runs for the algorithms that are summarized in Table 2.1, Table 2.2 and Table 2.3. In the tables the best results achieved are high-lighted in bold letters. It is observed that, in most cases, the proposed archive-free SA has smaller mean, best, worst GD and Δ values, which indicates the convergence



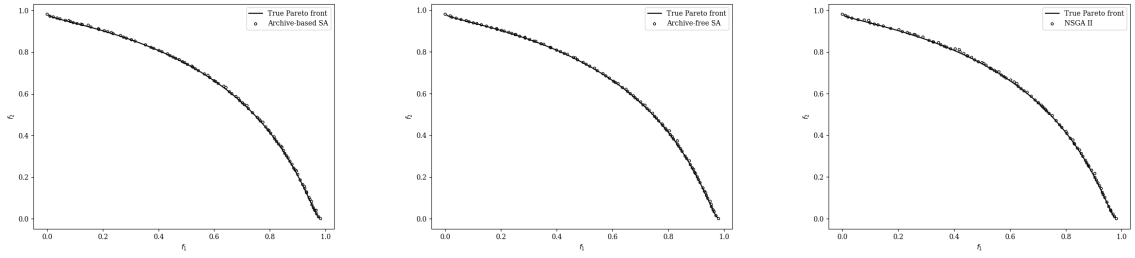
(a) (b) (c)

Figure 2.6 – Pareto front determined by (a) archive-based SA, (b) archive-free SA, (c) NSGA II on POL.



(a) (b) (c)

Figure 2.7 – Pareto front determined by (a) archive-based SA, (b) archive-free SA, (c) NSGA II on QUAD.



(a) (b) (c)

Figure 2.8 – Pareto front determined by (a) archive-based SA, (b) archive-free SA, (c) NSGA II on FON.

of most of the solutions to the true Pareto front while maintaining proper diversity among them. In comparison with archive-free SA and NSGA II, both have lower computational time compared to archive-based SA, which indicates that the population-based feature is

Table 2.1 – Comparative results (using GD) of proposed archive-based, archive-free SA approaches with NSGA II obtained after twenty independent runs on benchmark multi-objective functions: SCHN1, SCHN2, POL, QUAD, FON.

Algorithm	SCHN1				SCHN2			
	Mean	SD	Best	Worst	Mean	SD	Best	Worst
Archive-based	**0.00083**	**0.00003**	0.00075	0.00089	0.01281	0.00078	0.01175	0.01453
Archive-free	0.00084	0.00004	0.00077	0.00091	**0.01261**	0.00077	0.01040	0.01374
NSGA II	0.00113	0.00008	0.00104	0.00139	0.01319	**0.00066**	0.01170	0.01447

Algorithm	POL				QUAD			
	Mean	SD	Best	Worst	Mean	SD	Best	Worst
Archive-based	0.03471	0.00742	0.02905	0.06582	1.19464	**0.13965**	0.98349	1.50127
Archive-free	**0.02999**	0.00217	0.02689	0.03694	**1.00827**	0.14573	0.76512	1.43539
NSGA II	0.03200	**0.00188**	0.02770	0.03564	1.23190	0.18997	0.97229	1.87043

Algorithm	FON			
	Mean	SD	Best	Worst
Archive-based	0.00453	**0.00018**	0.00422	0.00483
Archive-free	**0.00445**	0.00021	0.00386	0.00478
NSGA II	0.00488	0.00026	0.00437	0.00545

effective in accelerating the convergence and in maintaining the diversity. However, in all the benchmark test cases the computational time of NSGA II is slightly lower followed by archive-free SA, perhaps for the reason that coding scheme. The structure of archive-free SA will be improved for efficient in future work.

The GD value numerically describes the convergence of the Pareto front. It is clear from Table 2.1 that the average GD values obtained by archive-free SA are smaller than archive-based SA except for SCHN1. Archive-free SA performs better than NSGA II in terms of mean GD value for all the test problems, which indicates the dynamic selection performs better in approximating the Pareto front on the convergence. In the case of QUAD problem the GD values of all methods are much higher than other functions. The convergence resistance arises our interest, and the convergence improvement analysis will be presented in the next section.

The Δ value measures the spread of the solutions in Pareto front. As described in Table 2.2, archive-based SA has smaller Δ values except for SCHN1 and SCHN2. Generally speaking, archive-free SA has better performance than archive-based SA and

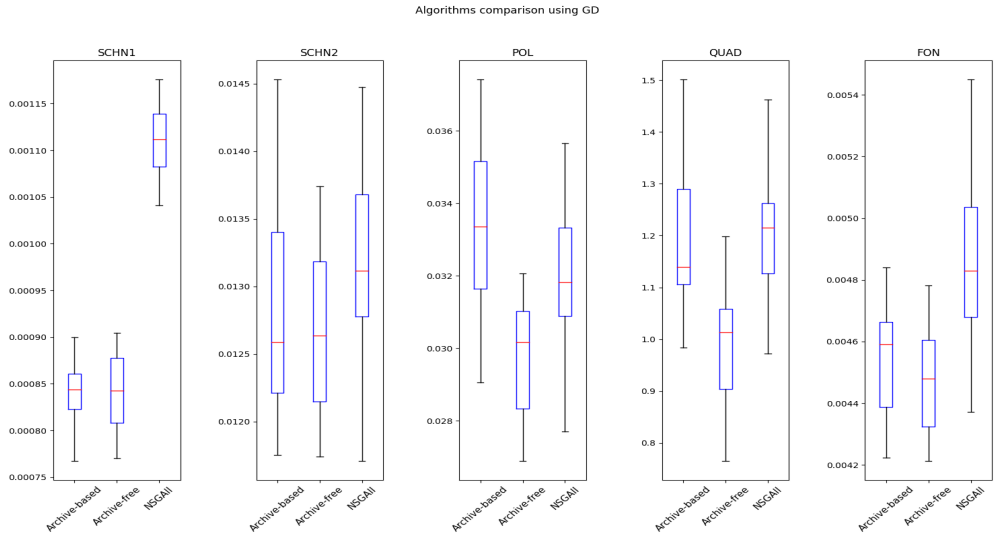


Figure 2.9 – Box plot (using GD) representing the comparison of the algorithms for unconstrained functions: SCHN1 , SCHN2, POL, QUAD, FON.

Table 2.2 – Comparative results (using Δ) of proposed archive-based, archive-free SA approaches with NSGA II obtained after twenty independent runs on benchmark multi-objective functions: SCHN1, SCHN2, POL, QUAD, FON.

Algorithm	SCHN1				SCHN2			
	Mean	SD	Best	Worst	Mean	SD	Best	Worst
Archive-based	0.39275	0.02380	0.34302	0.43290	1.01594	0.02449	0.97357	1.06629
Archive-free	**0.35921**	0.02322	0.31046	0.38939	**0.98399**	0.01534	0.95330	1.01561
NSGA II	0.36674	**0.02301**	0.32194	0.39779	0.99120	**0.01488**	0.96517	1.01685

Algorithm	POL				QUAD			
	Mean	SD	Best	Worst	Mean	SD	Best	Worst
Archive-based	0.94013	0.01142	0.92311	0.97720	0.42375	0.02956	0.34808	0.47220
Archive-free	**0.93182**	0.00862	0.92182	0.95684	**0.40076**	**0.02644**	0.36692	0.47861
NSGA II	0.94111	**0.00770**	0.92574	0.95387	0.43908	0.03304	0.38060	0.48988

Algorithm	FON			
	Mean	SD	Best	Worst
Archive-based	0.30562	**0.02097**	0.26727	0.34802
Archive-free	**0.27350**	0.02243	0.21923	0.30240
NSGA II	0.33212	0.03465	0.27762	0.39397

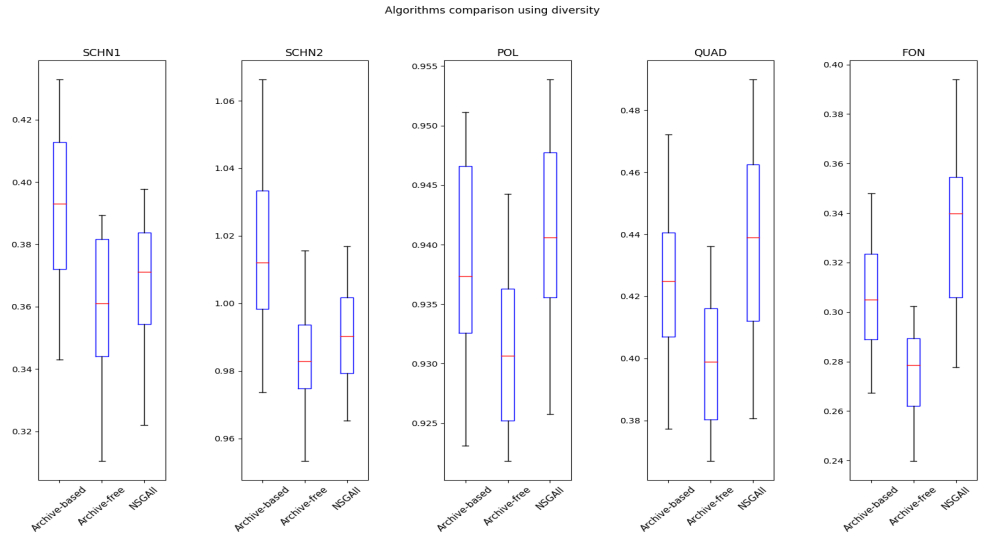


Figure 2.10 – Box plot (using Δ) representing the comparison of the algorithms for unconstrained functions: SCHN1 , SCHN2, POL, QUAD, FON.

Table 2.3 – Comparative results (using CT) of proposed archive-based, archive-free SA approaches with NSGA II obtained after twenty independent runs on benchmark multi-objective functions: SCHN1, SCHN2, POL, QUAD, FON.

Algorithm	SCHN1				SCHN2			
	Mean	SD	Best	Worst	Mean	SD	Best	Worst
Archive-based	96.12842	0.74248	94.79008	98.14508	85.40642	0.75281	84.57827	88.05640
Archive-free	5.69665	0.05147	5.61246	5.84559	5.00207	**0.01644**	4.97520	5.05383
NSGA II	**4.24316**	**0.03447**	4.17738	4.31883	**4.08061**	0.01850	4.03077	4.11573

Algorithm	POL				QUAD			
	Mean	SD	Best	Worst	Mean	SD	Best	Worst
Archive-based	53.28598	2.06484	49.79741	56.92427	83.56412	0.68213	82.18945	84.78586
Archive-free	4.26086	**0.03385**	4.21650	4.33138	4.32710	**0.04034**	4.27662	4.43632
NSGA II	**4.18612**	0.05853	4.11994	4.36026	**4.14793**	0.04049	4.10133	4.30211

Algorithm	FON			
	Mean	SD	Best	Worst
Archive-based	13.34312	1.15308	11.01524	15.40951
Archive-free	4.35912	0.06945	4.27545	4.57798
NSGA II	**4.29131**	**0.03376**	4.22335	4.35721

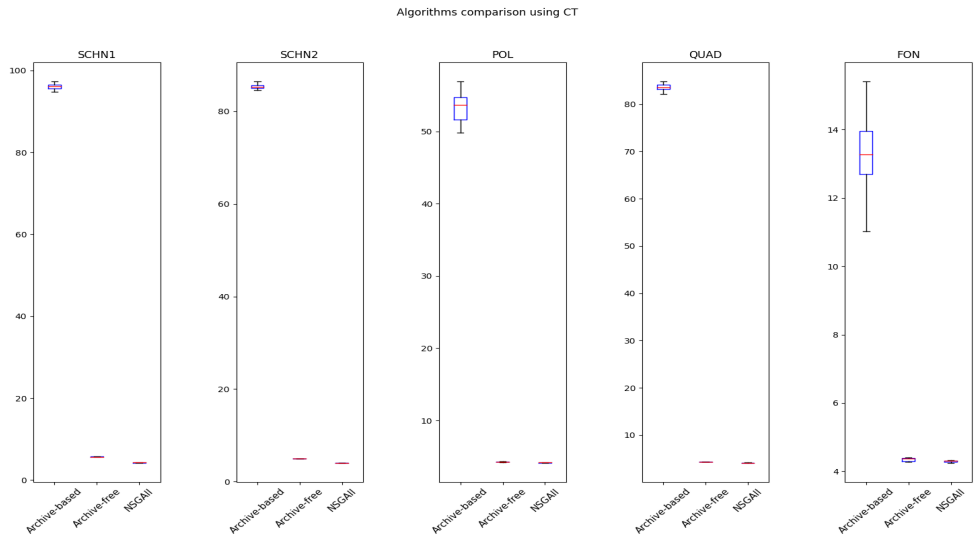


Figure 2.11 – Box plot (using CT) representing the comparison of the algorithms for unconstrained functions: SCHN1 , SCHN2, POL, QUAD, FON.

NSGA II in terms of the Δ, which indicates the solutions obtained by archive-free SA are spaced more evenly.

The box plot for the above three performance metrics (GD, Δ and CT) for twenty independent runs of each algorithm signify the distribution of each algorithm, is shown in Fig. 2.9, Fig. 2.10, Fig. 2.11. Note that the red lines represent the median, while the blue rectangles indicate the average performance. The box plot is an effective visualization tool to compare the performance of the proposed and comparative algorithms. It is observed that the computational time taken by the algorithms (Archive-free SA, NSGA II) over independent runs do not deviate much where are the performance metrics (GD and Δ values) deviate. Observed lower median and average performance values in GD and Δ justifies the superior performance of proposed archive-free SA. In the case of archive-based SA, the redundant computation makes it high cost to solve the complex problems.

2.3.2 Convergence resistance and improvement

The convergence resistance appears when there are many non-dominated points close to the Pareto front. Fig. 2.12 presents the Pareto set of the QUAD problem. It is observed that the obtained solution of archive-free SA deviates from the true Pareto set. Indeed, the convergence resistance exists in all the Pareto domination based algorithms. As the

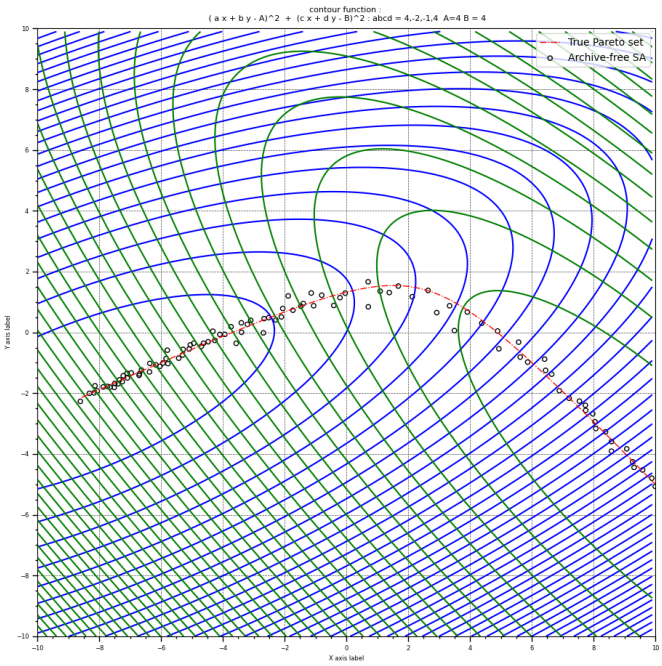


Figure 2.12 – Pareto set of QUAD in the contour plot.

optimization continues to approach the true Pareto front, the selection pressure increases with the number of non-dominated solutions. The problem is illustrated in Fig. 2.13.

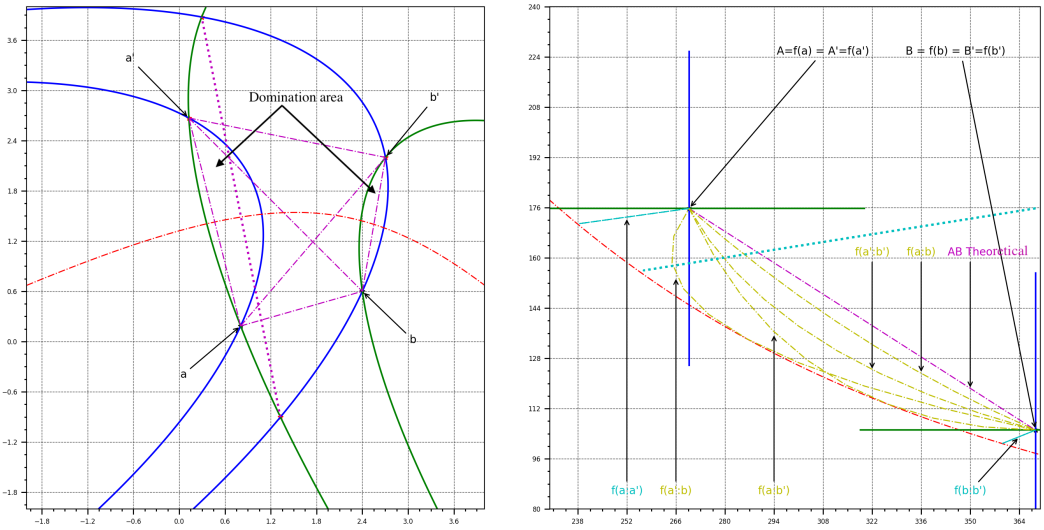


Figure 2.13 – Problem demonstration: (left) in design space, (right) in objective space.

For instance, point $A(')$ and $B(')$ are non-dominated to each other. The corresponding points in the design space are $a(')$ and $b(')$. To improve the convergence, it is possible to replace the point B with the better point, i.e., the solution dominates solution B. However, these solutions may situate in the narrow domination area, i.e. domination area of point B. Thus, the new generated neighborhood solutions have large possibility to be non-dominated to the current solution, such that the Pareto domination selection becomes of no use to improve the optimization convergence. To resolve the problem, we propose to use new selection criteria based on the distance measurement to improve the convergence.

After the pre-defined maximum iterations, the optimization may find an approximate Pareto front, i.e., the last population P in archive-free SA algorithm. Then use the distance as the selection reference, and push the optimization to the infeasible region P^{inf}. The overall idea is:

1. Define a fake target by shifting P towards infeasible domination region, expressed as $P^{inf} = P - \sigma$. In the QUAD minimization function, $\sigma > 0$.
2. Generate the new solution X_i' around current solution X_i, substituting X_i with X_i' that closest to the corresponding fake point in the objective space. For the generated solutions (red points), it should be not far from the current solution (green star), as shown in Fig. 2.14. The objective aims to minimize the distance between the infeasible solution (red star) and the new solution to help the optimization jump out the current narrow domination area and find a better solution, i.e. the blue star.

The new Pareto set of QUAD is presented in Fig. 2.15. Here, the optimization continues for an additional 100 iterations. It is proved that the obtained solutions are lying on the true Pareto set. The distance criteria improves the convergence without damaging the diversity. It does find better solutions but in the cost of function evaluations.

2.3.3 Multi-objective 0-1 Knapsack problem

The goal of a combinatorial optimization problem is to find the optimal ordering / permutation of a set of discrete items. A standard example is known as the knapsack problem that involves the profit maximization. The objective could have several dimensions instead of one. As an example, suppose there are two objectives, maximizing the profit while minimizing the number of items.

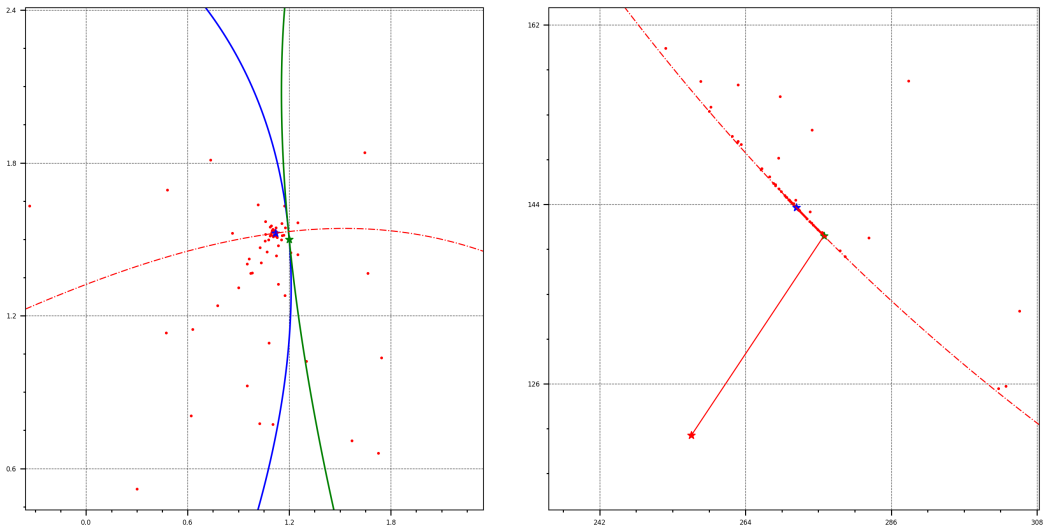


Figure 2.14 – Convergence improvement illustration: (left) in design space, (right) in objective space. red star: fake point; green star: current solution; blue star: improved solution; red point: new solution

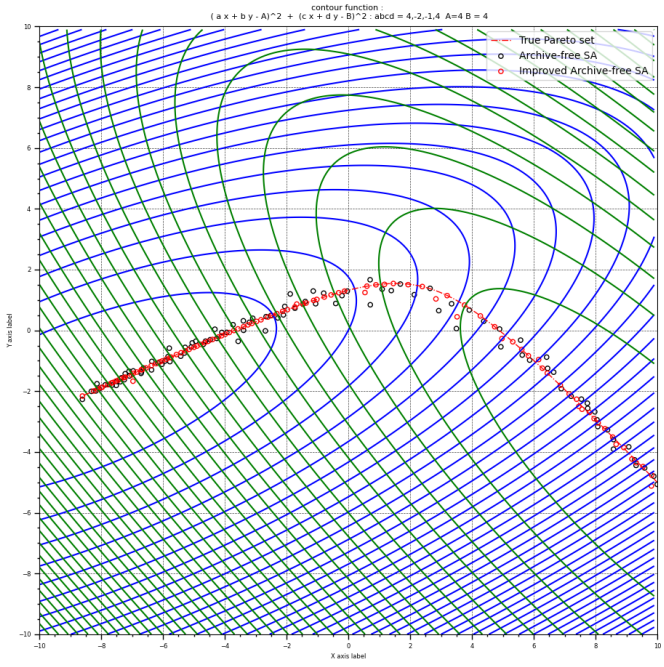
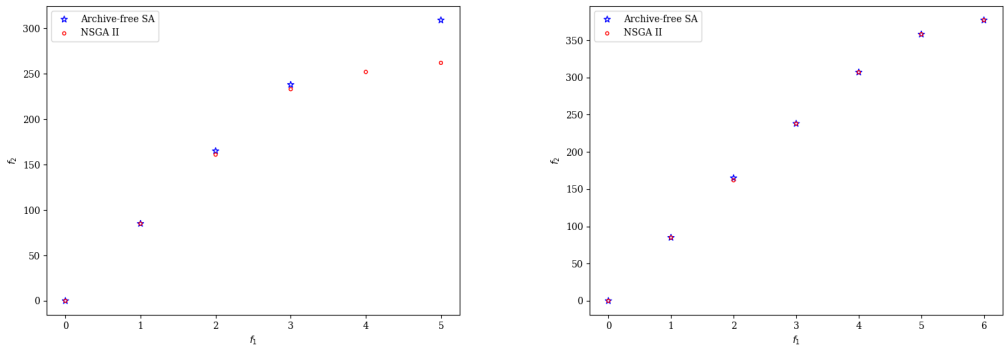


Figure 2.15 – Pareto set of QUAD function with improved convergence.

Given a set of n items and one knapsack, the multi-objective 0–1 knapsack problem (MOKP) can be stated as:

$$\begin{cases} \text{Minimize } f_1(x) = \sum_{i=1}^{n} x_i \\ \text{Maximize } f_2(x) = \sum_{i=1}^{n} p_i x_i \\ \text{subject to} \sum_{i=1}^{n} w_i x_i \leq c \\ x = (x_1, \ldots, x_n)^T \in \{0,1\}^n \end{cases} \tag{2.11}$$

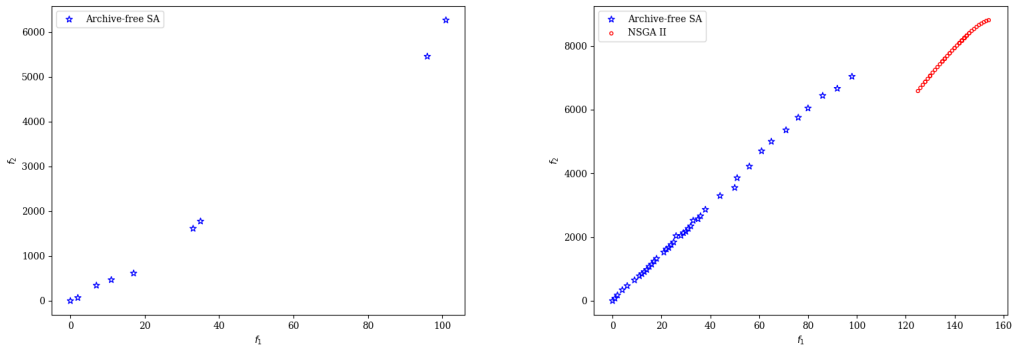
where p_i is the profit of item i, w_i is the weight of item i, and c is the capacity of knapsack. $x_i = 1$ means that item i is selected and put into the knapsack. The MOKP is a NP-hard problem which theoretically cannot be solved to optimality in a reasonable time. SA is a meta-heuristic which can compute approximate solutions to quite any NP-hard problem. Other meta-heuristics, i.e., genetic algorithms, start from a random or "greedy" solution, then improve it by changing some local parts of the solution in order to improve it. In this section, we consider five combinatorial instances for comparing the performances of archive-free SA and NSGA II, where the profit p_i and weight w_i are integers between 1 and 100, the capacity c is one tenth of the sum of weights.



(a) (b)

Figure 2.16 – 20 items knapsack: (a) $N = 10$, (b) $N = 100$.

To apply archive-free SA, the neighbor generation is defined as uniform random addition or removal of one item in the knapsack. Besides, we integrate the repair process for the infeasible solutions, that is, randomly remove the first or the last item. This section also experimentally investigates the effect of population size $N = 10, 100$. Table 2.4 summarizes the average computation time used by each algorithm for each instance with different number of items. With the same number of the iterations, it is evident from



(a) (b)

Figure 2.17 – 1000 items knapscak: (a) $N = 10$, (b) $N = 100$.

Table 2.4 – Comparative results (using CT) of proposed archive-free SA approaches with NSGA II obtained on benchmark multi-objective 0-1 knapsack: 10, 20, 30, 100, 1000 items.

Algorithm		10 items	20 items	30 items	100 items	1000 items
N=10	Archive-free	**0.54148**	**0.45314**	**0.42556**	**0.39477**	**0.41321**
	NSGA II	32.56178	5.18765	8.76503	20.68205	76.33751
N=100	Archive-free	**9.92223**	**8.23018**	**6.81624**	**5.91139**	**6.75522**
	NSGA II	228.64363	40.89791	29.19269	35.23269	143.34170

Table 2.4 that archive-free SA needs less computational time than NSGA II does. On average, archive-free SA requires about 9% of the computation time that NSGA II needs. In other words, archive-free SA is ten times faster than NSGA II. Besides, it is proved that SA algorithm has better exploration ability even with small population $N = 10$ in a rather large search space, as shown in Fig. 2.16 (a). And Fig. 2.17 (a) clearly indicates that archive-free SA successfully find approximate solutions whereas NSGA II failed. Furthermore, since NSGA II tries to improve the quality of the obtained solutions, it can solve the low-dimensional problem as shown in Fig. 2.16 (b), but a tendency to cluster appears as the number of items increases, as presented in Fig. 2.17 (b). It suggests that archive-free SA is more robust and effective than NSGA II for the combinatorial problem MOKP. Overall, we can claim that archive-free SA is computationally much cheaper and can produce better approximations than NSGA II on these MOKP test instances.

2.4 Conclusion

The ease of implementation makes SA as a popular method for solving large and practical problems. However, the individual-based feature limits the application, especially in the multi-objective problems. In this chapter, we have proposed to extend scalar SA algorithm to solve multi-objective optimization problems, named archive-based SA and archive-free SA algorithms.

Archive-based SA is designed using the new acceptance mechanism, where the solutions are divided into dominate, non-dominated, dominated states. It adopts the limited-size archive to keep the non-dominated solutions and applies the second annealing loop to the final archive. The use of archive brings the substantial benefits, but at the cost of computation time.

Archive-free SA employs the dynamic selection based on the concept of non-dominated sort and the mechanism of crowding distance calculation. The dynamic selection preserves the non-domination and distributed solutions in the population. It is a simple algorithm with the advantage of ease of implementation.

With the help of GD, Δ and CT metrics, experiments demonstrate the performance of the proposed algorithm. Five continuous benchmark functions have presented that proposed archive-free SA with lower computation time outperforms archive-based SA algorithm and NSGA II. Furthermore, an experimental study of convergence resistance and improvement has also been carried out. In the comparison of combinational problems, it has been shown that the SA-based method is superior in multi-objective 0–1 knapsack problems. Overall, the results proved that archive-free SA can provide, most of the time, very competitive results compared to others.

Chapter 3

Multi-objective layout problem model and interaction

3.1 Introduction

As explained in Chapter 1, the model of the layout optimization problem is a multi-objective formulation. The model aims to translate the designer's requirements into variables, constraints and optimization objectives. In general, the layout problem is locating rectangular components in the rectangular container. Based on the different functional properties of the components, the novel component including the solid and virtual parts are defined, and the accessibility to the components of a layout is explicitly expressed. Then, a new capacity index is defined to measure the feasible complexity of a layout problem and provide a priori information on whether the problem can be solved. An innovative approach is proposed to evaluate the capacity by finding the minimum occupied space of components. It is worth noting that we integrate a novel space generation method that rendering the problem as a combinatorial packing process.

Indeed, the multi-objective formulation of the studied layout optimization problem leads to the generation of a set of solutions which achieve the best compromise between all the optimization objectives. Therefore, it is necessary to provide the designer with aiding methods and tools to help him to choose the solution that best suits his personal expertise and judgment on the layout problem. The interactive environment, detailed in this chapter, first of all, consists in the interactivity with the optimization problem. The purpose of this interactivity is to modify the problem manually and locally by taking into account the judgment and expertise of the designer. Also, an innovative indicator, allowing to analyze the similarity between solutions generated by the optimization algorithm, is notably defined. Finally, the last part of this chapter is dedicated to the decision-making based on a multidimensional visualization of solutions and their performance comparisons.

To formulate the model, we define the following notations:

3.2 Multi-objective layout problem model

In most layout problems, the problem formulation comprises the various components, the geometrical and functional constraints, the qualitative and quantitative objectives, as described in Fig. 3.1.

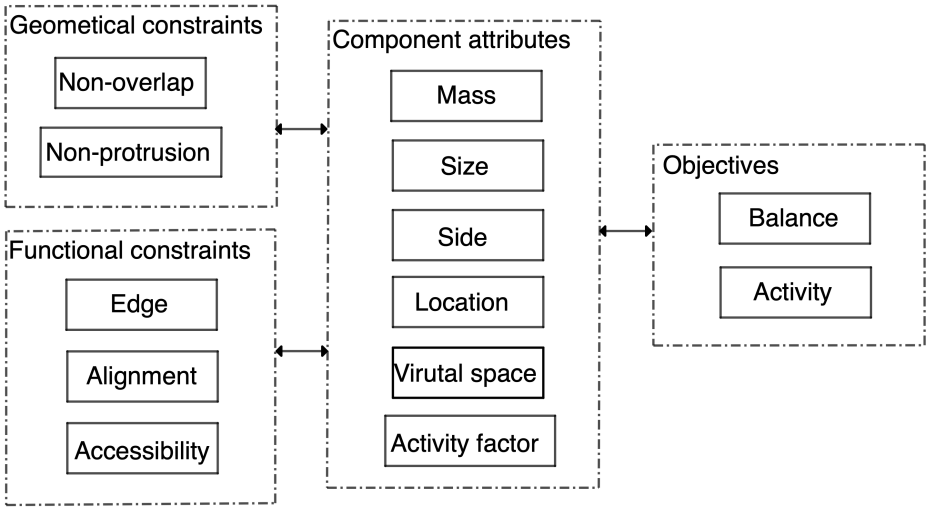


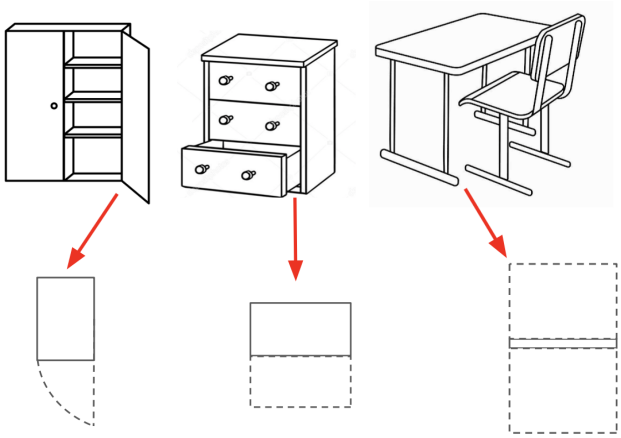
Figure 3.1 – Problem formulation.

3.2.1 Component definition

The layout problem aims to find the optimal arrangement of components in the 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. 3.2(a), the space of the drawer to pull it out in Fig. 3.2(b), and the space of desk to allow the user sit down as shown in Fig. 3.2(c) etc. Indeed, the cabinet, desk and drawer are modeled by solid components because they cannot overlap. These virtual spaces placed around them, are modedly by virtual components, can overlap each other, due to the non-simultaneous use. Moreover, a collapsible object is designed to be folded flat when 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. In this case, the decomposition of the components of the layout into solid and virtual components makes it

possible to take into account the particular requirement expressed by the designer. Based on the analysis above, the different components are summarized for the layout problem formulation:

1. Virtual components could overlap virtual components and have no mass. These virtual components are also the accessible spaces.
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 virtual and solid components.



(a) Cabinet (b) Drawer (c) Desk

Figure 3.2 – Component examples.

The component definition makes it possible to model most layout problems, for instance, the machine placement problem in the factory. The virtual components can model the free space around the machine that are necessary for the correct operation or maintenance, or represent the corridors used for the flow of goods between equipment. It is important to make this classification of the components because it can describe all the problems of layout in a generic and realistic way, whatever the requirements expressed by the designer.

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 components v_{ij} (dashed rectangles), as shown in Fig. 3.3. 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 dashed rectangle size (w_{ij}, h_{ij}). 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. Rectangular components are relatively common in the literature, and practical layout problems can be simplified to rectangles.

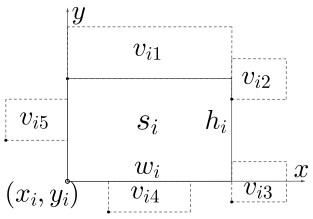
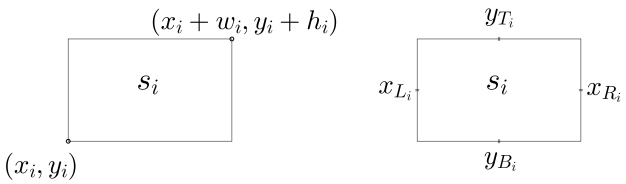


Figure 3.3 – Component c_i representation.

To simplify the model, we also introduce the side location. The side location of component s_i can be extracted from two extreme corners' coordinates $(x_i, y_i, x_i{+}w_i, y_i{+}h_i)$ in Fig. 3.4 (a). Each component s_i has four sides as shown in Fig. 3.4 (b), 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 $y_{T_i} = y_i + h_i$



(a) Corners' coordinates (b) Side location coordinates

Figure 3.4 – Component s_i side location representation.

3.2.2 Geometrical and functional constraints

The geometrical constraints of the layout problem are non-overlap and non-protrusion constraints. In reality, the rectangular shape of the component simplifies the constraints formulation. Due to the accessibility property, overlap between virtual components is

allowed, whereas overlap between two solid components has to be minimized. Thus, the mathematical intersection area between two components is defined as:

$$\begin{aligned} a_{ik} = &\max\left[0, \min\left(x_i + w_i, x_k + w_k\right) - \max\left(x_i - w_i, x_k - w_k\right)\right] \\ &\times \max\left[0, \min\left(y_i + h_i, y_k + h_k\right) - \max\left(y_i - h_i, y_k - h_k\right)\right] \end{aligned} \tag{3.1}$$

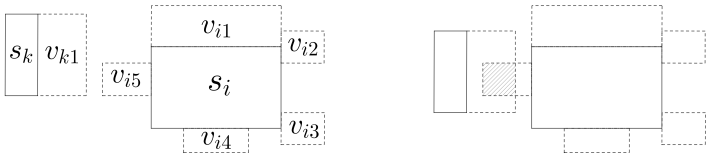
$$\begin{aligned} a_{kj} = \sum_{j}^{n_i} &\max\left[0, \min\left(x_{ij} + w_{ij}, x_k + w_k\right) - \max\left(x_{ij} - w_{ij}, x_k - w_k\right)\right] \\ &\times \max\left[0, \min\left(y_{ij} + h_{ij}, y_k + h_k\right) - \max\left(y_{ij} - h_{ij}, y_k - h_k\right)\right] \end{aligned} \tag{3.2}$$

$$\begin{aligned} a_{ij} = \sum_{j}^{n_k} &\max\left[0, \min\left(x_{kj} + w_{kj}, x_i + w_i\right) - \max\left(x_{kj} - w_{kj}, x_i - w_i\right)\right] \\ &\times \max\left[0, \min\left(y_{kj} + h_{kj}, y_i + h_i\right) - \max\left(y_{kj} - h_{kj}, y_i - h_i\right)\right] \end{aligned} \tag{3.3}$$

a_{ik} represents the intersection area between component s_i and component s_k. a_{kj} represents the sum of intersection area between solid component s_k and virtual component $v_{ij}, j = 1, ..., n_i$, similarly, a_{ij} is the sum of intersection area between solid component s_i and virtual component $v_{kj}, j = 1, ..., n_k$. So the non-overlap constraint of component c_i and component c_k is defined as follows:

$$A_{ik} = a_{ik} + a_{kj} + a_{ij} \leq 0 \tag{3.4}$$

Given two components $c_i = (s_i, v_{i1}, v_{i2}, v_{i3}, v_{i4}, v_{i5})$ and $c_k = (s_k, v_{k1})$, the overlap of virtual space is possible, as shown in Fig. 3.5, One case of non-overlap constraint is presented in Fig. 3.6, neither solid components overlap a_{ik} in Fig. 3.6 (a) nor solid-virtual components overlap a_{kj}, a_{ij} in Fig. 3.6 (b) (c), is allowed.

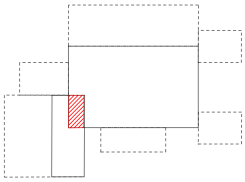


(a) Components (s_k, v_{k1}) and $(s_i, v_{i1}, v_{i2}, v_{i3}, v_{i4}, v_{i5})$

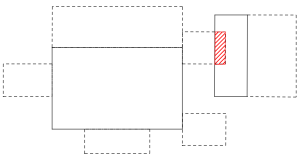
(b) Allowed overlap between virtual spaces

Figure 3.5 – Overlap between virtual components representation.

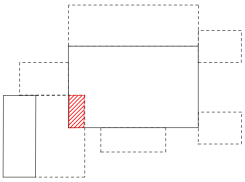
The non-protrusion constraint expresses the fact that the component (i.e. s_i) should stay inside one bounded rectangular space a, denoted as $a = (x_a, y_a, w_a, h_a)$, defined by



(a) Overlap a_{ik} between solid components s_i, s_k



(b) Overlap a_{kj} between the solid component s_k and virtual components v_{ij}



(c) Overlap a_{ij} between the solid component s_i and virtual components v_{kj}

Figure 3.6 – Non-overlap constraint representation.

the bottom left coordinates and sizes, as shown in Fig. 3.7. The component side locations

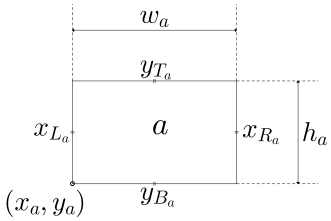


Figure 3.7 – Available space a representation.

$(x_{L_i}, y_{B_i}, x_{R_i}, y_{T_i})$ and available space side locations $(x_{L_a}, y_{B_a}, x_{R_a}, y_{T_a})$ should satisfy:

$$\max\{x_{L_a} - x_{L_i}, x_{R_i} - x_{R_a}, y_{T_i} - y_{T_a}, y_{B_a} - y_{B_i}\} \leq 0 \tag{3.5}$$

In some design problems, i.e. C & P problem, constraints are only geometrical. The functional constraints, including the edge and alignment, specify the functional requirements of components and guarantee the correct function of components in the layout problem. The basic idea is to translate these requirements into the generic mathematical functions.

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 used to force the component against the edge of the space because of a window or door. It is supposed that the component s_i is inside a rectangular space $a = (x_a, y_a, w_a, h_a)$, in order to force s_i to the side of a, 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:

$$\min\{(x_{L_a} - x_{L_i})^2, (y_{B_i} - y_{B_a})^2, (x_{R_i} - x_{R_a})^2, (y_{T_i} - y_{T_a})^2\} = 0 \tag{3.6}$$

Also, if the alignment of one side of component s_i to a particular side of component s_j is important, one of the following constraints should be added:

$$\begin{aligned} x_{L_i} &= x_{L_j} \quad or \quad x_{R_j} \\ x_{B_i} &= x_{B_j} \quad or \quad x_{T_j} \\ x_{R_i} &= x_{R_j} \quad or \quad x_{L_j} \\ x_{T_i} &= x_{T_j} \quad or \quad x_{B_j} \end{aligned} \tag{3.7}$$

Furthermore, certain functional constraints are difficult to translate into explicit mathematical functions: components are accessible from the container's entry which can be found in many layout problems, such as the layout of equipment in a room, the placement of machines in a factory, and the assembly of mechanical parts. On the one hand, virtual spaces associated with solid components represents the accessibility of component. On the other hand, virtual spaces may be inaccessible if there is no path to access it. The layout optimization algorithm in Chapter 4 proposes a method to address this special requirement and integrate it into the optimization procedure as a constraint.

3.2.3 Multi-objective formulation

The layout problem is a multi-objective problem. The first objective f_1 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:

$$f_1 = \sqrt{\left(X_{gra} - X'_{gra}\right)^2 + \left(Y_{gra} - Y'_{gra}\right)^2} \tag{3.8}$$

$$X_{gra} = \frac{\sum_{i=1}^{n} (x_{c_i} \times m_i)}{\sum_{i=1}^{n} m_i}, Y_{gra} = \frac{\sum_{i=1}^{n} (y_{c_i} \times m_i)}{\sum_{i=1}^{n} m_i} \tag{3.9}$$

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 relative position of components. In order to quantitatively describe the activity relationship between components, an activity factor is designed according to expert judgement 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:

$$f_2 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} \times \omega_{ij} \tag{3.10}$$

$$d_{ij} = \sqrt{\left(x_{c_i} - x_{c_j}\right)^2 + \left(y_{c_i} - y_{c_j}\right)^2} \tag{3.11}$$

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

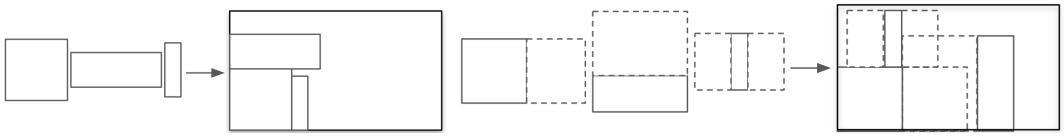
The multi-objective layout problem aims to find the arrangement (location and orientation) of components $\boldsymbol{c} = \{c_1, c_2, ..., c_n\}$, optimize objectives f_1, f_2 and satisfy geometrical and functional constraints, the formulation can be expressed as:

$$\begin{cases} variable & \boldsymbol{x} = \boldsymbol{c} \\ min & \boldsymbol{f}(\boldsymbol{x}) = f_1(\boldsymbol{x}), f_2(\boldsymbol{x}) \\ s.t. & \boldsymbol{g}(\boldsymbol{x}) : Eq.3.4, Eq.3.5, Eq.3.6, Eq.3.7 \end{cases} \tag{3.12}$$

3.3 Capacity index of layout problem

For a layout problem, 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 problem. This section proposes a method to define the compactness, namely capacity, of a layout problem and provides a priori information on the difficulty of problem solving.

During the conceptual design phase of the layout, the area occupied by components should be less than the container area. One example of three solid components packing is shown in Fig. 3.8(a). The occupied area of the solid components can be defined by the



(a) Solid components packing (b) Solid and virtual components packing

Figure 3.8 – Component packing.

sum of the area, and the density of solid components is expressed as:

$$\beta_s = \frac{\sum_{i=1}^{n} (w_i \times h_i)}{W \times H} \tag{3.13}$$

W, H are the size of the container space. If there are virtual components, then the density of the virtual ones is:

$$\beta_v = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n_i} (w_{ij} \times h_{ij})}{W \times H} \tag{3.14}$$

And the sum of components density is:

$$\beta_{sv} = \beta_s + \beta_v \tag{3.15}$$

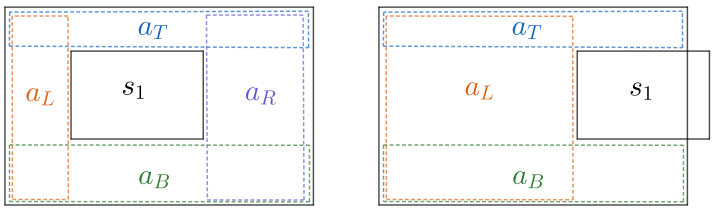
However, the overlap of virtual components is acceptable and the sum of the dimensions of the components does not consider the virtual component property. Thus, the density index will exaggerate the feasibility complexity and incorrectly indicates that the problem can not be solved (i.e. $\beta_{sv} > 1$). Therefore, it is necessary to define a reliable index that shows the problem feasibility taking into account the solid and virtual components.

One previous work estimated the feasibility using the intersection matrix with no geometry included [122]. To be more reliable, we define a capacity index β_c measuring the minimum occupied space of a given number of solid and virtual components. Indeed, the objective is to find the index value that corresponds to the compact layout configuration, if it is greater than 1, it indicates that the layout problem cannot be solved properly.

In order to find the minimum occupied space of components, we formulate the problem as a packing process. On the one hand, the main idea of packing is to maximize space utilization, in other words, to place all components as compact as possible. On the other hand, the packing process can be treated as a combination problem that constructively packs the components in a given sequence. Suppose we have 8 components, the number of permutations equals $8! = 40320$. Exploring all possibilities is quite time consuming. To resolve the problem, we propose the simulated annealing method for constructive packing. The SA optimizes the placement order; the construction places components that addresses the geometrical constraints and determines the position of components that minimize the occupied space. One example of three components packing is shown in Fig. 3.8(b). To pack components, we need to generate a packing order and strategy.

3.3.1 Space generation

The evaluation of the non-overlap constraint must be optimized in order to reduce the computational time. Thus, we propose a novel space generation algorithm and place the components with respect to the constraints that ensuring 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 Left $a_L = (x_{a_L}, y_{a_L}, w_{a_L}, h_{a_L})$, Right $a_R = (x_{a_R}, y_{a_R}, w_{a_R}, h_{a_R})$, Top $a_T = (x_{a_T}, y_{a_T}, w_{a_T}, h_{a_T})$ and Bottom $a_B = (x_{a_B}, y_{a_B}, w_{a_B}, h_{a_B})$, as shown in Fig. 3.9(a). In contrast, if the component space and available space partially intersect, some candidate available space may not exist, for example in Fig. 3.9(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.



(a) Complete included case (b) Partial intersected case

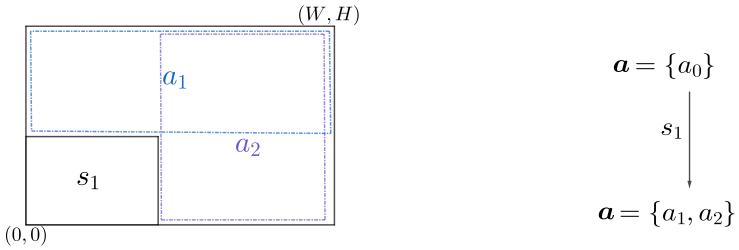
Figure 3.9 – Complete and partial space generation.

The space generation is inspired from [123]. The former algorithm was developed for the cutting problem and the virtual components are not considered. To account for non-overlap constraint of the component definition that involves the solid and virtual components, we propose an effective non-overlap constraint evaluation method. As mentioned before, the overlap of solid components is 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:

$$\forall i \in [1, k], \exists j \in [1, m], \quad a'_i \sqsubseteq a_j \tag{3.16}$$

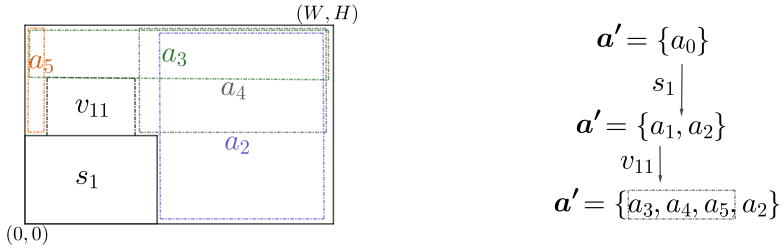
m and k represent the number of spaces in $\boldsymbol{a}$ and $\boldsymbol{a}'$. New virtual components will be placed in $\boldsymbol{a}$ to benefit overlap between virtual components, while new solid components will be placed in $\boldsymbol{a}'$ to guarantee non-overlap of solid components. After 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.

The space generation process of list $\boldsymbol{a}$ is described in Algorithm 4. The space generation checks the available space and component space that intersects in line 2. There are four possible generated spaces in line 6. If any available space is included in the generated spaces, remove the available space from the list in line 9. In contrast, set $a_{dir} = \emptyset$ if it is totally included in one of the available spaces in line 11. The updating aims to release computer memory storage space. And the same generation mechanism for $\boldsymbol{a}'$, except that the input is $(\boldsymbol{a}', c_i)$ and the output becomes $\boldsymbol{a}'$.



(a) Space generation of $\boldsymbol{a}$ (b) Tree of $\boldsymbol{a}$

Figure 3.10 – Space generation of s_1.



(a) Space generation of $\boldsymbol{a}'$ (b) Tree of $\boldsymbol{a}'$

Figure 3.11 – Space generation of (s_1, v_{11}).

Fig. 3.10 illustrates one space generation of s_1. 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,

Algorithm 4 Space generation

Input: current list $\boldsymbol{a}$, component s_i
Output: $\boldsymbol{a}$

```
1: for a in a do
2:   if a intersects s_i then
3:     for each side of s_i do
4:       /* x_{L_i}, x_{R_i}, x_{T_i}, x_{B_i} */
5:       if the side is included in a then
6:         Create corresponding space a_dir, dir in {L, R, T, B}
7:         for a in a do
8:           if a is included in a_dir then
9:             Remove a from a
10:          else if a_dir is included in a then
11:            a_dir = ∅, go to line 14
12:          end if
13:        end for
14:        if a_dir then
15:          a ∪ a_dir
16:        end if
17:      end if
18:    end for
19:  end if
20: end for
```

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. 3.10(b). Besides, the virtual component v_{11} placed in a_1 generates new available spaces $\{a_3, a_4, a_5\}$ in Fig. 3.11(b). 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, that is, Eq. 3.4 and Eq. 3.5.

3.3.2 Simulated annealing and constructive packing optimization

The idea of simulated annealing and constructive packing algorithm will be presented. It is worth noting that the SA considers the capacity β_c as the objective and placement order $\boldsymbol{c}$ as the variable X; the constructive packing integrates the space generation and constructs the compact configuration. The overall idea is described in Algorithm 5.

Algorithm 5 Simulated annealing and constructive packing

Input: X, X^*=X
Output: X^*

$\beta_c(X)$ = Pack (X)
while stop condition not met **do**
  while *iteration* in inner loop **do**
    X'=$X(\sigma)$ `/* Swap two elements of` X `*/`
    $\beta_c(X')$ = Pack (X')
    $\delta\beta_c = \beta_c(X')$-$\beta_c(X)$
    $p(\delta) = e^{-\delta\beta_c/t}$
    if $\delta\beta_c < 0$ or $p(\delta) > rand$ **then**
      $X = X'$
      $\beta_c(X)$=$\beta_c(X')$
      if $\beta_c(X') < \beta_c(X^*)$ **then**
        $X^* = X'$
      end if
    end if
  end while
  Decrease temperature t and step σ
end while

Here, the SA optimizes the state X in which the components are placed into the container, denoted as $X = \boldsymbol{c}$ and $X = (c_1, ..., c_n)$. In each iteration of inner loop, a new state X' will be generated by swapping elements. During the neighbor generation, 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:

$$\sigma = n * exp^{(-1/t)} \tag{3.17}$$

The integer parameter $\sigma \in [1, n], n$ is the number of components, determines the process of neighbor generation. 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 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 and reduced with the cooling rate α in the outer loop, $t = t * \alpha$. The optimization will stop if it reaches to the final temperature or it is

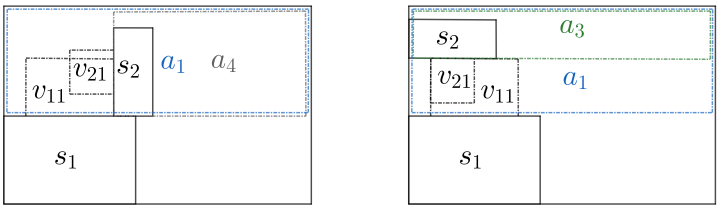
up to the number of iterations.

The constructive packing aims to construct the compact layout configuration by successively selecting and placing new components following the placement order. During the previous space generation, there will be multiple available spaces that can be used for the new component. In order to place the components compactly, the following conditions should be satisfied:

— Placement: The new solid component closes to these already placed solid components according to the bottom left convention.

— Selection: The overlap between the new virtual components and placed virtual components should be maximized.

To place a new component into the container, two criteria need to be measured: the size of the available space and the overlap of virtual space. Due to the integration of the novel space generation, the evaluation of geometrical constraints is more effective. Fig.3.12 illustrates an example of packing two components. The detailed steps of constructive



(a) Component placement in (a_4, a_1)

(b) Component placement in (a_3, a_1)

Figure 3.12 – Example of component placement.

packing are described as follows:

Step 1 : Initialize the available space in $\boldsymbol{a}$ and $\boldsymbol{a}'$ to the container space.

Step 2 : Place the first component into the bottom left corner of the container by convention and calculate the coordinates and size of the attached virtual component. Update the lists $\boldsymbol{a}$ and $\boldsymbol{a}'$ (i.e. placement of s_1, v_{11} in Fig.3.11).

Step 3 : Place the new component sequentially according to the placing order. i.e. place the second component s_2 and virtual components v_{21}. Enumerate all permutations of elements in the lists $\boldsymbol{a}'$ and $\boldsymbol{a}$, for each pair of space $(a', a), a' \in \boldsymbol{a}', a \in \boldsymbol{a}$, check four

rotations of the component and filter some pairs that does not satisfy the size requirement such as $(a_5, *)$. The illustrative example in Fig.3.12(a) uses (a_4, a_1). Suppose the component s_2 is placed to space of a_4 that closes to the component s_1. Then check if the coordinates (x_2, y_2) are inside the space a_4 by computing the relation of the pair of space (a_4, a_1) and the size of virtual component v_{21} through Eq.3.18. If the pair of space (a_4, a_1) satisfies Eq.3.18, then we compute the placement that closes to s_1 in the space (a_4, a_1) by Eq.3.19 and deduce the relative coordinates of v_{21} with this configuration.

$$\begin{cases} x_{a_4} \leq x_{a_1} + max(w_{21}, x_{a_4} - x_{a_1}) + w_2 \leq x_{a_4} + w_{a_4} \\ max(0, min(y_{a_4} + h_{a_4}, y_{a_1} + h_{a_1}) - max(y_{a_4}, y_{a_1})) \geq h_{21} \end{cases} \tag{3.18}$$

$$\begin{cases} x_2 = x_{a_1} + max(w_{21}, x_{a_4} - x_{a_1}) \\ y_2 = max(y_{a_4}, y_{a_1}) \end{cases} \tag{3.19}$$

Another example with pair of space (a_3, a_1) is shown in Fig.3.12(b), the placement has a 90° rotation compared to Fig.3.12(a). Determining the placement is rather straightforward by swapping the size of component c_2 and following the same idea of Eq.3.18 and Eq.3.19. In this way, the placement closes to the already placed components and satisfies non-overlap constraints.

If there is feasible solution, record the temporary placement and orientation of components. If there are more than one feasible candidates, select the placement that maximizes the overlap area between virtual spaces, here, the placement in Fig.3.12(b) is prior.

Step 4 : Update the lists $\boldsymbol{a}$ and $\boldsymbol{a}'$. Repeat placing new components until a complete layout is finished. Otherwise, marked the placing order as infeasible.

3.3.3 Capacity evaluation

For the layout problem that consists of solid and virtual components, we define the index of capacity as a function of the available space that evaluated by:

$$\beta_c = 1 - \frac{\sum \boldsymbol{a}'}{W \times H} \tag{3.20}$$

It is worth noting that the $\sum \boldsymbol{a}'$ represents the available space without intersection. Besides, when we place the components sequentially, it may generate some small spaces, which are empty, but no components can be placed. Indeed, these small spaces should be

identified as infeasible spaces.

We can deduce the relationship between the density and capacity as:

$$\beta_s \leq \beta_c \leq \beta_{sv} \tag{3.21}$$

In the layout 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.

3.4 Interaction environment

The designer plays an important role in the multi-objective optimization process, whether in the formulation of the optimization problem, the interaction with the algorithm, or the final decision-making. Therefore, an optimization design environment shown in Fig. 3.13, was created to allow the designer to interact with the layout problem during the optimization process, for example, the designer can change the number of the components inside the container. As the designer receives feedback of the optimization progression, the designer can add new components and suppress extra components, or select component that can be rotated or fixed in certain location. Overall, it allows the designer to define/save/import the layout problem as a project, and view/explore/save the optimization results. The graphical interface, is one of the tools present in the interaction layout optimization, developed within the framework of the Ph.D. The example considered in Fig. 3.13 refers to the layout optimization problem detailed in Chapter 5.

The interactive environment is developed based on the PyQt5 package and is accomplished using an object-oriented representation. It allows the designer to interact with physically relevant components during optimization. For a layout problem, it has four relevant interactive interfaces named $LayoutInteracface$, $NewProjectInterface$, $EditInterface$ and $OptimizationInterface$ respectively. The designer can interact with the components through $EditInterface$ to modify components' parameters. And the $NewProjectInterface$ allows the designer to create a new $Project$ through $ComponentData$ and $ContainerData$. $Optimization$ contains different optimization algorithms and the designer can define and modify optimization objectives and constraints in $OptimizationFunction$ and interact the optimization process through $OptimizationInterface$. Each time an optimization is performed, the program translates the object-oriented representation into a

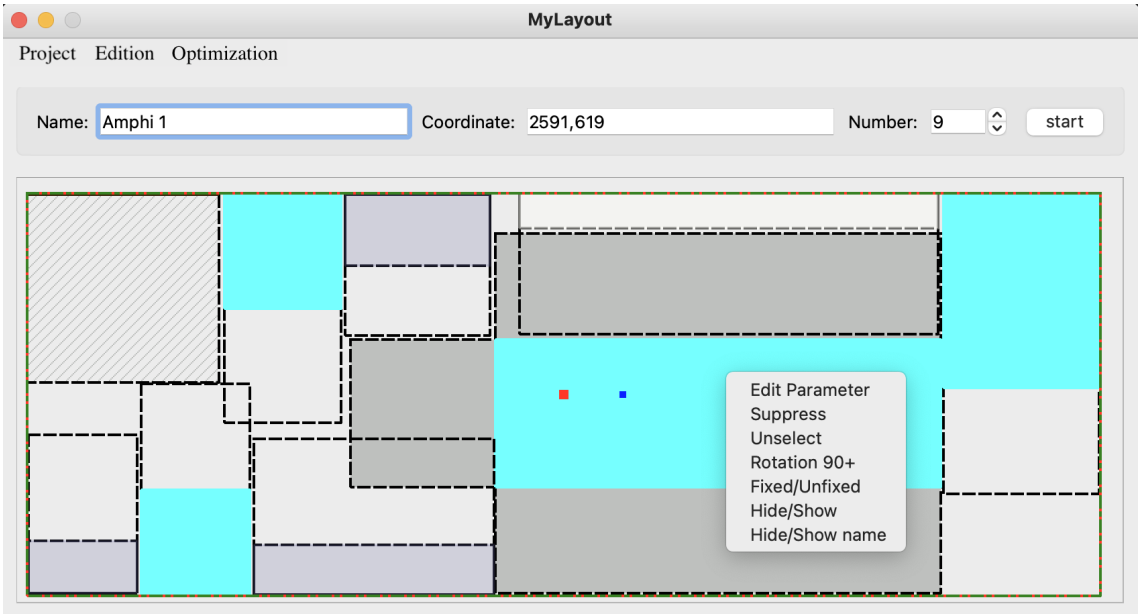


Figure 3.13 – Graphical interface representation.

set of mathematical design variables, objectives and constraints. The optimization results is displayed in $LayoutInteracface$. The environment structure is presented in Fig. 3.14.

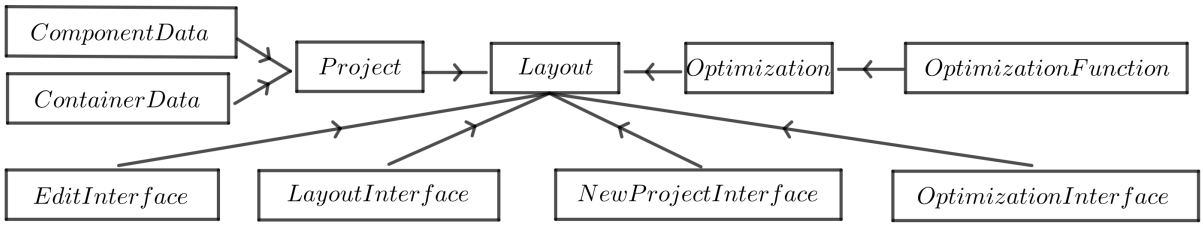


Figure 3.14 – Interactive configuration.

3.4.1 Interactivity with optimization problem

The interactive interface allows the designer to define a two-dimensional layout problem. The layout problem consists of the containers and rectangular components. For the container, the size can be varying according to the user defined width and height, as shown in Fig. 3.15.

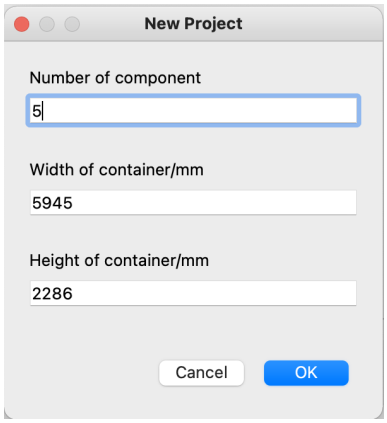


Figure 3.15 – New Project.

For each component, it has the solid component with mass (rectangle in color) and virtual component without mass (rectangle in dash line). Here the virtual component represents the space of accessibility. An edit environment has also been implemented in the graphical interface for viewing and editing the component in the layout problem. It is possible to modify the parameters of selected components (in grey) such as the name, mass, geometry and functional properties. Other functionalities are also offered in this interactive interface. It is notably possible to visualize, if they exist, the accessibility spaces of the components, to modify the virtual property of the components, to modify the geometrical model according to pre-defined constraints, rotation and attachment modes.

Besides, in order to help the designer perform the local optimization, three geometrical constraint modes are properly defined:

1. Cont_bd represents the container non-protrusion constraint where the component can be anywhere inside the container, expressed in Eq. 3.4.
2. DxDy_bd defines the distance dx, dy that the component can be moved from the current position, as shown in Fig. 3.17 (a).
3. Specific_bd stands for the rectangle space Lbx, Lby, Ubx, Uby ,in which the component can be located, as shown in Fig. 3.17 (b).

As the designer receives feedback of the optimization progression, he may want to change the definition of the geometrical constraints. If the optimization progresses into an undesirable area of the design space, the designer can dynamically add new constraints to prevent search in that area, i.e. DxDy_bd mode or Specific_bd mode. After seeing the optimization progression, the designer may decide to remove certain constraints in

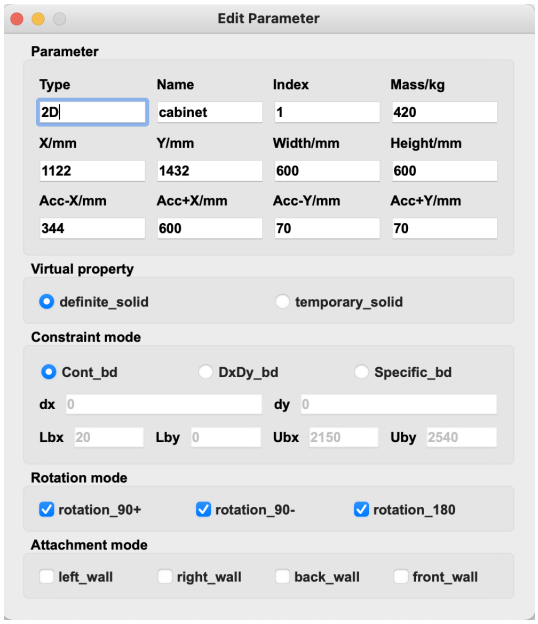
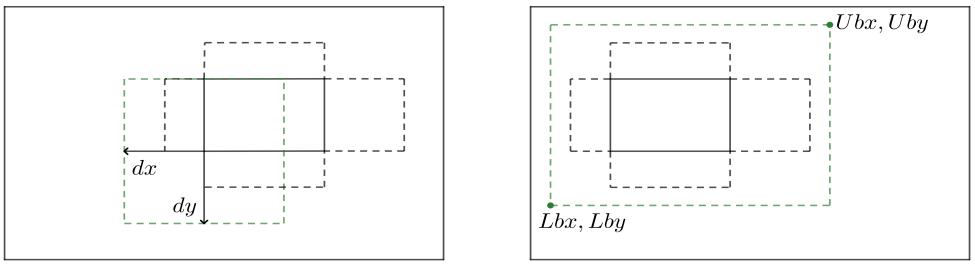


Figure 3.16 – Edit parameter.

order to achieve a better solution. Also, some constraints can be relaxed by modifying a numerical bound, such as a container mode constraint. Once the designer has seen some feasible design alternatives, he may choose to relax certain numerical bounds in order to achieve a better solution [124]. Overall, the interface allows to add, delete and modify the layout problem formulation. Modification of the layout problem can be used to guide search into an area of interest.

Furthermore, with the ability to modify problem formulation, the designer can also intervene the optimization process. For instance, he can intervene the optimization process



(a) DxDy mode (b) Specific mode

Figure 3.17 – Defined constraint modes.

by using the optimization interface to edit the optimization parameters (see Fig. 3.18), or force search into a new area of the space by manipulating the components.

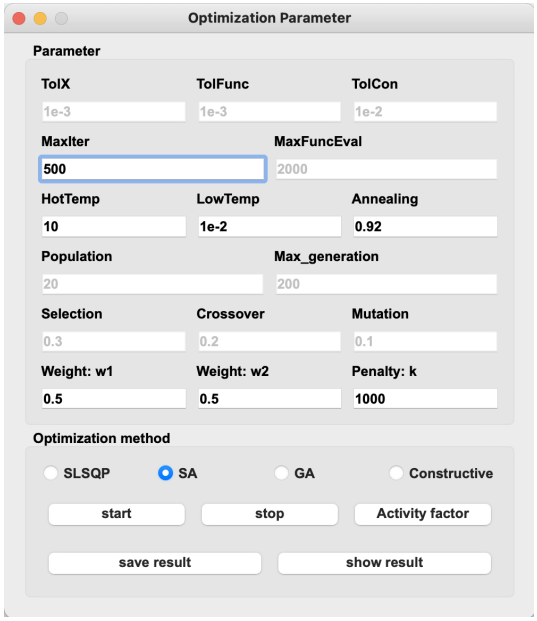


Figure 3.18 – Optimization parameter.

Thus, the presented design tool can be seen as a sketchpad for interactive optimization. As a typical procedure, the designer would

1. Define the layout optimization problem: Define the container and components, the optimization objectives, and the special or default constraints.
2. Select the optimizer: The optimization algorithm will be applied to find one or a set of layout solution(s).
3. Examine results: Check results visually and check performance values.
4. Iterate the optimization: Use information to refine the problem definition or guide search into a new layout.

3.4.2 Similarity indicator for decision-making

In traditional multi-objective optimization, layout problems may have more than one objectives to be optimized. Moreover, the results can be evaluated by the convergence and diversity, that is, the comparison of the desired and obtained solutions. When the optimization process continuously converges, the similarity among the solutions increases,

and the diversity of the solutions decreases. Therefore, converging to Pareto-front while keeping good diversity is essential. However, the layout problem is special:

— Subjective requirements are difficult to model mathematically, therefore, they are typically ignored in optimization models. for example the components should be accessible from the entry, which is not easy to be integrated into the problem formulation.

— 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.

Consequently, 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. When the solutions, have been generated by the algorithm, the designer needs to compare these solutions. Therefore, it is necessary to evaluate similarities among feasible solutions before showing the representative solutions. Maintaining the diversity of solutions is important to guarantee interaction after optimization. By displaying the obtained solutions, the designer can use his expertise to interact and explore the design space by manipulating locally the configuration of some components, and find the design that satisfies all the requirements. In other words, the designer plays a major role in the selection of the ideal result of the application. For the optimized solutions, there may be similarities within the design set. Therefore, we define a similarity indicator to evaluate the similarity of the design, which helps the designer distinguish between layout design. A similarity indicator represents how closely the current layout design resembles the others. Two kinds of similarity definitions are described below.

Grid difference

Usually, similarity computation 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 [125]. Nevertheless, evaluating the similarity globally is more generic than individual component comparison. Thus, we divide the layout configuration into grids and define a similarity indicator for each paired designs based on the element-wise difference.

To calculate the similarity indicator, first, permute all the layout designs that belongs to Pareto optimal set. For each paired layout, calculate an element-wise difference using

the grid representation, for example, if the same grid is occupied by the same component, then the grid is labeled as 1, otherwise it is marked as 0. Then, calculate the percentage of the same elements among all elements. The value of indicator is in the range of 0 to 1. The larger the indicator is, the more closely the layout designs are.

However, the grid difference can not differentiate the symmetrical configuration, and it becomes time consuming as the number of grids increases. So we propose another lightweight similarity indicator based on the relative position.

Relative position

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. 3.22. 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.

$$M = \begin{pmatrix} 00 & 01 & \cdots & 10 \\ 10 & 00 & \cdots & 01 \\ \vdots & \vdots & \vdots & \vdots \\ 11 & \cdots & \cdots & 00 \end{pmatrix} \tag{3.22}$$

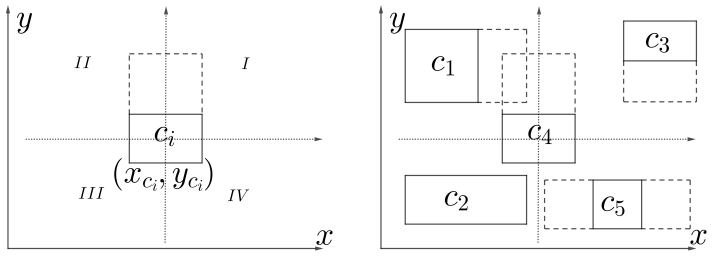
In a pairwise comparison of components (c_i, c_j), there are four possible relative positions I, II, III, IV of component c_j with respect to the reference component c_i, as shown in Fig. 3.19(a), it is determined according to the location of centroid, expressed as:

$$\alpha_{x_{ij}} = x_{c_i} - x_{c_j} \tag{3.23}$$

$$\alpha_{y_{ij}} = y_{c_i} - y_{c_j} \tag{3.24}$$

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. 3.19(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. 3.19(b), $M_{14} = 10$



(a) Definition (b) Representation

Figure 3.19 – Relative position.

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. 3.19(b), $M_{24} = 11$
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. 3.19(b), $M_{54} = 01$

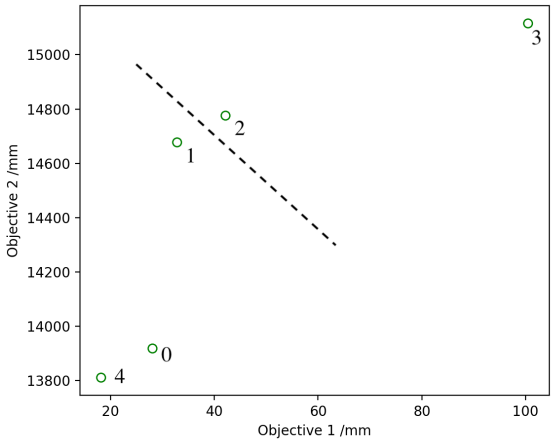
To evaluate the similarity of the relative position schemes, calculate an element-wise difference for each pair of the matrices M. The similarity value is expressed as the percentage of all elements that are the same. The similarity is between 0 to 1. The larger the value, the higher the similarity. 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.

Similarity analysis aims to keep a good diversity of the layout designs and reduce the designer selection workload. A similarity indicator defines how similar two layout designs are. The designer could select the most favorable solution based on the visualization tool.

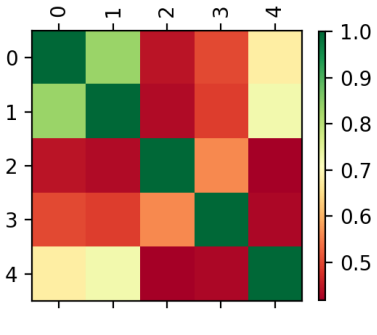
3.4.3 Solution visualization tools

When the algorithm generated a set of feasible solutions, the designer needs to compare the solutions. Therefore, it is recommended to display the layout design information both in objectives and model design spaces to help the designer select the ideal solution. The visualization tools presented in the followings allow for multidimensional visualization of the solution.

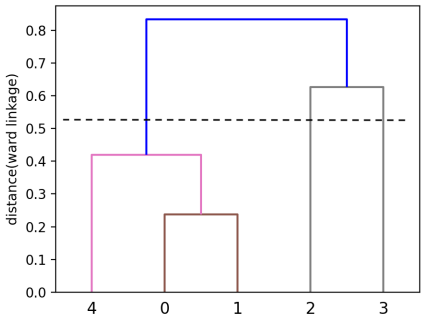
Scatter plot A scatter plot is mostly used for rendering solutions in the lower dimensional space. A scatter plot consists of points in an orthogonal frame that represent the objective values of the layout configurations. In particular, the value of the optimization



(a)



(b)



(c)

Figure 3.20 – Visualization tools.

objective is associated with each point. The point then forms the Pareto front of the optimization problem, and solutions can be compared according to their values. Fig. 3.20 (a) presents one scatter plot of solutions.

Matrix plot A matrix plot displays a two-dimensional matrix with rows of the matrix represented in rows, columns represented in columns, and the first row is at the top (corresponding to the original matrix representation). The similarity indicators for pairwise layout design formulate a similarity symmetric matrix, where the color in the grid depends on the similarity value, as illustrated in Fig. 3.20 (b).

Dendrogram A dendrogram is a diagram that shows the hierarchical relationship between objects. It is most commonly created as an output from hierarchical clustering. For the similarity matrix, we apply the hierarchy cluster algorithm [126] to build nested clusters by merging similar solutions successively. At each iteration, a distance matrix of clusters is maintained. When only one cluster remains, the algorithm stops, and this cluster becomes the root. The main use of a dendrogram is to work out the best way to allocate objects to clusters. The dendrogram in Fig. 3.20 (c) presents the hierarchical clustering of five solutions shown on the scatter plot. Besides, solutions are allocated to clusters by drawing a horizontal line (threshold) through the dendrogram. Solutions that are below the line are in clusters. In the example, we have two clusters. One cluster combines 0, 1 and 4, and a second cluster combining 2 and 3.

3.5 Conclusion

This chapter presents some innovative concepts allowing to model a layout optimization problem in a generic way. In order to translate the requirements of the designer into optimization variables, design constraints and objectives, first of all, the layout is formulated by the new component of solid and virtual parts. This definition makes it possible to create a model of the layout problem, which takes into account all the specificities of the problem. Besides, the geometrical constraints such as non-overlap and non-protrusion; the functional constraints including the orientation and accessibility are translated into explicit mathematical functions. Furthermore, the layout optimization problem, presented in this work, is a multi-objective formulation.

Then, a new capacity index is proposed to evaluate the feasible complexity of the layout problem. This index defines an alternative to the traditional calculation of the compactness of a layout problem. Contrary to the commonly used density calculation of compactness, this new index is based on the evaluation of the minimum occupied space of the components which reflects the geometrical information of the layout problem. The capacity evaluation is implemented by the simulated annealing and constructive packing optimization. The space generation is notably developed for the efficient geometrical constraint evaluation. This index makes it possible to demonstrate, in a reliable way, if the problem can be solved or not.

The interactive environment allows integrating mathematical optimization with human decision-making during conceptual design of the layout problem. The designer in-

teracts with the problem optimization representation by adding, deleting and modifying objectives, constraints and components. Also, the final decision-making uses the visualization tools to compare the solutions by analyzing the objectives and similarities that characterize quantitative information. In order to help the designer distinguish layout designs, a similarity indicator is notably proposed, which quantitatively expresses how similar the current layout design is to other layout designs. In summary, the main functions of the interactive environment are the visualization, modification, exploration of the geometric model of the solution.

Chapter 4

Multi-objective optimization of layout problem

4.1 Introduction

This chapter is devoted to the layout optimization approach. The optimization aims to find the optimal solutions of the layout problem formulated in Chapter 3. The layout problem, taking into account the virtual components and the accessibility to components. Two objectives, namely layout balance and activities, are considered. 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.

Therefore, an optimization algorithm based on constructive approach is proposed to solve the layout problem. The algorithm is based on the combinatorial optimization: archive-free SA algorithm, to determine the placement and configuration sequences; constructive placement algorithm, to place components sequentially. In this chapter, the constructive placement is first introduced to generate complete layout configurations. Then, the use of SA optimization is justified by the complexity analysis of the combinatorial layout problems. Finally, the optimization and the relevant model in multi-container case are described.

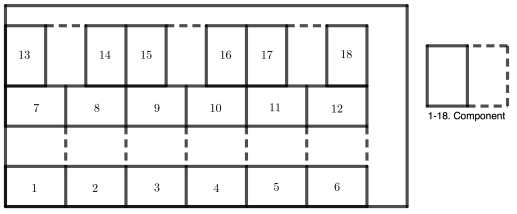
4.2 Solving simple layout examples

In this section, two different layout examples are formulated in comparison developed archive-free SA with NSGA II. The examples properties are summarized in Table 4.1, including the component number, the density and capacity, the size of the components, and the functional constraints. Both layout examples have the associated virtual components.

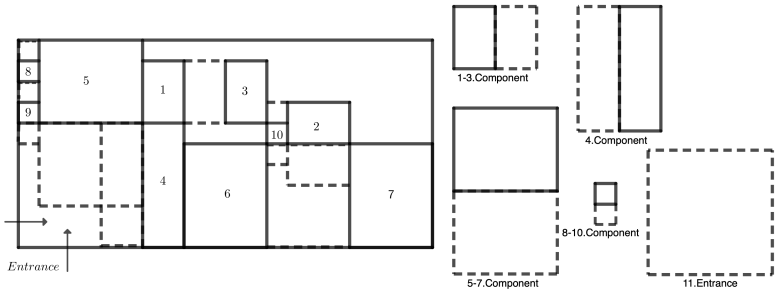
The capacity values are deduced using the method in Chapter 3, and the corresponding most compact layout configurations are shown in Fig. 4.1.

Table 4.1 – Properties of layout examples.

	Test 1	Test 2
Number of components	18	11
Density of solid components	0.54	0.47
Density of virtual components	0.54	0.66
Density of solid and virtual components	1.08	1.13
Capacity	0.81	0.75
Equal size	Yes	No
Accessibility	No	Yes



(a) Test 1



(b) Test 2

Figure 4.1 – Test examples.

Test 1 – Equal-sized component in Fig. 4.1(a)

The problem is concerned with placing 18 equal-sized components into a container, width is 4000 mm, and height is 2000 mm, respecting geometrical non-overlap and non-protrusion constraints. The virtual component, symbolized by the dotted rectangle, has the same size as the solid component. And the dimensions are given in Table 4.2.

Table 4.2 – Data in Test 1.

Item	Dim/w (mm)	Dim/h (mm)	Mass/m (kg)
Container	4000	2000	-
1-18.Components	600	400	50

Test 2 – Unequal-sized component in Fig. 4.1(b)

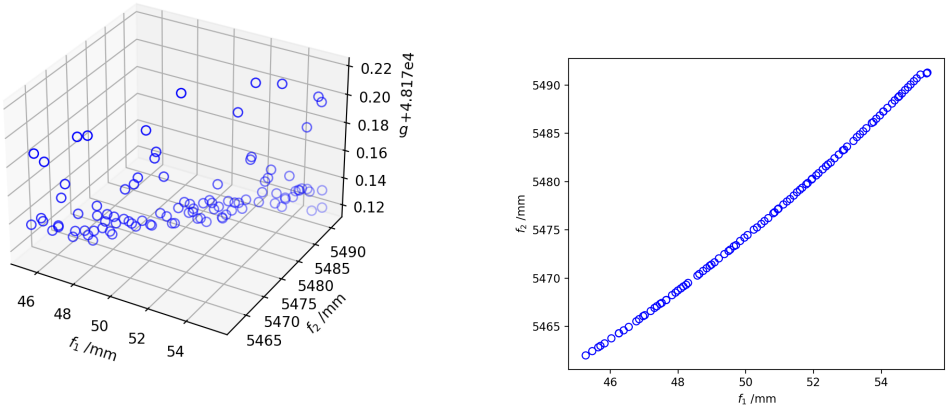
The unequal-sized component is more common and realistic. The problem involves accessibility requirements and geometrical constraints. It has two types of components: components 5-7 with 2-equal size virtual components, and the rest components each with 1-equal size virtual component. The entrance, fixed to the lower left corner of the container, is modelled as the virtual component. The container size is the same as in Test 1. The detailed dimensions of components are given in Table 4.3.

Table 4.3 – Data in Test 2.

Item	Dim/w (mm)	Dim/h (mm)	Mass/m (kg)
Container	4000	2000	-
1-2. Components	600	400	50
3. Component	600	400	200
4. Component	1200	400	100
5-7. Components	200	200	30
8-10. Components	1000	800	200
11. Entrance	1200	1200	-

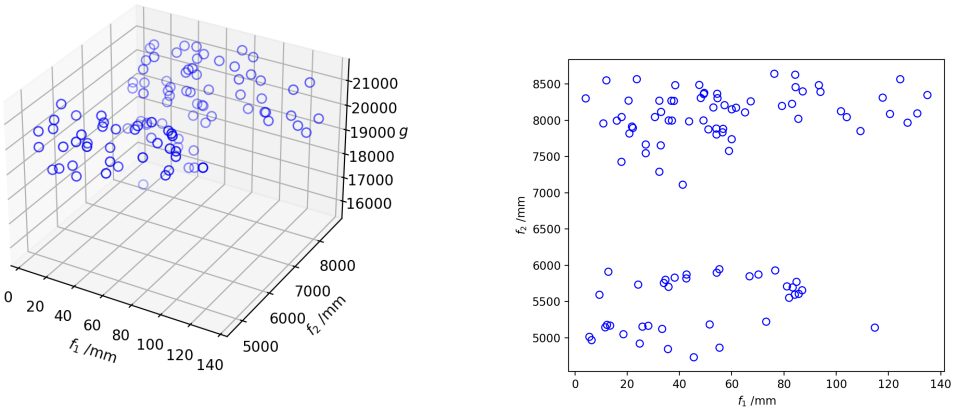
Both layout examples are formulated as the multi-objective optimization problems presented in Eq.3.12: optimize the balance and the activity under constraints. Here, we assume that the activity among components 1, 2, 3, 9 must be restricted so such that the distance must be maximized, thus their activity factors equal to -1 for minimization. We apply archive-free SA and NSGA II to solve the continuous problem formulations, population size $N = 100$ and the maximum iterations is taken as 1000. However, archive-free SA and NSGA II cannot really search for feasible regions in a reasonable time due to the limited space in the test examples with a capacity up to 0.81. Therefore, the algorithm may generate many infeasible solutions in which the non-overlap constraint cannot be satisfied.

In NSGA II algorithm, even if different operators are used to generate new individuals, the non-overlap satisfied individuals are still very sparse. In the end, the optimization is likely to converge into a small niche of the solution space. This phenomenon can



(a) (b)

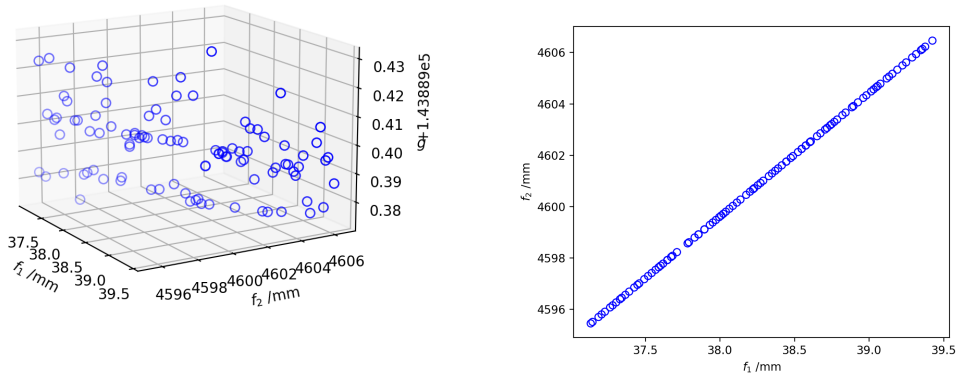
Figure 4.2 – NSGA II obtained solutions of Test 1.



(a) (b)

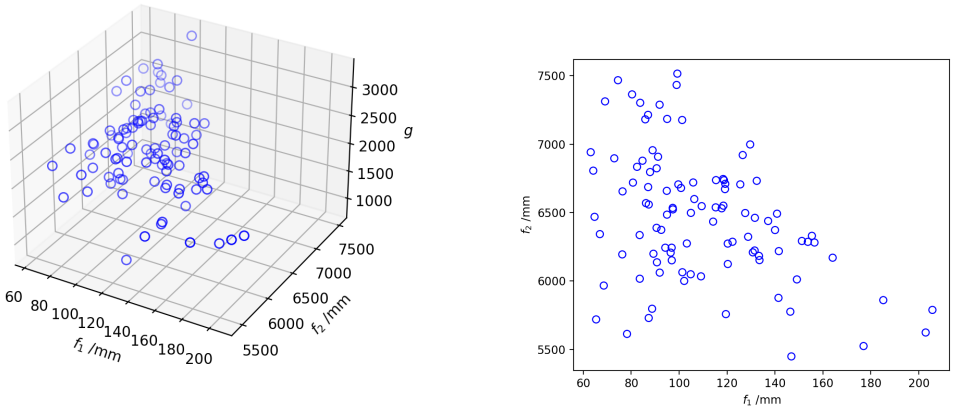
Figure 4.3 – Archive-free SA obtained solutions of Test 1.

be seen in Fig. 4.2 and Fig. 4.4, and the obtained solution is generated by the local movement of components in one type of layout design. In archive-free SA algorithm, it tries to explore the space by accepting the worse solutions, such that the process will not converge prematurely. However, the difficulty in finding the feasible solutions remains the same level. The obtained solutions are shown in Fig. 4.3 and Fig. 4.5. Moreover, the layout example is more complex than the common practical problems. It is evident from the results that finding feasible configurations using conventional optimization algorithm



(a) (b)

Figure 4.4 – NSGA II obtained solutions of Test 2.



(a) (b)

Figure 4.5 – Archive-free SA obtained solutions of Test 2.

is quite challenging.

Based on the above analysis, it proves that no method is superior than the other method. Therefore, it is better to guide the search into the feasible space. In the following sections, one alternative approach will be designed for efficient feasible solution generation in the layout problems.

4.3 Constructive placement for layout generation

Applying continuous optimization is challenging due to the constrained design space. Observe that the most of solutions not respect the non-overlap constraints and consequently are infeasible. Constructive placement algorithm is inspired from [123]. The former algorithm was developed for the cutting problem and the virtual components are not considered. To place components with respect to constraints, first of all, we integrate the placement strategy of solid and virtual components to satisfy the geometrical constraints; then we characterize component accessibility as a constraint during the construction process.

4.3.1 Placement strategy

As presented in Chapter 3, the placement of component will generate two kinds of available space, denoted as:

- $\boldsymbol{a}$: available spaces used for virtual components;
- $\boldsymbol{a}'$: available spaces used for solid components.

The placement strategy is proposed to place component in appropriate space with respect to the constraints. The strategy concerns two aspects, namely component placement and space selection.

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. To place a component $c_i = (s_i, v_{ij})$, $j = 1$, according to the selection of the available space, i.e., $a_i \in \boldsymbol{a}'$ and $a_j \in \boldsymbol{a}$, there will be two possibilities: a_i and a_j coincide or not.

1. If the selected space a_i and a_j coincide, then the component will be placed in the corners of selected space with four rotations. It ensures that 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. 4.6, and we have the configuration sequence $p_i = (1, 2, ..., 16)$.
2. Otherwise, the solid component will be placed in the corners of a_i, and the configurations in Fig. 4.6(b), (c) becomes the placements as shown in Fig. 4.7, where certain configurations will be adjusted according to the selected space a_j. One

example is given in Fig. 4.8, instead of placing c_2 in the corner in Fig. 4.8(a), the position is refined to avoid overlapping with c_1 in Fig. 4.8(b).

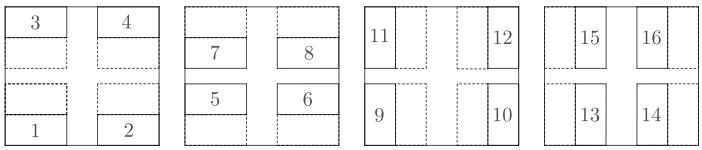


Figure 4.6 – Placement examples a_i and a_j are coincide.

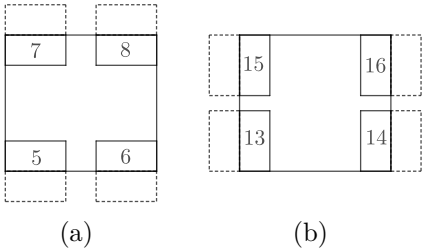


Figure 4.7 – Placement examples a_i and a_j are not coincide.

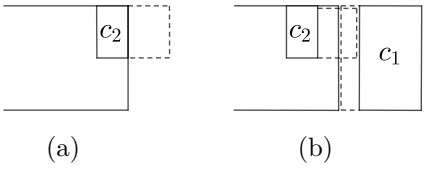


Figure 4.8 – Placement adjustment.

If the component $c_i = (s_i, v_{ij})$, $j > 1$, and v_{ij} is distributed in more than one direction ($x-$axis, $y-$axis), then the placement will be proceeded iteratively, presented in Algorithm 6. Assuming there is one component $c_1 = (s_1, v_{11}, v_{12}, v_{13}, v_{14})$ (see Fig. 4.9) selected to be placed with $p_1 = 16$.

1. If the selected space a_i and a_j coincide, then the component will be placed in the corners of selected space.
2. Otherwise, the placement of solid and virtual components is similar to the one directional case but is performed iteratively. One example is given in Fig. 4.10. Firstly, place s_1 to the bottom right corner in a_i; then place the virtual components

Algorithm 6 Component placement

for a_i *in* $\boldsymbol{a}'$ **do**
  Place s_i in a_i
  for a_j *in* $\boldsymbol{a}$ **do**
    Place v_{ij} in 1^{st} direction in a_j
    Adjust placement of s_i in a_i
    for a_k *in* $\boldsymbol{a}$ **do**
      Place v_{ij} in 2^{nd} direction in a_k
      Adjust placement of s_i in a_i and v_{ij} in 1^{st} direction in a_j
    end
  end
end

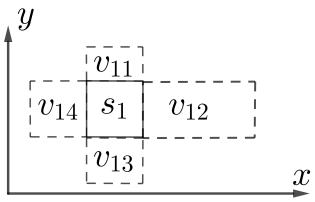


Figure 4.9 – Two directional virtual components representation.

v_{12} and v_{14} along $x-$ axis in a_j and adjust placement of s_1 at the same time; finally, place virtual components v_{11} and v_{13} along $y-$ axis in a_k and adjust placement of s_1, v_{12} and v_{14} to satisfy non-overlap constraints.

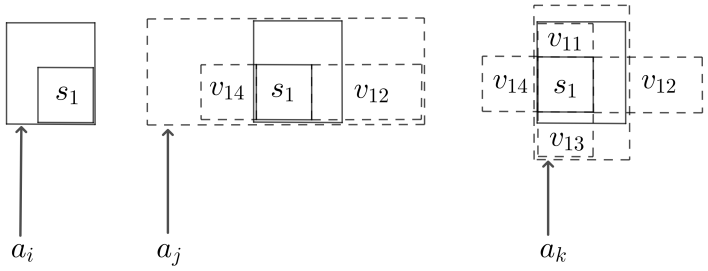


Figure 4.10 – Placement of two dimensional virtual components.

Space selection

To place a component, it should decide which available space will be used. The component placement is decided by the spaces in $\boldsymbol{a}$ that used for virtual components and

$\boldsymbol{a}'$ that used for solid components. The successive placement process can be treated as a combination problem. Thus, an effective space selection rule is essential for a constructive placement. Three selection strategies are proposed:

1. Selection strategy 1: Check all the combinations of spaces $(a_i, a_{j(k)})$, $a_i \in \boldsymbol{a}'$ and $a_{j(k)} \in \boldsymbol{a}$.
2. Selection strategy 2: Select one combination of spaces $(a_i, a_{j(k)})$, $a_i \in \boldsymbol{a}'$, $a_{j(k)} \in \boldsymbol{a}$ satisfying Eq. 3.16, a_i is the smallest sized space. The selection aims to successfully finish the construction process with less computational effort, namely space-filling placement.
3. Selection strategy 3: Select one combination of spaces $(a_i, a_{j(k)})$, $a_i \in \boldsymbol{a}'$, $a_{j(k)} \in \boldsymbol{a}$ satisfying Eq. 3.16, a_i is the largest sized space.

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 can be found in the next section that describes the summarized constructive placement algorithm.

4.3.2 Accessibility analysis

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 sufficient for the component accessibility. The proposed layout model is composed by a set of rectangular components. Therefore, the proposed method uses rectangle with size (w_r, h_r) to represent the accessible space required by the user, shown in Fig. 4.11, where rectangles represent the path taken by the user inside the layout, in order to reach to the component from the entrance.

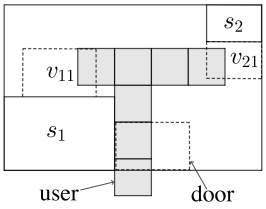


Figure 4.11 – Accessibility representation.

In our work, we characterizes component accessibility as a constraint during the construction process. Assume that k components have been placed in the container space, and the generated available spaces are kept in $\boldsymbol{a}$ and $\boldsymbol{a}'$. To place the $k+1$ component, the accessibility analysis is applied as summarized in Algorithm 7. The accessibility analysis

Algorithm 7 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 v_{ij}.

builds the connection tree using spaces in $\boldsymbol{a}$ that is 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:

$$max(0, min(x_{R_{a_i}}, x_{R_{a_j}}) - max(x_{L_{a_i}}, x_{L_{a_j}})) \geq w_r \tag{4.1}$$

$$max(0, min(y_{T_{a_i}}, y_{T_{a_j}}) - max(y_{B_{a_i}}, y_{B_{a_j}})) \geq h_r \tag{4.2}$$

where the rectangle size (w_r, h_r) represents the accessible space required by the user. Assuming one component 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. Using the recursion idea, Algorithm 8 illustrates traversing the connection tree to find a path. 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 presented in Fig. 4.12, the placement of v_{11} occupies a_1, and there is one path=[a_d, a_2, a_1].

To understand how the construction works, consider a simple two-dimensional layout problem. This problem consists in placing five components (five types of components) and one flexible door in a container, while respecting the geometrical constraints and ac-

Algorithm 8 find path(*start node* = a_d, *stop node* = a_v, *path* = [ ])

```
path = [path, startnode]
if start node=a_v then
  | return path
end
if start node not in connection tree then
  | return infeasible configuration
end
for node in connection tree do
  | if node not in path then
  |   | path = find path(start node, stop node, path)
  |   | if path exist then
  |   |   | return path
  |   | end
  | end
end
```

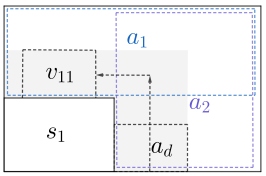
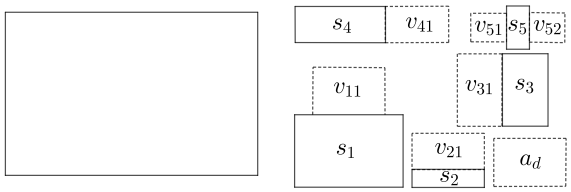


Figure 4.12 – Connection path [a_d, a_2, a_1].

cessibility constraints. Fig. 4.13 illustrates the components and container of the example. For the layout problem, a rectangle (w_r, h_r) represents the user working in the layout.



(a) Container (b) Components

Figure 4.13 – Layout problem of five components.

And the size can be set as the minimum space of the virtual components. When placing a new component into the container, one connection tree is built based on the intersection relationship of available spaces in $\boldsymbol{a}$. The connection is evaluated at each level of the tree. Once the tree generation is finished, check if there is one path for accessibility

(accessible from the door through available space; ignore the indirect connection between components).

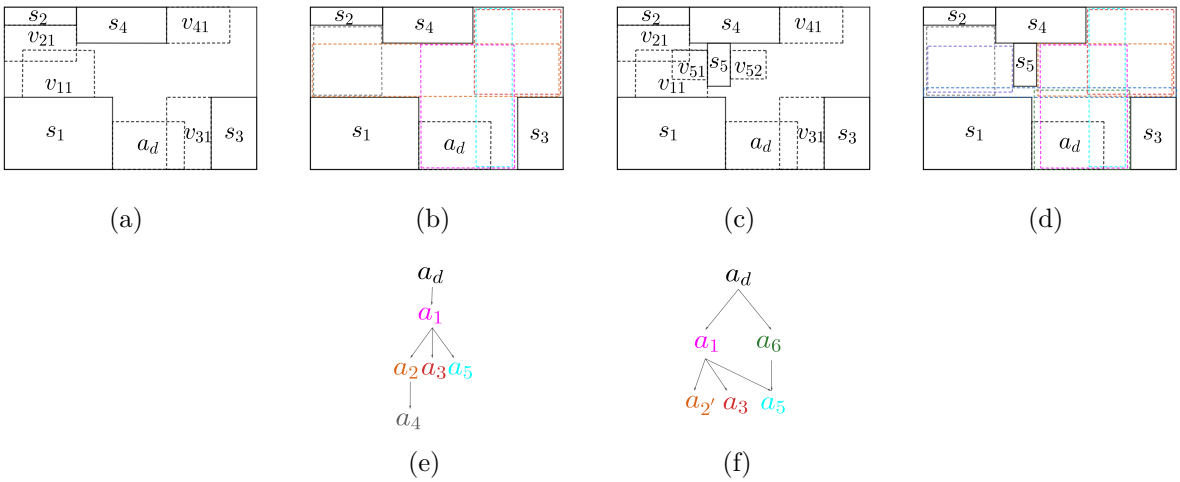


Figure 4.14 – Accessibility analysis (a) Placement of components $c_i = (s_i, v_{ij}), i \in (1, 2, 3, 4)$, (b) Space generation $\boldsymbol{a}$ of (s_1, s_2, s_3, s_4), (c) Placement of components $c_i = (s_i, v_{ij}), i \in (1, 2, 3, 4, 5)$, (d) Space generation $\boldsymbol{a}$ of $(s_1, s_2, s_3, s_4, s_5)$, (e) Connection tree $\boldsymbol{a}$ generated by (s_1, s_2, s_3, s_4), (f) Connection tree $\boldsymbol{a}$ generated by $(s_1, s_2, s_3, s_4, s_5)$.

Examples in Fig. 4.14 illustrate the accessibility analysis. The placements of (c_1, c_2, c_3, c_4) in Fig. 4.14(a) generate available spaces $\boldsymbol{a} = \{a_1, a_2, a_3, a_4, a_5\}$ in Fig. 4.14(b) in colors. The connection tree in Fig. 4.14(e) is generated based on the Eq. 4.1 and Eq. 4.2, 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=$[a_d,\ a_1,\ a_2]$
- 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. 4.14(c), then the space generation updates as in Fig. 4.14(d) 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. 4.14(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.

4.3.3 Constructive placement algorithm

The constructive placement process is summarized in Algorithm 9. In step 1, we initialize the available space as the container size. Then place the component sequentially to have the geometrically feasible configurations p_{ig} in step 2. The configuration selection is determined by the boundary classification in step 3, accessibility verification in step 4 and overlap maximization in step 5. The process continues with new space generation and next component placement in step 6.

Algorithm 9 Constructive placement

1: Initialize the available space as a_0 in $\boldsymbol{a}$ and $\boldsymbol{a}'$.
2: Place component c_i successively following the placement sequence $\boldsymbol{c}$. With the selected available space $(a', a), a' \in \boldsymbol{a}', a \in \boldsymbol{a}$ according to the placement strategy, go through the configuration sequence p_i to find all the feasible configurations that satisfy the geometric constraints, denoted as $p_{ig} = (i_1, i_2, ..., i_r), r \leq 16$.
3: 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 $p_{ib} = (\underbrace{j_1, ..., j_l}_{p_{ib_1}}, \underbrace{j_{l+1}, ..., j_r}_{p_{ib_2}})$.
4: Check the accessibility requirement of the obtained configurations and filter out the unsatisfied candidates. The final feasible configurations list is $p_{ia} = (\underbrace{k_1, ..., k_h}_{p_{ia_1}}, \underbrace{k_{h+1}, ..., k_q}_{p_{ia_2}}), q \leq r \leq 16$.
5: If there are more than one feasible configuration 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}.
6: Update $\boldsymbol{a}$ and $\boldsymbol{a}'$ with space generation. Continue placing components to complete the construction. Otherwise, the layout configuration is infeasible.

4.3.4 Constructive placement strategy comparison

Here, we use two layout examples that presented before to test the different placement strategies in the constructive placement. Constructive placement is designed to circumvent the difficulty arising from constraints. In each iteration, the placement order is generated randomly. With a given number of iterations, the more feasible solutions it finds, the better the performance.

Table 4.4 – Placement strategy comparison with fixed configuration sequence.

Number of solutions		Test 1	Test 2
Fix configuration sequence	Space selection strategy 1	1/100	2/100
	Space selection strategy 2	1/100	37/100
	Space selection strategy 3	1/100	5/100

Table 4.5 – Placement strategy comparison with permuted configuration sequence.

Number of solutions		Test 1	Test 2
Permute configuration sequence	Space selection strategy 1	22/100	5/100
	Space selection strategy 2	26/100	40/100
	Space selection strategy 3	13/100	6/100

Placement strategy with fixed configuration sequence

The strategy comparison results are summarized in Table 4.4 where the number of solutions is obtained with 100 iterations. Three strategies are compared with fixed configuration sequence. If we do not permute the configuration sequence, then the configuration sequence is fixed as $p_i = \{1, 2, ..., 16\}$. In Test 1, if there is no configuration permutation, the constructive placement will find one feasible solution with poor diversity no matter which strategy is used. 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 random placement sequence but not the configuration sequence. 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, consequently, more solutions are obtained compared to strategy 1 and 3.

Placement strategy with permuted configuration sequence

The strategy comparison results are summarized in Table 4.5 where the number of solutions is obtained with 100 iterations. Three strategies are compared with random configuration permutations.

In order to overcome the limitation of diversity, 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 1 compared to the case of fixed configuration sequence. Besides, it turns out that the performance of strategy 1 will decrease if there is accessibility requirement, for example, 22 out of 100 are feasible in Test 1 but 5 out of 100 feasible

in Test 2. However, strategy 2 significantly outperforms the strategy 1 and 3 in feasible solution generation.

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. In other words, under the same condition, different configurations have the same possibility to be selected during the construction process.

— Considering the search ability under constraints, strategy 2 is much better than strategy 3. Strategy 2 conducts the placement effectively and achieves better results compared to the other.

Based on the above analysis, space selection 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. These contributions were appeared in a peer-reviewed publication [127].

4.4 Optimization for layout problem

The constructive placement treats the layout generation as a discrete combinatorial problem. The complexity is first analyzed before presenting the proposed optimization algorithm.

4.4.1 Complexity analysis

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.

4.4.2 Layout optimization algorithm

In Chapter 2, the experimental results prove that archive-free SA is computationally cheaper and produce better approximations on MOKPs. Therefore, archive-free SA is used to improve the placement and configuration sequences in Algorithm.10. Given a state X_i, a layout with $\boldsymbol{f}_i$ is generated using the constructive placement. And the neighbors are generated by a swap operator: two components could be selected randomly based on step parameter σ. The mechanism is the same in the configuration sequence.

Algorithm 10 Optimization framework

```
/* Block of constructive placement */
for X_i in P do
    Given current state X_i = (c, p)
    for c_k in c do
        Select available space by placement strategy
        Place c_k with p_k ∈ p s.t. accessibility analysis
        Select configuration by placement strategy
        Space generation
    end
    Objective evaluations f_i
end
/* Block of optimization */
while stop condition not met do
    while iteration in inner loop do
        P' = neighbor generation (P)
        Constructive placement
        (R, R') = fast nondominated sort
        (CD, CD') = crwoding distance
        P = dynamic selection
    end
    decrease temperature t
end
```

As we noticed that, the layout problem is a constrained problem. The constraints involve the functional and geometric requirements. In the presence of constraints, each solution generated by constructive algorithm can be either feasible or infeasible. Thus, we introduce the infeasible violation I to measure the number of unplaced components during the construction process, $I \in [0, n]$. There are at most two situations:

1. If the placement reaches to the end of the construction process, it means all the components are feasible so the infeasible violation $I = 0$.
2. Otherwise, the construction will be interrupted earlier and the infeasible violation I equals to the number of unplaced components among $\boldsymbol{c}$.

Consequently, the *fast nondominated sort* is carried out using the following domination comparison:

— both solutions are feasible. The denomination is measured according to the original objective functions.

— Neither are feasible. The objective value becomes the infeasible violation. The infeasible solution with smaller infeasible violation denominates the other.

— Only one of the two is a feasible solution. The infeasible solution is dominated by the feasible solution.

The ranks of all feasible solutions are compared using the objective function values, while the ranks of infeasible solutions are compared using infeasible violations. Feasible solutions are assigned better ranks than infeasible solutions using the modified domination comparison.

4.4.3 Comparisons of optimization results

Since the placement is sequential, the placement sequence $\boldsymbol{c} = \{c_1, ..., c_n\}$ is one discrete variable in SA. Furthermore, in the placement strategy comparison, it turns out that the configuration sequence $\boldsymbol{p} = \{p_1, ..., p_n\}$ will also affect the solution candidates. Thus, we propose three formulations concerning the design variables and parameters in the layout optimization:

1. Design variable $X_i = (\boldsymbol{c})$, and parameter $p_i = \{1, ..., 16\}$;
2. Design variable $X_i = (\boldsymbol{c})$, and parameter p_i is permuted randomly;
3. Design variable $X_i = (\boldsymbol{c}, \boldsymbol{p})$.

In the first two cases, configuration sequence p_i is treated as one parameter (fixed or permuted), while in the last case, p_i is taken as one discrete variable. Here, the optimization is performed under 100 iterations with population size $N = 20$. The objective of the layout example is to find the optimal design which minimize its balance and maximize the required functional distance. For each case, the optimization has been run twenty times because of the stochastic behavior of the algorithm. A comparison among the results

obtained using the different formulations was performed to verify their performance to optimize layout, executing 2000 evaluations of objective functions for each of them.

In Fig. 4.15, it is shown the Pareto solutions obtained using optimization in Test 1. It is possible to observe that the fixed configuration sequence has not presented a good

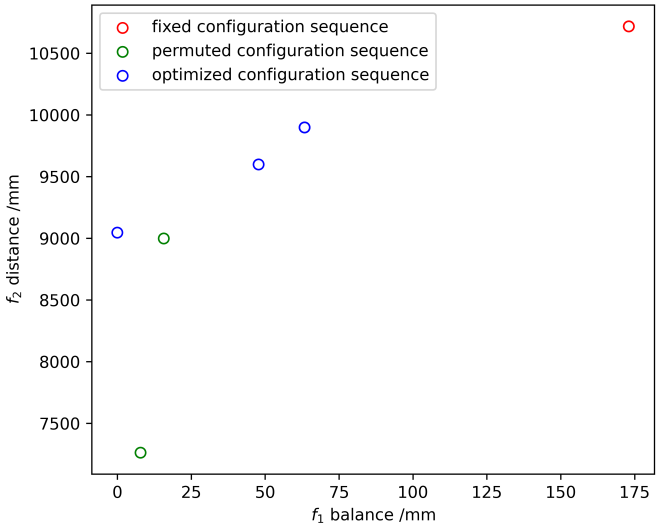


Figure 4.15 – Pareto solutions obtained using different formulations in Test 1.

performance. It has found only one extreme Pareto solution. Besides, the permuted configuration sequence has found two solutions but are dominated by solutions found by optimized configuration sequence. Optimizing configuration sequence found more Pareto solutions than others. In other words, optimized configuration sequence solutions dominated the most of the other ones.

In Test 2, the Pareto front obtained by different optimization formulations are shown in Fig. 4.16. It can be noted that although fixed configuration sequence presents six solutions, only one of them dominate the solutions found by the other algorithms in the mass balance around 100 mm. While two of eight is considered as the non-dominated solutions using permuted configuration sequence. In the balance region below 200 mm, optimized configuration sequence presented the best performance, finding more trade-off solutions than others algorithms where one solution slightly dominated by solutions found by permuted configuration sequence.

Through different multi-objective layout optimizations, several configurations of layout optimization designs were obtained, which found the Pareto solutions. From the results it

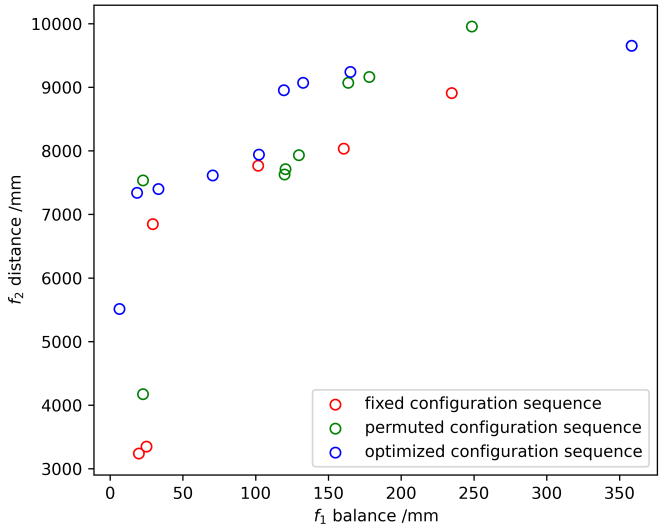


Figure 4.16 – Pareto solutions obtained using different formulations in Test 2.

can be said that optimizing configuration sequence had the best performance on finding the Pareto front.

4.5 Multi-container layout problem

In this study, the multi-container layout problem aims to find the layout configuration under flexible sub-container size. The objectives are to optimize the mass distribution and the activity relationships among components while satisfying the functional and geometric constraints. In the formula presented here, components and sub-container are represented by rectangles following current industrial practices. Overall, the multi-container layout problem model is stated as follows:

Given:

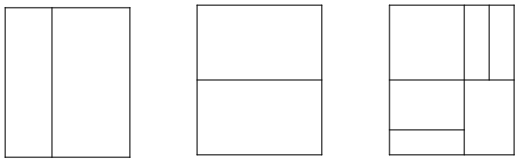
— n components with fixed dimensions and l sub-containers with flexible sizes;

— Fixed assignment of components to sub-containers $r : \{c_1, ..., c_n\} \rightarrow \{1, ..., l\}$;

— Total container dimensions W, H and the area $z = W \times H$

Determine:

— The sub-container size and the component placement, so as to find the Pareto front of the multi-container layout problem.



(a) "|" form (b) "-" form (c) "+" form

Figure 4.17 – Partition form

4.5.1 Boundary restrictions

For multi-container layout problem with fixed component assignment, the component placement is restricted by size of the corresponding sub-container. Depending on the sub-container size, the layout will be divided into different spaces. And, a complete layout configuration can be generated according to the component placement. The various partition forms are presented in Fig.4.17. In general, for a given layout area, the combination of partition form can divide the layout into different rectangular sub-container For example, if there are $l-1$ "|" form partitions inside the layout, then the layout area z will be divided into l sub-container with area $z_j = W_j \times H_j, j \in l$. Once the location of partition is determined, components will be arranged in each sub-container. Given the assignment of components r, $r_i = j$ if component $c_i, i \in n$, is assigned to sub-container $j \in l$. If partition size is ignored, the following conditions should be satisfied:

$$0 \leq z_j \leq z \tag{4.3}$$

$$\sum_{j=1}^{l} z_j = z \tag{4.4}$$

To make sure components can be placed inside each sub-container, the occupied area of components should be less than the sub-container area. Assume that for each sub-container j, component c_i placed inside is $r_i = j$. Thus lower bounds of sub-container area z_j should satisfy non-protrusion constraints of components:

$$\hat{\beta}_{c_j} \leq z_j \tag{4.5}$$

where $\hat{\beta}_{c_j}$ is the minimum occupied space of components inside sub-container j. Solving

Eq.4.4 subject to constraint Eq.4.5, the new boundary constraints of sub-container become

$$\hat{\beta}_{c_j} \leq z_j \leq \alpha_j + \hat{\beta}_{c_j} \tag{4.6}$$

$$\sum_{j=1}^{l} \alpha_j = z - \sum_{j=1}^{l} \hat{\beta}_{c_j} \tag{4.7}$$

By doing so, we transform the constrained global optimization of z_j in Eq.4.3 into local search region in Eq.4.6.

4.5.2 Extension to multi-container layout optimization

In general, the optimization framework for multi-container case is the same as the proposed algorithm for single-container case: archive-free SA optimizes multi-objective considering the sub-container $\boldsymbol{z} = (z_1, ..., z_l)$, which are continuous variables, and the placement-related sequences including placement sequences $\boldsymbol{c} = (c_1, ..., c_n)$ and configuration sequences $\boldsymbol{p} = (p_1, ..., p_n)$, which are discrete variables. Once the sequences and sub-container spaces are determined, the placement is used to construct the complete layout.

In each iteration, a new population will be generated by *neighbor generation*. The pseudo-code is described in Algorithm 11. Each individual, $X_i = \{\boldsymbol{z}, \boldsymbol{c}, \boldsymbol{p}\}$ is perturbed to generate a new individual $X_i' = \{\boldsymbol{z}', \boldsymbol{c}', \boldsymbol{p}'\}$.

Algorithm 11 *neighbor generation*

Input: P, τ, σ
Output:P'

1: **for** $i \leftarrow 1$ to N **do**
2: $\quad \boldsymbol{c}', \boldsymbol{p}' = swap(\boldsymbol{c}, \boldsymbol{p}, \sigma)$
3: $\quad lb = max(\boldsymbol{z} - \tau * (Ub - Lb), Lb)$
4: $\quad ub = min(\boldsymbol{z} + \tau * (Ub - Lb), Ub)$
5: $\quad \boldsymbol{z}' = lb + rand * (ub - lb)$
6: **end for**
7: return $P' = (X_1', ..., X_N')$

To generate a new state, the placement-related sequences in the same sub- container are perturbed by a swap operator. Moreover, changing of the sub-container area is defined by the neighborhood search. The area of sub-container is locally modified in its neighbor region in line 5. The neighborhood size is defined by the lower and upper boundary

Lb, Ub, where $Lb = (\hat{\beta}_{c_1}, ..., \hat{\beta}_{c_l})$, $Ub = (\alpha_1 + \hat{\beta}_{c_1}, ..., \alpha_l + \hat{\beta}_{c_l})$. The new boundary lb, ub defines the minimum and maximum movement of the current value and $\tau \in (0, 1)$.

4.6 Conclusion

The more constraints, the more the design space is divided into separate zones. Also, the larger the capacity index, the more difficult to find feasible configurations. Consequently, the layout problem is then more complex. The effective non-overlap evaluations must be implemented to reduce the computational time. The use of a stochastic algorithm is necessary considering the great complexity of layout problems, especially industrial applications. However, the experimental results on layout examples have demonstrated the feasible difficulty of the continuous optimizer.

This chapter has detailed the optimization for layout problems. It is based on the constructive placement and one stochastic algorithm, that is archive-free simulated annealing. The constructive placement, used for layout generation, selects and places components successively while respecting all constraints. Indeed, it solves the most difficult issue, feasibility, in the layout generation. Furthermore, the stochastic optimization is introduced to decrease the computational complexity in the search space. Moreover, the placement is independent of the optimization and can be extended to any other layout problem, i.e. multi-container case. The algorithm is implemented and tested in several application cases introduced in next chapter, and their results are compared quantitatively and qualitatively.

Chapter 5

Industrial applications

5.1 Introduction

Layout problems are inherently multidisciplinary tasks and are usually solved as optimization problems, expressed as finding the optimal arrangements of components inside the container to optimize the objectives and respect constraints. The research on industrial layout optimization intensified in recent years. Industrial layouts have complex environments and various design goals and constraints to ensure the layout is in a good functional state.

The layout problem presented in this chapter relates to the industrial application proposed by the French company Thales SIX France in Cholet. The application concerns the layout of shelter design. A proper shelter layout can effectively improve the system performance. Light and mobile shelter with on-board equipments, such as cabinets, desks and other electrical boxes, provides complete protection for personnel and against battlefield aggression. This technical shelter layout is often attached to the rear of a vehicle and is dedicated to communications missions during military operations. Its versatility means a variety of armed forces can use the shelter. The first two examples deal with the problem of the shelter layout in a single container. The last case study is a multi-container shelter layout. For each case study presented in this chapter, all the steps of the method are described: the description, the capacity analysis, the resolution of the problem, as well as the similarity analysis for later interaction. 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.

5.2 Single-container shelter problem

The chosen shelter layout is taken from [128]. This case study is a shelter with four cabinets, two desks and two electrical boxes as illustrated in Fig. 5.1.

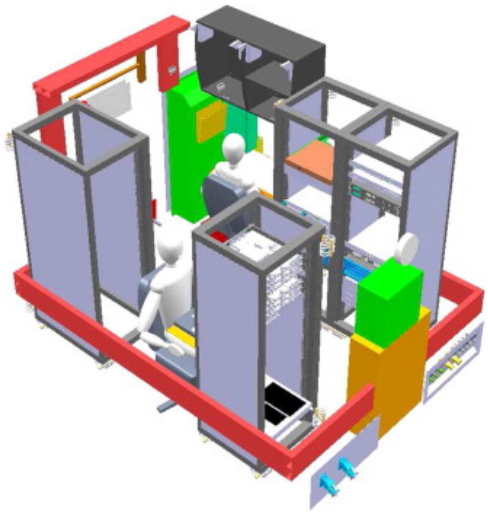


Figure 5.1 – Overview of CAD model of single-container shelter [128].

5.2.1 Problem description

In order to simplify the optimization formulation, 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-dimension, shown in Fig. 5.2.

We use the rectangle to represent each component. Considering the accessibility, a virtual component (light color), the same size as the solid component, 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. These components should be accessible from the fixed entrance.

The shelter has dimension $W = 2150$ mm,$H = 2740$ mm. The dimensions of the equipment, as well as the mass of each equipment, are indicated in Table 5.1.

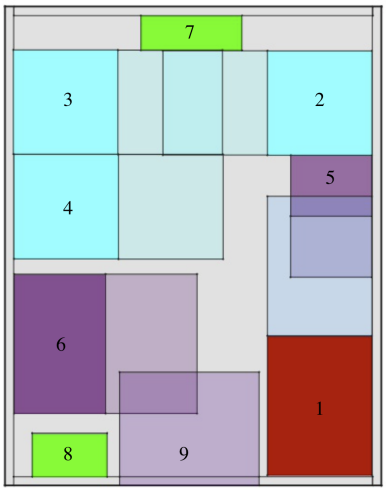


Figure 5.2 – 2D configuration model of single-container shelter.

Table 5.1 – Data of components in single-container shelter.

Item	Dim/w (mm)	Dim/h (mm)	Mass/m (kg)
1.cabinet	800	600	400
2.cabinet	600	600	300
3.cabinet	600	600	300
4.cabinet	600	600	274
5.desk	350	465	10
6.desk	525	800	30
7.box	200	580	45
8.electrical box	250	430	35
9.entrance	800	600	-

5.2.2 Problem formulation

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.

The geometrical constraints of the shelter are non-overlap and non-protrusion constraints. Besides, there are also functional remarks of the shelter:

1. The cabinets are restricted in the allowed spaces, where non-overlap between cabinets and the hatched rectangle space below the air conditioner in Fig. 5.3;
2. The electrical box 8 has to be placed against the front wall near the door or the wall behind, where no rotation is allowed;

3. The desk 5 is a tablet that can be folded down and it is therefore assumed that it can overlap all the virtual components present in the shelter.

These requirements are formulated as constraints that presented in Chapter 3, and will be handled by the constructive placement. The distance between cabinet 1 (energy box), cabinet 3, cabinet 4 and electrical box 8 should be maximized to limit interactions between electrical and energy networks. For simplicity, we list the activity relationship of these four components in Table 5.2. Two optimization objectives are therefore:

Table 5.2 – Activity factor of the single-container shelter.

Item	1.cabinet	3.cabinet	4.cabinet	8.energy box
1.cabinet	0	0	0	-1
3.cabinet	0	0	0	-1
4.cabinet	0	0	0	-1
8.energy box	-1	-1	-1	0

— minimize objective 1: distance between the center of gravity of all the components and the geometric center of the shelter, given by Eq.3.8 in Chapter 3.
— maximize objective 2: distance between cabinet 1 and the set made up of cabinets 3 and 4 and box 8, given by Eq.3.10 in Chapter 3.

The multi-objective layout optimization aims to find the arrangement (location and orientation) $\boldsymbol{p} = \{p_1, p_2, ..., p_n\}, n = 8$ of components $\boldsymbol{c} = \{c_1, c_2, ..., c_n\}, n = 8$, optimize objective 1 and objective 2 and satisfy geometrical (non-overlap and non-protrusion) and functional constraints (accessibility, edge on the wall), given by Eq.3.12 in Chapter 3.

5.2.3 Capacity evaluation of the layout

The shelter layout has eight solid components, seven virtual components (six accessibility spaces of the solid components that have the same dimensions to these related solid components, one virtual entrance). The density of solid and virtual components using the formula Eq.3.13, Eq.3.14 and Eq.3.15 in Chapter 3, are listed in Table 5.3. In previous study [128], the author introduces one free space below the air-conditioner (hatched rectangle, one free space at the shelter's entry, one virtual corridor located in the middle that connected to the door of the entry, which guarantees all solid components should be accessible, as shown in Fig. 5.3. However, if the virtual components (6 spaces of accessibility, 1 corridor, 1 door, 1 free space) are included and overlap between them are ignored, then the sum of the density is increased to $\beta_{sv} = 1.05 > 0.85$. Since we

Table 5.3 – Density of components.

Density	Value
Density of solid components	β_s =0.40
Density of virtual components	β_v =0.45
Sum of density	β_{sv} =0.85

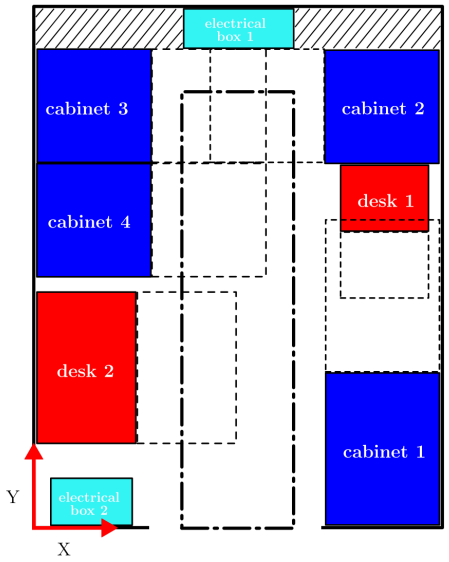


Figure 5.3 – Previous 2D configuration model [128].

employ the accessibility analysis, the corridor is of no use in our case. Besides, to place the cabinets in the allowed spaces, the specific constraint modes could be used in layout modelling as presented in Chapter 3, instead of using one virtual free space.

By applying the method presented in Chapter 3 with 1000 iterations, it takes 12 minutes to find the minimum occupied space of the case study where β_c =0.55, which shows a priori feasibility of the shelter optimization. The corresponding layout design is shown in Fig.5.4. In [122], the author use intersection matrix to calculate the capacity $\beta_c = 0.66$. However, the value is based on estimation and no geometry included.

Besides, when we place the components sequentially, it may generate some small spaces, which are empty, but no components can be placed. Indeed, these small spaces should be identified as infeasible spaces. After obtaining the prior feasible information, we apply optimization algorithm to find the set of feasible solutions of the shelter.

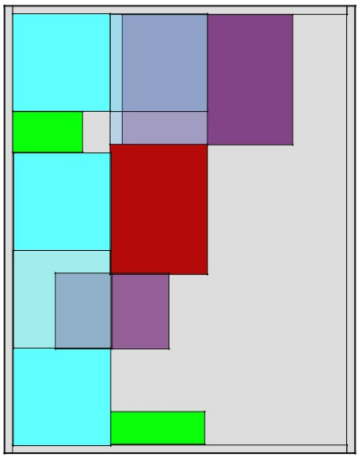


Figure 5.4 – The compact configuration with $\beta_c = 0.55$.

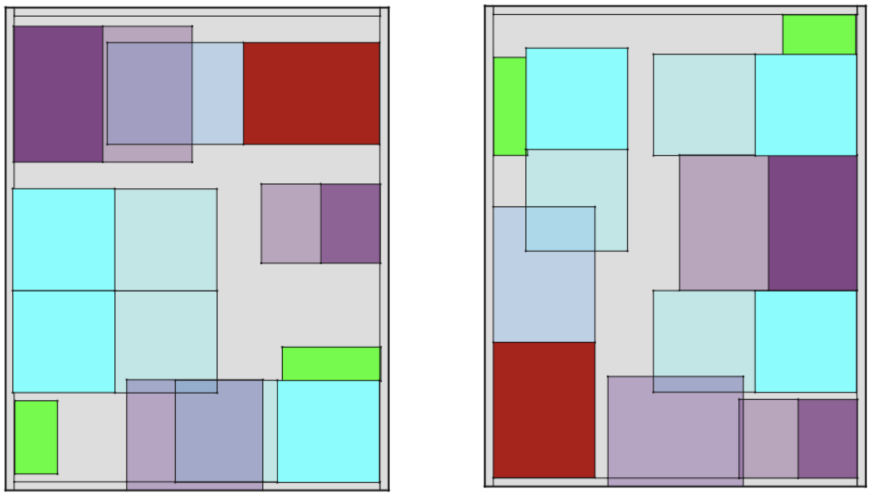
5.2.4 Optimization results and similarity analysis

An initial population has $N = 200$ random individuals. The initial temperature t_s is initialized as $t_s = 100$. Set the cooling rate $r = 0.9$ and the total number of iterations $L = 250$ to perform the annealing process. The algorithm searches for solutions by considering the geometrical and functional constraints and objectives of the problem formulation. Fig. 5.5 shows the previous optimal solution and an optimal solution obtained from the optimization. All layout designs satisfy the non-overlap, non-protrusion, accessibility and the additional user defined constraints.

Table 5.4 – Numerical results of solutions.

Objective	Initial solution	Previous optimal solution	Optimal solution
objective 1/mm(minimization)	254.1	83.3	15.8
objective 2/mm(maximization)	6048.8	6484.3	6971.7

For comparison, Table 5.4 presented the objective values of the initial solution, the previous optimal solution in the literature, the optimal solution that realizes the best compromise between optimization objectives. The numerical result illustrates that the optimal solution can realize much better objective values compared to the initial one and the previous one. Indeed, the initial configuration created by the engineers of Thales was generated from geometrical aspects. And the configuration of the initial expert solution is described in Fig.5.2. The previous optimal solution based on interactive modular optimization is presented in [125]. The experimental results prove that the proposed al-



(a) Previous optimal solution [125]. (b) Optimal solution.

Figure 5.5 – Optimal solutions configurations.

gorithm is effective in solving the layout problem under functional constraints. Besides, it can be seen that there is a significant difference between the optimal layout solution compared to the initial and the previous solution. It proves that the proposed layout can conduct better exploration and exploitation ability. At the same time, it can find better solutions in both objectives.

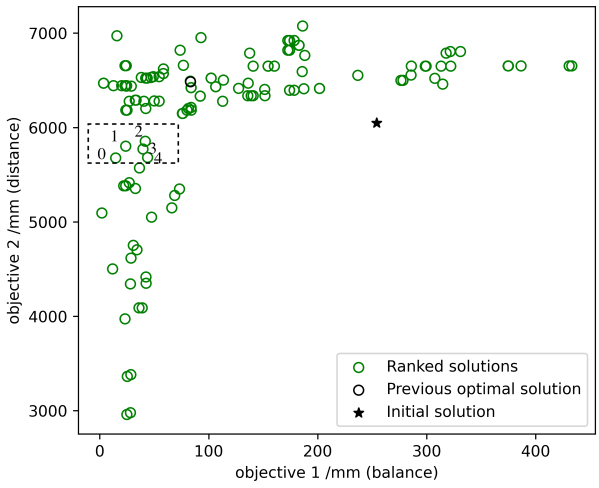
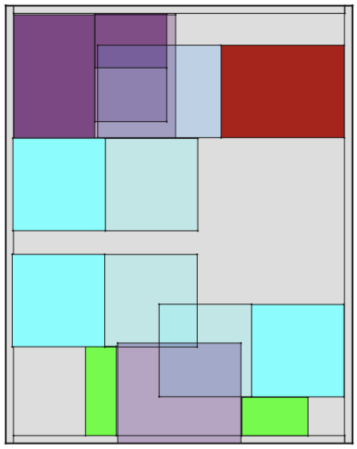
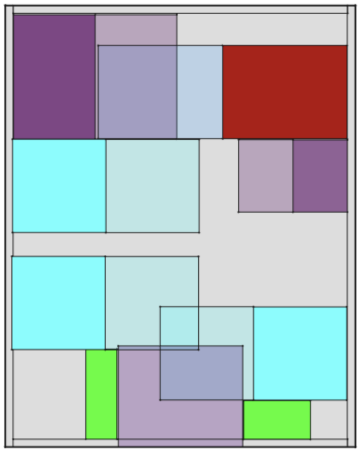


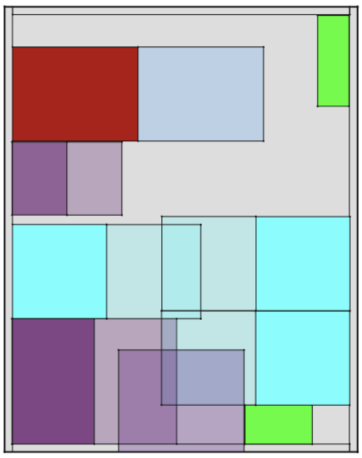
Figure 5.6 – Display of solutions in objective space.



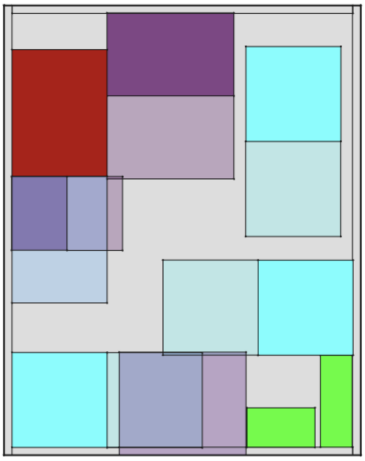
(a) solution 0.



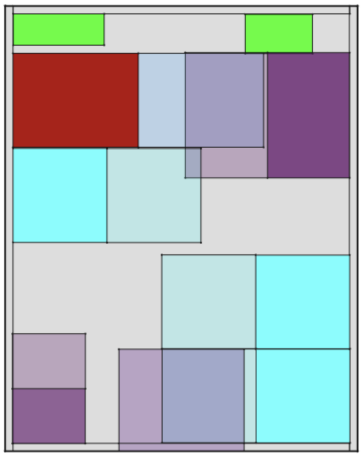
(b) solution 1.



(c) solution 2.

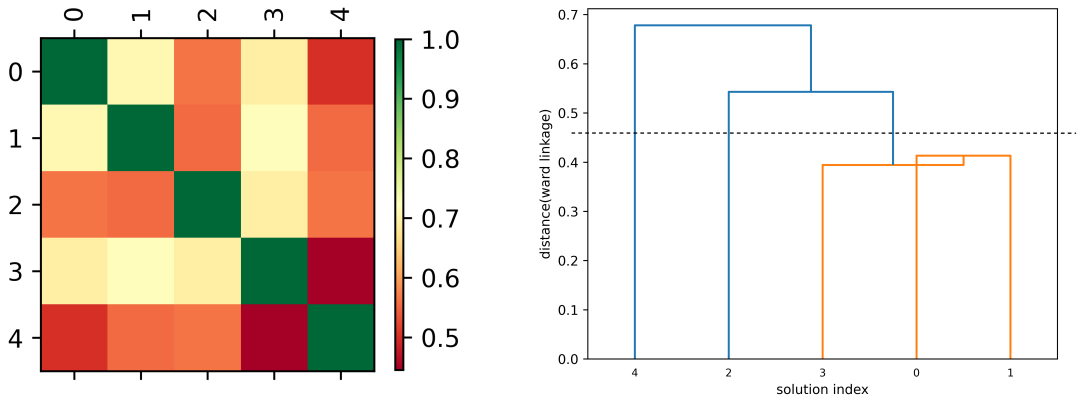


(d) solution 3.



(e) solution 4.

Figure 5.7 – Display of selected designs.



(a) Similarity matrix.

(b) Hierarchical cluster dendrogram.

Figure 5.8 – Similarity analysis.

The proposed optimization algorithm 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 $[1, 10]$ and the number of solutions is 135, as shown in Fig. 5.6. We can see that the exploration region distributed in objective space is no longer a niche. Besides, the proposed algorithm can generate more choices for designers. Then we randomly select five solutions, numbered from 0 to 4, as shown in Fig. 5.6. The corresponding configurations are given in Fig. 5.7 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. 5.8(a). By comparison, design 0 and design 1 have higher similarity value, while design 0 and design 4 have lower similarity value compared to others. 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. 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. 5.8(b). It can detect the geometrical differences between configurations and identify similar groups. For example, the similar design 0 and design 1 are assigned into one group, whereas designs 3 with less similarity values are grouped with design 0 and

design 1 into another cluster. Besides, by drawing a horizontal line (threshold) through the dendrogram, all the connected descendent links below a cluster node are in the same cluster if this node is below the cut threshold, that is, solution 0, solution 1 and solution 3 are in the same cluster. The visualization tool can provide information on the hierarchical similarity of designs, helping users to quickly select the preferred solution.

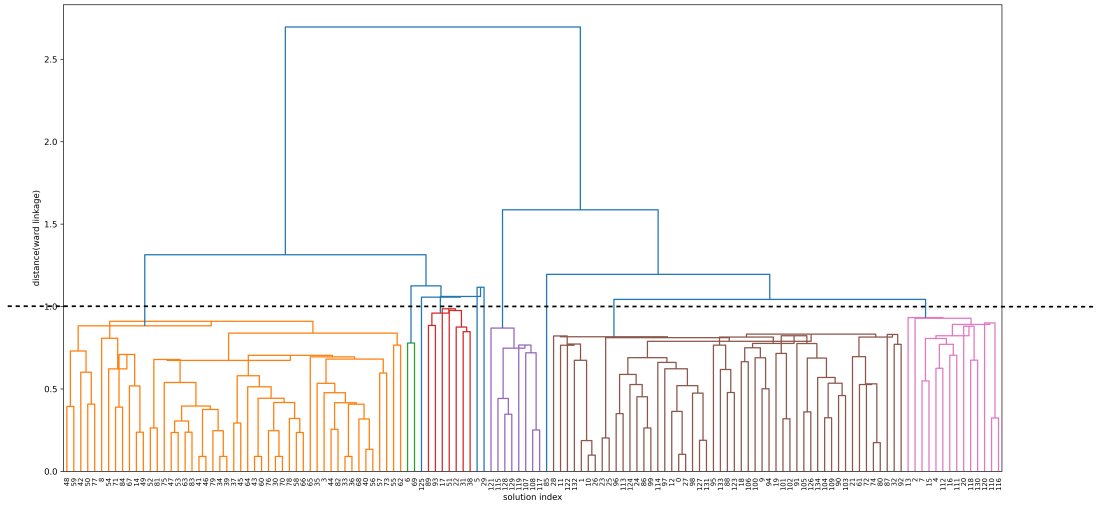


Figure 5.9 – Display of cluster dendrogram.

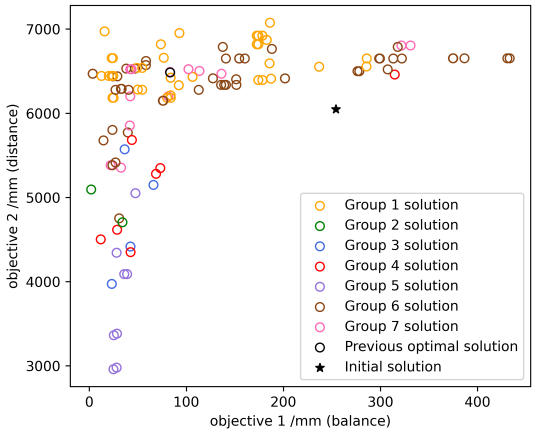


Figure 5.10 – Display of clustered solutions in objective space.

After preliminary experiments, we now apply the similarity analysis to ranked solutions, and clustering results are shown in Fig. 5.9. If we set the threshold to define the clusters, we can have the different grouped solutions, as shown in Fig. 5.10, the same grouped solutions have the same color. It is proved that, close points in objective space can have different 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.

The experimentation proves that the proposed optimization is effective in generating alternatives and finding high-qualified solutions in a reasonable computational time. Moreover, the similarity analysis demonstrates good diversity of the obtained layout set, which can be applied as an interactive tool.

5.3 Single-container shelter with big size components

The shelter with big size components as shown in Fig. 5.11, introduces the size difference issue into layout instances.

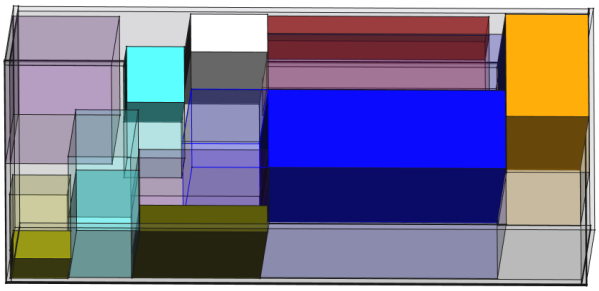


Figure 5.11 – Big-sized shelter representation.

5.3.1 Problem representation and formulation

In order to simplify the optimization formulation, there are two assumptions:

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

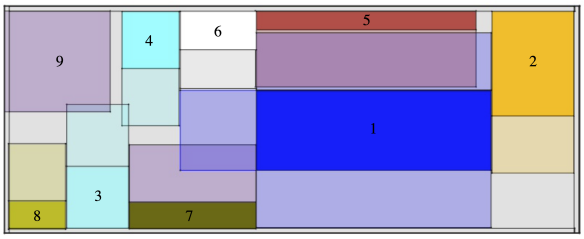


Figure 5.12 – Big-sized shelter representation.

The shelter is simplified as shown in Fig. 5.12. The container is rectangular with a width of 5945 mm and a height of 2286 mm. The big size component, namely amplifier component, has three virtual 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.

Each component is represented by a set of rectangles. The entrance, fixed to the upper left corner of the container, is modelled 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 and the mass are given in Table 5.5.

Table 5.5 – Data in shelter with big size components.

Item	Dim/w (mm)	Dim/h (mm)
1.amplifier	2469	841
2.ventilation	860	1100
3.energy box	650	650
4.operation box	600	600
5.battery	2320	200
6.air conditioner	800	406
7.extinguisher	1330	283
8.circuit board	600	300
9.entrance	1060	1060

Except for geometrical constraints (non-overlap and non-protrusion), additional constraints include edge on the wall, alignment, and accessibility of components.

1. The air conditioner and extinguisher must place against one wall of the container
2. The alignment specifies that ventilation must attach to the right side of amplifier.

These requirements are formulated as constraints that presented in Chapter 3, and will be handled by the constructive placement. The minimum functional distance between

amplifier and entrance has to be considered, thus the activity factor w_{19} taken as -1. Two optimization objectives are:

— minimize objective 1: distance between the center of gravity of all the components and the geometric center of the shelter, given by Eq.3.8 in Chapter 3.

— maximize objective 2: distance between amplifier and entrance, given by Eq.3.10 in Chapter 3.

The multi-objective layout optimization aims to find the arrangement (location and orientation) $\boldsymbol{p} = \{p_1, p_2, ..., p_n\}, n = 8$ of components $\boldsymbol{c} = \{c_1, c_2, ..., c_n\}, n = 8$, optimize objective 1 and objective 2, and satisfy geometrical (non-overlap and non-protrusion) and functional constraints (alignment, accessibility, edge on the wall), given by Eq.3.12 in Chapter 3.

5.3.2 Capacity evaluation of the layout

The shelter has eight solid components, and the density of solid components equals to 0.38. There are also eleven virtual components and the density of virtual components is up to 0.65. The density of solid and virtual components using the formula Eq.3.13, Eq.3.14 and Eq.3.15 in Chapter 3, are listed in Table 5.6. And the sum of the density

Table 5.6 – Density of components inside shelter.

Density	Value
Density of solid components	β_s =0.38
Density of virtual components	β_v =0.65
Sum of density	β_{sv} =1.03

β_{sv} =1.03 which is bigger than 1. However, by applying the calculation method presented in Chapter 3, we obtain a capacity index $\beta_c = 0.77$, which shows a priori feasibility of the optimization.

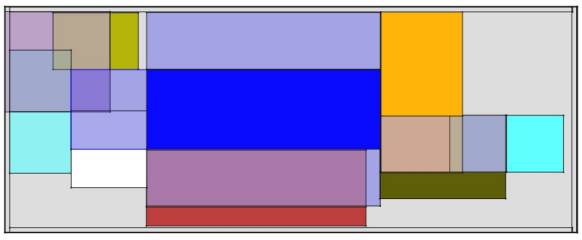


Figure 5.13 – Compact configuration of big-sized shelter.

5.3.3 Optimization results and similarity analysis

An initial population has $N = 200$ random individuals. The initial temperature t_s is initialized as $t_s = 100$. Set the cooling rate $r = 0.9$ and the total number of iterations $L = 250$ to perform the annealing process. The algorithm searches for solutions by considering the geometrical and functional constraints and objectives of the problem formulation, nine layout designs are Pareto-optimal solutions, shown in Fig. 5.14.

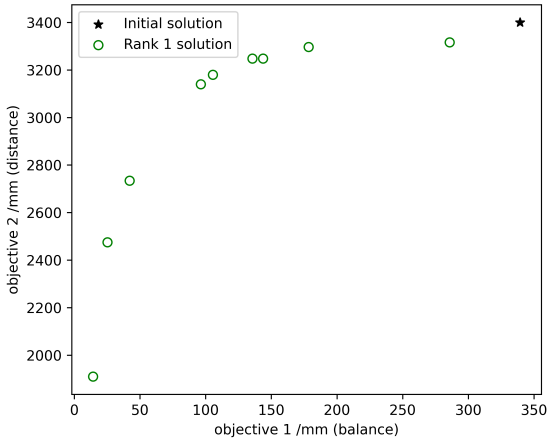
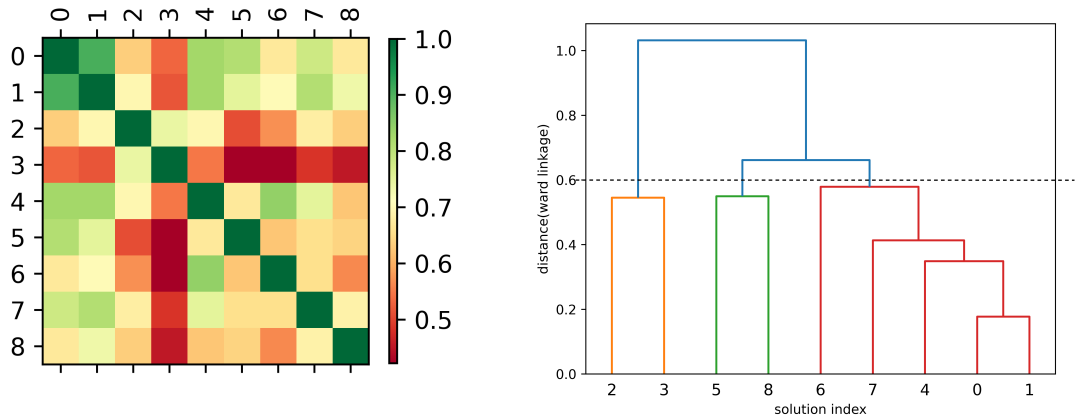


Figure 5.14 – Display of rank 1 solution.



(a) Similarity matrix.

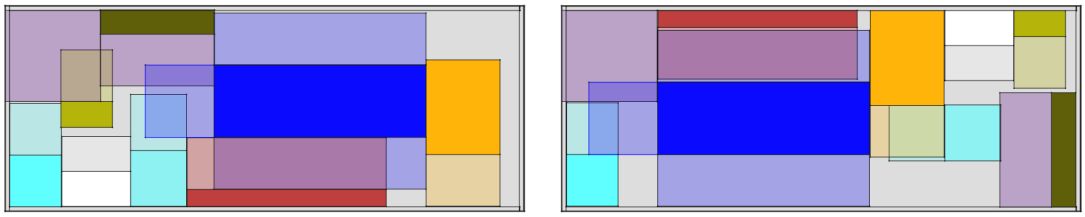
(b) Hierarchical cluster dendrogram.

Figure 5.15 – Similarity analysis of big size component shelter.

The initial configuration proposed by the expert in Fig. 5.12, has maximum functional

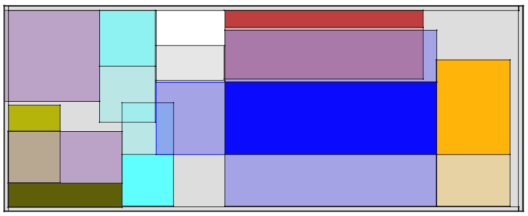
distance around 3400 mm and maximum gravity deviation about 350 mm. Compared with the initial solution, the obtained solutions of our optimization are non-dominated solutions in the objective; however, we cannot formulate all the requirements from the expert, so different non-dominated solutions are still good, and the expert finally chooses one of them as their final decision, not their initial proposal.

To analyze the performance of the design space, first of all, the similarity indicator is computed for the obtained layout configurations, and the similarity matrix is presented in Fig. 5.15 (a). It can be seen that solution 0 and solution 1 have the highest similarity value, whereas solution 3 and solution 5 have the lowest similarity. The hierarchical



(a) solution 3.

(b) solution 8.



(c) solution 1.

Figure 5.16 – Display of optimal designs.

similarity, given in Fig 5.15 (b), is consistent with the similarity information. Here, we assume there are three clusters, each representative is shown in Fig. 5.16, and the same grouped solutions have the same color, as shown in Fig. 5.17. Each solution is a new variant compared to the other. The designer could select the solution that achieves best

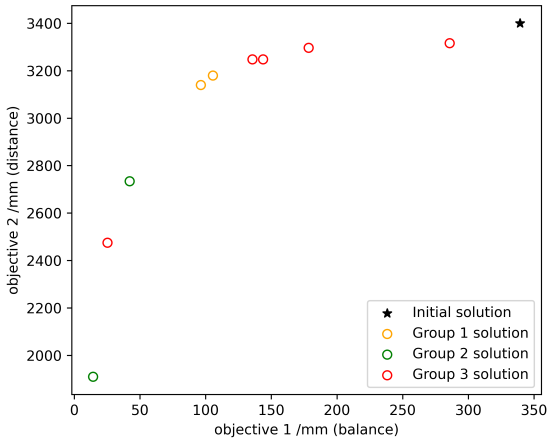


Figure 5.17 – Display of clustered solutions in objective space.

compromised objective value based on the visualization tool.

The layout application has a fairly large component with multiple virtual spaces. When the capacity of layout is large, the main objectives is to find the set of feasible alternatives. The higher the capacity, the harder it is to find a feasible solution. The optimization results confirm that the proposed optimization can effectively generate feasible solutions such that placement constraints are satisfied and tend to reduce computational efforts.

5.4 Multi-container shelter problem

This case study is an extension of the shelter layout problem with three zones [129]. This section details the optimization process implemented for this multi-container layout problem.

5.4.1 Representation of the shelter

The shelter studied here, as shown in Fig. 5.18, is a three-dimensional shelter with three different spaces, named storage zone, technical zone and operator zone. In order to simplify the optimization problem, there are two assumptions:

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

Figure 5.18 – Multi-container shelter layout representation

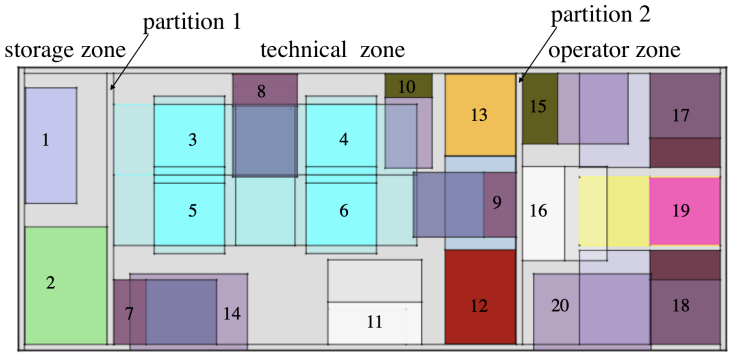


Figure 5.19 – 2D configuration model of multi-container shelter.

Consequently, the configuration of the problem is defined in two-dimensional, as shown in Fig. 5.19. And the evaluation of constraints is more easily realized. Moreover, each component is represented by a rectangle. The data of the container is given in Table 5.7. Dimensions described in Table 5.8, Table 5.9 and Table 5.10 match the configuration in Fig. 5.19.

Table 5.7 – 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

As suggested in Chapter 3, components can be separated into two categories: solid and virtual components. A set of virtual components attached to components (light color) represent accessibility spaces. 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

Table 5.8 – 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 5.9 – 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.entrance	1000	5	120

Table 5.10 – 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.entrance	1000	5	120

space.

- Component 1, component 2 and component 13 are solid components, indeed, these boxes do not need to be accessible,
- Component 3, component 4 and component 5 and component 6, each has one 600 mm virtual space and one 344 mm virtual space,
- Component 11, component 12 and component 16, each has an identical size virtual space,
- The rest components, each has a 600 mm virtual space.

The accessibility spaces of the cabinets are modelled to guarantee the free space in front

of the cabinets allowing the loading of material into them. Office accessibility spaces correspond to the spaces occupied by the operator in front of his workstation, Indeed, the virtual space either guarantees interaction or correct usability.

5.4.2 Problem formulation

The overall formulation of the problem including variables, constraints and objectives. The geometrical constraints of the shelter are non-overlap and non-protrusion constraints. 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.

These requirements are formulated as constraints that presented in Chapter 3, and will be handled by the constructive placement. 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 5.11. So the two objectives are defined as:

Table 5.11 – Activity factor of the multi-container shelter.

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

minimizing layout balance (objective 1) and maximizing circulation distance (objective 2). Two optimization objectives are therefore:

— minimize objective 1: distance between the center of gravity of all the components and the geometric center of the shelter, given by Eq.3.8 in Chapter 3.

— maximize objective 2: distance between box 12 and the set made up of cabinet 3, cabinet 4 and cabinet 5, given by Eq.3.10 in Chapter 3.

Overall, the multi-container layout problem model is stated as follows:

Given:

— Twenty components $\boldsymbol{c} = \{c_1, ..., c_{20}\}$with fixed dimensions and three sub-containers with flexible sizes,

— Fixed assignment of components to sub-containers $r : \{\{c_1, c_2\}, \{c_3, ..., c_{14}\}, \{c_{15}, ..., c_{20}\}\} \rightarrow \{1, 2, 3\}$,

— Total container dimensions $W = 5930, H = 2306$ and the area $z = W \times H$.

Determine:

— The sub-container size $z_i, i \in [1, 2, 3]$ and the arrangement (location and orientation) $\boldsymbol{p} = \{p_1, p_2, ..., p_n\}, n = 20$ of components $\boldsymbol{c} = \{c_1, c_2, ..., c_n\}, n = 20$, optimize objective 1 and objective 2 and satisfy geometrical (non-overlap and non-protrusion) and functional constraints (accessibility, edge), given by Eq.3.12 in Chapter 3.

5.4.3 Capacity evaluation

The assignment of components to each zone is fixed, so we analyze the density and capacity for each zone separately. The density of solid and virtual components in storage, technical and operator zone using the formula Eq.3.13 and Eq.3.14 in Chapter 3, are listed in Table 5.12, Table 5.13 and Table 5.14.

Table 5.12 – Density of components in storage zone.

Density	Value
Density of solid components	β_s =0.69
Density of virtual components	β_v =0.0
Sum of density	β_{sv} =0.69

If the overlap between virtual components are ignored, then the sum of the density of technical zone is $\beta_{sv} = 1.01 > 1$, and the sum of the density of operator zone is $\beta_{sv} = 1.08 > 1$, which indicates that the problem can not be solved. However, by applying the method presented in Chapter 3 with 1000 iterations, it found the minimum

Table 5.13 – Density of components in technical zone.

Density	Value
Density of solid components	β_s =0.36
Density of virtual components	β_v =0.65
Sum of density	β_{sv} =1.01

Table 5.14 – Density of components in operator zone.

Density	Value
Density of solid components	β_s =0.44
Density of virtual components	β_v =0.64
Sum of density	β_{sv} =1.08

occupied space of the storage zone where $\beta_{c_1} = 0.7$, the technical zone $\beta_{c_2} = 0.76$, and the operator zone $\beta_{c_3} = 0.79$, which shows a priori feasibility of the shelter optimization. The corresponding compact layout design is shown in Fig.5.20. Besides, when we place

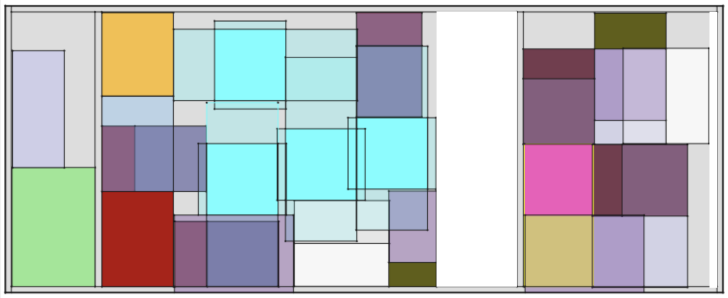


Figure 5.20 – Compact configuration of the shelter.

the components sequentially, it may generate some small spaces, which are empty, but no components can be placed. Indeed, these small spaces should be identified as infeasible spaces Thus, the minimum occupied space is modified as the product of minimum occupied length and width for each zone and the capacity is updated as: $\beta_{c_1} = 1$, $\beta_{c_2} = 0.82$, $\beta_{c_3} = 0.90$.

After obtaining the prior feasible information, we apply optimization algorithm to find the set of feasible solutions of the shelter.

5.4.4 Boundary estimation of three zones

The minimum width and height occupied space defines the minimum area $\hat{\beta}_{c_i}$ of sub-container $i, i = [1, 2, 3]$, :

$$\begin{aligned} Storage\ zone: &\quad \hat{\beta}_{c_1} = 700 \times 2306\ mm^2 \\ Technical\ zone: &\quad \hat{\beta}_{c_2} = 2795 \times 2306\ mm^2 \\ Operator\ zone: &\quad \hat{\beta}_{c_3} = 1553 \times 2306\ mm^2 \end{aligned}$$

The partition in the "|" form will be divided the layout container vertically such that the sub-container is mainly defined by the $x-$coordinate of each partition. The width of compartment $W = 5930\ mm$ and the width of each partition 57 mm and $= 53\ mm$. Thus, the maximum movement for the partition will be

$$5930 - 700 - 2795 - 1553 - 57 - 53 = 772\ mm$$

So we have $\alpha = 772 \times 2306\ mm^2$, the area is represented as the white regions in Fig.5.20. And the search complexity interval is reduced by $1 - 772/5920 = 86\%$.

5.4.5 Optimization results and similarity analysis

An initial population has $N = 200$ random individuals. The initial temperature t_s is initialized as $t_s = 100$. Set the cooling rate $r = 0.9$ and the total number of iterations $L = 250$ to perform the annealing process.

In the given number of iterations, 16 solutions of three-compartment layout under-determined zone area considering the formulated objectives and the constraints, are Pareto-optimal designs as shown in Fig. 5.21. And Fig. 5.22 plots the corresponding $x-$coordinate of each multi-container shelter layout configuration. The similarity indicators for paired layout designs formulate a similarity symmetric matrix, represented in Fig. 5.23(a). The visualization tool provides information on the hierarchical similarity of designs, shown in Fig. 5.23(b). Here, we assume there are five clusters, as shown in Fig. 5.24, the same grouped solutions have the same color. And each representative is shown in Fig. 5.25.

The most of the components of the shelter in the storage and operator zone are consistent with the initial proposed scheme shown in Fig.5.19 but with more diverse configuration in technical zone, which is the most complex area in this shelter. For the different

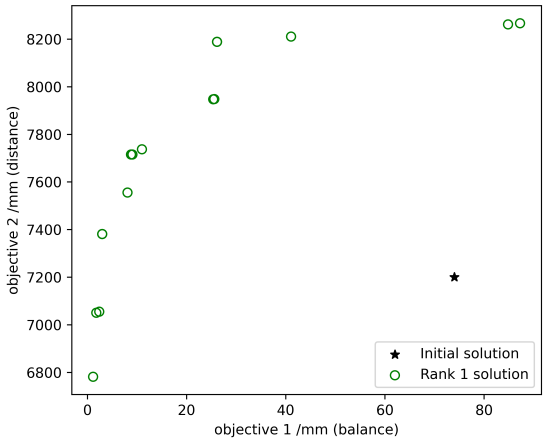


Figure 5.21 – Display of rank 1 solution.

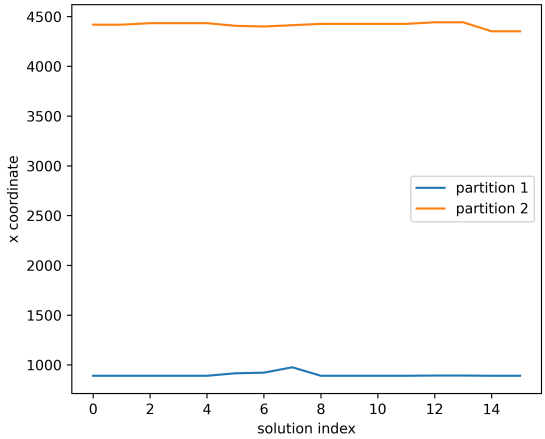
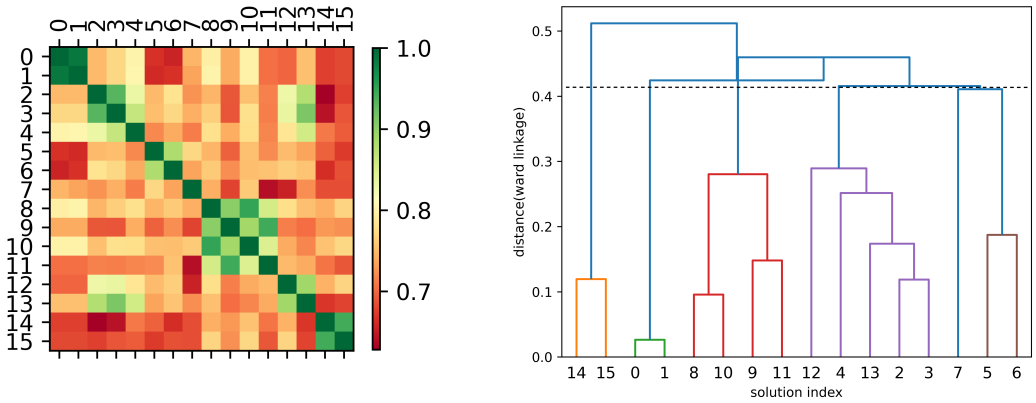


Figure 5.22 – x-coordinate of the partition.

zone area, the developed algorithm can find various shelter configurations using placement-related sequences. The solution initially created by the engineers of Thales is far from the obtained optimal solutions and is dominated by them as well. On the one hand, it turns out that the optimization algorithm can reach the similar determined sub-container configuration compared with the initial solution, for example solution 0 in Fig. 5.25. And it proves the feasibility of the proposed method. On the other hand, the optimized shelter configurations have better values in the objectives and demonstrates the effectiveness of the developed method. Due to the dense capacity of the shelter, it is difficult to satisfy



(a) Similarity matrix. (b) Hierarchical cluster dendrogram.

Figure 5.23 – Similarity analysis of multi-container shelter.

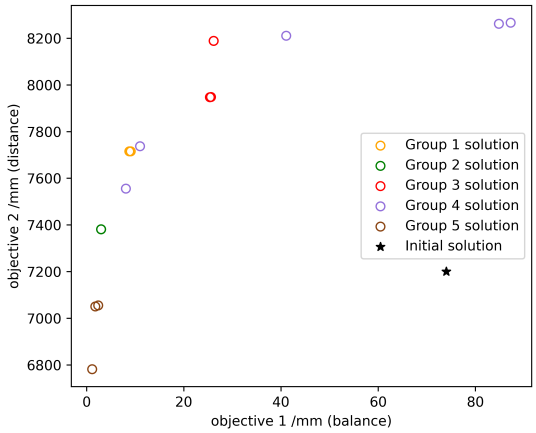
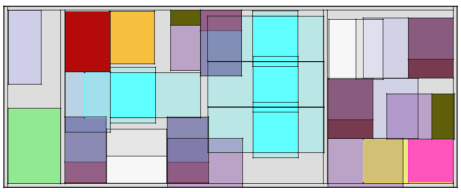
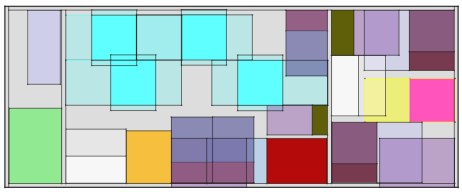


Figure 5.24 – Display of clustered solutions in objective space.

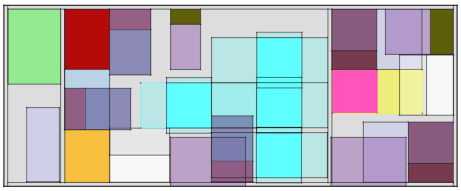
the non-overlap constraints. Besides, even if the solution is geometrically feasible, certain components may be inaccessible considering the limited available space. Consequently, finding feasible solutions that achieve better objective function values becomes extremely hard. However, the developed optimization algorithm resolves the difficulties arising from the problem. The constructive algorithm effectively generates feasible solutions while the SA algorithm optimizes the combination of shelter configurations.



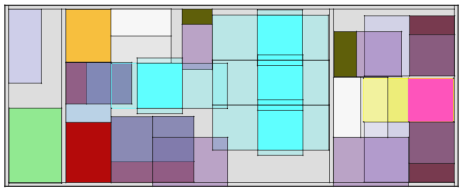
(a) solution 14.



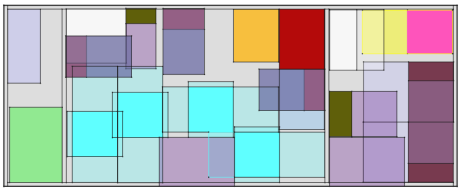
(b) solution 0.



(c) solution 9.



(d) solution 12.



(e) solution 5.

Figure 5.25 – Display of optimal designs.

5.5 Conclusion

This chapter has shown that the layout optimization approach, developed in this manuscript, can be applied to various industrial layout applications. The method was tested on a problem of arranging a shelter in single- and multi- container with very limited space. 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, whereas 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. The multi-objective simulated annealing is applied as a global optimizer and is in charge of efficiently exploring the search space. The constructive placement is used to place components following the sequences proposed by simulated annealing. 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. The similarity visualization, which supports the interaction approach, provides a tool suitable for solving multi-objective layout optimization problems.

CONCLUSION AND PERSPECTIVE

Conclusion

Layout is an important and an active research area. The thesis manuscript contributes to a multi-objective optimizer based on constructive approach to solve the novel layout problem model. The method is developed to address the difficulties of component accessibility in the multi-objective layout design and can be extended to layout problem with multiple containers. Accessibility is one particular layout functional constraint that expresses the maintainability, inspection as well as reachability to components. Moreover, interactive environment is designed. In particular, the capacity index and similarity indicator is integrated into the interactive optimization process.

The layout problems are inherently multidisciplinary tasks and can be solved as optimization problems. To translate the designer requirements into optimization variables, design constraints and objectives, first of all, the layout is formulated by the new component of solid and virtual parts. The model, taking into account the virtual components and the accessibility to components, requires an effective optimization algorithm for addressing the feasible difficulty. Without loss of generality, two objectives, namely layout balance and activities, are considered. Then, a new capacity index is proposed to evaluate the feasible difficulty of the layout problem. This index defines an alternative to the traditional calculation of the compactness of a layout problem. It makes it possible to demonstrate, in a reliable way, if the problem can be solved or not. Furthermore, the interactive environment allows integrating mathematical optimization with human decision-making during conceptual design of the layout problem, with a focus on similarity analysis applied to obtained layout alternatives and help the expert select the final decision.

The layout model contains multiple objectives and can be solved by a multi-objective optimizer which considers multiple objectives simultaneously and searches for a set of compromised solutions. Simulated annealing is a stochastic neighborhood search approach for global optimization. The ease of implementation makes simulated annealing a popular method for solving large and practical problems. However, the individual-based feature

limits the application, especially in the multi-objective problems. To overcome the issue, archive-free SA employs the dynamic selection based on the concept of non-dominated sort and the mechanism of crowding distance calculation for multi-objective optimization is proposed. Continuous and combinational problems are tested, overall, the results proved that archive-free SA can provide, most of the time, very competitive results compared to others, which makes it interesting to solve complex layout problems. For comparison, we have applied the proposed archive-free simulated annealing optimization approach to the simple layout examples that formulated by the multi-objective layout model. The experimental results has found the fewer layout alternatives and showed feasible difficulty in the layout optimization. The optimizer may get stuck and fail to jump out of (in-)feasible search region. Alternatively, the layout problem can be formulated as a combinatorial problem. Simulated annealing demonstrated the robustness in combinatorial optimization. Thus, a multi-objective optimization method integrates sequential placement with simulated annealing is developed instead.

Component accessibility is one important design requirements in the industrial engineering. The integration of virtual spaces is necessary but is not sufficient for the component accessibility. Therefore, additional accessibility analysis is conducted by the constructive placement. Different strategies have been proposed and investigated in order to be generic and efficient, thus suited to a wide set of layout problems. 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. The strategy comparisons confirm that the space-filling strategy can effectively generate feasible solutions and reduce computational efforts. Layout solution is constructed sequentially, which is a combination process. Simulated annealing search technique explores combinations of component configurations and optimizes both objectives simultaneously. Archive-free SA is in charge of exploring efficiently the search space to propose promising alternatives. Most of researches on layout problems focus on placing components on a single container. Nevertheless, the layout with multiple containers has always been a hot and challenging point. The method was then extended to layout with multiple containers. The optimization complexity can be highly reduced through the proposed boundary estimation which transforms the global search into local search.

The layout optimization method was tested on the problem of arranging a shelter in single- and multi- container with very limited space. The experimentation proves

that the proposed optimization is effective in ensuring accessibility and finding high-qualified solutions compared to the existing algorithms, which enable a truly interactive optimization process. Moreover, the similarity analysis demonstrates good diversity of the obtained layout set, which can be applied as an interactive tool.

Perspective

This thesis contributed to the multi-objective layout optimization approach. Even if the outcomes of the research were positive and allowed to effective and interactive optimization, there are still a few aspects of research to be explored. These directions of research are briefly described here:

— Introduce the interactions between the designer and the optimization algorithm. In addition to using the similarity analysis to help the designer compare and select the preferable solution,artificial intelligence can also be integrated to help the design further improve the solution. For example, analyzing the evaluation of objectives according to the expressed objective importance.

— Define different shapes of components. We assume all the components are rectangles in the layout model. The definition highly simplifies the constraints and objectives formulation. To go further, the layout could have other shapes such as circles, triangles and polygon etc.

— At present the approach has been applied the space generation of rectangular shapes. Further research could adapt the available space generation to the free-form component. The space generation is used to avoid overlapping component. One can rely on computer-aided design to detect collisions and report the overlap area; or use sets of circles replace the free-form shape, and then compute the overlap between components with simple distance computations.

— The proposed approaches have been applied to two-dimensional layout problems but it can be applied to other such as, the three-dimensional layout problem where components have different heights. Adapt the placement strategy to account for the effects of gravity and surface contact area is necessary.

Part II

French version

INTRODUCTION

L'optimisation d'agencement est multidisciplinaire. Les applications peuvent être l'agencement de station spatial, l'agencement de véhicule, l'agencement d'architecture, l'agencement de puce, l'agencement de système de fabrication etc. Un excellent agencement peut améliorer efficacement les performances du système. Les problèmes portent généralement sur la recherche des agencements optimaux des composants (i.e. équipements, machines) à l'intérieur du conteneur (i.e. atelier, usine) pour optimiser les objectifs et respecter les contraintes géométriques et fonctionnelles. Les composants les plus rencontrés sont représentés par des rectangles de tailles déterminées. Dans tous les problèmes d'agencement, les contraintes de non-chevauchement entre composants et les contraintes d'appartenance au conteneur sont présentes. L'orientation ou l'alignement doit définir les relations fonctionnelles entre les composants. Les contraintes fonctionnelles assurent le bon fonctionnement du système. Une majorité d'études optimise, par exemple, la distribution de masse, ou l'exigence de contiguïté et le coût de manutention du matériel.

Sans techniques d'optimisation, le problème d'agencement ne peut être résolu avec succès. Cependant, la région satisfaite par la contrainte, l'objectif non linéaire et non convexe de la formulation de l'agencement rendent l'optimisation complexe par nature. Les solutions satisfaites des contraintes peuvent être obtenues en pénalisant les violations de contraintes dans la fonction objectif ou générées à partir de domaine réalisable. Les problèmes d'agencement les plus couramment rencontrés ont de multiples objectifs qu'il convient d'optimiser. En fait, les problèmes multiobjectif peuvent être résolus par des techniques d'optimisation monoobjectif ou d'optimisation multiobjectif. Le premier cas transforme plusieurs objectifs en une fonction d'agrégation utilisant des poids prédéfinis, il existe donc une solution unique correspondante. Dans cette dernière approche, un optimiseur multiobjectif considère plusieurs objectifs simultanément et vise à trouver des solutions compromises, connu sous le nom de front de Pareto. De plus, le processus d'optimisation peut être utilisé comme aide à la décision pour le concepteur. Lorsqu'il est confronté à de multiples choix en fonction du risque et de l'incertitude, le décideur peut sélectionner la conception de la configuration pour obtenir le meilleur compromis de performances du système.

Ce manuscrit de thèse est donc structuré en plusieurs chapitres de la manière suivante. Le Chapitre 1 réalise un état de l'art des recherches effectuées en optimisation multiobjectif, notamment sur les les concepts fondamentaux, les approches classiques d'optimisation, les effets d'archive et les stratégies qui tentent d'améliorer la convergence et la diversité dans l'optimisation multiobjectif. Ce chapitre décrit également les problématiques d'agencement d'espace (la représentation, la formulation et les approches d'optimisation). Le Chapitre 2 est dédié aux méthode de recuit de simulation basé sur la population en tenant compte de l'archive externe pour le problème multiobjectif. Nous avons expliqué le cadre général et étudié les performances des cas basés sur des archives et sans archives. La comparaison est effectuée sur des instances de référence continues et combinatoires avec la méthode connue d'optimisation multiobjectif. En particulier, la résistance et l'amélioration de la convergence sont étudiées plus en détail. Le Chapitre 3 présente le nouveau modèle d'agencement qui prend en compte l'accessibilité des composants et des outils d'interaction conçus. Tout d'abord, la définition du composant, les contraintes de conception et les objectifs sont détaillés. Puis un indicateur de capacité est proposé pour évaluer la difficulté d'optimisation du tracé. Enfin, un environnement interactif permettant d'intégrer l'optimisation mathématique est présenté, avec un focus sur l'analyse de similarité laissant au concepteur de faire des choix par rapport aux solutions générées par l'algorithme d'optimisation. Le Chapitre 4 propose un ensemble de méthode permettant de résoudre ces problèmes d'agencement en intégrant l'analyse de l'accessibilité dans la méthode de recuit simulé. L'analyse de l'accessibilité est menée par la méthode du placement constructif. La procédure de placement et les comparaisons de stratégies de placement sont décrites en détail. Après cela, les approches orignales d'optimisation de la disposition multiobjectif avec le recuit et le placement simulés multiobjectif sont proposées. L'optimisation d'agencement multi-conteneurs est ensuite détaillée. Le Chapitre 5 présente des applications industrielles de la démarche d'optimisation proposée dans ce manuscrit. Une conclusion générale peut être déduite sur les performances d'optimisation selon laquelle un placement efficace satisfaisant les contraintes conduira à un optimiseur qui permettra un processus d'optimisation véritablement interactif.

Chapter 1

État de l'art

Du point de vue mathématique, l'optimisation consiste à identifier la solution, c'est-à-dire une ou plusieurs variables de décision, en minimisant ou en maximisant une fonction donnée ou un ensemble de fonctions tout en respectant un ensemble de contraintes. Ainsi, deux types de problèmes d'optimisation sont distingués dans la littérature : les problèmes d'optimisation monoobjectif et multiobjectif. Dans ce chapitre, dans un premier temps, nous avons décrit la formulation de problèmes monoobjectif et multiobjectif. Nous avons ainsi présenté les principales différences entre les solutions « optimales » dans ces deux cas et passé en revue les méthodes de résolution. Dans le passé, les techniques d'optimisation les plus couramment utilisées étaient déterministes, par exemple, la méthode basée sur le gradient qui utilisait des informations sur le gradient pour rechercher l'espace de solution près d'un point de départ initial. Par rapport aux approches stochastiques, les méthodes basées sur le gradient convergent plus rapidement et peuvent obtenir des solutions pour le problème convexe mais ne peuvent pas résoudre facilement les problèmes d'optimisation non convexes. D'autres types de méthodes d'optimisation sont des approches stochastiques. Ces méthodes conviennent à la recherche globale en raison de leur capacité à explorer et à trouver des zones prometteuses dans l'espace de recherche à un temps de calcul abordable.

Les méthodes déterministes et stochastiques dans l'optimisation monoobjectif peuvent être étendues à l'optimisation multiobjectif. Les approches basées sur la scalarisation, la domination, la décomposition, les indicateurs ainsi que les interactions ont été distinguées. Un consensus critique est apparu sur le fait que l'utilisation de l'archive externe tout au long du processus d'optimisation peut améliorer les performances au détriment de l'efficacité des calculs. Pour les algorithmes d'optimisation multiobjectif, l'une des caractéristiques clés est la capacité de trouver une bonne approximation du compromis optimal parmi les objectifs multiples. Cependant, deux questions restent ouvertes. Tout d'abord, la sélection de la solution devient fortement dépendante de la mesure de la diversité de sorte que la convergence du front de Pareto vers le vrai front de Pareto est progressive-

ment détériorée. Un deuxième problème est lié à l'espace de décision. Une majorité d'algorithmes se concentrent uniquement sur la distribution des solutions dans l'espace objectif. Néanmoins, une bonne représentation des solutions dans l'espace décisionnel est également importante du point de vue du processus décisionnel.

La deuxième grande partie de ce chapitre est le problème d'agencement. Le problème de coupe et d'emballage (C & P) est courant dans l'industrie. Un problème de coupe vise à maximiser le nombre de composants placés à l'intérieur du conteneur et un problème d'emballage se réfère à la minimisation du nombre de conteneurs utilisés pour placer tous les composants. Dans un problème d'agencement, la représentation des composants, les objectifs et les contraintes peuvent être différents. Par exemple, les composants sont connectés géométriquement dans un problème C & P alors que le placement des composants est connecté fonctionnellement dans un problème d'agencement. Les problèmes d'agencement font référence à la recherche d'arrangements optimaux de plusieurs composants dans la zone du conteneur. La procédure principale d'un problème d'agencement commence généralement par la représentation du problème, suivie de la formulation du problème, puis de l'optimisation du problème. Nous avons fourni une classification d'agencement en termes de représentation et de formulation. De l'examen de la littérature, on peut conclure qu'il s'agit toujours d'un domaine de recherche ouvert et actif. Cela motive l'auteur à travailler dans l'optimisation d'agencement. Donc, les recherches précédentes dans le domaine ont été analysées en tenant compte des approches de formulation et de résolution.

Le concepteur définit la description du problème, y compris les données et les exigences spécifiées. Et l'information sera traduite en formulation de problème. Il produit le modèle résolu par des techniques d'optimisation. Diverses méthodes ont été utilisées pour résoudre le problème d'agencement, à savoir l'approche exacte et l'approche méta-heuristique. Dans la littérature actuelle, il existe des tendances des techniques d'optimisation combinant l'effort de construction et la méthode méta-heuristique pour résoudre le problème d'agencement. Le concept d'approche de construction a beaucoup retenu l'attention des chercheurs en raison de sa faisabilité. De plus, nous avons clôturé ce chapitre en présentant un problème d'agencement de important, à savoir le problème d'agencement multi-conteneurs. Nous avons remarqué que la recherche sur la taille des conteneurs flexibles est un domaine relativement peu couvert.

Chapter 2

Recuit simulé basé sur la population pour les problèmes multiobjectif

Au cours de la dernière décennie, les algorithmes méta-heuristiques ont été reconnus pour être très efficaces dans la résolution de problèmes d'optimisation multiobjectif. Le recuit simulé a un avantage sur les autres algorithmes méta-heuristiques en termes de facilité de mise en œuvre et donne des solutions raisonnablement bonnes pour de nombreux problèmes pratiques. Il part d'un point initial et recherche de nouveaux points proches pour trouver la solution optimale globale. Il y a eu quelques tentatives pour étendre recuit simulé à l'optimisation multiobjectif. Dans la plupart des premières tentatives, une seule fonction objective a été construite en combinant les différents objectifs en une seule fonction objective à l'aide d'une méthode de somme pondérée. Le problème est de savoir comment choisir les poids à l'avance. Il existe également des alternatives utilisées à cet égard. Afin de développer un optimiseur efficace avec une structure simple, deux algorithmes de recuit simulé multiobjectif sont proposés : l'un est recuit simulé basé sur l'archive et l'autre est recuit simulé sans archive. Les deux algorithmes intègrent l'idée de domination de Pareto.

Recuit simulé basée sur l'archive adopte l'archive de taille limitée pour conserver les solutions non dominées. Il est conçue à l'aide du nouveau mécanisme d'acceptation, où les solutions sont divisées en états dominés, non dominés et dominés. Si la solution actuelle et la nouvelle solution ne sont pas dominées, il n'est pas facile de dire lequel est le meilleur selon les seuls critères de domination. Par conséquent, nous comparons les deux cas en considérant l'archive : si la nouvelle solution n'est dominée par aucun point de l'archive, alors nous la considérons comme une meilleure solution; si la nouvelle solution est dominée par un point existant, alors nous acceptons la nouvelle solution lorsqu'elle se rapproche du front de Pareto optimal. Enfin, il applique la deuxième boucle de recuit à l'archive

finale à la région de recherche près des solutions non dominées actuelles pour agrandir le nombre de points finaux le long du front de Pareto. L'utilisation de l'archive apporte des avantages substantiels, mais au prix du temps de calcul.

Recuit simulé sans archive utilise la sélection dynamique basée sur le concept de tri non dominé et le mécanisme de calcul de la distance d'encombrement. La sélection dynamique préserve la non-domination et les solutions distribuées dans la population. C'est un algorithme simple avec l'avantage de la facilité de mise en œuvre. Il intègre la capacité de la population à rechercher différentes régions et à collecter des informations auprès de différents individus, et conserve les solutions non dominées bien distribuées dans la population plutôt que dans les archives externes. Afin d'élargir l'espace de recherche tout en accélérant la vitesse de convergence, une sélection dynamique basée sur des critères de tri rapide sans domination et de distance d'encombrement est intégrée à l'optimisation. Un individu est représenté par un vecteur à deux éléments, le rang et la valeur de la distance d'encombrement. La sélection dynamique compare le meilleur individu avec le plus petit rang mais la plus grande distance d'encombrement dans la population actuelle avec l'individu séquentiel dans la nouvelle population, et en sélectionne l'un des deux jusqu'à ce qu'il y ait une population complète.

Les performances de nos méthodes proposées sont validées et comparées avec l'algorithme génétique bien connu NSGA II sur un certain nombre de problèmes multiobjectif continus, un ensemble de problèmes de sac à dos multiobjectif combinatoires 0-1. Les principales raisons de choisir NSGA II pour la comparaison sont qu'il est basé sur l'évolution de la population et qu'il surpasse de nombreux algorithmes populaires. Les problèmes de test sont choisis de manière à étudier systématiquement différents aspects d'un algorithme : convergence, précision, efficacité, etc.

De plus, nous analysons la résistance à la convergence et la méthodologie d'amélioration. La résistance de convergence apparaît lorsqu'il existe de nombreux points non dominés proches du front de Pareto. Pour améliorer la convergence, il est possible de remplacer le point courant par le meilleur point basé sur la mesure de distance plutôt que sur la domination. L'idée est de générer de nouvelles solutions proches des solutions actuelles et de minimiser la distance entre les solutions pour aider l'optimisation à sortir de la zone de domination étroite actuelle et à trouver une meilleure solution voisine. Le critère de distance améliore la convergence sans nuire à la diversité. Il trouve de meilleures solutions mais au prix des évaluations de fonctions.

Chapter 3

Modèle de problème d'agencement multiobjectif et interaction

Comme expliqué au Chapitre 1, le modèle du problème d'optimisation d'agencement est une formulation multiobjectif. Le modèle vise à traduire les exigences du concepteur en variables, contraintes et objectifs d'optimisation. En général, le problème d'agencement consiste à placer des composants rectangulaires dans le conteneur rectangulaire. Sur la base des différentes propriétés fonctionnelles des composants, le nouveau composant comprenant les parties solides et virtuelles est défini, et l'accessibilité aux composants d'un agencement est explicitement exprimée. es composants virtuels peuvent chevaucher des composants virtuels et n'avoir aucune masse. Ces composants virtuels sont aussi les espaces accessibles. Les composants solides ne peuvent pas chevaucher des composants solides ou virtuels et avoir une masse. La définition des composants permet de modéliser la plupart des problèmes d'implantation, par exemple le problème de placement des machines dans l'usine. Les composants virtuels peuvent modéliser l'espace libre autour de la machine nécessaire au bon fonctionnement ou à la maintenance, ou représenter les couloirs utilisés pour la circulation des marchandises entre les équipements. Les contraintes géométriques telles que les contraintes de non-chevauchement entre composants et les contraintes d'appartenance au conteneur; les contraintes fonctionnelles dont l'orientation et l'accessibilité sont traduites en fonctions mathématiques explicites. Sans perte de généralité, deux objectifs, à savoir l'équilibre de l'aménagement et les activités, sont considérés.

Ensuite, un nouvel indice de capacité est défini pour mesurer la complexité réalisable d'un problème d'agencement et fournir des informations a priori sur la possibilité de résoudre le problème. Une approche innovante est proposée pour évaluer la capacité en trouvant l'espace minimum occupé des composants. Afin de trouver l'espace occupé minimum des composants, nous formulons le problème comme un processus d'emballage. D'une part, l'idée principale de l'emballage est de maximiser l'utilisation de l'espace, en

d'autres termes, de placer tous les composants aussi compacts que possible. D'autre part, le processus d'emballage peut être traité comme un problème de combinaison qui emballe de manière constructive les composants dans une séquence donnée. Afin de placer les composants de manière compacte, les conditions suivantes doivent être satisfaites : le nouveau composant solide se rapproche de ces composants solides déjà placés selon la convention en bas à gauche; le chevauchement entre les nouveaux composants virtuels et les composants virtuels placés doit être maximisé. La génération d'espace est notamment développée pour l'évaluation efficace des non-chevauchement contraintes. Grâce à l'intégration de la nouvelle génération d'espace, l'évaluation des contraintes géométriques est plus efficace. L'évaluation de la capacité est mise en œuvre par le recuit simulé et l'optimisation de l'emballage constructif. Cet indice permet de démontrer, de manière fiable, si le problème peut être résolu ou non.

L'environnement interactif permet d'intégrer l'optimisation mathématique à la prise de décision humaine lors de la conception conceptuelle du problème d'agencement. Le concepteur interagit avec la représentation de l'optimisation du problème en ajoutant, supprimant et modifiant des objectifs, des contraintes et des composants par interface graphique, développée dans ce doctorat. Aussi, la prise de décision finale utilise les outils de visualisation pour comparer les solutions en analysant les objectifs et les similarités qui caractérisent les informations quantitatives. Afin d'aider le concepteur à distinguer les schémas de configuration, un indicateur de similarité est notamment proposé, qui exprime quantitativement la similarité du schéma de configuration actuel avec d'autres schémas de configuration. Deux types de définitions de similarité sont proposées en fonction de la différence de grille et de la position relative. La différence de grille définit un indicateur de similarité basé sur la différence élément par élément qui ne peut pas différencier la configuration symétrique et devient chronophage à mesure que le nombre de grilles augmente. Alors que l'indicateur de similarité basé sur la position relative peut surmonter ces problèmes. Un tracé matriciel affiche une matrice bidimensionnelle pour une conception d'agencement par paires formulée par des indicateurs de similarité. Un dendrogramme est un diagramme qui montre sla relation hiérarchique entre les similitudes d'agencement. Un nuage de points présente des solutions dans l'espace objectif. Les outils de visualisation sont développés pour la visualisation multidimensionnelle des solutions. En résumé, les principales fonctions de l'environnement interactif sont la visualisation, la modification, l'exploration du modèle géométrique de la solution.

Chapter 4

Optimisation multiobjectif du problème d'agencement

Ce chapitre est consacré à l'approche d'optimisation d'agencement. L'optimisation vise à trouver les solutions optimales du problème d'agencement formulé au Chapitre 3. Le problème d'agencement, en tenant compte des composants virtuels et de l'accessibilité aux composants. Deux objectifs, à savoir l'équilibre de l'aménagement et les activités, sont considérés. Intégrer l'accessibilité des composants comme contraintes fonctionnelles assure la maintenance ou le bon fonctionnement des composants. Cependant, la prise en compte des contraintes fonctionnelles augmente la complexité de l'optimisation de l'agencement.

L'utilisation d'un algorithme stochastique est nécessaire compte tenu de la grande complexité des problèmes d'implantation, en particulier des applications industrielles. Cependant, les résultats expérimentaux utilisant recuit simulé sans archive sur des exemples d'agencement ont démontré la difficulté réalisable de l'optimiseur continu. Observez que la plupart des solutions ne respectent pas les contraintes de non-chevauchement et sont par conséquent irréalisables. Par conséquent, un algorithme d'optimisation basé sur une approche constructive est proposé pour résoudre le problème.

Le placement constructif est introduit pour générer des configurations d'agencement complètes. La stratégie de placement est proposée pour placer le composant dans un espace approprié par rapport aux contraintes. La stratégie concerne deux aspects, à savoir le placement des composants et la sélection de l'espace. Pour un composant, il a quatre configurations de rotation. Le composant sera placé dans les coins de l'espace sélectionné avec quatre rotations. Le placement garantit que moins d'espace de marge est généré et que la contrainte de non-chevauchement est satisfaite automatiquement. De plus, nous caractérisons l'accessibilité des composants comme une contrainte lors du processus de construction. Le placement, non seulement intègre la génération d'espace de composants solides et virtuels garantissant le non-chevauchement des composants, mais

introduit également l'idée de chemin de connexion assurant l'accessibilité des composants. Pour placer un composant, il convient de décider quel espace disponible sera utilisé. Trois stratégies de sélection sont proposées : à savoir vérifier toutes les combinaisons ; sélectionnez l'espace le plus petit ; sélectionnez l'espace le plus grand. Les comparaisons de stratégies confirment que sélectionnez l'espace le plus petit, à savoir la stratégie de remplissage d'espace peut générer efficacement des solutions réalisables et réduire les efforts de calcul.

Le placement constructif traite la génération d'agencemet comme un problème combinatoire discret. Dans le Chapitre 2, les résultats expérimentaux prouvent que la recuit simulé sans archive est moins couteux en termes de calcul et produit de meilleures approximations sur l'optimisation combinatoire. Recuit simulé sans archive se charge d'explorer efficacement l'espace de recherche pour proposer des alternatives prometteuses. Le placement constructif est conçu pour contourner la difficulté découlant des contraintes. L'algorithme est basé sur l'optimisation combinatoire : algorithme recuit simulé sans archive, pour déterminer les séquences de placement et de configuration; algorithme de placement constructif, pour placer les composants séquentiellement.

La méthode a ensuite été étendue à l'agencement avec plusieurs conteneurs. Le modèle de problème de disposition multi-conteneurs est énoncé comme suit : étant donné l'affectation fixe des composants aux sous-conteneurs, déterminez la taille du sous-conteneur et le placement des composants, de manière à trouver le front de Pareto du problème de disposition multi-conteneurs. Pour un problème de disposition multi-conteneurs avec affectation de composants fixes, le placement des composants est limité par la taille du sous-conteneur correspondant. Pour s'assurer que les composants peuvent être placés à l'intérieur de chaque sous-conteneur, la zone occupée des composants doit être inférieure à la zone du sous-conteneur. Ainsi, nous appliquons la méthode du Chapitre 3 pour trouver l'espace minimum occupé des composants pour chaque sous-conteneur. Estimant ainsi de nouvelles limites inférieures et supérieures pour chaque partition de sous-conteneur. Et la complexité de l'optimisation peut être fortement réduite grâce à l'estimation des limites proposée qui transforme la recherche globale en recherche locale.

Chapter 5

Applications industrielles

Ce chapitre est dédié aux applications industrielles de la démarche d'optimisation d'agencement proposée dans ce manuscrit. Pour chaque cas d'étude présenté dans ce chapitre, toutes les étapes de la méthode sont décrites: la description, la formulation, l'analyse de capacité, la résolution du problème, ainsi que l'analyse de similarité pour une interaction ultérieure. Les problèmes d'optimisation d'agencement du shelter présentés dans ce chapitre portent sur des exemples industrielles proposé par l'entreprise français Thales SIX France, implantée à Cholet. Un shelter est un abri technique mobile dans lequel sont disposés des équipements, tels des armoires, des bureaux et autres boîtiers électriques. Ce local technique est le plus souvent fixé à l'arrière d'un véhicule et est dédié à des missions de communications lors d'opérations militaires.

Le premier problème d'optimisation d'agencement présentés dans ce chapitre portent sur l'agencement d'un mono-conteneur shelter en deux dimensions. Comme le suggère le Chapitre 3, les composants peuvent être séparés en deux catégories: composants solides et composants virtuels. En appliquant la méthode de calcul exposée dans le Chapitre 3, nous obtenant un indice de capacité égale à 0.55, ce qui montre à priori la feasiabilité du problème d'optimisation d'agencement. Par ailleurs, si les composants de cet agencement avaient tous été considérés comme des composants solides et si nous avions appliqué la formulation traditionnelle de la densité, la compacité du problème serait égale à 0.85. L'expérimentation prouve que l'optimisation proposée est efficace pour générer des alternatives et trouver des solutions hautement qualifiées dans un temps de calcul raisonnable. De plus, l'analyse de similarité démontre une bonne diversité de l'ensemble d'agencement obtenu, qui peut être appliqué comme un outil interactif.

Le second problème d'optimisation d'agencement proposé dans ce chapitre porte sur un problème du shelter avec le composant de grande taille. Les composants sont en solides et virtuels. Le composant de plus grande taille, à savoir le composant amplificateur, comporte trois composants virtuels : deux composants virtuels d'une largeur de 2469 mm et d'une hauteur de 600 mm, un composant virtuel d'une largeur de 800 mm et d'une

hauteur de 841 mm, et occupe près de la moitié de l'espace conteneur. En appliquant la méthode de calcul exposée dans le Chapitre 3, nous obtenant un indice de capacité égale à 0.77, démontrant à priori la difficulté du problème du shelter. Les résultats de l'optimisation confirment que l'optimisation proposée peut générer efficacement des solutions réalisables telles que les contraintes de placement sont satisfaites et tendent à réduire les efforts de calcul.

Le troisième cas d'étude est une extension du duex premiers problèmes d'agencement en trois conteneurs: zone de stockage nommée, zone technique et zone opérateur. Les composants sont en solides et virtuels. L'affectation des composants à chaque zone est fixe, nous analysons donc la densité et la capacité de chaque zone séparément. Si le chevauchement entre les composants virtuels est ignoré, la somme de la densité de la zone technique égale à 1.01 et la somme de la densité de la zone opérateur égale à 1.08, d´emontrant à priori l'infaisabilité du problème du shelter. En appliquant la méthode de calcul exposée dans le Chapitre 3, nous obtenant un indice de capacité égale à 0.76 et 0.79. Du fait de la capacité dense de l'abri, il est difficile de satisfaire les contraintes de non-chevauchement. De plus, même si la solution est géométriquement faisable, certains composants peuvent être inaccessibles compte tenu de l'espace limité disponible. Par conséquent, trouver des solutions réalisables qui permettent d'obtenir de meilleures valeurs de fonction objectif devient extrêmement difficile. Cependant, l'algorithme d'optimisation développé résout les difficultés découlant du problème. L'algorithme constructif génère efficacement des solutions réalisables tandis que l'algorithme SA sans archive optimise la combinaison des configurations d'abris. De plus, l'analyse de similarité démontre une bonne diversité de l'ensemble d'agencement obtenu, qui peut être appliqué comme un outil interactif.

CONCLUSION ET PERSPECTIVE

Conclusion

L'agencement est un domaine de recherche important et actif dans les instituts de recherche et l'industrie. Le manuscrit de thèse contribue à un optimiseur multiobjectif basé sur une approche constructive pour résoudre le nouveau modèle de problème d'agencement. La méthode est développée pour résoudre les difficultés d'accessibilité des composants dans la conception d'agencement multiobjectif et peut être étendue au problème d'agencement avec plusieurs conteneurs. L'accessibilité est une contrainte fonctionnelle particulière d'agencement qui exprime la maintenabilité, l'inspection ainsi que l'accessibilité aux composants. De plus, un environnement interactif est conçu. En particulier, l'indice de capacité et l'indicateur de similarité sont intégrés dans le processus d'optimisation interactif.

Les problèmes d'agencement sont par nature des tâches multidisciplinaires et peuvent être résolus comme des problèmes d'optimisation. Pour traduire les exigences du concepteur en variables d'optimisation, contraintes de conception et objectifs, tout d'abord, l'implantation est formulée par le nouveau composant de pièces solides et virtuelles. Le modèle, prenant en compte les composants virtuels et l'accessibilité aux composants, nécessite un algorithme d'optimisation efficace pour aborder la difficulté réalisable. Sans perte de généralité, deux objectifs, à savoir l'équilibre de l'aménagement et les activités, sont considérés. Ensuite, un nouvel indice de capacité est proposé pour évaluer la difficulté de faisabilité du problème d'agencement. Cet indice définit une alternative au calcul traditionnel de la compacité d'un problème d'agencement. Il permet de démontrer, de manière fiable, si le problème peut être résolu ou non. De plus, l'environnement interactif permet d'intégrer l'optimisation mathématique à la prise de décision humaine lors de la conception conceptuelle du problème d'agencement, en mettant l'accent sur l'analyse de similarité appliquée aux alternatives d'agencement obtenues et aide l'expert à sélectionner la décision finale.

Le modèle d'agencement contient plusieurs objectifs et peut être résolu par un optimiseur multiobjectif qui considère plusieurs objectifs simultanément et recherche un

ensemble de solutions compromises. Le recuit simulé est une approche de recherche de voisinage stochastique pour l'optimisation globale. La facilité de mise en œuvre fait du recuit simulé une méthode populaire pour résoudre des problèmes importants et pratiques. Cependant, la fonctionnalité basée sur l'individu limite l'application, en particulier dans les problèmes multiobjectif. Pour surmonter le problème, SA sans archive utilise la sélection dynamique basée sur le concept de tri non dominé et le mécanisme de calcul de la distance d'encombrement pour l'optimisation multiobjectif est proposé. Des problèmes continus et combinatoires sont testés, dans l'ensemble, les résultats ont prouvé que SA sans archive peut fournir, la plupart du temps, des résultats très compétitifs par rapport aux autres, ce qui la rend intéressante pour résoudre des problèmes d'agencement complexes. À titre de comparaison, nous avons appliqué l'approche d'optimisation de recuit simulé sans archive proposée aux exemples d'agencement simples formulés par le modèle d'agencement à objectifs multiples. Les résultats expérimentaux ont trouvé le moins d'alternatives d'agencement et ont montré des difficultés réalisables dans l'optimisation d'agencement. L'optimiseur peut rester bloqué et ne pas sortir d'une zone de recherche (in) faisable. Alternativement, le problème d'agencement peut être formulé comme un problème combinatoire. Le recuit simulé a démontré la robustesse de l'optimisation combinatoire. Ainsi, une méthode d'optimisation multiobjectif intégrant le placement séquentiel avec un recuit simulé est développée à la place.

L'accessibilité est une exigence importante dans l'industrielle. L'intégration d'espaces virtuels est nécessaire mais pas suffisante pour l'accessibilité des composants. Par conséquent, une analyse d'accessibilité supplémentaire est effectuée par le placement constructif. Différentes stratégies ont été proposées et étudiées afin d'être génériques et efficaces, donc adaptées à un large éventail de problèmes d'agencement. Le placement, non seulement intègre la génération d'espace de composants solides et virtuels garantissant le non-chevauchement des composants, mais introduit également l'idée de chemin de connexion assurant l'accessibilité des composants. Les comparaisons de stratégies confirment que la stratégie de remplissage d'espace peut générer efficacement des solutions réalisables et réduire les efforts de calcul. La solution d'agencement est construite séquentiellement, ce qui est un processus de combinaison. La technique de recherche de recuit simulé explore des combinaisons de configurations de composants et optimise les deux objectifs simultanément. SA sans archive se charge d'explorer efficacement l'espace de recherche pour proposer des alternatives prometteuses. La plupart des recherches sur les problèmes d'agencement se concentrent sur le placement des composants sur un seul con-

teneur. Néanmoins, la disposition avec plusieurs conteneurs a toujours été un point chaud et difficile. La méthode a ensuite été étendue à l'agencement avec plusieurs conteneurs. La complexité de l'optimisation peut être fortement réduite grâce à l'estimation de limite proposée qui transforme la recherche globale en recherche locale.

La méthode d'optimisation d'agencement a été testée sur le problème de l'aménagement d'un abri en mono- et multi-conteneur avec un espace très limité. L'expérimentation prouve que l'optimisation proposée est efficace pour assurer l'accessibilité et trouver des solutions hautement qualifiées par rapport aux algorithmes existants, ce qui permet un processus d'optimisation véritablement interactif. De plus, l'analyse de similarité démontre une bonne diversité de l'ensemble d'agencement obtenu, qui peut être appliqué comme un outil interactif.

Perspective

Cette thèse a contribué à l'approche d'optimisation multiobjectif de l'agencement. Même si les résultats de la recherche ont été positifs et ont permis une optimisation efficace et interactive, il reste encore quelques aspects de la recherche à explorer. Ces directions de recherche sont brièvement décrites ici:

— Introduire les interactions entre le concepteur et l'algorithme d'optimisation. En plus d'utiliser l'analyse de similarité pour aider le concepteur à comparer et sélectionner la solution préférable, l'intelligence artificielle peut également être intégrée pour aider la conception à améliorer encore la solution. Par exemple, analyser l'évaluation des objectifs selon l'importance objective exprimée.

— Définir différentes formes de composants. Nous supposons que tous les composants sont des rectangles dans le modèle de disposition. La définition simplifie grandement la formulation des contraintes et des objectifs. Pour aller plus loin, l'agencement pourrait avoir d'autres formes telles que des cercles, des triangles et des polygones, etc.

— À l'heure actuelle, l'approche a été appliquée à la génération spatiale de formes rectangulaires. D'autres recherches pourraient adapter la génération d'espace disponible à la composante de forme libre. La génération d'espace est utilisée pour éviter le chevauchement des composants. On peut compter sur la conception assistée par ordinateur pour détecter les collisions et signaler la zone de chevauchement ; ou utilisez des ensembles de cercles pour remplacer la forme de forme libre, puis cal-

culez le chevauchement entre les composants avec de simples calculs de distance.

— Les approches proposées ont été appliquées à des problèmes d'agencement bidimensionnel, mais elles peuvent être appliquées à d'autres problèmes, tels que le problème d'agencement tridimensionnel où les composants ont des hauteurs différentes. Adapter la stratégie de placement pour tenir compte des effets de la gravité et de la surface de contact est nécessaire.

BIBLIOGRAPHY

[1] Ana Cuco et al., « Multi-objective design optimization of a new space radiator », *in*: *Optimization and Engineering* 12 (Sept. 2011), pp. 393–406, DOI: https://doi.org/10.1007/s11081-011-9142-6.

[2] Xuelian Gao et al., « Layout Optimization Design of Power IoT Chips », *in*: *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2019, pp. 1620–1624, DOI: https://doi.org/10.1109/IAEAC47372.2019.8998045.

[3] Giovani Fossati, Letícia Miguel, and Walter Paucar Casas, « Multi-objective optimization of the suspension system parameters of a full vehicle model », *in*: *Optimization and Engineering* 20 (Mar. 2019), DOI: https://doi.org/10.1007/s11081-018-9403-8.

[4] Machi Zawidzki and Jacek Szklarski, « Multi-objective optimization of the floor plan of a single story family house considering position and orientation », *in*: *Advances in Engineering Software* 141 (2020), p. 102766, ISSN: 0965-9978, DOI: https://doi.org/10.1016/j.advengsoft.2019.102766.

[5] Sadegh Niroomand et al., « Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems », *in*: *Expert Systems with Applications* 42.*19* (2015), pp. 6586–6597, ISSN: 0957-4174, DOI: https://doi.org/10.1016/j.eswa.2015.04.040.

[6] Chao Ou-Yang and Amalia Utamima, « Hybrid Estimation of Distribution Algorithm for solving Single Row Facility Layout Problem », *in*: *Computers and Industrial Engineering* 66 (Sept. 2013), pp. 95–103, DOI: https://doi.org/10.1016/j.cie.2013.05.018.

[7] Ashish Saraswat, Uday Venkatadri, and Ignacio Castillo, « A framework for multi-objective facility layout design », *in*: *Computers & Industrial Engineering* 90 (2015), pp. 167–176, ISSN: 0360-8352, DOI: https://doi.org/10.1016/j.cie.2015.09.006.

[8] J. Cagan, K. Shimada, and S. Yin, « A survey of computational approaches to three-dimensional layout problems », *in*: *Computer-Aided Design* 34.*8* (2002), pp. 597–611, ISSN: 0010-4485, DOI: https://doi.org/10.1016/S0010-4485(01)00109-9.

[9] Ana Cuco, Fabiano Sousa, and Antônio Silva Neto, « A multi-objective methodology for spacecraft equipment layouts », *in*: *Optimization and Engineering* 16 (Mar. 2014), DOI: https://doi.org/10.1007/s11081-014-9252-z.

[10] Farhad Ghassemi Tari and Hossein Neghabi, « A new linear adjacency approach for facility layout problem with unequal area departments », *in*: *Journal of Manufacturing Systems* 37 (2015), pp. 93–103, ISSN: 0278-6125, DOI: https://doi.org/10.1016/j.jmsy.2015.09.003.

[11] Zhang Lin and Zhang Yingjie, « Solving the Facility Layout Problem with Genetic Algorithm », *in*: *2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*, 2019, pp. 164–168, DOI: 10.1109/IEA.2019.8715148.

[12] S. Szykman and J. Cagan, « Constrained Three-Dimensional Component Layout Using Simulated Annealing », *in*: *Journal of Mechanical Design* 119.*1* (Mar. 1997), pp. 28–35, ISSN: 1050-0472, DOI: https://doi.org/10.1115/1.2828785.

[13] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi, « Optimization by simulated annealing », *in*: *science* 220.*4598* (1983), pp. 671–680.

[14] John Henry Holland et al., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, 1992.

[15] R. Eberhart and J. Kennedy, « A new optimizer using particle swarm theory », *in*: *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43, DOI: 10.1109/MHS.1995.494215.

[16] Ilias Vasilopoulos et al., « Gradient-based Pareto front approximation applied to turbomachinery shape optimization », *in*: *Engineering With Computers* (Jan. 2021), DOI: 10.1007/s00366-019-00832-y.

[17] T. Murata and H. Ishibuchi, « MOGA: multi-objective genetic algorithms », *in*: *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, vol. 1, 1995, pp. 289–, DOI: 10.1109/ICEC.1995.489161.

[18] A. SUPPAPITNARM et al., « A SIMULATED ANNEALING ALGORITHM FOR MULTIOBJECTIVE OPTIMIZATION », *in*: *Engineering Optimization* 33.*1* (2000), pp. 59–85, DOI: 10.1080/03052150008940911.

[19] Xin-She Yang, Mehmet Karamanoglu, and Xingshi He, « Flower pollination algorithm: A novel approach for multiobjective optimization », *in*: *Engineering Optimization* 46.*9* (2014), pp. 1222–1237.

[20] Kalyanmoy Deb et al., « Evolutionary algorithms for multi-criterion optimization in engineering design », *in*: *Evolutionary algorithms in engineering and computer science* 2 (1999), pp. 135–161.

[21] « On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization », *in*: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-1.*3* (1971), pp. 296–297, DOI: 10.1109/TSMC.1971.4308298.

[22] Oscar Brito Augusto, Fouad Bennis, and Stephane Caro, « A new method for decision making in multi-objective optimization problems », *in*: *Pesquisa Operacional* 32 (Aug. 2012), pp. 331–369.

[23] E. Zitzler and L. Thiele, « Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach », *in*: *IEEE Transactions on Evolutionary Computation* 3.*4* (1999), pp. 257–271, DOI: 10.1109/4235.797969.

[24] Coello Coello C.A., Pulido G.T., and Lechuga M.S., « Handling multiple objectives with particle swarm optimization », *in*: *IEEE Transactions on Evolutionary Computation* 8.*3* (2004), pp. 256–279, DOI: 10.1109/TEVC.2004.826067.

[25] Seyedali Mirjalili et al., « Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems », *in*: *Advances in Engineering Software* 114 (2017), pp. 163–191, ISSN: 0965-9978, DOI: https://doi.org/10.1016/j.advengsoft.2017.07.002, URL: https://www.sciencedirect.com/science/article/pii/S0965997816307736.

[26] Dongkyung Nam and Cheol Hoon Park, « Multiobjective Simulated Annealing: A Comparative Study to Evolutionary Algorithms », *in*: *International Journal of Fuzzy Systems* 2 (Jan. 2000).

[27] Kevin I Smith, « A study of simulated annealing techniques for multi-objective optimisation », *in*: (2006).

[28] Kalyanmoy Deb et al., « A fast and elitist multiobjective genetic algorithm: NSGA-II », *in*: *IEEE transactions on evolutionary computation* 6.*2* (2002), pp. 182–197.

[29] Min-Yuan Cheng and Doddy Prayogo, « Symbiotic Organisms Search: A new metaheuristic optimization algorithm », *in*: *Computers & Structures* 139 (2014), pp. 98–112, ISSN: 0045-7949, DOI: https://doi.org/10.1016/j.compstruc.2014.03.007, URL: https://www.sciencedirect.com/science/article/pii/S0045794914000881.

[30] Arnapurna Panda and Sabyasachi Pani, « A Symbiotic Organisms Search algorithm with adaptive penalty function to solve multi-objective constrained optimization problems », *in*: *Applied Soft Computing* 46 (2016), pp. 344–360, ISSN: 1568-4946, DOI: https://doi.org/10.1016/j.asoc.2016.04.030, URL: https://www.sciencedirect.com/science/article/pii/S1568494616301788.

[31] Ke Li et al., « Two-Archive Evolutionary Algorithm for Constrained Multiobjective Optimization », *in*: *IEEE Transactions on Evolutionary Computation* 23.*2* (2019), pp. 303–315, DOI: 10.1109/TEVC.2018.2855411.

[32] J. David Schaffer, « Multiple Objective Optimization with Vector Evaluated Genetic Algorithms », *in*: *Proceedings of the 1st International Conference on Genetic Algorithms*, USA: L. Erlbaum Associates Inc., 1985, pp. 93–100, ISBN: 0805804269.

[33] Qingfu Zhang and Hui Li, « MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition », *in*: *IEEE Transactions on Evolutionary Computation* 11.*6* (2007), pp. 712–731, DOI: 10.1109/TEVC.2007.892759.

[34] Kalyanmoy Deb and Himanshu Jain, « An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints », *in*: *IEEE Transactions on Evolutionary Computation* 18.*4* (2014), pp. 577–601, DOI: 10.1109/TEVC.2013.2281535.

[35] Chunteng Bao, Lihong Xu, and Erik D. Goodman, « A new dominance-relation metric balancing convergence and diversity in multi- and many-objective optimization », *in*: *Expert Systems with Applications* 134 (2019), pp. 14–27, ISSN: 0957-4174, DOI: https://doi.org/10.1016/j.eswa.2019.05.032, URL: https://www.sciencedirect.com/science/article/pii/S0957417419303616.

[36] Ke Li et al., « An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition », *in*: *IEEE Transactions on Evolutionary Computation* 19.*5* (2015), pp. 694–716, DOI: 10.1109/TEVC.2014.2373386.

[37] C.M. Fonseca, L. Paquete, and M. Lopez-Ibanez, « An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator », *in*: *2006 IEEE International Conference on Evolutionary Computation*, 2006, pp. 1157–1163, DOI: 10.1109/CEC.2006.1688440.

[38] Hao Wang et al., « On Steering Dominated Points in Hypervolume Indicator Gradient Ascent for Bi-Objective Optimization », *in*: *NEO 2015: Results of the Numerical and Evolutionary Optimization Workshop NEO 2015 held at September 23-25 2015 in Tijuana, Mexico*, ed. by Oliver Schütze et al., Cham: Springer International Publishing, 2017, pp. 175–203, DOI: 10.1007/978-3-319-44003-3_8.

[39] Eckart Zitzler and Simon Künzli, « Indicator-Based Selection in Multiobjective Search », *in*: Sept. 2004, pp. 832–842, ISBN: 978-3-540-23092-2, DOI: 10.1007/978-3-540-30217-9_84.

[40] Nicola Hochstrate, Boris Naujoks, and Michael Emmerich, « SMS-EMOA: Multiobjective selection based on dominated hypervolume », *in*: *European Journal of Operational Research* 181 (Feb. 2007), pp. 1653–1669, DOI: 10.1016/j.ejor.2006.08.008.

[41] Gladston Moreira and Luís Paquete, « Guiding under uniformity measure in the decision space », *in*: *2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 2019, pp. 1–6, DOI: 10.1109/LA-CCI47412.2019.9037034.

[42] Adriana Menchaca-Méndez and Carlos A. Coello Coello, « GD-MOEA: A New Multi-Objective Evolutionary Algorithm Based on the Generational Distance Indicator », *in*: *EMO*, 2015.

[43] Yiping Liu, Gary G. Yen, and Dunwei Gong, « A Multimodal Multiobjective Evolutionary Algorithm Using Two-Archive and Recombination Strategies », *in*: *IEEE Transactions on Evolutionary Computation* 23.*4* (2019), pp. 660–674, DOI: 10.1109/TEVC.2018.2879406.

[44] Petri Eskelinen et al., « Pareto navigator for interactive nonlinear multiobjective optimization », *in*: *OR Spectrum* 32 (Oct. 2010), pp. 211–227, DOI: 10.1007/s00291-008-0151-6.

[45] Julien Bénabès et al., « Interactive optimization strategies for layout problems », *in*: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 4.*3* (2010), pp. 181–190.

[46] Yi-Nan Guo et al., « Novel Interactive Preference-Based Multiobjective Evolutionary Optimization for Bolt Supporting Networks », *in*: *IEEE Transactions on Evolutionary Computation* 24.*4* (2020), pp. 750–764, DOI: 10.1109/TEVC.2019.2951217.

[47] Laura García-Hernández et al., « A novel hybrid evolutionary approach for capturing decision maker knowledge into the unequal area facility layout problem », *in*: *Expert Systems with Applications* 42.*10* (2015), pp. 4697–4708.

[48] Jeremy Michalek and Panos Papalambros, « Interactive design optimization of architectural layouts », *in*: *Engineering Optimization* 34 (Sept. 2002), pp. 485–501.

[49] Fieldsend J.E., Everson R.M., and Singh S., « Using unconstrained elite archives for multiobjective optimization », *in*: *IEEE Transactions on Evolutionary Computation* 7.*3* (2003), pp. 305–323, DOI: 10.1109/TEVC.2003.810733.

[50] Zhenan He, Gary G. Yen, and Jun Zhang, « Fuzzy-Based Pareto Optimality for Many-Objective Evolutionary Algorithms », *in*: *IEEE Transactions on Evolutionary Computation* 18.*2* (2014), pp. 269–285, DOI: 10.1109/TEVC.2013.2258025.

[51] Hiroyuki Sato, Hernán E. Aguirre, and Kiyoshi Tanaka, « Controlling Dominance Area of Solutions and Its Impact on the Performance of MOEAs », *in*: *Evolutionary Multi-Criterion Optimization*, ed. by Shigeru Obayashi et al., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 5–20.

[52] Hiroyuki Sato, Hernán E. Aguirre, and Kiyoshi Tanaka, « Self-Controlling Dominance Area of Solutions in Evolutionary Many-Objective Optimization », *in*: *Simulated Evolution and Learning*, ed. by Kalyanmoy Deb et al., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 455–465.

[53] Marco Laumanns et al., « Combining Convergence and Diversity in Evolutionary Multiobjective Optimization », *in*: *Evol. Comput.* 10.*3* (Sept. 2002), pp. 263–282, ISSN: 1063-6560, DOI: 10.1162/106365602760234108, URL: https://doi.org/10.1162/106365602760234108.

[54] Yuan Yuan et al., « A New Dominance Relation-Based Evolutionary Algorithm for Many-Objective Optimization », *in*: *IEEE Transactions on Evolutionary Computation* 20.*1* (2016), pp. 16–37, DOI: 10.1109/TEVC.2015.2420112.

[55] Ye Tian et al., « A Strengthened Dominance Relation Considering Convergence and Diversity for Evolutionary Many-Objective Optimization », *in*: *IEEE Transactions on Evolutionary Computation* 23.*2* (2019), pp. 331–345, DOI: 10.1109/TEVC.2018.2866854.

[56] Yan-Yan Tan et al., « MOEA/D-SQA: A multi-objective memetic algorithm based on decomposition », *in*: *Engineering Optimization - ENG OPTIMIZ* 44 (Sept. 2012), pp. 1–21, DOI: 10.1080/0305215X.2011.632008.

[57] Qiuzhen Lin et al., « Multimodal Multiobjective Evolutionary Optimization With Dual Clustering in Decision and Objective Spaces », *in*: *IEEE Transactions on Evolutionary Computation* 25.*1* (2021), pp. 130–144, DOI: 10.1109/TEVC.2020.3008822.

[58] David E. Goldberg and Jon Richardson, « Genetic Algorithms with Sharing for Multimodal Function Optimization », *in*: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, Cambridge, Massachusetts, USA: L. Erlbaum Associates Inc., 1987, pp. 41–49, ISBN: 0805801588.

[59] Tjalling C Koopmans and Martin Beckmann, « Assignment problems and the location of economic activities », *in*: *Econometrica: journal of the Econometric Society* (1957), pp. 53–76.

[60] Gintaras Palubeckis, « Fast local search for single row facility layout », *in*: *European Journal of Operational Research* 246.*3* (2015), pp. 800–814.

[61] Xiaoxiao Song et al., « Efficient multi-objective simulated annealing algorithm for interactive layout problems », *in*: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 15 (Oct. 2021), pp. 441–451, DOI: https://doi.org/10.1007/s12008-021-00773-1.

[62] Jing-fa Liu et al., « A new energy landscape paving heuristic for satellite module layouts », *in*: *Frontiers of Information Technology & Electronic Engineering* 17 (Oct. 2016), pp. 1031–1043, DOI: https://doi.org/10.1631/FITEE.1500302.

[63] Leonardo Chwif, Marcos R Pereira Barretto, and Lucas Antonio Moscato, « A solution to the facility layout problem using simulated annealing », *in*: *Computers in industry* 36.*1-2* (1998), pp. 125–132.

[64] Philipp Hungerländer and Miguel F. Anjos, « A semidefinite optimization-based approach for global optimization of multi-row facility layout », *in*: *European Journal of Operational Research* 245.*1* (2015), pp. 46–61, ISSN: 0377-2217, DOI: https://doi.org/10.1016/j.ejor.2015.02.049.

[65] Fumiya Kudo, Tomohiro Yoshikawa, and Takeshi Furuhashi, « A study on analysis of design variables in Pareto solutions for conceptual design optimization problem of hybrid rocket engine », *in*: *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011, pp. 2558–2562, DOI: https://doi.org/10.1109/CEC.2011.5949936.

[66] Lee K-Y, Han S-N, and Myung-Il Roh, « Optimal Compartment Layout Design for a Naval Ship Using an Improved Genetic Algorithm », *in*: *Marine Technology* 39 (July 2002), pp. 159–169, DOI: https://doi.org/10.5957/mt1.2002.39.3.159.

[67] Jeremy Michalek, R. Choudhary, and Panos Papalambros, « Architectural layout design optimization », *in*: *Engineering Optimization* 34 (Sept. 2002), pp. 461–484, DOI: https://doi.org/10.1080/03052150214016.

[68] Hamidreza Jafaryeganeh, Manuel Ventura, and Carlos Guedes Soares, « Parametric modelling for adaptive internal compartment design of container ships: Proceedings of the 3rd International Conference on Maritime Technology and Engineering (MARTECH 2016, Lisbon, Portugal, 4-6 July 2016) », *in*: June 2016, pp. 655–661, ISBN: 978-1-138-03000-8, DOI: 10.1201/b21890-85.

[69] Dinesh Singh and Supriya Deshmukh, « Multi-objective facility layout problems using BBO, NSBBO and NSGA-II metaheuristic algorithms », *in*: *International Journal of Industrial Engineering Computations* 10 (Aug. 2018), DOI: 10.5267/j.ijiec.2018.6.006.

[70] Benoit Montreuil, « A modelling framework for integrating layout design and flow network design », *in*: *Material handling'90*, Springer, 1991, pp. 95–115.

[71] Yann Briheche et al., « Optimization of Radar Search Patterns in Localized Clutter and Terrain Masking under Direction-Specific Scan Update Rates Constraints »,

in: *IET Radar Sonar and Navigation* (2018), DOI: 10.1049/iet-rsn.2017.0244, URL: https://hal.archives-ouvertes.fr/hal-01705380.

[72] Zhao Xiaoning and Yang Weina, « Research on Layout Problem of Multi-layer Logistics Facility Based on Simulated Annealing Algorithm », *in*: *2011 Fourth International Conference on Intelligent Computation Technology and Automation*, vol. 1, 2011, pp. 892–894, DOI: 10.1109/ICICTA.2011.224.

[73] « A nonlinear optimization approach for solving facility layout problems », *in*: *European Journal of Operational Research* 57.*2* (1992), Facility Layout, pp. 174–189, ISSN: 0377-2217, DOI: https://doi.org/10.1016/0377-2217(92)90041-7.

[74] Maghsud Solimanpur and Amir Jafari, « Optimal solution for the two-dimensional facility layout problem using a branch-and-bound algorithm », *in*: *Computers & Industrial Engineering* 55.*3* (2008), pp. 606–619, ISSN: 0360-8352, DOI: https://doi.org/10.1016/j.cie.2008.01.018, URL: https://www.sciencedirect.com/science/article/pii/S0360835208000387.

[75] RD Meller, « The multi-bay manufacturing facility layout problem », *in*: *International Journal of Production Research* 35.*5* (1997), pp. 1229–1237.

[76] E. Shayan and A. Chittilappilly, « Genetic algorithm for facilities layout problems based on slicing tree structure », *in*: *International Journal of Production Research* 42 (Oct. 2004), pp. 4055–4067, DOI: https://doi.org/10.1080/00207540410001716471.

[77] H. Murata et al., « VLSI module placement based on rectangle-packing by the sequence-pair », *in*: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15.*12* (1996), pp. 1518–1524, DOI: 10.1109/43.552084.

[78] Sadan Kulturel-Konak and Abdullah Konak, « Linear Programming Based Genetic Algorithm for the Unequal Area Facility Layout Problem », *in*: *International Journal of Production Research* 51 (July 2013), DOI: 10.1080/00207543.2013.774481.

[79] Yavuz A. Bozer and Chi-Tai Wang, « A graph-pair representation and MIP-model-based heuristic for the unequal-area facility layout problem », *in*: *European Journal of Operational Research* 218.*2* (2012), pp. 382–391, ISSN: 0377-2217, DOI: https://doi.org/10.1016/j.ejor.2011.10.052, URL: https://www.sciencedirect.com/science/article/pii/S0377221711010058.

[80] Amir Sadrzadeh, « A genetic algorithm with the heuristic procedure to solve the multi-line layout problem », *in*: *Computers & Industrial Engineering* 62.*4* (2012), pp. 1055–1064, ISSN: 0360-8352, DOI: https://doi.org/10.1016/j.cie.2011.12.033.

[81] Kazi Shah Nawaz Ripon et al., « Adaptive variable neighborhood search for solving multi-objective facility layout problems with unequal area facilities », *in*: *Swarm and Evolutionary Computation* 8 (2013), pp. 1–12, ISSN: 2210-6502, DOI: https://doi.org/10.1016/j.swevo.2012.07.003.

[82] Ada Che, Yipei Zhang, and Jianguang Feng, « Bi-objective optimization for multi-floor facility layout problem with fixed inner configuration and room adjacency constraints », *in*: *Computers & Industrial Engineering* 105 (2017), pp. 265–276, ISSN: 0360-8352, DOI: https://doi.org/10.1016/j.cie.2016.12.018.

[83] Kyu-Yeul Lee, Myung-Il Roh, and Hyuk-Su Jeong, « An improved genetic algorithm for multi-floor facility layout problems having inner structure walls and passages », *in*: *Computers & Operations Research* 32.*4* (2005), pp. 879–899.

[84] Julien Bénabès, Emilie Poirson, and Fouad Bennis, « Integrated and interactive method for solving layout optimization problems », *in*: *Expert Systems with Applications* 40 (Nov. 2013), pp. 5796–5803, DOI: https://doi.org/10.1016/j.eswa.2013.03.045.

[85] Jaewoo Seo et al., « Pin Accessibility-Driven Cell Layout Redesign and Placement Optimization », *in*: *Proceedings of the 54th Annual Design Automation Conference*, June 2017, pp. 1–6, DOI: https://doi.org/10.1145/3061639.3062302.

[86] Julien Bénabès et al., « Accessibility in Layout Optimization », *in*: *2nd International Conference On Engineering Optimization*, Lisbonne, Portugal, Sept. 2010.

[87] Farouq Halawa, Sreenath Chalil Madathil, and Mohammad T. Khasawneh, « Multi-objective unequal area pod-structured healthcare facility layout problem with daylight requirements », *in*: *Computers & Industrial Engineering* 162 (2021), p. 107722, ISSN: 0360-8352, DOI: https://doi.org/10.1016/j.cie.2021.107722.

[88] Wei Xie and Nikolaos V. Sahinidis, « A branch-and-bound algorithm for the continuous facility layout problem », *in*: *Computers & Chemical Engineering* 32.*4* (2008), pp. 1016–1028, ISSN: 0098-1354, DOI: https://doi.org/10.1016/j.compchemeng.2007.05.003.

[89] Jonathan Hathhorn, Esra Sisikoglu, and Mustafa Y. Sir, « A multi-objective mixed-integer programming model for a multi-floor facility layout », *in*: *International Journal of Production Research* 51.*14* (2013), pp. 4223–4239, DOI: https://doi.org/10.1080/00207543.2012.753486.

[90] Abbas Ahmadi and Mohammad Jokar, « An Efficient Multiple-stage Mathematical Programming Method for Advanced Single and Multi-Floor Facility Layout Problems », *in*: *Applied Mathematical Modelling* 40.*9* (Jan. 2016), pp. 5605–5620, DOI: https://doi.org/10.1016/j.apm.2016.01.014.

[91] M. Mir and M.H. Imam, « A hybrid optimization approach for layout design of unequal-area facilities », *in*: *Computers & Industrial Engineering* 39.*1* (2001), pp. 49–63, ISSN: 0360-8352, DOI: https://doi.org/10.1016/S0360-8352(00)00065-6.

[92] Abdelahad Chraibi et al., « A Particle Swarm Algorithm for Solving the Multi-objective Operating Theater Layout Problem », *in*: *IFAC-PapersOnLine* 49.*12* (June 2016), pp. 1169–1174, DOI: https://doi.org/10.1016/j.ifacol.2016.07.663.

[93] Besbes Mariem et al., « A methodology for solving facility layout problem considering barriers – genetic algorithm coupled with A* search », *in*: *Journal of Intelligent Manufacturing.* 31 (Mar. 2019), pp. 615–640, DOI: https://doi.org/10.1007/s10845-019-01468-x.

[94] M. Méndez et al., « Proposal and Comparative Study of Evolutionary Algorithms for Optimum Design of a Gear System », *in*: *IEEE.Access.* 8 (2020), pp. 3482–3497, DOI: https://doi.org/10.1109/ACCESS.2019.2962906.

[95] B. Medjdoub and B. Yannou, « Separating topology and geometry in space planning », *in*: *Computer-Aided Design* 32.*1* (2000), pp. 39–61, ISSN: 0010-4485, DOI: https://doi.org/10.1016/S0010-4485(99)00084-6.

[96] B Medjdoub and B Yannou, « Dynamic space ordering at a topological level in space planning », *in*: *Artificial Intelligence in Engineering* 15.*1* (2001), pp. 47–60, ISSN: 0954-1810, DOI: https://doi.org/10.1016/S0954-1810(00)00027-3.

[97] Yann Briheche et al., « Branch-and-Bound Method for Just-in-Time Optimization of Radar Search Patterns », *in*: *Nature-Inspired Methods for Metaheuristics Opti-*

mization, Jan. 2020, pp. 465–488, DOI: https://doi.org/10.1007/978-3-030-26458-1_25.

[98] Surya Prakash Singh, « Solving facility layout problem: three-level tabu search metaheuristic approach », *in*: *International Journal of Recent Trends in Engineering* 1.*1* (2009), p. 73.

[99] N. Bozorgi, M. Abedzadeh, and M. Zeinali, « Tabu search heuristic for efficiency of dynamic facility layout problem », *in*: *The International Journal of Advanced Manufacturing Technology* 77 (Mar. 2014), pp. 689–703.

[100] Pyari Mohan Pradhan et al., « Energy Efficient Layout for a Wireless Sensor Network using Multi-Objective Particle Swarm Optimization », *in*: *2009 IEEE International Advance Computing Conference*, 2009, pp. 65–70, DOI: 10.1109/IADCC.2009.4808982.

[101] Sabri Pllana, Suejb Memeti, and Joanna Kolodziej, « Customizing Pareto Simulated Annealing for Multi-Objective Optimization of Control Cabinet Layout », *in*: *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, 2019, pp. 78–85, DOI: 10.1109/CSCS.2019.00021.

[102] Ranjan Hasda, Rajib Bhattacharjya, and Fouad Bennis, « Modified genetic algorithms for solving facility layout problems », *in*: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 11 (2016), pp. 713–725, DOI: https://doi.org/10.1007/s12008-016-0362-z.

[103] Mostafa Mazinani, Mostafa Abedzadeh, and Navid Mohebali, « Dynamic facility layout problem based on flexible bay structure and solving by genetic algorithm », *in*: *The International Journal of Advanced Manufacturing Technology* 65 (Mar. 2012), pp. 929–943, DOI: https://doi.org/10.1007/s00170-012-4229-6.

[104] Sumin Kang and Junjae Chae, « Harmony search for the layout design of an unequal area facility », *in*: *Expert Systems with Applications* 79 (2017), pp. 269–281, ISSN: 0957-4174, DOI: https://doi.org/10.1016/j.eswa.2017.02.047.

[105] Hasan Hosseini nasab et al., « Classification of facility layout problems: a review study », *in*: *The International Journal of Advanced Manufacturing Technology volume* 94 (2018), pp. 957–977, DOI: https://doi.org/10.1007/s00170-017-0895-8.

[106] Robin S. Liggett and William J. Mitchell, « Optimal space planning in practice », *in*: *Computer-Aided Design* 13.*5* (1981), Special Issue Design optimization, pp. 277–288, ISSN: 0010-4485, DOI: https://doi.org/10.1016/0010-4485(81)90317-1, URL: https://www.sciencedirect.com/science/article/pii/0010448581903171.

[107] Farouq Halawa, Sreenath Chalil Madathil, and Mohammad T. Khasawneh, « Integrated framework of process mining and simulation–optimization for pod structured clinical layout design », *in*: *Expert Systems with Applications* 185 (2021), p. 115696, ISSN: 0957-4174, DOI: https://doi.org/10.1016/j.eswa.2021.115696.

[108] Xueping Li, Zhaoxia Zhao, and Kaike Zhang, « A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins », *in*: *Industrial and Systems Engineering Research Conference*, May 2014, pp. 2039–2048.

[109] José Fernando Gonçalves and Mauricio GC Resende, « A biased random-key genetic algorithm for the unequal area facility layout problem », *in*: *European Journal of Operational Research* 246 (2015), pp. 86–107, DOI: https://doi.org/10.1016/j.ejor.2015.04.029.

[110] Jiazhen Huo, Jing Liu, and Hong Gao, « An NSGA-II Algorithm with Adaptive Local Search for a New Double-Row Model Solution to a Multi-Floor Hospital Facility Layout Problem », *in*: *Applied Sciences* 11.*4* (2021), p. 1758, ISSN: 2076-3417, DOI: https://doi.org/10.3390/app11041758.

[111] Hüsamettin Bayram and Ramazan Şahin, « A new simulated annealing approach for the traveling salesman problem », *in*: *Mathematical and Computational Applications* 18 (Dec. 2013), pp. 313–322, DOI: https://doi.org/10.3390/mca18030313.

[112] Daniel Delahaye, Supatcha Chaimatanan, and Marcel Mongeau, « Simulated annealing: From basics to applications », *in*: *International Series in Operations Research & Management Science (ISOR)* 272 (2019), pp. 1–35, DOI: https://doi.org/10.1007/978-3-319-91086-4_1.

[113] Rosmalina Hanafi and Erhan Kozan, « A Hybrid Constructive Heuristic and Simulated Annealing for Railway Crew Scheduling », *in*: *Computers & Industrial Engineering* 70 (Apr. 2014), pp. 11–19, DOI: https://doi.org/10.1016/j.cie.2014.01.002.

[114] Maral Zafar Allahyari and Ahmed Azab, « Mathematical modeling and multi-start search simulated annealing for unequal-area facility layout problem », *in*: *Expert Systems with Applications* 91 (2018), pp. 46–62, ISSN: 0957-4174, DOI: https://doi.org/10.1016/j.eswa.2017.07.049.

[115] Yujie Xiao, Yoonho Seo, and Minseok Seo, « A two-step heuristic algorithm for layout design of unequal-sized facilities with input/output points », *in*: *International Journal of Production Research* 51.*14* (2013), pp. 4200–4222, DOI: https://doi.org/10.1080/00207543.2012.752589.

[116] Abbas Ahmadi, Mir Saman Pishvaee, and Mohammad Reza Akbari Jokar, « A survey on multi-floor facility layout problems », *in*: *Computers & Industrial Engineering* 107 (2017), pp. 158–170, ISSN: 0360-8352, DOI: https://doi.org/10.1016/j.cie.2017.03.015.

[117] S Bernardi and M F Anjos, « A two-stage mathematical-programming method for the multi-floor facility layout problem », *in*: *Journal of the Operational Research Society* 64.*3* (2013), pp. 352–364, DOI: 10.1057/jors.2012.49.

[118] Kyu-Yeul Lee, Myung-Il Roh, and Hyuk-Su Jeong, « An improved genetic algorithm for multi-floor facility layout problems having inner structure walls and passages », *in*: *Computers & Operations Research* 32.*4* (2005), pp. 879–899, ISSN: 0305-0548, DOI: https://doi.org/10.1016/j.cor.2003.09.004.

[119] Dimitrios I Patsiatzis and Lazaros G Papageorgiou, « Optimal multi-floor process plant layout », *in*: *Computers & Chemical Engineering* 26.*4* (2002), pp. 575–583, ISSN: 0098-1354, DOI: https://doi.org/10.1016/S0098-1354(01)00781-5.

[120] J. Blank and K. Deb, « pymoo: Multi-Objective Optimization in Python », *in*: *IEEE Access* 8 (2020), pp. 89497–89509.

[121] Walid Ben-Ameur, « Computing the Initial Temperature of Simulated Annealing », *in*: *Computational Optimization and Applications* 29 (Dec. 2004), pp. 369–385, DOI: 10.1023/B:COAP.0000044187.23143.bd.

[122] Julien Bénabès et al., « 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, 2012, pp. 727–734, DOI: http://doi.org/10.1115/DETC2012-70640.

[123] KK Lai and Jimmy WM Chan, « Developing a simulated annealing algorithm for the cutting stock problem », *in*: *Computers & Industrial Engineering* 32.*1* (1997), pp. 115–127, ISSN: 0360-8352, DOI: https://doi.org/10.1016/S0360-8352(96)00205-7.

[124] Xiaoxiao Song et al., « Interactive Design Optimization of Layout Problems », *in*: *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems*, Aug. 2021, pp. 387–395, DOI: https://doi.org/10.1007/978-3-030-85914-5_41.

[125] Julien Bénabès et al., « 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, Aug. 2011, pp. 553–562, DOI: http://doi.org/10.1115/DETC2011-47925.

[126] Daniel Müllner, *Modern hierarchical, agglomerative clustering algorithms*, 2011, arXiv: 1109.2378 [stat.ML].

[127] Xiaoxiao Song et al., « Multi-objective optimization of layout with functional constraints », *in*: *Optimization and Engineering* (July 2022), pp. 1–34, DOI: https://doi.org/10.1007/s11081-022-09754-z.

[128] Julien Bénabès, Emilie Poirson, and Fouad Bennis, « Integrated and interactive method for solving layout optimization problems », *in*: *Expert Systems with Applications* 40 (2013), pp. 5796–5803, ISSN: 0957-4174, DOI: https://doi.org/10.1016/j.eswa.2013.03.045.

[129] Xiaoxiao Song et al., « Multi-objective layout optimization of industrial environment », *in*: *2022 8th International Conference on Optimization and Applications (ICOA)*, 2022, pp. 1–6, DOI: 10.1109/ICOA55659.2022.9934713.

Titre : Optimisation interactive et multiobjectif d'agencement d'espace

Mot clés : Optimisation multiobjectif, Recuit simulé, Agencement d'espace, Indice de capacité, Placement constructif, Intégration de l'accessibilité

Résumé : Dans tous les problèmes d'agencement, les contraintes de non-chevauchement entre composants et les contraintes d'appartenance au conteneur sont présentes. Un modèle d'agencement multiobjectif avec contraintes fonctionnelles est développé. Intégrer l'accessibilité des composants comme contraintes fonctionnelles assure la maintenance ou le fonctionnement des composants. Cependant, les contraintes fonctionnelles augmentent la complexité d'optimisation d'agencement, notée indice de capacité. Par conséquent, un nouvel algorithme d'optimisation multiobjectif est proposé en utilisant le placement constructif et le recuit simulé pour rechercher des solutions de compromis entre les objectifs multiples. Ensuite, un indicateur de similarité est défini pour évaluer les similaires entre les solutions proposées par l'algorithme. Les expériences indiquent que l'approche d'optimistion proposée fonctionne bien pour garantir l'accessibilité et trouver efficacement des solutions optimales dans les problèmes industriels d'agencement d'espace à un ou plusieurs conteneurs, où l'analyse de similarité démontre une bonne diversité solutions proposées par l'algorithme, qui peut être appliqué en tant qu'outil interactif outil pour le concepteur.

Title: Layout optimization based on multi-objective interactive approach

Keywords: Multi-objective optimization, Simulated annealing, Layout problem, Capacity index, Constructive placement, Accessibility integration

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. A multi-objective layout model with functional constraints is developed. Integrating the accessibility of components as functional constraints ensures components maintenance or proper operation. However, the functional constraints increase the layout optimization complexity, denoted as capacity index. Therefore, a novel multi-objective optimization algorithm is proposed using the constructive placement and the simulated annealing to search for compromised solutions between the multiple 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 in single- and multi- container layout applications, where the similarity analysis demonstrates good diversity of the obtained layout set, which can be applied as an interactive tool.