2D multi-objective placement algorithm for free-form components

Guillaume Jacquenot, Fouad Bennis, Jean-Jacques Maisonneuve, Philippe Wenger

Proceedings of ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2009
August 30 - September 2, 2009, San Diego, USA

DETC2009-87603

2D MULTI-OBJECTIVE PLACEMENT ALGORITHM FOR FREE-FORM COMPONENTS

Guillaume Jacquenot[*(1,2)], **Fouad Bennis**[(1)], **Jean-Jacques Maisonneuve**[(2)], **Philippe Wenger**[(1)]
[(1)] École Centrale de Nantes, IRCCyN, UMR CNRS 6597, Nantes, France
[(2)] SIREHNA, Nantes, France
Email: guillaume.jacquenot@gmail.com

ABSTRACT

This article presents a generic method to solve 2D multi-objective placement problem for free-form components. The proposed method is a relaxed placement technique combined with an hybrid algorithm based on a genetic algorithm and a separation algorithm. The genetic algorithm is used as a global optimizer and is in charge of efficiently exploring the search space. The separation algorithm is used to legalize solutions proposed by the global optimizer, so that placement constraints are satisfied. A test case illustrates the application of the proposed method. Extensions for solving the 3D problem are given at the end of the article.

1 Nomenclature

m	Number of components
$\boldsymbol{v}$	Vector of positioning variables
$\boldsymbol{v}_i$	Vector of variables containing position variables of component i. $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$
O_i	Component i
n_i	Number of circles of component i
A_i	Area of component i
ρ_i	Density of component i
S_{ij}	j^{th} circle of the i^{th} component
$\boldsymbol{c}_{ij}$	Vector of coordinates of the j^{th} circle of the i^{th} component
f_{ijkl}^{pen}	Penetration penalty between the j^{th} circle of the i^{th} component and the l^{th} circle of the k^{th} component
f_{ij}^{pro}	Protrusion penalty of the j^{th} circle of the i^{th} component
C	Container / Enclosure
$\text{cl}(S)$	Closure of S

2 Introduction

The problem of placing a set of components inside an enclosure is known as the placement problem. A placement problem asks to find all placement variables of all components so that objectives are minimized and constraints are satisfied. Solving a placement problem consists in finding one or several solution that minimizes the objectives and respects a set of constraints. Among all constraints, each placement problem presents non-overlap and non-protrusion constraints. These constraints respectively express the fact that components should not collide with each other and that each component must lay inside the boundaries of the container. These problems are non-linear and most of the time $\mathscr{NP}$-complete, meaning that solutions associated with the corresponding decision problem can be checked with a polynomial algorithm.

Placement problems gather Cutting & Packing (C&P) problems and layout problems. In a C&P problem, components are only geometrically related to each other, whereas in a layout problem, components are geometrically and functionnally related to each other [1]. The underlying objective of each C&P problem is a compaction objective: either a maximum number of components has to be placed in the container, or a minimum number of containers has to be used to place all components.

[*]Address all correspondence to this author.

This characteristics, that may also be found in a layout problem, can influence the choice of modelling.

Wäscher *et al.* [2] recently proposed a typology of C&P problems and improved the one proposed by Dyckhoff [3]. Among all C&P problems, knapsack problems [4], bin packing problems [5], nesting problems [6, 7] container loading problems [8] may be the most representative. Layout problems also gather a wide panel of problems, including Facility Layout (FL) problems [9], Very Large Scale Integration (VLSI) problems [10] and engineering problems [11].

Cagan *et al.* [12] proposed a survey on 3D layout problem, in which the different aspects of layout problems are discussed. Grignon *et al.* [13] proposed a multi-objective genetic algorithm framework to solve 3D layout problems, where individuals are clouds of solutions. Aladahalli *et al.* [1, 14] developed a pattern search algorithm to solve 3D layout problems, where objectives and constraints are aggregated in a single merit function. Yi *et al.* [15] used a genetic algorithm (NSGA-II) to find the positions of a set of components of a trunk. CAD is used to model the problem and perform geometric computations. Non-overlap and non-protrusion constraints act as a penality when computing the rank of each solution. Tiwari *et al.* [16] proposed a 3D bottom-left-back strategy to solve a 3D C&P problem. The voxelization of components allows to pack complex geometry components inside the container. A steady-state genetic algorithm is used to generate the packing order and find the correct orientation of each component. Placement constraints are automatically satisfied with the voxel-based representation. Zhang *et al.* [11] used a combination of soft computing techniques (GA/PSO) to place a set of components in a satellite module under behavioral constraints. Components are modeled by either parallelepipeds or cylinders, which allow using analytical functions to evaluate overlap and protrusion constraints. Dong *et al.* [17] proposed a shape-morphing method to design components and solve a layout problem at the same time.

The objective of this article is to introduce a new placement method adapted to multi-objective problems. The first section gives a general presentation of different placement methods, the second section concerns the proposed method. An example is discussed in the third section, finally the different elements needed to develop the 3D placement method are presented.

3 Placement methods

Placement problems have generated a large amount of literature, however all placement techniques proposed can be classified in two categories: legal placement method and relaxed placement method. A legal placement method is a method that ensures that all placement constraints are satisfied while building the solution. This is typically the case when using an encoding scheme (*e.g.* bottom-left heuristic). Tiwari *et al.* [18] proposed a survey of various encoding schemes, which can be used to propose different placements. These placement techniques are perfectly suited for problems presenting compaction objectives, such as the minimization of the wire-length of components inside an electronic module. Relaxed placement methods allow the non-respect of placement constraints during the elaboration of the solution. These methods are mainly used when dealing with complex geometries and can easily model any placement constraints.

When using a legal placement method, the decision variables of the problem are usually permutations, which indicate the order of introduction of the components inside the container. Other more evolved encoding schemes have been proposed such as the sequence pair encoding, in which permutations introduce topological relations between components. The decision variables of relaxed placement techniques are directly the positioning variables of the components.

Placement problems are generally multi-objective problems. However, most of the problems are treated as single-objective unconstrained problems, where objectives and contraints are aggregated in a single merit function. Such a formulation has several shortcomings, first of all only the convex part of the Pareto front can be identified. Second, the set of weights used to aggregate functions can be difficult to identify, and may lead on strong assumptions on the solutions obtained. A multi-objective formulation is therefore preferred: multi-objective algorithms will search for the set of non-dominated points in the objective space given by efficient solutions, as shown on figure 1.

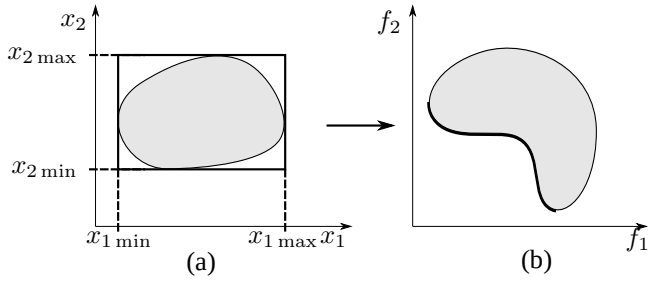


Figure 1. ILLUSTRATION OF THE VARIABLE SEARCH SPACE AND OBJECTIVE SEARCH SPACE FOR A BI-OBJECTIVE MINIMIZATION PROBLEM. THICK DARK LINE ON SUBFIGURE (B) REPRESENTS PARETO FRONT OF THE PROBLEM, *I.E.* THE SET OF NON-DOMINATED POINTS.

4 The proposed method

The objective of the proposed method is to be generic, thus suited to a wide set of placement contraints and objectives. A relaxed placement technique is therefore chosen. The variables of

the problem are directly the positioning variables of the components. These variables can be of different types. Translation variables are modeled as continuous variables and orientation variables can be continuous or discrete depending on the designer's needs.

The difficulty of such a modelling will be to propose feasible solutions, *i.e.* solutions that satisfy placement constraints. The proposed method is based on the use of a separation algorithm, which objective is to legalize unfeasible solutions. A global optimizer is used to generate different promising solutions. The different elements needed for the resolution of the problem are presented in the following subsections.

4.1 Geometry handling

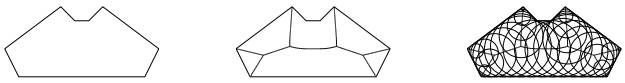
When using a relaxed placement method, collision or overlap detection may represent most of the computational effort. The objective is therefore to use a correct geometric representation, that will be suited to our needs.

Several techniques can be used to detect a collision between two components. The first one consists in computing the overlap area of each pair of polygons. However, this technique is too costly. No-fit polygon (*NFP*) and inner-fit polygon (*IFP*) can also be used [19]. NFP and IFP present the advantage of allowing collision detection with a simple test of a point being in a polygonal region. These features depend on the orientation of the components, meaning that these elements have to be recomputed for each orientation. When components are convex, this can be done in linear time with respect to the number of vertices of polygons. Otherwise, a convex decomposition is needed. However, this approach can hardly be adapted to 3D problems even in the case of polytopes.

In case one is interested in the free rotation of components, components can be transformed into sets of circles. The transformation proposed is based on the use of medial axis, which corresponds to the centers of maximum inscribed circles. Figure 2 illustrates how components are converted to sets of circles. First, the medial axis of the polygon is computed. Second, a set of circles which centers are located on the medial axis are inserted. Circles are inserted such that the distance between two centers is lower than a distance chosen by the designer. We choose this representation for which collision detections between components can be performed with simple distance computations, and for which a separation algorithm has been proposed.

4.2 Separation algorithm

Different separation algorithms have been proposed, [20,21, 22,23] however the key idea is always the same: given a configuration that does not satisfy placement contraints, the objective of the separation algorithm is to minimize the non-respect of overlap and protrusion constraints. These constraints are gathered in



(a) Original shape (b) Medial axis (c) Circle placement

Figure 2. DIFFERENT STEPS OF THE CONVERSION OF A POLYGON TO A SET OF CIRCLES.

a penalty function F characterizing the violation of the placement constraints.

The separation algorithm and the formalism used here are the ones proposed by Imamichi *et al.* [23]. Let's consider a collection of m components $\mathscr{O} = \{O_1, \ldots, O_m\}$. Each component O_i consists of n_i circles $\{S_{i1}, S_{i2} \ldots, S_{in_i}\}$. Let $\boldsymbol{c}_{ij}$ be the vector that represents the center of circle S_{ij}, r_{ij} be the radius of S_{ij} $(i = 1, \ldots, m; j = 1, \ldots, n_i)$. Circles are denoted with the capital letter S, because the separation algorithm can also be applied in 3D with Spheres.

The separation problem is an unconstrained minimization problem defined by

$$(\text{Sep}) \begin{cases} \min \ F(\boldsymbol{v}) = \omega_{\text{pen}} F_{\text{pen}}(\boldsymbol{v}) + \omega_{\text{pro}} F_{\text{pro}}(\boldsymbol{v}) \\ \text{s.t.} \quad \boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m) \end{cases} \tag{1}$$

where vector $\boldsymbol{v}$ represents the placement variables, which represent degrees of freedom (*dof*) of the components. ω_{pen} and ω_{pro} are positive parameters representing the weights of the penetration and protrusion function. These values are respectively set to $1/3$ and $2/3$. The total penetration and protrusion penalty functions can be mathematically written as follows:

$$\begin{aligned} F_{\text{pen}}(\boldsymbol{v}) &= \sum\nolimits_{1 \leqslant i < k \leqslant m} \sum\nolimits_{j=1}^{n_i} \sum\nolimits_{l=1}^{n_k} f_{ijkl}^{\text{pen}}(\boldsymbol{v}) \\ F_{\text{pro}}(\boldsymbol{v}) &= \sum\nolimits_{i=1}^{m} \sum\nolimits_{j=1}^{n_i} f_{ij}^{\text{pro}}(\boldsymbol{v}) \end{aligned} \tag{2}$$

where f_{ijkl}^{pen} and f_{ij}^{pro} represent the local penalties of the penetration and protrusion, defined by the penetration depth. The penetration depth δ [24] of two components O_1 and O_2 is defined as the minimum translation distance of component O_2 so that O_1 and O_2 do not overlap. These penalties write

$$\begin{aligned} f_{ijkl}^{\text{pen}}(\boldsymbol{v}) &= \left(\delta\left(S_{ij}(\boldsymbol{v}_i), S_{kl}(\boldsymbol{v}_k)\right)\right)^2 \\ f_{ij}^{\text{pro}}(\boldsymbol{v}) &= \left(\delta\left(S_{ij}(\boldsymbol{v}_i), \text{cl}\left(\overline{C}\right)\right)\right)^2 \end{aligned} \tag{3}$$

The penetration penalty of two circles S_{ij} and S_{kl} can be finally written as:

$$f_{ijkl}^{\text{pen}}(\boldsymbol{v}) = \left(\max\left\{r_{ij} + r_{kl} - \left\|\boldsymbol{c}_{ij}(\boldsymbol{v}_i) - \boldsymbol{c}_{kl}(\boldsymbol{v}_k)\right\|, 0\right\}\right)^2 \tag{4}$$

Regarding non-protrusion constraints, inward offsets are used to evaluate if circles belong to the container. These constraints can be checked with a simple test of a point being inside a polygon. If the center of a circle is not contained inside the inward offset polygon, the minimum distance between the center and this polygon is used to build the constraint function and its gradient.

The function F is a piecewise continuous function, therefore its gradient can be used to find its minimum. The gradient of function F is

$$\nabla F(\boldsymbol{v}) = \omega_{\text{pen}} \nabla F_{\text{pen}}(\boldsymbol{v}) + \omega_{\text{pro}} \nabla F_{\text{pro}}(\boldsymbol{v}) \tag{5}$$

The gradient $\nabla F(\boldsymbol{v})$ is computed by evaluating all components of gradients ∇F_{pen} and ∇F_{pro} as follows:

$$\begin{aligned} \nabla_i F_{\text{pen}}(\boldsymbol{v}) &= \frac{\partial F_{\text{pen}}(\boldsymbol{v})}{\partial \boldsymbol{v}_i} = \sum_{j=1}^{n_i} \sum_{k=\{1,\dots,m\}\setminus i} \sum_{l=1}^{n_k} \frac{\partial f_{ijkl}^{\text{pen}}(\boldsymbol{v})}{\partial \boldsymbol{v}_i} \\ \nabla_i F_{\text{pro}}(\boldsymbol{v}) &= \frac{\partial F_{\text{pro}}(\boldsymbol{v}_i)}{\partial \boldsymbol{v}_i} = \sum_{j=1}^{n_i} \frac{\partial f_{ij}^{\text{pro}}(\boldsymbol{v}_i)}{\partial \boldsymbol{v}_i} \end{aligned} \tag{6}$$

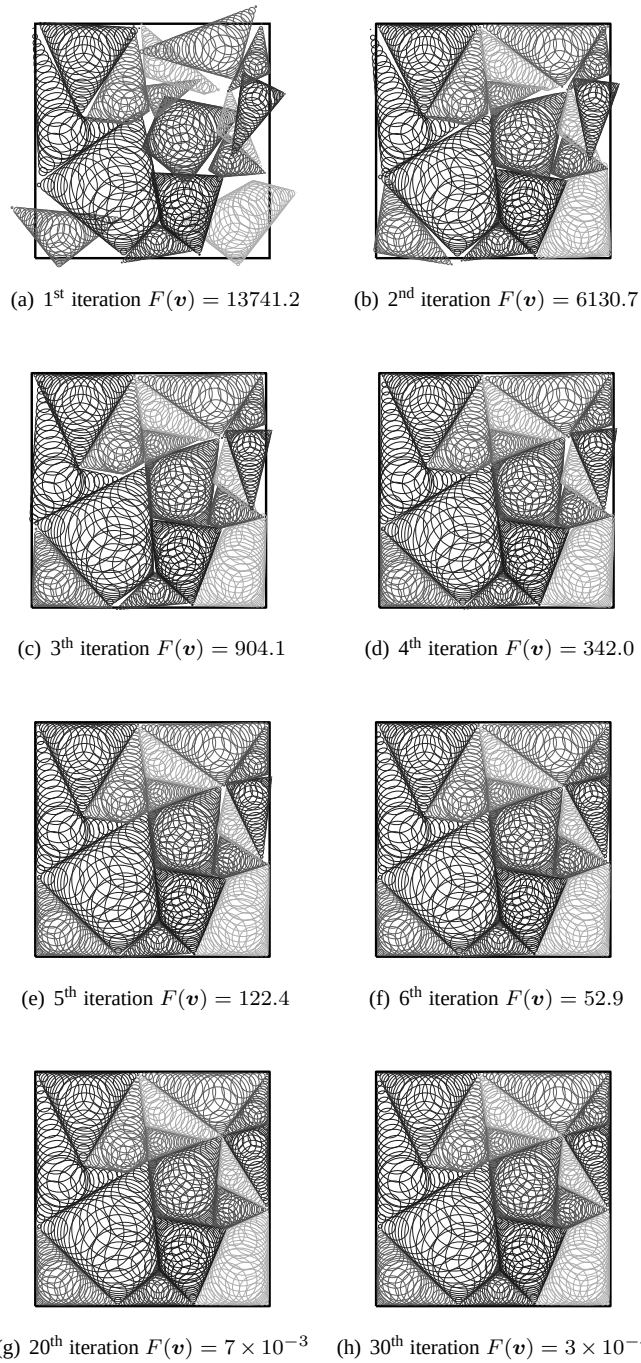
For further details on the expressions of gradients, the reader can refer to Imamichi *et al.* [23]. The separation problem is solved using the BFGS quasi-Newton method [25]. This technique is iterative and moves all components at the same time, unlike different separation heuristics that move one component at a time [6]. Figure 3 illustrates how the separation algorithm works on a solution proposed by the global optimizer. The order of magnitude of the function F depends on the circle conversion. However, when placement constraints are satisfied, the function F should always be zero.

The separation algorithm allows one to find a feasible solution from a solution that does not satisfy placement constraints. The generation of solutions is the role of the global optimizer.

4.3 Global optimizer

The global optimizer is in charge of exploring efficiently the search space to propose promising solutions. The genetic algorithm (GA) Omni-Optimizer [27] is used. Its objective is to improve the current population at each new generation. Based on NSGA-2 [28], this generational GA is designed to handle single and multi-objective problems. This algorithm was originally chosen for its capacities to find multi-modal solutions. Three variation operators are used to generate new solutions:

- The crossover operator takes two solutions and generates two offspring solutions by crossing over all parent variables under a certain probability. The standard crossover scheme *SBX* is used for real variables [29] and the two point crossover is used for discrete variables. The crossover operator is used with extreme parsimony: indeed when the com-



(a) 1st iteration $F(\boldsymbol{v}) = 13741.2$ (b) 2nd iteration $F(\boldsymbol{v}) = 6130.7$

(c) 3th iteration $F(\boldsymbol{v}) = 904.1$ (d) 4th iteration $F(\boldsymbol{v}) = 342.0$

(e) 5th iteration $F(\boldsymbol{v}) = 122.4$ (f) 6th iteration $F(\boldsymbol{v}) = 52.9$

(g) 20th iteration $F(\boldsymbol{v}) = 7 \times 10^{-3}$ (h) 30th iteration $F(\boldsymbol{v}) = 3 \times 10^{-4}$

Figure 3. SEPARATION ALGORITHM AT WORK ON INSTANCE *DIGHE 1* [26]. 16 COMPONENTS MODELLED WITH 515 CIRCLES. EACH COMPONENT HAS ONE ROTATION DOF.

pacity of the problem increases, crossing over two solutions do not produce promising solutions (*i.e.* offspring solutions present too much overlaps when compared with their parent solutions), even when using the restricted selection operator. Introduced by Deb *et al.* [27], the restricted selection operator consists in selecting two parents which genotypic distance is minimum.

- Mutations generate new solutions from one parent. The polynomial mutation operator is used for real variables, and a bit wise mutation is used for discrete variables.
- A swap operator that exchanges the position of two components has been added. This classical operator in placement problems allows one to generate new solutions from one parent solution.

4.4 The proposed algorithm

The proposed algorithm is presented figure 4. The structure of the algorithm is very close to a generational genetic algorithm. The separation algorithm is nested in the genetic algorithm, and modifies component position so that the solution proposed respect placement constraints. The initial population can be generated randomly, or the designer can provide a set of solutions from its choice or also solutions obtained from a placement heuristic such as bottom-left placement technique.

Before evaluating a solution, the algorithm checks if placement constraints are satisfied. If so, the different objectives of the solution are evaluated and the algorithm moves to the next solution. Otherwise, the separation algorithm is run and modifies the solution so that placement constraints are respected. The solution is then evaluated. At the end of the local optimization performed by the separation algorithm, the value of function F returned is used as a constraint violation indicator for the genetic algorithm. This indicator will then be taken into account in the genetic operations. As a consequence, a solution that does not respect placement constraints will not be selected in the constrained binary tournament when compared to a feasible solution.

4.5 Characteristics of the method

This subsection sums up the different characteristics and possibilities of the proposed method:

- Components can be restricted to lay in an area defined by the designer. It consists in reducing the inward offset domains of a component.
- Alignement constraints can be included in the modelling and in the separation algorithm, therefore automatically being satisfied.
- The designer can easily interact with solutions, and can propose solutions even if these solutions do not respect placement constraints.

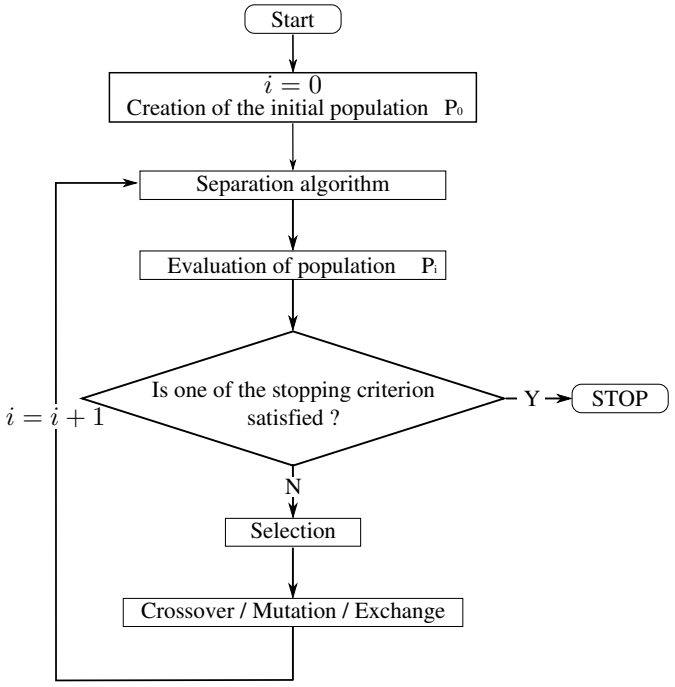


Figure 4. GENETIC ALGORITHM PROPOSED TO SOLVE PLACEMENT PROBLEMS.

- Distance queries between components can easily be satisfied. For example, a minimum distance d between two components i and k can be obtained by virtually growing radius of each circle of components i and k of $d/2$ when considering their possible overlap.
- In the case, components are all rectangular in 2D and parallelepipedic in 3D, the separation algorithm can be adapted to handle these special cases. The objective of the separation algorithm may be to reduce the sum of overlaping areas/volumes, for which an analytical function and gradient can easily be found.
- The separation algorithm may be optional for problems with a low compacity. In this case, the method is equivalent to use a simple genetic algorithm to solve a placement problem, as Yi *et al.* [15] already did.

5 Simulation results

5.1 Data of the problem

Let's consider the problem of loading a car trunk, in which a set of 11 components should be placed (Figures 5a & 5b). Two objectives are taken into account: the first one consists in minimizing the euclidean distance between the geometric center of gravity of the trunk and the center of gravity of the assembly. The second objective consists in minimizing the inertia moment of the assembly around the y axis of the trunk (Figure 5a). The first

Table 1. COMPONENT PROPERTIES

Type of components	T1	T2	T3	T4	T5	T6
Number of components	2	2	1	1	2	3
Density of components	0.7	0.6	0.6	0.8	0.3	0.5

objective gives information on the static behavior of the trunk, whereas the second is more about the dynamic behavior.

Component properties are given in table 1. The grayscale of components will be used to code their density: the darker a component, the larger its density. The global compacity of this problem is evaluated to 79.5%. Orientations of components are chosen to be discrete. For each component, the set of possible orientations is $\{0°, 90°, 180°, 270°\}$, except for symetrical components which orientation is chosen among the set $\{0°, 90°\}$.

The problem has 22 real variables coding component positions and 11 discrete variables coding component orientations. The separation algorithm only translates components to legalize solutions proposed by the global optimizer. The formulation of the problem can be formulated as follows:

$$\mathscr{P} \begin{cases} \min f_1(\boldsymbol{v}) = \sqrt{d_x^2 + d_y^2} \\ \min f_2(\boldsymbol{v}) = \sum_{i=1}^{m} \rho_i I_{i/yy} \\ \text{s.t. placement constraints are satisfied} \end{cases} \tag{7}$$

where

$$\begin{aligned} d_x &= \frac{1}{\sum_{i=1}^{m} \rho_i A_i} \sum_{i=1}^{m} \rho_i A_i (x_i - x_G) \\ d_y &= \frac{1}{\sum_{i=1}^{m} \rho_i A_i} \sum_{i=1}^{m} \rho_i A_i (y_i - y_G) \\ I_{i/yy} &= \iint_{(x,y)\in P_i} (x - x_G)^2 dxdy \end{aligned} \tag{8}$$

with (x_G, y_G) the cordinates of the geometric center of gravity.

Parameters of the genetic algorithm are given table 2. The probability of crossover of real and discrete variables are very low: in this example, the global optimizer looks like more an evolution strategy than a genetic algorithm. These parameters are set to such low values because the high compacity of problem does not permit to generate promising solutions with crossovers. For high compacity problems, the priority is given to mutation and swap operators.

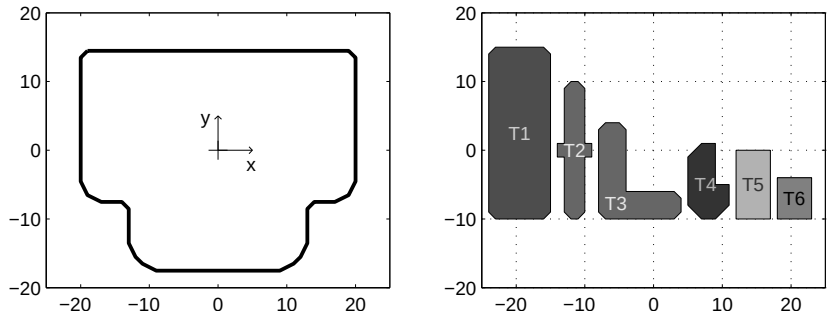
5.2 Results and analysis

Figure 6 presents a set of efficient solutions extracted from the Pareto front of figure 7. Subfigures are sorted in the increasing order of objective 1 and decreasing order for objective 2. On each solution presented, every component is in contact with another one, and one can observe that there is very little space to move components. One can also observe that the darkest components, *i.e.* the components with the largest density, are located close to the center of the trunk or along the y-axis: these components have a larger impact on the objective functions and by placing them close to the geometric center of gravity of the trunk, one optimizes both objective functions. Once the Pareto front is found, a multi-criteria analysis can be performed to choose one solution.

Table 2. PARAMETERS OF THE GA

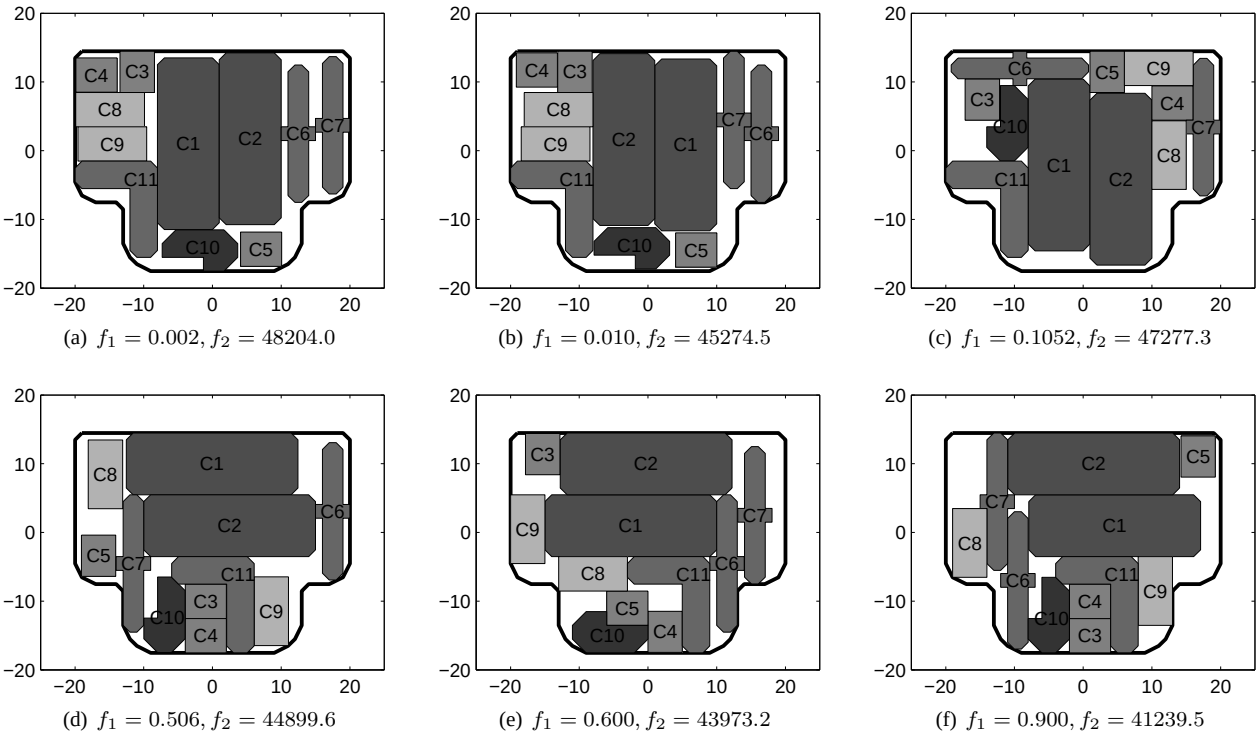
Number of generations	100
Number of individuals	100
Crossover probability of real variables	0.05
Mutation probability of real variables	0.4
Crossover probability of binary/discrete variables	0
Mutation probability of binary/discrete variables	0.3
Distribution index η_c for real variable crossover	5
Distribution index η_m for real variable mutation	5
Swapping probability of two components	0.05
Relative coefficient used for the ε-domination	0.001
Use of phenotypic and genotypic distance	Yes
Use of restricted selection operator	Yes

Figure 7 presents the different trade-off surfaces obtained for the different simulations. One simulation corresponds to one execution of the genetic algorithm. Each simulation evaluates $10,000$ solutions in 15 minutes. The problem is solved with 13 different initial random populations. Each point corresponds to a feasible solution, meaning that the corresponding solution satisfies placement constraints. Identical symbols represent the trade-off surface obtained from one simulation, *i.e.* the Pareto set obtained. The first conclusion that can be drawn from this figure is that there are large disparities between results. Two different initial populations do not converge towards the same trade-off surface. If one has a look at the solutions proposed at the end of a simulation, one can see that the layouts associated with these solutions are very close to each other, meaning that diversity of the population in the variable space and in the objective space has been lost. The loss of diversisty can be explained by the fact that the first feasible solutions found generate a selection pressure on the other unfeasible solutions. As a consequence, unfeasible but promising solutions are not selected for next generations,



(a) Trunk geometry (b) Components

Figure 5. PRESENTATION OF THE CONTAINER AND THE DIFFERENT COMPONENTS OF THE EXAMPLE.



(a) $f_1 = 0.002, f_2 = 48204.0$ (b) $f_1 = 0.010, f_2 = 45274.5$ (c) $f_1 = 0.1052, f_2 = 47277.3$

(d) $f_1 = 0.506, f_2 = 44899.6$ (e) $f_1 = 0.600, f_2 = 43973.2$ (f) $f_1 = 0.900, f_2 = 41239.5$

Figure 6. EXAMPLE OF SOLUTIONS EXTRACTED FROM THE PARETO FRONT.

making them disappear. After a certain number of generations, all solutions begin to look like each other and the convergence stalls, the genetic operators do not succeed in proposing new efficient solutions. Deb *et al.* [27] have introduced the concept of ε-domination to preserve diversity in the population. This relaxation on the dominance relation allows inferior solutions to be kept in the population while preserving diversity. This operator could be extended to constraints in order to preserve diversity of population for problems with high compacity. It may also be interesting to regenerate a part of the population when one detects that diversity is being lost. Tiwari *et al.* [16] use this feature inside their steady-state genetic algorithm.

Numerical results tend to prove that the higher the compacity of the problem is, the harder it is to find the set of efficient solutions, and once a feasible solution has been found it is difficult to change completely the topology of the solution. If the same problem is solved without one component of type 1, the compacity falls to 59.6%, and the trade-off surfaces obtained are identical whatever the initial population. When compacity of the problem grows, the problem transforms itself into a C&P problem, where the main objective is to find a feasible solution.

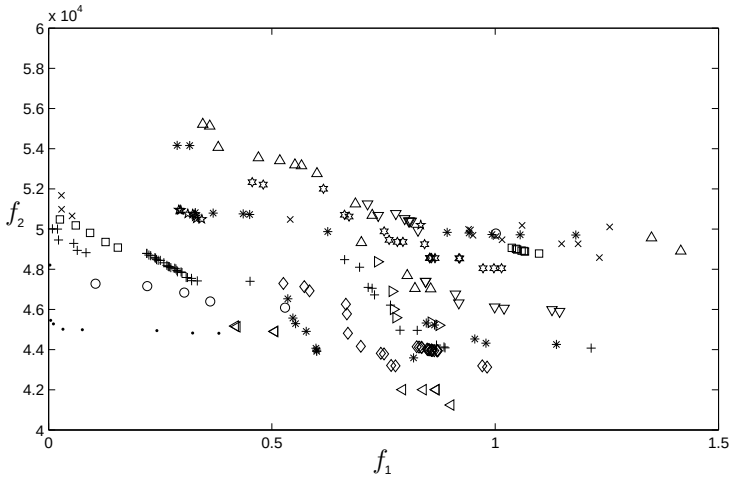


Figure 7. PRESENTATION OF DIFFERENT PARETO FRONTS OBTAINED FOR DIFFERENT SEEDS.

6 3D separation algorithm

The proposed method can be adapted to 3D, the difficulty being in the adaptation of the separation algorithm. The genetic algorithm can easily be adapted to the 3D version of the problem. Imamichi *et al.* [23] has already adapted his separation algorithm to 3D, however only with simple components and enclosure. To adapt the separation algorithm, the first task consists in converting polyhedra in sets of spheres. Hubbard [30] and Bradshaw *et al.* [31] have already solved this problem by approximating polyhedra with hierarchies of spheres based on component medial-axis surfaces (skeletal representations of components). Non-overlap constraints can be checked by simple distance computations between sphere centers. Non-protrusion constraints require to compute the minimum distance between sphere centers and the boundary of the enclosure. The challenge of such an algorithm is to find the correct trade-off between the number of spheres required to represent components and the computational complexity of the collision detection. Preliminary studies reveal that only decompositions based on medial axis provide interesting results. Other methods such as sphere-trees based on octree decomposition are prone to premature convergence during the separation optimization.

7 Conclusion

This paper has introduced a 2D multi-objective placement method for complex geometry components. The proposed relaxed placement technique is based on the hybridation of a genetic algorithm and a separation algorithm, and allows one to solve placement problems with several types of placement constraints. Applications of the proposed method can be found in engineering domains, where placement problems have no particular specificities and can not be treated with classical methods.

Results show that high quality solutions can be obtained with appropriate parameters for the genetic algorithms. Immediate work includes a study of the influence of the initial population and a statistical study of the influence of the parameters of the genetic algorithm. Several tests need to be performed to see if diversity can be maintained inside the population of solutions. Trade-off surfaces obtained for different simulations must be numerically evaluated with multi-objective indicators (*e.g.* delineation, distance, diversity and hypervolume) to quantify the differences between them.

Several extensions can be included in this work: the collision detection process can be speeded up by using a hierarchical description of components, articulated components could be taken into account with the circle decomposition of components. Future works also include comparison of different modelling techniques for high compacity problems: indeed placement problems with a large compacity are close to packing problems, for which fast and efficient technique have been proposed [16].

REFERENCES

[1] Aladahalli, C., Cagan, J., and Shimada, K., 2007. "Objective function effect based pattern search - theoretical framework inspired by 3D component layout". *Journal of Mechanical Design*, **129**(3), Mar., pp. 243–254.

[2] Wäscher, G., Haußner, H., and Schumann, H., 2007. "An improved typology of cutting and packing problems". *European Journal of Operational Research*, **183**, Dec., pp. 1109–1130.

[3] Dyckhoff, H., 1990. "A typology of cutting and packing problems". *European Journal of Operational Research*, **44**(2), Jan., pp. 145–159.

[4] Kellerer, H., Pferschy, U., and Pisinger, D., 2004. *Knapsack Problems*. Springer, Berlin, Germany.

[5] Carlier, J., Clautiaux, F., and Moukrim, A., 2007. "New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation". *Computers & Operations Research,* **34**(8), Aug., pp. 2223–2250.

[6] Egeblad, J., Nielsen, B. K., and Odgaard, A., 2007. "Fast neighborhood search for two- and three-dimensional nesting problems". *European Journal of Operational Research*, **127**(3), Dec., pp. 1249–1266.

[7] Gomes, M. A., and Oliveira, J. F., 2006. "Solving irregular strip packing problems by hybridising simulated annealing and linear programming". *European Journal of Operational Research*, **171**(3), June, pp. 811–829.

[8] Pisinger, D., 2002. "Heuristics for the container loading problem". *European Journal of Operational Research*, **141**, pp. 382–392.

[9] Drira, A., Pierreval, H., and Hajri-Gabouj, S., 2007. "Facility layout problems: A survey". *Annual Reviews in Control*, **31**(2), pp. 255–267.

[10] Egeblad, J., 2003. "Placement techniques for VLSI layout using sequence-pair legalization". Master's thesis, Department of Computer Science, University of Copenhagen, July.

[11] Zhang, B., Teng, H.-F., and Shi, Y.-J., 2008. "Layout optimization of satellite module using soft computing techniques". *Applied Soft Computing*, **8**(1), pp. 507–521.

[12] Cagan, J., Shimada, K., and Yin, S., 2002. "A survey of computational approaches to three-dimensional layout problems.". *Computer-Aided Design*, **34**(8), pp. 597–611.

[13] Grignon, P. M., and Fadel, G. M., 2004. "A GA based configuration design optimization method". *Journal of Mechanical Design*, **126**(1), Jan., pp. 6–15.

[14] Aladahalli, C., Cagan, J., and Shimada, K., 2007. "Objective function effect based pattern search - an implementation for 3D component layout". *Journal of Mechanical Design*, **129**(3), Mar., pp. 255–265.

[15] Miao, Y., Fadel, G. M., and Gantovnik, V. B., 2008. "Vehicle configuration design with a packing genetic algorithm". *International Journal of Heavy Vehicle Systems*, **15**, 12, pp. 433–448(16).

[16] Tiwari, S., Fadel, G., and Fenyes, P., 2008. "A fast and efficient three dimensional compact packing algorithm for free-form objects". In Proceedings of ASME International Design Engineering & Computers and Information in Engineering Conference (ASME DETC/CIE 2008). DETC2008-50097.

[17] Dong, H., Fadel, G. M., and Blouin, V. Y., 2006. "Vehicle component layout with shape morphing - an initial study". In ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference.

[18] Tiwari, S., Fadel, G., and Gantovnik, V., 2006. "A survey of various encoding schemes and associated placement algorithms applied to packing and layout problems". In Proceedings of IDETC/CIE. ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference.

[19] Bennell, J. A., and Song, X., 2008. "A comprehensive and robust procedure for obtaining the nofit polygon using minkowski sums". *Computers & Operations Research*, **35**(1), pp. 267–281.

[20] Imamichi, T., Yagiura, M., and Nagamochi, H., 2007. "An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem". In Proceedings of the Third International Symposium on Scheduling, Tokyo, Japan, no. 9, pp. 132–137.

[21] Imamichi, T., and Nagamochi, H., 2007. "A multi-sphere scheme for 2D and 3D packing problems". In SLS, pp. 207–211.

[22] Imamichi, T., Arahori, Y., Gim, J., Hong, S.-H., and Nagamochi, H., 2008. Removing overlaps in label layouts using multi-sphere scheme. Tech. Rep. 2008-006, Department of Applied Mathematics and Physics, Kyoto University and School of Information Technologies, University of Sydney.

[23] Imamichi, T., and Nagamochi, H., 2008. "Designing algorithms with multi-sphere scheme". In ICKS '08: Proceedings of the International Conference on Informatics Education and Research for Knowledge-Circulating Society (icks 2008), IEEE Computer Society, pp. 125–130.

[24] Agarwal, P. K., Guibas, L. J., Har-Peled, S., Rabinovitch, A., and Sharir, M., 2000. "Penetration depth of two convex polytopes in 3D". *Nordic Journal of Computing*, **7**(3), pp. 227–240.

[25] Liu, D. C., and Nocedal, J., 1989. "On the limited memory BFGS method for large scale optimization". *Math. Program.*, **45**(3), pp. 503–528.

[26] Dighe, R., and Jakiela, M. J., 1996. "Solving pattern nesting problems with genetic algorithms employing task decomposition and contact detection". *Evolutionary Computation*, **3**(3), pp. 239–266.

[27] Deb, K., and Tiwari, S., 2008. "Omni-Optimizer: A generic Evolutionary Algorithm for single and multi-objective optimization". *European Journal of Operational Research*, **185**, Mar., pp. 1062–1087.

[28] Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T., 2000. "A fast elitist Non-dominated Sorting Genetic Algorithm for multi-objective optimization: NSGA-II". In Proceedings of the Parallel Problem Solving from Nature VI Conference, Springer. Lecture Notes in Computer Science No. 1917, pp. 849–858.

[29] Deb, K., and Agrawal, R. B., 1995. "Simulated binary crossover for continuous search space". *Complex Systems*,

9, pp. 115–148.

[30] Hubbard, P. M., 1996. "Approximating polyhedra with spheres for time-critical collision detection". *ACM Transactions on Graphics*, **15**(3), pp. 179–210.

[31] Bradshaw, G., and O'sullivan, C., 2000. "Extracting geometric models from medieval moulding profiles for case-based reasoning.". In KES, R. J. Howlett and L. C. Jain, eds., IEEE, pp. 531–536.