Constraint Solving and Particle Swarm Optimization for Fixture Layout Optimization

Anna Vitali
Alma Mater Studiorum University of Bologna, Italy

Roberto Amadini
Alma Mater Studiorum University of Bologna, Italy

Vittorio Maniezzo
Alma Mater Studiorum University of Bologna, Italy

Maurizio Gabbrielli
Alma Mater Studiorum University of Bologna, Italy

Abstract

In the wood industry, ensuring the stability and immobility of a workpiece during machining is essential. Vacuum suction cups are commonly used as fixtures, as they can rotate to better conform to the workpiece perimeter. However, this rotational capability introduces additional complexity into fixture placement. This paper addresses the problem of optimally positioning rotatable fixtures by combining two complementary techniques. First, we solve a restricted version of the problem that neglects rotations using a constraint solver, yielding an initial feasible solution. This solution is then refined using a Particle Swarm Optimization algorithm that accounts for rotation. Experimental results show that the quality of the initial solution significantly influences the final placement, and that the proposed approach can effectively support expert operators in making improved decisions.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming

Keywords and phrases Constraint Programming, Mixed-Integer Programming, Particle Swarm Optimization, Fixture Layout Optimization, Industrial Application

Digital Object Identifier 10.4230/LIPIcs.CP.2026.56

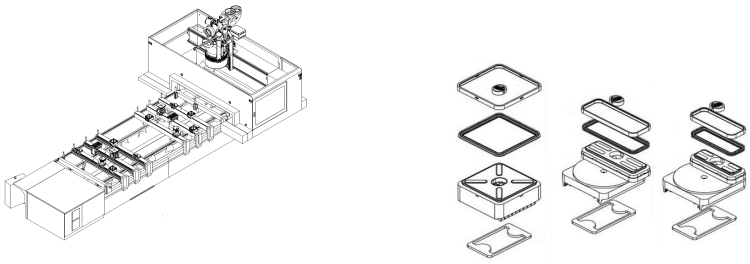
Funding This work is supported by PNRR – M4C2 – Investimenti 1.3, 3.3 e 3.4.

1 Introduction and Related Works

In the wood industry, finished products are usually obtained by processing semi-finished components, such as wood panels, using a dedicated machine called a *work center*. A typical work center (Fig. 1a) comprises a worktable with movable bars that translate along the horizontal axis, above which vacuum suction cup fixtures can be positioned; these fixtures slide along the vertical axis and can freely rotate in order to keep the workpiece stable during machining.

The right positioning of the fixtures reduces the effects of vibrations caused by the cutting tools, enhancing the quality of the final processed product [29, 12]. Modern machines employ square or rectangular suction cup fixtures featuring a square base that allows them to be placed on top of the bars and rotate through 360 degrees (Fig. 1b). The center of rotation coincides with the centroid of the square base, enabling the suction cup to remain outward-facing and thus ensuring better adhesion along the perimeter of the workpiece.

The problem of determining a fixture layout that maximizes workpiece stability has been widely studied. Li et al. [16] proposed a flexible fixtures system for sheet metal assembly based on assembly variation analysis using the Lagrangian conditional extremum method. Zhang et al. [32] developed a non-linear model for fixture layout optimization of large workpieces via the Augmented Lagrangian method. Selvakumar et al. [26] combined Artificial



(a) Work center. **(b)** Vacuum suction cup fixtures.

Figure 1 On the left, a work center featuring a worktable with movable bars. On the right, fixtures used to secure the workpiece during machining operations.

Neural Networks (ANNs) with Design of Experiments (DOE) for machining fixture layout optimization. Padmanaban et al. [23] used Finite Element Analysis (FEA) and discrete and continuous Ant Colony Algorithms (ACA) to reduce sheet metal deformation. Liao [19] employed a two-phase Genetic Algorithm (GA) to identify and refine locating and clamping zones for workpieces. Li and Melkote [17] applied Sequential Quadratic Programming (SQP) for simultaneous optimization of fixture layout and clamping forces. Wang et al. [31] integrated neural networks with the Bat Algorithm (BA) to minimize deformation of workpieces. Liu et al. [20] proposed Improved Particle Swarm Optimization (IPSO) for large thin-walled parts, while Sundararaman et al. [28] combined Response Surface Methodology (RSM) with GA and PSO, achieving superior results with PSO.

However, most of these works focus on the *metal industry* and address the determination of the number and placement of fixtures according to the "N-2-1" locating principle, with the goal of reducing workpiece deformation. This principle is an adaptation of the classical "3-2-1" scheme and requires N locators on the primary surface, two on the secondary surface, and one on the tertiary surface in order to uniquely constrain the position and orientation of the workpiece [31]. This approach cannot be applied to work centers used in the wood industry, since the workpiece can only be supported from below. Moreover, fixtures used in metalworking differ from those used in woodworking, as they typically cannot rotate. While allowing fixture rotation can significantly improve workpiece stability, it also increases the complexity of the optimization problem.

In this paper, we aim to determine a sound, and possibly optimal, placement of fixtures under the bottom surface of the workpiece, explicitly accounting for the rotation angle of each fixture. Our approach consists of two steps. First, we compute a feasible, but generally sub-optimal, placement using a constraint solver. This represents a simplification of the original problem, as fixture rotations are not considered. Second, the feasible solution is improved through a local search approach based on *Particle Swarm Optimization* (PSO), which explicitly accounts for fixture rotations.

This work extends [30], where a *Constraint Programming* (CP) model for fixture layout optimization without rotation was developed. We extend that model to handle pass-through machining and areas to be extruded, since a workpiece may contain regions where no fixture can be placed. CP provides native support for useful *global* constraints (e.g., `no-overlap`) and can be combined with *Large Neighborhood Search* (LNS), which has proven effective in this context.

In addition, we introduce a newly defined, and generally more efficient, *Mixed Integer Programming* (MIP) model. Both CP and MIP models can be solved using appropriate solvers to obtain an initial feasible solution. However, neither model accounts for fixture

rotation, as allowing rotation angles that are not multiples of 90° significantly increases the number of possible configurations and makes the problem substantially harder to solve. Furthermore, inclined fixtures complicate overlap checking; in CP solvers, this also requires approximations due to the lack of native support for continuous variables. For these reasons, fixture rotation is addressed by improving the CP or MIP solution using a PSO-based approach, which iteratively refines the incumbent solution while allowing fixtures to rotate freely.

PSO is particularly well-suited for this refinement phase due to its faster convergence and stronger global search capability in high-dimensional and multi-objective optimization problems. Additionally, PSO's simple structure, fewer control parameters, and ease of hybridization enable robust optimization across diverse manufacturing, scheduling, and design applications compared to Genetic algorithms (GA), Simulated Annealing (SA), and gradient-based methods [14].

The main contributions of this article can be summarized as follows:

- an extension of the CP model in [30], including new pass-through machining constraints, the definition of a revised objective function and the use of LNS when supported by the CP solver;
- the introduction of a new MIP model providing a linear formulation of the constraints;
- the integration of CP and MIP solutions with a PSO algorithm enabling fixture rotations;
- the validation of the proposed approach on *real* industrial case studies, with comparisons against solutions produced by an expert operator.

Empirical results show that both the CP and MIP models consistently outperform the expert operator, and that PSO is able to further improve all initial layouts, resulting in higher moments of inertia.[1]

Paper structure. The paper is organized as follows. Section 2 presents the background context. Section 3 describes the problem. Section 4 introduces the CP model. Section 5 presents the MIP model. Section 6 describes the PSO-based algorithm used to improve the solution and account for fixture rotation. Section 7 discusses the experimental results, and Section 8 concludes the paper.

2 Background

We provide background notions on principal moments of inertia and Particle Swarm Optimization. We assume the reader is already familiar with the basic concepts of CP and MIP.

2.1 Principal moments of inertia

To maximize the stability of the workpiece through fixture placement, the *principal moments* of inertia of the fixture layout system can be analyzed.

The moment of inertia is a geometric property that quantifies a structure's resistance to bending and torsional forces. It is computed from the distribution of mass relative to a given axis and plays a crucial role in determining the stiffness of structural elements under load. Higher moments of inertia indicate greater resistance to deformation, making this quantity a key factor in the design of fixtures that minimize workpiece deflection during machining operations [13, 5].

[1] This work is developed in collaboration with a leading company in the woodworking machinery sector. The name of the company is omitted for privacy reasons.

To compute the principal moments of inertia of a rectangular fixture system [3, 7, 11], each fixture can be represented as a polygon i of m vertices $(x_{i,1}, y_{i,1}), \ldots, (x_{i,m}, y_{i,m})$ with area A_i, centroid coordinates (cx_i, cy_i) and rotation angle α_i. We first compute the *absolute moments* of inertia of each polygon with respect to the global coordinate system:

$$J'_{x,i} = \tfrac{1}{12} \sum_{j=1}^{m} (y_{i,j}^2 + y_{i,j}y_{i,j+1} + y_{i,j+1}^2)(x_{i,j}y_{i,j+1} - x_{i,j+1}y_{i,j})$$
$$J'_{y,i} = \tfrac{1}{12} \sum_{j=1}^{m} (x_{i,j}^2 + x_{i,j}x_{i,j+1} + x_{i,j+1}^2)(x_{i,j}y_{i,j+1} - x_{i,j+1}y_{i,j})$$
$$J'_{xy,i} = \tfrac{1}{24} \sum_{j=1}^{m} (x_{i,j}y_{i,j+1} + 2x_{i,j}y_{i,j} + 2x_{i,j+1}y_{i,j+1} + x_{i,j+1}y_{i,j})(x_{i,j}y_{i,j+1} - x_{i,j+1}y_{i,j}).$$

The moments of inertia referred to the centroid of each polygon are then computed as:

$$J''_{x,i} = J'_{x,i} - A_i\, cy_i^2, \qquad J''_{y,i} = J'_{y,i} - A_i\, cx_i^2, \qquad J''_{xy,i} = J'_{xy,i} - A_i\, cx_i\, cy_i.$$

Since a polygon can be rotated by an angle α_i, the corresponding rotated moments are:

$$J_{x,i} = J''_{x,i} \cos^2 \alpha_i + J''_{y,i} \sin^2 \alpha_i - 2J''_{xy,i} \sin \alpha_i \cos \alpha_i,$$
$$J_{y,i} = J''_{x,i} \sin^2 \alpha_i + J''_{y,i} \cos^2 \alpha_i - 2J''_{xy,i} \sin \alpha_i \cos \alpha_i,$$
$$J_{xy,i} = (J''_{x,i} - J''_{y,i}) \sin \alpha_i \cos \alpha_i + J''_{xy,i}(\cos^2 \alpha_i - \sin^2 \alpha_i).$$

The total moments of inertia with respect to the global axes are then computed as:

$$J_x = \sum_{i=1}^{n} J_{x,i} + A_i(cy_i - g_y)^2, \qquad J_y = \sum_{i=1}^{n} J_{y,i} + A_i(cx_i - g_x)^2,$$
$$J_{xy} = \sum_{i=1}^{n} J_{xy,i} + A_i(cx_i - g_x)(cy_i - g_y)$$

where g_x and g_y denote the coordinates of the overall centroid of the system, computed as:

$$g_x = \frac{\sum_{i=1}^{n} A_i\, cx_i}{A}, \qquad g_y = \frac{\sum_{i=1}^{n} A_i\, cy_i}{A}$$

where $A = \sum_{i=1}^{n} A_i$ is the total area. The principal moments of inertia of the overall system are obtained by first evaluating:

$$J'_{xg} = J_x - A\, g_y^2, \qquad J'_{yg} = J_y - A\, g_x^2, \qquad J'_{xyg} = J_{xy} - A\, g_x\, g_y,$$

and then computing:

$$J_\xi = \tfrac{1}{2}\left((J'_{xg} + J'_{yg}) - \sqrt{(J'_{xg} - J'_{yg})^2 + 4(J'_{xyg})^2}\right),$$
$$J_\eta = \tfrac{1}{2}\left((J'_{xg} + J'_{yg}) + \sqrt{(J'_{xg} - J'_{yg})^2 + 4(J'_{xyg})^2}\right).$$

Layouts that provide good stability for the workpiece are those maximizing the sum $J_\xi + J_\eta$.

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an iterative computational technique inspired by swarming theory and originally motivated by studies of collective behavior in bird flocks [27, 21].

PSO operates on a swarm of particles, each characterized by two vectors: the position p_i and the velocity v_i. Each particle's position represents a candidate solution and is associated with a *fitness* value that measures the solution quality. At each iteration $t = 1, 2, \ldots$ the particles explore the search space by updating their velocities and positions according to the following equations:

$$v_i^{(t+1)} = w\, v_i^{(t)} + c_1 r_1 \left(p_i^* - p_i^{(t)}\right) + c_2 r_2 \left(g^* - p_i^{(t)}\right)$$
$$p_i^{(t+1)} = p_i^{(t)} + v_i^{(t+1)}$$

where:

- w is the *inertia weight*, controlling how much of the previous velocity is retained;
- c_1 is the *cognitive coefficient*, guiding the particle toward its *personal best position* p_i^*;
- c_2 is the *social coefficient*, guiding particles toward the *global best position* g^* of the swarm;
- r_1 and r_2 are random numbers typically drawn from a uniform distribution in $[0, 1]$, introducing stochasticity into the cognitive and social components, respectively.

3 Problem Description

The layout optimization problem addressed in this work consists of determining, given the workpiece shape, the available rectangular fixtures (possibly of different sizes), and the available worktable bars, a selection of fixtures, and corresponding fixture types, that maximize workpiece stability.

To ensure the feasibility of a placement, the following constraints must be satisfied:

1. the number of fixtures placed must not exceed their availability;
2. fixtures must be positioned so as to satisfy the minimum safety distances;
3. fixtures must lie within the boundaries of the workpiece and beneath its surface;
4. fixtures must not overlap each other;
5. fixtures must not be located beneath areas or regions to be extruded.

The mathematical formalization of these constraints depends on the choice of decision variables and is therefore addressed separately in Sections 4 and 5 for the CP and MIP models, respectively. However, both models are defined over the same set of input *parameters*, which are listed below:

- the width $\texttt{WTab}$ and height $\texttt{HTab}$ of the machine worktable;
- the minimum horizontal distance $\texttt{WMin}$ between two fixtures placed on different bars;
- the minimum vertical distance $\texttt{HMin}$ between two fixtures placed on the same bar;
- the number $\texttt{NTypes}$ of different fixture types;
- the number $\texttt{NFix}_i$ of available fixtures for each type $i \in [1..\texttt{NTypes}]$;
- the total number of available fixtures $\texttt{NFix} = \sum_{i=1}^{\texttt{NTypes}} \texttt{NFix}_i$;
- the minimum $\texttt{NMin}$ and maximum $\texttt{NMax}$ number of fixtures to place;
- the width $\texttt{WFix}_i$ and height $\texttt{HFix}_i$ of each fixture of type i;
- the workpiece shape, by a set of linear inequalities of the form $\texttt{a}\,x + \texttt{b}\,y + \texttt{c} \leq 0$;
- the number $\texttt{NExt}$ of areas to be extruded;
- the width $\texttt{WExt}_i$, height $\texttt{HExt}_i$, and the coordinates $(\texttt{XExt}_i, \texttt{YExt}_i)$ of the lower-left corner of the smallest axis-aligned rectangle enclosing each area $i \in [1..\texttt{NExt}]$ to be extruded.

The last point reflects the fact that, to facilitate the solving process, we approximate extruded areas with their smallest enclosing rectangles. In practice, these regions usually correspond to simple geometries such as circles, rectangles, or ellipses; more complex shapes are typically refined manually.

Both the CP and MIP models presented in the following sections are defined over these parameters and share the same goal: maximizing workpiece stability by maximizing the sum of the principal moments of inertia. The ideal *objective function* would therefore be:

$$f_{iner} = J_\xi + J_\eta \tag{1}$$

where J_ξ and J_η are the principal moments of inertia defined in Section 2.

However, computing J_ξ and J_η involves nonlinear operations such as squaring, square roots, and transcendental functions (sine and cosine), which are difficult to handle efficiently with CP and MIP solvers [15]. For this reason, following [30], we adopt an alternative objective function f_{dist} that maximizes the sum of the pairwise Manhattan distances between fixture centroids, augmented by the area of the selected fixtures:

$$f_{dist} = \sum_{1 \le i < j \le n} \left(|cx_i - cx_j| + |cy_i - cy_j| \right) + \sum_{k=1}^{n} (\texttt{WFix}_{t_k} \cdot \texttt{HFix}_{t_k}) \tag{2}$$

where (x_k, y_k) are the coordinates of the lower-left corner of the k-th placed fixture, having type t_k (n is the total number of placed fixtures) and centroid $(cx_k, cy_k) = (x_k + \frac{\texttt{WFix}_{t_k}}{2}, y_k + \frac{\texttt{HFix}_{t_k}}{2})$.

Note that this formulation differs from [30], where the semi-perimeter $\texttt{WFix}_{t_k} + \texttt{HFix}_{t_k}$ is used instead of the area $\texttt{WFix}_{t_k} \cdot \texttt{HFix}_{t_k}$, in order to preserve the dimensional consistency. In this work, we adopt the area term to prioritize fixtures with larger contact surfaces, thereby enhancing stability in nontrivial industrial workpieces. In contrast, [30] evaluates the approach only on a proof-of-concept example involving a simple geometry. We empirically verified that this approach provides a weaker surrogate for the inertia objective in more complex geometries.[2]

The rationale behind f_{dist} is to increase stability by spreading fixtures as far apart as possible, thereby pushing them toward the boundaries of the workpiece. The additional term favoring larger fixtures reflects their higher contribution to overall stability.

Optimizing f_{dist} corresponds to solving a *restricted* version of the original problem in which fixture rotations are not allowed. This restriction reduces the feasible solution space; consequently, every solution of the restricted problem is also feasible for the original problem with rotations. In principle, this restriction might also make the problem infeasible even though the original problem admits a solution, as feasibility might only be achievable through fixture rotation. In practice, this situation does not occur in our scenario, since it would require very small workpieces or extremely complex geometries that fall outside the scope of the considered industrial cases.

We therefore use CP or MIP solvers to solve the restricted problem to optimality with respect to f_{dist}, obtaining an initial feasible solution. This solution is then optionally refined using the PSO-based approach described in Section 6, which instead employs the original objective function f_{iner} as the fitness function to evaluate and improve fixture placements.

[2] In the experiments of Section 7, the Pearson correlation coefficient [4] between f_{dist} and f_{iner} when using the area-based formulation is much higher (0.79) than the semi-perimeter-based one (0.39).

4 Constraint Programming Model

We formalize the decision variables and constraints of the CP model, taking advantage of the *global constraints* [2] that characterize the CP paradigm. The model is defined over the parameters introduced in Section 3 and aims to maximize the objective function f_{dist} formalized in Equation 2.

4.1 Variables

To determine which fixtures are selected and their placement on the worktable, we define three arrays of integer decision variables x, y, and t of size $\texttt{NFix}$ such that, for $i = 1, \ldots, \texttt{NFix}$:

- $(x_i, y_i) \in [0..\texttt{WTab}] \times [0..\texttt{HTab}]$ denotes the coordinates of the lower-left corner of the i-th fixture;
- $t_i \in [0..\texttt{NTypes}]$ denotes the type of the i-th fixture, where type 0 represents an unused fixture.

For each fixture type $i \in [1..\texttt{NTypes}]$, we also define an integer variable $n_i \in [0..\texttt{NFix}_i]$, counting the number of selected fixtures of type i. Finally, we define an integer variable n, corresponding to the total number of fixtures employed.

4.2 Constraints

For each fixture type $i \in [1..\texttt{NTypes}]$, we exploit the `count` global constraint to enforce

$$n_i = \texttt{count}(t, i)$$

which counts the number of fixtures of type i in the array t. The total number of placed fixtures is then defined as $n = \sum_{i=1}^{\texttt{NTypes}} n_i$.

Since not all available fixtures must necessarily be used, we enforce the following invariant:

$$(i > n) \iff (x_i = \texttt{WTab} \ \wedge \ y_i = \texttt{HTab} \ \wedge \ t_i = 0).$$

This ensures that unused fixtures are assigned type 0 and placed at a dummy location $(\texttt{WTab}, \texttt{HTab})$ outside the feasible region.

The workpiece shape is defined by linear inequalities of the form $\texttt{a}\,x + \texttt{b}\,y + \texttt{c} \leq 0$. To ensure that each selected fixture i is placed within the workpiece, we must satisfy $\texttt{a}\,x'_i + \texttt{b}\,y'_i + \texttt{c} \leq 0$, where:

$$x'_i = \begin{cases} x_i & \text{if } \texttt{a} \leq 0, \\ x_i + \texttt{WFix}_{t_i} & \text{if } \texttt{a} > 0, \end{cases} \qquad y'_i = \begin{cases} y_i & \text{if } \texttt{b} \leq 0, \\ y_i + \texttt{HFix}_{t_i} & \text{if } \texttt{b} > 0. \end{cases}$$

Areas to be extruded are approximated by axis-aligned rectangles, allowing exclusion constraints to be efficiently modeled using the `diffn` global constraint [2]. Given a set of rectangles defined by their origins and dimensions, this constraint ensures that they do not overlap:

$$\texttt{diffn}\big(x \oplus \texttt{XExt},\ y \oplus \texttt{YExt},\ [\texttt{WFix}_{t_i} \mid i \in [1..n]] \oplus \texttt{WExt},\ [\texttt{HFix}_{t_i} \mid i \in [1..n]] \oplus \texttt{HExt}\big)$$

where $\oplus$ denotes array concatenation. We use the non-strict version of `diffn`, in which zero-width rectangles can be positioned anywhere, ignoring the $\texttt{NFix} - n$ unused fixtures, to ensure that fixtures do not overlap with each other or with extrusion areas.

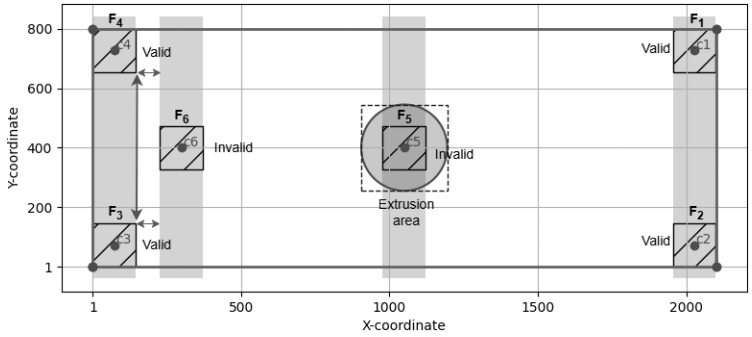


Figure 2 Example of fixture placement. Fixtures sharing the same centroid x-coordinate are placed on the same bar and must satisfy the vertical safety distance, while fixtures on different bars must satisfy the horizontal safety distance. Areas to be extruded are approximated by bounding rectangles and cannot host fixtures.

To enforce the minimum *vertical* distance `HMin` between fixtures placed on the same bar, and the minimum *horizontal* distance `WMin` between fixtures placed on different bars we impose:

$$\forall 1 \le i < j \le n: \quad \begin{cases} y_i + \texttt{HFix}_{t_i} + \texttt{HMin} \le y_j \ \vee \ y_j + \texttt{HFix}_{t_j} + \texttt{HMin} \le y_i, & \text{if } cx_i = cx_j, \\ x_i + \texttt{WFix}_{t_i} + \texttt{WMin} \le x_j, & \text{if } cx_i \neq cx_j. \end{cases}$$

where cx_i denotes the x-coordinate of the centroid of fixture i. Centroid coordinates are used instead of lower-left coordinates because fixtures may be wider than their supporting bar: while x_i may fall outside the horizontal span of the bar, the centroid cx_i always lies within it.

In this way, if two fixtures share the same centroid x-coordinate, they are forced to satisfy the minimum vertical distance `HMin`; otherwise, the minimum horizontal distance `WMin` is enforced. Since `WMin` is defined to be greater than the bar width, explicit modeling of bar positions is not required, because each fixture centroid is assumed to lie on the centerline of a bar.

Figure 2 illustrates valid and invalid fixture positions. Specifically, fixture F_6 is invalid because it violates the horizontal safety distance from fixtures F_3 and F_4, while fixture F_5 is invalid as it overlaps an area to be extruded.

Finally, symmetries that arise when fixtures of the same type are permuted without changing their spatial arrangement must be addressed. To break such symmetries, we impose a *lexicographic* ordering on fixture positions using the `lex_chain` global constraint: $(x_{i-1}, y_{i-1}) \preceq (x_i, y_i)$ for $i \in [2..\texttt{NFix}]$. Since fixtures do not overlap, ordering only consecutive fixtures is sufficient.

5 Mixed-Integer Programming Model

In this section, we present the Mixed-Integer Programming (MIP) formulation of the fixture layout problem. Compared to the CP model described in Section 4, all the constraints are expressed in linear form, and the positional decision variables are *continuous* rather than integer-valued. This important relaxation comes from the nature of MIP solving which, unlike most CP solvers, naturally handles non-negative continuous variables. Nevertheless, a number of discrete variables are still required to capture the combinatorial structure of the problem, as detailed below.

5.1 Variables

Variables x_i and y_i for $i = 1, \ldots, \texttt{NMax}$ have exactly the same semantics as those defined in Section 4.1, with the only difference that they are now continuous instead of discrete. Fixture types are encoded with a 0-1 matrix T of size $\texttt{NMax} \times \texttt{NTypes}$, whose semantics is:

$$T_{i,j} = \begin{cases} 1 & \text{if fixture } i \text{ is selected and has type } j, \\ 0 & \text{otherwise.} \end{cases}$$

The linearization of the constraints defined in Section 4.2 comes at the expense of the introduction of auxiliary 0-1 variables, detailed as follows:

- $used_i$, denoting whether fixture i is selected;
- $same\_bar_{i,j}$, denoting whether fixtures i and j lie on the same bar;
- $above\_in\_bar_{i,j}$, $below\_in\_bar_{i,j}$, denoting whether fixture i is above or below fixture j on the same bar;
- $above_{i,j}$, $below_{i,j}$, $left_{i,j}$, $right_{i,j}$, indicating whether fixture i is positioned above, below, to the left, or to the right of fixture j, respectively.
- $above\_ext_{i,j}$, $below\_ext_{i,j}$, $left\_ext_{i,j}$, $right\_ext_{i,j}$ indicating whether fixture i is positioned above, below, to the left, or to the right of an area j to be extruded, respectively.

We also define auxiliary variables w_i, h_i and t_i to denote, respectively, the width w_i, the height h_i, and the type t_i of each fixture.

5.2 Constraints

For each fixture $i \in [1..\texttt{NMax}]$ we impose:

$$used_i = \sum_{j=1}^{\texttt{NTypes}} T_{i,j} \qquad w_i = \sum_{j=1}^{\texttt{NTypes}} \texttt{WFix}_j \cdot T_{i,j} \qquad h_i = \sum_{j=1}^{\texttt{NTypes}} \texttt{HFix}_j \cdot T_{i,j} \qquad t_i = \sum_{j=1}^{\texttt{NTypes}} j \cdot T_{i,j}$$

To enforce availability constraints, we impose:

$$\sum_{i=1}^{\texttt{NMax}} T_{i,j} \leq \texttt{NFix}_j, \quad \forall j \in [1..\texttt{NTypes}] \qquad \texttt{NMin} \leq \sum_{i=1}^{\texttt{NMax}} used_i \leq \texttt{NMax}$$

To determine whether two fixtures lie on the same bar for all $i \in [1..\texttt{NMax}]$ and $j \in [i+1..\texttt{NMax}]$ we impose the following constraints:

$$same\_bar_{i,j} \leq used_i \qquad same\_bar_{i,j} \leq used_j$$
$$same\_bar_{i,j} = above\_in\_bar_{i,j} + below\_in\_bar_{i,j}$$
$$cx_i - cx_j \leq M(1 - same\_bar_{i,j}) \qquad cx_j - cx_i \leq M(1 - same\_bar_{i,j})$$

where $M = \texttt{WTab} \cdot \texttt{HTab}$ is the *"big-M"* constant. To enforce the appropriate safety distances:

$$cx_i + \texttt{WMin} \leq cx_j + M(same\_bar_{i,j} + right_{i,j})$$
$$cx_j + \texttt{WMin} \leq cx_i + M(same\_bar_{i,j} + left_{i,j})$$
$$y_i + h_i + \texttt{HMin} - y_j \leq M(1 - above\_in\_bar_{i,j})$$
$$y_j + h_j + \texttt{HMin} - y_i \leq M(1 - below\_in\_bar_{i,j}).$$

To prevent overlap between fixtures, for all $i, j \in [1..\texttt{NMax}]$ with $i < j$, we impose:

$$above_{i,j} + below_{i,j} + left_{i,j} + right_{i,j} \geq 1$$
$$x_i + w_i \leq x_j + M(1 - left_{i,j}) \qquad y_i + h_i \leq y_j + M(1 - below_{i,j})$$
$$x_j + w_j \leq x_i + M(1 - right_{i,j}) \qquad y_j + h_j \leq y_i + M(1 - above_{i,j})$$

The same constraints are applied to prevent overlap between fixtures and extrusion areas. For each fixture $i \in [1..\texttt{NMax}]$ and extrusion area $j \in [1..\texttt{NExt}]$, the above inequalities are instantiated using the variables $above\_ext_{i,j}$, $below\_ext_{i,j}$, $left\_ext_{i,j}$, and $right\_ext_{i,j}$.

The workpiece is defined by linear inequalities of the form $ax + by + c \leq 0$, so for each fixture $i \in [1..\texttt{NMax}]$ we impose $ax'_i + by'_i + c \leq M(1 - used_i)$, where x'_i and y'_i are defined as in the CP model (see Section 4.2).

To break symmetries, we impose an ordering on fixture types: $t_i \geq t_{i+1}$ for $i \in [1..\texttt{NMax}-1]$. We linearize f_{dist} with a big-M formulation unfolding each term of the form $u = |v|$ into:

$$u \geq v \ \wedge \ u \geq -v \ \wedge \ u \leq v + M(1-b) \ \wedge \ u \leq -v + Mb \ \wedge \ b \in \{0,1\}.$$

6 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is employed to explicitly account for fixture rotation, enabling the maximization of the sum of the principal moments of inertia, i.e., the objective function $f_{iner} = J_\xi + J_\eta$ defined in Equation 1.

Algorithm 1 presents the pseudocode of the implemented PSO approach. The position of each particle $s \in [1..\texttt{swarm\_size}]$ is represented by a matrix P_s with four rows and NMax columns, where NMax denotes the number of available fixtures. Each column of P_s corresponds to a fixture, while each row encodes one of the problem decision variables:

- $P_s[1, j]$: the x-coordinate of the lower-left corner of the j-th fixture;
- $P_s[2, j]$: the y-coordinate of the lower-left corner of the j-th fixture;
- $P_s[3, j]$: the type t associated with the j-th fixture;
- $P_s[4, j]$: the rotation angle α of the j-th fixture.

Each position matrix P_s represents a candidate solution associated with particle s. The availability of an initial solution obtained from the CP or MIP model guarantees that the swarm is initialized from a feasible configuration.

During the initialization phase (lines 1–5 of Algorithm 1), the first particle is assigned the provided initial solution P_0, whereas the remaining particles are initialized using randomly perturbed versions of P_0. The velocity matrix V_s of each particle s, which stores the velocity components corresponding to each element of the position matrix P_s, is randomly initialized, and the personal best position P^*_s is initially set equal to the particle's starting position.

In this formulation, the linear constraints of the MIP model are incorporated into the PSO framework and treated as soft constraints, with the non-overlap constraints adapted to account for fixture rotations. Accordingly, for each particle position P_s, the violation of the j-th constraint is evaluated by means of a binary function $cv_j(P_s)$, which indicates whether the constraint is violated or satisfied. The overall constraint violation $cv(P_s)$ for position P_s is computed as the sum of each individual constraint violation.

Once all particles have been initialized, they are evaluated in terms of the objective function $f_{iner}(P_s)$ and the constraint violation $cv(P_s)$, as reported in lines 6–10 of Algorithm 1. Based on this evaluation, the global best position of the swarm, denoted by P^*, is identified.

When fixtures are axis-aligned, non-overlap can be enforced through coordinate-wise comparisons, exploiting the orientation of rectangle vertices. However, introducing rotations makes this criterion no longer valid, since the global position of each vertex depends on the rotation angle and is computed using trigonometric transformations. Consequently, vertex-based non-overlap verification requires orientation-dependent logic, increasing implementation complexity. Moreover, vertex-based constraints are insufficient for oriented rectangles, as intersections may occur without any vertex of one rectangle lying inside the other, for instance in edge-to-edge configurations.

Algorithm 1 Particle Swarm Optimization for Fixture Layout Optimization.

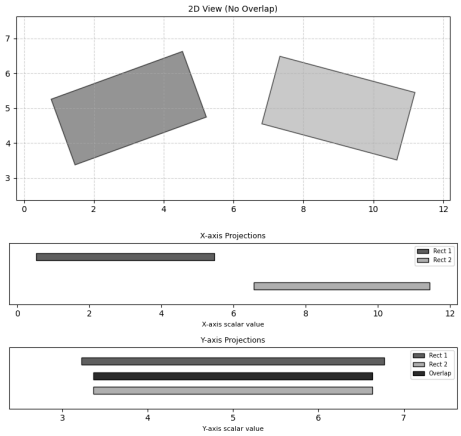
Require: Swarm size `swarm_size`, maximum number of iterations `max_iter`, inertia weight w, cognitive coefficient c_1, social coefficient c_2, variable bounds, initial solution P_0, maximum number of fixtures `NMax`
Ensure: Feasible solution P^* not worse than P_0

1: **for** $s = 1$ to `swarm_size` **do**
2: **if** $s = 1$ **then**
3: $P^* \leftarrow P_s^* \leftarrow P_s \leftarrow P_0$
4: **else**
5: $P_s^* \leftarrow P_s \leftarrow P_0$ + random perturbation
6: **if** $cv(P_s) = 0$ **then**
7: **if** $f_{iner}(P_s) > f_{iner}(P^*)$ **then**
8: $P^* \leftarrow P_s$
9: **end if**
10: **end if**
11: **end if**
12: Randomly initialize velocity matrix V_s
13: **end for**
14: **for** $k = 1$ to `max_iter` **do**
15: **for** $s = 1$ to `swarm_size` **do**
16: **for** $i = 1$ to 4 **do**
17: **for** $j = 1$ to `NMax` **do** ▷ Update velocity and position
18: Randomly generate $r_1, r_2 \in [0, 1]$
19: $V_s[i,j] \leftarrow wV_s[i,j] + c_1 r_1 (P_s^*[i,j] - P_s[i,j]) + c_2 r_2 (P^*[i,j] - P_s[i,j])$
20: $P_s[i,j] \leftarrow P_s[i,j] + V_s[i,j]$
21: **end for**
22: **end for**
23: **if** P_s outside the workpiece **then**
24: Identify the nearest vertex of the workpiece and the farthest vertex of the fixture
25: Translate P_s inside the workpiece area by the minimum required displacement
26: **end if**
27: **if** $cv(P_s) = 0$ **and** $cv(P_s^*) = 0$ **then** ▷ Deb's rules
28: **if** $f_{iner}(P_s) > f_{iner}(P_s^*)$ **then**
29: $P_s^* \leftarrow P_s$
30: **if** $f_{iner}(P_s^*) > f_{iner}(P^*)$ **then**
31: $P^* \leftarrow P_s^*$
32: **end if**
33: **end if**
34: **else if** $cv(P_s) < cv(P_s^*)$ **then**
35: $P_s^* \leftarrow P_s$
36: **end if**
37: **end for**
38: **end for**
39: **return** P^*

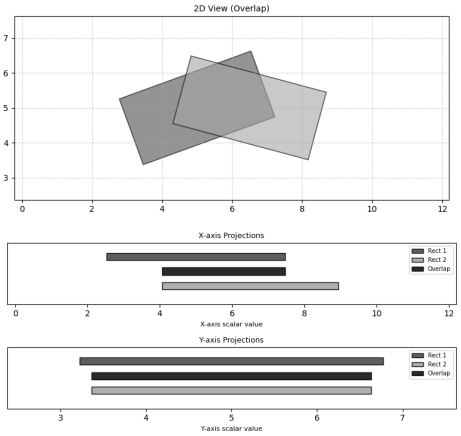
For this reason, the *Separation Axis Theorem* [18] is adopted to enforce the non-overlap constraint. The theorem states that two oriented rectangles do not overlap if and only if there exists at least one axis along which the projections of the two shapes are disjoint. Such an axis is called a separation axis. Checking non-overlap is thus performed by projecting both rectangles onto their principal axes and verifying the absence of intersection in at least one of these directions, as illustrated in Fig. 3.

During the search phase (lines 18–20 of Algorithm 1), the velocity matrix V_s and the position matrix P_s associated with particle s are iteratively updated according to the standard PSO update equations introduced in Section 2.2.

Whenever possible, constraint violations arising from updated positions are mitigated by adopting dedicated geometric repair algorithms, as reported in lines 23–26 of Algorithm 1. In particular, if a fixture lies partially or entirely outside the workpiece boundaries, the nearest vertex of the polygon representing the workpiece is identified, followed by the vertex of the fixture that is farthest from this point. The fixture is then translated by the minimum displacement required to ensure that all its coordinates lie within the workpiece boundary.



(a) Separation Axis Theorem with no overlap.



(b) Separation Axis Theorem with overlap.

Figure 3 Separation Axis Theorem for oriented rectangles. Each figure consists of three subplots. The first subplot illustrates the spatial configuration of the rectangles. The second and third subplots show the projections of their principal axes onto the x- and y-axes, respectively. An overlap occurs when the projections intersect along both axes; the overlap is highlighted in black in the second and third subplots.

The repair strategy is limited to geometric violations, as these constraints are the most suitable for correction without adversely affecting the satisfaction of other constraints. Their adoption has been shown to improve the quality of the positions found.

To guide the search toward constraint-respecting positions, as illustrated in lines 27–36 of Algorithm 1, candidate solutions are evaluated using Deb's rules [8]:

- Between two feasible solutions, retain the one with the higher objective value.
- Between a feasible and an infeasible solution, retain the feasible one.
- Between two infeasible solutions, retain the one with the lower constraint violation.

After updating the personal best position P_s^* of each particle s using Deb's rules, the global best position P^* of the swarm is updated accordingly. At the end of the final iteration, the best position found by the swarm is returned.

7 Evaluation

We evaluate our approach on five distinct workpieces: a stair step of a spiral staircase, a stair step of a simple staircase, a music speaker, a dashboard, and a coffee table (see Table 3). The work center is equipped with 24 square fixtures with dimensions 145×145 mm and 12 rectangular fixtures measuring 180×65 mm. For all workpieces the minimum and maximum number of fixtures were set to `NMin` = 3 and `NMax` = 6, respectively, except for the coffee table which, due to its large dimensions, required `NMin` = 6 and `NMax` = 10.

Both the CP and MIP models were implemented in the *MiniZinc* [22] language, to establish a fair comparison of their performance. The CP model was solved with Gecode [25], Chuffed [6], and OR-Tools [9] CP solvers, while for the MIP model we used the commercial solver Gurobi [10], with a time limit of 300 s for all solvers. To ensure a fair comparison, all solvers were executed in single-threaded mode.

We also include Gecode+LNS, a variant of Gecode implementing Large Neighborhood Search (LNS) [24]. Specifically, we configured the LNS procedure to randomly fix 80% of the decision variables corresponding to the x- and y-coordinates. A constant restart strategy is adopted, triggering a restart after 1000 explored nodes.

Table 1 Objective values for f_{dist} and f_{iner}. Optimal solutions with respect to f_{dist} are marked with *. For each workpiece, the maximum values of f_{dist} and f_{iner} are highlighted in bold, including ties. Note that solutions equivalent in terms of f_{dist} may differ in f_{iner}, hence different entries may be highlighted.

Workpiece	Obj.	Gecode	Gecode+LNS	Chuffed	OR-Tools	Gurobi
Stair step spiral staircase	f_{dist}	45655*	45629	45655*	45655*	**45662.155***
	f_{iner}	2.13968×10^9	1.94527×10^9	2.13968×10^9	2.13968×10^9	$\mathbf{2.17089 \times 10^9}$
Stair step simple staircase	f_{dist}	72823	72823	72823*	72747	**72826***
	f_{iner}	6.76573×10^9	6.76573×10^9	6.76512×10^9	6.68322×10^9	$\mathbf{6.76638 \times 10^9}$
Dashboard	f_{dist}	**55582***	**55582**	**55582***	**55582***	**55582***
	f_{iner}	7.10094×10^9	$\mathbf{7.10721 \times 10^9}$	7.08735×10^9	7.10094×10^9	7.08735×10^9
Speaker	f_{dist}	**49548***	**49548**	**49548***	**49548***	**49548***
	f_{iner}	$\mathbf{2.93433 \times 10^9}$	$\mathbf{2.93433 \times 10^9}$	$\mathbf{2.93433 \times 10^9}$	$\mathbf{2.93433 \times 10^9}$	$\mathbf{2.93433 \times 10^9}$
Coffee table	f_{dist}	198824	159541	198824	123812	**201686.505**
	f_{iner}	2.01861×10^{10}	1.69684×10^{10}	2.18655×10^{10}	2.01861×10^{10}	$\mathbf{2.19472 \times 10^{10}}$

To enable direct integration into the target industrial system, the PSO algorithm was implemented in Java. The hyperparameters of the PSO algorithm were tuned using the Optuna framework [1], resulting in the following configuration: `swarm_size` = 1374, `max_iter` = 2912, w = 0.669, c_1 = 2.385, and c_2 = 0.558. The tests were executed on a 64-bit x86 Windows machine equipped with an Intel Core i5 processor and 16 GB of RAM.

Table 1 reports the best values obtained with respect to the objective function f_{dist}, together with the corresponding *a posteriori* values of f_{iner}, computed for the solution that is optimal with respect to f_{dist}. Identical values of f_{dist} do not necessarily correspond to identical fixture placements; consequently, solutions that are optimal under f_{dist} may yield different values of f_{iner}. The optimal f_{dist} values obtained by the CP and MIP models may also differ, as the latter employs continuous variables, thereby enlarging its feasible region – and potentially its optimal set. Note that the moments of inertia are highly sensitive to positional variations: even small differences in fixture placement may result in significant changes in f_{iner}.

Among the CP solvers, Chuffed exhibits the most robust behavior on the more challenging instances. In particular, it proves optimality for the simple stair step, unlike Gecode and OR-Tools, and attains higher f_{dist} values than OR-Tools for both the simple stair step and the coffee table. When optimality cannot be proven within the time limit, OR-Tools generally returns solutions with lower objective values than Gecode and Chuffed. Being local-search based, Gecode+LNS cannot solve any of the problems to optimality. However, it still manages to find the optimal value in 3 cases out of 5 and, for the Dashboard problem, it is the approach achieving the best f_{iner} value.

Gurobi solves to optimality all the instances except the coffee table, where however it achieves the highest objective value. Apart from the solver capabilities, we argue that these results come from the continuous nature of the x- and y-variables.

Table 2 reports the solve times of the different solvers, expressed in seconds, under a time limit of 300 s. Among the CP solvers, Chuffed is able to prove optimality in all cases except for the coffee table workpiece. However, when both Chuffed and OR-Tools reach the optimal solution, OR-Tools is generally faster than Chuffed. Gecode and OR-Tools reach the time limit on the same workpieces, with OR-Tools being faster when it successfully solves the model. The Gecode+LNS approach always consumes the full time limit, even when attaining the same objective value as the other CP solvers. Although Gurobi generally produces superior objective values, for the stair step of the spiral staircase and dashboard workpieces it requires longer computational times than the CP solvers.

Table 2 Solve time in seconds to reach the best f_{dist} solution. Timeout (300 s) is indicated by "–".

Workpiece	Gecode	Gecode+LNS	Chuffed	OR-Tools	Gurobi
Stair step spiral staircase	1.207	–	0.254	0.090	3.895
Stair step simple staircase	–	–	2.184	–	0.293
Dashboard	0.061	–	0.113	0.045	0.205
Speaker	26.692	–	0.346	0.154	0.042
Coffee table	–	–	–	–	–

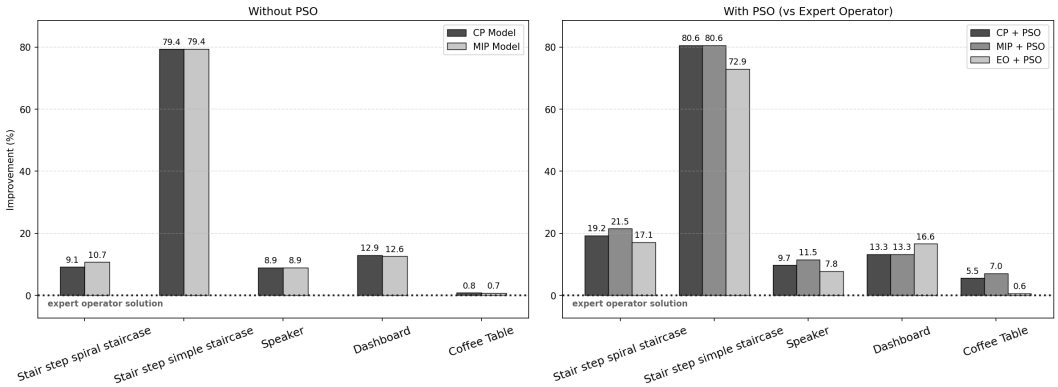


Figure 4 Percentage improvement relative to the solution provided by the expert operator. The left plot illustrates the improvements achieved by the CP and MIP models without PSO integration, whereas the right plot presents the improvements obtained by the CP and MIP models with PSO integration, alongside the improvement of the expert operator's solution when combined with PSO.

For each workpiece, the best CP and MIP solutions with respect to f_{iner} were used as initial solutions for the PSO algorithm. The performance obtained after the PSO refinement was then evaluated and compared. As a baseline, we also considered the solution obtained by initializing the PSO algorithm with the manual layout provided by an expert operator.

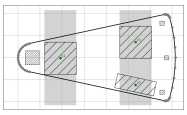
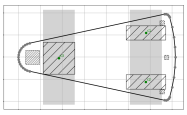
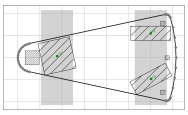
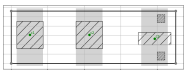
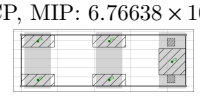
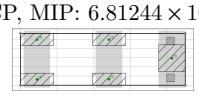
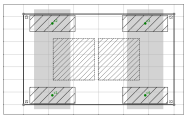
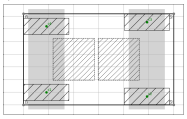
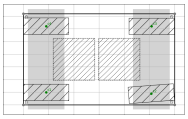
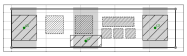
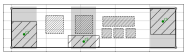
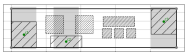
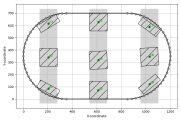
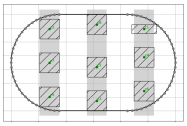
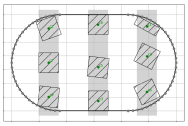
Figure 4 shows the percentage improvement in f_{iner} relative to the expert operator's baseline. The left plot reports the improvements achieved by the CP and MIP models without PSO, whereas the right plot shows the results after PSO integration, including the refinement of the expert solution.

The most significant improvement (79.4%) is observed for the simple stair step, where both CP and MIP models achieve identical gains over the expert configuration. For the speaker, CP and MIP provide equivalent improvements prior to refinement; however, after PSO integration, the MIP-based solution performs better due to a more effective rotation of the fixtures.

On the dashboard instance, the CP model slightly outperforms the MIP model. Nevertheless, the refinement of the expert solution through PSO achieves the highest improvement (16.6%), representing the only case in which the expert-based initialization combined with PSO surpasses the model-based approaches. This difference is likely attributable to the stochastic nature of the PSO algorithm rather than to a structurally superior starting layout, since the expert operator's initial solution is inferior to those proposed by both CP and MIP.

For the coffee table, the improvements obtained by CP and MIP alone are modest compared to the other instances. However, PSO integration increases the improvement to 5.5% for CP and 7.0% for MIP, confirming the benefit of the refinement phase for larger geometries.

Table 3 Summary of the best solutions, corresponding solution providers, and moment of inertia values for the different workpieces. The workpiece perimeter is shown with solid lines. Dashed lines indicate extrusion areas, fixture regions are marked with diagonal hatching, and supporting bars are shown as shaded rectangles.

Workpiece	Expert Operator (EO)	Best Solution without PSO	Best Solution with PSO
Stair step spiral staircase	1.96057×10^9 	MIP: 2.17089×10^9 	MIP: 2.33771×10^9 
Stair step simple staircase	3.77213×10^9 	CP, MIP: 6.76638×10^9 	CP, MIP: 6.81244×10^9 
Speaker	2.69490×10^9 	CP, MIP: 2.93433×10^9 	MIP: 2.95062×10^9 
Dashboard	6.29544×10^9 	CP: 7.10721×10^9 	EO: 7.12000×10^9 
Coffee table	2.17954×10^{10} 	MIP: 2.19472×10^{10} 	MIP: 2.30040×10^{10} 

Visual comparisons of the best solutions obtained for the CP and MIP models for the different workpieces, as well as the corresponding solutions integrated with PSO, are presented in Table 3. Note that the bars can be positioned beneath the area to be extruded, since the height of the fixtures prevents the cutting tools from coming into contact with them.

From a geometric perspective, avoiding alignment with symmetry axes contributes to improved stability, a behavior encouraged by the f_{iner} objective. For example, in the dashboard instance, PSO relocates the smallest fixture away from the central region while preserving safe distances from the larger fixture on the left. Similarly, for the simple stair step, the expert solution aligns fixtures along a symmetry axis, whereas the MIP and PSO-refined configurations distribute them along the perimeter, resulting in a substantial increase in the moment of inertia.

Rotation plays a decisive role in the spiral stair step and coffee table instances, where fixture reorientation improves adherence along the perimeter. However, rotation is not universally beneficial: in the spiral stair step, rotating the top-right fixture to mirror the bottom one would reduce the moment of inertia by decreasing their mutual distance. For the speaker instance, the limited geometry severely restricts the range of admissible rotations. Within these constraints, the MIP-based PSO refinement applies a slight rotation to the bottom-left fixture, yielding a marginal improvement, whereas the CP-based refinement rotates the top-right fixture by a larger angle, ultimately leading to a worse result, as illustrated in Figure 4.

For the coffee table, the MIP model outperforms the expert operator by employing larger fixtures to increase inertia. The selected fixture types are then preserved by PSO which refines their positions, mitigating symmetry alignment and improving perimeter adherence

through controlled rotations. Note that the horizontal and vertical safety distance constraints prevent the simultaneous use of nine large square fixtures; consequently, at least one fixture must be of the smaller rectangular type.

Overall, the best trade-off between computational time and objective value is achieved by integrating the MIP model with PSO. For large workpieces composed of polylines or arcs, the expert operator can still achieve highly competitive results; however, these solutions can be further improved when used as initial configurations for the PSO algorithm.

Although the MIP-based approach generally performs best, the CP model achieves results that are often close in quality. Moreover, CP may prove more effective in scenarios involving a larger number of fixture types and multiple extrusion areas. This is due to the stronger propagation capabilities of the `diffn` constraint and the fact that type and overlap decisions are modeled through discrete variables that cannot be relaxed, unlike the x and y variables in the MIP formulation.

8 Conclusions

This paper proposes a solution strategy for the fixture layout optimization problem in the wood industry, based on the integration of constraint solving with meta-heuristic algorithms.

A Constraint Programming (CP) model is developed to address the fixture layout problem for a given workpiece by determining both the location and the type of fixtures, while satisfying geometric, resource availability, and safety constraints. The performance of this model is then compared with that of a newly formulated Mixed-Integer Programming (MIP) model.

Neither the MIP nor the CP model explicitly accounts for fixture rotation angles among their decision variables. Allowing rotation angles that are not multiples of 90° significantly increases the number of feasible configurations, thereby increasing the computational complexity of the problem. For this reason, the best solutions obtained from the optimization models are used as initial solutions for a Particle Swarm Optimization (PSO) algorithm, which incorporates fixture rotation angles as decision variables.

The results demonstrate that both the CP and MIP models consistently produce solutions that outperform those provided by the expert operator. In addition, the application of PSO further improves the configurations obtained from the optimization models as well as the configuration proposed by the expert operator, leading to measurable increases in the objective value. Therefore, the proposed strategy provides expert operators with an automated approach for fixture layout generation and can also be employed to improve manually designed solutions.

Future research may focus on enhancing the performance of the integrated MIP-PSO and CP-PSO approaches for large and irregular workpieces by exploiting potential symmetries through problem decomposition. Additional directions include extending the models to handle multiple workpieces simultaneously placed on the machine worktable and investigating alternative meta-heuristic or refinement strategies.

References

1 Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019. doi:10.1145/3292500.3330701.

2 Nicolas Beldiceanu, Mats Carlsson, Sophie Demassey, and Thierry Petit. Global constraint catalogue: Past, present and future. *Constraints An Int. J.*, 12(1):21–62, 2007. doi:10.1007/S10601-006-9010-8.

3 Odone Belluzzi. *Scienza delle costruzioni*, volume 2. N. Zanichelli, 1961.

4 Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

5 P H Bischoff and S P Gross. Equivalent moment of inertia based on integration of curvature. *Journal of Composites for Construction*, 15(3):263–273, 2011.

6 Geoffrey Chu, Peter J. Stuckey, Andreas Schutt, Thorsten Ehlers, Gregory Gange, and Kevin Francis. Chuffed: A lazy clause generation solver. https://github.com/chuffed/chuffed, 2018. Accessed: 2025-05-27.

7 Harold Wilberforce Coultas. *Theory of Structures: A Textbook Covering the Syllabuses of the B. Sc.(Eng.) AM Inst. CE and AMI Mech. E. and AMI Struct. E. Examinations in this Subject*. Pitman, 1925.

8 Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, 186(2-4):311–338, 2000.

9 Google LLC. OR-Tools: Google's operations research tools. https://github.com/google/or-tools, 2024. Accessed: 2025-05-27.

10 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2025. URL: https://www.gurobi.com.

11 Russell Charles Hibbeler and Kiang-Hwee Tan. *Structural analysis*. Pearson Prentice Hall Upper Saddle River, NJ, 2006.

12 Michael R Jackson, Robert M Parkin, and Neil Brown. Waves on wood. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 216(4):475–497, 2002.

13 İ Kalkan. Deflection prediction for reinforced concrete beams through different effective moment of inertia expressions. *International Journal of Engineering Research and Development*, 5(1):1–1, 2013.

14 Mr Ninad K Kulkarni, Ms Sujata Patekar, Ms Trupti Bhoskar, Mr Omkar Kulkarni, GM Kakandikar, and VM Nandedkar. Particle swarm optimization applications to mechanical engineering-a review. *Materials Today: Proceedings*, 2(4-5):2631–2639, 2015.

15 Sven Leyffer, Jeff Linderoth, James Luedtke, Andrew Miller, and Todd Munson. Applications and algorithms for mixed integer nonlinear programming. In *Journal of Physics: Conference Series*, volume 180, page 012014. IOP Publishing, 2009.

16 Bing Li, Hongjian Yu, Xiaojun Yang, and Ying Hu. Variation analysis and robust fixture design of a flexible fixturing system for sheet metal assembly, 2010.

17 Bo Li and Shreyes N Melkote. Optimal fixture design accounting for the effect of workpiece dynamics. *The International Journal of Advanced Manufacturing Technology*, 18(10):701–707, 2001.

18 Cheng Liang and Xiaojian Liu. The research of collision detection algorithm based on separating axis theorem. *Int. J. Sci*, 2(10):110–114, 2015.

19 Y Gene Liao. A genetic algorithm-based fixture locating positions and clamping schemes optimization. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 217(8):1075–1083, 2003.

20 Changhui Liu, Ying Zheng, Jing Wang, Ke Jin, Jianbo Yu, and Jianfeng Liu. Fixture layout optimization for large thin-walled parts based on improved particle swarm optimization algorithm. *The International Journal of Advanced Manufacturing Technology*, 132(7):3579–3592, 2024.

21 DS Liu, Kay Chen Tan, Chi Keong Goh, and Weng Khuen Ho. On solving multiobjective bin packing problems using particle swarm optimization. In *2006 IEEE International Conference on Evolutionary Computation*, pages 2095–2102. IEEE, 2006.

22 Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc: Towards a standard cp modelling language. In *International Conference on Principles and Practice of Constraint Programming*, pages 529–543. Springer, 2007. doi:10.1007/978-3-540-74970-7_38.

23 KP Padmanaban, KP Arulshri, and G Prabhakaran. Machining fixture layout design using ant colony algorithm based continuous optimization method. *The International Journal of Advanced Manufacturing Technology*, 45(9):922–934, 2009.

24 David Pisinger and Stefan Ropke. Large neighborhood search. In *Handbook of metaheuristics*, pages 99–127. Springer, 2018.

25 Christian Schulte, Mikael Lagerkvist, and Guido Tack. Gecode. *Software download and online material at the website: http://www. gecode. org*, pages 11–13, 2006.

26 S Selvakumar, KP Arulshri, KP Padmanaban, and KSK Sasikumar. Design and optimization of machining fixture layout using ann and doe. *The International Journal of Advanced Manufacturing Technology*, 65(9):1573–1586, 2013.

27 Ying Sun, Wanyuan Shi, and Yuelin Gao. A particle swarm optimization algorithm based on an improved deb criterion for constrained optimization problems. *PeerJ Computer Science*, 8:e1178, 2022. doi:10.7717/PEERJ-CS.1178.

28 KA Sundararaman, KP Padmanaban, and M Sabareeswaran. Optimization of machining fixture layout using integrated response surface methodology and evolutionary techniques. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 230(13):2245–2259, 2016.

29 Juifen C Trappey and C Richard Liu. A literature survey of fixture design automation. *The International Journal of Advanced Manufacturing Technology*, 5(3):240–255, 1990.

30 Anna Vitali, Roberto Amadini, and Maurizio Gabbrielli. Fixture layout optimization in wood industry: A case study. In *Proceedings of the 27th International Symposium on Principles and Practice of Declarative Programming*, PPDP '25, New York, NY, USA, 2025. Association for Computing Machinery. doi:10.1145/3756907.3756914.

31 Z Wang, Y Yang, B Yang, and Y Kang. Optimal sheet metal fixture locating layout by combining radial basis function neural network and bat algorithm. adv mech eng 8 (12): 168781401668190, 2016.

32 Xiaoping Zhang, Wen Yu Yang, and Miao Li. Fixture layout and clamping force optimization for large-scale workpiece using augmented lagrangian method. *Applied Mechanics and Materials*, 29:560–565, 2010.