Bending Simulation Framework for Rapid Feasibility Checks of Sheet Metal Parts

Sergey E. Slyadnev[1]

[1]Quaoar Studio LLC, sergey@quaoar.pro

Abstract. Finding a possible bending sequence for a folded sheet metal part remains a challenging problem in CAD/NC integration field. Solving this problem calls for heuristic methods to prevent greedy search. In automated quotation systems, it is often enough to identify just any feasible sequence to validate the manufacturability of a part. If a feasible sequence cannot be found, the corresponding part is flagged for human review. If no manufacturability issues are detected, a part might be passed over for automatic quotation and preparation of technical drawings, thus saving the valuable engineer's time. As a result, bending feasibility analysis helps human operators to focus on potentially problematic parts while the processing of simpler parts can be fully automated.

This study presents a complete method for selecting a feasible bending sequence and validating it through a specifically designed simulation procedure. The initial CAD part is defined in the boundary representation (B-rep) form and is communicated as a STEP file to the algorithm. The part is subjected to an automatic feature recognition procedure in order to extract its bending properties. The geometry is then idealized into a hierarchically structured arrangement of flanges represented with sparse surface triangulations. The bending sequence search is guided by simulation. Our method aims at incrementally unfolding a part by choosing a tuple of "final" bends on each iteration. The selected "final" bends are unfolded, and the procedure repeats until the part becomes flat or no "final" bends can be selected. The simulation continues frame by frame, with collision detection performed at each intermediate folded state of the geometry. The collision detector examines self-intersections of the folded shape and finds possible clashes with the externally defined geometries of punch tools.

Keywords: sheet metal, bending sequence, unfolding, feature recognition
DOI: https://doi.org/10.14733/cadaps.2026.85-100

1 INTRODUCTION

Folded sheet metal parts (Fig. 1) are frequently communicated without features and require preprocessing for use in downstream operations, such as quotation and process planning. Even when a part originates from a feature-based modeler, the available design features could at times be unsuitable for subsequent processing. For instance, there are various ways to create a hole or a bend in a folded part without preserving the associated design feature in the history tree. Feature recognition seeks to extract information about flanges, cutting contours, and, most importantly, the properties of bends. With this data, it becomes possible to unfold the part and, in this way, obtain its dual flat pattern representation. The flat pattern shape of a folded sheet metal part is of key importance for both fabrication cost estimation and process planning.

A bent part may automatically be unfolded into a flat workpiece by utilizing information about bend angles, radii, and material thickness. Furthermore, to precisely account for material deformation at bends, existing unfolding algorithms usually employ the k-factor coefficient. Although these "one-step" algorithms might be capable of discovering potential design flaws (e.g., overlaps in the flat pattern or missing bend reliefs), they do not consider the dynamics of the process and the tools employed. Still, prior to actual fabrication, we must decide on tool selection and bending sequence, which are crucial machining factors. Bending sequence simulation allows tracing the folding process in dynamics with realistic geometry of punches and dies plugged in.

Finding a truly optimal (from a manufacturing effort perspective) bending sequence is a computationally prohibitive task. Given n bends in a part, the exhaustive search would require at least $\mathcal{O}(n!)$ operations (the number of possible permutations) to explore all possibilities. If we are given a bending sequence to test, we can quickly check it by direct verification in a simulator. However, such a sequence is normally unknown. Alternatively, we may prefer to quickly identify that the part in question is infeasible, i.e., there is no bending sequence to arrive at the given folded state of a workpiece. This study introduces an automatic feasibility verification procedure that expedites the assessment of part feasibility in "instant quotation" systems.

J.R. Duflou et al. [6] provided a comprehensive review of typical problems to be resolved by a process planning system for sheet metal bending. Their literature review touches on such important aspects of bending simulation as the behavior of physical materials, representation techniques for the simplified "foil" shapes, bend sequencing in the presence of constraints and collision detection. Another rarely discussed aspect of a planner is the idea of part type classification based on the final folded shape. For example, "box" shapes are an important special case for a simulator, as they require specific control over the geometry of punches. All in all, the study by J.R. Duflou may serve as a detailed introduction to CAPP methods in air bending.

Early researchers, such as M. Hoffmann et al. [7], stated that for better efficiency, simulations should be conducted on a properly constructed model, referred to as a "foil" shape in their study. In the work from 1997, S.K. Ong et al. [12] indicated that bend sequences can be examined in both "forward" and "reversed" time directions, indicating that the "reversed" search offers substantial benefits over the "forward" approach. A different research group that favored "reverse" simulation was L.J. de Vin et al. [4]. As in other studies, the authors attempted to reduce the search space by employing heuristic rules. Their PART-S system incorporated all essential components that form a comprehensive sheet metal Computer-Aided Process Planning (CAPP) tool, including feature recognition, unfolding, setup identification, and tooling.

J.C. Rico et al. [14] implemented a "forward" method of bending sequence analysis. They decomposed a sheet metal part into a set of simpler ("basic") shapes, assuming that all bends are aligned. This problem formulation is applicable to swivel bending machines but cannot be employed for press brakes. To generalize the approach, it would be necessary to switch from planar profiles of typical bending patterns to 3D basic shapes and enable spatial collision checks.

Zhang Lichao et al. [9] reported promising results in "reversed" planning with the graph search technique. Their method of bend sequencing is based on an improved A-star algorithm that incorporates "hard precedence" constraints. Among such constraints, they employ the position of flanges with respect to the back gauge, the location of the part's center of gravity, and compliance of the part dimensions with the specified tolerances. We deliberately avoid using graph search techniques, assuming them to be much more computationally expensive than a simple feasibility test employed in our study. At the same time, we acknowledge that utilization of heuristic-based graph search methods is a promising approach to deducing *optimal* sequences as opposed to "just any" sequences that are enough for feasibility analysis.

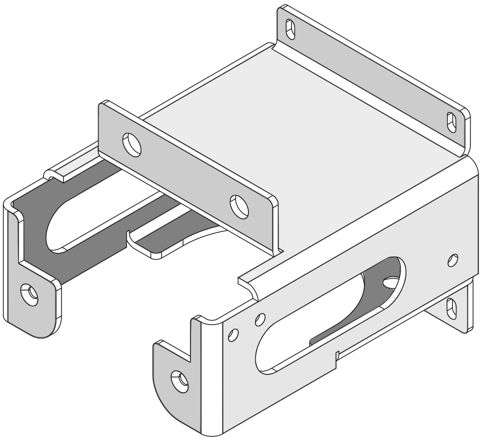


Figure 1: An example of a folded sheet metal part.

2 FEATURE RECOGNITION

Bending simulation uses sheet metal features such as the topology of the flanges, fold angles and directions, bend radii, etc. This information is traditionally extracted by a range of specific algorithms falling under the category of "feature recognition" as opposed to the "design by features" paradigm [16].

Our feature recognition procedure starts from an automatically selected "seed" face and propagates over the smoothly connected bend faces until a side of a sheet metal part ends up entirely visited. The key outcome of the recognition process is the attributed adjacency graph G [8] enriched with the information about features as specific "labels" associated with its nodes. Our recognition algorithm uses ideas similar to what has been published by Yang et al. [20], although it was developed independently.

In this section, we introduce the key aspects of our feature recognition algorithm. This algorithm plays a central role in the subsequent simulation: having a part feature recognized is a prerequisite for running the bending simulation operator U_s introduced in Section 3. The simulator uses the topology of flanges encoded by means of the unfolding tree. The recognition stage also determines the angles, directions, and radii of bends used for simulation.

2.1 Main Recognition Heuristics

The feature recognition algorithm employed in our study is based on the apparatus of the attributed adjacency graph (AAG) [19]. This data structure is available in the open-source feature recognition framework named Analysis Situs [18]. AAG is formally equivalent to a face adjacency graph (FAG) [3], which is a powerful tool facilitating feature recognition and unfolding algorithms for sheet metal and origami-like shapes [10]. The difference between AAG and FAG is the ability of the former to hold custom attributes that correspond to the recognized features and their properties in our study.

For the sake of clarity, we illustrate the main recognition steps by an example of a simple folded part depicted in Fig. 2. Following the metaphor by A. McKay and A. de Pennington [11], we aim at superimposing a proper "coat" for a sheet metal part that highlights its key machining features to be used down the road in bending simulation.

The algorithm begins with selecting the seed faces f_s and f_m that can be utilized to initiate propagation over the opposing sides of a sheet metal part. The fundamental assumption here is that a valid sheet metal part is a two-sided object, meaning that its inner and outer shells can be topologically separated. One simple

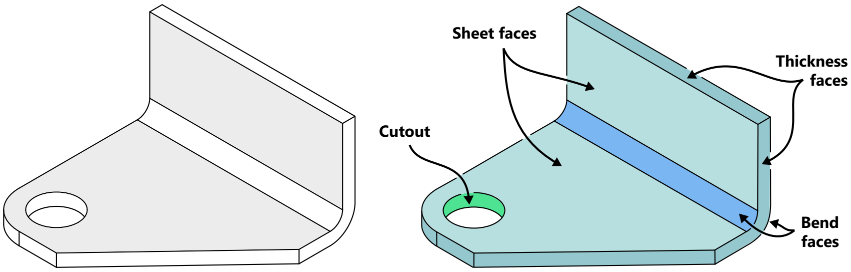


Figure 2: A simple folded sheet metal part before (left) and after feature recognition (right).

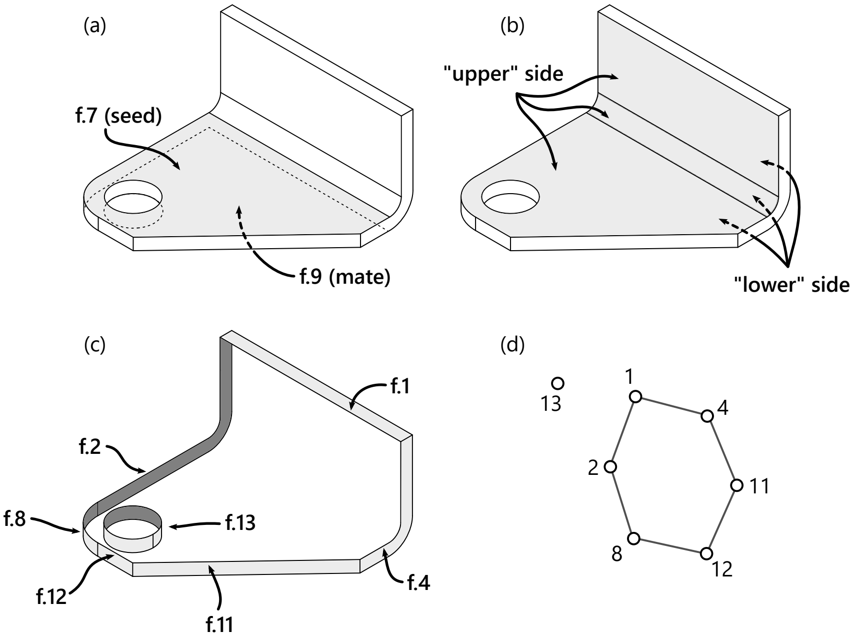


Figure 3: Feature recognition highlights: (a) seed face selection; (b) side propagation; (c) "thickness" faces; (d) subgraph $\tau \subset G$ of all "thickness" faces (two connected components in this case).

heuristic is that the planar face with the maximum bounded area can be chosen as a seed face. Another rule proposed by A. Salem et al. [15] defines the seed face as the "most adjacent" to other faces. Some other approaches to seed face selection can be employed for rolled parts or for specifically thick materials (e.g., exposed to waterjet cutting) but we leave them out of the present scope for brevity.

In Fig. 3, the seed face is $f_s = 7$. The opposite face $f_m = 9$ is searched by casting a ray inside the part's material (Fig. 3a). At this stage, the material thickness t is also computed. Starting from both seeds f_s and f_m the recognition process continues by recursive propagation (Algorithm 1).

The sheet faces are collected into face sets σ_1 and σ_2 for the "upper" and "lower" sides of the part correspondingly (Fig. 3b). The propagation process terminates when a face being iterated ends with a sharp dihedral angle ("cliff"). Such "cliffs" indicate that a cutting (or "thickness" face) has been reached.

Let the graph γ be a subgraph of the graph G: $\gamma \subseteq G$. The set difference of two graphs G and γ, denoted $G \setminus \gamma$, is the graph whose vertex (resp. edge) set is the difference of the vertex (resp. edge) sets of G and

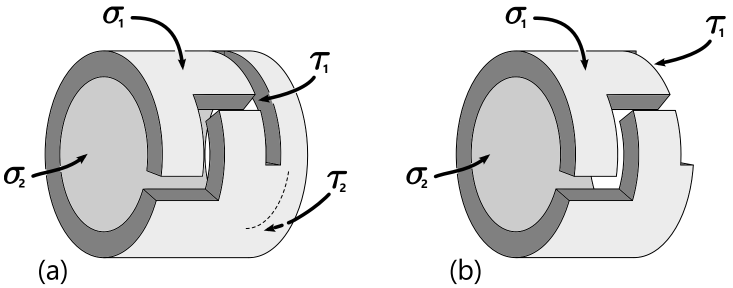


Figure 4: Round tube (a) and a rolled sheet metal part (b) with a cutting edge τ_1.

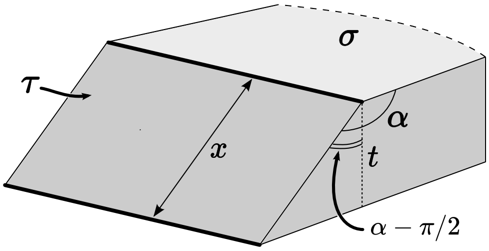


Figure 5: Thickness measurement over a slanted cut with a dihedral angle α.

of γ. The set union $\cup$ is defined similarly with no requirement for the operand graphs to have any vertices in common. We define "thickness candidate" faces as $\tau = G \setminus (\sigma_1 \cup \sigma_2)$.

Let $\mathtt{cc}(G)$ be the number of connected components in the graph G. For a valid sheet metal part, we require that $\mathtt{cc}(G \setminus \tau) \equiv \mathtt{cc}(\sigma_1 \cup \sigma_2) = 2$. This means that the shape is broken down into two unconnected sides when all "thickness" faces are eliminated.

Another set of heuristics is applied to the "thickness candidate" faces τ (Fig. 3c). The connected components $\tau_i (i = 1, ..., \mathtt{cc}(\tau))$ correspond to the cutting contours, including the outer and inner cuts. Let $V(G)$ be the vertex set of a graph G. We require that if $|V(\tau_i)| > 2$, then the degree of each vertex in τ_i is 2. This means that the thickness faces are arranged into simple loops like depicted in Fig. 3d.

We employ additional heuristics to distinguish bent (or rolled) parts from closed tubes. Let $\sigma = \sigma_1 \cup \sigma_2$ be the subgraph of all sheet faces in G. Let $\phi \subset \tau$ be the subgraph of all inner cuts over the part (holes, cutouts, other inside features). If $\exists \tau_i \in \tau$ such as $\forall \sigma_j \in \sigma : (N(\sigma_j) \setminus \phi) \subseteq \tau_i \ (j = 1, 2)$, then the part has a cutting edge, i.e., it can be unfolded. Here $N(u) = \{v \in V(G) : \{u, v\} \in E(G)\}$ is a neighborhood of a vertex u in the graph G, $E(G)$ is the edge set of the graph G. Fig 4 illustrates a couple of cases where this condition effectively distinguishes a closed tube (a) from a rolled part (b). A rolled part can be unfolded without any extra cuts, while a tube is closed (having genus 1 in this case) and cannot be unwrapped to a plane.

The `CheckThicknessHeuristics` procedure employed in Algorithm 1 accommodates different checks of a topological and geometric nature. One geometric check is the actual distance, denoted x, between the sheet faces from σ_1 and σ_2. With α as a dihedral angle between the thickness face in question and its neighboring sheet face, the distance x may be computed using the following formula with t as the material thickness (Fig. 5):

$$x = \frac{t}{\cos\left(\alpha - \frac{\pi}{2}\right)}$$

.

The actual thickness of a face can be measured in its (u, v) space (the parametric domain). If the measured value exceeds the theoretically predicted value x, then the corresponding connected component τ gets excluded from a possible "thickness candidate" set. Normally, at this stage, further recognition becomes pointless and the process stops.

Algorithm 1 Recognize folded part

```
procedure RECOGNIZE($G$)                         ▷ A part is defined with its AAG $G$.
    $f_s \leftarrow$ SELECTSEEDFACE($G$)                      ▷ E.g., max-area face.
    $f_m, t \leftarrow$ FINDMATEFACE($f_s, G$)   ▷ Done by ray casting, thickness $t$ gets computed.
    $\sigma_1 \leftarrow$ PROPAGATEOVERFLANGES($f_s, G$)
    $\sigma_2 \leftarrow$ PROPAGATEOVERFLANGES($f_m, G$)
    $\tau \leftarrow G \setminus (\sigma_1 \cup \sigma_2)$                  ▷ Thickness candidate faces.
    if CHECKTHICKNESSHEURISTICS($\tau, t, G$) then
        EXTRACTBENDPROPERTIES($G$)
        BUILDUNFOLDINGTREE($G$)
    end if
end procedure
```

2.2 Bend Properties

During sheet metal bending, the inner surface of the bend compresses while the outer one stretches (Fig. 6a). Therefore, somewhere in between, there is a zone separating the tension from the compression. In this zone (called "neutral axis"), the material gets zero distortion. The radius of the neutral axis is computed as $R_N = r + k \cdot t$, where r is the inner bend radius, k is the material-dependent k-factor and t is the sheet thickness. Once R_N is computed, bend allowance can be calculated as $R_N \cdot \alpha$, where α is the bend angle. It is the job of the feature recognizer to extract t, α, r and some other properties of a bend, including its length, direction of folding (w.r.t. the reference plane f_s), groups of physically constrained bends, etc.

The concept of a "border trihedron" is employed to calculate elementary bend properties (Fig. 6b). This trihedron is defined as a local coordinate frame that is attached to an edge where a bend meets a flange. The axes are labeled according to a particular convention. The $\mathbf{V}_x$ axis is directed along the bend's edge. The $\mathbf{V}_y$ axis points inside the face. The $\mathbf{V}_z$ axis is orthogonal to $\mathbf{V}_x$ and $\mathbf{V}_y$.

With a couple of border trihedrons computed, the bend angle is derived as the angle between $\mathbf{V}_y$ axes. Bend direction is determined by the convexity of the bend angle (concave for `UP` and convex for `DOWN`).

2.3 Bend Grouping

The equality of the following properties is verified when confirming multiple bend features to be performed in a single folding operation: angle, axis, inner radius, and direction (`UP` or `DOWN`). If these properties are identical, it is possible to incorporate the corresponding bend features into a single bend line, thereby reducing the manufacturing effort. The final decision to combine bend features together is, of course, made on the shop floor. However, the software can provide an upper estimate of the number of folding operations needed.

The property-based classification does not take into account the situation where completely separate flanges support the bends being grouped. For instance, bends 8 and 9 in Fig. 8 could have been grouped due to their alignment in the folded state of geometry. Moreover, the corresponding bend lines are aligned in the flat pattern. However, the alignment property may be disregarded depending on the applied bending sequence (e.g., if the 5th bend in Fig. 8 is done first). Consequently, it is impossible to determine the bend grouping of the part based solely on its geometric properties in either the flat or the folded state.

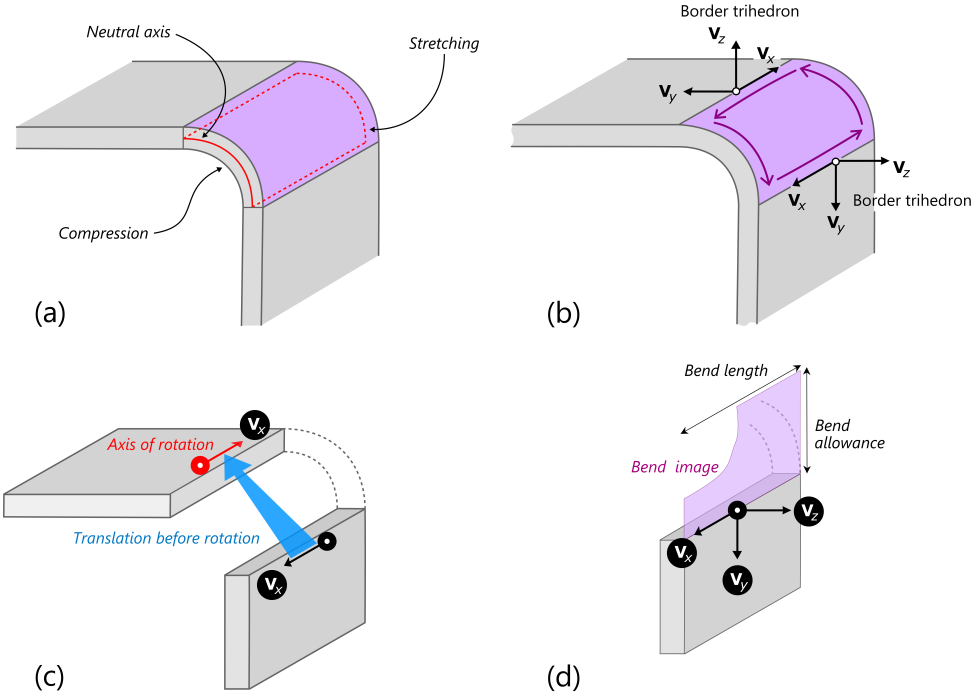


Figure 6: To bend properties extraction.

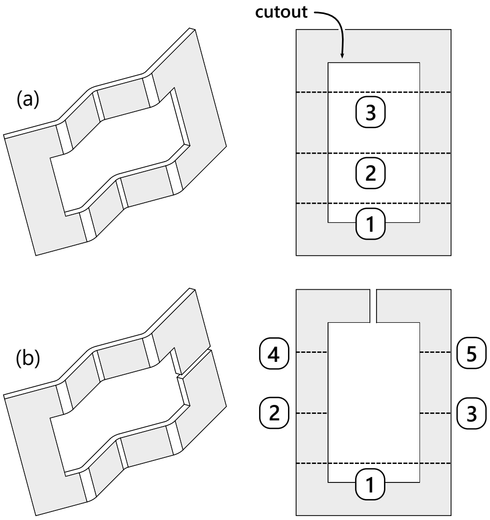


Figure 7: Bend grouping depends on topological constraints represented with inner loops in a flat pattern.

A closed contour of cutting faces (a cutout) is a loop that corresponds to a circuit of the neighboring sheet faces. It appears that cutouts are not merely geometric features: they also organize the topology of a part at a local scale. This allows us to employ cutouts for the bend grouping logic. In Fig. 7a, bends 1, 2, and 3 are unable to move freely as long as they are constrained by the corresponding cutout (a topologically closed

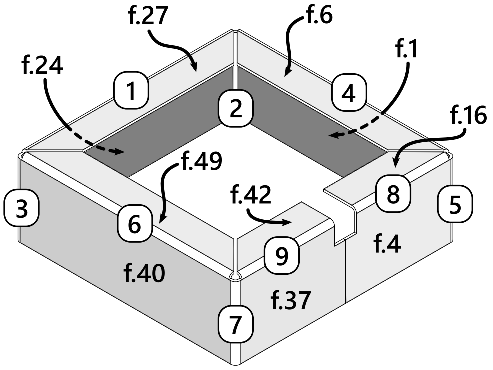


Figure 8: An example of an "infeasible" folded sheet metal part with all bends and flange faces numbered.

loop). However, this constraint is eliminated when the loop is cut off (Fig. 7b). In production, it may still be favorable to constrain the bends 1-3 and 4-5; however, this is not mandatory.

2.4 Unfolding Tree

Once all features are extracted, the properties of bends also become explicit, making it possible to unfold a part. Unfolding is used to prepare the cutting contours of the blank sheet, so the unfolding algorithm has to maintain high accuracy. The exact unfolding algorithm operates in a "one-shot" manner, calculating the flattening transformations without simulating the folding process. This operator is denoted $\mathbb{U}$ in what follows.

The flat pattern's plane is defined by the reference face f_s, which is the starting point for unfolding. The algorithm recursively visits all flanges accessible through the bend faces from this face. This propagation is performed for a single side of a sheet metal body. The unfolding tree data structure is being constructed together with the iteration procedure. The arcs represent the predecessor-successor relationship, while the nodes correspond to the flange faces. The root node is determined by the initial reference face f_s.

The nodes of the unfolding tree hold supplementary attributes, which aid in the formation of a flat pattern shape at the end. For each node, a list of references to the corresponding bend faces is provided (a flange may be connected through multiple bend faces). Next, we calculate a flattening transformation matrix for each child node in relation to its parent node. In order to arrive at the ultimate location of a flange in the reference plane, it is necessary to accumulate the transformations of all nodes along a path from this node to the root. A transformation matrix for a single flange is computed as a product of elementary transforms:

$$\mathtt{T} = \mathtt{T}_{BA} \cdot \mathtt{R} \cdot \mathtt{T}_{R}$$

Therefore, the flattening transformation $\mathtt{T}$ is a composite of three transformations: $\mathtt{T}_R$, $\mathtt{R}$, and $\mathtt{T}_{BA}$. $\mathtt{T}_R$ is a translation to align the edges of the flanges (Fig. 6c). The rotation by the bend angle is encoded in the $\mathtt{R}$ matrix, while the bend allowance translation is denoted by $\mathtt{T}_{BA}$ (Fig. 6d). The $\mathtt{T}_{BA}$ transformation is influenced by the k-factor, which is defined externally based on the material.

With the unfolding tree in place, we acquire everything that is required to construct the geometric representation of a flat pattern. All flanges are paved to the reference plane in accordance with the order of the unfolding tree and the encoded transformations. Gaps are left between the flanges during the paving process to accommodate the unrolled bends. The geometry of a flange is essentially a repositioned version of the original face without distortions. Unwrapping bend features is more difficult. We employ a method that involves unwrapping each folded trimming curve into 2D using a polynomial approximation. This way, the gaps between the flattened flanges are filled with the unfolded images of the bends' trimming contours.

3 BENDING SIMULATION

3.1 Searching for Final Bends

Let us consider a sheet metal part depicted in Fig. 8. This part would not trigger any feasibility checks if unfolding transformations are computed w.r.t. the U operator (i.e., "one-shot" unfolder). However, if the geometry of punches is taken into account, there is no way to bend this part completely on a press brake machine because of collisions with tools.

Therefore, we have to introduce a simulation-driven unfolding operator $\mathrm{U}_s[S, B, \alpha]$ that would capture the manufacturability issues undetected by U. Here B is a set of all bends and α is the angular increment for collision frames. The simulation model S is synthesized from the initial boundary representation of a folded part and has the topology determined by the unfolding tree. The ultimate simulation framework should employ both U and U_s to extract as much information for manufacturing as possible.

For a part with n bends, the number of possible bend sequences N equals the number of permutations, i.e., $N = n!$. All intermediate folded states of geometry can be represented with a graph model as depicted in Fig. 9. Iterating all possible paths in this graph is a computationally prohibitive task, especially taking into account that each folded state of geometry needs to be checked for collisions.

Algorithm 2 Find bending sequence

```
procedure FindSequence($S, B, \alpha$)                    ▷ Initial state $S$ is with all flanges folded.
    stop ← false                                          ▷ Prepare iterations.
    sequence ← $\emptyset$                                ▷ Collected sequence.
    processed ← $\emptyset$                               ▷ Visited bends.
    repeat
        $\Gamma \leftarrow \emptyset$                     ▷ All bends which are feasible on this iteration.
        for $b \subset B$ do                              ▷ Let's collect all feasible bends for the current state of $S$.
            if $b \in$ processed then
                continue
            end if
            if IsBendFeasible($b$) then
                $\Gamma \leftarrow b$
                processed ← $b$
            end if
        end for
        if $\Gamma = \emptyset$ then                      ▷ No feasible bend remains.
            stop ← true
            continue
        end if
        $S \leftarrow$ Unfold($\Gamma$, $\alpha$)         ▷ Unfold feasible bends.
        sequence ← $\Gamma$
    until !stop
    return sequence
end procedure
```

J.R. Duflou [5] proposed problem-size reduction methods to "bring automatic manufacturability verification a few steps closer to reality." According to him, a common approach in a traditional exhaustive search strategy is to apply a backwards unfolding check that starts by identifying bends that can be performed as final ones in the process plan. For the "infeasible" part depicted in Fig. 8, it requires only n iterations to conclude that

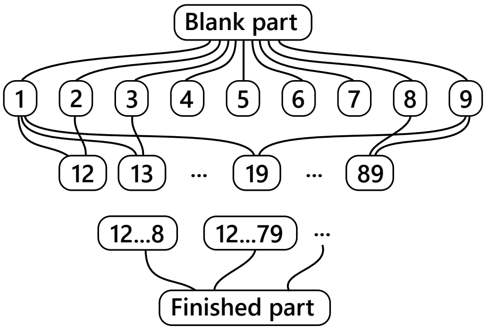


Figure 9: A graph representation of transitions between the folded states of geometry. Each node encodes a sequence of bend IDs that bring a part to a specific intermediate folded state.

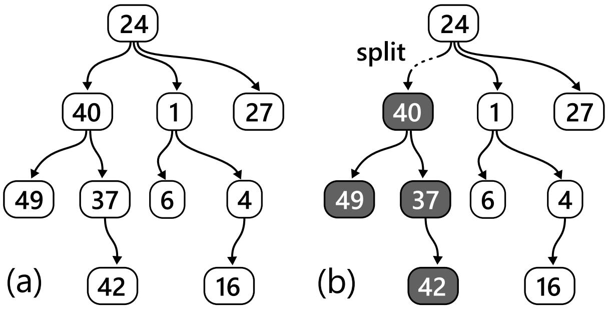


Figure 10: The unfolding tree of the sheet metal part depicted in Fig. 8: (a) is the initial tree with face 24 as a base one; (b) is a topological split over a bend line between faces 40 and 24.

the part is infeasible because no bend can be regarded as a final one.

Algorithm 2 progressively identifies all bends that can be regarded as "last" ones to obtain the next intermediate folded state of geometry $S := S$. All feasible bends are reported in a single group and unfolded to generate a new folded state. The process repeats until the part gets completely unfolded or there remains no feasible bend to report. To conclude if the part is feasible, it is enough to ensure that all bends were listed in the collected groups. In the worst case, Algorithm 2 hunts down one "final" bend at each iteration, so it has to check $(n) + (n-1) + (n-2) + ... + 1 = \frac{n^2-n}{2}$ bends that yield $\mathcal{O}(n^2)$ complexity as the upper bound.

Algorithm 2 aggregates not individual bend lines but groups $\Gamma_j = \{b_{j_1}, b_{j_2}, ...\}$ of bend lines considered "final" at each iteration. To determine the feasibility of a bend b_i inside its group, it is checked in isolation, disregarding the movement of other flanges caused by bends $b_k, k \neq i$. This simplification enables rapid iteration throughout the combinatorial space of sequences, although it may occasionally result in false positives. More exhaustive methods of exploring the search space, such as those reported by M. Shpitalni and D. Saddan [17] may overcome the issue of false positives but will significantly increase the computational complexity of the approach. Since we are interested in "instant" manufacturability checks, spending hundreds and even tens of seconds back-tracing the configuration graph (Fig. 9) is not an option.

3.2 Simulation Model

The simulation model S employed in collision testing must allow for quick isolation of "left" and "right" flanges relative to the bend line where folding happens (Fig. 11). Due to the need to verify numerous folded

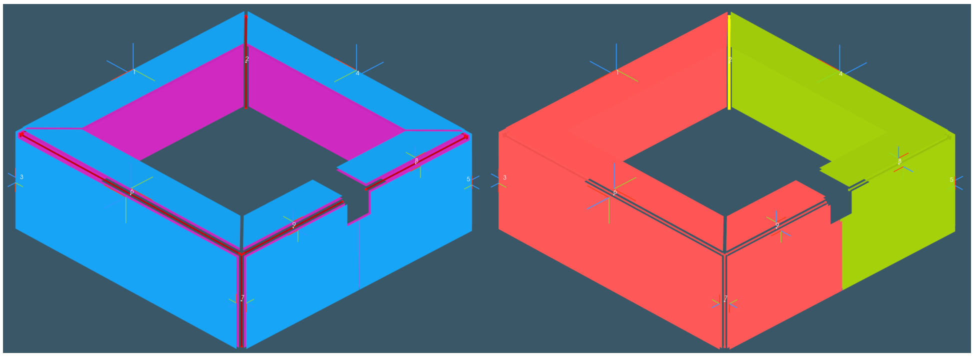


Figure 11: The initial folded state of the simulation model (on the left) and its topological split (see Fig. 10b) over the bend 2 (on the right).

states during simulation, it is essential that the generation of each intermediate folded state is done with the highest computational efficiency possible. For collision testing, sparse mesh representations are favored in the simulation model. The simulation model is synthesized from the topology of the unfolding tree and the boundary representation of the part.

Fig. 10 illustrates the unfolding tree for the "infeasible" part depicted in Fig. 8. The numerical identifiers of the graph nodes are equal to the indices of the corresponding CAD faces assigned by a topological iterator. All bend faces are eliminated from the graph, although their indices are stored in the graph edges.

Some elements of the bending simulation framework are depicted in Fig. 12. The simulation model allows for folding and unfolding with respect to the specific bend line (Fig. 12a,b). Each pair of flanges gets an associated hierarchy of collision boxes, making it possible to identify self-collisions over the part (Fig. 12c). If punch tools are loaded, then their presence is also captured by the corresponding collision boxes (Fig. 12d).

D. Raj Prasanth and M.S. Shunmugam [13] empirically demonstrated that efficient collision detection for the simulation of folding may be achieved by hierarchies of axis-aligned bounding boxes (AABB). AABB-based collision detection generally outperforms that of tighter volume decompositions utilizing oriented bounding boxes (OBB). Consequently, our methodology utilizes AABB-based bounding volume hierarchies with the surface area heuristic (SAH) as a criterion for volume partitioning.

3.3 Punches

Bending operations are executed using various tools and holders, whereas tools consist of dies and punches of different shapes and lengths. Our goal is to avoid interferences between the workpiece, the tool, and the machine. U. Alva and S.K. Gupta [2] describe how to build parametric models of punches for bending multiple parts in a single setup. They enumerate the following criteria for a punch shape:

- *Compatibility with bend geometry.* The punch radius should be compatible with the inside radius of the bend. The angle of the punch should be smaller than the bend angle.
- *Interference.* The shape of the punch should be such that there is no part-tool interference between the punch and any intermediate workpiece shape.
- *Punch strength.* The punch should be strong enough to withstand the bending forces.

The "instant" simulation logic can incorporate the first two criteria due to their inherently geometric nature. We also assume that a manufacturer has various existing punches (e.g., Fig. 13) and will not design a new tool to fabricate another sheet metal part coming in. That is a reasonable assumption for mass-market

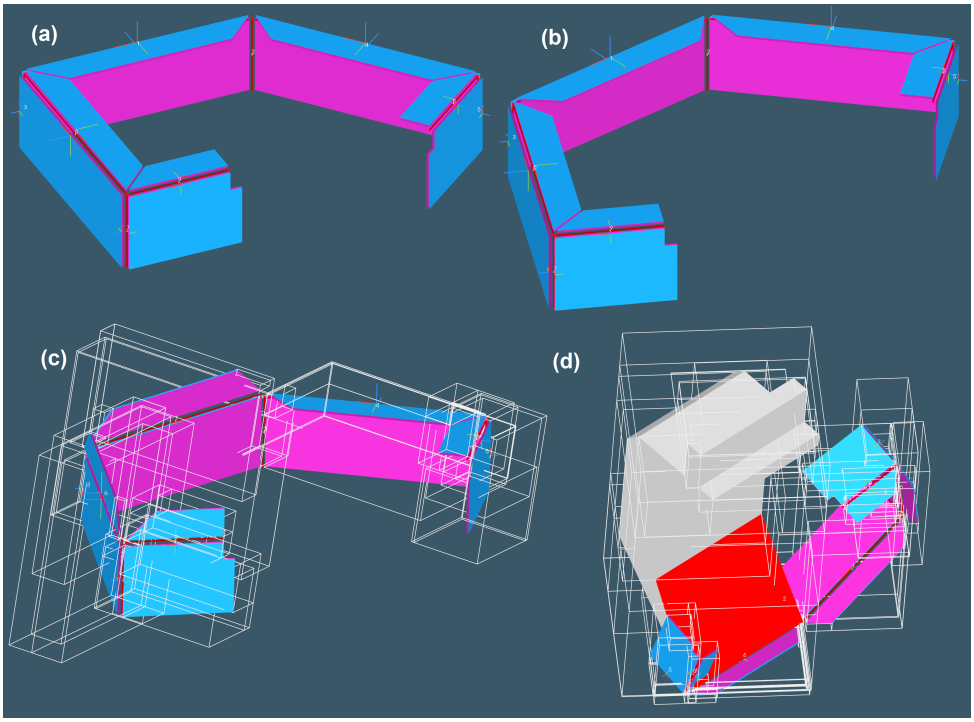


Figure 12: Some intermediate stages of bending simulation with collision checks (c,d) and without such (a,b).

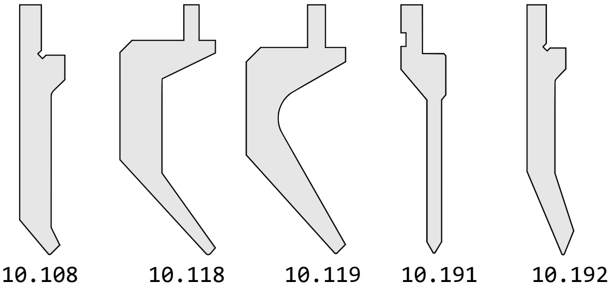


Figure 13: Some typical punch profiles used in the simulation are defined as non-parametric planar contours. The codes of the tools are taken from the online catalog by Artizono Machinery [1]. The tools are dynamically extruded by the corresponding bend lengths for collision testing.

products. The simulation algorithm is data-driven in the sense that the system takes punch definitions from an external database (e.g., such as [1]). As a result, a technologist may add the available tools to the simulator and this way improve its accuracy.

The available punch profiles are dynamically extruded into solid models whose meshes are used in the collision tests. Each profile is also mirrored to emulate flipping the part in the press brake machine. For a bend to be feasible, it is enough to find at least one tool that does not yield any collisions with the part being folded. The number of employed tools and their lengths give information about the required setup and may also be used for automated quotation. Some parts may appear infeasible only with respect to the plugged

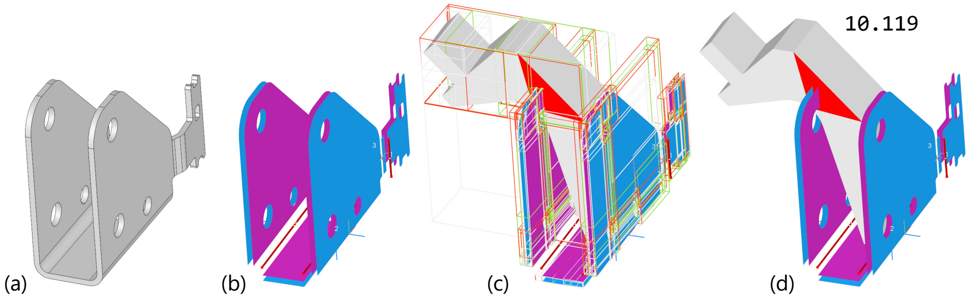


Figure 14: The simulation process in stages: (a) the initial model; (b) the simulation ("foil") model; (c) a punch tool and the simulation model with their collision boxes; (d) the collision zone emphasized.

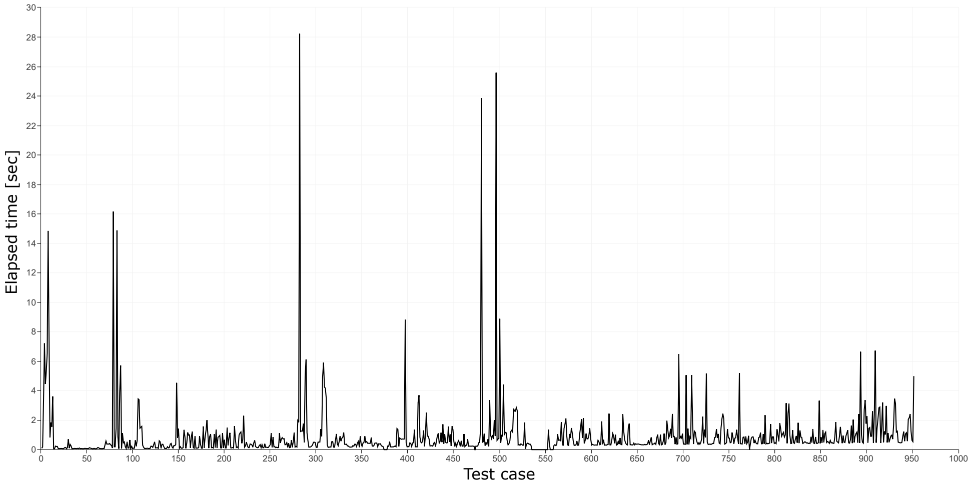


Figure 15: The simulation timing was measured over 950 test cases of realistic sheet metal parts.

tools, while changing the geometry of punches may allow getting rid of the detected collisions (Fig. 14).

4 CONCLUSIONS AND FUTURE WORK

Incorporating the bending sequence finder into a widely used Manufacturing-as-a-Service (MaaS) platform demonstrated that the proposed method can swiftly detect bending issues, offering "instant" feedback for real-life sheet metal parts with numerous bend features.

Typically, the simulator does not need to show high precision, and the material distortion on folding can be idealized by rotating the "left" and "right" sides of a model around a bending axis. Nevertheless, in specific cases, false-positive collisions may be reported due to the algorithm's inability to account for k-factor values at bends. This behavior can be improved by substituting the rotation transformation with a more complex movement of the flanges on folding.

The proposed approach got a thorough evaluation over about one thousand tests. The algorithm's main performance bottleneck (Fig. 15) is discovered for perforated sheet metal parts, as their simulation models comprise hundreds to thousands of triangles. These problems can be resolved through the appropriate defeaturing of the flanges, as the presence of small holes and cutouts has no impact on the selection of a bending sequence. At the same time, the defeaturing of isolated features can be done rapidly by eliminating the corresponding B-rep faces and contours from the part's topology.

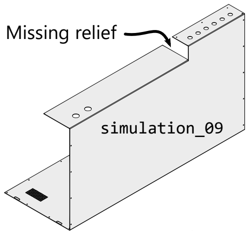
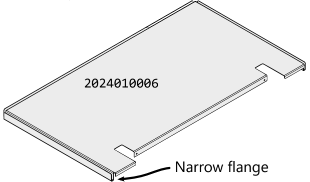
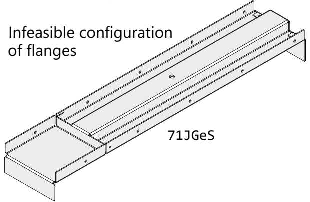
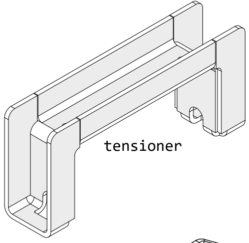
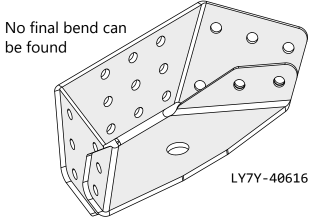
Part	N	F_a	F_s	$\{\Gamma\}$	Time [s]
	3	0	0	2-3	0.5
	11	0	0	1-2-3	0.48
	15	0	0	$\emptyset$	1.2
	10	0	0	$\emptyset$	0.4
	5	1	0	$\emptyset$	0.24

Table 1: Some of the detected bending feasibility issues. The measured time is related to simulation alone, excluding feature recognition. The tests were conducted on a personal laptop with `Intel(R) Core(TM) i7-10870H CPU @ 2.20GHz, 32GiB RAM, Windows 10 x64`.

Table 1 enumerates some of the feasibility issues detected by the algorithm. Let $F_a = \{0, 1\}$ be the actual feasibility indicator (0 for infeasible, 1 for feasible) and $F_s = \{0, 1\}$ be the simulated feasibility indicator. Then the accuracy of the simulation can be measured as $\rho = \frac{\sum_i \rho_i}{M}$, where M is the number of test cases and ρ_i is a per-case accuracy:

$$\rho_i = \begin{cases} 1 & F_s = F_a \\ 0 & \text{otherwise} \end{cases}$$

The presented simulation framework comprises several highly efficient subroutines: feature recognition, topology-based separation of the simulation model's flanges, and BVH-based collision detection. While our focus was on the folded parts fabricated with press brakes, the same approaches are reusable (with little adaptation) for simulating swivel bending machines.

In the scope of recognition, the employed heuristics for the "thickness candidate" faces τ are sometimes too strict. The requirement of degree 2 for all "thickness" faces prohibits any machined edges over the cutting contour of a part. As a result, the presence of fillets and chamfers in a folded sheet metal prevents it from being recognized completely. The possible approaches to generalize sheet metal feature recognition for parts with machined edges are yet to be discovered.

Another limitation of the approach is the impossibility of handling flat hems (i.e., the bends that are folded by 180 degrees). The presence of flat hems is detected specifically by the feature recognizer, so these features may potentially be excluded from the simulation logic.

Finally, modern MaaS platforms tend to provide a highly interactive user interface for reporting the detected design issues. As the bending simulation algorithm operates with realistically looking 3D models, all intermediate frames may be recorded in an animation file (e.g., of the commonly used glTF format) and provided to the users for visual inspection of the folding process.

ORCID

Sergey Slyadnev, http://orcid.org/0009-0009-0716-7632

REFERENCES

[1] Artizono Machinery. Press Brake Tooling. https://quaoar.su/files/papers/sheet_metal/Press-Brake-Toolings-Catalog.pdf.

[2] Alva, U.; Gupta, S.K.: Automated design of sheet metal punches for bending multiple parts in a single setup. Robotics and Computer-Integrated Manufacturing, 17(1), 33–47, 2001. ISSN 0736-5845. http://doi.org/10.1016/S0736-5845(00)00035-1. 10th International Conference on Flexible Automation and Intelligent.

[3] Ansaldi, S.; De Floriani, L.; Falcidieno, B.: Geometric modeling of solid objects by using a face adjacency graph representation. In Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '85, 131–139. Association for Computing Machinery, New York, NY, USA, 1985. ISBN 0897911660. http://doi.org/10.1145/325334.325218.

[4] de Vin, L.; de Vries, J.; Streppel, A.; Klaassen, E.; Kals, H.: The generation of bending sequences in a capp system for sheet-metal components. Journal of Materials Processing Technology, 41(3), 331–339, 1994. ISSN 0924-0136. http://doi.org/10.1016/0924-0136(94)90169-4.

[5] Duflou, J.: Design verification for bent sheet metal parts: A graphs approach. International Transactions in Operational Research, 4(1), 67–73, 1997. http://doi.org/10.1111/j.1475-3995.1997.tb00063.x.

[6] Duflou, J.R.; Vancza, J.; Aerens, R.: Computer aided process planning for sheet metal bending: A state of the art. Computers in Industry, 56(7), 747–771, 2005. ISSN 0166-3615. http://doi.org/10.1016/j.compind.2005.04.001.

[7] Hoffmann, M.; Geißler, U.; Geiger, M.: Computer-aided generation of bending sequences for die-bending machines. Journal of Materials Processing Technology, 30(1), 1–12, 1992. ISSN 0924-0136. http://doi.org/10.1016/0924-0136(92)90035-Q.

[8] Joshi, S.; Chang, T.: Graph-based heuristics for recognition of machined features from a 3d solid model. Computer-Aided Design, 20(2), 58–66, 1988. ISSN 0010-4485. http://doi.org/10.1016/0010-4485(88)90050-4.

[9] Lichao, Z.; Yi, Z.; Qiang, Z.; Fafu, H.: Robust sheet metal bend sequencing method based on A-star algorithm. In 2011 IEEE International Conference on Computer Science and Automation Engineering, vol. 2, 711–715, 2011. http://doi.org/10.1109/CSAE.2011.5952603.

[10] Liu, W.; Tai, K.: Optimal design of flat patterns for 3d folded structures by unfolding with topological validation. Computer-Aided Design, 39(10), 898–913, 2007. ISSN 0010-4485. http://doi.org/10.1016/j.cad.2007.05.015.

[11] McKay, A.; de Pennington, A.: Shape embedding: A means of superimposing alternative design descriptions on shape models. Computer-Aided Design, 152, 103366, 2022. ISSN 0010-4485. http://doi.org/10.1016/j.cad.2022.103366.

[12] Ong, S.; De Vin, L.; Nee, A.; Kals, H.: Fuzzy set theory applied to bend sequencing for sheet metal bending. Journal of Materials Processing Technology, 69(1), 29–36, 1997. ISSN 0924-0136. http://doi.org/10.1016/S0924-0136(96)00035-0.

[13] Prasanth, D.R.; Shunmugam, M.S.: Collision detection during planning for sheet metal bending by bounding volume hierarchy approaches. International Journal of Computer Integrated Manufacturing, 31(9), 893–906, 2018. http://doi.org/10.1080/0951192X.2018.1466394.

[14] Rico, J.C.; Gonzalez, J.M.; Mateos, S.; Cuesta, E.; Valino, G.: Automatic determination of bending sequences for sheet metal parts with parallel bends. International Journal of Production Research, 41(14), 3273–3299, 2003. http://doi.org/10.1080/0020754031000095158.

[15] Salem, A.; Abdelmaguid, T.F.; Wifi, A.S.; Elmokadem, A.: Towards an efficient process planning of the v-bending process: an enhanced automated feature recognition system. The International Journal of Advanced Manufacturing Technology, 91, 4163 – 4181, 2017. http://doi.org/10.1007/s00170-017-0104-9.

[16] Shah, J.J.; Mantyla, M.: Parametric and Feature Based CAD/CAM: Concepts, Techniques, and Applications. John Wiley & Sons, Inc., USA, 1st ed., 1995. ISBN 0471002143.

[17] Shpitalni, M.; Saddan, D.: Automatic determination of bending sequence in sheet metal products. CIRP Annals, 43(1), 23–26, 1994. ISSN 0007-8506. http://doi.org/10.1016/S0007-8506(07)62155-6.

[18] Slyadnev, S.; Malyshev, A.; Turlapov, V.: CAD model inspection utility and prototyping framework based on OpenCascade. In GraphiCon 2017, 323–327. Perm, Russia, 2017.

[19] Slyadnev, S.; Malyshev, A.; Turlapov, V.: On the role of graph theory apparatus in a CAD modeling kernel. In 30th International Conference on Computer Graphics and Machine Vision, Saint Petersburg, 2020. http://doi.org/10.51130/graphicon-2020-2-3-70.

[20] Yang, Y.; Hinduja, S.; Owodunni, O.O.; Heinemann, R.: Recognition of features in sheet metal parts manufactured using progressive dies. Computer-Aided Design, 134, 102991, 2021. ISSN 0010-4485. http://doi.org/10.1016/j.cad.2021.102991.