Title:
Bending Simulation Framework for Rapid Feasibility Checks of Sheet Metal Parts

Authors:
Sergey E. Slyadnev, sergey@quaoar.pro, Quaoar Studio LLC

Keywords:
Sheet Metal, Bending Sequence, Unfolding, Feature Recognition

DOI: 10.14733/cadconfP.2025.255-260

Introduction:

Finding a possible bend sequence for a folded sheet metal part remains a challenging problem in the 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 identified, the corresponding part is flagged for human review. If no manufacturability issues are detected, a part might be passed over for 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 and highlight possible fabrication difficulties, while the processing of simpler parts can be fully automated.

Early researchers, such as M. Hoffmann et al. [3], 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. [6] 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. L.J. de Vin et al. [1] attempted to reduce the search space by employing heuristic rules in "reversed" simulation. J.C. Rico et al. [8] 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. Zhang Lichao et al. [5] 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.

This study presents a simple 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, which is communicated as a STEP file being the input of the discussed algorithm. The part is then idealized into a hierarchically structured arrangement of flanges represented with sparse surface triangulations.

The bend 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 checks for self-intersections of the folded geometry and examines possible collisions with the geometries of punches. The simulation algorithm is data-driven in the sense that the system takes punch definitions from an external database. As a result, a technologist may add the available tools to the simulator and this way improve its accuracy.

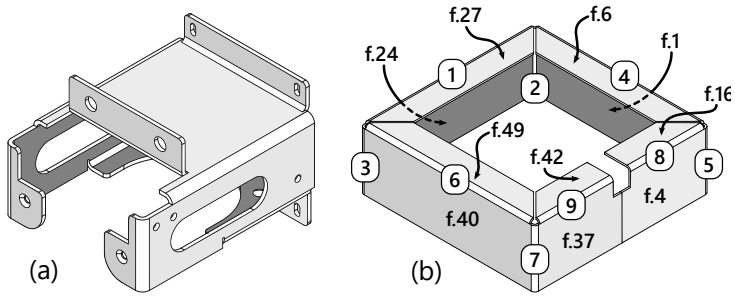

Fig. 1: An example of a folded sheet metal part (a). An example of an "infeasible" folded sheet metal part with all bends and flange faces numbered (b).

Feature Recognition:

Folded sheet metal parts (Fig. 1a) are often communicated without features and therefore require pre-processing for recovering the information about flanges, bends, and, more importantly, bend properties.

The employed feature recognition approach 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. One simple 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. [9] defines the seed face as the "most adjacent" to other faces. The key outcome of the recognition process is the attributed adjacency graph G [4] 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. [11], although it was developed independently.

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 it has to maintain high accuracy. The exact unfolding algorithm operates in a "one-shot" manner, meaning the flattening transformations are calculated without simulating the folding process. This operator is called U in what follows.

Bending Sequence Simulation Principles:

Let us consider a sheet metal part depicted in Fig. 1b. 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 graph. 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. 2a. 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.

R. Duflou [2] 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

Algorithm 1 Find bending sequence

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

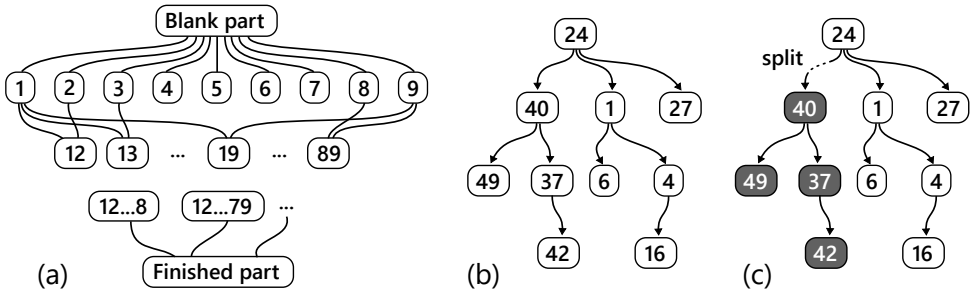

Fig. 2: A graph representation of transitions between the folded states of geometry (a). The unfolding tree of the sheet metal part depicted in Fig. 1b: (b) is the initial tree with face 24 as a base one; (c) is a topological split over a bend line between faces 40 and 24.

performed as final ones in the process plan. For the "infeasible" part depicted in Fig. 1b, it requires only n iterations to conclude that the part is infeasible because no bend can be regarded as a final one.

The Algorithm 1 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, the Algorithm 1 hunts down one "final"

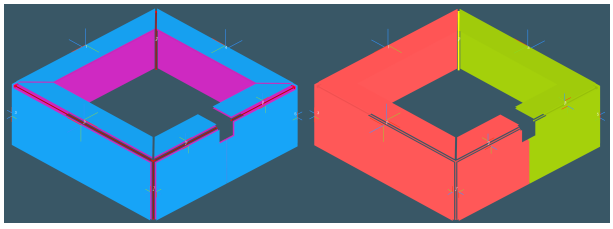

Fig. 3: The initial folded state of the simulation model (on the left) and its topological split (see Fig. 2c) over the bend 2 (on the right).

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.

The Algorithm 1 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 [10] 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. 2a) is not an option.

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. 3). Due to the need to verify numerous folded states during simulation, it is essential that the generation of each intermediate folded state be 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. 2b illustrates the unfolding tree for the "infeasible" part depicted in Fig. 1b. 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. 4. The simulation model allows for folding and unfolding with respect to the specific bend line (Fig. 4a,b). Each pair of flanges gets an associated hierarchy of collision boxes, making it possible to identify self-collisions over the part (Fig. 4c). If punch tools are loaded, then their presence is also captured by the corresponding collision boxes (Fig. 4d).

D. Raj Prasanth and M.S. Shunmugam [7] 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.

Conclusions:

Incorporating the bending sequence detector into a widely used Manufacturing-as-a-Service (MaaS) plat-

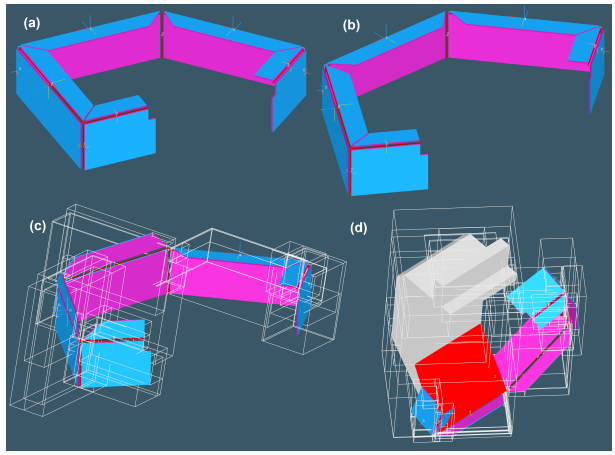

Fig. 4: Some elements of bending simulation with (c,d) and without (a,b) collision testing.

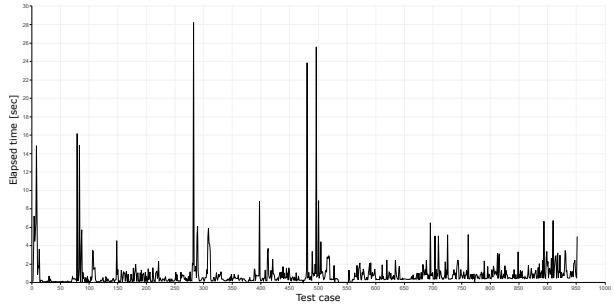

Fig. 5: Simulation time over 950 test cases of realistic sheet metal parts. The tests were conducted on a personal laptop with `Intel(R) Core(TM) i7-10870H CPU @ 2.20GHz, 32GiB RAM, Windows 10 x64`.

form 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 simplified 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.

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:

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

We employed this formal metric to analyze all sheet metal parts that have been identified as "infeasible" to verify the correctness of the simulation. Some examples of parts that yield $F_s = 0$ can be found in the full-text version of the paper.

The algorithm's main performance bottleneck (Fig. 5) is discovered for perforated sheet metal parts, as their simulation models comprise hundreds to thousands of triangles. These problems can be resolved through appropriate defeaturing of the flanges, as the presence of small holes and cutouts has no impact on the selection of a bending sequence.

References:

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.