The Algorithmic Foundations of Reverse Unfolding in Automated Panel Bending: A Deep Analysis of Salvagnini P4 Sequencing Logic

1. Introduction: The Deterministic Imperative in Sheet Metal CAPP

The transition from manual craftsmanship to Flexible Manufacturing Systems (FMS) in the sheet metal industry has necessitated a profound transformation in the domain of Computer-Aided Process Planning (CAPP). In traditional environments, the press brake operator serves as a real-time heuristic solver, intuitively managing the spatial orientation of a workpiece, adjusting grasp positions, and resolving collision paths on the fly. However, the advent of automated panel benders—exemplified by the Salvagnini P4 architecture—removes the human operator from the loop, replacing intuitive adaptability with rigid kinematic constraints. This shift imposes a zero-tolerance requirement for process planning errors. A sequencing mistake that might cause a minor delay for a human operator can result in catastrophic machine collisions, irreversible "loss of grasp" events, or the production of scrap in an automated cell.[1]

Consequently, the burden of manufacturing intelligence has shifted entirely to the software layer. The "brain" of the panel bender must not only interpret the geometric design of a part but also effectively reverse-engineer the physical actions required to create it. This report provides an exhaustive analysis of the "Reverse Unfolding" strategy, the dominant algorithmic paradigm used to solve the Bend Sequencing Problem (BSP) for automated machinery. By inverting the manufacturing timeline—starting from the finished 3D topology and computationally "peeling" flanges back to a flat pattern—these algorithms navigate the combinatorial explosion of potential sequences with orders of magnitude greater efficiency than forward-simulation approaches.[3]

We will dissect the computational mechanics of this strategy, exploring the graph-theoretical frameworks used to model precedence constraints, the geometric engines that determine flange "removability," and the heuristic optimization techniques (specifically A* search) that balance physical feasibility with cycle-time efficiency. This analysis specifically targets the constraints of the Salvagnini P4, integrating considerations for its unique manipulator kinematics and Automatic Blankholder Adjustment (ABA) technology.[5]

2. The Computational Logic of Reverse Unfolding

The choice of a reverse unfolding strategy over a forward bending simulation is not merely a matter of implementation preference; it is a fundamental response to the topology of the search space in sheet metal manufacturing. To understand the efficiency of the reverse approach, one must first appreciate the mathematical scale of the problem it attempts to solve.

2.1 The Combinatorial Explosion and Search Space Asymmetry

The Bend Sequencing Problem (BSP) is a classic example of an NP-hard combinatorial problem. For a sheet metal part with N bends, the theoretical number of sequential permutations is $N!$ (factorial). For a moderately complex chassis with 12 bends, there are approximately 479 million possible sequences. When one introduces the variables of tool selection, backgauge positioning, and manipulator grasping orientation (which can change between bends), the state space expands exponentially.[3]

The computational advantage of reverse unfolding arises from the inherent asymmetry of this state space. In a **Forward Simulation**—which mimics the physical additive process—the initial state is a flat sheet of metal. This state is geometrically unconstrained; devoid of flanges, tabs, or vertical protrusions, the flat sheet presents no obstructions to the machine tooling. Consequently, the search tree at the root node ($t = 0$) has a branching factor of N. The algorithm could theoretically select *any* of the N bend lines as the first operation. However, the vast majority of these initial choices lead to inevitable "dead ends" deep in the search tree—configurations where the remaining bends cannot be formed because the geometry created in the early steps physically blocks access to the subsequent bend lines. A forward-planning algorithm is thus forced to explore a diverging tree structure, often wasting significant computational resources simulating partial sequences that are ultimately infeasible.[4]

In contrast, **Reverse Unfolding** initiates the search at the "Goal State"—the finished, fully folded 3D part. This state is characterized by maximum geometric constraint. Flanges are folded, tabs are tucked into corners, and internal features are enclosed by surrounding walls. In this highly constrained configuration, the number of "valid moves" (bends that can be theoretically unfolded or removed) is extremely low. For a typical box-like enclosure, despite having perhaps 12 total bends, only 2 or 3 flanges might be physically accessible to the unfolding algorithm without colliding with adjacent geometry. This radically reduces the effective branching factor at the root of the reverse search tree. By starting at the point of maximum constraint, the reverse algorithm effectively "prunes" the search space before traversal even begins, filtering out up to 90% of invalid sequences simply by observing the topology of the finished part. The search space effectively resembles a funnel rather than a

diverging cone, guiding the solver toward valid solutions with much higher probability.[3]

2.2 Global vs. Local Constraint Visibility

Another critical efficiency gain in reverse unfolding involves the nature of collision detection. In forward simulation, validating a move requires a predictive analysis: "If I bend this flange now, will it prevent me from bending the internal tab ten steps later?" This requires deep look-ahead and continuous path simulation. In reverse unfolding, constraints are explicitly visible. If Flange A physically overlaps Flange B in the folded model, it is geometrically obvious that Flange B cannot be unfolded until Flange A is removed. The algorithm does not need to predict this interference; it simply observes it as a static condition of the current state.

This visibility allows the use of rapid **Global Accessibility Checks**. Before engaging in expensive fine-mesh collision detection, the solver can perform lightweight bounding-box intersections or ray-casting operations. If a ray projected from the normal vector of a flange strikes another face of the part, the flange is immediately flagged as "blocked," and the branch is terminated. This capability to execute rapid "fail-fast" checks on the static geometry is a primary driver of the computational speed associated with reverse engineering algorithms.[9]

2.3 The Rigid Body Assumption and Reversibility

The validity of the reverse strategy hinges on the principle of mechanical reversibility. While the plastic deformation of metal is physically irreversible (due to work hardening and grain structure changes), the *geometric kinematics* of the bending process are generally treated as reversible rigid body transformations in CAPP systems. If a flange can be rotated from its final angle θ_{final} to 0° without intersecting the tool or the workpiece, it follows that the tooling can rotate the flange from 0° to θ_{final} without collision, provided the machine axes can reach the position.

This assumption simplifies the simulation significantly. Instead of modeling complex material flow, the algorithm treats each flange as a rigid polygon mesh connected by a hinge (the bend line). The operation of "Unfolding" is computationally identical to "Bending" but with the time vector inverted. The generated sequence of "unfoldings" is simply reversed at the end of the process to produce the valid CNC manufacturing code.[9]

3. Determining Flange "Removability": The Heuristic Predicate

In the context of the reverse simulation, the central decision logic revolves around a single predicate function: IsRemovable(Flange, CurrentState). This function is not a simple binary check but the result of a hierarchical validation process that rigorously tests the flange

against geometric, tooling, and kinematic constraints. A flange is only added to the "Candidate List" for the next step if it satisfies all layers of this validation stack.

3.1 Geometric Removability: Dynamic Collision Detection

The most fundamental criterion for removability is self-intersection. A flange cannot be unfolded if its motion path passes through another section of the sheet metal part. While static interference checks (checking the start and end positions) are necessary, they are not sufficient. Complex features such as return flanges, hems, or curled edges can be collision-free in both the folded and flat states but collide during the transition swing.

To address this, modern algorithms employ **Dynamic Unfolding Simulation**. This technique discretizes the unfolding rotation into a series of time steps or "frames" (e.g., every 2° or 5° of rotation). At each frame, the algorithm updates the transformation matrix of the active flange's mesh and performs a Boolean intersection test against the static mesh of the remaining part.[9]

$$\text{Intersection}(V_{swept}(F), V_{static}) = \emptyset$$

Where $V_{swept}(F)$ represents the swept volume of the flange F throughout its rotation, and V_{static} represents the volume of the rest of the part. To optimize performance, this is often implemented using a broad-phase/narrow-phase collision detection architecture. A **Bounding Volume Hierarchy (BVH)**, typically using Axis-Aligned Bounding Boxes (AABBs) or Oriented Bounding Boxes (OBBs), is used for the broad phase. If the bounding volume of the rotating flange does not overlap the bounding volume of the static part, the expensive mesh-triangle intersection test is skipped.[3]

3.2 Tooling Removability: The "Forward-in-Reverse" Check

A flange may be geometrically removable relative to the part, but impossible to form due to tool interference. In reverse unfolding, the algorithm must verify that the tooling required to create the bend would not have collided with the part configuration existing at that stage. This effectively asks, "Could the tool have been here?"

The simulation loads a "Dictionary" of tool profiles corresponding to the specific machine configuration (e.g., the P4's upper and lower bending blades, the blankholder segments, and the counterblade).[3] The virtual tool is instanced at the bend line coordinates. The system then performs a collision check between this virtual tool volume and the *current* state of the folded part.

Crucially, this logic handles the "enclosure" constraint. For example, if the algorithm attempts to unfold a bottom flange of a box while the side flanges are still present (in the reverse state),

the collision check will reveal that the side flanges intersect the tool holder. This collision signals that the side flanges are "blocking" the bottom flange from a tooling perspective. Consequently, the side flanges must be unfolded (removed) *before* the bottom flange becomes valid. This mechanism naturally discovers the dependency order required for deep box bending without explicit rule-coding.[10]

3.3 Kinematic Removability: The Manipulator Constraint

The Salvagnini P4 differs fundamentally from a press brake in its part handling. The operator does not hold the part; a robotic manipulator does. This introduces a rigorous kinematic constraint: **Grasp Validity**. A flange is only removable if the machine can securely hold the part while that flange is being processed.[11]

The IsRemovable predicate must therefore evaluate the *stability* of the part. The algorithm analyzes the available flat surface area on the component in its current folded state. It attempts to find a valid "Grasp Configuration" where:

1. The manipulator's suction cups or grippers fit entirely within a flat, non-porous region (avoiding holes and cutouts).[14]
2. The grasp region is sufficiently far from the active bend line to avoid collision with the bending blades.
3. The Center of Gravity (CoG) of the part lies within the stability polygon of the active suction cups to prevents sagging or slippage during the rapid accelerations of the P4 cycle.[15]

If the only available surface for grasping is the flange currently being unfolded, a paradox exists. The algorithm must then search for a "Regrip" sequence—a set of intermediate moves where the part is placed down and re-grasped at a different location. If no valid grasp or regrip strategy can be found that enables the unfolding operation, the flange is deemed **Not Removable** due to kinematic infeasibility.[16]

4. Precedence Constraints and the Dependency Graph

While collision detection handles immediate physical impossibilities, many manufacturing constraints are logical or topological in nature. To manage these, the sequencing algorithm constructs a **Dependency Graph (DG)** (often referred to as a Precedence Graph). This graph formally encodes the "must-follow" relationships between features, transforming the open search space into a structured topological sort problem.

4.1 Graph Architecture and Feature Recognition

The construction of the Dependency Graph begins with high-level **Feature Recognition**. The raw Boundary Representation (B-Rep) of the CAD model is parsed into a **Face Adjacency**

Graph (FAG), where nodes represent planar faces and edges represent the bend lines connecting them.[18] Through subgraph isomorphism algorithms, the system identifies topological patterns such as "Tabs" (leaf nodes connected to larger faces), "Side Flanges" (nodes with specific aspect ratios), and "Internal Tabs" (nodes located inside the perimeter of another face).[20]

The Dependency Graph $G = (V, E)$ is then built where:

- **Vertices (V):** The set of all bend lines $\{b_1, b_2, ..., b_n\}$.
- **Directed Edges (E):** A relationship $b_i \rightarrow b_j$ indicating that bend b_i must be processed (unfolded) *before* bend b_j becomes available.

4.2 Encoding "Tab vs. Side" Constraints

One of the most critical precedence relationships in sheet metal is the interaction between a Side Flange and its associated Corner Tabs. The user query specifically highlights this interaction, which is governed by strict rigid-body logic.

Scenario: A standard box corner where a small Tab (T) is attached to the short edge of a Side Flange (S). Both are bent at 90°.

Analysis in Reverse: In the finished state, the Tab is bent relative to the Side, and the Side is bent relative to the Base. If the algorithm attempts to unfold the Side Flange (S) first, the Tab (T) remains attached to it, maintaining its 90° orientation relative to S. As S rotates down to become flat with the Base, the Tab (T) swings with it. In the flattened state of S, the Tab T will be protruding vertically from the sheet.

Constraint: This vertical protrusion creates a high risk of collision. The protruding tab may strike the machine table, the blankholder, or the manipulator arm as the part is moved. Furthermore, the tooling required to unfold the Side Flange (S) typically requires a flat clearance zone along the bend line; the protruding Tab would interfere with the blade.

Graph Definition: To avoid this, the Tab must be "flattened" relative to the Side *before* the Side is flattened relative to the Base. Thus, the Tab is a prerequisite for the Side.

- **Edge:** $T \rightarrow$ (Unfold Tab, then Unfold Side).
- **Forward Implication:** In the forward process, the Side is bent up (carrying the flat tab), and *then* the Tab is bent. This matches the standard "Bend the big flanges first, then the small details" heuristic often used in manual bending, although the P4 can sometimes

invert this depending on clearance.[21]

4.3 Corner Interlocks and Overlaps

The Dependency Graph is also the primary mechanism for resolving physical overlaps at corners.

Scenario: Flange A has a tab that tucks *behind* or *under* Flange B (a butt joint or overlap joint).

Geometric Reality: In the folded state, Flange B physically obstructs the unfolding path of Flange A. A ray cast from Flange A hits Flange B.

Graph Definition: The algorithm detects this occlusion during the initial interference scan. It establishes a hard constraint that the "covering" flange must be removed to expose the "covered" flange.

- **Edge:** $B \rightarrow$ (Unfold B, then Unfold A).

4.4 Internal Constraints and Cycle Detection

Features located inside cutouts (Internal Tabs) present a unique topological constraint. The surrounding sheet material effectively forms a "frame" that blocks tool access. In reverse logic, this frame (the surrounding flanges) must be unfolded to clear the path for the tool to access the internal bend.

- **Edge:** Surrounding Flanges $\rightarrow$ Internal Tab.

A crucial function of the Dependency Graph is **Cycle Detection**. If the graph generation process results in a cycle (e.g., $A \rightarrow B \rightarrow C \rightarrow A$), it indicates a topological impossibility, such as a "Pinwheel" interlock where every flange overlaps the next. A standard sequential bending process cannot fabricate such a part; it would require simultaneous bending or specialized segmented tooling that can retreat from a closed lock. The algorithm uses Depth First Search (DFS) or Johnson's Algorithm to detect these cycles immediately.[18] If a cycle is found, the part is flagged as "Infeasible" or "Requires Manual Review," saving the system from entering an infinite search loop.[23]

5. Algorithmic Implementation: A* Search and Optimization

The Dependency Graph identifies which moves are *valid* (i.e., the set of nodes with an In-Degree of 0). However, at any given step, there may be multiple valid flanges to unfold. The system must choose the *best* one to minimize production time and cost. To achieve this, modern CAPP systems utilize the *A (A-Star) Search Algorithm**.

5.1 The Cost Function Formulation

The A* algorithm evaluates paths through the state space by minimizing a cost function $f(n)$:

$$f(n) = g(n) + h(n)$$

Where $g(n)$ is the accumulated cost of the sequence from the start node to the current node n, and $h(n)$ is a heuristic estimate of the cost to reach the goal (flat pattern). For the Salvagnini P4, the cost factors included in $g(n)$ are heavily weighted by the mechanical characteristics of the machine [7]:

1. **Rotation Cost (W_{rot}):** The P4 manipulator must physically rotate the sheet to present different sides to the bending blades (0°, 90°, 180°, 270°). This rotation is a mechanically slow operation compared to the rapid stroke of the bending blade. Therefore, the algorithm assigns a high penalty to any sequence transition that requires a change in orientation ($Side_i \neq Side_{i+1}$). This drives the solver to "clear a side" (unfold all available flanges on one edge) before rotating the part.
    - *Table 1: Relative Cost Weighting (Approximation)*

Operation	Relative Cost Weight	Impact
Standard Bend	1.0	Baseline
ABA Adjustment	2.5	Slight delay for tool resizing
Part Rotation (90°)	5.0	Significant mechanical delay
Manipulator Regrip	15.0	High risk + High delay

2. **Regrip Cost (W_{grip}):** As noted in Section 3.3, releasing and re-grasping the part is a high-cost operation. It introduces tolerance errors (positioning uncertainty) and consumes significant cycle time. The cost function heavily penalizes sequences that force a regrip. The algorithm will aggressively search for a "Golden Grasp"—a single manipulator position that allows the maximum number of bends to be performed without letting go.[26]
3. **ABA Continuity (W_{tool}):** The P4 features an Automatic Blankholder (ABA) that adjusts

its length to match the bend line. While faster than a manual tool change, resizing the ABA still takes time. The algorithm groups bends of similar lengths (e.g., all 100mm tabs) to minimize the number of ABA adjustment cycles.[5]

5.2 Heuristic Look-Ahead ($h(n)$)

The efficiency of A* depends on the quality of the heuristic $h(n)$. A sophisticated heuristic for panel bending analyzes the topology of the remaining folded bends.

- **Face Clustering Heuristic:** If the remaining bends are distributed across all four sides of the part, the heuristic adds a cost proportional to at least 3 rotations ($3 \times W_{rot}$). If the remaining bends are all on one side, the estimated rotation cost is zero.
- **Constraint Density:** If a specific bend has a high number of dependencies (it blocks many other bends), the heuristic might prioritize its removal to "unlock" the rest of the model, increasing the branching factor for future steps and allowing better optimization.

By combining the rigorous constraints of the Dependency Graph with the cost-aware optimization of A*, the software produces a sequence that is not only geometrically valid but also economically optimal, minimizing the "air time" (non-value-added time) of the machine.[27]

6. Conclusion

The "Reverse Unfolding" strategy stands as the cornerstone of modern computational process planning for sheet metal fabrication. Its widespread adoption is driven by its ability to align the computational solver with the physical reality of the problem: the finished part is the most constrained state, and therefore the most logical starting point for planning. By filtering the search space through the "funnel" of reverse topology, the algorithm solves the $N!$ sequencing problem in polynomial time.

For advanced automation like the Salvagnini P4, this strategy is augmented by rigorous topological modeling. The construction of Dependency Graphs ensures that complex feature interactions—such as the interlocking of tabs and side flanges—are respected as hard constraints. Simultaneously, the definition of "Removability" is expanded beyond simple geometry to include the kinematic limitations of the robotic manipulator and the collision profiles of the universal tooling.

Through the integration of graph theory, geometric simulation, and heuristic search, the control software effectively acts as an expert system, deriving the optimal manufacturing strategy from pure geometry. It transforms the sequence planning process from a trial-and-error human effort into a deterministic, mathematical calculation, enabling the "Batch One" flexibility that defines Industry 4.0 manufacturing.

Table 2: Summary of Constraint Logic in Reverse Unfolding

Constraint Type	Constraint Logic	Graph Representation	Reverse Action
Geometric Block	Flange A physically covers Flange B	$A \rightarrow$	Unfold A before B
Tooling Block	Tool volume for Bend B hits Flange A	$A \rightarrow$	Unfold A before B
Tab/Side Hierarchy	Tab T is attached to Side S	$T \rightarrow$	Unfold T before S
Corner Interlock	Side A tucks under Side B	$B \rightarrow$	Unfold B before A
Internal Cutout	Window Frame blocks tool access	Frame $\rightarrow$ Internal	Unfold Frame before Internal

References

- **Reverse Strategy & Search Space:** [3]
- **Geometric & Dynamic Collision:** [3]
- **Dependency Graphs & Topology:** [18]
- *Optimization (A) & Heuristics:** [7]
- **Salvagnini P4 Specifics (ABA, Manipulator):** [1]
- **Feature Recognition (Tabs):** [20]

Works cited

1. Automatic Sheet Metal Bending Machine - Salvagnini P4, accessed February 1, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders/P4
2. Panel Bender -P4 - MachineryHost, accessed February 1, 2026, https://f.machineryhost.com/71b2fab0316bfa7fa24ce0a0e5460bad/Brochure%20-%20Panel%20Bender%20-P4.pdf
3. On sheet metal unfolding. Part 12: searching for feasible bends | by ..., accessed February 1, 2026,

https://analysis-situs.medium.com/on-sheet-metal-unfolding-part-12-searching-for-feasible-bends-2a91a8976431
4. 255 Title: Bending Simulation Framework for Rapid Feasibility Checks of Sheet Metal Parts Authors - CAD Conference, accessed February 1, 2026, https://www.cad-conference.net/files/CAD25/CAD25_255-260.pdf
5. Sheet metal panel bender P2 - Salvagnini, accessed February 1, 2026, https://www.salvagninigroup.com/en-INT/products/panel-benders/P2
6. The automatic hybrid adaptive Panel Bender. - Sanson Machinery Group, accessed February 1, 2026, http://www.sansonmachinery.com/wp-content/uploads/2013/12/P4Xe-UK-3962020312.pdf
7. Bending Sequence Automation for Sheet Metal | PDF | Computers - Scribd, accessed February 1, 2026, https://www.scribd.com/document/668802543/Bend-order-in-sheetmetal
8. Manufacturability-Driven Decomposition of Sheet Metal Products - Carnegie Mellon University Robotics Institute, accessed February 1, 2026, https://www.ri.cmu.edu/pub_files/pub1/wang_cheng_hua_1997_1/wang_cheng_hua_1997_1.pdf
9. On sheet metal unfolding. Part 9: dynamic unfolding | by Analysis Situs, accessed February 1, 2026, https://analysis-situs.medium.com/on-sheet-metal-unfolding-part-9-dynamic-unfolding-067e7aa95f1e
10. Geometry-based Bend Feasibility Matrix for bend sequence planning of sheet metal parts | Request PDF - ResearchGate, accessed February 1, 2026, https://www.researchgate.net/publication/339802527_Geometry-based_Bend_Feasibility_Matrix_for_bend_sequence_planning_of_sheet_metal_parts
11. Using Features and Their Constraints to Aid Process Planning of Sheet Metal Parts - Robotics and Automation, 1995. Proceedings.,, accessed February 1, 2026, https://www.ri.cmu.edu/pub_files/pub2/wang_cheng_hua_1995_1/wang_cheng_hua_1995_1.pdf
12. Automated design of sheet metal punches for bending multiple parts in a single setup, accessed February 1, 2026, https://quaoar.su/files/papers/sheet_metal/alva2001.pdf
13. Intelligent system for generating and executing a sheet metal bending plan - Google Patents, accessed February 1, 2026, https://patents.google.com/patent/US20040019402A1/en
14. Sheets handling bending - manipulator for steel plates - Dalmec, accessed February 1, 2026, https://www.dalmec.com/en/sheets_handling/
15. Learning a Real Time Grasping Strategy - Joanna Bryson, accessed February 1, 2026, https://joanna-bryson.squarespace.com/s/learning-a-real-time-grasping-strategy.pdf
16. p4-benders.pdf - Empire Machinery, accessed February 1, 2026, https://www.empire-machinery.com/wp-content/uploads/2021/06/p4-benders.pdf

17. Optimized bending sequences of sheet metal bending by robot | Request PDF - ResearchGate, accessed February 1, 2026, https://www.researchgate.net/publication/238173170_Optimized_bending_sequences_of_sheet_metal_bending_by_robot
18. sheet metal features - Analysis Situs, accessed February 1, 2026, https://analysissitus.org/features/features_sheet-metal.html
19. Methods and systems for feature recognition.pdf - ALARVY, accessed February 1, 2026, https://alarvy.com/downloads/Methods%20and%20systems%20for%20feature%20recognition.pdf
20. Automatic Feature Recognition Techniques for the Integration of CAD and CAM: A Review, accessed February 1, 2026, https://dl.astm.org/ssms/article/8/1/83/17347/Automatic-Feature-Recognition-Techniques-for-the
21. Sheet Metal Tabs - 2021 - SOLIDWORKS Design Help, accessed February 1, 2026, https://help.solidworks.com/2021/english/SolidWorks/sldworks/c_sheet_metal_tabs.htm
22. feature recognition and design advisory system for sheet metal components - ResearchGate, accessed February 1, 2026, https://www.researchgate.net/publication/242295693_FEATURE_RECOGNITION_AND_DESIGN_ADVISORY_SYSTEM_FOR_SHEET_METAL_COMPONENTS
23. Sequence planning of sheet metal parts manufactured using progressive dies, accessed February 1, 2026, https://d-nb.info/128415999X/34
24. Optimization of strip-layout using graph-theoretic methodology for stamping operations on progressive die - Semantic Scholar, accessed February 1, 2026, https://pdfs.semanticscholar.org/9ef0/898924d84ab09a47769578ac8b5447500f15.pdf
25. A Bending Sequence Planning Algorithm Based on Multiple-Constraint Model - Scientific.net, accessed February 1, 2026, https://www.scientific.net/AMR.1042.26
26. Automated Process Planning for Sheet Metal Bending Operations - Carnegie Mellon University's Robotics Institute, accessed February 1, 2026, https://www.ri.cmu.edu/pub_files/pub1/kim_kyoung_k_1998_1/kim_kyoung_k_1998_1.pdf
27. ARTICLE TEMPLATE Automatic computation of bending sequences for wire bending machines - OuluREPO, accessed February 1, 2026, https://oulurepo.oulu.fi/bitstream/handle/10024/43425/nbnfi-fe2023032733289.pdf?sequence=1&isAllowed=y
28. Methods for the sequencing of sheet metal bending operations - ResearchGate, accessed February 1, 2026, https://www.researchgate.net/publication/261592157_Methods_for_the_sequencing_of_sheet_metal_bending_operations
29. Bending Simulation Framework for Rapid Feasibility Checks of Sheet Metal Parts - CAD Journal, accessed February 1, 2026, https://www.cad-journal.net/files/vol_23/CAD_23(1)_2026_85-100.pdf