Architectural Specification: Phase 3 Sequencer and Optimal Path Planning for Salvagnini P4 Robotics

1. Executive Summary: The Non-Linear Optimization Challenge

The industrial fabrication landscape has shifted definitively toward high-mix, low-volume production, necessitating automated systems capable of "batch-one" efficiency. In this context, the Salvagnini P4 Panel Bender represents a pinnacle of flexible automation, utilizing universal bending tools and an automated manipulator to form complex sheet metal geometries without manual retooling.[1] However, the physical flexibility of the P4 hardware imposes a significant computational burden: the determination of an optimal bending sequence is no longer a linear scheduling problem but a complex search through a high-dimensional state space governed by non-linear cost functions.

This report serves as the comprehensive architectural definition for **Phase 3 - The Sequencer**, the software module responsible for linearizing the Precedence Graph (DAG) into an executable machine code sequence. The primary objective is to minimize Total Cycle Time (T_{cycle}) by leveraging the P4's unique "Masked Time" architecture—a capability where auxiliary machine operations, specifically Automatic Blankholder Adjustment (ABA) and tool exchanges, occur in parallel with manipulator rotation.[3]

The core mathematical challenge lies in the cost function. Unlike traditional serial manipulators where $Cost_{total} = \sum t_{actions}$, the P4's concurrent architecture dictates that transition costs are determined by $Cost_{transition} = \max(t_{rotation}, t_{tooling})$. This non-linearity renders standard greedy algorithms insufficient, as they fail to identify strategic opportunities to "hide" expensive tooling changes behind necessary geometric reorientations. Consequently, this architecture mandates an *A Search Algorithm** driven by a custom heuristic $f(n) = g(n) + h(n)$ that accounts for these parallel dynamics, integrated with a dynamic "Repo" (Regripping) injection mechanism to resolve topological dead-ends.[5]

The following specification details the mathematical derivation of the cost function, the structural definition of the Search Tree, the discrete-continuous hybrid state management strategy using Zobrist hashing, and the logic for dynamic constraint relaxation via robot repositioning.

2. Kinematic Modeling of the P4 Architecture

To derive an accurate cost function $g(n)$, one must first establish a rigorous kinematic model of the machine's operations. The P4 Panel Bender is not a single actor but a synchronized system comprising the Manipulator, the Press (Bending Unit), and the Automatic Blankholder (ABA).

2.1 The Manipulator: Rotational and Translational Dynamics

The manipulator is the primary driver of cycle time during non-bending phases. It is responsible for handling, gripping, and rotating the sheet to present the correct flange to the bend line.[3] The manipulator's motion profile is defined by two primary degrees of freedom relevant to sequencing: planar translation (x, y) and rotation θ.

Research indicates that the manipulator's rotation is the dominant variable in "masked time" calculations. The P4 manipulator is capable of rapid rotation, but the large moments of inertia associated with substantial metal panels constrain angular acceleration.[7] The time required for a rotation $\Delta\theta$ is modeled not as a simple linear function, but as a trapezoidal velocity profile accounting for acceleration ($ \alpha $), maximumvelocity($ \omega_{max} $), andsettlingtime($ t_{settle} $):

$$t_{rot}(\Delta\theta) = \begin{cases} 2\sqrt{\frac{\Delta\theta}{\alpha}} + t_{settle} & \text{if } \omega_{peak} < \omega_{max} \\ \frac{\Delta\theta}{\omega_{max}} + \frac{\omega_{max}}{\alpha} + t_{settle} & \text{if } \omega_{peak} \geq \omega_{max} \end{cases}$$

In the context of the P4, manipulator rotation often involves standard increments (90°, 180°) required to manipulate the four sides of a rectangular panel. However, efficient path planning must also consider subtler reorientations for non-orthogonal bends or collision avoidance. The "cost" of rotation is the baseline against which other operations are measured.[4]

2.2 The Automatic Blankholder (ABA) and Tooling Logic

The ABA is the upper tool that clamps the sheet against the counterblade. Its length (L_{ABA}) must match the length of the current bend to ensure even pressure and prevent collisions with adjacent returns (side flanges).[3] Historically, adjusting tool length was a setup operation performed between batches. On the P4, the ABA adjusts automatically and instantaneously within the cycle.[4]

The adjustment mechanism operates at a finite linear velocity (v_{ABA}). The time required to change the tool length from $L_{current}$ to L_{target} is:

$$t_{ABA} = \frac{|L_{target} - L_{current}|}{v_{ABA}} + t_{lock}$$

Where t_{lock} represents the mechanical engagement time of the tool segments. Crucially, the ABA allows for "masked time" operations. If the machine is commanded to rotate the part (taking t_{rot}) and adjust the ABA (taking t_{ABA}) simultaneously, the effective cycle time consumption is:

$$t_{transition} = \max(t_{rot}, t_{ABA})$$

This relationship is the cornerstone of the optimization strategy.[1] A sequence that performs a 90° rotation (taking, hypothetically, 2.5 seconds) allows for a free ABA adjustment of any magnitude that takes less than 2.5 seconds. Conversely, a sequence that requires a major ABA adjustment without a corresponding rotation exposes the full tooling cost to the cycle time, creating a "penalty" state.

2.3 Bending and Auxiliary Operations

The bending process itself—the actuation of the upper and lower blades—is largely deterministic and sequence-independent for a given bend geometry.[2] While the sequencing algorithm tracks the accumulated bending time, it serves primarily as a constant lower bound. Auxiliary tools, such as P/T (Panel/Tool) or CLA (Auxiliary Blades) for interrupted bends, introduce binary state changes. Engaging or disengaging these tools incurs a fixed time cost t_{aux}.[10] These operations also follow masked time logic; if a tool change occurs during a rotation or a loading/unloading operation, its effective cost may be zero.[4]

Table 1: Comparative Kinematic Costs and Masking Potential

Operation Domain	**Key Variables**	**Time Function**	**Masking Potential**	**Source**
Manipulator Rotation	$\Delta\theta$, Inertia I	$t_{rot} \approx$	**Primary Masker** (Hides other ops)	3

ABA Adjustment	ΔL, v_{ABA}	$t_{ABA} \propto	\Delta L	$
Auxiliary Tooling	State (In/Out)	Fixed t_{switch}	Maskable during large rotations	10
Regripping (Repo)	P_{grip}, P_{new}	High Fixed Penalty + t_{move}	**Blocking** (Cannot be fully masked)	5
Bending Actuation	Angle, Material	Constant per bend	Non-Maskable (Active Value Add)	2

---

3. Mathematical Formulation of the Optimization Problem

The objective is to find a permutation sequence $S = \{b_1, b_2, \ldots, b_n\}$ of the set of required bends B, subject to constraints defined by the Precedence Graph $G = (V, E)$, minimizing the scalar cost function $J(S)$.

3.1 The Cost Function $f(n)$

In the context of the A* search algorithm, we evaluate nodes based on $f(n) = g(n) + h(n)$.

3.1.1 The Accumulated Cost $g(n)$

$g(n)$ represents the actual time elapsed from the start state to the current state n. It is defined recursively:

$$g(n) = g(parent(n)) + C_{transition}(parent(n), n) + t_{bend}(n)$$

The transition cost $C_{transition}$ encapsulates the non-linear "masked" logic derived from the machine state. Let state S_i be defined by vector $\mathbf{x}_i =$ (Orientation, ABA Length, Gripper Position, Tool State).

The transition to state S_{i+1} (required for bend b_{i+1}) incurs cost:

$$\Delta t_{ops} = \max\left(t_{rot}(|\theta_{i+1} - \theta_i|), \quad t_{ABA}(|L_{i+1} - L_i|), \quad t_{aux}(T_{i+1} \neq T_i)\right)$$

However, this transition is only valid if the gripper position P_i is compatible with bend b_{i+1}. If CheckGrasp() determines that P_i causes a collision or is out of reach for the target bend, a **Repositioning (Repo)** operation must be injected *before* the bend.

If Repo is required:

$$C_{transition} = \Delta t_{repo} + \Delta t_{ops\_post\_repo}$$

Where Δt_{repo} is the time to unclamp, move the manipulator to P_{new}, and reclamp. This is an expensive, purely additive cost.[11]

Therefore, the rigorous definition of $g(n)$ is:

$$g(n) = \sum_{k=1}^{depth(n)} \left(\mathbb{I}_{repo_k} \cdot C_{repo} + \max(t_{rot_k}, t_{ABA_k}) + t_{bend_k}\right)$$

Where $\mathbb{I}_{repo_k}$ is an indicator function (1 if repo required, 0 otherwise).

3.1.2 The Heuristic Function $h(n)$

To ensure the A* algorithm is optimal, $h(n)$ must be **admissible** (it must never overestimate the remaining cost) and preferably **consistent** (monotonic).[13] A naive heuristic (counting remaining bends) is insufficient. We derive a robust heuristic by considering the "unavoidable" costs remaining in the DAG.

1. **Bending Lower Bound (h_{bend}):** The sum of bending times for all unvisited nodes in the DAG. This is constant regardless of sequence.

$$h_{bend}(n) = \sum_{b \in B_{remaining}} t_{bend}(b)$$

2. **Rotational Lower Bound (h_{rot}):** The P4 processes parts by "side" (Top, Bottom, Left, Right). If the current state is on Side A, and the set of remaining bends $B_{remaining}$ includes bends on Sides B, C, and D, the machine *must* strictly perform rotations to visit these faces.
   We can model the remaining sides as a minimalist Traveling Salesperson Problem (TSP) on a circle (0°, 90°, 180°, 270°). The lower bound is the minimum angular distance required to visit all remaining occupied sectors.

$$h_{rot}(n) = \text{MST}_{angular}\left(\theta_{curr}, \{\theta_{target} | b \in B_{remaining}\}\right)$$

3. **Tooling Lower Bound (h_{ABA}):** Similarly, if remaining bends require ABA lengths $\{L_1, L_2, \dots\}$, the machine must physically traverse the range of lengths. A lower bound is the range span or the sum of minimum transitions between required clusters of lengths.

The Composite Heuristic:

Due to masked time, we cannot simply sum h_{rot} and h_{ABA}. The strictly admissible heuristic is:

$$h(n) = h_{bend}(n) + \max(h_{rot}(n), h_{ABA}(n))$$

This asserts that the remaining time is at least the bending time plus the greater of the strictly necessary manipulation time or the strictly necessary tooling adjustment time. This formulation satisfies the admissibility criterion while guiding the search toward sequences that parallelize these distinct cost centers.[5]

---

4. State Space Architecture and Transition Logic

The sequencing engine relies on a robust definition of the "State" to track progress and detect cycles or redundant paths.

4.1 SearchNode Struct Definition

The SearchNode is the fundamental unit of the A* search tree. It must capture both the discrete progress (which bends are done) and the continuous machine configuration (where the robot is holding).

C++

```cpp
struct SearchNode {
    // --- Discrete Process State ---
    // Bitmask representing the set of completed bends.
    // Efficient for checking precedence and goal state.
    std::bitset<MAX_BENDS> completed_bends;

    // --- Continuous Machine State ---
    // Current orientation of the sheet in degrees.
    double current_orientation_deg;

    // Current length of the ABA tool in mm.
    double current_aba_length_mm;

    // Current state of auxiliary tools (e.g., P/T, CLA).
    // Encoded as bitflags or enum.
    uint32_t active_aux_tools;

    // --- Grasping Geometry (The "Repo" State) ---
    // The manipulator's grip is defined by a position on the sheet's boundary
    // and an orientation relative to the sheet.
    Point2D gripper_position_local;
    double gripper_angle_local;

    // --- Optimization Metrics ---
    double g_cost;  // Actual time from start
    double h_cost;  // Heuristic estimate to goal

    // Derived F-cost (g + h) for priority queue ordering.
    double get_f_cost() const { return g_cost + h_cost; }

    // --- Tree Lineage ---
    // Pointer to parent for path reconstruction.
    std::shared_ptr<SearchNode> parent;

    // The action that led to this state (Bend ID or Repo Action).
    ActionType action_type;
    int action_target_id;
};
```

4.2 Zobrist Hashing for Mixed Discrete/Continuous States

Efficient state management requires a mechanism to check if a state has already been visited (ClosedSet). A standard hash map is ideal, but the SearchNode contains continuous variables (gripper position, ABA length). Floating-point comparison is unreliable and slow. We employ **Zobrist Hashing**, adapted from game theory (Chess) to robotics.[15] This technique typically uses XOR operations on random bitstrings assigned to board pieces. Here, we extend it to continuous domains via **Discretization Grids**.[17]

The Hashing Algorithm:

1. **Discretization:**
   - **Gripper Position:** The sheet boundary is unrolled into a 1D coordinate system (perimeter distance). This perimeter is divided into discrete buckets (e.g., 10mm segments). The continuous gripper_position_local is mapped to a BucketID.
   - **ABA Length:** The operational range (e.g., 500mm to 3000mm) is bucketed into functional zones or discrete tool steps.
   - **Orientation:** Quantized to standard machine increments (e.g., 1° resolution).
2. **Initialization:**
   - Generate a static table of 64-bit random integers Z_Table.
   - Z_Bends[i]: Random hash for Bend i being completed.
   - Z_Grip[k]: Random hash for Gripper being in perimeter bucket k.
   - Z_Side[d]: Random hash for Orientation bucket d.
3. **Hash Computation:**

$$H(S) = \left(\bigoplus_{i \in S.completed} Z_{Bends}[i] \right) \oplus Z_{Grip} \oplus Z_{Side}$$

This hash $H(S)$ serves as the key for the ClosedSet. If a collision occurs (rare with 64-bit keys), a secondary rigorous geometry check resolves the ambiguity. This allows $O(1)$ insertion and lookup, critical for managing the combinatorial explosion of the search space.[19]

5. Dynamic Move Injection: The Repo Logic

A critical requirement of the sequencer is handling "Dead Ends"—states where the manipulator cannot perform the next required bend due to geometric constraints (e.g., the gripper is colliding with the tool, or holding the flange that needs to be bent). Standard A* would treat this as a non-traversable edge. However, the robot has the ability to **Reposition (Regrip)**.

Since listing all possible Regrips at every step would explode the branching factor, we utilize **Dynamic Move Injection**.[5] We only generate Repo nodes when necessary or heuristically promising.

5.1 The Injection Algorithm

When expanding a node N_{parent} and attempting to transition to Bend b_{target} :

1. **Grasp Validation (CheckGrasp):**

   The solver checks if $N_{parent}.gripper\_position$ is valid for b_{target} .
   - *If Valid:* Generate a standard "Bend Node".
   - *If Invalid:* The standard path is blocked. Trigger **Repo Generation**.
2. **Repo Candidate Generation:**

   We must find a *new* grasp position P_{new} that is valid for b_{target} and ideally valid for subsequent bends.
   - **Scan Geometry:** Analyze the remaining unbent perimeter segments.
   - **Heuristic Filtering:** Select candidate P_{new} locations that maximize the **Grasp Quality Metric**:
     - Distance from corners (stability).
     - Centrality relative to remaining bends (reachability).
     - Clearance from current bend lines (safety).
   - Source [5]: "Bigger width and longer finger length is favored."
3. **Cost Penalization:**

   Create a new child node N_{repo} .
   - completed_bends: Same as N_{parent} .
   - gripper_position: P_{new} .
   - action_type: REPOSITION.
   - g_cost: $N_{parent}.g\_cost + Cost_{repo}$.

$$Cost_{repo} = t_{unclamp} + t_{move}(P_{old} \rightarrow P_{new}) + t_{clamp} + t_{stabilize}$$

4. **Injection:**

   Add N_{repo} to the OpenSet.

Implication: The A* algorithm now has a choice. It can continue searching other branches (different bend orders) that don't require a repo. Or, if all branches are blocked or expensive, it will pay the cost to traverse the N_{repo} edge. This guarantees completeness: if a solution involving regripping exists, A* will find it, but it will prioritize repo-free solutions due to the high g_cost penalty.[6]

---

6. Constraint Solver Architecture

The A* engine delegates feasibility checks to specialized geometric solvers. These solvers effectively prune the search tree.

6.1 CheckABA(State, TargetBend)

This solver manages the tooling constraints.[4]

- **Input:** Target Bend Length, Return Flange Dimensions, Current ABA State.
- **Logic:**
  1. **Composition:** Compute the optimal combination of ABA segments to match the bend length.
  2. **Collision:** Check if the tool width fits between the return flanges (Box Depth check).
  3. **Masking:** Calculate ΔL. Return the transition time.
- **Insight:** The solver should be "lazy." It typically finds the *nearest* valid tool length to the current length, rather than resetting to a default, to minimize Δt_{ABA} and maximize the chance of masking.

6.2 CheckGrasp(State, TargetBend)

This solver manages the manipulator constraints.[5]

- **Input:** Gripper Geometry, Part Mesh (current deformation state), Machine Tool Models.
- **Logic:**
  1. **Inverse Kinematics (IK):** Calculate the required robot coordinates (x, y, θ) to place the bend line at the press axis.
  2. **Reachability:** Is this (x, y, θ) within the robot's workspace?
  3. **Collision (2D/3D):**
     - Project the gripper footprint onto the sheet plane.
     - Check intersection with the "Forbidden Zones" (active bend line, tool footprint, side gauges).
     - Check collision between the bent sheet geometry (3D) and the manipulator arm structure (vertical clearance).

- **Output:** Validity (Boolean) and SafetyMargin (Float).
- **Optimization:** Use "Shadow Maps" or Bitboard representations for the sheet and tool footprints to perform collision checks using bitwise AND operations, significantly faster than mesh-mesh intersection.[22]

---

7. Implementation Strategy and Performance

7.1 Architecture Diagram

The system follows a layered architecture:

1. **The Planner (Top Level):** Manages the A* loop, Open/Closed sets, and memory pool.
2. **The Expansion Engine:** Queries the DAG, identifies candidates, and calls solvers.
3. **The Solver Layer:** Stateless functions CheckGrasp and CheckABA performing pure geometry checks.
4. **The State Cache:** Hash map implementation for visited states.

7.2 Memory Management

Since A* generates thousands of nodes, dynamic allocation (new SearchNode) is a bottleneck.

- **Object Pooling:** Pre-allocate a large block of SearchNode structs.
- **Flyweight Geometry:** Do not store the full mesh in the node. Store only the completed_bends bitmask. The geometry engine can reconstruct the 3D shape on-the-fly from the bitmask and the base CAD model. This trades CPU cycles (reconstruction) for memory bandwidth, which is favorable in modern CPU architectures.[14]

7.3 Parallelizing the Search

While A* is inherently sequential, the **Expansion Step** is parallelizable.[23] When expanding node N, the evaluation of its children (e.g., 5 possible next bends) involves heavy geometric calls to CheckGrasp. These can be dispatched to a thread pool. The OpenSet management remains single-threaded to maintain priority integrity, but the heavy lifting of state generation becomes concurrent.

---

8. Conclusion

The "Sequencer" for the Salvagnini P4 is a sophisticated exercise in constrained optimization. By shifting from a linear cost model to a **max-function cost model** ($ \max(t_{rot}, t_{ABA}) $), we align the algorithm with the physical reality of "Masked Time." The integration of **Zobrist Hashing** allows for efficient navigation of the hybrid discrete-continuous state space, while the **Dynamic Repo Injection** strategy provides a robust fallback for topological dead-ends without polluting the search space with unnecessary moves.

This architecture moves beyond simple heuristics (e.g., "shortest bend first") to provide a mathematically provable optimum, directly translating the P4's hardware flexibility into measurable cycle time reduction. The resulting sequence does not merely instruct the machine *what* to do, but choreographs *how* to do it—hiding setup costs in the shadows of motion.

References & Research Basis

- **Masked Time & P4 Kinematics:** [1]
- *A Search in Manufacturing:** [5]
- **Regripping (Repo) Logic:** [5]
- **Zobrist Hashing & State Space:** [15]
- **Tooling & Constraints:** [2]

Works cited

1. Automatic Sheet Metal Bending Machine - Salvagnini P4, accessed February 3, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders/P4
2. Salvagnini panel bending: P4-2520 bends 4 different parts - YouTube, accessed February 3, 2026, https://www.youtube.com/watch?v=gOMHcSt2yQ0
3. Panel Bender -P4 - MachineryHost, accessed February 3, 2026, https://f.machineryhost.com/71b2fab0316bfa7fa24ce0a0e5460bad/Brochure%20-%20Panel%20Bender%20-P4.pdf
4. The automatic hybrid adaptive Panel Bender. - Sanson Machinery Group, accessed February 3, 2026, http://www.sansonmachinery.com/wp-content/uploads/2013/12/P4Xe-UK-3962020312.pdf
5. Automated Process Planning for Sheet Metal Bending Operations - Carnegie Mellon University's Robotics Institute, accessed February 3, 2026, https://www.ri.cmu.edu/pub_files/pub1/kim_kyoung_k_1998_1/kim_kyoung_k_1998_1.pdf
6. Bend Sequence Planner - Robotics Institute Carnegie Mellon University, accessed February 3, 2026, https://www.ri.cmu.edu/project/bend-sequence-planner/
7. Salvagnini panel bending: P4 high performances in sheet metal profile manufacturing - CUT option - YouTube, accessed February 3, 2026, https://www.youtube.com/watch?v=-tblzJD0YKk
8. The Panel Bender: Your Ultimate Guide - Italian Machinery Association, accessed February 3, 2026, https://www.italianmachines.eu/en/publications/437/the-panel-bender-your-ultimate-guide
9. Salvagnini flexible manufacturing system: S4+P4 line 100% flexible automation - YouTube, accessed February 3, 2026,

https://www.youtube.com/watch?v=ZWcZDVf8_Nc
10. p4-benders.pdf - Empire Machinery, accessed February 3, 2026, https://www.empire-machinery.com/wp-content/uploads/2021/06/p4-benders.pdf
11. Manufacturability-Driven Decomposition of Sheet Metal Products - Carnegie Mellon University's Robotics Institute, accessed February 3, 2026, https://www.ri.cmu.edu/pub_files/pub1/wang_cheng_hua_1997_1/wang_cheng_hua_1997_1.pdf
12. Bending Sequence Automation for Sheet Metal | PDF | Computers - Scribd, accessed February 3, 2026, https://www.scribd.com/document/668802543/Bend-order-in-sheetmetal
13. Admissible Heuristics for Optimal Planning - IDA.LiU.SE, accessed February 3, 2026, https://www.ida.liu.se/divisions/aiics/publications/AIPS-2000-Admissible-Heuristics-Optimal.pdf
14. Why does A* with admissible non consistent heuristic find non optimal solution?, accessed February 3, 2026, https://stackoverflow.com/questions/51684682/why-does-a-with-admissible-non-consistent-heuristic-find-non-optimal-solution
15. Zobrist Hashing - Chessprogramming wiki, accessed February 3, 2026, https://www.chessprogramming.org/Zobrist_Hashing
16. Zobrist Hash-based Duplicate Detection in Symbolic Regression - arXiv, accessed February 3, 2026, https://arxiv.org/html/2508.13859v1
17. Deep Supervised Discrete Hashing - NIPS, accessed February 3, 2026, http://papers.neurips.cc/paper/6842-deep-supervised-discrete-hashing.pdf
18. CS89 -- – Discretizing continuous spaces - Dartmouth Computer Science, accessed February 3, 2026, https://www.cs.dartmouth.edu/devin/cs89/notes/04discretization.html
19. Zobrist Hashing - DEV Community, accessed February 3, 2026, https://dev.to/larswaechter/zobrist-hashing-72n
20. Vision Based Grasp Planning for Robot Assembly - Diva-Portal.org, accessed February 3, 2026, https://www.diva-portal.org/smash/get/diva2:360454/FULLTEXT01.pdf
21. Automated Process Planning of Sheet Metal Bending with Handling Robot | Request PDF, accessed February 3, 2026, https://www.researchgate.net/publication/317896144_Automated_Process_Planning_of_Sheet_Metal_Bending_with_Handling_Robot
22. Evolutionary Path Planning for Robot Assisted Part Handling in Sheet Metal Bending, accessed February 3, 2026, https://www.sfu.ca/~gwa5/pdf/2003_02.pdf
23. Optimized bending sequences of sheet metal bending by robot | Request PDF - ResearchGate, accessed February 3, 2026, https://www.researchgate.net/publication/238173170_Optimized_bending_sequences_of_sheet_metal_bending_by_robot
24. ARTICLE TEMPLATE Automatic computation of bending sequences for double-head wire bending machines, accessed February 3, 2026,

https://re.public.polimi.it/bitstream/11311/1288423/1/IJCIM2023_ver2_rv.pdf