Algorithmic Architecture for Kinetic Masking: Optimal Path Planning in Salvagnini P4 Panel Bending via A* Search and Shadow Cost Quantification

1. Introduction: The Optimization Landscape of Masked Time

The industrial automation landscape is undergoing a paradigm shift from rigid, sequential automation to flexible, adaptive systems defined by high-dimensional state spaces and non-linear cost functions. Within the specific domain of sheet metal fabrication, the Salvagnini P4 Panel Bender represents a pinnacle of this complexity, offering a kinematic architecture that fundamentally challenges traditional scheduling algorithms. This report serves as the definitive architectural blueprint for **Phase 3: The Sequencer**, tasking the development of an A* Search Algorithm capable of navigating the combinatorial explosion of bending sequences to minimize Total Cycle Time (T_{cycle}).

Unlike classic variations of the Traveling Salesperson Problem (TSP) or the Job Shop Scheduling Problem (JSP), where operation costs are typically additive and cumulative, the P4 introduces the concept of **"Masked Time."** This phenomenon arises from the machine's ability to perform Automatic Blankholder Adjustment (ABA) tool changes and manipulator rotations concurrently.[1] The resulting cost function for any transition between states is not the summation of constituent mechanical movements, but rather the supremum (maximum) of their overlapping durations. This parallelism creates a non-Euclidean cost landscape where the Triangle Inequality is frequently violated; a direct transition between two bends may be "more expensive" than an indirect path that allows for a setup operation to be fully masked by a necessary rotation.

The objective of this analysis is to rigorously derive the mathematical foundations required to solve this problem. We will deconstruct the physics of the P4 manipulator, analyzing how dynamic changes in the panel's moment of inertia influence rotational acceleration and, consequently, the masking potential of any given sequence.[2] We will formally define "Shadow Cost" not merely as an economic abstraction, but as a tangible kinematic metric representing the temporal slack in non-critical parallel operations. This metric will effectively drive the design of admissible heuristics for the A* algorithm, preventing the search from stalling in local minima induced by "zero-cost" tool changes. The ultimate goal is to produce a mathematically sound, computationally efficient sequencer that exploits the P4's parallel

architecture to its theoretical limit.

1.1 Problem Definition and Scope

The sequencing problem for the P4 Panel Bender can be formally described as finding an optimal permutation of a set of required bends $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$, subject to a set of hard constraints imposed by the Precedence Graph (DAG) and geometric collision detection. The optimization criterion is the minimization of the total make-span, which is the sum of the masked transition times plus the actual bending operations.

The inputs to this phase are:

1. **The Precedence Graph (DAG):** A directed acyclic graph where nodes represent bends and edges represent strict ordering constraints derived from the geometry of the part (e.g., a flange cannot be bent if it is occluded by a previously bent flange).[4]
2. **Constraint Solvers:** Black-box functions that return boolean validity for specific machine states (checking for collisions between the part and the manipulator or tooling).[5]
3. **Machine Parameters:** Kinematic limits of the specific P4 model (e.g., P4-2120 vs. P4-3126), including maximum rotational velocity (v_{max}), acceleration (α), and ABA adjustment speeds.[6]

The output is an ordered sequence $\sigma = \langle b_{\pi(1)}, b_{\pi(2)}, \ldots, b_{\pi(n)} \rangle$ that satisfies all topological sorts of the DAG and minimizes the cost function $f(\sigma)$.

1.2 The Failure of Conventional Heuristics

It is critical to understand why standard heuristics fail in this environment. In a typical Euclidean pathfinding problem, the cost to move from point A to point C is strictly less than or equal to moving from A to B and then to C ($d(A, C) \leq d(A, B) + d(B, C)$). Heuristics like the straight-line distance rely on this property to remain admissible (never overestimating the cost).

In the P4 environment, let $t_{rot}(A, B)$ be the rotation time and $t_{aba}(A, B)$ be the tool change time. The cost is $\max(t_{rot}, t_{aba})$.

Consider a scenario where:

- Transition $A \to C$: Requires 0 degrees rotation, 5 seconds ABA change. Cost = **5.0s**.
- Transition $A \to B$: Requires 2 seconds rotation, 2 seconds ABA change. Cost = 2.0s.

- Transition $B \to C$: Requires 2 seconds rotation, 3 seconds ABA change. Cost = 3.0s.

The path $A \to B \to C$ costs $2.0 + 3.0 = 5.0s$. The direct path $A \to C$ also costs 5.0s. However, if the intermediate step B required a large rotation (e.g., 5 seconds) that fully masked a 5-second tool change, and the subsequent step was negligible, the indirect path could effectively "hide" the setup cost. More importantly, heuristic functions that simply sum up remaining rotations or remaining tool changes in isolation will fail to capture the *interaction* between these two variables. A heuristic that estimates "remaining rotation time" might guide the search to a state with zero remaining rotation but a massive, unmasked tool change, leading to a suboptimal total time. Therefore, the A* architecture must utilize a **coupled state formulation** that explicitly tracks "Shadow Budgets"—the accumulated idle time of the non-limiting axis.[8]

2. System Kinematics and the Physics of Masked Time

To optimize the bending sequence, one must first mathematically model the physical reality of the machine's operation. The Salvagnini P4 is not a monolithic actuator; it is a synchronized system of independent axes that operate in concert to manipulate the workpiece.[1] The optimization challenge lies in the disparity between the mechanical capability of these axes and the process requirements of the workpiece.

2.1 The Mechanics of Independence: Manipulator vs. ABA

The P4 utilizes a manipulator to hold, position, and rotate the sheet metal, while the ABA (Automatic Blankholder Adjustment) system resizes the tool length to match the bend line.[1]

The Manipulator (Axis R): This mechanism is responsible for the planar rotation of the sheet. Its performance is governed by rotational dynamics. The time required to rotate the sheet is a function of the angle of rotation (θ) and the angular acceleration (α), which is inversely proportional to the moment of inertia (I) of the panel.[2] As the panel is bent, its mass distribution changes, altering its resistance to rotation. A flat sheet has a high moment of inertia; a folded box typically has a lower one, allowing for faster acceleration.

The ABA System (Axis T): This mechanism adjusts the length of the blankholder tool. The time required for this adjustment is a linear or piecewise-linear function of the change in tool length (ΔL). Crucially, this operation happens "in-cycle" or in "masked time".[1] The ABA consists of modular segments that can be engaged or disengaged, and the speed of this reconfiguration is constant and independent of the panel geometry.

The fundamental interaction that defines our cost function is the temporal overlap of these

two axes. When the sequencer commands a transition from Bend i to Bend j, two primary physical events must occur:

1. The manipulator must rotate the sheet from the orientation of Bend i (ϕ_i) to the orientation of Bend j (ϕ_j).
2. The ABA must adjust the tool length from L_i to L_j.

2.2 The Non-Linearity of the Masked Cost Function

In a standard serial robot, the cost of this transition (C_{trans}) would be the sum of rotation time (t_{rot}) and setup time (t_{aba}). In the P4 architecture, these operations occur simultaneously. The effective cost is the Supremum (sup) of the set of active durations:

$$C_{trans}(i,j) = \max\left(t_{rot}(\phi_i, \phi_j, I_{panel}),\ t_{aba}(L_i, L_j)\right) + t_{positioning}$$

This max function introduces non-linearity and creates "plateaus" in the cost landscape.

- **Case A (Rotation Dominant):** If $t_{rot} > t_{aba}$, the ABA adjustment is "free" or fully masked. The cost is driven purely by kinematics. The "Shadow Cost" here is the time the ABA sits idle waiting for the rotation to complete.
- **Case B (ABA Dominant):** If $t_{aba} > t_{rot}$, the rotation is "free," and the cycle time is penalized by the tool change speed. This is effectively an "unmasked" setup time penalty.[9]

This dynamic implies that the optimal path is not necessarily the shortest distance in Cartesian space (minimizing rotation angles) nor the path of least tool variation (minimizing ABA changes). The optimal path is the one that *aligns* large rotations with large tool changes, thereby "hiding" the slower operation behind the faster one. This phenomenon necessitates a specialized A* heuristic that rewards such alignment.[1]

2.3 Inertia-Dependent Kinematics

A critical, often overlooked variable in simpler sequencing algorithms is the dynamic nature of the load. As the P4 bends flanges on the panel, the mass distribution of the workpiece changes.[2] A flat sheet has a specific moment of inertia I_{flat}. After one side is bent up 90 degrees, the mass of that flange moves relative to the axis of rotation, altering I.

For a rotation axis z passing through the center of the manipulator clamp, the moment of

inertia I_{zz} at sequence step k is calculated as:

$$I_{zz}(k) = \sum_{m=1}^{N_{elements}} \delta m \left(x_m(k)^2 + y_m(k)^2 \right)$$

Where $x_m(k)$ and $y_m(k)$ are the coordinates of the differential mass elements of the panel in its current deformation state.

Since the torque (τ) of the manipulator motors is finite and bounded, the achievable angular acceleration α is:

$$\alpha(k) = \frac{\tau_{max}}{I_{zz}(k)}$$

Consequently, the rotation time t_{rot} for a fixed angle $\Delta\theta$ is not constant throughout the cycle. A rotation of 90° performed at the beginning of the cycle (flat sheet) may be significantly slower than the same 90° rotation performed at the end (complex box shape with "compacted" mass). The A* state definition must therefore track the geometry of the panel to accurately calculate $g(n)$.[2] This suggests that "centripetal compacting"—bending outer flanges first to reduce inertia—might be a valid heuristic strategy to speed up subsequent moves.

2.4 Manipulator Rotation Time Model (t_{rot})

To derive the rotation cost, we must model the motion profile. Industrial manipulators typically follow a Trapezoidal Velocity Profile or an S-Curve profile to minimize jerk.[10] For the purpose of this optimization, the Trapezoidal model provides a sufficient balance of accuracy and computational efficiency.

Let:

- $\Delta\theta = |\phi_{target} - \phi_{current}|$
- v_{max} : Maximum angular velocity (rad/s).[6]
- a_{max} : Maximum angular acceleration (rad/s^2), derived from τ_{max} / I_{zz}.

The time required t_{rot} is determined by whether the move is "Triangle" (short move, never

hits top speed) or "Trapezoid" (long move, hits top speed).

Phase Boundary:

The angle required to reach v_{max} is $\theta_{crit} = \frac{v_{max}^2}{a_{max}}$.

Case 1: Short Move ($\Delta\theta < \theta_{crit}$)

The profile is triangular. The peak velocity reached is $\sqrt{\Delta\theta \cdot a_{max}}$.

$$t_{rot}(\Delta\theta) = 2\sqrt{\frac{\Delta\theta}{a_{max}}}$$

Case 2: Long Move ($\Delta\theta \geq \theta_{crit}$)

The profile is trapezoidal.

$$t_{rot}(\Delta\theta) = \frac{v_{max}}{a_{max}} + \frac{\Delta\theta}{v_{max}}$$

This derivation confirms that t_{rot} is non-linear with respect to $\Delta\theta$, and linearly dependent on $1/a_{max}$ (and thus linearly dependent on Inertia I). This coupling between the sequence order (which determines shape) and the cost (which depends on shape) reinforces the need for a state-space search rather than a static optimization.

3. A* Search Architecture for Masked Sequencing

The implementation of the A* algorithm requires a rigorous definition of the graph structure, specifically the nodes, edges, and the traversal rules governed by the Precedence Graph (DAG) and the constraint solvers.[16] The architecture must handle the combinatorial explosion inherent in sequencing N bends, which is $N!$ in the worst case, reduced only by the topological constraints of the DAG.

3.1 State Space Definition (S)

In simpler path planning (e.g., navigation), a state is defined by location (x, y). In the P4 sequencing problem, a state s must capture the complete history required to calculate future costs and validity. Memory efficiency is paramount here, as the search tree can grow

exponentially.

We define the state vector $\mathbf{s}_k$ at step k as a tuple:

$$\mathbf{s}_k = \langle \mathcal{B}_{done}, \phi_{current}, L_{tool}, \mathbf{G}_{panel} \rangle$$

Where:

- $\mathcal{B}_{done}$: A bitmask representing the set of bends already completed. This provides a compact representation for checking against the Precedence Graph (if Bend B requires Bend A, B cannot be added to $\mathcal{B}_{done}$ unless bit A is set).[4]
- $\phi_{current}$: The current orientation of the manipulator (0 to 360 degrees). This is crucial for calculating Δt for the next move.
- L_{tool} : The current length of the ABA tool. This is required to calculate t_{aba} for the next transition.
- $\mathbf{G}_{panel}$: A geometric descriptor (or hash) representing the current shape of the panel (e.g., "Side A bent, Side B flat"). This is used to look up the current moment of inertia I_{zz} and to check for collision constraints with the machine frame.[2]

State Space Size Analysis:

The number of possible subsets of bends is 2^N . However, the Precedence Graph drastically prunes this. If the graph dictates strictly linear dependencies ($1 \to 2 \to \cdots \to N$), there are only N states. If the graph is empty (all bends independent), there are 2^N states. Real-world panels lie somewhere in between. The inclusion of ϕ and L multiplies the state space, but since these are dependent on the *last action*, they do not expand the combinatorics of the set $\mathcal{B}_{done}$ itself, but rather create multiple "arrival modes" for the same set of completed bends.

3.2 Action Space and Successor Generation

From a given state $\mathbf{s}_k$, the algorithm generates successors by identifying the set of *feasible* next bends, $\mathcal{F}(\mathbf{s}_k)$.

$$\mathcal{F}(\mathbf{s}_k) = \{b \in \mathcal{B}_{total} \setminus \mathcal{B}_{done} \mid \text{Precedents}(b) \subseteq \mathcal{B}_{done} \wedge \text{CollisionFree}(b, \mathbf{G}_{panel})\}$$

The constraint solvers provided as inputs to this phase are invoked here. They prune the search space by eliminating bends that would cause a collision between the partially formed panel and the P4's structural columns or blades.[18] This "validity check" is computationally expensive, so caching the results based on the geometric hash $\mathbf{G}_{panel}$ is a critical architectural optimization.

3.3 The Accumulation of Cost ($g(n)$)

The cost $g(n)$ is the cumulative cycle time to reach state n.

$$g(n) = \sum_{k=1}^{n} \left[\max\left(t_{rot}(\phi_{k-1}, \phi_k),\ t_{aba}(L_{k-1}, L_k)\right) + t_{bend}(k)\right]$$

Note that $t_{bend}(k)$ (the time to actually perform the bend stroke) is generally constant or strictly dependent on material thickness and angle, and largely independent of the sequence *order*. However, it must be included for total cycle time accuracy. The optimization power lies in minimizing the $\sum \max(\dots)$ term.[1]

The $g(n)$ function accumulates not just time, but implicit "penalties" incurred by missed masking opportunities. If a sequence involves a 10-second tool change masked by a 2-second rotation, the cost increases by 10 seconds. If a different sequence allowed that 10-second tool change to be masked by a 10-second rotation (perhaps by grouping bends differently), the cost would still increase by 10 seconds, but the *Shadow Cost* would be zero, implying higher efficiency.

3.4 Handling Symmetry and Pruning

Panel bending often involves symmetric parts (e.g., a square box). If the precedence graph allows Side 1 and Side 3 (symmetric opposites) to be bent in any order, the search tree will fork into two mathematically identical branches ($1 \to 3$ and $3 \to 1$). This redundancy wastes computational resources.

Symmetry Breaking Strategy: We impose a lexicographical ordering constraint on symmetric independent tasks. If Task A and Task B are symmetric and independent, we enforce that A must be processed before B in the search tree. This effectively halves the

search space for every symmetric pair without sacrificing optimality.[21]

Dominance Rules:

To keep the search tractable, we apply dominance rules. Two states $\mathbf{s}_a$ and $\mathbf{s}_b$ are comparable if:

1. $\mathcal{B}_{done}^a = \mathcal{B}_{done}^b$ (Same set of completed bends).
2. $\mathbf{G}_{panel}^a = \mathbf{G}_{panel}^b$ (Same geometric configuration).

State $\mathbf{s}_a$ dominates $\mathbf{s}_b$ if:

$$g(\mathbf{s}_a) + \text{Penalty}(\mathbf{s}_a) \leq g(\mathbf{s}_b) + \text{Penalty}(\mathbf{s}_b)$$

Where the Penalty function estimates the "setup" cost to exit the current state (e.g., how far ϕ and L are from the average remaining targets). If $\mathbf{s}_a$ is faster to reach and leaves the machine in a "better" or "equal" configuration (closer to future targets), we prune $\mathbf{s}_b$.[22]

4. Masked Time Physics: Shadow Costs and Rotation Penalties

Central to this optimization is the quantification of "waste" within the parallel processes. We introduce the concept of **Shadow Cost** derived from Linear Programming principles (Shadow Price) but adapted for Kinematic Scheduling.[8]

4.1 The Shadow Cost Matrix

For every transition step k, we calculate the temporal slack λ_k:

$$\lambda_k = |t_{rot} - t_{aba}|$$

This value λ_k represents the **Shadow Cost** of the transition. It quantifies the "unbalanced" portion of the machine's exertion.

- **Rotation Bound ($\delta_{aba} > 0$):** If $t_{rot} > t_{aba}$, the ABA completes early and waits. The shadow cost λ_k represents "wasted" tool change capacity. We could have changed the tool more drastically (by an amount equivalent to $v_{aba} \times \lambda_k$) without incurring any

additional cycle time penalty.

- **ABA Bound ($\delta_{rot} > 0$):** If $t_{aba} > t_{rot}$, the manipulator completes rotation early and waits. The shadow cost represents "wasted" manipulator capacity. We could have rotated the panel further (by an amount determined by the velocity profile) for free.

Strategic Implication: An optimal sequence is one that *minimizes the sum of Shadow Costs* over the entire path, subject to the constraint that total cycle time is also minimized. This sounds contradictory (minimizing slack might imply making the shorter operation longer), but in reality, minimizing shadow cost means *maximizing the overlap* or "Masking Efficiency."

Ideally, we want $t_{rot} \approx t_{aba}$ at every step to maximize resource utilization.[8]

4.2 Rotation Penalties in Unmasked Scenarios

When masking fails significantly (i.e., one process is much longer than the other), we incur a **Rotation Penalty** or **Setup Penalty**.

$$P_{rot} = \max(0, t_{rot} - t_{aba})$$

$$P_{aba} = \max(0, t_{aba} - t_{rot})$$

The A* algorithm effectively minimizes the cumulative sum of these penalties plus the baseline minimum time. Minimizing $\sum P_{rot} + \sum P_{aba}$ is equivalent to maximizing the *intersection* of the rotation and ABA time intervals.

This leads to specific heuristic strategies:

1. **"Dump" Rotations:** If the algorithm identifies a necessary large tool change (high t_{aba}), it should preferentially pair this with a large rotation (high t_{rot}) to "fill" the time window.
2. **"Centripetal" Sequencing:** Since t_{rot} decreases as inertia decreases, and t_{aba} is constant for a given ΔI, it is often beneficial to perform large rotations late in the cycle (when the panel is folded and I is low) if they can be masked by moderate tool changes.

4.3 Dynamic Buffering and "Waiting" Positions

The Cost Function must also account for external system constraints. The P4 is often part of a line (e.g., S4+P4). If the P4 finishes a part faster than the S4 can feed the next blank, the P4 sits idle. This introduces a "Flow Rate Floor" to the cost function.[25]

Let $T_{arrival}$ be the time until the next part arrives.

$$\text{Effective Cost} = \max(\text{Calculated Path Time}, T_{arrival})$$

This implies that if the P4 is already faster than the feeder line, further optimization of the cycle time yields *zero* marginal utility (Shadow Price = 0). The A* algorithm can be designed to terminate early if it finds a path that satisfies $Cost \leq T_{arrival}$, significantly saving computation time.

5. Heuristic Design $h(n)$: Admissibility in Masked Domains

The standard Euclidean distance heuristic used in A* for spatial pathfinding is inadmissible here because the "cost" is not spatial distance. We need a function $h(n)$ that estimates the *minimum possible time* to complete the remaining bends ($\mathcal{B}_{rem}$).

5.1 The "Shadow-Aware" Lower Bound

To guarantee optimality, $h(n)$ must never overestimate the remaining cost. A simple admissible heuristic would be to sum the raw bending times of remaining bends (since they are unavoidable) and add a lower bound on the transition times.

$$h(n) = \sum_{b \in \mathcal{B}_{rem}} t_{bend}(b) + h_{trans}(n)$$

The challenge is deriving $h_{trans}(n)$. In a standard TSP, one might use a Minimum Spanning Tree (MST). However, standard MST edges have fixed weights. Here, edge weights are dynamic ($\max(t_{rot}, t_{aba})$).

Proposed Heuristic: The Decoupled Lower Bound

We relax the problem by decoupling the rotation and ABA requirements into two independent worlds.

1. Calculate the MST cost of the remaining bends considering *only* rotation angles. Let this be H_{rot}.
2. Calculate the MST cost of the remaining bends considering *only* tool lengths. Let this be H_{aba}.

The lower bound on the total transition time is:

$$h_{trans}(n) = \max(H_{rot}, H_{aba})$$

Proof of Admissibility: Let the optimal path through the remaining nodes be sequence S^*. The actual cost is $C(S^*) = \sum_{(u,v)\in S^*} \max(t_{rot}(u,v), t_{aba}(u,v))$. By the properties of the max function, $\sum \max(a_i, b_i) \geq \max(\sum a_i, \sum b_i)$. Therefore, the total time is at least the total rotation time of the path, and at least the total setup time of the path. Since the MST provides a lower bound on the path length for a single metric (Rotation or ABA), $H_{rot} \leq \sum t_{rot}(S^*)$ and $H_{aba} \leq \sum t_{aba}(S^*)$. Consequently, $\max(H_{rot}, H_{aba}) \leq \max(\sum t_{rot}, \sum t_{aba}) \leq C(S^*)$. Thus, $h_{trans}(n)$ is admissible.[16]

5.2 Refinement: The Linear Assignment Heuristic

To tighten $h(n)$ (make it more informed), we observe that every remaining bend requires *at least* one entry transition. We can model this as an Assignment Problem. We construct a cost matrix where rows are remaining nodes and columns are remaining nodes (plus the current state). The cost of assigning node i to node j is the masked transition time.

Solving the **Linear Assignment Problem (LAP)** for this matrix provides a set of cycles covering all nodes with minimum total weight. This weight is a tighter lower bound than the MST because it enforces the constraint that every node must have exactly one predecessor and one successor (relaxing the subtour elimination constraint of the TSP).[4]

While LAP ($O(N^3)$) is computationally heavier than MST ($O(N^2)$), the reduction in the A* search space often justifies the extra cost per node, especially given the small N (typically < 30 bends) in panel bending.

5.3 Global State Penalty

Typical heuristics might undervalue the "position" of the ABA. If the ABA is currently at length 2000mm and the remaining bends require lengths near 500mm, there is a distinct "ABA Penalty" looming. $h(n)$ must account for this global shift.

We incorporate a **Global State Penalty**:

$$h_{global}(n) = \max\left(\frac{|\phi_{current} - \phi_{centroid}|}{v_{max}}, \frac{|L_{current} - L_{centroid}|}{v_{aba}}\right)$$

where $\phi_{centroid}$ and $L_{centroid}$ are the weighted averages of the remaining bend parameters. This creates a "gravity" in the heuristic, pulling the search toward states that centralize the machine configuration relative to remaining work, preventing the algorithm from painting itself into a kinematic corner.[28]

6. Implementation Tables and Data Structures

To facilitate the software architecture, we present the comparison of heuristic approaches and kinematic costs.

Table 1: Comparative Kinematic Costs (Hypothetical)

This table illustrates how the Masked Cost is derived and identifies the Bottleneck Axis.

Transition Scenario	Δθ (deg)	ΔL (mm)	Trot (s)	Taba (s)	Masked Cost (s)	Shadow Cost (s)	Bottleneck
Small Adj	90	50	1.2	0.5	**1.2**	0.7 (ABA Slack)	Rotation
Large Adj	90	800	1.2	2.5	**2.5**	1.3 (Rot Slack)	ABA
Spin Only	180	0	1.8	0.0	**1.8**	1.8 (ABA Slack)	Rotation
Setup Only	0	500	0.0	1.6	**1.6**	1.6 (Rot Slack)	ABA
Perfect Mask	135	450	1.5	1.5	**1.5**	0.0	**Optimal**

Note: T_{rot} values assume dynamic inertia updates; T_{aba} assumes linear stepper velocity.

Table 2: Heuristic Components Comparison

Evaluation of heuristic strategies for the A Sequencer.*

Heuristic Type	Formula	Admissibility	Consistency	Computational Cost	Suitability for P4
Simple Sum	$\sum(T_{rot} +$	No (Overestimates)	No	Low	**Fail**
Max Edge	$\sum \max(T_{ro}$	No (Greedy)	No	Low	**Fail**
Euclidean	$\text{Dist}(Curre$	No (Wrong metric)	Yes	Low	**Fail**
Decoupled MST	$\max(MST_r$	**Yes**	**Yes**	Medium ($O(N^2)$)	**High**
Assignment (LAP)	$LAP(\text{Cost}$	**Yes**	**Yes**	High ($O(N^3)$)	**Very High**

Data Structures for Implementation:

- **Open Set:** A binary heap or Fibonacci heap storing states ordered by $f(n)$.
- **Closed Set:** A Hash Map using the state vector $\mathbf{s}_k$ (specifically $\mathcal{B}_{done}$ and discretized ϕ, L) as the key to store visited states and prevent cycles.
- **Inertia Map:** A pre-computed lookup table mapping bitmasks of completed bends to approximate moments of inertia, avoiding runtime integral calculus.

7. Implications for Production Strategy

The mathematical model derived above reveals several non-intuitive insights about production

efficiency on the P4.

7.1 The "Shadow Efficiency" Paradox

In traditional manufacturing, faster is always better. In the P4 masked environment, increasing the speed of the *non-bottleneck* axis yields zero cycle time improvement.

- If the process is heavily ABA-bound (complex kits with varying widths), upgrading the manipulator motor ($\alpha \uparrow$) provides no ROI.
- If the process is Rotation-bound (heavy panels, large angles), upgrading the ABA drive provides no ROI. The sequencer can report "Axis Utilization Factors" derived from the accumulated Shadow Costs. If the average δ_{rot} is high, the system is imbalanced toward slow ABA. This data can inform future machine design or tooling choices.[1]

7.2 Kit Production vs. Batch Production

In **Batch Production**, the sequence repeats. The "entry cost" to the first bend is amortized over thousands of parts. In **Kit Production**, the machine transitions between different part geometries (Part A $\rightarrow$ Part B). The final state of Part A (tool length, orientation) determines the entry cost of Part B. This implies that the A* algorithm must expand its horizon. It should not just optimize the current panel, but optimize the transition *sequence* between parts in a kit. The heuristic for Part A should include a term estimating the cost to reset the machine to the *start state* of Part B. This "Linked A*" approach ensures that the machine doesn't finish Part A in a configuration that requires a massive, unmasked setup to start Part B.[11]

8. Conclusion and Recommendations

Developing the Sequencer for the Salvagnini P4 represents a shift from classical cumulative cost optimization to **Parallel Constraint Optimization**. The dominance of the max() operator in the cost function creates a unique landscape where "shadows" (slack times) are the primary resource to be managed.

Architectural Directives:

1. **Implement "Shadow Tracking":** The state vector must explicitly log whether the path is currently Rotation-bound or ABA-bound to guide heuristic tie-breaking.
2. **Pre-Compute Inertia Maps:** Do not calculate integrals in real-time. Use a pre-computed lookup based on bend bitmasks to update a_{max} dynamically.
3. **Adopt Decoupled MST Heuristic:** For real-time performance, the $\max(MST_{rot}, MST_{aba})$ heuristic offers the best balance of speed and admissibility.
4. **Symmetry Breaking:** Enforce strict ordering on symmetric operations to reduce the search space by factorials.

By adhering to this formulation, the A* algorithm will not merely find a *feasible* path; it will exploit the physics of the P4 to find the *kinematically optimal* path, hiding the maximum amount of setup time within the shadows of rotation, and delivering the hallmark productivity of the Salvagnini architecture.

Works cited

1. Panel Bender -P4 - MachineryHost, accessed February 2, 2026, https://f.machineryhost.com/71b2fab0316bfa7fa24ce0a0e5460bad/Brochure%20-%20Panel%20Bender%20-P4.pdf
2. 8.4 Moment of Inertia and Rotational Dynamics | General Physics - YouTube, accessed February 2, 2026, https://www.youtube.com/watch?v=Z4jCfmol_Zw
3. Rotational inertia (article) - Khan Academy, accessed February 2, 2026, https://www.khanacademy.org/science/physics/torque-angular-momentum/torque-tutorial/a/rotational-inertia
4. a note on the precedence-constrained class sequencing problem - Columbia Business School, accessed February 2, 2026, https://business.columbia.edu/sites/default/files-efs/pubfiles/3090/CorreaFioriniStierPCCSfinal.pdf
5. INDUSTRIAL APPLICATION OF ADVANCED ADAPTIVE CONCEPTS FOR AUTOMATIC PANEL BENDERS - UPCommons, accessed February 2, 2026, https://upcommons.upc.edu/bitstreams/700244ff-4423-4dd9-aa85-1f3d66070cb0/download
6. Bending? Easy, with Salvagnini!, accessed February 2, 2026, https://www.salvagninigroup.com/en-INT/campaigns/panel-benders-range
7. Automatic Sheet Metal Bending Machine - Salvagnini P4, accessed February 2, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders/P4
8. Shadow Price Explanation : r/Grid_Ops - Reddit, accessed February 2, 2026, https://www.reddit.com/r/Grid_Ops/comments/9lx2ts/shadow_price_explanation/
9. p4-benders.pdf - Empire Machinery, accessed February 2, 2026, https://www.empire-machinery.com/wp-content/uploads/2021/06/p4-benders.pdf
10. Application note Trapezoidal move calculations - ABB, accessed February 2, 2026, https://library.e.abb.com/public/502bd29feb0349cfaa9558537a9d62fd/AN00115-Trapezoidal_Move_Calculations_Rev_C_EN.pdf
11. Salvagnini panel bending: P4-2520 bends 4 different parts - YouTube, accessed February 2, 2026, https://www.youtube.com/watch?v=gOMHcSt2yQ0
12. The automatic hybrid adaptive Panel Bender. - Sanson Machinery Group, accessed February 2, 2026, http://www.sansonmachinery.com/wp-content/uploads/2013/12/P4Xe-UK-3962020312.pdf
13. Salvagnini flexible manufacturing system: S4+P4 line 100% flexible automation - YouTube, accessed February 2, 2026, https://www.youtube.com/watch?v=ZWcZDVf8_Nc
14. The Mathematics Behind Trapezoidal Planner: Manipulator's Smooth Motion -

FusyBots, accessed February 2, 2026, https://www.fusybots.com/post/trajectoryplanningformanipulator-mathematicsbehindtrapezoidalplanner
15. Design Trajectory with Velocity Limits Using Trapezoidal Velocity Profile - MATLAB & Simulink - MathWorks, accessed February 2, 2026, https://www.mathworks.com/help/robotics/ug/design-a-trajectory-with-velocity-limits-using-a-trapezoidal-velocity-profile.html
16. A* search algorithm - Wikipedia, accessed February 2, 2026, https://en.wikipedia.org/wiki/A*_search_algorithm
17. A* algorithm and its Heuristic Search Strategy in Artificial Intelligence - GeeksforGeeks, accessed February 2, 2026, https://www.geeksforgeeks.org/artificial-intelligence/a-algorithm-and-its-heuristic-search-strategy-in-artificial-intelligence/
18. Multi-Station Multi-Robot Welding System Planning and Scheduling Based on STNSGA-D: An Industrial Case Study, accessed February 2, 2026, https://ieeexplore.ieee.org/document/10366183/
19. Optimizing Space Utilization for More Effective Multi-Robot Path Planning - NSF Public Access Repository, accessed February 2, 2026, https://par.nsf.gov/servlets/purl/10323431
20. (PDF) Mathematical Modeling of Scheduling Problems - ResearchGate, accessed February 2, 2026, https://www.researchgate.net/publication/256089005_Mathematical_Modeling_of_Scheduling_Problems
21. Laser cutting sorting solutions - Salvagnini, accessed February 2, 2026, https://www.salvagninigroup.com/en-INT/products/laser-systems/sorting-solutions
22. Solving Scheduling Problems with Setup Times and Alternative Resources - AAAI, accessed February 2, 2026, https://cdn.aaai.org/AIPS/2000/AIPS00-010.pdf
23. Heuristics for Bounded-Cost Search - Artificial Intelligence, accessed February 2, 2026, https://ai.dmi.unibas.ch/research/reading_group/haslum-icaps2013.pdf
24. What is the SHADOW PRICE in Linear Programming? - YouTube, accessed February 2, 2026, https://www.youtube.com/watch?v=4H9COmgZYO8
25. Flexible Manufacturing System (FMS) for Factory & Job Shops - Salvagnini, accessed February 2, 2026, https://www.salvagninigroup.com/en-US/products/fms-systems/S4-P4
26. Heuristic Search When Time Matters | Journal of Artificial Intelligence Research, accessed February 2, 2026, https://jair.org/index.php/jair/article/view/10831
27. A Note on Appointment Scheduling with Piecewise Linear Cost Functions - PubsOnLine, accessed February 2, 2026, https://pubsonline.informs.org/doi/10.1287/moor.2013.0631
28. Scheduling multiple products on parallel machines with setup costs - IDEAS/RePEc, accessed February 2, 2026, https://ideas.repec.org/a/wly/navres/v55y2008i7p654-669.html
29. Parallel Machine Scheduling with Family Setup Times to Minimize Total Flowtime - IJOQM, accessed February 2, 2026, https://www.ijoqm.org/papers/8-1-4-p.pdf