Innovation Series https://www.innoseries.com

Applicability Boundaries and an Evaluation Protocol for a Differential Heuristic in Segmented SIPP for Risk-Aware 3-D UAV Path Finding

Jingyu Li[1], Juan Pan[2,*]

[1] China Academy of Telecommunications Technology, Beijing, China

[2] China Academy of Information and Communications Technology, Beijing, China

Corresponding Author: Juan Pan (2087800775@qq.com)

Abstract: This paper reports two things together: a lower bound for segmented safe-interval path planning (SIPP), and the audit protocol that decides whether the bound earns its keep. The setting is three-dimensional multi-agent path finding (MAPF) over static risk-weighted edges. For each agent a, one reverse Dijkstra pass from its own goal g_a fills a table $H_a[x]$ equal to the exact static spatial distance $C^*_{\text{space}}(x)$. The table has a single source: it answers "cost from s to g_a" and holds no entry for "cost from s to an intermediate via-point v". The bound $H_{\text{diff}}(s,v) = H_a[s] - H_a[v]$ closes that gap at no second table, and admissibility and consistency are proved on the static spatial graph. On the 18-agent benchmark, low-level expansions fall from 98926 to 19641 (-80.1%). Over 24 Latin-hypercube scenes the median saving per high-level node is 3.02-fold (quartiles 1.80, 3.91). Under pure CBS at $w = 1.0$ the net factor is 0.99—no gain, and we report it as such. Two boundaries surfaced during the audits. Adding a turn cost while the low-level state carries only the voxel position costs 0.38%–0.95% of low-level optimality; an inadmissible Euclidean control returns feasible paths 16.15% above the optimum at the root. Section 5 states the protocol in full. Focal-CBS solves 24 of 24 scenes under the tested budgets; prioritized planning solves 4. The a posteriori ratio is $\rho \leq 1.000434$ with the prioritized-planning cost as numerator. Every exactness claim concerns the static spatial graph; optimality for general time-varying SIPP is not established here.

Keywords: Multi-UAV path planning; Conflict-based search; Safe-interval path planning; Admissible heuristic; True-distance heuristic; Differential lower bound; Evaluation protocol; Benchmark design

1. Introduction

Low-altitude emergency logistics and disaster reconnaissance send several unmanned aerial vehicles (UAVs) through building-dense three-dimensional space, where smoke and wind fields vary from cell to cell. Multi-agent path finding (MAPF) over such a space prices every edge with a risk-aware additive cost. Grid search normally leans on a metric Euclidean distance. Here that distance is measured in metres while the objective is measured in cost, so the two units do not match and the distance stops being a lower bound. A planner that trusts it terminates with a path in hand

and no guarantee behind it. Terrain makes the coupling worse. One narrow gate forces the agents to interact, and every high-level node then pays for a fresh single-agent search.

Write $C^*(x)$ for the true minimum cost of reaching agent a's goal from state x under that same objective, and $h(x)$ for what the low-level search uses in place of it. The estimate is admissible when $h(x) \le C^*(x)$ everywhere, consistent when $h(x) \le \omega(x, x') + h(x')$ on every edge. Tightness is the ratio h/C^*; the value 1 means equality state by state.

Conflict-based search (CBS) and its bounded-suboptimal variant ECBS remain the standard two-level frameworks [1,2]. Whatever the low level returns, both inherit; the low-level cost convention therefore carries more weight here than it usually would. On the risk-aware objective an unscaled metric Euclidean heuristic overestimates the static-space cost on 99.973% of audited states, with mean ratio $\mathbb{E}[h/C^*] = 4.231$. Paths come back feasible. Low-level optimality is quietly gone. A "solved" flag, alone, certifies nothing.

What follows is not a new shortest-path algorithm. The reverse-Dijkstra step is a known construction [3,4]. What this paper adds is narrower: (i) the conditions under which that construction holds on static nonnegative directed edge weights; (ii) the single-source limitation that segments introduce; (iii) a differential bound closing that gap without a second table, with a static-graph proof and an end-to-end break-even condition; (iv) the audit protocol: 24 sampled scenes, four audit families, and negative controls run on purpose, the 'metric' variant among them.

2. Related Work

2.1 CBS-style MAPF Solvers

CBS resolves conflicts by branching on agent–time constraints in a high-level tree and handing each branch to a constrained single-agent search [1]. ECBS runs focal search on the high level, trading optimality for speed [2]. Safe-interval path planning (SIPP) carries forbidden time intervals into the low-level search [6]. Continuous-time and kinematic MAPF relax the temporal and execution models [7–9]. Prioritized planning (PP) plans agents one after another in a fixed order; feasibility and returned cost both move with that order, and completeness is not guaranteed in general [10,11].

2.2 Heuristics: from Admissible Lower Bounds to True-distance Heuristics

Memory-based heuristics tabulate exact distances from chosen pivots and read off lower bounds for other terminals through the triangle inequality. Sturtevant et al. give the general framework [3]. The differential special case—where the bound is the difference of two tabulated distances—traces back to Ng and Zhang [5] and to the ALT construction of Goldberg and Harrelson [4]. What we use is the full-table, goal-pivot instance on a directed risk-weighted spatial graph: one table per agent, one source per table. Constraint-aware heuristic caching has recently been studied for MAPF, where the cached value may depend on a constraint context [12]. Our table is not of that kind. It is purely spatial, and constraints only ever weaken it. UAV path planning more broadly uses weighted A* or metaheuristics carrying length, threat, altitude and energy terms [13–15]; those objectives rarely carry the low-level optimality an optimal CBS needs, so they are not baselines here.

2.3 Benchmark and Evaluation Practice

Wall clock is the default efficiency metric in MAPF papers; Section 5 gives the reason node counts replace it here. Latin hypercube sampling (LHS) generates the scenarios in preference to independent uniform draws, because stratifying the five factors lowers the variance of the estimated

median gain at a fixed budget of 24 scenes [17]. Audit-style validation—checking a heuristic against a reference value computed offline over the entire state space rather than only on solved instances—is, to our knowledge, rarely applied to MAPF heuristics. It is also the part of our protocol we would most like to see adopted elsewhere.

3. Problem Formulation

3.1 Scene and Discretization

Building data from an urban subregion are discretized at 10-m horizontal resolution into a $64 \times 64 \times 24$ grid. The layer-centre span is roughly $640 \times 640 \times 57.5$ m, over 85643 free voxels and 12.9% hard obstacles. At each step an agent moves to one of the 26 neighboring voxels or stays put. A feasible-transition predicate removes diagonal transitions that clip building corners. A no-fly wall with a two-cell (20-m) gate forms the bottleneck. The signed-distance-field (SDF) safety distance is 12 m, which exceeds the roughly 10 m from the gate centerline to either wall; passing the gate therefore creates conflict and SDF risk at the same time.

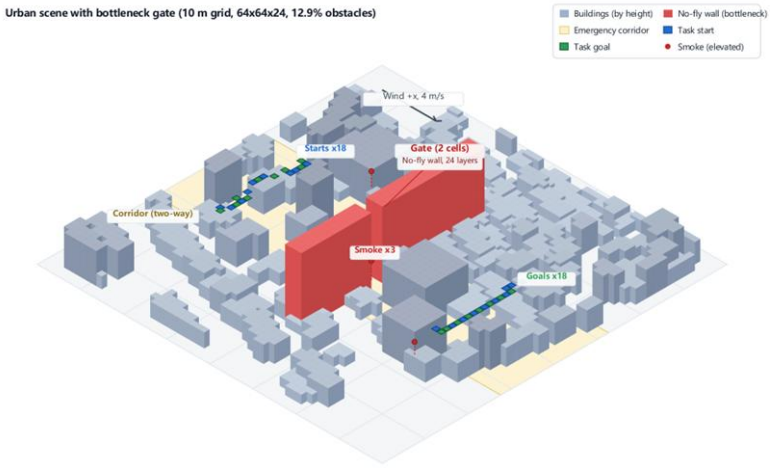


Figure 1: Urban Low-altitude Scene With the Bottleneck Gate.

3.2 Risk-aware Edge Cost

Let u be the current voxel, v the successor, and $\Delta = v - u$ the physical displacement. The directed edge weight is:

$$\omega(u, v) = c(v)\|u - v\| + \phi(v, \Delta) \geq 0 \tag{1}$$

Where

$$\begin{gathered} c(v) = w_{\text{len}}\eta_{\text{len}} + w_{\text{sm}}\eta_{\text{sm}}\rho_{\text{smoke}}(v) + w_{\text{sdf}}\eta_{\text{sdf}}\text{viol}(v)^2 + w_{\text{h}}\eta_{\text{h}}\mathbf{1}_{\mathcal{B}}(v); \\ \phi(v, \Delta) = w_{\text{w}}\eta_{\text{w}}\|\Delta\| \\ \max(0, -\cos\theta)\|W(v)\| + + w_{\text{v}}\eta_{\text{v}}|\Delta_{\text{z}}| \end{gathered} \tag{2}$$

Here ρ_{smoke} is normalized smoke concentration, $W(v)$ the wind vector, and θ the angle between the horizontal displacement and the wind. The SDF violation ratio is $\text{viol}(v) = \max 0, (d_{\text{safe}} - d(v))/d_{\text{safe}}$. Every traversal occupies one time step, and the calibration coefficients absorb that fixed per-step duration. The implementation carries no explicit airspeed, so the wind term is a wind-exposure proxy in equivalent-metre units rather than a physical dose; η_{w} takes up the conversion. Waiting costs $c_{\text{wait}} = 0.3$ per time step. Because every edge weight contains

the length term $w_{\text{len}}\eta_{\text{len}}\|u-v\|$, the scalar $c_{\min} \triangleq w_{\text{len}}\eta_{\text{len}} = 0.14$, in cost per metre, lower-bounds the cost of any move spanning a Euclidean distance $\|u-v\|$; that scalar is the unit conversion behind the Euclidean fallback of Section 4. The start voxel is not charged. Trajectory cost is $J(\pi) = \sum\omega$ up to arrival. The turn term appears as an evaluation metric only: it depends on three consecutive states and cannot sit inside J.

The six objective weights are $(0.14,0.08,0.08,0.32,0.12,0.03)$ for length, smoke, wind, SDF, vertical motion and height-band penalty, with scale factors $(1.0,5.0,0.3,14.0,1.5,0.5)$; the turn weight is 0.23. Two of the six involve altitude and differ in kind. The vertical term penalizes the displacement $|\Delta_z|$ of a move, so it belongs to ϕ, which reads the displacement. The height-band term is an indicator of an absolute altitude band, so it belongs to $c(v)$, which reads the voxel. These are the configuration used throughout; they are not a proposed weighting rule. Because the wind term depends on (v,Δ), the graph is directed, and a reverse search must keep the original edge weight on the transposed edge.

3.3 Weights

The resulting unit-value equivalents $w_\star\eta_\star$ are 0.14, 0.40, 0.024, 4.48, 0.18 and 0.015, spanning about a factor of 300 with the safety term on top at 4.48. The turn weight $w_{\text{turn}} = 0.23$ stays out of J and serves as an evaluation metric only, so the six $w_\star$ sum to 0.77 rather than 1.

The base weights were fixed by configuration, not by argument. We swept each weight by ± 50 (groups S1–S3 over smoke, SDF and vertical) and ran a marginal-exchange-rate analysis on the smoke weight: $|\Delta\text{dose}/\Delta\text{cost}|$ falls from 0.224 to 0.097 to 0.042 as w_{smoke} goes 0.04→0.08→0.12, and the knee sits near 0.08, the value we kept. An analytic-hierarchy check on the resulting coefficients returns CR = 0.157, above the usual 0.10 threshold; its ordering (SDF>length>smoke>vertical>wind>height) is reported as a sanity check only, not as a validated calibration. The calibration failed to meet its own criterion, and we say so plainly.

4. Method

4.1 Segmented SIPP and the G reweighting

The low level solves a minimum-cost path under vertex and edge forbidden time instants, using static A^* where no constraint applies and SIPP otherwise. A high-level split can impose a via-point: the agent must occupy v at time t_v. Several via-points split the trajectory into segments.

For an intermediate segment write $G = q - tc_{\text{wait}}$, where q is accumulated cost and t is time. The potential shift $p(x) = -tc_{\text{wait}}$ produces this reduced cost—the classical reweighting that makes edge weights nonnegative before Dijkstra runs [16]. Reaching the mandated time t_v costs $G + t_v c_{\text{wait}}$ in total; since the second term is constant, minimizing the total and minimizing G are the same problem. With $G(x) = q(x) - t(x)c_{\text{wait}}$ for a search state x, an action contributes $\Delta G = \Delta q - \Delta t c_{\text{wait}}$. Waiting $\tau \geq 0$ steps gives $\Delta q = \tau c_{\text{wait}}$ and $\Delta t = \tau$, so $\Delta G = 0$. A move gives $\Delta q = \omega(u,v)$ and $\Delta t = 1$, so $\Delta G = \omega(u,v) - c_{\text{wait}}$. Under (G1) every increment is nonnegative. Dijkstra then applies, and the first pop of a state at the mandated time t_v is already optimal, so the remaining OPEN entries need not be scanned. The optimal cost is recovered as $G_{\min} + t_v c_{\text{wait}}$. Note that G is measured against an all-waiting baseline and may be negative; only the nonnegativity of its increments is needed, and the increment does not depend on how long the agent waited beforehand. The implementation admits no move spanning more than one step, so that generalization is not needed. Two conditions gate the transform. (G1) requires every nonzero move to satisfy $\omega(u,v) - c_{\text{wait}} > 0$—exactly the nonnegativity condition Johnson's reweighting imposes on the reweighted graph. (G2) requires the

agent to be able to wait at v from its arrival time to t_v under all vertex, edge and dynamic-occupancy constraints, checked as a conjunction over the whole waiting interval: every intermediate voxel free of vertex constraints at the corresponding instants, every traversed edge free of edge constraints, and the voxel itself free of dynamic occupancy throughout.

Figure 2 shows the resulting segment structure and how far heuristic quality differs between the two segment types.

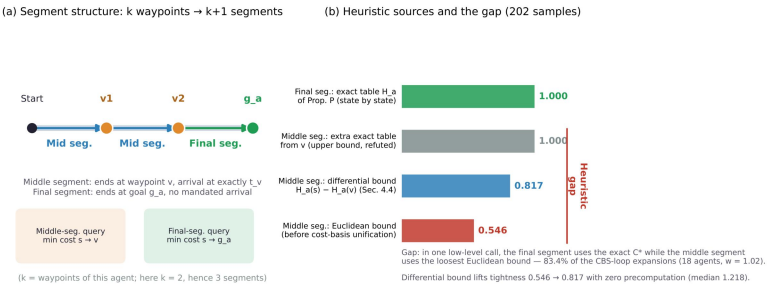


Figure 2: Segmented SIPP and the heuristic gap between the two segment types. (a) k via-points split a trajectory into $k+1$ segments; the last segment ends at the goal with no mandated arrival time. (b) Heuristic tightness (h/C^*) on 202 sampled (agent, segment endpoint, state) cases.

4.1 Exact distance on the static spatial graph.

Proposition 1

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \omega)$ be a directed graph with time-independent nonnegative edge weights and additive path cost. If $C^*_{\text{space}}(x)$ is the minimum cost from x to agent a's goal g_a, then a single-source Dijkstra search from g_a on the transposed graph, keeping each original edge weight, yields

$$H_a[x] = C^*_{\text{space}}(x), x \in \mathcal{V} \tag{3}$$

The table is admissible, consistent and tight on the static spatial graph.

This is a static-space result. Under SIPP constraints the table stays a lower bound, $H_a[x] \le C^*_{\text{SIPP}}(x, I, t)$, but equality generally fails. A time-varying field, negative edge weights, or a state-dependent objective each needs a different construction. Those are applicability boundaries, not claims that no shortest-path method exists for such cases.

Table 1: Applicability conditions and core audit results

ID	Condition / audit	Meaning	Checks	$\mathbb{E}[h/C^*]$	Result
R1	Static nonnegative edge weights	Reverse Dijkstra on the spatial graph	1541574 states	1.000	0 violations; exact spatial distance
R5	Metric Euclidean control	Units do not match the risk objective	1541574 states	4.231	1541154 violations (99.973%); inadmissible
R6	Differential bound	$H_a[s] - H_a[v]$ for an intermediate terminal	6080653 spatial states	0.764	0 violations on the static spatial graph
B1	Turn objective without	Low level minimizes $\sum\omega$ but reports turn	41622498 (state,	0.952	h admissible on the static graph;

direction state	cost	direction) pairs	low level suboptimal by +0.38%–+0.95%

4.2 Query Gap in Segmented Search

The table H_a covers all 85643 free states, but it is single-source. Its only source is agent a's goal g_a; state by state it returns "the cost from s to g_a" rather than "from s to an arbitrary third point v". The terminal of an intermediate segment is exactly such a via-point $v \neq g_a$, so the table alone cannot answer it. Two routes reach the query at equal precision: build a second table with v as source, or derive a lower bound to v out of H_a. Section 4 shows that the first route does not pay here, which is why we take the second. Together with the two segment types of Section 4, this gap is the structural reason behind the heuristic gap in Figure 2.

4.3 Differential Bound for an Intermediate Terminal

The goal table cannot answer the cost from s to an arbitrary via-point v, so we reuse it:

$$\begin{aligned} H_{diff}(s,v) &= \boldsymbol{H_a}[s] - \boldsymbol{H_a}[v]; \\ h_{seg}(s) &= max\{c_{min}\|s-v\|, H_{diff}(s,v)\}; \\ h_g(s,t) &= max\{0, h_{seg}(s) - (t_v - t)c_{wait}\} \end{aligned} \tag{4}$$

The difference is used only when both entries are finite, otherwise, the implementation falls back to the Euclidean lower bound.

Theorem 1

On the static spatial graph, $H_{diff}(s,v) \leq C^*_{space}(s \to v)$. For every edge $s \to s'$, $H_{diff}(s,v) - \boldsymbol{H_{diff}}(s',v) \leq \omega(s,s')$. Taking the maximum with another admissible and consistent lower bound preserves both properties.

Proof sketch: The triangle inequality gives $H_a[s] \leq C^*_{space}(s \to v) + \boldsymbol{H_a}[v]$; rearranging proves admissibility. Subtracting the same $H_a[v]$ from the one-step triangle inequality proves consistency, and the maximum property follows directly. On the G-graph the time-adjusted expression of (4) is admissible and consistent under (G1)–(G2). The audit reported here covers the static spatial graph only and is not a complete time-stamped SIPP audit.

The next lemma carries the conclusion onto the G-graph of Section 4, the space the implementation actually searches.

Lemma 1

Under (G1), let the edge weights on the G-graph be $c_G(x,x') = \Delta G$. For every edge $x = (s,t) \to x' = (s',t')$ we have $h_g(x) \leq c_G(x,x') + \boldsymbol{h_g}(x')$ and $h_g(x) \leq G^*(x)$, the minimum remaining G value.

Proof

Move edges have $c_G = \omega(s,s') - \boldsymbol{c_{wait}}$; the claim follows from $h_{seg}(s) \leq \omega(s,s') + \boldsymbol{h_{seg}}(s')$ and $t' - t = \tau \geq 1$; if $h_g(x') = 0$, substituting $h_{seg}(s') \leq (t_v - t')c_{wait}$ gives the same result. Wait edges have $c_G = 0$, and h_g is non-decreasing in t, so $h_g(x) \leq \boldsymbol{h_g}(x')$. For admissibility, $G^*(x) = \sum_{move}(\omega - c_{wait}) \geq \boldsymbol{h_{seg}}(s) - n_{\text{move}}c_{wait}$ with $n_{\text{move}} \leq t_v - t$, hence $G^*(x) \geq \boldsymbol{h_{seg}}(s) - (t_v - t)c_{wait}$; (G1) adds

$G^*(x) \geq 0$, whose maximum is $h_g(x)$.

Three implementation points matter. H_a must be the original table, returning empty for unreachable states, because a generic heuristic interface would fall back to the Euclidean distance, raise $H_a[v]$ and destroy admissibility. When $H_a[s]$ or $H_a[v]$ is unreachable the difference is undefined and the Euclidean term takes over. A negative H_{diff} is truncated by the maximum. $H_a[v]$ is computed once per segment, and no additional table is built. Figure 3 shows the tight and degenerate cases.

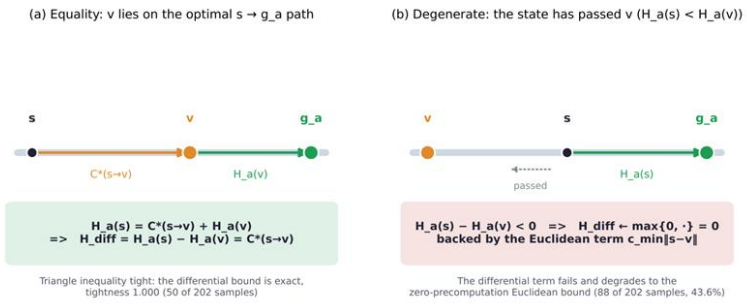


Figure 3: Two regimes of the differential bound. (a) When v lies on the optimal $s \to g_a$ path, the triangle inequality is tight and the bound equals the exact cost (50 of 202 samples). (b) Once the state has passed v, the difference turns negative and the Euclidean term $c_{min}\|s - v\|$ is used instead (88 of 202 samples, 43.6%).

Algorithm 1 restates the construction in the symbols of (3)–(4). Part (a) relaxes the in-edges of the settled state x on the transposed graph, keeping the original weight $\omega(u, x)$, so the settled value is $C^*_{space}(x)$. Part (b) is the query of (4) with the two fallbacks above.

Algorithm 1. Segment-terminal heuristic. Part (a) runs once per agent in $O(|\mathcal{E}| + |\mathcal{V}|log|\mathcal{V}|)$ time and $O(|\mathcal{V}|)$ space; Part (b) answers one query in $O(1)$ time.

Require: Directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \omega)$ with $\omega \geq 0$; agent goal g_a; segment terminal (v, t_v); query state (s, t); constants $c_{\min}$, c_{wait}.
Ensure: Table H_a with $H_a[x] = C^*_{\text{space}}(x)$; heuristic value $h_g(s, t)$.
Part (a): reverse Dijkstra, once per agent
$H_a[x] \leftarrow \infty$ for all $x \in \mathcal{V}$; $H_a[g_a] \leftarrow 0$; $Q \leftarrow \{(0, g_a)\}$
while $Q \neq \emptyset$ do
  $(d, x) \leftarrow$ extract-min(Q)
  if $d \geq H_a[x]$ then
    discard x and continue
  end if
  $H_a[x] \leftarrow d$
  for each in-edge $(u, x) \in \mathcal{E}$ do
    if $d + \omega(u, x) < H_a[u]$ then
      $H_a[u] \leftarrow d + \omega(u, x)$; insert $(H_a[u], u)$ into Q
    end if
  end for
end while
Part (b): differential query, once per expansion

16: if $H_a[s] = \infty$ or $H_a[v] = \infty$ then
17: $h_{\text{seg}}(s) \leftarrow c_{\min}\|s - v\|$
18: else
19: $h_{\text{seg}}(s) \leftarrow \max\{c_{\min}\|s - v\|, H_a[s] - H_a[v]\}$
20: end if
21: $h_g(s,t) \leftarrow \max\{0, h_{\text{seg}}(s) - (t_v - t)c_{\text{wait}}\}$
22: return $h_g(s,t)$

4.4 When not to build an exact table for the segment terminal

A second table with v as source costs one reverse Dijkstra, settling $|S_{\text{free}}| = 85643$ states. What it buys is $n(v)\Delta n(v)$, the number of reuses times the expansions saved each time, so break-even needs $n(v)\Delta n(v) > |S_{\text{free}}|$. The 520 intermediate segments of the 18-agent benchmark reduce to 27 distinct positions, and none clears the condition. Grant the optimistic assumption that intermediate-segment expansions drop to zero: 27 tables (2312361 settled states) still overshoot the gain upper bound (81421 expansions) by 28.4-fold. The reason is structural: the busiest position is the least reused, while the table cost is indifferent to reuses. Hence the differential route, which costs no extra precomputation.

4.5 High-level search and end-to-end gain

The high level keeps a constraint tree whose node cost is the sum of all agents' low-level ω costs; OPEN is ordered by that sum. When the suboptimality factor is $w > 1$, focal search switches on [2]: a focal list holds the nodes with $\Sigma\omega \le w \cdot \min_{\text{OPEN}} \Sigma\omega$ and is popped by (number of conflicts, $\Sigma\omega$, Σturn, creation order). The secondary key is the conflict count rather than a secondary heuristic as in ECBS, which biases the search toward nodes that are nearly conflict-free. Both levels must share one cost convention. Since the turn cost depends on three consecutive steps and cannot be an edge weight, the high-level node cost takes $\Sigma\omega$ alone; otherwise the negative increment "adding a constraint makes the cost drop" appears and branch and bound collapses. We measured 2650 such negative increments under an inconsistent convention; after unification the count is 0.

The bound acts only on intermediate segments, so its gain depends jointly on constraint accumulation, single-segment search size, and how the high level walks the tree. On the focal main benchmark intermediate segments are 83.4% of loop expansions; enabling the bound takes them from 81421 to 2136 (-97.4%) while the high-level node count and the solution cost stay put, and total low-level expansions fall from 98926 to 19641 (-80.1%). Under pure CBS ($w = 1.0$) the picture reverses. Intermediate segments are only 31.7%; they save 7367 expansions, the last segment adds 7853 because low-level calls rise from 814 to 974, and the net factor is 0.99. Over 24 scenes the improvement in low-level expansions per high-level node runs from a minimum of 1.22 through lower quartile 1.80, median 3.02, upper quartile 3.91, to a maximum of 12.83, with no scene slower. These numbers support the claim that the gain comes from intermediate-segment expansion volume inflated by via-point constraints. They do not support "the differential heuristic speeds CBS up by $N\times$". Figure 4 gives the per-scene distribution.

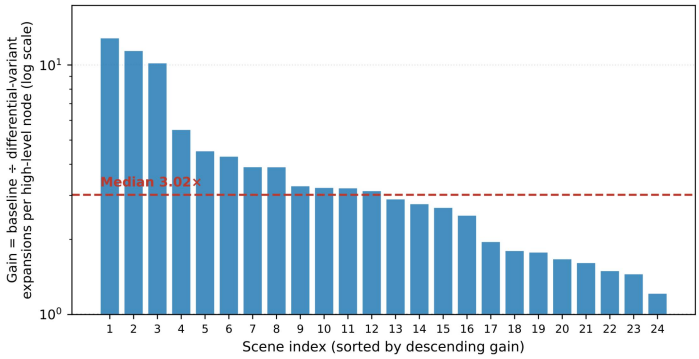


Figure 4: Per-scenario search gain of the differential bound over 24 LHS scenarios, paired per scenario and shown on a log scale. The median gain is 3.02-fold.

5. Evaluation Protocol

5.1 What an Audit Must Certify, and what it Cannot

An audit here compares h against a reference value over a stated set of states and reports violations. Three properties have to be kept apart, because conflating them is the most common way a benchmark misleads. *Admissibility* is a statement about h on every state. *Low-level optimality* is a statement about what the planner returns, and it depends on the search state, not only on h. *Feasibility* is the weakest of the three: a path that respects the constraints. Row B1 of Table 1 separates the first two—h passes with 0 violations among 41622498 pairs while the returned solution is 0.38%–0.95% off. The 'metric' control separates the last two, returning a feasible path that is 16.15% suboptimal at the root. A report that lists only "solved" collapses all three into one bit.

The authority of an audit stops at its state space. Row R6 covers 6080653 static-spatial-graph states, not time-stamped SIPP states; it certifies admissibility and consistency on the spatial graph, and the SIPP side follows from the remark after Proposition 1 rather than from a direct check.

5.2 Sample Sizes and Decision Rules

The static audit covers 18 agents $\times$ 85643 free states $=$ 1541574 states. That is the entire state space rather than a sample; a sampled audit would have left the 99.973% violation rate for R5 undetectable, since the rate is an aggregate over states the heuristic never touches on solved instances. The direction-aware B1 audit multiplies by 27 incoming directions, giving 41622498 (state, direction) pairs. R6 audits 71 agent–terminal combinations. The decision rule is a hard threshold, not a trend: 0 violations on the audited set, with any nonzero count treated as failure. On a search that expands millions of nodes an inadmissible estimate is eventually caught, and the resulting cost is unbounded.

Five heuristic variants are compared. 'cost', 'tight', 'risk' and 'exact' are admissible with increasing tightness. 'metric' is an inadmissible negative control included on purpose, since without it the four admissible variants would agree and the agreement would look like a proof. The reference implementation was cross-checked on 5200 sampled adjacency and cost cases; we report that as cross-validation, not as a formal proof of code independence, which we cannot give.

5.3 Primary Metric and its Alternatives

Expanded nodes are the primary efficiency metric. Wall-clock time was rejected after repeated runs on one machine varied by up to $3.6\times$—wider than any effect under test. Node counts are deterministic for a fixed search order and therefore comparable across variants on the same instance.

They are not comparable across instances or across papers, and we do not use them that way. Solution cost is reported alongside, but the objective J is configuration-specific and is not used for cross-paper numerical comparison. Where a ratio is quoted we name its numerator and denominator explicitly, because ρ computed from different numerators differs in the fourth decimal.

5.4 Reproducibility

The 24 scenarios are generated by Latin hypercube sampling over agent count, gate width, risk intensity, wind direction and wind speed with seed 2026; the PP permutations are drawn once per scene under seed $20260916 + i$ for scene i. Both seeds are stated so that the scenario set and the order set can be regenerated exactly. The main benchmark is a single 18-agent instance with a two-cell gate, a smoke source at $(33,35)$ and eight root conflicts. It is a fixed, hand-built instance rather than a draw from the sampled family, and results on it should not be read as averages.

6. Experiments and Audits

6.1 Setup

Table 2 reports the focal configuration ($w = 1.02$) under a budget of 3000 high-level nodes or 600 s.

Table 2: 18-agent main benchmark with focal search ($w = 1.02$), five heuristic variants in columns.

Metric	cost	metric	tight	risk	exact
Solved	No	Yes †	No	Yes	Yes
High-level nodes	4	13	33	782	110
Low-level exp.	1,259,885	413,487	1,045,809	1,016,216	19,641
Root expansions	599,315	52,818	164,663	25,853	1,294
Solution cost	–	2187.358732	–	1881.345054	1881.607782
Time (s)	662.8	113.2	602.9	450.1	14.0

†Feasible but not optimal, because the low-level heuristic is inadmissible.

6.2 Main Benchmark

The four admissible variants return the same root cost, 1880.008549. The exact table therefore removes futile search rather than changing the quality of the unconstrained solution. In the segmented pure-CBS measurement the loop-expansion average is 20.4 for ‘exact’ against 327.6 for ‘risk’, a factor of 16.1; the root audit alone cannot establish the segmented-SIPP gain, and this loop-level comparison is the relevant low-level measure. The inadmissible ‘metric’ control expands only 52818 nodes at the root, an order of magnitude fewer than ‘cost’, and so looks “faster”. It is not: its root cost sits 16.15% higher. The optimality criterion is the agreement of the four admissible root costs, not the flag.

6.3 Prioritized Planning and a Posteriori Bound

Six priority orders are tested per LHS scene: the natural task order and five reproducible random permutations, 144 orders in total. Focal-CBS solves 24 of 24 scenes. PP solves 4 of 24 scenes and 8 of 144 orders, and in the six scenes above 412 CBS nodes it finds no solution under any tested order. These are results under the stated orders and budgets; they are not a claim that the underlying MAPF instances are infeasible. On the single 18-agent benchmark PP’s best feasible cost is 1881.303155 with

1664 low-level expansions, against 1881.607782 and 19641 expansions for CBS. Sorted orders were not tried, so the solve rate is performance under random orders. PP's failure here is not a timeout: the total wall clock per scene over six orders peaks at 69.5 s, far below the 150 s per-order and 600 s per-scene budgets. On this instance PP's best cost is in fact no worse than CBS's and uses 11.8-fold fewer low-level expansions, so the claim is not "faster than PP" but a concrete characterization of PP incompleteness where a solution must be guaranteed.

For this benchmark the smallest retained OPEN cost gives the lower bound 1880.487608 and the best feasible solution gives the upper bound 1881.303155:

$$1880.487608 \le C^*_{\mathrm{MAPF}} \le 1881.303155$$

$$\rho = \frac{\mathrm{cost}}{C^*_{\mathrm{MAPF}}} \le 1.000434. \tag{5}$$

This is an a posteriori interval for the recorded search state, not a universal a priori guarantee. It leans on implementation conditions that are not separately proved here: that all valid constraint-tree nodes stay in OPEN, that the low level returns the constrained optimum for every node, that disjoint splitting discards no solution, and that the PP solution uses the same objective. With the CBS solution as numerator the ratio reads $\rho \le 1.000596$. Taking the smallest OPEN cost as a lower bound is standard in anytime search [18,19], and EECBS applies it at the CBS high level [20]; here it is recomputed read-only outside the solver. For reference, the a priori guarantee of ECBS is $\mathrm{cost} \le wC^*_{\mathrm{MAPF}}$, that is 2% for $w = 1.02$. After exhausting 100000 high-level nodes at $w = 1.0$ the frontier cost 1880.714014 still carries 6 conflicts, so it is neither a feasible cost nor a lower bound.

6.4 Three Failure Modes of Evaluation

The audits above were designed around three ways in which a MAPF benchmark can report success while measuring nothing.

Reading a flag as a certificate. A planner returns "solved" when it has a path, which says nothing about cost. 'metric' does exactly this and is 16.15% off at the root. The remedy is a criterion independent of the flag; here it is the agreement of four admissible root costs at 1880.008549.

Reading fewer expansions as progress. An inadmissible heuristic prunes harder precisely because it is wrong, so expansion count falls while solution cost rises. The two must be read together; Table 2 keeps the solution-cost row next to the expansion rows for that reason.

Reporting a metric the planner never minimized. The turn cost is computed and reported, but it depends on three consecutive positions and cannot be an edge weight on a position-only state space. The planner minimizes $\Sigma\omega$; the reported objective also carries $\Sigma\mathrm{turn}$. The mismatch costs 0.38%–0.95% of low-level optimality, median +0.61%, and stays invisible unless a direction-aware reference is computed offline. Our audit reference values come from min-plus iteration on the augmented state (voxel, incoming direction).

7. Discussion and Conclusion

The results are limited to one parameterized urban bottleneck scene, a $64 \times 64 \times 24$ grid, 18–24 agents, and the stated sampling and search budgets. Time-varying risk fields and alternative weight configurations need separate validation. The objective J is configuration-specific and is not used for cross-paper numerical comparison.

Two findings carry the paper. A goal-oriented reverse-Dijkstra table is exact on a static nonnegative directed spatial graph, yet being single-source it cannot answer every segment-terminal

query; the differential bound closes that gap without a second table. The bound saves a great deal of low-level search when via-point constraints inflate intermediate segments—81421 down to 2136 expansions, -97.4%—and essentially nothing, a net factor of 0.99, when those segments are a small share of the search. Both outcomes belong in the result.

Where the construction stops holding is as follows. It requires time-independent edge weights. Under a time-varying risk field the reverse-Dijkstra table no longer returns exact distances, and the differential bound would have to be rebuilt on a time-expanded or re-anchored representation; we have not costed that, and make no claim about the regime. Negative edge weights, and objectives depending on more than position and incoming direction, fall outside Proposition 1. The bound is also strictly looser than the exact value whenever the triangle inequality is not tight, which is the case for 43.6% of the audited samples; the observed benefit therefore comes from reducing a very loose baseline, not from achieving exactness.

The audit we did not run is the time-stamped SIPP audit: this work certifies the spatial graph, and the SIPP side rests on the lower-bound remark after Proposition 1. Extending it to dynamic risk fields and to scenes with more than 24 agents would test whether the 83.4% intermediate-segment share that drives the gain is a property of bottleneck geometry or an artifact of the present scenario family. That is the most useful next step.

References

[1] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” Artificial Intelligence, vol. 219, pp. 40–66, 2015, doi: 10.1016/j.artint.2014.11.006.

[2] M. Barer, G. Sharon, R. Stern, and A. Felner, “Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem,” Proc. Int. Symp. Combinatorial Search, vol. 5, no. 1, pp. 19–27, 2014, doi: 10.1609/socs.v5i1.18315.

[3] N. R. Sturtevant, A. Felner, M. Barer, J. Schaeffer, and N. Burch, “Memory-based heuristics for explicit state spaces,” Proc. Int. Joint Conf. Artificial Intelligence, pp. 609–614, 2009.

[4] A. V. Goldberg and C. Harrelson, “Computing the shortest path: A* search meets graph theory,” Proc. ACM–SIAM Symp. Discrete Algorithms, pp. 156–165, 2005.

[5] K. C. Ng and H. Zhang, “Differential heuristics for search,” Proc. Int. Conf. Tools with Artificial Intelligence, pp. 98–105, 2002.

[6] M. Phillips and M. Likhachev, “SIPP: Safe interval path planning for dynamic environments,” Proc. IEEE Int. Conf. Robotics and Automation, pp. 5628–5635, 2011, doi: 10.1109/ICRA.2011.5980306.

[7] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, “Multi-agent pathfinding with continuous time,” Artificial Intelligence, vol. 305, Art. no. 103662, 2022, doi: 10.1016/j.artint.2022.103662.

[8] A. Andreychuk and K. Yakovlev, “Multi-agent path finding with kinematic constraints via conflict based search,” in Robotics and Artificial Intelligence, LNCS, vol. 12412, pp. 29–45, 2020, doi: 10.1007/978-3-030-59535-7_3.

[9] W. Hoenig, T. K. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, “Multi-agent path finding with kinematic constraints,” Proc. Int. Conf. Automated Planning and Scheduling, vol. 26, no. 1, pp. 477–485, 2016, doi: 10.1609/icaps.v26i1.13796.

[10] M. Cap, M. Vokrinek, and A. Kleiner, “Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures,” Proc. Int. Conf. Automated Planning and Scheduling, vol. 25, no. 1, pp. 324–332, 2015, doi: 10.1609/icaps.v25i1.13696.

[11] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, “Searching with consistent prioritization for

multi-agent path finding," Proc. AAAI Conf. Artificial Intelligence, vol. 33, no. 1, pp. 7643–7650, 2019, doi: 10.1609/aaai.v33i01.33017643.

[12] H. T. To, S. Nguyen, and N. H. Pham, "CAHC: A general conflict-aware heuristic caching framework for multi-agent path finding," arXiv:2512.12243, 2025, doi: 10.48550/arXiv.2512.12243.

[13] Y. Yu and M. Le, "A spatio-temporal collaborative improved multi-strategy dung beetle optimization algorithm for 3D path planning of multiple unmanned aerial vehicles in urban environments," Aerospace, vol. 13, no. 6, Art. no. 506, 2026, doi: 10.3390/aerospace13060506.

[14] M. C. Gencal, B. Ata, M. Kurucan, et al., "An improved roosters algorithm for constrained 3D UAV path planning in urban environments," Scientific Reports, vol. 15, Art. no. 40975, 2025, doi: 10.1038/s41598-025-24752-8.

[15] T. Lei, S. Liu, and Y. Lei, "Hierarchical path planning for low-altitude logistics UAVs in complex urban environments: An enhanced weighted A* approach with geometric smoothing," Frontiers in Robotics and AI, vol. 13, Art. no. 1802470, 2026, doi: 10.3389/frobt.2026.1802470.

[16] D. B. Johnson, "Efficient algorithms for shortest paths in sparse networks," J. ACM, vol. 24, no. 1, pp. 1–13, 1977, doi: 10.1145/321992.321993.

[17] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," Technometrics, vol. 21, no. 2, pp. 239–245, 1979, doi: 10.1080/00401706.1979.10489755.

[18] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in Advances in Neural Information Processing Systems, vol. 16, pp. 767–774, 2003.

[19] J. T. Thayer and W. Ruml, "Bounded suboptimal search: A direct approach using inadmissible estimates," Proc. Int. Joint Conf. Artificial Intelligence, pp. 674–679, 2011.

[20] J. Li, W. Ruml, and S. Koenig, "EECBS: A bounded-suboptimal search for multi-agent path finding," Proc. AAAI Conf. Artificial Intelligence, vol. 35, no. 14, pp. 12353–12362, 2021, doi: 10.1609/aaai.v35i14.17466.