Risk-Bounded Multi-Agent Visual Navigation via Iterative Risk Allocation

Viraj Parimi, Brian Williams

Massachusetts Institute of Technology
vparimi@mit.edu, williams@csail.mit.edu

Abstract

Safe navigation is essential for autonomous systems operating in hazardous environments, especially when multiple agents must coordinate using only high-dimensional visual observations. While recent approaches successfully combine Goal-Conditioned RL (GCRL) for graph construction with Conflict-Based Search (CBS) for planning, they typically rely on deleting edges with high risk before running CBS to enforce safety. This binary strategy is overly conservative, precluding feasible missions that require traversing high-risk regions, even when the aggregate risk is acceptable. To address this, we introduce a framework for Risk-Bounded Multi-Agent Path Finding (Δ-MAPF), where agents share a user-specified global risk budget (Δ). Rather than permanently discarding edges, our framework dynamically distributes per-agent risk budgets (δ_i) during search via an Iterative Risk Allocation (IRA) layer that integrates with a standard CBS planner. We investigate two distribution strategies: a greedy surplus-deficit scheme for rapid feasibility repair, and a market-inspired mechanism that treats risk as a priced resource to guide improved allocation. The market-based mechanism yields a tunable trade-off wherein agents exploit available risk to secure shorter, more efficient paths, but revert to longer, safer detours under tighter budgets. Experiments in complex visual environments show that our dynamic allocation framework achieves higher success rates than baselines and effectively leverages the available safety budget to reduce travel time.

Project Website —
https://rb-visual-mapf-mers.csail.mit.edu

1 Introduction

Safe and efficient multi-agent navigation is critical in domains like disaster relief (Wang and Zlatanova 2016), large-scale inspection (Im and Lee 2023), and environmental monitoring (Peralta et al. 2022), where failures are costly in both resources and potential harm to people or the environment. Existing Multi-Agent Path Finding (MAPF) (Stern et al. 2019) methods range from exhaustive approaches like Conflict-Based Search (CBS) (Sharon et al. 2015), which provide high-quality solutions but scale poorly, to prioritized schemes like Priority-Based Search (PBS) (Ma et al. 2018), which trade optimality for scalability. CBS's modular design has enabled many enhancements (Barer et al. 2021; Boyarski et al. 2015; Cohen et al. 2021) extending to lifelong planning (Ma et al. 2017), information-guided planning (Olkin, Parimi, and Williams 2024) and more recently diffusion-guided planning (Shaoul et al. 2024; Parimi and Williams 2025). However, these methods typically assume access to an explicit or implicit graph with valid transitions and known costs, an assumption that breaks down when agents must operate directly from high-dimensional visual observations where the graph structure is unknown.

Goal-conditioned reinforcement learning (GCRL) (Mirowski et al. 2017; Schaul et al. 2015; Pong et al. 2020) excels at learning a single navigation policy directly from complex observations across many goals, making it well suited for visually rich and unstructured environments. However, GCRL alone often struggles on long-horizon tasks (Levy, Platt, and Saenko 2019; Nachum et al. 2018), especially when balancing risk against efficiency. While Constrained RL (Altman 2021), Control Barrier Functions (CBFs) and Safe MARL (Zhang et al. 2025; Zhao et al. 2020; Cheng et al. 2019) offer robust tools for enforcing safety, they typically operate reactively by shaping low-level actions to satisfy hard state constraints. In addition, recent hybrid approaches (Eysenbach, Salakhutdinov, and Levine 2019; Feng, Parimi, and Williams 2025) integrate GCRL with graph-based planning. These methods build an intermediate waypoint graph from a replay buffer, learn distance and risk critics, prune edges deemed unsafe, and then apply CBS to coordinate multiple agents. This approach yields safer waypoint-based plans that respect the learned safety critics. However, such static edge pruning is fundamentally binary, as edges exceeding a local threshold are permanently discarded. This rigidity proves to be overly conservative in missions where goals require entering hazardous regions, or when accepting a small, controlled amount of risk could dramatically reduce travel time or even make an otherwise infeasible mission possible (Figs. 1–3). Figures 1–2 show representative 2D examples (single- and multi-agent), while Figure 3 shows the analogous effect in a visually complex environment. This is a challenge that demands *budgeted risk acceptance* rather than strict avoidance.

Parallel to these developments, risk allocation in optimal control methods like Iterative Risk Allocation (IRA)

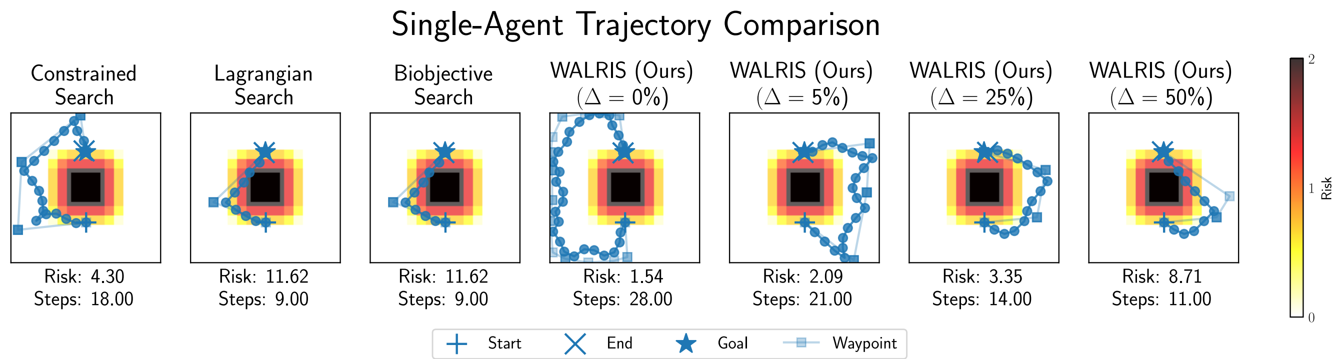

Figure 1: Single-agent trajectory comparison on 2D navigation task. Cumulative risk and step count are shown below each plot. Standard baselines are either rigidly conservative (Constrained Search) or overly aggressive (Lagrangian, Biobjective). In contrast, our planner (using the WALRIS strategy) enables a tunable trade-off. As the risk budget Δ increases, the agent accepts tighter clearances to the hazard to reduce path length, smoothly transitioning from safe detours to efficient direct routes.

(Ono and Williams 2008) and its extensions, such as MIRA (Ono 2012; Ono and Williams 2010), treat a global chance constraint as a shared resource. By redistributing risk from "easy" constraints to "hard" ones, they mitigate the conservatism of uniform risk allocations. However, these methods rely on convex trajectory optimization in continuous-state spaces and do not address the discrete, combinatorial structure of multi-agent conflict resolution, nor do they handle planning over unstructured and learned graphs.

To address these limitations, we introduce the *Risk-Bounded Multi-Agent Path Finding* (Δ-MAPF) framework, where all agents share a user-specified global risk budget Δ. Instead of statically pruning unsafe edges, we augment the standard CBS constraint tree with an IRA layer that maintains per-agent budgets $\boldsymbol{\delta}$ and redistributes them upon node infeasibility. Within this framework, we investigate two complementary risk distribution strategies grounded in economic principles. The first, *EQUIRIS*, is a greedy surplus-deficit scheme that shifts risk in an equity-like fashion from agents with slack to those in need, serving as a fast feasibility repair heuristic. The second, *WALRIS*, is a Walrasian tatonnement-inspired mechanism that treats risk as a priced resource and lets agents independently trade off path length against risk at a shared price signal. By varying the global budget Δ, WALRIS yields a smooth, user-controllable trade-off. When the global budget is generous, agents exploit higher local risk allowances to find shorter paths, while tighter budgets induce longer but safer detours aligning behavior with user-defined safety preferences.

In summary, our contributions are threefold. First, we formalize the Δ-MAPF problem on learned waypoint graphs subject to a global risk bound Δ. Second, we augment the CBS constraint tree with a discrete Iterative Risk Allocation (IRA) layer that dynamically redistributes per-agent budgets via two complementary strategies, EQUIRIS or WALRIS. Finally, we demonstrate superior safety-efficiency trade-offs over baselines in both 2D and complex visual environments, as well as a ROS2/Gazebo integration controlling multiple Crazyflie drones in simulation and hardware.

2 Preliminaries

We consider multi-agent navigation in visually rich environments, where each agent must reach a goal while ensuring that the *total accumulated risk* across all agents does not exceed a global risk bound Δ.

2.1 Goal-Conditioned Reinforcement Learning

In GCRL, an agent interacts with an environment modeled as a Markov Decision Process (MDP) $(\mathcal{S}, A, P, R, \gamma)$ augmented with a goal $s_g \in \mathcal{S}$. Here $\mathcal{S}$ denotes the high-dimensional state space, A is the action space, $P(s_{t+1} \mid s_t, a_t)$ is the transition dynamics, $R(s, a, s_g)$ is a goal-dependent reward providing feedback on progress towards s_g, and $\gamma \in [0, 1)$ is the discount factor. The agent learns a policy $\mu_\theta(a \mid s, s_g)$ that maximizes the expected cumulative reward conditioned on both the current state s and goal s_g. While GCRL is effective for learning short-horizon skills, often leveraging reward shaping (Chiang et al. 2019) or demonstrations (Lynch et al. 2019; Nair et al. 2018), we use it here to abstract low-level visual control into a navigable graph via learned distance and risk critics.

2.2 Learned Distance and Risk Graph

Following Feng, Parimi, and Williams, we employ a dual critic architecture training two value functions, $\mathcal{Q}^d_\theta(s, a, s_g)$ and $\mathcal{Q}^c_\theta(s, a, s_g)$, to estimate the distance and risk between state-goal pairs. For a pair of states (s_i, s_j) and action $a_{ij} = \mu_\theta(s_i, s_j)$ derived from the policy, we define the metrics

$$d_\mu(s_i, s_j) \leftarrow \mathcal{Q}^d_\theta(s_i, a_{ij}, s_j),$$
$$c_\mu(s_i, s_j) \leftarrow \mathcal{Q}^c_\theta(s_i, a_{ij}, s_j).$$

where d_μ approximates shortest-path distance and c_μ estimates the cumulative risk of traversing from s_i to s_j. From the agent's replay buffer, we sample a set of states $\mathcal{B}$ and

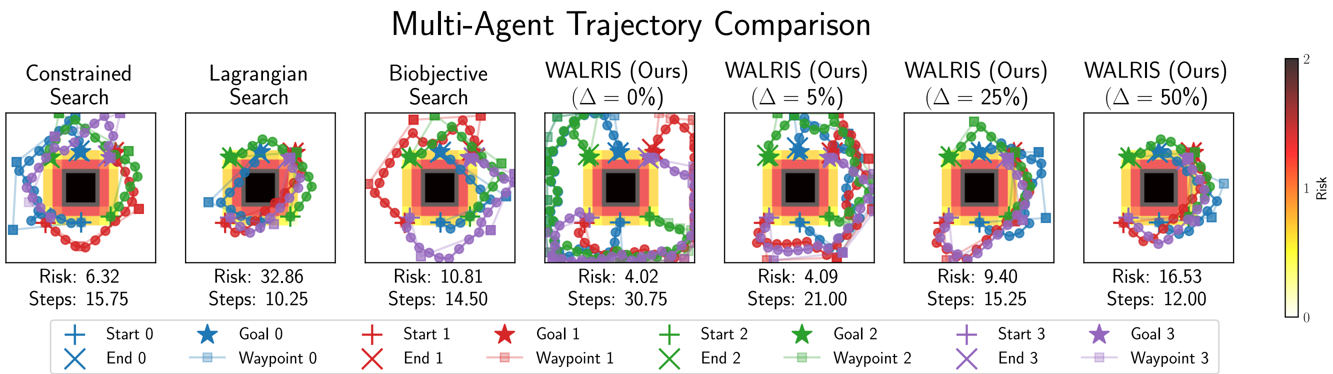

Figure 2: Multi-agent trajectory comparison on the 2D navigation task. Cumulative risk and average step count are annotated below each plot. While baselines are statically locked into specific trade-offs (e.g., Constrained Search is safe but inefficient; Lagrangian is efficient but violates safety), our framework enables a dynamic spectrum of behavior. At strict budgets ($\Delta = 0\%$), agents coordinate to take wide, safe detours; as the budget relaxes (e.g., $\Delta = 50\%$), they collectively "spend" the risk resource to cut through the center, significantly reducing travel time.

build a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}_d, \mathcal{W}_c)$ with

$$\begin{aligned}
\mathcal{V} &= \mathcal{B}, \\
\mathcal{E} &= \{e_{s_i \to s_j} \mid s_i, s_j \in \mathcal{B}\}, \\
\mathcal{W}_d(e_{s_i \to s_j}) &= \begin{cases} d_\mu(s_i, s_j), & \text{if } d_\mu(s_i, s_j) < \mathrm{d}_{\max}, \\ \infty, & \text{otherwise}, \end{cases} \\
\mathcal{W}_c(e_{s_i \to s_j}) &= c_\mu(s_i, s_j).
\end{aligned}$$

Unlike Feng, Parimi, and Williams, we do not prune edges based on predicted risk. All edges with finite distances are retained, including those that may pass through hazardous regions. The risk critic $\mathcal{Q}^c_\theta$ instead provides additional risk information to the planner, which is used to reason about trade-offs under the global risk bound Δ.

2.3 Global Risk-Bounded MAPF

Let $\mathcal{A} = \{a_1, \ldots, a_N\}$ be a set of N agents, each with a start-goal pair $(s_i, g_i) \in \mathcal{V} \times \mathcal{V}$. Our objective is to find a joint plan $\Pi = \{\pi_1, \ldots, \pi_N\}$ of collision-free paths, where each π_i connects s_i to g_i, such that the total accumulated risk does not exceed a user-specified global budget Δ. We define the length of a single path π_i as $\ell(\pi_i) = \sum_{e \in \pi_i} \mathcal{W}_d(e)$, where $\mathcal{W}_d(e)$ is the learned distance cost associated with edge e. We then define the risk of a single path π_i as $\rho(\pi_i) = \sum_{e \in \pi_i} \mathcal{W}_c(e)$, where $\mathcal{W}_c(e)$ is the learned risk associated with edge e. The Δ-MAPF problem seeks to minimize the sum of path lengths, $\mathcal{J}(\Pi) = \sum_{i=1}^N \ell(\pi_i)$ subject to the global risk constraint $\sum_{i=1}^N \rho(\pi_i) \leq \Delta$.

This formulation treats risk as a *shared resource*, enabling the system to handle heterogeneous environments where tasks naturally vary in difficulty. For instance, in a search-and-rescue scenario, one agent may need to enter a hazardous zone while others remain in safe areas. Enforcing uniform per-agent risk limits would likely render such missions infeasible. Instead, by pooling the budget, our approach allows some agents to draw a larger share of risk when necessary for mission success. Furthermore, Δ serves as a single control knob that can adjust the degree of the imbalance. If a resulting plan is deemed too aggressive for a single agent, the user can tighten Δ, shrinking the available resource pool and forcing the planner to redistribute risk, yielding more conservative and balanced solutions.

3 Approach

We address the Δ-MAPF problem by augmenting a standard CBS planner with an *Iterative Risk Allocation* (IRA) layer. While standard CBS focuses solely on resolving spatio-temporal conflicts, our planner simultaneously manages the distribution of the global risk budget Δ. To do so, we explicitly maintain a vector of local risk budgets $\boldsymbol{\delta} = [\delta_1, \ldots, \delta_N]$ within the high-level search nodes allowing the planner to dynamically redistribute risk among agents when local constraints become too tight. Inspired by iterative risk allocation methods (Ono and Williams 2008; Ono 2012), we adapt this concept here as a discrete, reallocation step. Whenever a node becomes infeasible under its current budgets, we adjust $\boldsymbol{\delta}$ and continue the search in that branch. Upon finding a valid joint plan, agents execute the generated waypoints using the underlying goal-conditioned policy.

3.1 High-Level Search

The high-level search explores a Constraint Tree (CT), as in standard CBS, but augments each node with a risk-allocation state. A node is defined as a tuple $\mathcal{P} = (\mathcal{C}, \Pi, \boldsymbol{\delta}, \boldsymbol{\phi}, \mathcal{J})$ containing the spatio-temporal constraints $\mathcal{C}$, the current set of single-agent paths $\Pi = \{\pi_1, \ldots, \pi_N\}$, and the sum-of-costs objective $\mathcal{J} = \mathcal{J}(\Pi) = \sum_i \ell(\pi_i)$. Additionally, we store the local risk budget vector, $\boldsymbol{\delta} = [\delta_1, \ldots, \delta_N]$, and a boolean validity vector $\boldsymbol{\phi} \in \{0, 1\}^N$. Here, $\phi_i = 1$ indicates that a valid path for agent a_i is stored in Π satisfying both $\mathcal{C}$ and δ_i has been found, while $\phi_i = 0$ implies otherwise.

The high-level search (Algorithm 1) maintains a priority queue $\mathcal{F}$ of CT nodes. Nodes are prioritized primarily by the sum-of-costs $\mathcal{J}(\Pi)$, breaking ties with the number of unresolved collisions and, finally, the number of recent risk real-

Algorithm 1: Δ-MAPF: High-Level Search

1: Create root node $\mathcal{P}_0$ with initial (unconstrained) paths Π^0, and risk allocations $\boldsymbol{\delta}^0$.
2: $\mathcal{F}$.Insert($\mathcal{P}_0$)
3: **while** $\mathcal{F}$ not empty and time not exceeded **do**
4: $\quad \mathcal{P} \leftarrow \mathcal{F}$.ExtractMin() $\triangleright$ *Pop node from the frontier set with lowest priority key*
5: $\quad \Pi \leftarrow \mathcal{P}.\Pi,\ \boldsymbol{\delta} \leftarrow \mathcal{P}.\boldsymbol{\delta},\ \boldsymbol{\phi} \leftarrow \mathcal{P}.\boldsymbol{\phi}$
6: $\quad$ **if** $\exists i$ s.t. $\phi_i = 0$ **then**
7: $\quad\quad \mathcal{A}_{\text{fail}} \leftarrow \emptyset$
8: $\quad\quad$ **for** each a_i with $\phi_i = 0$ **do**
9: $\quad\quad\quad (\pi_i, \ell_i, \rho_i) \leftarrow \text{RBA*}(a_i, \mathcal{P}, \delta_i)$
10: $\quad\quad\quad$ **if** FAIL **then**
11: $\quad\quad\quad\quad \mathcal{A}_{\text{fail}} \leftarrow \mathcal{A}_{\text{fail}} \cup \{a_i\}$
12: $\quad\quad\quad$ **else**
13: $\quad\quad\quad\quad$ Update Π with π_i and set $\phi_i \leftarrow 1$
14: $\quad\quad\quad$ **end if**
15: $\quad\quad$ **end for**
16: $\quad\quad$ **if** $\mathcal{A}_{\text{fail}} \neq \emptyset$ **then**
17: $\quad\quad\quad \boldsymbol{\delta}' \leftarrow$ ReallocateRisk($\mathcal{P}, \mathcal{A}_{\text{fail}}$)
18: $\quad\quad\quad$ **if** $\boldsymbol{\delta}' \neq$ FAIL **then**
19: $\quad\quad\quad\quad \mathcal{P}' \leftarrow$ copy of $\mathcal{P},\ \mathcal{P}'.\boldsymbol{\delta} \leftarrow \boldsymbol{\delta}'$
20: $\quad\quad\quad\quad$ Update $\mathcal{P}'.\phi_j \leftarrow 0\ \forall a_j \in \mathcal{A}_{\text{fail}}$
21: $\quad\quad\quad\quad \mathcal{F}$.Insert($\mathcal{P}'$)
22: $\quad\quad\quad$ **end if**
23: $\quad\quad\quad$ **continue**
24: $\quad\quad$ **end if**
25: $\quad$ **end if**
26: $\quad \mathcal{K} \leftarrow$ DetectCollisions(Π)
27: $\quad$ **if** $\mathcal{K} = \emptyset$ and $\sum_{i=1}^{N} \rho(\pi_i) \leq \Delta$ **then**
28: $\quad\quad$ **return** Π $\triangleright$ *Solution found*
29: $\quad$ **end if**
30: $\quad c \leftarrow$ SelectCollision($\mathcal{K}$)
31: $\quad$ Generate disjoint split constraints for collision c
32: $\quad$ **for** each new constraint on agent a_k **do**
33: $\quad\quad \mathcal{P}' \leftarrow$ copy of $\mathcal{P}$ with added constraint
34: $\quad\quad (\pi'_k, \ell'_k, \rho'_k) \leftarrow \text{RBA*}(a_k, \mathcal{P}', \delta_k)$
35: $\quad\quad$ **if** FAIL **then**
36: $\quad\quad\quad \boldsymbol{\delta}' \leftarrow$ ReallocateRisk($\mathcal{P}', \{a_k\}$)
37: $\quad\quad\quad$ **if** $\boldsymbol{\delta}' \neq$ FAIL **then**
38: $\quad\quad\quad\quad \mathcal{P}'.\boldsymbol{\delta} \leftarrow \boldsymbol{\delta}',\ \mathcal{P}'.\phi_k \leftarrow 0$
39: $\quad\quad\quad\quad \mathcal{F}$.Insert($\mathcal{P}'$)
40: $\quad\quad\quad$ **end if**
41: $\quad\quad$ **else**
42: $\quad\quad\quad$ Update $\mathcal{P}'.\Pi$ with π'_k
43: $\quad\quad\quad \mathcal{P}'.\mathcal{J} \leftarrow$ SumOfCosts($\mathcal{P}'.\Pi$), $\mathcal{P}'.\phi_k \leftarrow 1$
44: $\quad\quad\quad \mathcal{F}$.Insert($\mathcal{P}'$)
45: $\quad\quad$ **end if**
46: $\quad$ **end for**
47: **end while**
48: **return** No solution found.

locations. This favors nodes that are both low-cost and stable in their risk distributions. The main loop begins by computing initial single-agent paths on the learned waypoint graph (ignoring risk) and an initial risk allocation $\boldsymbol{\delta}^0$ (Section 3.3).

Each iteration proceeds in two phases. In *Phase 1* (lines 6-25), we attempt to compute valid paths for any agents marked as invalid ($\phi_i = 0$) using the risk-constrained RBA* planner (Section 3.2). If any agent fails to find a path within its budget δ_i, the set of failing agents is collected, and a risk allocation step is attempted. If successful, a new child node with the updated budgets is added to $\mathcal{F}$ while an unsuccessful reallocation causes the current node to be pruned.

In *Phase 2* (lines 26-46), once all agents have valid paths, the search reduces to standard CBS. We detect collisions in Π, return a solution if none remain and $\sum_i \rho(\pi_i) \leq \Delta$ is satisfied. Otherwise, we branch on a selected collision using disjoint split (Li et al. 2019). For each child, we attempt to replan the newly constrained agent with its current budget. Crucially, if this replanning fails due to the new constraint, we immediately trigger the risk allocation layer to determine whether budget adjustment can restore feasibility. In this way, the CT search and the risk-allocation layer interact tightly where CBS handles discrete conflicts, while the allocation layer reshapes $\boldsymbol{\delta}$ to keep promising branches feasible under the global risk bound.

3.2 Low-Level Search

The low-level single-agent planner is a risk-bounded variant of A*, denoted RBA*. Given an agent a_i, a set of constraints from node $\mathcal{P}$, and a budget δ_i, it searches the learned waypoint graph for a path π_i that minimizes length $\ell(\pi_i)$ while satisfying $\rho(\pi_i) \leq \delta_i$. To ensure soundness, RBA* operates on an augmented state space (u, r), where $u \in \mathcal{V}$ is the current node and r is the accumulated risk. During search, we prune any state where $r > \delta_i$. Crucially, we employ dominance pruning (Stewart and White 1991) where a path to node u with length ℓ and risk ρ is discarded if and only if there exists a previously discovered path to u with length $\ell' \leq \ell$ and risk $\rho' \leq \rho$. Formally, the planner returns

$$\pi_i^*(\delta_i) \in \underset{\pi_i \in \Pi_{\mathcal{P}} : \rho(\pi_i) \leq \delta_i}{\operatorname{argmin}} \ell(\pi_i).$$

where $\Pi_{\mathcal{P}}$ is the set of paths satisfying the spatio-temporal constraints in $\mathcal{P}$.

Besides RBA*, the risk allocation layer uses two simpler unconstrained A*-based queries on the same learned waypoint graph at a CT node $\mathcal{P}$.

Minimum feasible risk MinFeasibleRisk($a_i, \mathcal{P}$). This query finds the safest path regardless of length. We run A* using learned risk weights $\mathcal{W}_c$ and a risk-based heuristic to find $\pi_i^{\text{risk}} \in \operatorname{argmin}_{\pi_i \in \Pi_{\mathcal{P}}} \rho(\pi_i)$, yielding the lower bound $\delta_i^{\min} \leftarrow \rho(\pi_i^{\text{risk}})$

Risk of shortest path LenMinRisk($a_i, \mathcal{P}$). This query finds the shortest path regardless of risk. We run A* using the learned distance weights $\mathcal{W}_d$ to find $\pi_i^{\text{len}} \in \operatorname{argmin}_{\pi_i \in \Pi_{\mathcal{P}}} \ell(\pi_i)$, yielding $\delta_i^{\max} \leftarrow \rho(\pi_i^{\text{len}})$.

3.3 Risk Budget Management

The core innovation of our approach is its explicit management of per-agent risk budgets under the global constraint Δ. This has two components: (i) determining an initial allocation at the root, and (ii) iterative reallocation strategy when agents fail to find feasible paths.

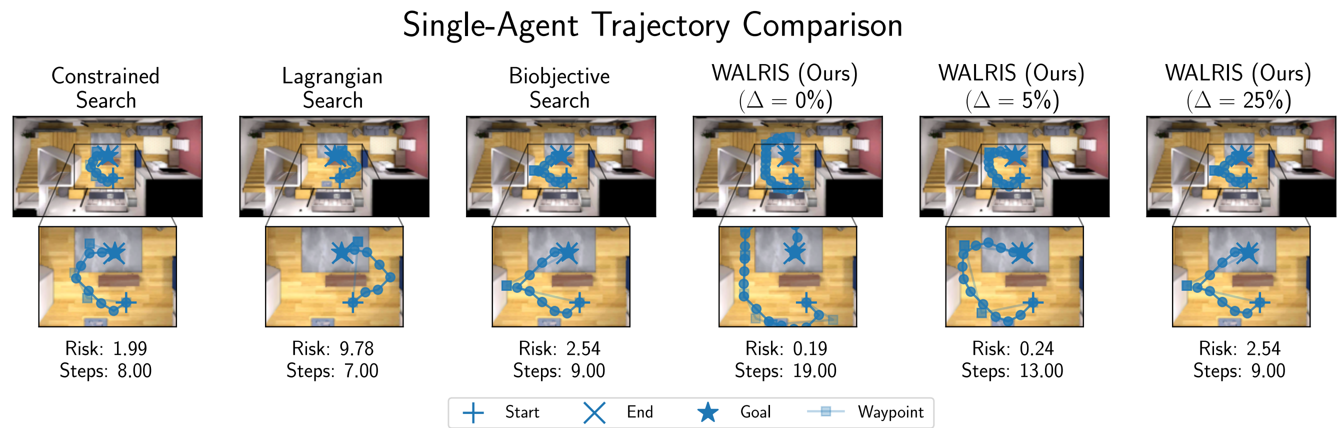

Figure 3: Single-agent trajectory comparison on the visual navigation task. Cumulative risk and step count are shown below each plot. While baselines produce static solutions that can be excessively risky (Lagrangian) or rigid, our planner enables a dynamic trade-off controlled by Δ. At strict budgets ($\Delta = 0\%$), the agent takes a significant detour to ensure maximal safety, smoothly transitioning to direct, efficient routes as the allowed risk budget increases.

Initial Budget Allocation At the root of the CT, we compute an initial allocation $\boldsymbol{\delta}^0$ based on metrics derived from the agents' unconstrained paths on the learned waypoint graph. We use three schemes defined by a utility term w_i:

- **Uniform**: Divide the budget equally, $\delta_i = \Delta/N$.
- **Utility-based**: $\delta_i = \Delta \cdot \frac{w_i}{\sum_j w_j}$, e.g., $w_i = \rho_i$ to give more budget to agents with riskier unconstrained paths.
- **Inverse utility-based**: $\delta_i = \Delta \cdot \frac{1/w_i}{\sum_j (1/w_j)}$, e.g., $w_i = \ell_i$ to favor shorter, potentially riskier paths.

Following initialization, the search invokes the IRA layer to adjust $\boldsymbol{\delta}$ whenever a node becomes infeasible. To do so, we introduce two economically inspired strategies: *EQUIRIS*, a surplus-deficit scheme that transfers risk from agents with slack to those in need of risk in an equity-like fashion, and *WALRIS*, a Walrasian tatonnement-inspired (Tuinstra 2012) mechanism that treats risk as a traded resource balancing global demand against the budget Δ.

EQUItable RIsk-bounded Search (EQUIRIS) EQUIRIS is a greedy surplus-deficit scheme designed to "rescue" infeasible nodes by transferring risk budget from agents with slack to those in deficit. As outlined in Algorithm 2, EQUIRIS proceeds in three steps. First (lines 3-6), it quantifies the total deficit δ_{req} by summing the difference between the minimum required risk $\delta_i^{\min}$ (computed via MINFEASIBLERISK) and the current budget δ_i for all failing agents. Second (lines 8-11), it calculates the total available surplus δ_{avail} from the passing agents. If $\delta_{\text{req}} > \delta_{\text{avail}}$, even the most generous redistribution cannot satisfy all failing agents, so the node is declared infeasible and pruned. Finally (lines 15-27), we construct the new allocation $\boldsymbol{\delta}'$ by assigning each failing agent its minimum requirement $\delta_i^{\min}$ and greedily deducting the balance from the passing agents' surpluses until the deficit is covered.

EQUIRIS is computationally efficient, relying solely on minimum-risk queries to rebalance risk distribution. However, it does not explicitly reason about the length-risk trade-off. By focusing only on feasibility repair, it may miss reallocations that could yield superior sum-of-costs objectives.

Algorithm 2: EQUIRIS Strategy

1: **function** REALLOCATERISK($\mathcal{P}$, $\mathcal{A}_{\text{fail}}$)
2: $\quad\delta_{\text{req}} \leftarrow 0$
3: $\quad$**for** each failing agent $a_i \in \mathcal{A}_{\text{fail}}$ **do**
4: $\quad\quad\delta_i^{\min} \leftarrow$ MINFEASIBLERISK$(a_i, \mathcal{P})$
5: $\quad\quad\delta_{\text{req}} \leftarrow \delta_{\text{req}} + (\delta_i^{\min} - \mathcal{P}.\delta_i)$
6: $\quad$**end for**
7: $\quad\delta_{\text{avail}} \leftarrow 0$
8: $\quad$**for** each passing agent $a_j \notin \mathcal{A}_{\text{fail}}$ **do**
9: $\quad\quad\delta_j^{\min} \leftarrow$ MINFEASIBLERISK$(a_j, \mathcal{P})$
10: $\quad\quad\delta_{\text{avail}} \leftarrow \delta_{\text{avail}} + (\mathcal{P}.\delta_j - \delta_j^{\min})$
11: $\quad$**end for**
12: $\quad$**if** $\delta_{\text{req}} > \delta_{\text{avail}}$ **then**
13: $\quad\quad$**return** FAIL ▷ *Not enough surplus budget*
14: $\quad$**end if**
15: $\quad\boldsymbol{\delta}' \leftarrow \mathcal{P}.\boldsymbol{\delta}$
16: $\quad$**for** each failing agent $a_i \in \mathcal{A}_{\text{fail}}$ **do**
17: $\quad\quad\delta_i' \leftarrow \delta_i^{\min}$ ▷ *Assign minimum required*
18: $\quad$**end for**
19: $\quad\delta_{\text{rem}} \leftarrow \delta_{\text{req}}$ ▷ *Deduct from surplus agents*
20: $\quad$**for** each passing agent $a_j \notin \mathcal{A}_{\text{fail}}$ **do**
21: $\quad\quad\Delta\delta_j \leftarrow \mathcal{P}.\delta_j - \delta_j^{\min}$
22: $\quad\quad\epsilon_j \leftarrow \min(\Delta\delta_j, \delta_{\text{rem}})$
23: $\quad\quad\delta_j' \leftarrow \delta_j' - \epsilon_j$
24: $\quad\quad\delta_{\text{rem}} \leftarrow \delta_{\text{rem}} - \epsilon_j$
25: $\quad\quad$**if** $\delta_{\text{rem}} \le 0$ **then break**
26: $\quad\quad$**end if**
27: $\quad$**end for**
28: $\quad$**return** $\boldsymbol{\delta}'$
29: **end function**

WALrasian RIsk-bounded Search (WALRIS) While EQUIRIS efficiently repairs feasibility, its fixed ordering of

Algorithm 3: WALRIS Strategy

1: **function** REALLOCATERISK($\mathcal{P}$, $\mathcal{A}_{\text{fail}}$)
2: $\boldsymbol{\delta} \leftarrow \mathcal{P}.\boldsymbol{\delta}$ ▷ *current allocation is starting point*
3: **for** each agent $a_i \in \mathcal{A}$ **do**
4: $\delta_i^{\min} \leftarrow$ MINFEASIBLERISK$(a_i, \mathcal{P})$
5: $\delta_i^{\max} \leftarrow$ LENMINRISK$(a_i, \mathcal{P})$
6: **end for**
7: **if** $\sum_i \delta_i^{\min} > \Delta$ **then return** `FAIL`
8: **end if**
9: **if** $\sum_i \delta_i^{\max} \leq \Delta$ **then return** $\{\delta_i^{\max}\}_{i=1}^{N}$
10: **end if**
11: $(p_{\min}, p_{\max}) \leftarrow$ INITPRICEBOUNDS$(\delta_i^{\min}, \delta_i^{\max})$
12: $\mathcal{J}^* \leftarrow \infty,\ \boldsymbol{\delta}^{\text{best}} \leftarrow$ `None`, $k \leftarrow 0$
13: **while** $p_{\max} - p_{\min} \geq \epsilon$ **and** $k < K_{\max}$ **do**
14: $p \leftarrow (p_{\min} + p_{\max})/2$
15: **for** each agent $a_i \in \mathcal{A}$ **do**
16: Define a discrete neighborhood set $\mathcal{N}_i$ around δ_i clipped to $[\delta_i^{\min}, \delta_i^{\max}]$
17: $s_i^* \leftarrow \infty$
18: **for** each $\hat{\delta} \in \mathcal{N}_i$ **do**
19: $(\pi_i, \ell_i, \rho_i) \leftarrow$ RBA*$(a_i, \mathcal{P}, \hat{\delta})$
20: **if** `FAIL` **then continue**
21: **end if**
22: $s_i \leftarrow \ell_i + p \cdot \rho_i$
23: **if** $s_i < s_i^*$ **then**
24: $s_i^* \leftarrow s_i; \delta_i^{\text{new}} \leftarrow \hat{\delta}; \pi_i^{\text{new}} \leftarrow \pi_i$
25: **end if**
26: **end for**
27: **if** $s_i^* = \infty$ **then return** `FAIL`
28: **end if**
29: $\delta_i \leftarrow \delta_i^{\text{new}}$, Update Π with π_i^{new}
30: **end for**
31: $\mathcal{J}(\Pi) \leftarrow$ SUMOFCOSTS(Π)
32: **if** $\sum_i \rho(\pi_i) \leq \Delta$ **then**
33: **if** $\mathcal{J}(\Pi) < \mathcal{J}^*$ **then**
34: $\mathcal{J}^* \leftarrow \mathcal{J}(\Pi); \boldsymbol{\delta}^{\text{best}} \leftarrow \boldsymbol{\delta}$
35: **end if**
36: $p_{\max} \leftarrow p$ ▷ *risk underused; decrease upper price bound*
37: **else**
38: $p_{\min} \leftarrow p$ ▷ *risk overused; increase lower price bound*
39: **end if**
40: $k \leftarrow k + 1$
41: **end while**
42: **if** $\boldsymbol{\delta}^{\text{best}} =$ `None` **then return** `FAIL`
43: **end if**
44: **return** $\boldsymbol{\delta}^{\text{best}}$
45: **end function**

surplus donors limits its ability to improve the global objective. It may overlook allocations where drawing surplus from a different subset of agents would yield a lower total path length. WALRIS addresses this by introducing a market-based allocator inspired by Walrasian tatonnement (Tuinstra 2012; Ono and Williams 2010). This method treats risk as a scarce, priced resource. Instead of prescribing rigid transfers, WALRIS broadcasts a scalar *price of risk* $p \geq 0$, allowing each agent to independently improve its local trade-off between path length and risk.

As detailed in Algorithm 3, WALRIS first computes the feasible risk range $[\delta_i^{\min}, \delta_i^{\max}]$ for each agent. If $\sum_i \delta_i^{\min} > \Delta$, no feasible allocation exists and the node is pruned. Conversely, if $\sum_i \delta_i^{\max} \leq \Delta$, the budget is sufficient for all agents to take their length-optimal paths. In the non-trivial case where budget enforces a trade-off, we initialize a price interval $[p_{\min}, p_{\max}]$ and search for a "clearing" price. We defer the initialization of this interval to the Supplement Sec. B.2. Inside the optimization loop (lines 14-30), given a candidate price p, each agent a_i explores a discrete neighborhood $\mathcal{N}_i$ around its current budget. For each candidate $\hat{\delta} \in \mathcal{N}_i$, the agent computes the path using RBA* and selects the path that minimizes the price augmented objective

$$s_i(\hat{\delta}, p) = \ell(\pi_i^*(\hat{\delta})) + p \cdot \rho(\pi_i^*(\hat{\delta})).$$

After collecting responses, WALRIS aggregates the total risk $\sum_i \rho(\pi_i)$ and the sum-of-costs $\mathcal{J}(\Pi)$. If $\sum_i \rho(\pi_i) \leq \Delta$, the allocation is feasible. We record it if it improves the best known $\mathcal{J}(\Pi)$ and then *decrease* $p_{\max}$ to lower the cost of risk, encouraging agents to find shorter, riskier paths. Conversely, if $\sum_i \rho(\pi_i) > \Delta$, we *increase* $p_{\min}$ to make risk more expensive, forcing agents towards safer, longer paths. This bisection process continues until convergence or a budget of iterations is exhausted. By enabling agents to "buy" risk based on their marginal utility, WALRIS achieves a more globally coordinated and efficient distribution of the safety budget than greedy methods. Since the allocation layers are heuristic in nature, Supplement Sec. B.1 discusses their optimality and completeness properties.

4 Experiments

Our experiments evaluate our framework against several baselines along three key questions:

Q1 (Adaptability): Does the planner effectively leverage varying risk budgets Δ to trade-off safety and efficiency?
Q2 (Goal Success): Can it maintain high success rates for distant goals while enforcing a global risk bound?
Q3 (Scalability): Do these advantages persist as the number of agents increases?

Environments. We use a simple 2D navigation task and visually rich indoor scenes. For 2D navigation, we use the Central Obstacle map from Feng, Parimi, and Williams, with state $s = (x, y) \in \mathbb{R}^2$ and actions $a = (dx, dy) \in [-1, 1]^2$. The per-state risk cost

$$C(s) = \begin{cases} 2 - 2h(s)/r & 0 \leq h(s) \leq r, \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

depends on the distance $h(s)$ to the nearest obstacle boundary, with radius of influence r ($r = 10$ in 2D, $r = 1$ in visual tasks), yielding higher risk near obstacles and zero risk in free space. The GCRL agent is trained with sparse rewards, receiving -1 per step. For visual navigation, we adopt four ReplicaCAD scenes (Straub et al. 2019) in Habitat-Sim (Szot et al. 2021; Savva et al. 2019; Puig et al. 2023). Agents receive first-person RGB observations,

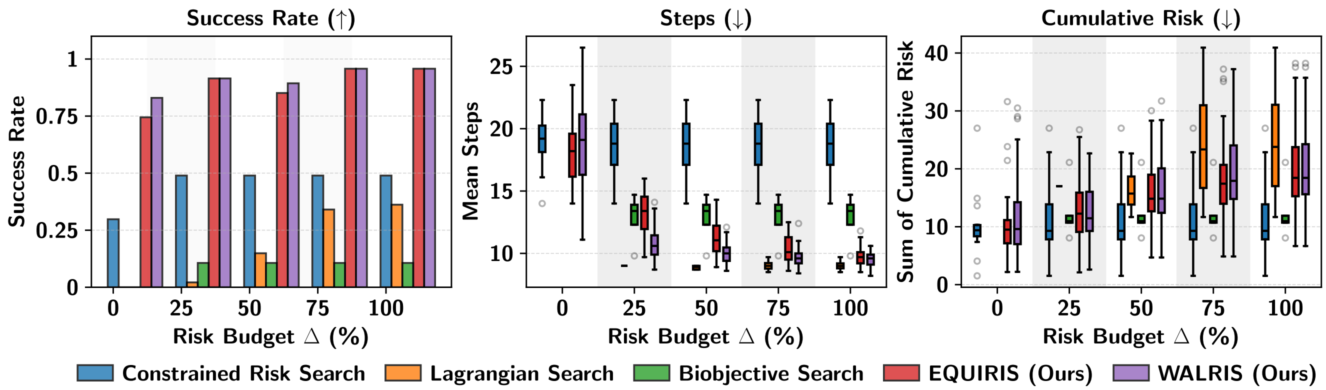

Figure 4: Quantitative performance on the 2D point environment (10 agents, Hard difficulty) as a function of the risk budget Δ. **Left:** Success rate. **Center:** Average steps (conditioned on success). **Right:** Total cumulative risk. **EQUIRIS** and **WALRIS** (Ours) maintain high success rates at strict budgets ($\Delta = 0\%$) where Lagrangian and Biobjective baselines fail. As Δ increases (moving right), our planner reduces travel time (Center), smoothly transitioning from safe detours to efficient trajectories.

represented as 32×32 images from the four cardinal directions concatenated into a panoramic view. The action space matches the 2D setting, and we reuse the same critic architectures from Feng, Parimi, and Williams.

Defining risk bounds. Since our framework operates under a user-specified global risk budget Δ, we first calibrate a meaningful range per instance to ensure consistent benchmarking. For each instance, we run standard CBS on the learned waypoint graph twice, once to minimize total path length yielding an upper risk bound, $\overline{\Delta}$ (the risk incurred by the shortest-path solution), and once to minimize total risk, yielding a lower risk bound, $\underline{\Delta}$. This defines an instance-specific interval $[\underline{\Delta}, \overline{\Delta}]$. We then evaluate all methods at five risk levels $(0\%, 25\%, 50\%, 75\%, \text{ and } 100\%)$ within this interval, modeling user preferences ranging from strong risk aversion to aggressive efficiency. This calibration is strictly for experimental rigor. In practice, a user could specify a single budget Δ, and our framework can be applied directly without these auxiliary CBS runs.

Evaluation protocol. For each environment (5 total), agent count ($N \in \{5, 10\}$) and difficulty level, we generate 50 problem instances by sampling start-goal pairs at different distances, yielding 1500 distinct problems. We define difficulty using the learned waypoint-graph diameter D: easy, medium, and hard instances have start-goal shortest-path distances concentrated around $D/8$, $D/4$, and $D/2$, respectively. Each trial has a time limit of $60 \times N$ seconds. We report three metrics: (i) **Success Rate** (fraction of collision-free and risk-compliant runs), (ii) **Average Steps** (conditioned on success), (iii) **Cumulative Risk** ($\sum_i \rho(\pi_i)$). All experiments use fixed random seeds for problem generation and policy evaluation. Unless stated otherwise, results in the main text use a uniform initial risk allocation at the root. An ablation in the Supplement Sec. A.1 compares this to alternative initializations. For WALRIS, each agent a_i uses a small symmetric neighborhood $\mathcal{N}_i = \{\delta_i - \eta, \delta_i, \delta_i + \eta\} \cap [\delta_i^{\min}, \delta_i^{\max}]$. We set the neighborhood step size $\eta = 0.05 \cdot \Delta$, the price bisection tolerance to $\varepsilon = 10^{-3}$, and cap the number of WALRIS iterations at $K_{\max} = 20$. Finally, to validate practical applicability, we integrated the planner into a ROS2 stack (Macenski et al. 2022) to command multiple Crazyflie drones in Gazebo and hardware. See Supplement Sec. D for additional details and the project website for videos.

Baselines. We compare against three baselines that all use the same learned waypoint graph. **Constrained Risk Search**, follows Feng, Parimi, and Williams and prunes edges whose predicted risk exceeds a threshold to form a *safe* graph and then runs CBS to minimize path length. **Lagrangian Search** instead runs CBS on the full graph with a single edge weight given by a linear scalarization of distance and risk using the learned Lagrange multiplier. Finally, **Bi-Objective Search** (MO-CBS) performs a multi-objective CBS search (Ren, Rathinam, and Choset 2021) on the full graph to approximate the (distance, risk) Pareto front, selecting the shortest path solution that satisfies Δ.

Q1: Adaptability to user-specified risk: Qualitative analysis (Figures 1, 2, and 3) confirms that our framework effectively uses the global budget Δ as a tunable control knob. At the tightest setting ($\Delta = \underline{\Delta}$), the planner generates conservative routes that maintain large clearance from high-risk regions, at the expense of longer paths. As Δ is relaxed (e.g., 25% of the interval), the planner produces more direct trajectories that pass closer to high-risk regions. This contrasts sharply with Lagrangian and Biobjective Search, which tend to lock into a single behavioral mode regardless of the specific constraint.

These qualitative patterns are backed by aggregate statistics (Figures 4 and 5). Our approach (red and purple) exhibits a clear monotonic trend. At low budgets, average steps are high and total cumulative risk is low. As Δ increases, the trend inverts in a controlled manner. Notably, WALRIS capitalizes on the available budget more effectively than EQUIRIS, consistently finding shorter paths at medium-to-high Δ. This confirms that the market-based mechanism

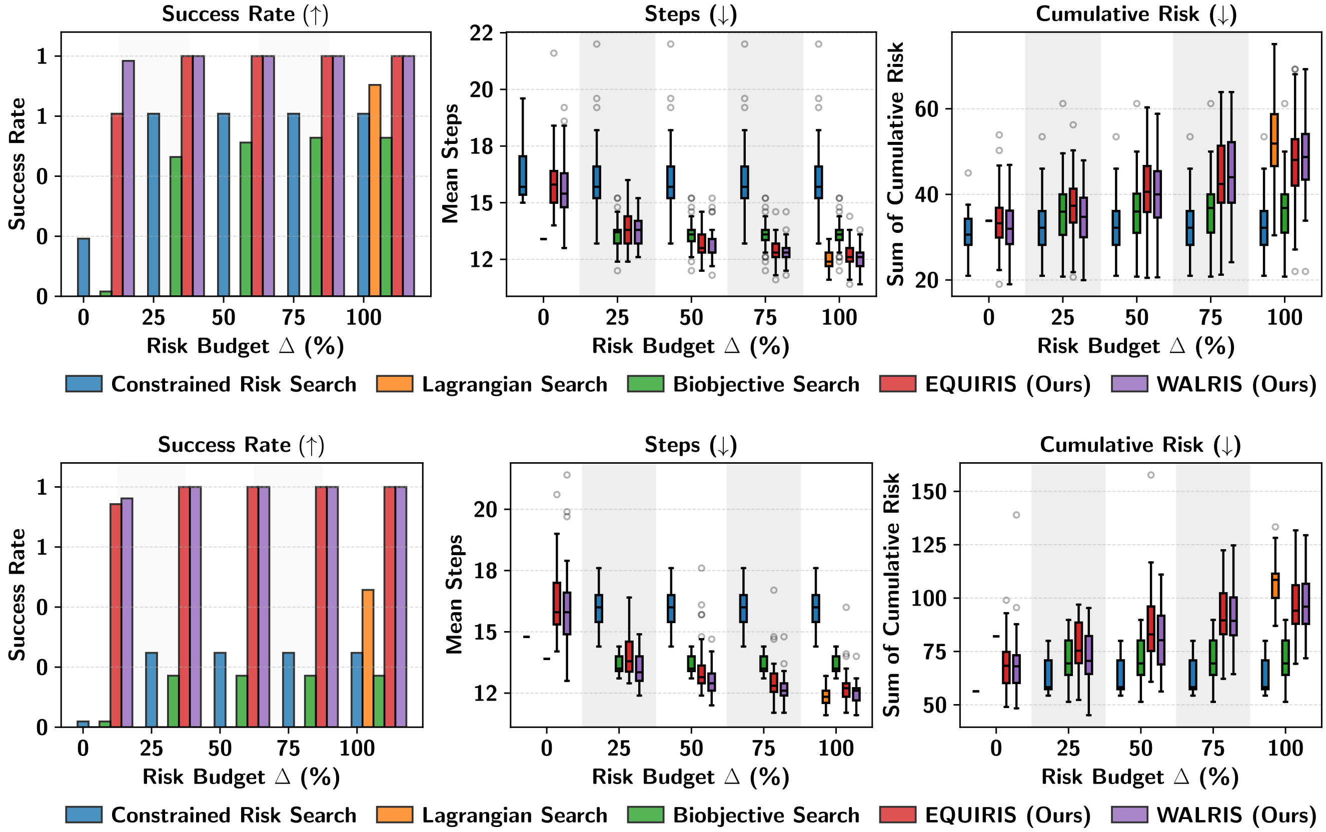

Figure 5: Quantitative results on the visual navigation task (Scene: SC2 Staging 08, Hard difficulty) for 5 agents (top) and 10 agents (bottom). **Left:** Success rate. **Center:** Average steps. **Right:** Total cumulative risk. Our strategies (EQUIRIS and WALRIS) demonstrate superior robustness, maintaining near-perfect success rates even at tight risk budgets ($\Delta \approx 0\%$) where baselines like Lagrangian and Biobjective Search struggle or fail completely. Additionally, the step plots (Center) confirm that our planner effectively exploits relaxed budgets to reduce travel time, scaling reliably to higher agent counts.

succeeds in "spending" the risk resource to purchase efficiency where the greedy scheme falls short.

Q2: Goal success with safety: Our framework achieves high success rates while enforcing the global risk bound. Across all risk levels, both strategies outperform the strongest baselines. The main degradation occurs at the tightest budget ($\Delta = \underline{\Delta}$), where the feasible solution space is extremely narrow. Here, EQUIRIS's success rate drops to 74% on the 2D environment and 76% on the visual domain, while WALRIS remains more robust. This behavior is expected as EQUIRIS relies on a single surplus-deficit redistribution, whereas WALRIS can often recover from difficult constraints via iterative price adjustments that explore the allocation space more thoroughly. Importantly, once Δ is relaxed slightly above this extreme setting, both EQUIRIS and WALRIS quickly recover to near-perfect success rates.

Q3: Scalability to more agents: We assess scalability by increasing the team size from 5 to 10 agents (Figure 5), consistent with prior work (Ren, Rathinam, and Choset 2021). Despite the increased density of inter-agent conflicts, the characteristic safety-efficiency trade-off persists. As Δ increases, average steps monotonically decrease for both group sizes, indicating that the benefits of risk allocation do not collapse under congestion. WALRIS demonstrates superior scalability, maintaining high success rates across Δ even with 10 agents. While larger teams provide a larger pool of surplus risk for EQUIRIS to harvest, its greedy updates struggle to coordinate tight interactions as effectively as WALRIS's price-mediated negotiation. Overall, these results suggest that the proposed IRA layer scales gracefully across team sizes.

5 Conclusion

We introduced Δ-MAPF problem to overcome the conservatism of graph pruning in safe visual navigation. By treating risk as a shared resource, our approach dynamically allocates local budgets via Iterative Risk Allocation (IRA). This allows agents to selectively accept risk to shorten paths, while ensuring the global mission remains safe. Extensive experiments on 2D and visual environments, validated by hardware-in-the-loop multi-drone demonstration, confirm that our strategies yield superior performance and a tunable, scalable trade-off between safety and efficiency.

Acknowledgments

We gratefully acknowledge support from the Defence Science and Technology Agency, Singapore. We thank Meng Feng for sharing the RL models used in this paper; Morgan Schaefer, Shashank Swaminathan, and Kristoff Misquitta for careful proofreading; and Shreya Chaudhary for help with the hardware demonstrations. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations or agencies.

References

Altman, E. 2021. *Constrained Markov decision processes*. Routledge.

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2021. Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem. *Proceedings of the International Symposium on Combinatorial Search*, 5: 19–27.

Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Tolpin, D.; Betzalel, O.; and Shimony, E. 2015. ICBS: improved conflict-based search algorithm for multi-agent pathfinding. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, 740–746. AAAI Press. ISBN 9781577357384.

Cheng, R.; Orosz, G.; Murray, R. M.; and Burdick, J. W. 2019. End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks. arXiv:1903.08792.

Chiang, H.-T. L.; Faust, A.; Fiser, M.; and Francis, A. 2019. Learning Navigation Behaviors End-to-End with AutoRL. arXiv:1809.10124.

Cohen, L.; Uras, T.; Kumar, T.; and Koenig, S. 2021. Optimal and Bounded-Suboptimal Multi-Agent Motion Planning. *Proceedings of the International Symposium on Combinatorial Search*, 10: 44–51.

Eysenbach, B.; Salakhutdinov, R. R.; and Levine, S. 2019. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems*, 32.

Feng, M.; Parimi, V.; and Williams, B. 2025. Safe Multi-Agent Navigation guided by Goal-Conditioned Safe Reinforcement Learning. arXiv:2502.17813.

Im, J.; and Lee, B.-Y. 2023. Multi-Agent Inspection Path Planning with Large-Scale Vehicle Routing Problem. *Journal of Aerospace Information Systems*, 20: 1–9.

Levy, A.; Platt, R.; and Saenko, K. 2019. Hierarchical Reinforcement Learning with Hindsight. arXiv:1805.08180.

Li, J.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2019. Disjoint Splitting for Multi-Agent Path Finding with Conflict-Based Search. In *ICAPS*, 279–283. AAAI Press.

Lynch, C.; Khansari, M.; Xiao, T.; Kumar, V.; Tompson, J.; Levine, S.; and Sermanet, P. 2019. Learning Latent Plans from Play. arXiv:1903.01973.

Ma, H.; Harabor, D.; Stuckey, P. J.; Li, J.; and Koenig, S. 2018. Searching with Consistent Prioritization for Multi-Agent Path Finding. arXiv:1812.06356.

Ma, H.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2017. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. arXiv:1705.10868.

Macenski, S.; Foote, T.; Gerkey, B.; Lalancette, C.; and Woodall, W. 2022. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66): eabm6074.

Mirowski, P.; Pascanu, R.; Viola, F.; Soyer, H.; Ballard, A. J.; Banino, A.; Denil, M.; Goroshin, R.; Sifre, L.; Kavukcuoglu, K.; Kumaran, D.; and Hadsell, R. 2017. Learning to Navigate in Complex Environments. arXiv:1611.03673.

Nachum, O.; Gu, S.; Lee, H.; and Levine, S. 2018. Data-Efficient Hierarchical Reinforcement Learning. arXiv:1805.08296.

Nair, A.; McGrew, B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2018. Overcoming Exploration in Reinforcement Learning with Demonstrations. arXiv:1709.10089.

Olkin, J.; Parimi, V.; and Williams, B. 2024. Multi-Agent Vulcan: An Information-Driven Multi-Agent Path Finding Approach. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 10253–10259.

Ono, M. 2012. *Robust, Goal-Directed Plan Execution with Bounded Risk*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Ono, M.; and Williams, B. C. 2008. Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint. In *2008 47th IEEE Conference on Decision and Control*, 3427–3432. IEEE.

Ono, M.; and Williams, B. C. 2010. Market-based risk allocation for multi-agent systems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, 1423–1424. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9780982657119.

Parimi, V.; and Williams, B. C. 2025. Diffusion-Guided Multi-Arm Motion Planning. arXiv:2509.08160.

Peralta, F.; Pearce, M.; Poloczek, M.; Reina, D. G.; Toral, S.; and Branke, J. 2022. Multi-objective path planning for environmental monitoring using an autonomous surface vehicle. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, 747–750. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392686.

Pong, V.; Gu, S.; Dalal, M.; and Levine, S. 2020. Temporal Difference Models: Model-Free Deep RL for Model-Based Control. arXiv:1802.09081.

Puig, X.; Undersander, E.; Szot, A.; Cote, M. D.; Partsey, R.; Yang, J.; Desai, R.; Clegg, A. W.; Hlavac, M.; Min, T.; Gervet, T.; Vondrus, V.; Berges, V.-P.; Turner, J.; Maksymets, O.; Kira, Z.; Kalakrishnan, M.; Malik, J.; Chaplot, D. S.; Jain, U.; Batra, D.; Rai, A.; and Mottaghi, R. 2023. Habitat 3.0: A Co-Habitat for Humans, Avatars and Robots.

Ren, Z.; Rathinam, S.; and Choset, H. 2021. Multi-objective Conflict-based Search for Multi-agent Path Finding. *CoRR*, abs/2101.03805.

Savva, M.; Kadian, A.; Maksymets, O.; Zhao, Y.; Wijmans, E.; Jain, B.; Straub, J.; Liu, J.; Koltun, V.; Malik, J.; Parikh, D.; and Batra, D. 2019. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Schaul, T.; Horgan, D.; Gregor, K.; and Silver, D. 2015. Universal Value Function Approximators. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 1312–1320. Lille, France: PMLR.

Shaoul, Y.; Mishani, I.; Vats, S.; Li, J.; and Likhachev, M. 2024. Multi-Robot Motion Planning with Diffusion Models. *arXiv e-prints*, arXiv:2410.03072.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219: 40–66.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.; et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 10, 151–158.

Stewart, B. S.; and White, C. C. 1991. Multiobjective A*. *J. ACM*, 38(4): 775–814.

Straub, J.; Whelan, T.; Ma, L.; Chen, Y.; Wijmans, E.; Green, S.; Engel, J. J.; Mur-Artal, R.; Ren, C.; Verma, S.; Clarkson, A.; Yan, M.; Budge, B.; Yan, Y.; Pan, X.; Yon, J.; Zou, Y.; Leon, K.; Carter, N.; Briales, J.; Gillingham, T.; Mueggler, E.; Pesqueira, L.; Savva, M.; Batra, D.; Strasdat, H. M.; Nardi, R. D.; Goesele, M.; Lovegrove, S.; and Newcombe, R. 2019. The Replica Dataset: A Digital Replica of Indoor Spaces. *arXiv preprint arXiv:1906.05797*.

Szot, A.; Clegg, A.; Undersander, E.; Wijmans, E.; Zhao, Y.; Turner, J.; Maestre, N.; Mukadam, M.; Chaplot, D.; Maksymets, O.; Gokaslan, A.; Vondrus, V.; Dharur, S.; Meier, F.; Galuba, W.; Chang, A.; Kira, Z.; Koltun, V.; Malik, J.; Savva, M.; and Batra, D. 2021. Habitat 2.0: Training Home Assistants to Rearrange their Habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Tuinstra, J. 2012. *Price Dynamics in Equilibrium Models: The Search for Equilibrium and the Emergence of Endogenous Fluctuations*, volume 16. Springer Science & Business Media.

Wang, Z.; and Zlatanova, S. 2016. Multi-agent based path planning for first responders among moving obstacles. *Computers, Environment and Urban Systems*, 56: 48–58.

Zhang, M.; Li, N.; Chen, Y.; Li, J.; Zhang, X.-Y.; Zhao, H.; Liu, J.; and Chen, W. 2025. Learning Verified Safe Neural Network Controllers for Multi-Agent Path Finding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(22): 23369–23377.

Zhao, H.; Zeng, X.; Chen, T.; Liu, Z.; and Woodcock, J. 2020. Learning Safe Neural Network Controllers with Barrier Certificates. arXiv:2009.09826.