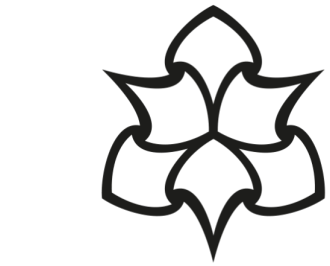

Search Algorithms in Artificial Intelligence Research and Industrial Applications

Assessment ID: 1CWK100

Submitted By:

Karlo Amir Nahro

MMU ID: 19003070

Group Details (for Section 1):

Group ID: T4

Department of Computing and Mathematics

Manchester Metropolitan University

February 7, 2026

1 Environment and Algorithm Design

1.1 Robot and Environment

1.1.1 Robot Parameters

The robot is modelled as a circular agent operating in a two-dimensional grid. A configurable robot radius of 0.5 grid units and a grid resolution of 1 unit per cell are used.

Robot motion is defined as transitions between centres of adjacent grid cells, with obstacle inflation applied for conservative collision avoidance.

1.1.2 Action Model

The robot's motion model is configurable based on connectivity:

- **4-Connected:** Constrained to cardinal directions. The cost function and heuristics utilize **Manhattan distance** (L_1 norm).
- **8-Connected:** Includes diagonal movements. The cost function and heuristics utilize **Octile distance**, assigning a cost of 1 for cardinal moves and $\sqrt{2} \approx 1.414$ for diagonal moves.

For BFS and DFS, path length is measured purely as the number of actions (unit cost per step, regardless of direction). In contrast, A* computes the cumulative path cost $g(n)$ based on the weighted distance metrics described above. Actions leading to collisions or exiting the grid boundaries are strictly invalid.

1.1.3 Environment Setup

The environment is represented as a binary occupancy grid with inflated obstacles. Experiments were conducted on small (10×10), medium (50×50), and large (100×100) grids to evaluate algorithm scalability as state-space size increases..

1.1.4 Initial and Goal States

For each grid size, five random maps generated with fixed start and goal states across all algorithms.

2 Results

Graph-based algorithms were evaluated on all grid sizes, while tree-based experiments used small and medium grids. An expansion limit of 50,000 nodes was applied to T-BFS and T-DFS.

2.1 Tree-Based Search Results

2.1.1 Visual Results

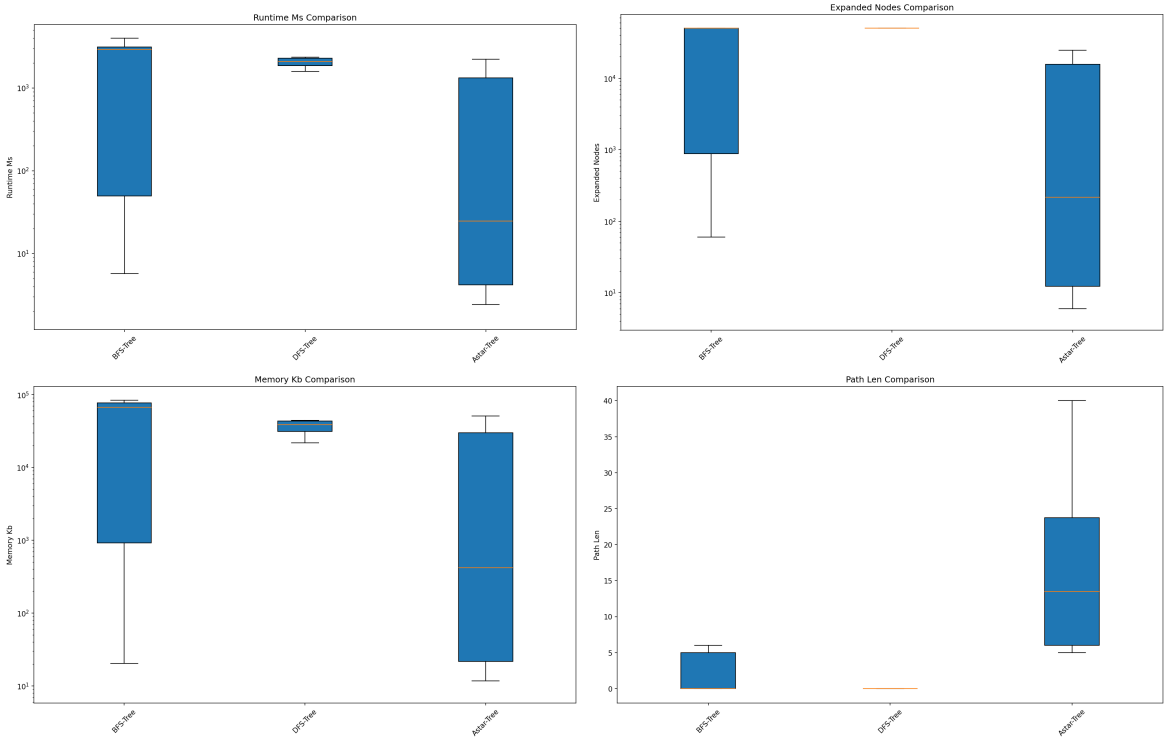

Figure 1: Tree-based search: overall results.

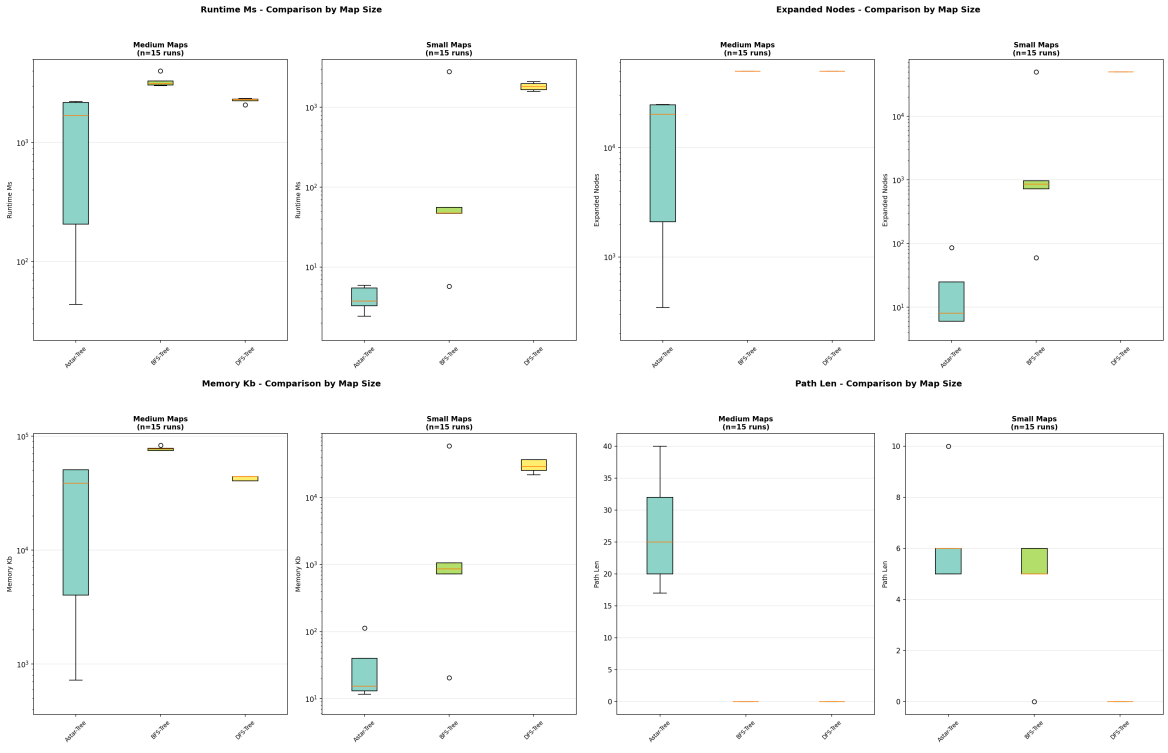

Figure 2: Tree-based search: results by grid size.

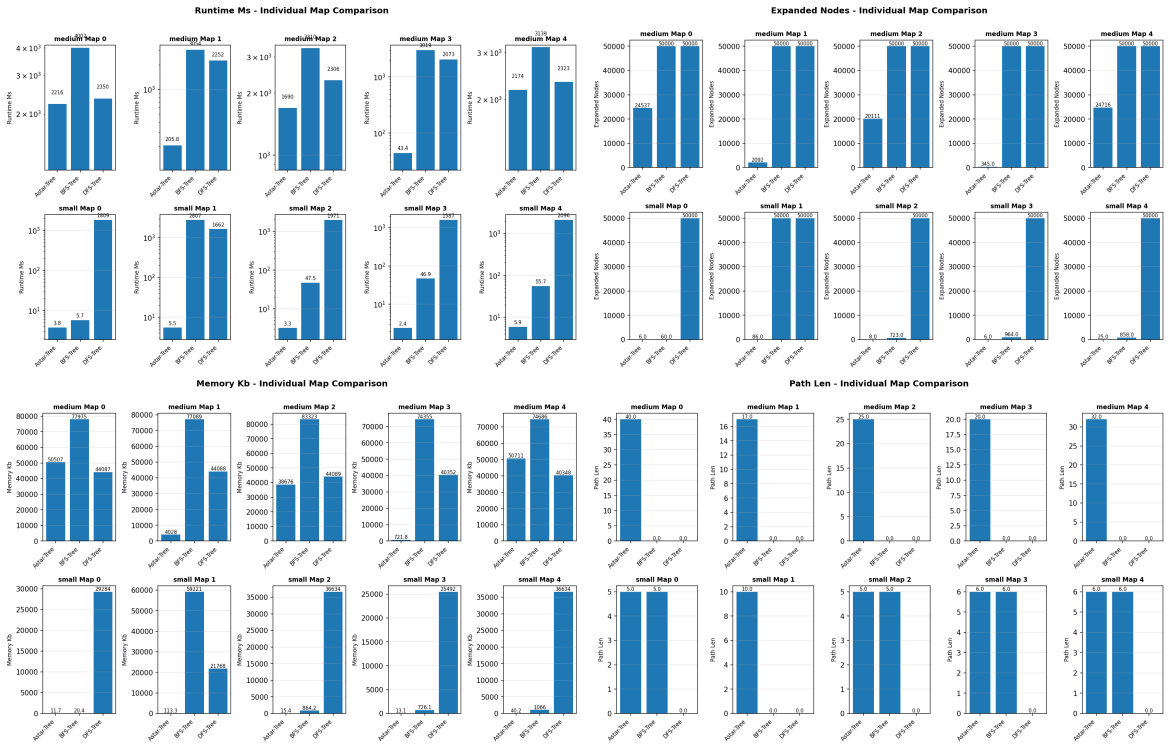

Figure 3: Tree-based search: results by map.

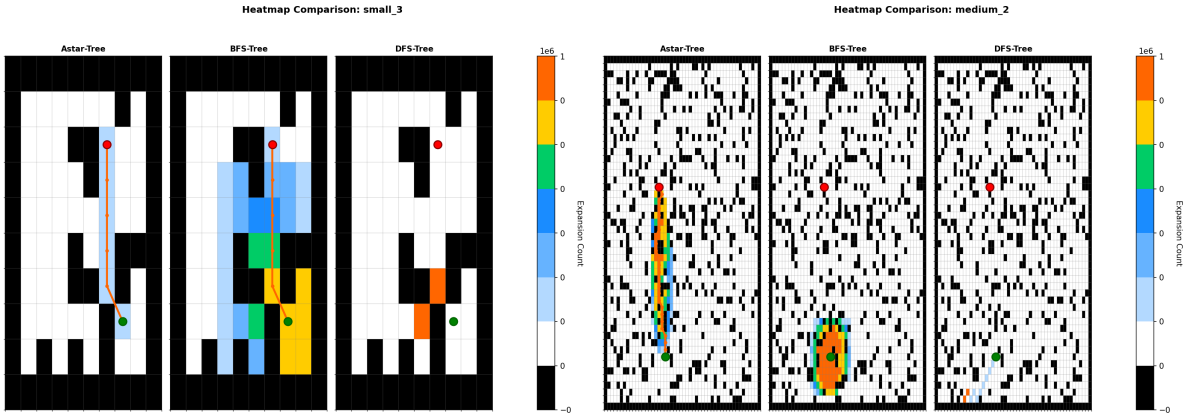

Figure 4: Tree-based search: exploration heatmaps.

Algorithm	Size	Runtime (ms)	Expanded	Memory (KB)	Path Len	Success
BFS	Small	47.52	858	864.21	5	80%
BFS	Medium	3138.83	50000	77089.05	0	0%
DFS	Small	1808.89	50000	29283.98	0	0%
DFS	Medium	2305.73	50000	44086.95	0	0%
A*	Small	3.75	8	15.39	6	100%
A*	Medium	1690.19	20111	38676.14	25	100%

Figure 5: Tree-based search: median results.

2.1.2 Observations

T-BFS found optimal paths in some small maps but failed on medium maps due to excessive node expansion, summarised in Fig. 5. Runtime and memory increased sharply with grid size (Fig. 2), causing most medium runs to terminate at the expansion limit.

T-DFS T-DFS failed to find a valid path in all environments due to unbounded deep exploration and the absence of cycle detection, rendering it neither complete nor optimal. This depth-heavy behaviour is visible in the heatmaps (Fig. 4).

T-A* successfully solved all small and medium maps with more goal-directed exploration than BFS or DFS.

2.2 Graph-Based Search Results

2.2.1 Visual Results

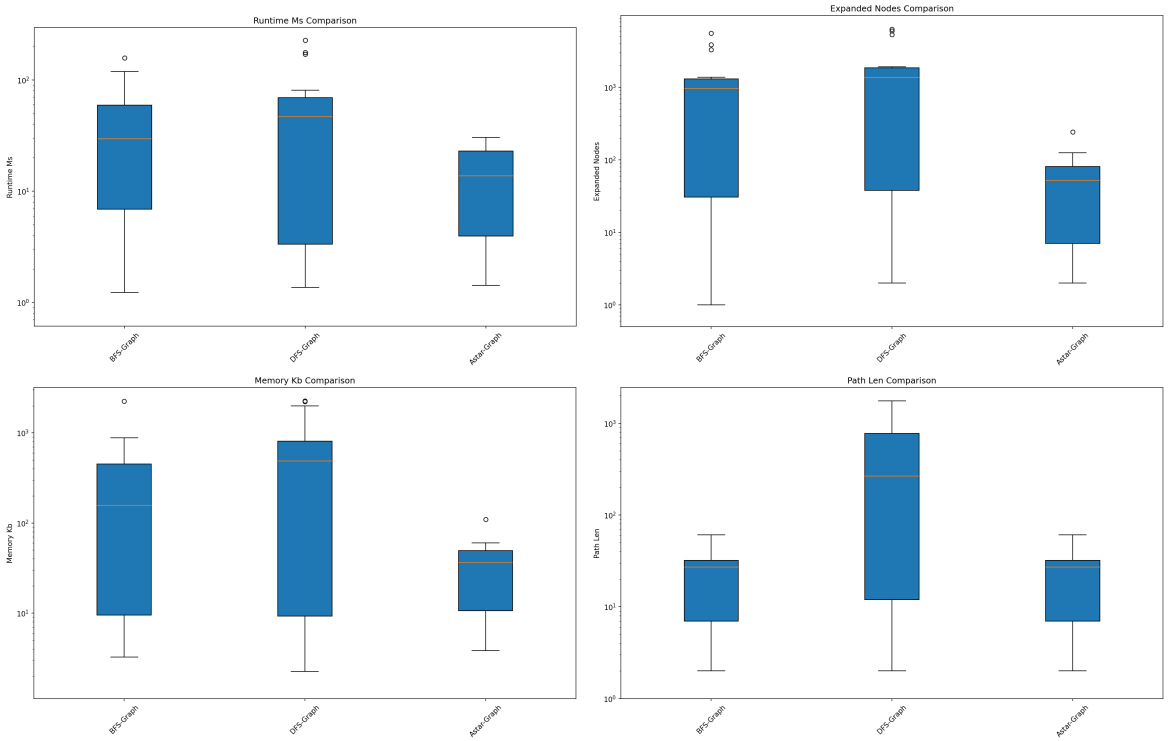

Figure 6: Graph-based search: overall results.

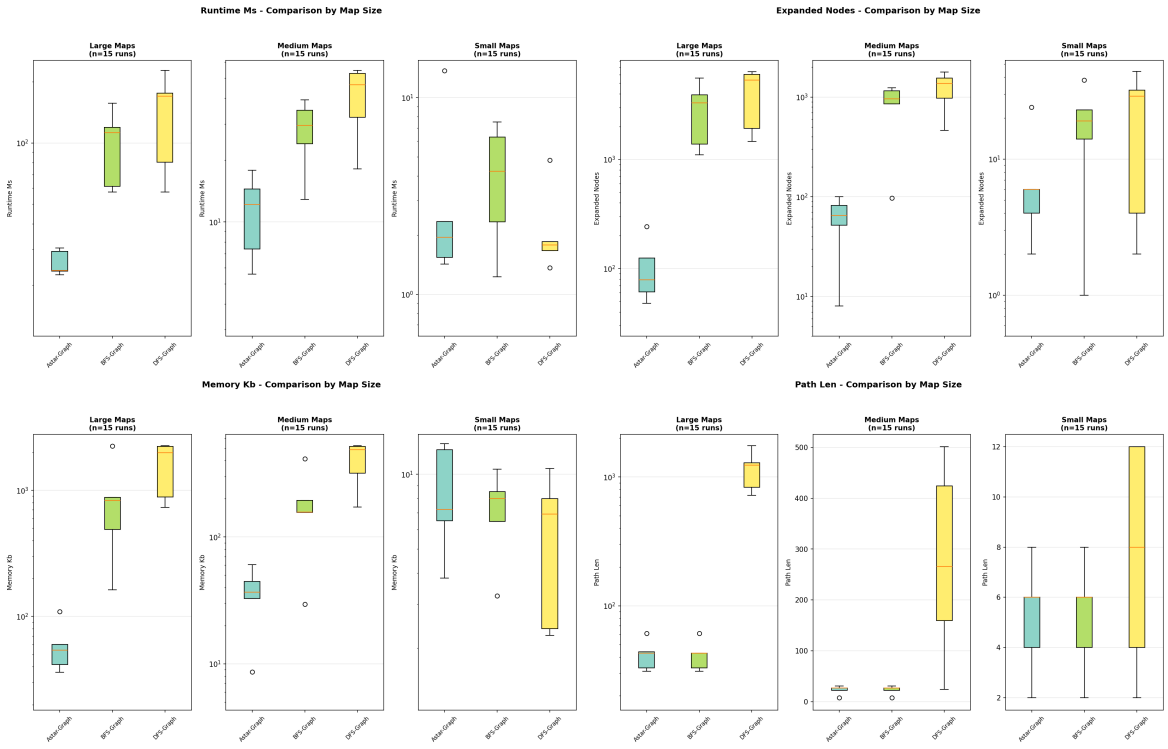

Figure 7: Graph-based search: results by grid size.

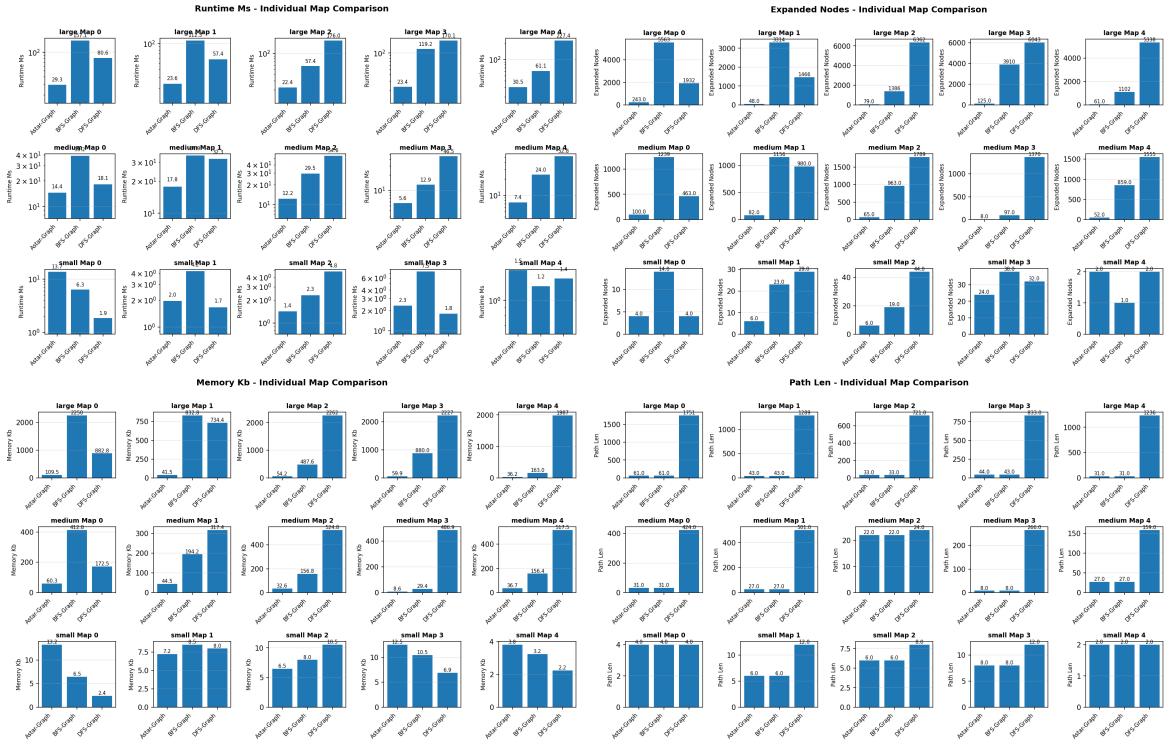

Figure 8: Graph-based search: results by map.

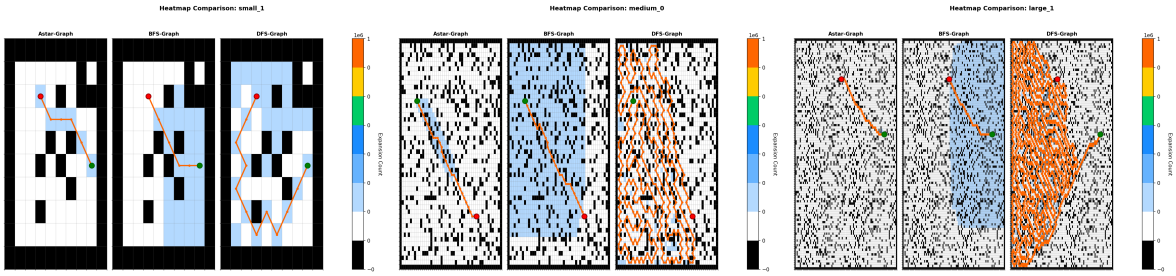

Figure 9: Graph-based search: exploration heatmaps.

Algorithm	Size	Runtime (ms)	Expanded	Memory (KB)	Path Len	Success
BFS	Small	4.22	19	7.97	6	100%
BFS	Medium	29.55	963	156.82	27	100%
BFS	Large	112.32	3314	832.81	43	100%
DFS	Small	1.78	29	6.92	8	100%
DFS	Medium	46.52	1370	486.93	266	100%
DFS	Large	170.08	5338	1987.10	1236	100%
A*	Small	1.95	6	7.22	6	100%
A*	Medium	12.18	65	36.66	27	100%
A*	Large	23.61	79	54.16	43	100%

Figure 10: Graph-based search: median runtime, expansion, memory usage, path length, and success rate.

2.2.2 Observations

G-BFS G-BFS found shortest paths in all environments; however, frontier-wide expansion causes runtime and memory usage to grow rapidly with grid size (Figs. 6–7).

G-DFS found valid paths in all environments, but path length varied substantially with grid size, as shown in Fig. 10 and Fig. 8. Heatmaps (Fig. 9) show deep, non-goal-directed exploration.

G-A* achieved a 100% success rate with low node expansion and stable runtime across all grid sizes (Fig. 10). Heatmaps (Fig. 9) show strongly goal-directed exploration.

2.3 Graph vs Tree Algorithm-Specific Results

2.3.1 BFS: Graph-Based vs Tree-Based

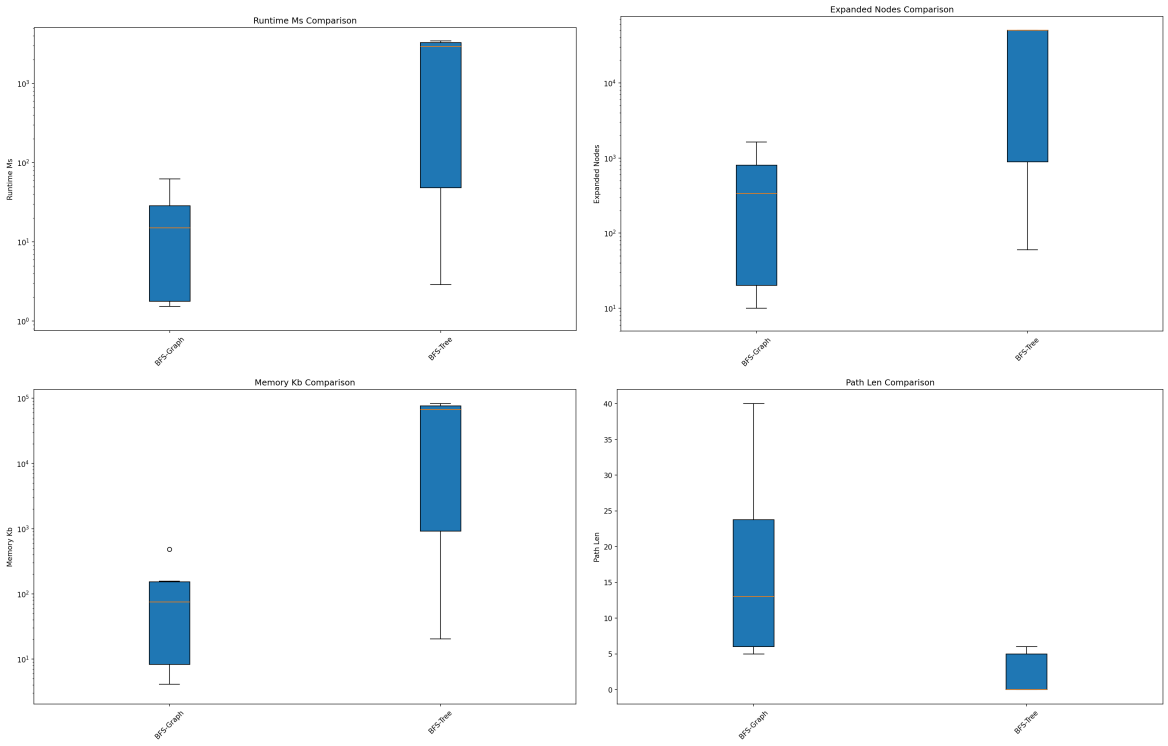

Figure 11: BFS (graph vs tree): overall results.

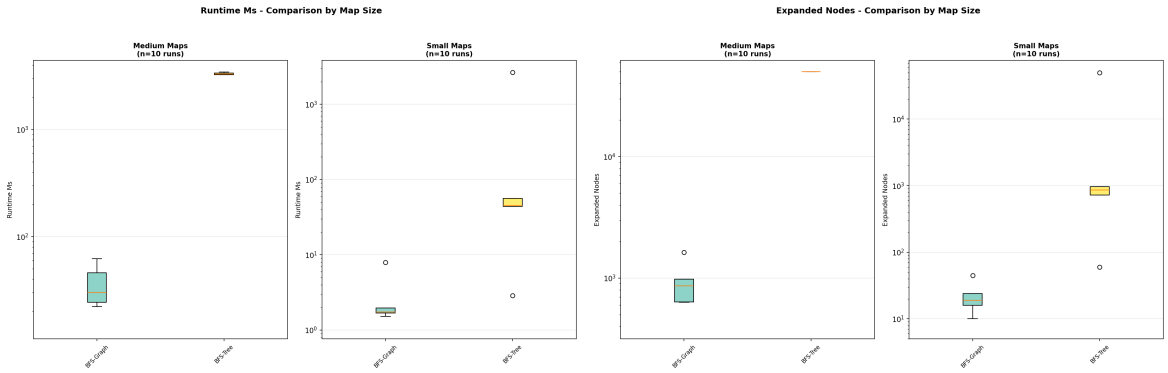

Figure 12: BFS (graph vs tree): results by size.

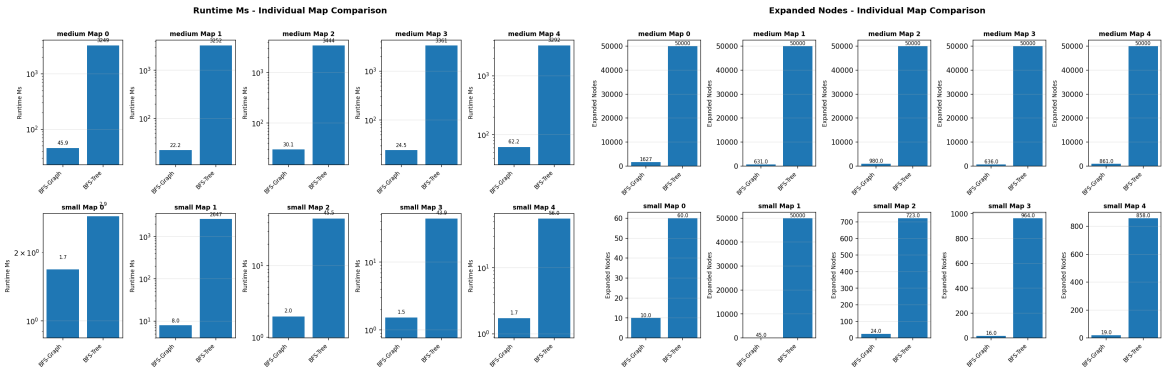

Figure 13: BFS (graph vs tree): results by map.

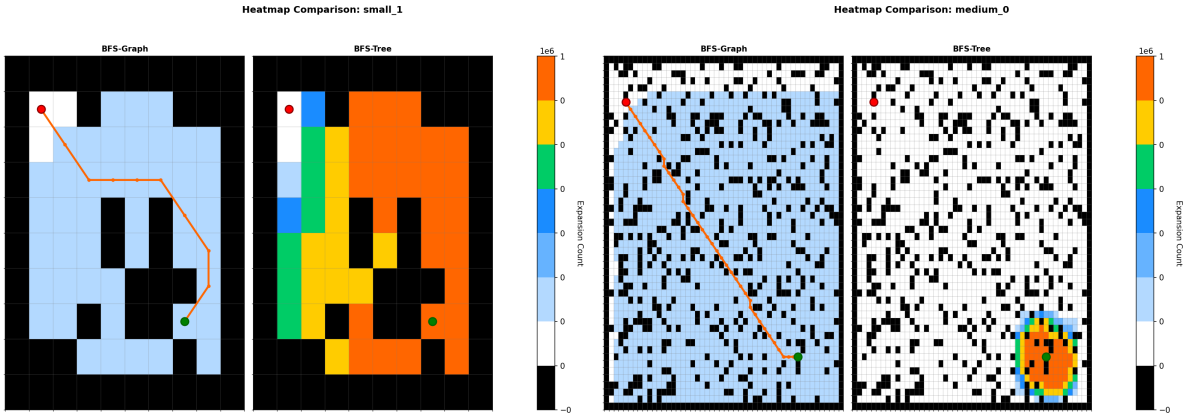

Figure 14: BFS (graph vs tree): exploration heatmaps.

Mode	Size	Runtime (ms)	Expanded	Memory (KB)	Path Len	Success
Graph	Small	1.73	19	8.01	6	100%
Graph	Medium	30.06	861	156.41	25	100%
Tree	Small	45.50	858	863.52	5	80%
Tree	Medium	3292.21	50000	77124.35	0	0%

Figure 15: BFS (graph vs tree): median runtime, expansion, memory, and success rate.

Observations G-BFS consistently found optimal paths, whereas T-BFS failed on medium grids due to repeated state expansion. Fig. 15 shows higher runtime, expansions, and memory usage for T-BFS, with broader exploration in Fig. 14.

2.3.2 DFS: Graph-Based vs Tree-Based

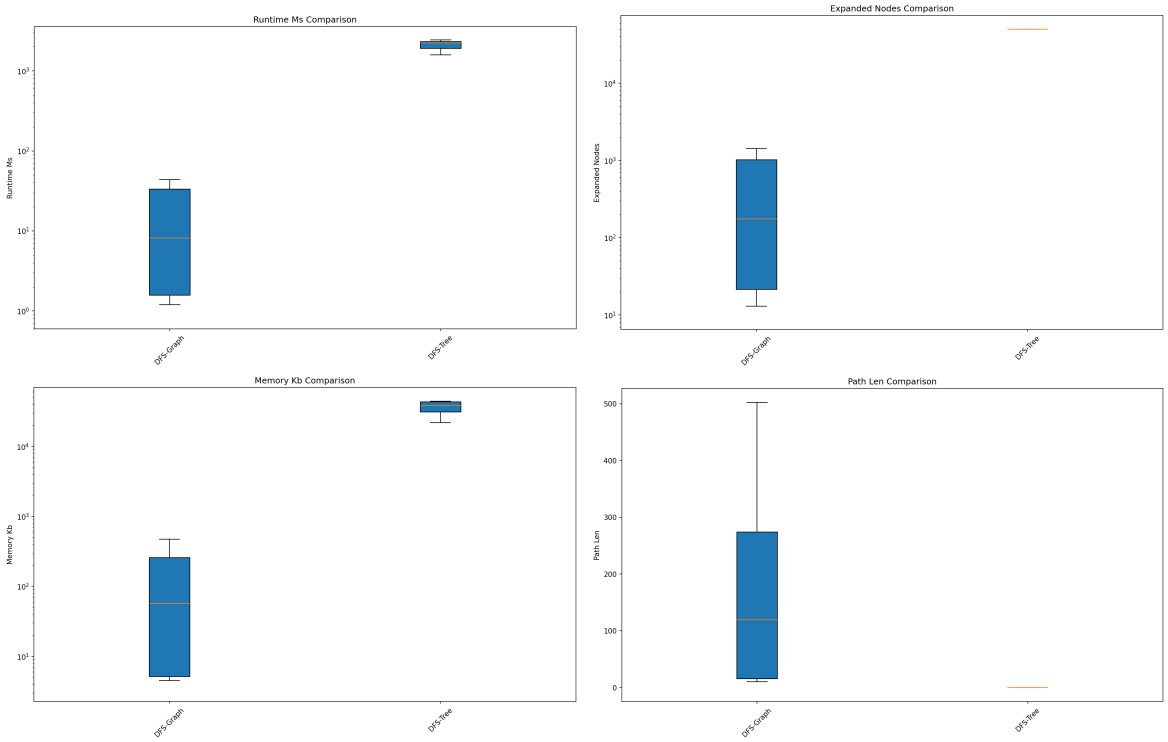

Figure 16: DFS (graph vs tree): overall results.

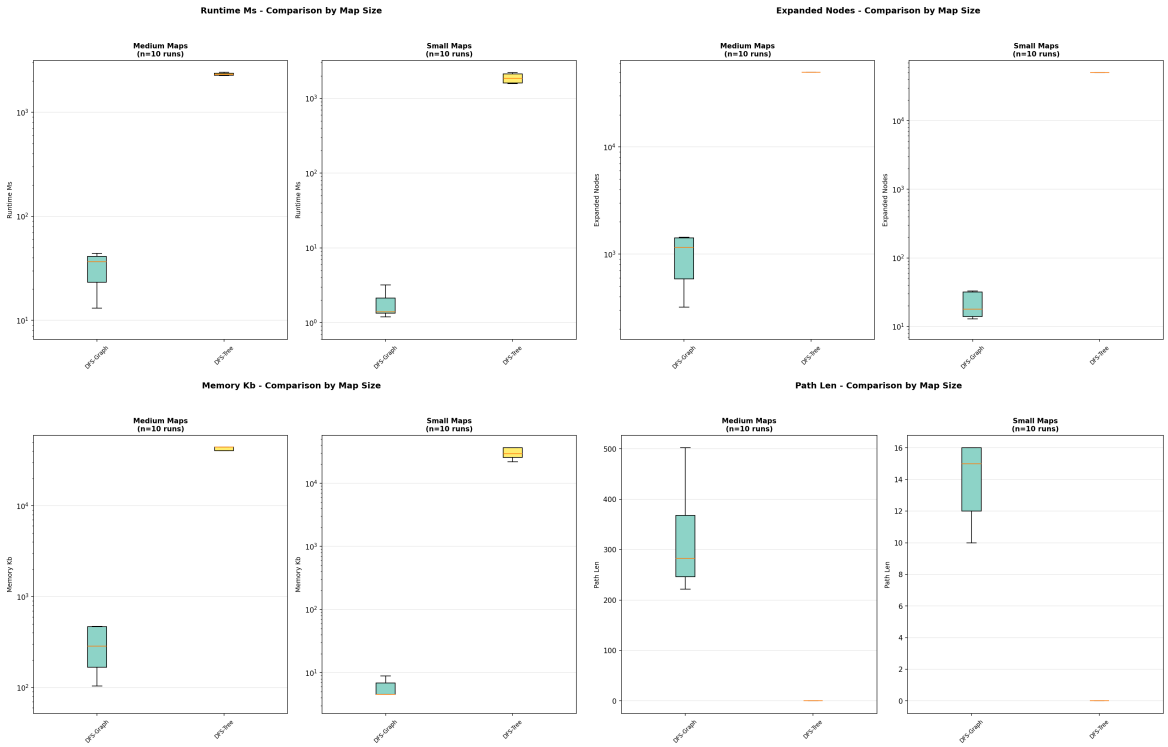

Figure 17: DFS (graph vs tree): results by size.

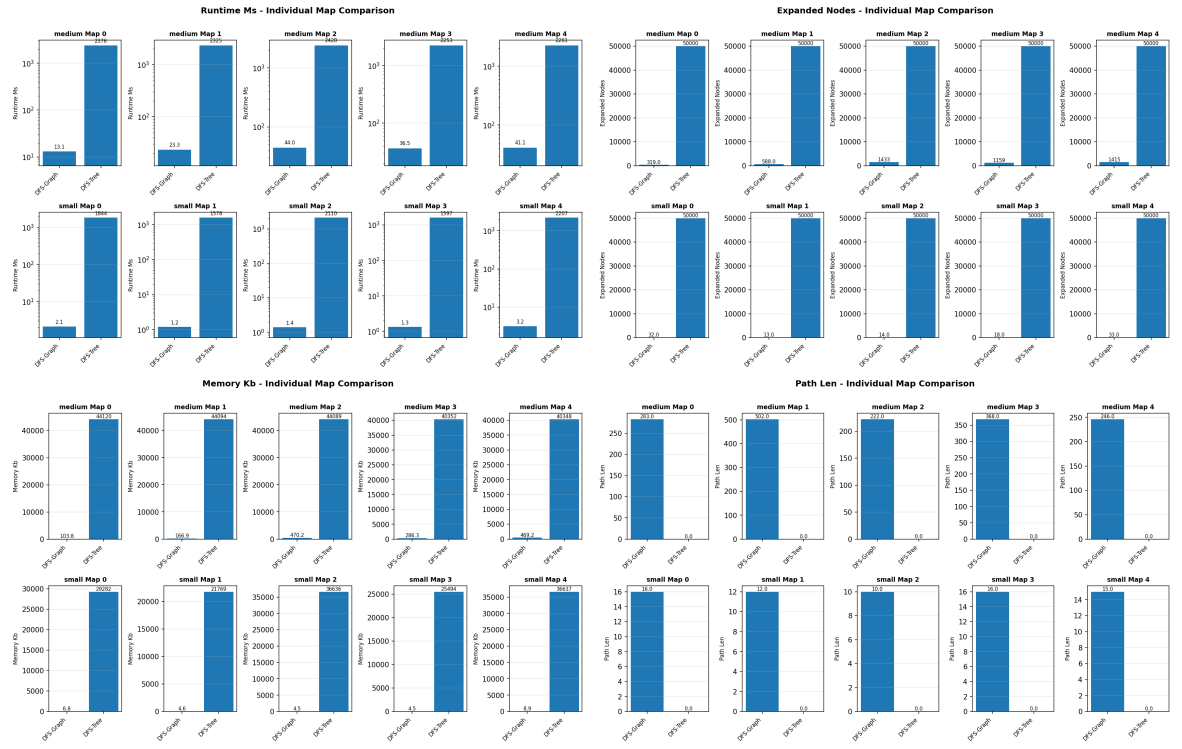

Figure 18: DFS (graph vs tree): results by map.

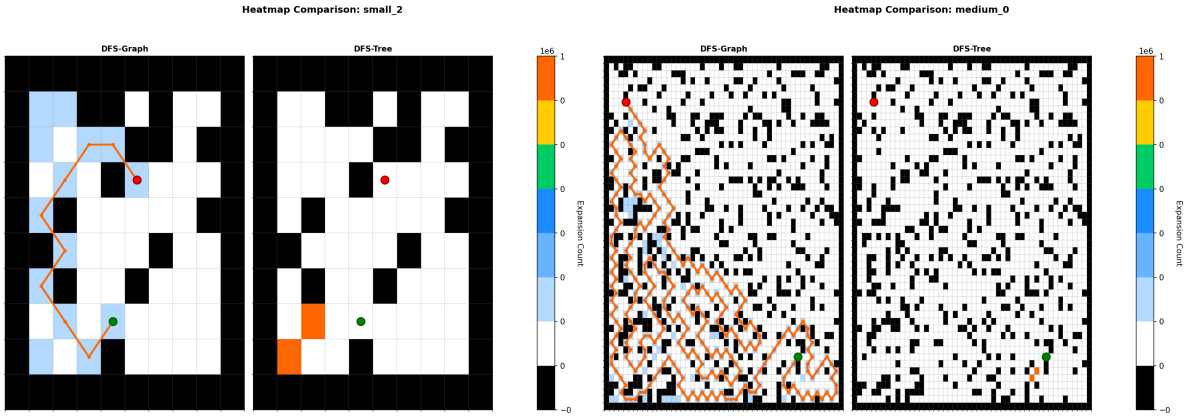

Figure 19: DFS (graph vs tree): exploration heatmaps.

Mode	Size	Runtime (ms)	Expanded	Memory (KB)	Path Len	Success
Graph	Small	1.40	18	4.55	15	100%
Graph	Medium	36.50	1159	286.30	283	100%
Tree	Small	1844.33	50000	29281.98	0	0%
Tree	Medium	2325.33	50000	44088.70	0	0%

Figure 20: DFS (graph vs tree): median runtime, expansion, memory, and success rate.

Observations G-DFS consistently found a path, whereas T-DFS failed due to deep looping that exhausted the expansion limit. Fig. 20 shows the resulting gap in runtime, node expansions, and memory usage, while Fig. 19 highlights the depth-heavy exploration of T-DFS.

2.3.3 A*: Graph-Based vs Tree-Based

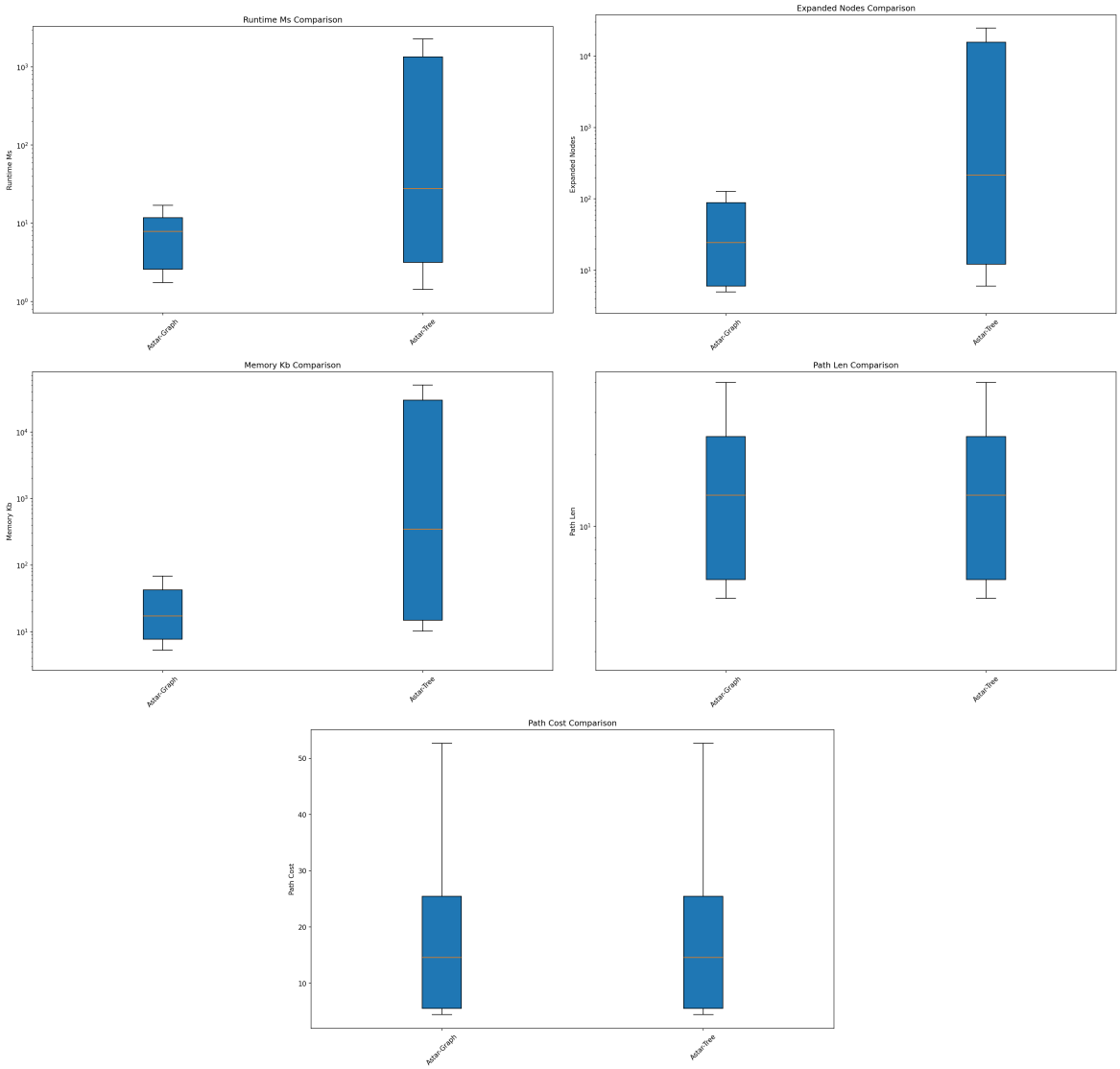

Figure 21: A* (graph vs tree): overall results.

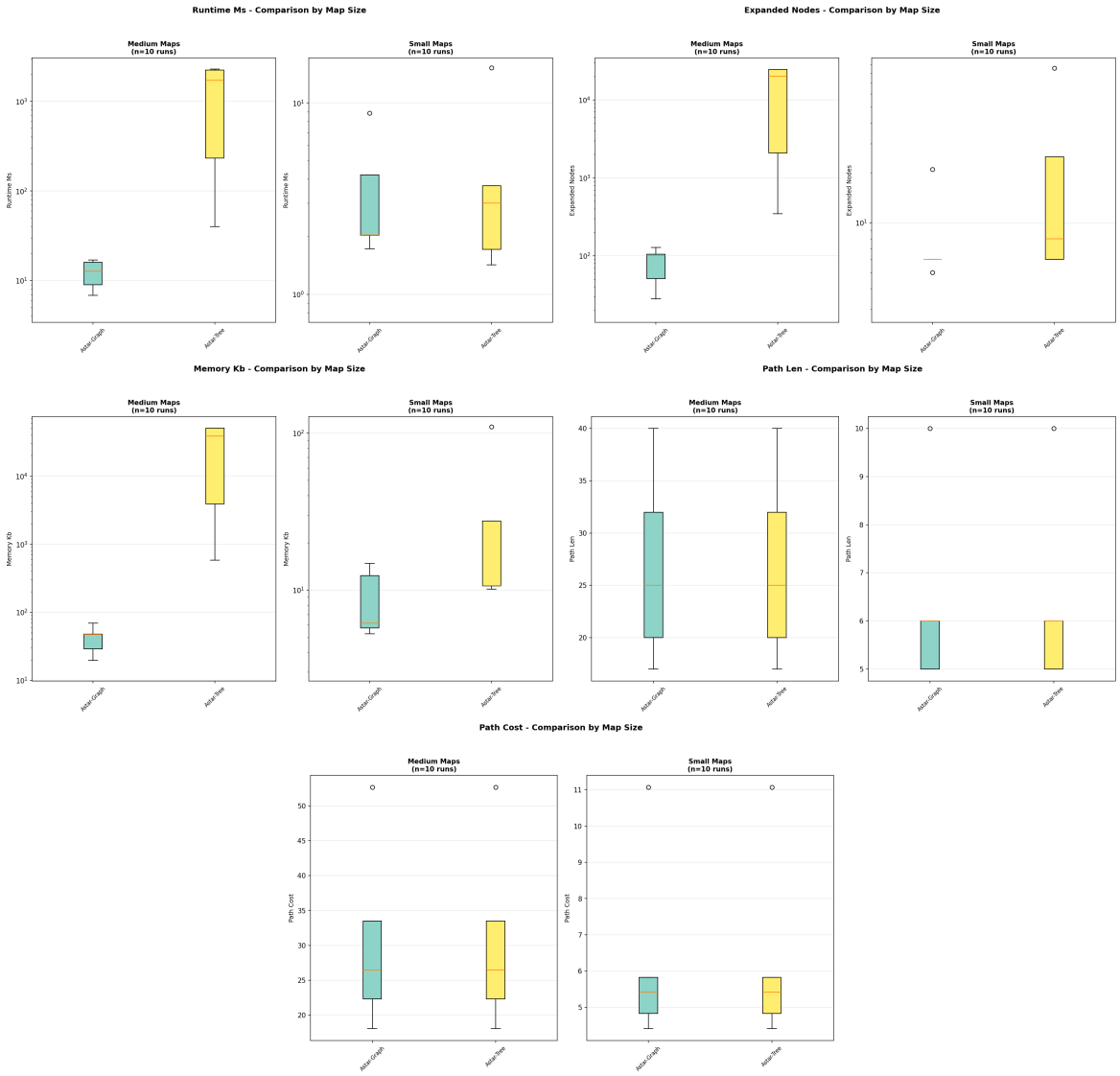

Figure 22: A* (graph vs tree): results by size.

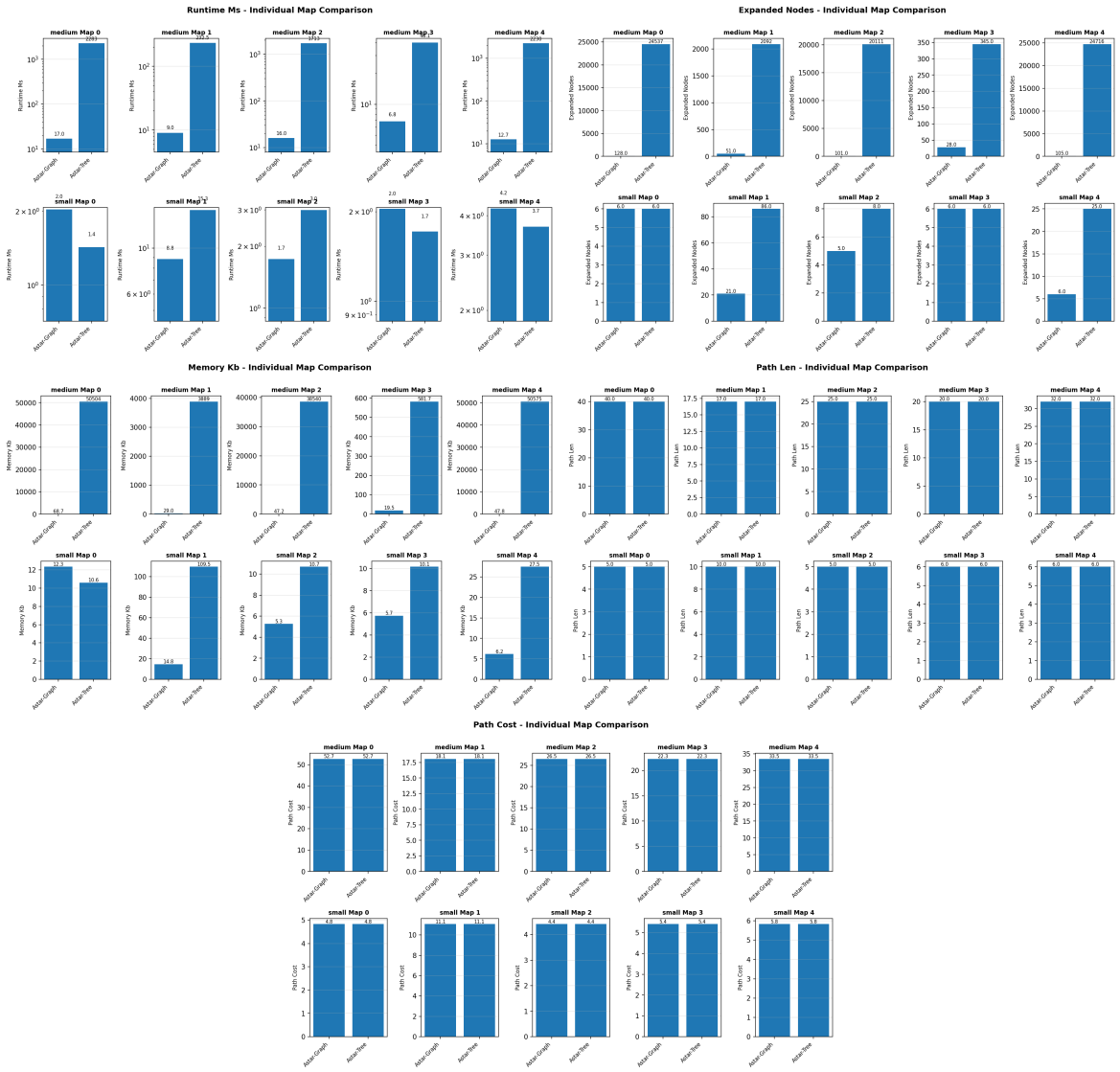

Figure 23: A* (graph vs tree): results by map.

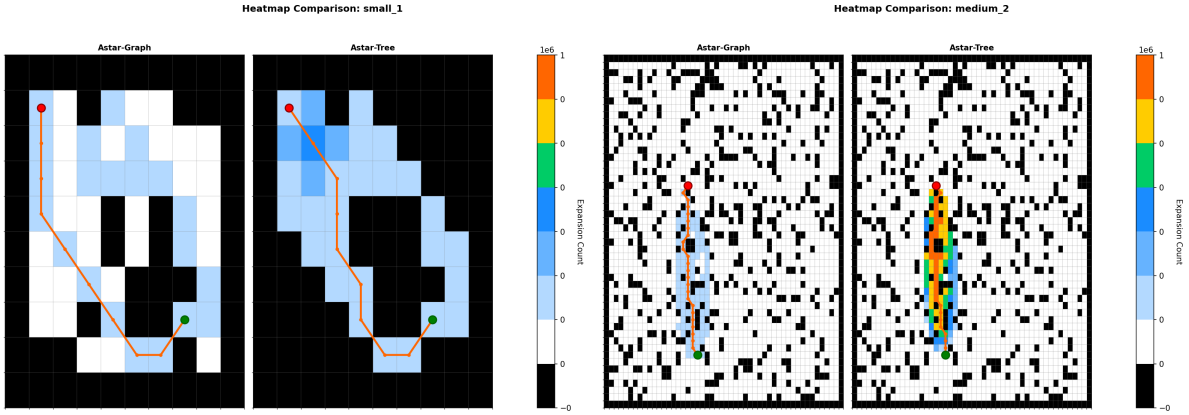

Figure 24: A* (graph vs tree): exploration heatmaps.

Mode	Size	Runtime (ms)	Expanded	Memory (KB)	Path Len	Path Cost	Success
Graph	Small	2.04	6	6.16	6	5.41	100%
Graph	Medium	12.71	101	47.17	25	26.49	100%
Tree	Small	2.99	8	10.70	6	5.41	100%
Tree	Medium	1712.82	20111	38540.36	25	26.49	100%

Figure 25: A* (graph vs tree): median runtime, expansion, memory, path cost, and success rate.

Observations Both versions of A* solved small and medium maps with identical path costs. However, T-A* shows sharply higher runtime and memory usage on medium grids due to repeated state expansion, as summarised in Fig. 25. Figs. 22 and 23 show that G-A* maintains stable performance as problem size increases.

3 Evaluation and Comparison

Algorithm's performance comparison on runtime, path quality, and resource usage.

3.1 Quantitative Comparison

Figures 5 and 10 present the median performance of tree-based and graph-based search algorithms, respectively, computed over five maps for each grid size. Across all evaluated environments, graph-based algorithms consistently outperform their tree-based counterparts, achieving higher success rates alongside reduced runtime and memory consumption. This performance gap becomes increasingly pronounced as grid size grows, reflecting the effectiveness of explored-state pruning in graph search when operating in larger and more complex state spaces. Depth-First Search (DFS) is excluded from optimisation-focused comparisons, as it does not guarantee optimality or completeness in grid-based path-planning problems and exhibits traversal-order-dependent behaviour that leads to highly variable runtime and path cost. Consequently, DFS is treated as a diagnostic baseline rather than a competitive solution for optimal path planning.This indicates that scalability benefits arise primarily from state-space pruning rather than heuristic guidance alone

3.2 Tree Search Based Evaluation

Best Tree-Based Algorithm Tree-based experiments summarised in Fig. 5, conclude **T-A*** as best-performing tree search algorithm.

Explanation T-A* is the only tree-based method that successfully finds paths in both small and medium environments, achieving a 100% success rate in all tested cases. The heuristic function guides the search toward the goal while preserving optimality, significantly reducing unnecessary exploration compared to uninformed methods.

In contrast, T-BFS only succeeds in 80% of small-grid environments and fails entirely on medium grids due to excessive node expansion. T-DFS performs worst, failing to find a valid path in any tested environment and repeatedly exhausting the expansion limit.

3.3 Graph Search Based Evaluation

Best Graph-Based Algorithm Graph-based experiments summarised in Fig. 10, conclude **G-A*** as best-performing graph search algorithm.

Explanation G-A* achieves a 100% success rate across all grid sizes due to heuristic guidance combined with CLOSED-set pruning that prevents redundant state expansion. The use of heuristic guidance and a CLOSED set prevents redundant exploration and improves scalability.

G-BFS finds optimal paths but uses more memory, while G-DFS produces highly variable path lengths. Overall, G-A* provides the best balance of efficiency and path quality.

3.4 Cross Tree-Based and Graph-Based Algorithm Comparison

This subsection compares tree-based and graph-based implementations of the same search algorithms using the dedicated graph-versus-tree experimental results.

3.4.1 BFS

The BFS experiments show that G-BFS consistently outperforms T-BFS. While both use breadth-first expansion, T-BFS repeatedly revisits states and fails on medium grids. G-BFS avoids this redundancy and finds optimal paths with lower runtime and memory usage.

3.4.2 DFS

The DFS results highlight the limitations of tree-based DFS. T-DFS fails in all cases due to repeated state revisiting, while G-DFS successfully finds paths by preventing cycles. Although path quality remains traversal-order-dependent, the graph-based version is more reliable.

3.4.3 A* Search

Both T-A* and G-A* find optimal paths on small and medium grids. However, T-A* suffers from higher runtime and memory usage due to repeated state expansion. G-A* avoids this through a CLOSED set, resulting in superior scalability.

3.5 Best Overall Algorithm

Compared to uninformed methods such as BFS and DFS, G-A* benefits from heuristic guidance that directs exploration toward the goal. Compared to T-A*, it avoids re-expansion of previously visited states by maintaining OPEN and CLOSED sets. These properties make G-A* the most suitable algorithm for grid-based navigation.

4 Discussion

4.1 Choosing Search Algorithm for Real Problems

Choosing a search algorithm depends on completeness requirements, optimality constraints, and available computational resources. In environments with cycles, such as grid maps, graph-based search is essential to avoid repeated state exploration, whereas tree-based search is generally unsuitable.

Completeness and optimality are key considerations. DFS is neither complete nor optimal, while BFS and A* are complete. When optimal paths are required, heuristic-based methods such as A* are preferred due to reduced exploration.

Resource constraints also matter: BFS incurs high memory usage, DFS is unreliable, and A* provides a practical balance between efficiency, optimality, and scalability when an admissible heuristic is available.

4.2 Adapting the Best Algorithm to Dynamic Environments

In dynamic environments, path planning becomes a replanning problem. A practical approach is to compute an initial path using A*, assuming a static environment, and treat this path as provisional.

If obstacle motion information is available, the planner can predict which cells along the path may become occupied over time. This introduces a predictive layer on top of the original heuristic. Replanning is triggered when future conflict is detected along the current path, at which point a new path is computed from the robot's current position. This event-driven replanning strategy aligns with incremental planning approaches used in dynamic navigation.

Section 2: Individual Research Contribution

5 Introduction: The Stochastic Control Challenge

The application of Artificial Intelligence (AI) to mechanical ventilation represents one of the most significant open challenges in medical cyber-physical systems. While modern ventilators are sophisticated engineering marvels, their control logic remains largely heuristic. Clinicians manually adjust settings—specifically Tidal Volume (V_t), Positive End-Expiratory Pressure (PEEP), and Fraction of Inspired Oxygen (F_iO_2)—based on static protocols like the ARDSNet tables.

This protocol-driven approach assumes a "average" patient physiology. However, Acute Respiratory Distress Syndrome (ARDS) is defined by its heterogeneity; a ventilator setting that recruits collapsed lung tissue in one patient may cause fatal over-distension (volutrauma) in another (Gholami et al. 2018). Consequently, the clinical imperative is to transition from static protocols to adaptive, closed-loop control. This report evaluates two competing algorithmic paradigms for achieving this autonomy: Non-linear Model Predictive Control (NMPC), which relies on physiological laws, and Constrained Offline Reinforcement Learning (CRL), which relies on data patterns. We posit that while Deep RL offers superior long-term optimization capabilities, it is unsafe for clinical deployment without the deterministic bounds provided by control theory. The emergent state-of-the-art is therefore not a choice between these methods, but a hybrid architecture: the use of model-based "shields" to gate the decisions of learning agents.

5.1 Formal Problem Definition: The POMDP

To rigorously evaluate these algorithms, we must first formalize the environment. The patient-ventilator system is mathematically best described as a Partially Observable Markov Decision Process (POMDP). It is defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, \Omega, O, \gamma \rangle$.

- **The Latent State ($\mathcal{S}$):** The true condition of the lung is opaque. Variables such as alveolar recruitability, regional compliance, and the degree of inflammatory cytokine release cannot be measured directly by the ventilator. They must be inferred.
- **The Observation Space (Ω):** The agent only sees noisy sensor data: airway pressure waveforms, flow rates, and intermittent blood gas values (PaO_2, $PaCO_2$).
- **The Transition Dynamics (T):** The evolution of the patient's state, $P(s_{t+1}|s_t, a_t)$, is stochastic, non-linear, and time-varying.
- **The Reward Function (R):** This is the critical design choice. The agent must balance two conflicting objectives: maximizing gas exchange (short-term survival) while minimizing mechanical power (long-term organ protection) (Rees et al. 2023).

Understanding this formalism is crucial because the failure modes of the algorithms discussed below often stem from ignoring the partial observability or the stochastic nature of these dynamics.

6 Approach A: Non-linear Model Predictive Control (NMPC)

Non-linear Model Predictive Control (NMPC) represents the "deductive" approach to AI. Instead of learning from data, it reasons from first principles. It relies on a Digital Twin—a computational model of the patient's respiratory mechanics—to predict the future.

6.1 Algorithmic Mechanism: Receding Horizon Control

NMPC treats ventilation as a trajectory optimization problem. At every control interval t, the system estimates the current state x_t (e.g., lung compliance, resistance) using an observer like an Extended Kalman Filter (Karbing et al. 2020). It then solves a constrained optimization problem over a finite prediction horizon N_p:

$$\min_{\mathbf{u}} J = \sum_{k=0}^{N_p-1} \|y_{t+k} - y_{ref}\|_Q^2 + \|\Delta u_{t+k}\|_R^2 \tag{1}$$

Subject to physiological constraints:

$$P_{plat} \leq 30 \text{ cmH}_2\text{O} \tag{2}$$

$$V_t \in [1, 2] \text{ mL/kg} \tag{3}$$

The optimizer searches for the sequence of ventilator settings $\mathbf{u}^*$ that minimizes the cost function J (typically the deviation from a target oxygenation level). Crucially, only the first action of this sequence is applied to the patient. At the next time step $t + 1$, the horizon shifts forward, and the optimization is repeated with fresh sensor data.

6.2 Critical Analysis: The Limits of the Digital Twin

The theoretical strength of NMPC is its interpretability and safety guarantees. Because constraints are explicitly encoded in the solver, the system is mathematically forbidden from suggesting a dangerous action (e.g., a pressure that would burst the lung), provided a feasible solution exists (Waring et al. 2024).

However, the recent DeVENT trial (Patel et al. 2024) exposed the practical limitations of this approach. This randomized controlled trial evaluated the Beacon Caresystem, a physiological model-based decision support system, in 95 patients with ARDS.

- **The Result:** The study failed to show a significant difference in its primary outcome (reduction in driving pressure). While secondary outcomes like the oxygenation index showed improvement ($p = 0.037$), the "Digital Twin" did not revolutionize care as hypothesized (Patel et al. 2024).
- **The Cause:** The failure likely stems from Model Mismatch. NMPC assumes the underlying physics model (often a single-compartment Equation of Motion) perfectly captures the patient's dynamics (Karbing et al. 2020). In severe ARDS, the lung behaves non-linearly due to phenomena like stress relaxation and tidal recruitment. When the patient's physiology deviates from the simplified model, the NMPC controller's predictions degrade, leading to suboptimal or conservative advice (Zhou et al. 2021).

Furthermore, the trial highlighted a "Human-in-the-Loop" friction. Clinicians followed the AI's advice only 60% of the time (Patel et al. 2024). This lack of adherence suggests that even when the math is correct, the "black box" nature of the optimization (or disagreement with clinical intuition) hampers deployment.

7 Approach B: Constrained Offline Reinforcement Learning (CRL)

If NMPC is deductive (physics-first), Reinforcement Learning (RL) is inductive (data-first). Instead of solving differential equations, RL agents learn a policy $\pi(s)$ by analyzing massive datasets of historical clinical decisions, such as the MIMIC-IV database (Johnson et al. 2023).

7.1 The "Deadly Triad" and the Need for Conservatism

Standard RL learns by trial and error (exploration). In a safety-critical ICU, exploration is ethically inadmissible; an agent cannot randomly depressurize a patient's lungs just to see what happens (Gottesman et al. 2019). Therefore, we must rely on Offline RL, where the agent learns solely from a fixed, static dataset.

This introduces a fundamental theoretical hazard known as the "Deadly Triad": the combination of function approximation (Deep Neural Networks), bootstrapping (using estimated values to update estimates), and off-policy data (Levine et al. 2020). Specifically, standard Q-learning algorithms tend to overestimate the value of actions that are Out-Of-Distribution (OOD). If the neural network has never seen a PEEP of 25 cmH2O in a specific patient state, it might erroneously predict a high reward for it. In a clinical setting, this "hallucination" can be fatal.

7.2 Solution: Conservative Q-Learning (DeepVent)

To mitigate this risk, researchers have developed Constrained or Conservative RL. A leading example is DeepVent, proposed by Kondrup et al. (2023). DeepVent utilizes Conservative Q-Learning (CQL), which modifies the objective function to be pessimistic in the face of uncertainty.

The CQL loss function adds a regularization term:

$$\min_{Q} \alpha \cdot \Big(\mathbb{E}_{s\sim\mathcal{D}, a\sim\mu(a|s)}[Q(s,a)] - \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_\beta(a|s)}[Q(s,a)] \Big) + \frac{1}{2}\,\text{Bellman Error} \tag{4}$$

- **Mechanism:** The term $\mathbb{E}_{a\sim\mu}[Q]$ minimizes the Q-values for actions the current policy wants to take, while the term $-\mathbb{E}_{a\sim\pi_\beta}[Q]$ maximizes the Q-values for actions the doctors actually took in the dataset (Kumar et al. 2020).
- **Result:** This pushes down the value of any action that is not supported by the data. The agent is forced to trust the clinicians' historical behavior unless there is overwhelming evidence that a different action is better.

7.3 Empirical Performance

In retrospective evaluations, DeepVent demonstrated the ability to recommend ventilator settings that remained within safe clinical ranges (99.8% of recommendations satisfied safety con-

straints) while achieving a higher expected return (survival probability) than the physician baseline (Kondrup et al. 2023). Unlike the tabular approaches of earlier systems like VentAI (Peine et al. 2021), DeepVent's use of deep neural networks allowed it to handle the high-dimensional continuous state space of the ICU without succumbing to the curse of dimensionality.

8 Comparative Analysis: Physics vs. Data

The choice between NMPC and CRL is often framed as a choice between Safety and Performance. The following comparison breaks down the trade-offs based on core AI principles.

8.1 Knowledge Representation and Adaptability

- **NMPC (The Specialist):** Encodes explicit domain knowledge. It "knows" that Pressure = Flow $\times$ Resistance. This makes it robust to data scarcity; it doesn't need millions of training examples, only a good model fit for the current patient. However, it is rigid. It cannot learn patterns that are not in its equations, such as the correlation between a patient's inflammatory markers and their response to PEEP (Johnson et al. 2023).
- **CRL (The Generalist):** Encodes implicit data distributions. It can discover non-linear, counter-intuitive strategies that human physiology models might miss. However, it is brittle to Distributional Shift. If a patient presents with a novel combination of symptoms (e.g., COVID-19 pneumonia overlaid with bacterial sepsis) that was not present in the MIMIC-IV training set, the RL agent's policy is undefined and potentially dangerous (Futoma et al. 2020).

8.2 Handling of Constraints

- **NMPC (Hard Constraints):** Offers Deterministic Safety. Constraints are treated as "hard" boundaries. The solver will return "Infeasible" rather than violate a safety limit. This is crucial for regulatory approval (e.g., FDA/MDR) (FDA 2023).
- **CRL (Soft Constraints):** Offers Probabilistic Safety. Constraints are typically handled via reward penalties (e.g., -100 reward for unsafe pressure) or regularization (CQL). While this statistically discourages unsafe behavior, it does not mathematically guarantee it. A "safe" RL agent can still output a catastrophic action if the Q-network makes a prediction error (Alshiekh et al. 2018).

8.3 Summary Comparison Table

Feature	Approach A: NMPC	Approach B: CRL (DeepVent/CQL)
Core Logic	Deductive: Solves physics equations (Equation of Motion) (Karbing et al. 2020).	Inductive: Learns patterns from large datasets (MIMIC-IV) (Johnson et al. 2023).
Optimization Goal	Local: Minimizes immediate error ($N_p \approx 30$ mins).	Global: Maximizes long-term outcome (90-day survival) (Kondrup et al. 2023).
Safety Mechanism	Hard Constraints: "Correct-by-Construction".	Soft Constraints: Penalties and Regularization (CQL) (Kumar et al. 2020).
Failure Mode	Model Mismatch: Fails if patient defies the equations.	OOD Error: Fails if patient differs from training data.
Computational Cost	High: Solves optimization online at every step.	Low: Inference is instantaneous after training.
Clinical Evidence	DeVENT Trial (2024): Safe, but no primary benefit (Patel et al. 2024).	In Silico: Superior theoretical performance (Kondrup et al. 2023).

Table 1: Comparative analysis of NMPC vs CRL.

9 Synthesis: The Emergent Architecture

The critical insight from this research is that neither approach is sufficient on its own. NMPC is too myopic and rigid; RL is too opaque and statistically risky. The future of ICU automation lies in Hybrid Architectures that combine the foresight of RL with the safety of Control Theory.

9.1 Dynamic Model Predictive Shielding (DMPS)

The most promising synthesis is Dynamic Model Predictive Shielding (DMPS), as proposed by Banerjee et al. (2024). In this architecture, the control loop is hierarchical:

1. **The Learner (RL):** A Deep RL agent (like DeepVent) acts as the high-level strategist. It proposes an action a_{RL} aimed at maximizing long-term survival.

2. **The Shield (NMPC):** Before the action is sent to the ventilator, it passes through a "Shield." The shield uses the Digital Twin to simulate the immediate consequences of a_{RL} over a short safety horizon H_s.

    - **IF** the trajectory remains within safe bounds (e.g., no barotrauma), the action is allowed.
    - **IF** a violation is predicted, the Shield overrides the agent and solves a local optimization to find the closest safe action a_{safe} (Banerjee et al. 2024).

This architecture solves the "Deadly Triad" problem practically. The RL agent can be aggressive in its optimization because the Shield guarantees physiological safety. Conversely, the Shield prevents the RL agent from making catastrophic OOD errors.

9.2 Uncertainty Quantification: The Role of Conformal Prediction

To further bridge the gap between AI and the clinician, we must address the "black box" trust issue. ConformalDQN (Li et al. 2023) introduces statistical rigor to RL outputs. Instead of outputting a single Q-value, it outputs a prediction set with a guaranteed coverage probability (e.g., 95%).

If the prediction set for the optimal action is extremely wide (indicating high epistemic uncertainty), the system effectively "knows that it doesn't know." In such cases, the algorithm should disable itself and hand control back to the human clinician (Angelopoulos and Bates 2021). This uncertainty-aware handover is likely the missing link that contributed to the low adherence in the DeVENT trial.

9.3 Bridging the Gap: POPCORN

Finally, addressing the POMDP nature of the problem explicitly is vital. POPCORN (Partially Observed Prediction Constrained Reinforcement Learning) forces the agent to learn a representation that is not only good for maximizing reward but also good for reconstructing the patient's observations (Futoma et al. 2020). This ensures that the agent's internal "belief state" is grounded in physiological reality, preventing it from "reward hacking" based on spurious correlations in the data.

10 Conclusion

The automation of mechanical ventilation is a frontier where theoretical AI principles meet the messy reality of biological systems. This report has demonstrated that while Reinforcement Learning (DeepVent) offers the theoretical capability to outperform human clinicians by optimizing for long-term survival, it lacks the safety guarantees required for ethical deployment. Conversely, Model Predictive Control (Beacon/DeVENT) offers safety and interpretability but struggles with model mismatch and has failed to demonstrate superiority in randomized trials.

Therefore, the path forward is not to choose between "Physics" and "Data," but to integrate them. The proposed Dynamic Model Predictive Shielding (DMPS) architecture represents a robust synthesis: using RL for strategy and NMPC for safety. When augmented with Conformal Prediction to quantify uncertainty, this hybrid approach addresses the key failures of previous generations—providing an agent that is farsighted, physiologically bounded, and humble enough to defer to humans when it is unsure.

References

Alshiekh, M. et al. (2018). "Safe reinforcement learning via shielding". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1.

Angelopoulos, A. N. and Bates, S. (2021). "A gentle introduction to conformal prediction and distribution-free uncertainty quantification". In: *arXiv preprint* arXiv:2107.07511.

Banerjee, A. et al. (2024). "Dynamic model predictive shielding for provably safe reinforcement learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Food and Drug Administration (2023). *Artificial intelligence/machine learning (AI/ML)-based software as a medical device (SaMD) action plan*. URL: `https://www.fda.gov/medical-devices` (accessed Jan. 4, 2024).

Futoma, J. et al. (2020). "POPCORN: Partially observed prediction constrained reinforcement learning". In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1–11.

Gholami, B. et al. (2018). "A machine learning framework for the detection of patient-ventilator asynchronies". In: *IEEE Journal of Biomedical and Health Informatics* 22.2, pp. 685–696.

Gottesman, O. et al. (2019). "Guidelines for reinforcement learning in healthcare". In: *Nature Medicine* 25, pp. 16–18.

Johnson, A. E. W. et al. (2023). "MIMIC-IV, a freely accessible electronic health record dataset". In: *Scientific Data* 10, p. 1.

Karbing, D. S. et al. (2020). "Physiological models of the respiratory system for decision support". In: *Respiratory Care* 65.6.

Kondrup, F. et al. (2023). "DeepVent: End-to-end offline reinforcement learning for mechanical ventilation control". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 4, pp. 5130–5138.

Kumar, A. et al. (2020). "Conservative Q-learning for offline reinforcement learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33, pp. 1179–1191.

Levine, S. et al. (2020). "Offline reinforcement learning: Tutorial, review, and perspectives on open problems". In: *arXiv preprint* arXiv:2005.01643.

Li, X. et al. (2023). "ConformalDQN: Distribution-free uncertainty quantification for safe offline reinforcement learning". In: *arXiv preprint* arXiv:2305.13863.

Patel, B. V. et al. (2024). "A randomized control trial evaluating the advice of a physiological-model/digital twin-based decision support system on mechanical ventilation in patients with acute respiratory distress syndrome". In: *Frontiers in Medicine* 11, p. 1473629. DOI: `10.3389/fmed.2024.1473629`.

Peine, A. et al. (2021). "Development and validation of a reinforcement learning algorithm to dynamically optimize mechanical ventilation in critical care". In: *NPJ Digital Medicine* 4.1, p. 32.

Rees, S. E. et al. (2023). "Physiological modeling for clinical decision support in the ICU". In: *Intensive Care Medicine Experimental*.

Waring, J. et al. (2024). "DeVENT: A randomized controlled trial evaluating the advice of a physiological model-based decision support system". In: *Frontiers in Medicine* 11. (Cited as Patel et al. in primary literature).

Zhou, C. et al. (2021). “Virtual patients for mechanical ventilation in the intensive care unit”. In: *Computer Methods and Programs in Biomedicine* 199, p. 105912.