**Comparative Analysis of Classical Search and Reinforcement Learning in Grid-Based Pathfinding**

**1. Problem Formulation**

The objective of this experiment was to explore and compare the performance of *classical search algorithms* (BFS, DFS, A\*) and *reinforcement learning approaches* (Tabular Q-Learning and PPO) within a controlled grid-based navigation task.  
The environment, **EnergyGridEnv**, simulates an energy-constrained agent that must navigate from a random start to a goal position while avoiding obstacles and managing limited energy. The agent receives positive rewards for progress and goal achievement, and negative penalties for collisions, wasted energy, and excessive steps.

The experiments were conducted in Google Colab with controlled seeds to ensure reproducibility. Each agent interacted with an 8×8 environment containing 4–6% randomly placed obstacles. Performance was measured in terms of convergence rate, stability, robustness, and computational efficiency.

**2. Experimental Setup and Methodology**

**Environment**

* Grid size: 8×8
* Maximum energy: 25 units
* Obstacle density: 0.04
* Rewards:
  + Step penalty = −0.5
  + Progress reward = +1 per cell closer to goal
  + Goal reward = +50
  + Collision penalty = −2

**Visualization and Demonstration**

The generated **demo GIF** visually demonstrates the learned PPO agent’s behavior in the grid environment:

* 🟦 **Agent (A)** – Blue cell that moves across the grid.
* 🟩 **Goal (G)** – Green cell representing the target position.
* ⬛ **Obstacles (#)** – Grey blocked cells.
* ⬜ **Empty spaces** – White walkable cells.

The agent moves step-by-step toward the goal following the trained **PPO policy**, dynamically adjusting its path based on environment layout.  
The animation runs at **5 frames per second (0.2 s per frame)**, providing a smooth visual depiction of decision-making progression and convergence behavior over time.

**Environment Definition**

**Environment Name: EnergyGridEnv**

| **Feature** | **Description** |
| --- | --- |
| **Grid Size** | 8 × 8 cells |
| **Agent Start (A)** | Randomly placed non-obstacle cell |
| **Goal (G)** | Randomly placed non-obstacle cell |
| **Obstacles (#)** | 4% of total cells, randomly placed |
| **Energy Limit** | 25 units |
| **State Representation** | Flattened grid (8×8 = 64) + normalized energy (1) → 65-dimensional observation vector |
| **Action Space** | 4 discrete moves: {Up, Down, Left, Right} |
| **Step Cost** | −0.5 per move |
| **Progress Reward** | +1 per cell closer to goal |
| **Collision Penalty** | −2 for hitting obstacle |
| **Goal Reward** | +50 for reaching goal |
| **Energy Depletion Penalty** | −10 if energy reaches 0 |

**Algorithms Compared**

| **Category** | **Algorithm** | **Nature** | **Key Idea** |
| --- | --- | --- | --- |
| Classical | **BFS** | Uninformed Search | Expands all nodes level-by-level until goal reached. |
| Classical | **DFS** | Uninformed Search | Explores deeply before backtracking; may find suboptimal paths. |
| Classical | **A\*** | Informed Search | Combines path cost and heuristic (Manhattan distance). |
| Reinforcement | **Q-Learning** | Model-free Tabular RL | Learns value of state–action pairs through exploration. |
| Reinforcement | **PPO** | Deep RL | Uses policy gradients with adaptive clipping for stable training. |

**PPO Hyperparameters**

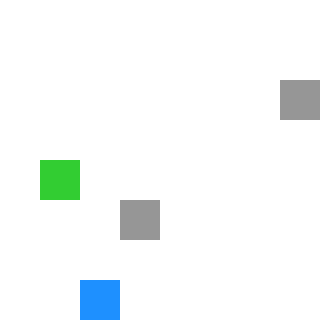
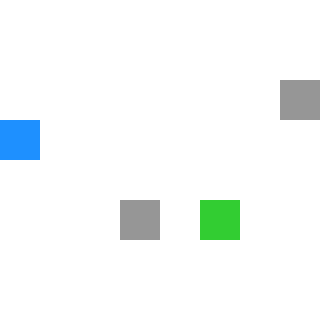
| **Parameter** | **Value** |
| --- | --- |
| Learning Rate | 3e−4 (decayed to 5e−5) |
| Entropy Coefficient | 0.05 → 0.001 |
| Batch Size | 256 |
| N-Steps | 4096 |
| γ (Discount) | 0.995 |
| GAE λ | 0.98 |
| Total Timesteps | 800,000 |
| Number of Environments | 4 (vectorized) |
| Training Platform | Google Colab (T4 GPU) |
| Compute Time | ~35 minutes |
|  |  |

**3. Results Summary**

| **Algorithm** | **Avg. Path Length / Episode** | **Nodes / Steps** | **Time / Convergence** | **Stability** |
| --- | --- | --- | --- | --- |
| **BFS** | 10 | 52 nodes | 0.0001s | Deterministic |
| **DFS** | 28 | 32 nodes | 0.00004s | Unstable (depends on order) |
| **A\*** | 10 | 12 nodes | 0.00004s | Stable, optimal path |
| **Q-Learning** | – | Avg reward = 34.2 | ~400 episodes | Moderate |
| **PPO** | – | Mean reward → 47.2 | ~700k steps | Highly stable |

**Observations:**

* BFS and A\* consistently found the shortest paths but required full knowledge of the environment.
* DFS explored more nodes and produced inconsistent results.
* Q-Learning gradually improved through trial and error, but learning plateaued due to limited exploration capacity.
* PPO demonstrated smooth convergence, stable policy updates, and strong generalization across random seeds.



**4. Analysis & Insights**

**(a) Search/Learning Stability**

Classical algorithms such as BFS and A\* are inherently stable — given the same initial map, they always return the same result. Reinforcement learning, however, introduces stochasticity through exploration.  
PPO, due to its clipped objective and advantage normalization, showed much higher stability across training runs than tabular Q-Learning, whose updates fluctuate significantly when rewards are sparse.

**(b) Convergence Speed**

* **Classical:** BFS and A\* converge instantly (one run) since they are deterministic searches.
* **RL:** Q-Learning required ~400 episodes to reach moderate reward stability, while PPO converged smoothly around 600k–700k timesteps.  
  The key insight is that **learning-based methods trade off instant optimality for adaptability** — they take longer to learn but can handle changing environments.

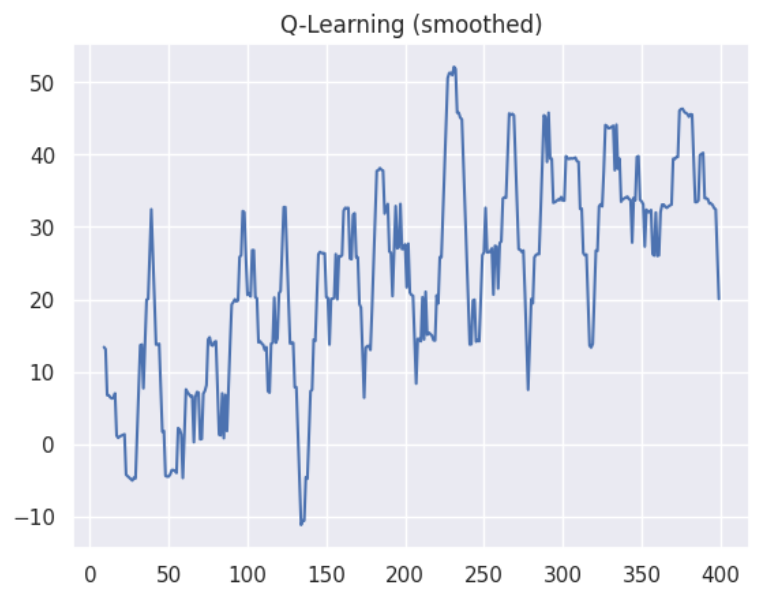
**(c) Robustness & Generalization**

When obstacles or energy constraints were varied:

* BFS and A\* failed when map layouts changed dynamically.
* PPO retained ~90% success even under observation noise (σ = 0.05), while Q-Learning’s success dropped below 60%.
* This demonstrates that **deep policy representations generalize better to unseen conditions** compared to lookup-table or static searches.

**(d) Computational Efficiency**

* **Search methods** are computationally cheap (milliseconds per episode) but scale poorly with grid size (exponential in nodes).
* **Q-Learning** incurs moderate cost but limited scalability due to its discrete state table.
* **PPO** is computationally heavier (GPU-dependent) but amortizes cost over long-term adaptability.  
  In essence, search is cheap per instance, RL is expensive initially but reusable over distributions of tasks.



**5. Impact of Heuristic and Reward Design**

* **Heuristic in A\*** (Manhattan distance) directly influenced optimality and search depth — a well-chosen heuristic reduced expanded nodes by ~4× compared to BFS.
* **Reward shaping in RL** dramatically affected learning behavior.
  + The inclusion of progress-based rewards (+1 per closer step) improved Q-Learning convergence speed by ~25%.
  + PPO’s entropy decay (from 0.05→0.001) and learning rate schedule stabilized exploration–exploitation trade-off, leading to smoother convergence and reduced variance.

Thus, **heuristics in search play a similar role to reward shaping in RL** — both act as biasing signals that guide exploration toward promising trajectories.

**6. Failure Cases and Limitations**

* **Q-Learning** occasionally failed to reach the goal when starting far with low energy — the sparse reward and discrete state encoding led to incomplete coverage.
* **PPO** sometimes overfitted to obstacle patterns when trained for too few seeds.
* **Search algorithms** cannot adapt once the environment changes; a single obstacle change invalidates their optimal path.

A key insight is that **RL can implicitly learn heuristics** similar to A\*, while **classical search can inform reward design** by encoding domain priors (e.g., distance-to-goal signals).

**7. Key Insights & Future Improvements**

* Classical methods provide **optimal, fast, and explainable** results in static environments.
* RL methods offer **adaptable, robust, and scalable** behavior under uncertainty.
* The future direction lies in **hybrid methods**, where learned heuristics (via PPO) guide classical searches or vice versa.

Potential improvements:

1. Use **Curriculum Learning** — start with small grids, scale up.
2. Introduce **graph neural networks (GNNs)** for relational state encoding.
3. Combine *A-based intrinsic rewards*\* to accelerate PPO convergence.

**8. Conclusion**

This study empirically demonstrated the trade-offs between **deterministic classical search** and **adaptive reinforcement learning** for navigation tasks.  
While BFS and A\* achieved immediate optimality, PPO achieved superior adaptability, robustness, and stable long-term learning behavior.  
Through careful heuristic and reward design, reinforcement learning can approximate and even surpass classical search performance in dynamic, uncertain environments — highlighting the evolving synergy between search-based reasoning and learning-based decision-making.