**Part 1 — Literature review & Problem statement**

**1. Evolution of search and learning paradigms — overview**

The study of intelligent agent decision-making has two broad historical tracks that converge today: **classical search/planning** (symbolic, combinatorial methods) and **learning-based control** (statistical, data-driven methods).

**1.1 Classical search strategies (uninformed → informed)**

* **Depth-First Search (DFS)** and **Breadth-First Search (BFS)** are foundational uninformed search algorithms from classical AI (e.g., Russell & Norvig). They explore a discrete state-space graph by expanding nodes in a stack (DFS) or queue (BFS) order. They are *complete* (BFS) or memory-efficient (DFS) in certain contexts, but do not exploit problem structure.
* **A\*** (Hart, Nilsson & Raphael, 1968) introduced *heuristic search*: combining the cost-so-far (g(n)) with an estimate-to-goal (h(n)) to form (f(n)=g(n)+h(n)). When (h) is *admissible* (never overestimates) and consistent, A\* is optimal and efficient compared to blind search, because it focuses expansion toward promising states.

Classical search methods are exact and interpretable, producing explicit plans (paths). Their major limitation is **scalability**: search complexity grows exponentially with branching factor and path length in large discrete state spaces, and they assume access to a perfect transition model (deterministic/known dynamics).

**1.2 Reinforcement learning (value-based, policy-based, actor-critic)**

Reinforcement Learning (RL) treats sequential decision-making under uncertainty as learning to maximize cumulative reward through interaction with the environment (Sutton & Barto).

* **Value-based methods (Q-Learning)** estimate a value function (Q(s,a)) and derive a policy by greedy selection. Tabular Q-learning is simple, model-free, and provably converges (under conditions) for finite MDPs. Its limitations: it does not scale to large/continuous state spaces without function approximation, and exploration/exploitation must be carefully handled.
* **Policy-based methods (Policy Gradient, e.g., REINFORCE)** directly optimize a parametrized policy (\pi\_\theta(a|s)). They can naturally handle continuous action spaces and stochastic policies but can be high variance and require careful variance reduction.
* **Actor-Critic methods** combine policy (actor) and value (critic) learning, reducing variance and improving sample efficiency. Modern stable algorithms include:
  + **PPO (Proximal Policy Optimization)** — a policy-gradient method using a clipped surrogate objective to avoid destructive policy updates. PPO is simple, efficient, and widely used for on-policy continuous & discrete control.
  + **SAC (Soft Actor-Critic)** — an off-policy actor-critic algorithm that optimizes a stochastic policy with a maximum-entropy objective. SAC excels in sample efficiency and stable learning in continuous control.

RL methods handle stochastic, unknown environments and scale better with function approximation (neural nets). However, they are **data-hungry**, sensitive to reward design (sparse vs shaped), and can be unstable without tuning (learning rates, entropy, normalization).

**2. Strengths and limitations of representative algorithms (in the context of grid navigation)**

| **Algorithm** | **Family** | **Strengths (for 8×8 EnergyGrid)** | **Limitations** |
| --- | --- | --- | --- |
| DFS | Uninformed search | Low memory for deep but narrow search; easy to implement | Not complete in infinite graphs; may explore long useless branches; no guarantee of shortest path |
| BFS | Uninformed search | Finds shortest path (in steps) for unweighted grids; simple and deterministic baseline | Rapidly grows memory with branching; explores vast frontier even if goal is far |
| A\* | Heuristic search | If Manhattan (or admissible) heuristic used, focuses search toward goal; finds optimal path in grid with obstacles; fast for small maps | Needs an admissible heuristic; still exponential worst-case; requires full model (deterministic transitions) |
| Q-Learning (tabular) | Value-based RL | Simple, model-free; guaranteed convergence for finite MDP with sufficient exploration; good baseline for discrete grid | Scales poorly with state size; slow learning for sparse rewards (goal only); requires discretized observation (we have discrete grid so possible) |
| PPO | On-policy policy gradient / actor-critic hybrid | Stable policy updates (clipping), works well with neural policies, simple to tune; can learn stochastic policies that generalize | Less sample-efficient than off-policy; requires many timesteps; sensitive to reward shaping and entropy schedule |
| SAC | Off-policy actor-critic (max-entropy) | High sample efficiency, robust in continuous action spaces; yields exploratory policies via entropy term | More complex implementation and hyperparameters; primarily designed for continuous actions (can be adapted to discrete) |
|  |  |  |  |

**Practical implications for this project**

* **Classical search** (BFS/A\*) is ideal as a *baseline* because it produces exact paths and lets us measure search efficiency (nodes expanded, time).
* **Q-Learning** is a natural value-based RL baseline (discrete state/action).
* **PPO** provides a modern policy-gradient benchmark that can operate with function approximators (MLP policy), enabling generalization and handling of shaped rewards.
* **SAC** is optional (bonus) if you want an actor-critic comparison; it tends to be sample-efficient but requires adapting to discrete actions or using continuous relaxations.

**3. Proposed experimental environment — *EnergyGridEnv***

This environment is deliberately simple but expressive: an 8×8 grid, agent has limited energy, randomly placed obstacles, and a randomly sampled goal location. It allows direct comparison between model-based search (A\*/BFS/DFS), tabular RL (Q-learning), and modern RL (PPO). Below is a formal specification you can include in the report.

**3.1 High-level description**

* Grid: (8 \times 8)
* Agent: occupies a single cell; moves in four cardinal directions (up, down, left, right).
* Goal: single target cell (G) randomly sampled at reset.
* Obstacles: fixed set sampled at environment creation (density parameter).
* Energy constraint: agent has max\_energy units; each move costs 1 energy.
* Termination: agent reaches goal (success) OR energy depleted OR step limit exceeded.

**3.2 Formal MDP components**

* **State space (S)**  
  A vector representing the grid flattened plus remaining energy:  
    
  s=[grid\_flattened\_size=64 (cell values: obstacle= -1,empty=0, agent=0.5, goal=1.0),max\_energyenergy​]∈R65  
    
  Alternatively for tabular algorithms use discrete tuple states ((x\_{agent}, y\_{agent}, x\_{goal}, y\_{goal}, energy)).
* **Action space (A)**  
  Discrete, 4 actions:  
  0 = up, 1 = down, 2 = left, 3 = right.
* **Transition dynamics (T(s,a))**  
  Deterministic: move to adjacent cell unless blocked by grid border or obstacle, in which case agent stays put (collision). Each step reduces energy by 1. State updates agent position and energy.
* **Reward design (shaped)**  
  The reward function is designed to encourage progress toward the goal while penalizing wasted moves and collisions. The shape used in your environment (and recommended) is:
  + **Step penalty**: baseline small negative per step, e.g., (r\_{\text{step}} = -0.2).
  + **Progress reward**: positive increment when Manhattan distance to goal decreases:

rprogress​=+α⋅(Δdist)(α>0, e.g., α=2.0)

where .

* + **Goal reward**: large positive reward for reaching the goal, e.g., (r\_{\text{goal}} = +80).
  + **Collision penalty**: negative reward for attempting to move into obstacle, e.g., (-2.0).
  + **Energy depletion penalty**: large negative ending reward for failing (energy ≤ 0 or steps limit), e.g., (-10).

Total reward per step:

r=rstep​+rprogress​+(collision penalty)+(goal or failure bonus/penalty)

**Rationale**: Shaped reward (progress + step penalty) combats sparse reward issues, guides policy search toward the goal, and preserves optimality if shaped conservatively. However, shaping must be tuned: too-large step penalties discourage exploration; too-large progress terms may cause greedy shortsighted behavior.

* *A heuristics (for classical experiments)*\*  
  Use **Manhattan distance** (h(n) = |x\_{n} - x\_{goal}| + |y\_{n} - y\_{goal}|) for admissible heuristic in grid with unit move costs. For experiments, vary heuristic strength (Manhattan, zero heuristic (Dijkstra), and inadmissible heuristics) to test sensitivity.
* **Termination conditions**
  + Success: agent reaches goal cell — terminal with success flag.
  + Failure: energy ≤ 0 or steps > step\_limit (e.g., (4 \times \text{size}^2)) — terminal failure.
  + Max episode length: optional upper bound for learning stability.

**3.3 Experimental protocol and metrics**

To compare algorithms fairly, adopt the following controlled setup:

* **Seed control**: fix random seeds for environment sampling and algorithm initialization to enable reproducibility.
* **Computational budget**: cap RL training runs (≤ 30 minutes per run as required; or cap timesteps e.g., 150k–800k depending on compute).
* **Baselines**:
  + **Search**: BFS, DFS, A\* (Manhattan heuristic) from initial agent position to goal — record path length, nodes expanded, runtime, and path optimality.
  + **Tabular RL**: Q-learning — discretize state as ((x\_a,y\_a,x\_g,y\_g)) (ignore energy for initial runs or include as discrete levels). Log episodic returns, average over last N episodes.
  + **Policy RL**: PPO — MLP policy on flattened obs. Key modifications: observation normalization, reward shaping, entropy annealing, learning rate schedule.
* **Ablations**:
  + Leave out progress reward (sparse reward scenario).
  + Vary heuristic (A\* with Manhattan vs admissible relaxed heuristic).
  + Entropy and LR schedule ablations for PPO.
* **Evaluation metrics**:
  + **Success rate**: fraction of evaluation episodes that reach the goal.
  + **Average return**: mean episodic reward.
  + **Convergence speed**: episodes or timesteps until certain performance threshold.
  + **Stability**: standard deviation across seeds / runs.
  + **Computational cost**: runtime and memory usage.
  + **Search efficiency**: nodes expanded vs path cost for classical methods.

**4. Comparative expectations and insights (what to look for)**

* *Classical search (A)*\* will produce shortest paths and should outperform RL in initial sample efficiency (it finds a path immediately if obstacles known). It cannot generalize across dynamic or stochastic changes unless re-planned.
* **Tabular Q-learning** will eventually learn a mapping from positions to actions but is sample-inefficient for large state spaces and may struggle with sparse goal-only rewards — reward shaping (progress bonus) is crucial.
* **PPO** should learn smoother policies and generalize to noisy observations; with normalization, entropy annealing, and learning-rate schedules it can reach high success rates but requires many interactions. PPO is robust to small observation noise and can be stabilized with the techniques you added (VecNormalize, learning-rate decay, entropy schedule).
* **Trade-offs**: A\* gives exact solution when model known; RL offers adaptability when the model is unknown or non-stationary. Heuristics and reward shaping are the knobs that move algorithm performance — the experiments in this project are designed to quantify those effects.

**5. Short summary**

This project situates itself at the intersection of model-based planning and model-free learning: by implementing classical search methods (DFS, BFS, A\*) and modern RL (Q-learning and PPO) on a single controlled environment (EnergyGridEnv), we can directly compare sample efficiency, robustness, and how heuristic/reward design influence behavior. The experimental design uses an 8×8 energy-limited grid with obstacles, a shaped reward scheme emphasizing progress, and a consistent protocol for evaluation and ablation (seeds, timesteps, logging). This setup both satisfies the assignment requirements and enables the bonus work (novel environment + ablation studies).

**Appendix — concrete parameter suggestions (recommended defaults for experiments)**

* Grid size: size=8 (as required)
* Obstacles: obstacle\_density=0.04 (≈12 obstacles)
* Max energy: max\_energy=25
* Step penalty: -0.2 (tune between -0.05 and -0.5)
* Progress reward α: 2.0
* Goal reward: +80
* Q-learning hyperparams (tabular): alpha=0.6, gamma=0.98, epsilon=0.12
* PPO hyperparams: learning\_rate initial 3e-4 → final 5e-5 (linear), ent\_coef initial 0.02 → 0.005 (anneal), n\_steps=2048–4096, batch\_size=128–256, n\_epochs=10, gamma=0.995, gae\_lambda=0.98.
* Evaluation runs: 10–20 deterministic episodes per checkpoint.