Lecture 8, Jan 19, 2026

Optimal Graph-Based Planning

- We are now interested in finding the optimal path, given nonuniform edge costs
- *Bellman's principle of optimality*: An optimal policy has the property that regardless of the initial states and decisions, the remaining decisions must constitute an optimal policy with regard to the state resulting from the initial decisions, i.e. every point along the path is the start of another optimal path
  - This means that an optimization problem in discrete time can be stated in a recursive, step-by-step form where we start from the final state and work backwards
- Define a cost of a sequence π_K as $L(\pi_K) = \displaystyle\sum_{k=1}^{K} l(x_k, u_k) + l_F(x_F)$ where $l(x_k, u_k)$ is the cost of taking action u_k in state x_k, and $l_F(x_F)$ is the cost of the final state
- Let $G^*(x)$ denote the cost-to-go (i.e. optimal cost of a path to the goal from x), then we can work backwards from the final state and get $G^*(x) = \min\limits_{u} \{\, l(x,u) + G^*(f(x,u)) \,\}$ (*Bellman's equation*) where $f(x,u)$ transitions state x by applying input u
  - Using this we can define a *value iteration* approach
- *Dijkstra's algorithm* prioritizes the queue based on cost-to-come, so that at each step we expand the node with the smallest path-to-come, so that at each step we expand the node with the smallest path-to-come, so that at each step we expand the node with the smallest path-to-come, so that at each step we expand the node with the smallest path-to-come
- A* prioritizes states based on the sum of the cost-to-come and a lower bound heuristic on the cost-to-go (e.g. Manhattan or Euclidean distance)
  - This prevents us from expanding states that would not get us closer to the goal, since these states will have a higher cost-to-come but not a lower heuristic estimate of the cost-to-go
  - Since the heuristic is a lower bound, the algorithm is still optimal
  - Note that when we first find a path, we're not done, but we can skip processing all remaining nodes in the queue with a priority equal or higher than that of the goal since those nodes cannot possibly lead to a better path
- Lifelong planning A* (LPA*) can reuse results from previous A* runs when only a few edge costs have changed
  - All cells that have an incoming edge cost change are checked for inconsistency, by recomputing the cost-to-come based on predecessor's values
  - This progresses outwards until all cells have consistent costs, and the search is restarted using this information
- Focused Dynamic A* (or D*) is an improved version of LPA* that replans while moving towards the goal, making them even more efficient than A^*