ST. FRANCIS XAVIER UNIVERSITY

CSCI-564 CONSTRAINT PROCESSING AND HEURISTIC SEARCH

LECTURE 14 – REAL-TIME SEARCH

Dr. Jean-Alexis Delamer

Recap

- Optimal solution can be found using heuristic search.
- The state space can be very large implying an exponential time and space effort.
  - Some methods exists to deal with these constraints.
  - Linear-space search.
  - State space pruning.
- It is also possible to accept nonoptimal solutions to deal with these constraints.

Recap

- With optimal or suboptimal solutions, the algorithms always find a feasible solution.

- What does this imply?
  - The algorithm needs to terminate.
    - It requires time.
    - Hard to predict how long it will take.

Real-Time Search

- Real time search describes search methods that need a constant search time between action executions.
  - Ex: An algorithm that needs 2 seconds to compute the next action.

- We will consider a more restrictive definition.
  - Real-time search as a variant of agent-centered search.

Real-Time Search

- Definition (Agent-centered search).
  - Restricts the search to the part of the state space around the current state of the agent.
- Example:
  - The current position of a mobile robot.
  - The current board position of a game.
- Why?
  - Interleaving searches and action executions has advantages for intelligent systems (agents) that interact with the world.
  - Decision time can be critical, so we need to guarantee how long it will take.
    - Autonomous cars, airplanes, etc.

Real-Time Search

- The part of the state space around the current state of the agent is immediately relevant for the agent.
  - Contains the state that the agent will be in soon.
    - Ex: Closest obstacles for an autonomous vehicle.
  - Sometimes the only part of the state space known by the agent.

- Agent-centered search does not search all the way from the start state to the goal state.

Real-Time Search

- Agent-centered search:
  - Decides on the local search space.
  - Searches it.
  - Determines which action to executes.
  - Then executes the actions and repeat the process.

- The best-known example of agent-centered search is game playing.
  - Ex: Chess, GO, etc.
  - The states correspond to board positions.
  - Games-playing programs perform minmax search with a limited horizon (lookahead) to decide the action.

Real-Time Search

- Why using a limited lookahead?
  - Limit the search space, so reduce the time to calculate the action.
  - Future moves of the opponent cannot be predicted with certainty.
    - Nondeterministic search tasks.
    - Results in an information limitation that can be solved by enumerating all possible moves.

- Agent-centered search choose a move in reasonable amount of time while focusing on the most relevant part of the search space.

StFX

Real-Time Search

- The previous search algorithm we saw like A*
  - They compute the optimal solutions, the minimal-cost paths.
  - Then follow them.

- These algorithms are called offline algorithm (or offline search).
  - The search is not done while executing the plan.
  - Ex: Navigation systems compute the entire path before you start the trip.

- Agent-centered search methods are online algorithms (online search).
  - Ex: Autonomous cars

Real-Time Search

- The online algorithms are characterized with greedy action-selection steps.
  - It aims to solve suboptimal search tasks.

- They are suboptimal, because they are looking for any path from the start to the goal.
  - They are revolving. They repeat the same process until they reach the goal.

Real-Time Search

- Real-Time search methods have two advantages:
  - Time constraints:
    - Can execute actions with a soft or hard time constraints.
      - Can allow an amount of time to the algorithm. (hard constraint)
      - Or a range. (soft constraint)
    - Since the local search spaces are independent of the sizes of the state spaces.
    - Objective is to minimize the execution cost.
  - Sum of search and execution cost:
    - Execute actions before the complete consequences are known.
      - Likely to incur some overhead in terms of the execution cost (suboptimal).
      - Outweighed by a decrease in the search cost.
      - Because they allow agents to gather information early in nondeterministic state spaces.
        - Update the knowledge regularly.
    - Decrease the sum of the search and execution cost compared to search methods that calculate the entire solution.

Real-Time Search

- Agent-centered search methods must ensure:
  - The search does not cycle without making progress toward a goal state.
    - A potential problem since actions are executed before their consequences are completely known.
  - It remains possible to achieve the goal and it will do so.
  - The goal remains achievable if:
    - No actions exist of which the execution makes it impossible.
    - Avoid their execution in case they do exist.
    - The methods can reset the agent into the start state.

Real-Time Search

- The real-time search methods store a value, called h-value for each state encountered.
- It updates them as the search progress.

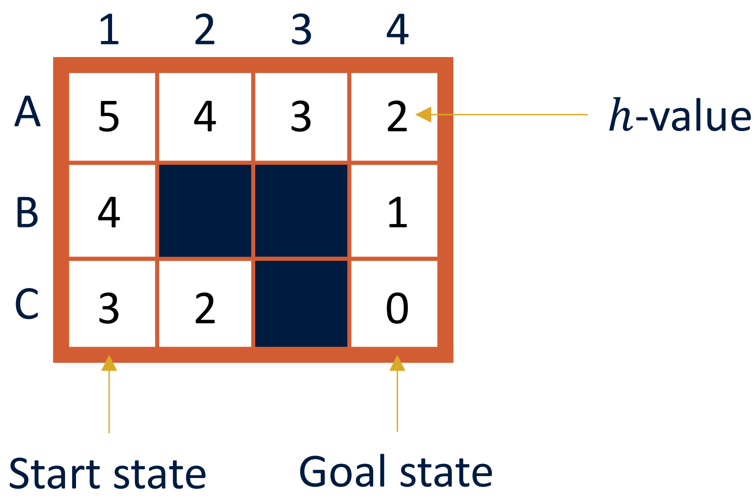
- Why?
  - Focus the search.
  - Avoid cycling.
  - It’s a large part of the search time.

LRTA*

- Learning real-time A* (LRTA*) is the most popular real-time search algorithm.
- The h-values approximates the goal distances of the states.
  - Very similar to A*.
  - Can be initialized with a heuristic.
  - It can be zero if no heuristic are available.

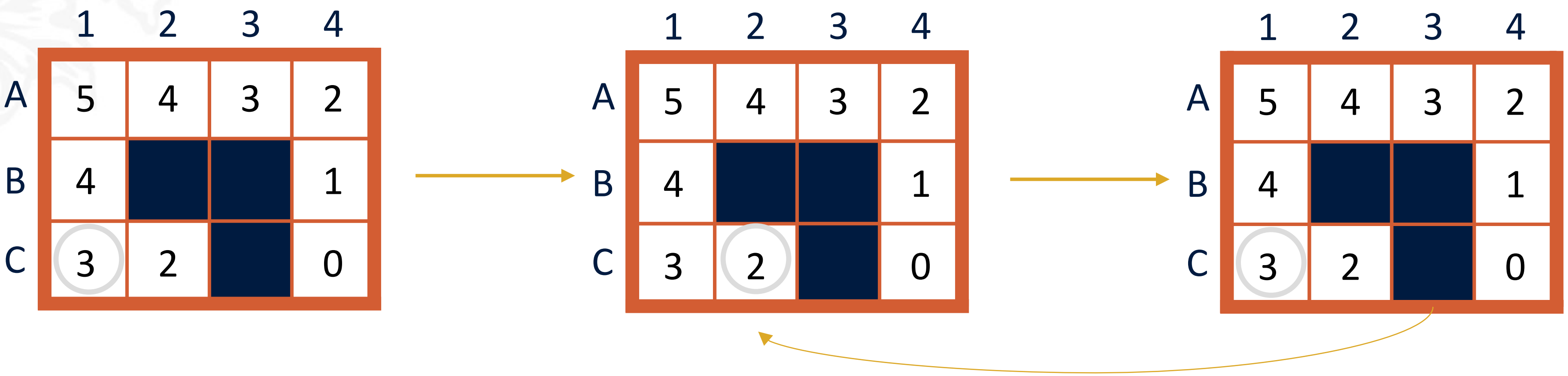
LRTA*

- Example of a mobile robot.
  - All costs are one.
  - Navigates until the goal is reached.
  - h-values are initialized with the Manhattan distances.



LRTA*

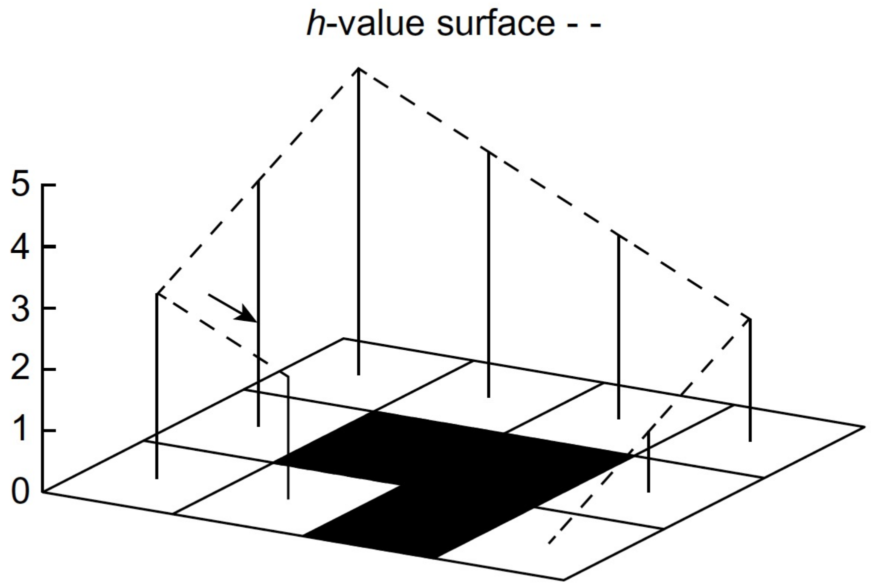
- When starting in C1:
  - It is more advantageous to go in C2.
    - It has a cost of 3.
  - Going in B1 has a cost of 5.
- Choosing the the minimal cost-to-go does not always reach the goal.



LRTA*

- It could cycle forever due to a local minimum of the h-value.
  - Local minimum are known problem in heuristic search.
  - It's not intuitive that the algorithm needs to "climb" first.

- How would you solve this problem?
  - We could randomized the action-selection.



LRTA*

- LRTA* performs a search around the current state of the agent to determine which action to execute.
- It operates as follows:
  - Local Search Space-Generation Step.
    - The local search space can be any set of nongoal states containing the current state.
    - A local search space is minimal iff it contains only the current state.
    - It is maximal iff it contains all nongoal states.
  - Value-Update Step.
    - Assign each state in the local search space its correct goal distance.
    - Assuming that the h-value of the states just outside of the local search space correspond to their correct goal distances.
    - In other words, it assigns each state in the local search space the minimum execution cost for getting from it to a state just outside the local search space plus the remaining execution cost to reach the goal (h-value).

LRTA*

- It operates as follows:
  - Local Search Space-Generation Step.
    - The local search space can be any set of nongoal states containing the current state.
    - A local search space is minimal iff it contains only the current state.
    - It is maximal iff it contains all nongoal states.
  - Value-Update Step.
    - In other words, it assigns each state in the local search space the minimum execution cost for getting from it to a state just outside the local search space plus the remaining execution cost to reach the goal (h-value).
  - Action-Selection Step.
    - Selects the first action that is supposed to minimize the execution cost.
  - Action-Execution Step.
    - Execute the selected action and updates the state of the agent.
    - If the new state is outside of the local search space, it repeat the process.

LRTA*

	1	2	3	4
A	5	4	3	2
B	4			1
C	3	2		0

Minimal local search space

R

	1	2	3	4
A	5	4	3	2
B	4			1
C	3	4		0

L

	1	2	3	4
A	5	4	3	2
B	4			1
C	5	4		0

U

	1	2	3	4
A	5	4	3	2
B	6			1
C	5	4		0

Because it's not the minimal cost path, we launch a new trial, with a restart of the agent.

After this, you will reach the goal in 9 actions.

LRTA*

	1	2	3	4
A	5	4	3	2
B	6			1
C	5	4		0

R

	1	2	3	4
A	5	4	3	2
B	6			1
C	5	6		0

L

	1	2	3	4
A	5	4	3	2
B	6			1
C	7	6		0

U

	1	2	3	4
A	5	4	3	2
B	6			1
C	7	6		0

We restart, but we keep the updated h-values

Because it's not the minimal cost path, we launch a new trial, with a restart of the agent.

After this, you will reach the goal in 9 actions.

LRTA*

	1	2	3	4
A	5	4	3	2
B	6			1
C	7	6		0

We restart, but we keep the updated h-values

R

	1	2	3	4
A	5	4	3	2
B	6			1
C	7	6		0

L

	1	2	3	4
A	5	4	3	2
B	6			1
C	7	6		0

U

	1	2	3	4
A	5	4	3	2
B	6			1
C	7	6		0

In the next trials, the robot follows this minimal-cost path, so we stop.

After this, you will reach the goal in 7 actions.

LRTA*

- Formaly:
  - S denotes the finite set of states, with $s \in S$ the start state and $T \subseteq S$ the set of goal states.
  - $A(u) \neq \emptyset$ denotes the finite set of actions that can be executed in state $u \in S$.
  - $0 < w(u, a) < \infty$ denotes the action cost from the execution of action $a \in A(u)$ in state $u \in S$.
  - $w_{\min} = \min_{u \in S, a \in A(u)} w(u, a)$ denotes the minimal action cost of any action.
  - $Succ(u, a) \subseteq S$ denotes the set of successor states that results from the execution of action $a \in A(u)$, in state $u \in S$.
  - $a(u) \in Succ(u, a)$ denotes the state that results from an actual execution of action $a \in A(u)$, in state $u \in S$.
  - In deterministic state spaces, $a(u) = Succ(u, a)$. After an action a in state u, there is only one successor.

LRTA*

Procedure LRTA*
Input: Search task with initial h-values
Side effect: Updated h-values

$u \leftarrow s$;; Start in start state
while $(u \notin T)$;; While goal not achieved
  Generate S_{lss} with $u \in S_{lss}$ and $S_{lss} \cap T = \emptyset$;; Generate local search space
  Value-Update-Step(h, S_{lss}) ;; Update h-values, see Algorithm 11.2
  repeat ;; Repeat
    $a \leftarrow \arg\min_{a \in A(u)} \{w(u,a) + h(\mathit{Succ}(u,a))\}$;; Select action
    $u \leftarrow a(u)$;; Execute action
  until $(u \notin S_{lss})$;; Until local search space exited (optional)

Procedure Value-Update-Step
Input: Search task with h-values and local search space
Side effect: Updated h-values

for each $u \in S_{lss}$;; For each state in local search space
  $\text{temp}(u) \leftarrow h(u)$;; Backup h-value
  $h(u) \leftarrow \infty$;; Initialize h-value
while $(|\{u \in S_{lss} \mid h(u) = \infty\}| \neq 0)$;; While infinite h-values exist
  $v \leftarrow \arg\min_{u \in S_{lss} | h(u) = \infty}$
    $\max\{\mathit{temp}(u), \min_{a \in A(u)}\{w(u,a) + h(\mathit{Succ}(u,a))\}\}$;; Determine state
  $h(v) \leftarrow \max\{\mathit{temp}(v), \min_{a \in A(v)}\{w(v,a) + h(\mathit{Succ}(v,a))\}\}$;; Update h-value
  if $(h(v) = \infty)$ **return** ;; No improvement possible