St. Francis Xavier University

CSCI-564
CONSTRAINT PROCESSING AND HEURISTIC SEARCH

LECTURE 15- REAL-TIME SEARCH

Dr. Jean-Alexis Delamer

Recap

- Real-Time Search methods are very interesting for online application.
- LRTA* is the most used algorithm.
  - Explore a local search space.
    - The local search space can be minimal.
    - Or maximal.
  - Update the h-values after each trial.

Analyze of LRTA*

- Lemma.
  - For all times $t = 0,1,2,\ldots$ (until termination): Consider the $(t+1)^{st}$ value-update step of LRTA*. Let S^t_{lss} refer to its local search space. Let $h^t(u) \in [0,\infty]$ and $h^{t+1}(u) \in [0,\infty]$ refer to the h-values immediately before and after, respectively, the value-update step. Then, for all states $u \in S$, the value-update step terminates with

$$h^{t+1}(u) = \begin{cases} h^t(u), & \text{if } s \notin S^t_{lss} \\ \max\{h^t(u), \min\limits_{a \in A(u)} \{w(u,a) + h^{t+1}(Succ(u,a))\}\}, & \text{otherwise} \end{cases}$$

We update only the state in the local search space.

Analyze of LRTA*

- A disadvantage of LRTA* is that it cannot solve all search tasks.
  - Because it interleaves searches and action executions.
- All search methods can solve problem for which the goal distance of the start state is finite.
- Why interleaving searches and actions execution limits the solvable search tasks?
  - Actions are executed before their consequences are known.
    - Even if the goal distance of the start is finite, LRTA* could accidentally executes actions that lead to a state with infinite goal distance.

Analyze of LRTA*

- LRTA* is guaranteed to solve all search tasks in safely explorable state spaces.
- State spaces are safely explorable iff the goal distances of all states are finite.
  - The depth of the search tree is finite.
- For safely explorable state spaces where all action costs are one, $d \leq n - 1$.
  - All states that cannot be reached from the start state or can be reached but through a goal state can be deleted.

Analyze of LRTA*

- Safe explorable state spaces guarantee that LRTA* can reach a goal state no matter which actions it has executed in the past.
- Do you have an example of a safely explorable state space?
  - Strongly connected state spaces.
    - Every state can be reached from every other state.
- When the state space is not safely explorable.
  - LRTA* will end up in a goal state.
  - Or reach a state with goal distance infinity and then executes actions forever.
- How would you modify LRTA* to solve this problem?
  - Get information from the local search space to detect that the goal is not reachable anymore.
  - Complicated and not done in the literature.

Analyze of LRTA*

- We will assume that the state spaces are safely explorable.

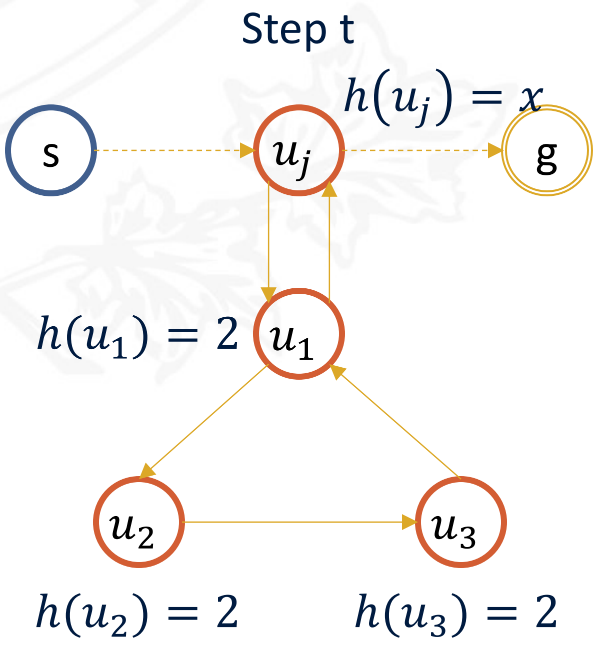
- Theorem: LRTA* always reaches a goal state with a finite execution cost in all safely explorable spaces.

Analyze of LRTA*

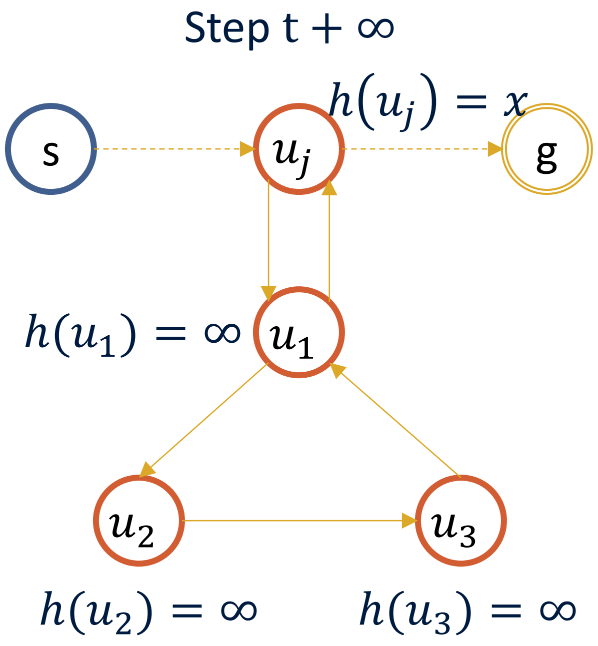
- The idea behind the proof:
  - If LRTA* did not reach a goal state, then it would cycle forever.
  - Since the state space is safely explorable, there must be some way out of the cycle.
  - We want to show that LRTA* will eventually executes an action that takes it out of the cycle.

Analyze of LRTA*

- How to show that LRTA* will execute an action that leave the the cycle?



The heuristic will grow after each trial.
At one point the h-value of the states in the cycle will be superior to the h-value of a state outside the cycle.



Analyze LRTA*

- We will discuss the performance of LRTA*.
  - The performance is measured by its execution cost.
- The complexity of LRTA* is its worst-case execution cost.
  - Remember that we want to know how it scales as the state spaces get larger.
  - We measure the size of the state space as $x = nd$, the product of the number of states and the depth.
- Quick recap:
  - $O(x)$ is the upper complexity bound.
  - $\Omega(x)$ is the lower complexity bound.
  - $\Theta(x)$ is the tight complexity bound.

Analyze LRTA*

- Calculate the upper bound on the Execution Cost of LRTA*.

- First, we calculate the upper bound of the execution cost LRTA* at time t.
- Lemma.
  - For all times $t = 0,1,2, \ldots$ it holds that the execution cost of LRTA* with admissible initial h-values h^0 at time t is at most $\sum_{u \in S}[h^t(u) - h^0(u)] - (h^t(u^t) - h^0(u^0))$.

Analyze LRTA*

- Proof by induction:
  - Done in class.

- We use this lemma to derive the upper bound on the execution cost.
- Theorem (Completeness of LRTA*):
  - LRTA* with admissible initial h-values h^0 reaches a goal state with an execution cost of at most $h^0(s) + \sum_{u \in S}[\delta(u, T) - h^0(u)]$.
- Proof:
  - $\sum_{u \in S}[h^t(u) - h^0(u)] - \left(h^t(u^t) - h^0(u^0)\right) \leq \sum_{u \in S}[\delta(u, T) - h^0(u)] + h^0(u^0)$
  - $= h^0(s) + \sum_{u \in S}[\delta(u, T) - h^0(u)]$

Analyze LRTA*

- Since the goal distances are finite in safely explorable state spaces and the minimal action cost $w_{\min}$ is strictly positive.
- The previous theorem shows that LRTA* reaches a goal state with an execution cost of at most $\sum_{u \in S} \delta(u, T)$.
  - Thus, after at most $\frac{\sum_{u \in S} \delta(u,T)}{w_{min}}$ actions.
- One consequence is that search tasks where all states are clustered around the goal are easier to solve with LRTA*.
  - Why?

Analyze LRTA*

- Example of $(n^2 - 1)$-puzzle:
  - It is a problem considered hard, because it has a small goal density.
  - The 8-puzzle has 181 440 states, but one goal.
    - You could think that you need to explore lot of states before finding the goal.
  - But the average goal distance is only 21.5!
    - And it’s maximal distance 30.
  - Why?
    - The tiles forms a ring around the center.
    - The tiles are never moved far away from the goal by LRTA*.
    - Even if a mistakes is made.
  - So LRTA* is perfect for this problem.

1	2	3
8		4
7	6	5

Features of LRTA*

- There are some features of LRTA* we didn't speak about.
- Heuristic knowledge:
  - LRTA* uses heuristics to guide the search.
  - The larger the initial h-values, the smaller upper bound on its execution cost.
    - By larger, we mean more informed (closest to the real distance).
  - LRTA* is fully informed iff the initial h-values equals the goal distances.
  - Its execution cost is worst-case optimal.
    - No other search methods can do better in the worst-case.

Features of LRTA*

- Fine-grained control:
  - You can choose how much search to perform between actions by varying the sizes of the local search spaces.
    - Large local search space performs a complete search, like A*.
    - It slows the search but provides the minimal-cost paths and minimize the executions.
    - Minimal local search spaces perform almost no searches.
  - For time constraints problems, LRTA* can be used as an anytime algorithm.
    - Algorithms that can solve a search tasks with any bound on their search cost.
    - You can stop the search anytime and have an action to execute.
    - The quality depends on the time allowed.
    - Particularly useful in robotics and adversarial games.

Features of LRTA*

- Fine-grained control:
  - In this context we can distinguish two types of agents.
  - Fast-Acting agents:
    - A smaller amount of search between actions.
    - Agents for which the execution speed is fast compared to their search speed.
    - Examples of the sliding puzzles. The action are only an update of values in memory.
  - Slow-Acting agents:
    - A larger amount of search between actions.
    - Agents for which the search speed is fast compared to their execution speed.
    - Robots are examples of slow-acting agents. It takes time to move, so you can search longer.
      - Exceptions of critical systems!

Features of LRTA*

- Improvement of execution cost:
  - If the heuristic is not completely informed the execution cost is not minimal.
  - Assuming LRTA* solves a series of search tasks in the same state space with the same sets of goal.
    - If the initial h-values are admissible for the first search task.
    - They are also admissible for the first search task after the updates.
    - Then, they are admissible for the other search tasks.
  - The start states can be different while keeping the h-values.
    - Because the admissibility does not depend on the start state.
  - You can reuse this knowledge and improve the execution cost.
    - After some time, you could obtain a fully informed heuristic.

StFX

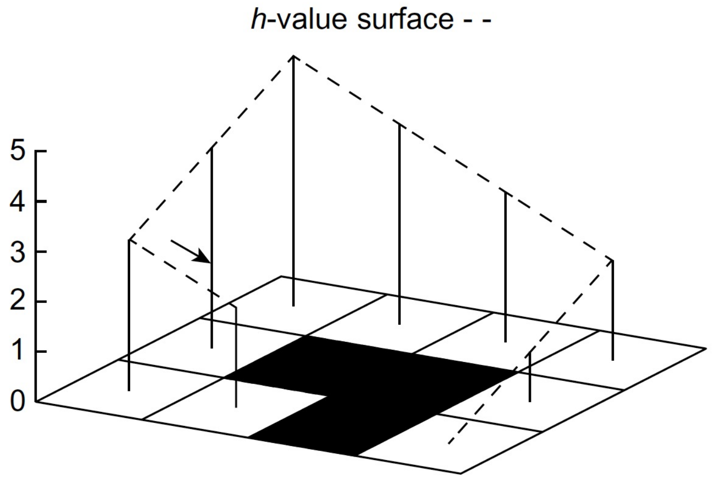
Features of LRTA*

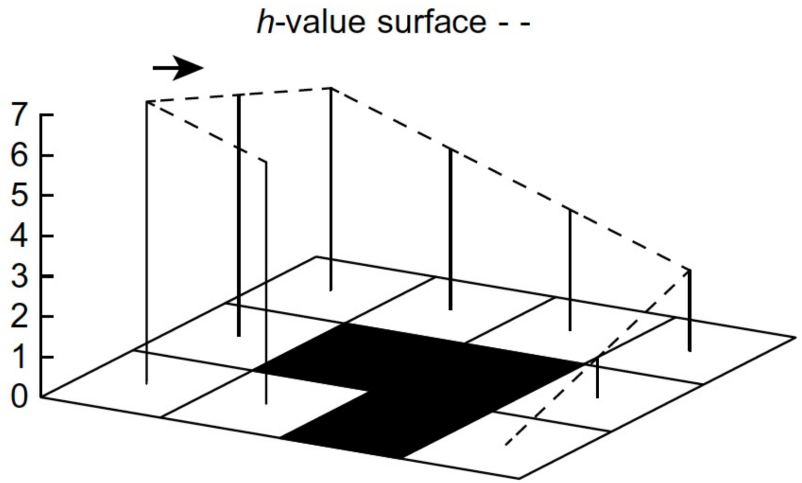
- Improvement of execution cost:
  - Theorem (Convergence of LRTA*):
    - Assume that LRTA* maintains h-values across a series of search tasks in the same safely explorable state space with the same set of goal states.
    - Then, the number of search tasks for which LRTA* with admissible initial h-values reaches a goal state with an execution cost of more than $\delta(s, T)$ is bounded from above.
  - Proof (informal):
    - Assume that LRTA* solves the same search task repeatedly from the same start state.
    - The h-values no longer change after a finite number of searches.
    - LRTA* follows the same minimal-cost path from the start to a goal during all future searches.

Features of LRTA*

- Improvement of execution cost:

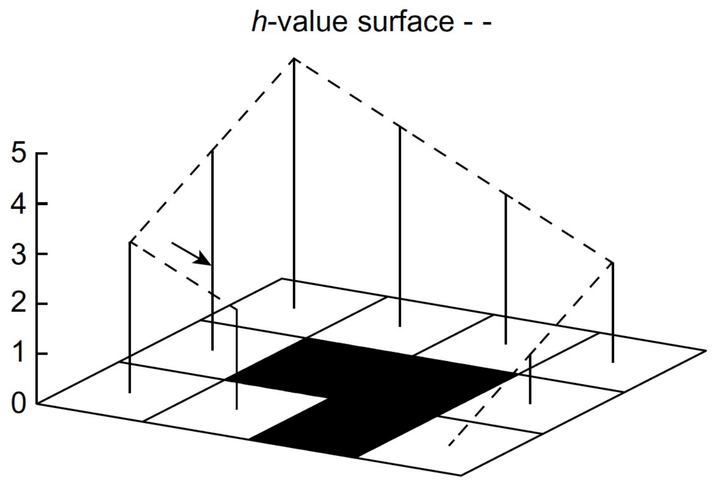
	1	2	3	4
A	5	4	3	2
B	4			1
C	3	2		0





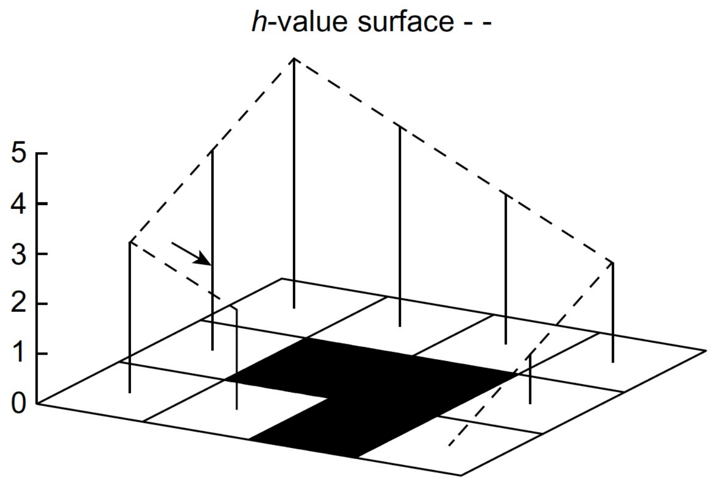
Variants of LRTA*

- Some variants of LRTA* have been proposed.
- Variants with local search spaces of varying sizes:
  - With small local search spaces, you need to executes a lot of actions before escaping depressions (valleys).
  - You can increase the size of the local search spaces to find a path outside the valley.



Variants of LRTA*

- How can you detect valleys?
  - If the current h-value is smaller than the cost-to-go of every actions.
- When you detect a depression:
  - You start increasing the local search space.
  - You stop when all the states part of the valley are inside it.
  - States stop to be included when an action exists with a cost-to-go inferior to the h-value.



Variants of LRTA*

- Variants with minimal lookahead.
  - LRTA* needs to predict the successor states of actions.
  - We can decrease its lookahead further.
    - We associate the values with state-action pairs rather than states.
    - We call it q-value $q(u, a)$.

Procedure Min-LRTA*
Input: Search task with initial q-values
Side Effect: Updated q-values

$u \leftarrow s$;; Start in start state
while $(u \notin T)$;; While goal not achieved
  $a \leftarrow \arg\min_{a \in A(u)} q(u,a)$;; Select action
  $q(u,a) \leftarrow \max\{q(u,a), w(u,a) + \min_{a' \in A(Succ(u,a))} q(Succ(u,a),a')\}$;; Update q-value
  $u \leftarrow a(u)$;; Execute action

Variants of LRTA*

- Reinforcement learning is based on q-values.
- You try to learn the value of each action in each states.
- It’s very interesting, because it works in model-free problem.
  - Problem where you don’t know the model.