ST. FRANCIS XAVIER UNIVERSITY

CSCI-564 CONSTRAINT PROCESSING AND HEURISTIC SEARCH

LECTURE 18 – SELECTIVE SEARCH

Dr. Jean-Alexis Delamer

StFX

Recap

- The heuristic search strategies seen so far were based on a systematic enumeration of the state space.
- Search heuristics accelerated the exploration.
- We discussed some exceptions in which we sacrificed the optimality for a better time and space performance.

Selective search

- Selective search algorithms.
  - Generic term to cover local search and randomized search.
  - They are satisficing.
    - Do not always return the optimal solution.
    - But very good results in practice.

Selective Search

- Local search:
  - Given the local neighborhood of states.
  - Find a state with optimal global cost.
  - As the algorithm is inherently incomplete, we aim for local optimum.
    - State optimal in the local neighborhood.

Hill Climbing

- Hill climbing is the simplest local search algorithm.
- Very simple idea:
  1. Start from a state s.
  2. Move to a neighbor t with better score.
  3. Repeat.

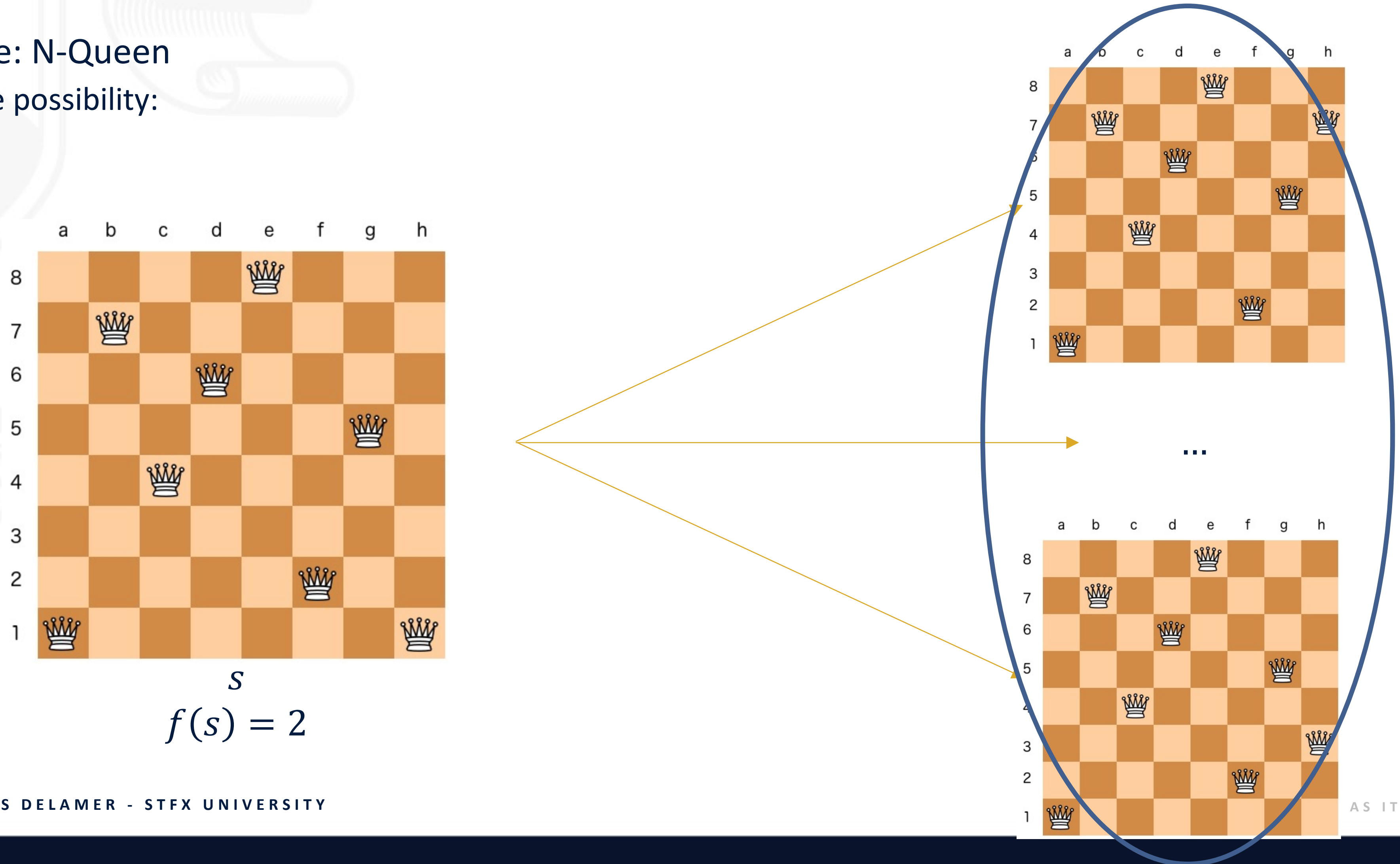
Hill Climbing

- What is a neighbor?
  - You need to define it.
  - The neighborhood of a state is the set of neighbors.
    - Can be called 'move set'
  - Neighborhood is slightly different from successors.
    - The neighbors don't have to be valid states.
    - Or the direct successors of the state.

Hill Climbing

- Example: N-Queen
  - One possibility:

Neighborhood of s



s

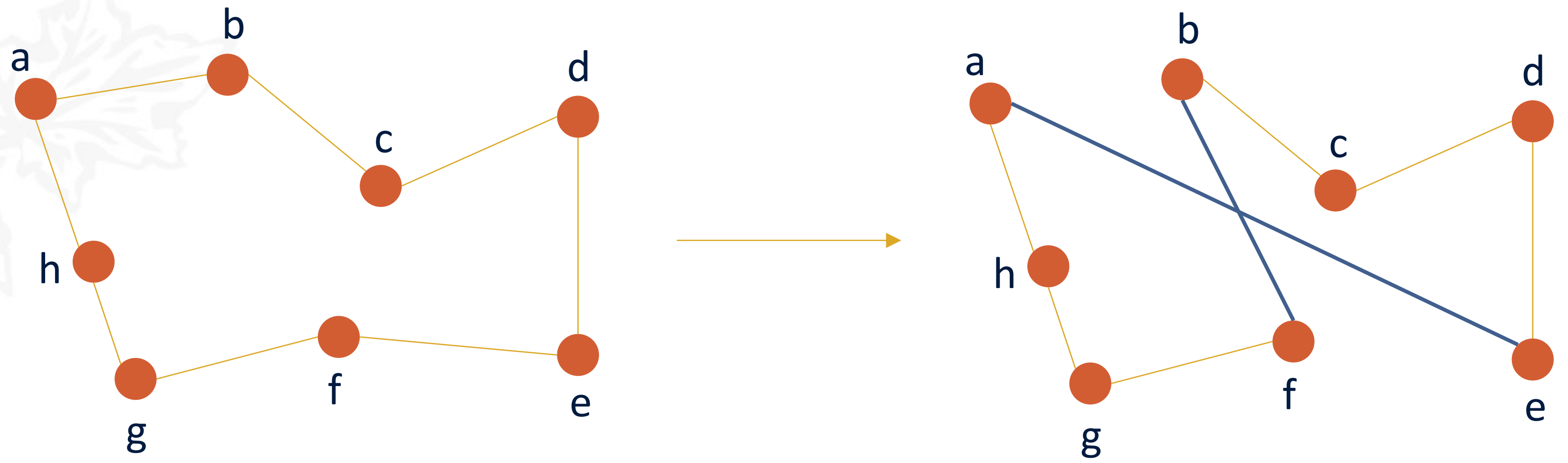
$f(s) = 2$

$f = 2$

$f = 1$

Hill Climbing

- Example: TSP



Path: a, b, c, d, e, f, g, h

Path: a, e, d, c, b, f, g, h

Hill Climbing

- Hill climbing selects the best successor node in the neighborhood.
  - Use an evaluation function denoted f.
- The successor serves as the actual node.
  - We're not searching in a tree anymore.
  - We're "jumping" from nodes to nodes.

- In a maximization problem, we call it hill climbing.
- In a minimization problem, we call it gradient descent.

Hill Climbing

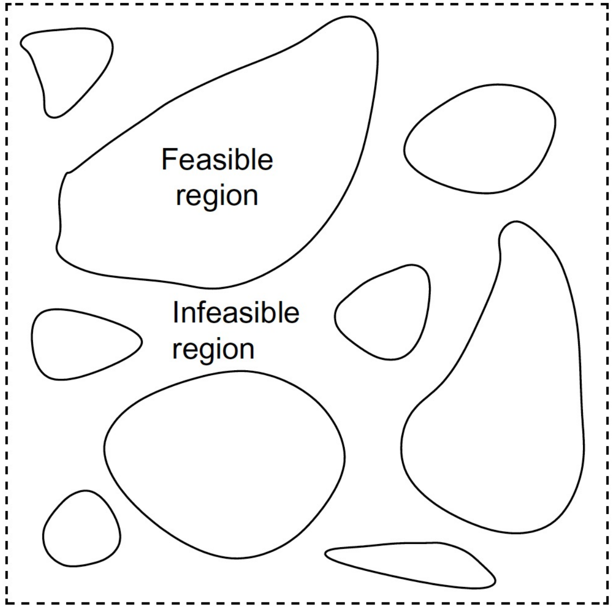
Procedure Hill-Climbing
Input: State space min. problem with initial state s and neighbor relation *Succ*
Output: State with low evaluation

```
u ← v ← s; h ← f(s)                     ;; Initialize search
do                                      ;; Loop until local optimum found
    Succ(u) ← Expand(u)                 ;; Generate successors
    for each v ∈ Succ(u)                ;; Consider successors
        if (f(v) < f(u)) u ← v          ;; Evaluation improved
while (u ≠ v)                           ;; Generate successors
return u                                ;; Output solution
```

Very simple !

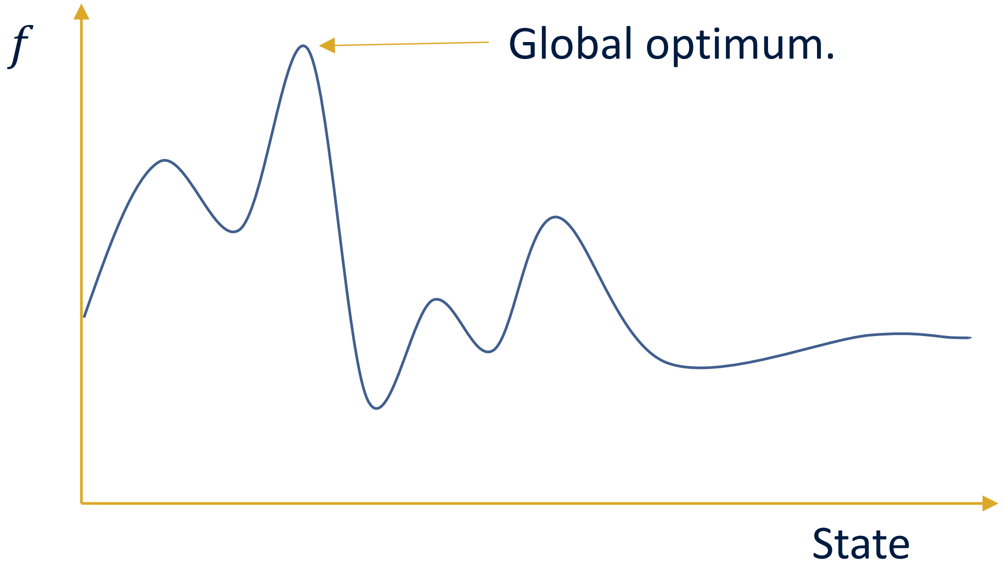
Hill Climbing

- Very simple, but there are some issues.
  - The feasibility problem.
  - The algorithms can be stuck.



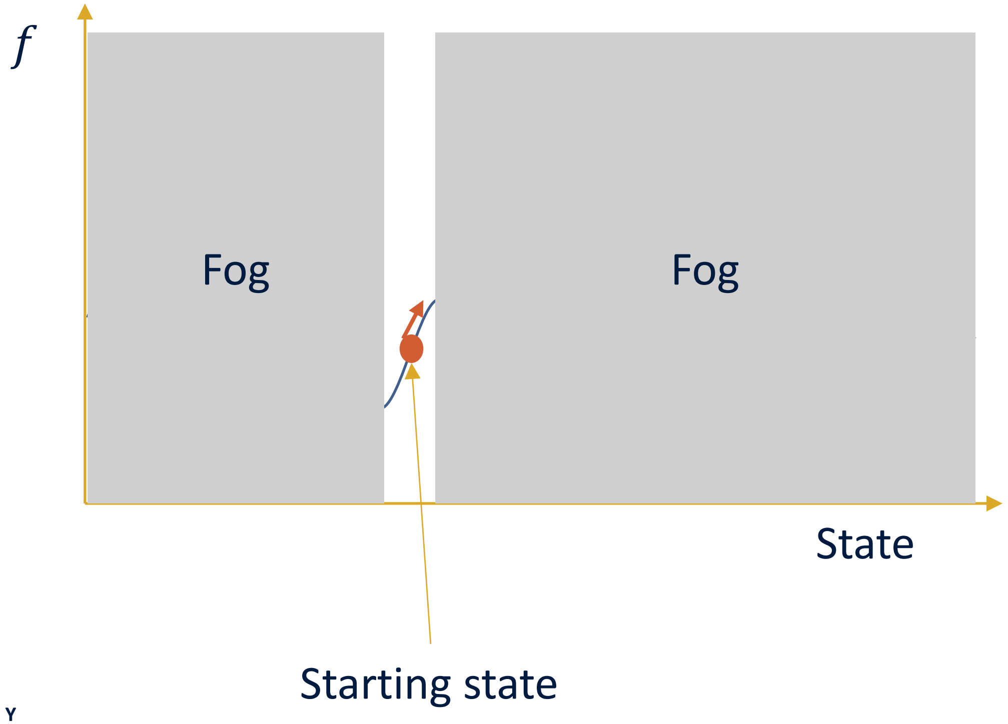
Hill Climbing

- Useful conceptual picture: f surface = 'hills' in state space.



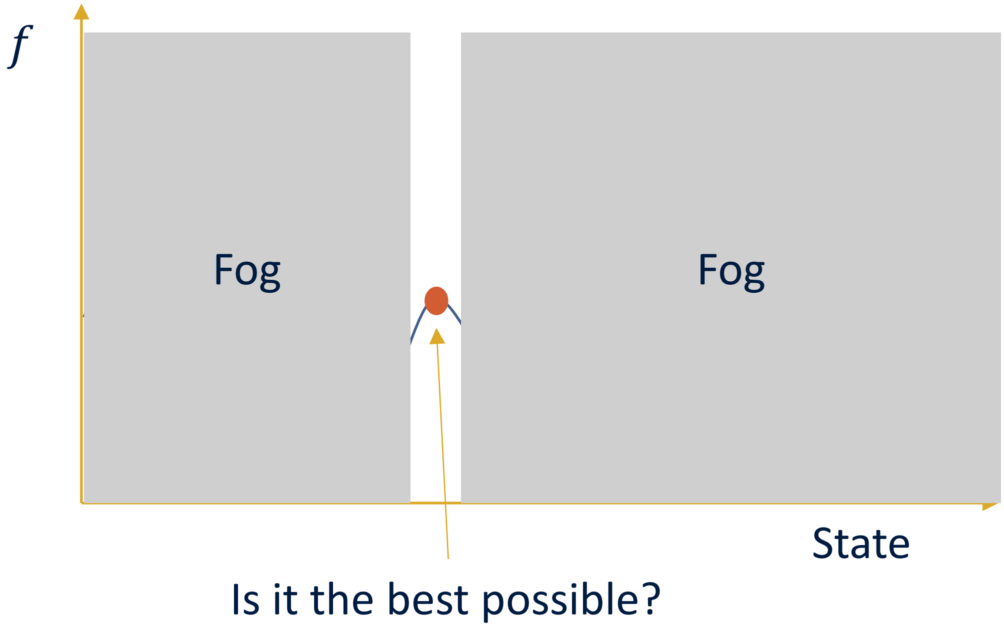
Hill Climbing

- But we can't see the landscape all at once.
  - Only see the neighborhood.
  - Like climbing in fog.



Hill Climbing

- The problem of local optimum.

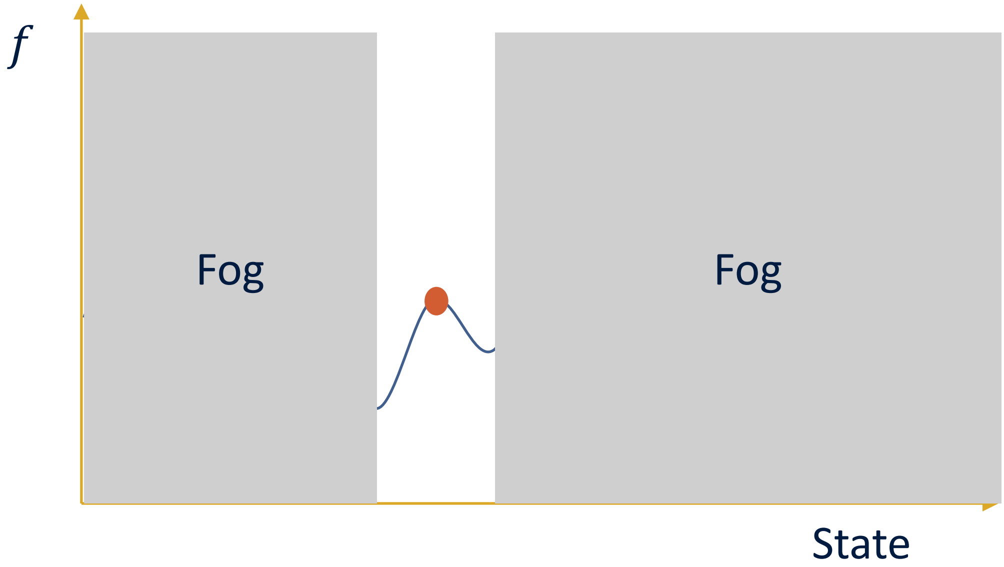


Hill Climbing

- Do you have a solution?

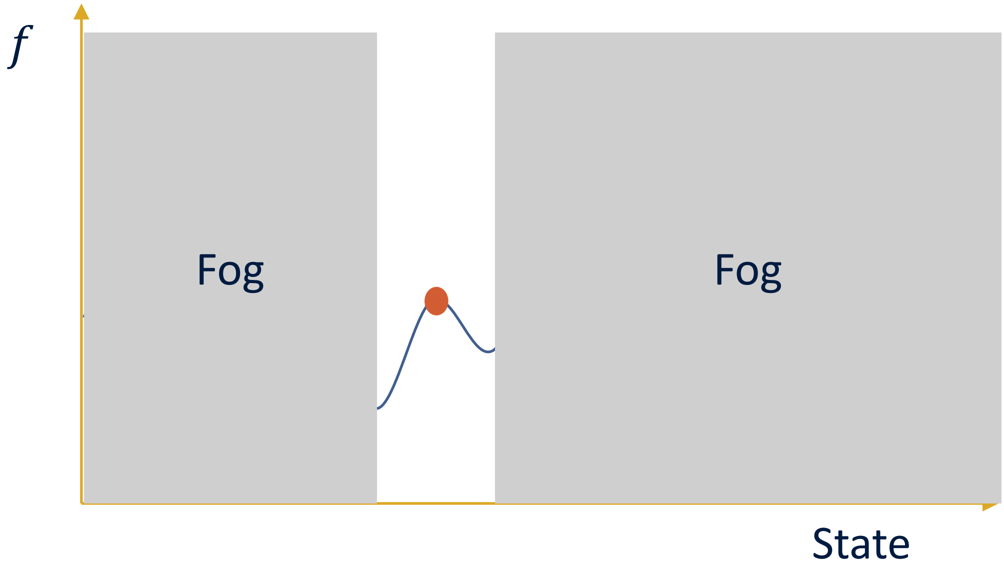
Hill Climbing

- Do you have a solution?
  - We could increase the neighborhood.



Hill Climbing

- Do you have a solution?
  - We could increase the neighborhood.
  - Have an impact on the time complexity.



We're still in a local optimum.

Hill Climbing

- How can we escape local optimum?
  - Repeated hill climbing with random start
    1. When stuck, pick a random new start, run basic hill climbing from there.
    2. Repeat this k times.
    3. Return the best of k local optimum.
    - Can be very effective
    - Should be tried whenever hill climbing is used.

- The algorithm is very greedy, so it is more likely to stay on a local optimum.

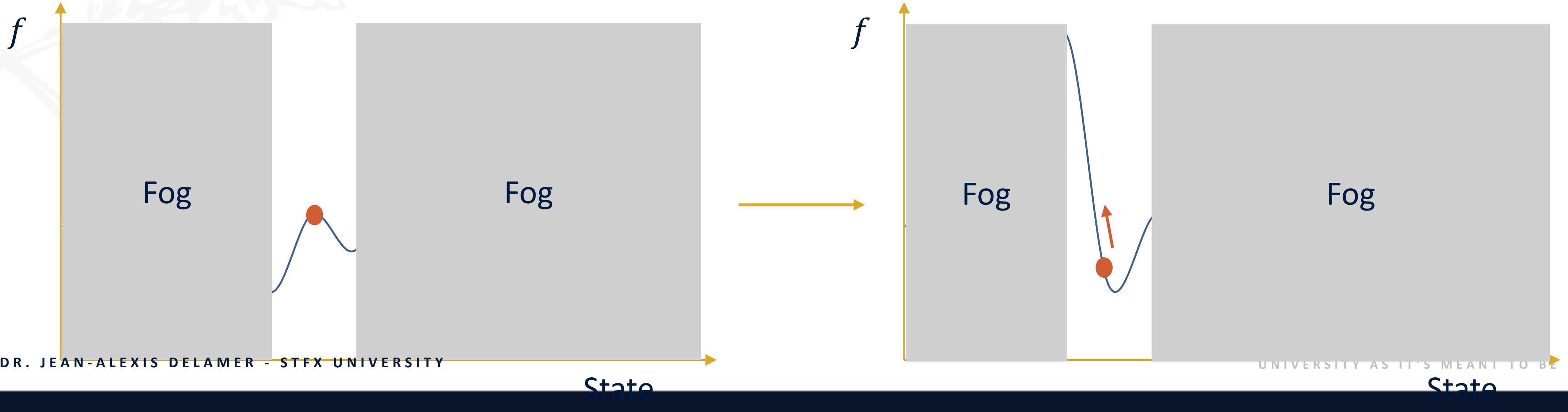
Hill Climbing

- How can we make the algorithm less greedy?
  - Stochastic hill climbing.
    - Randomly select among better neighbors.
    - The probability depends on the quality of the state.
- What if the neighborhood is too large to enumerate?
  - First-choice hill climbing.
    - Randomly generate neighbors, one at a time.
    - If better, take the move.
- What do you think?

Hill Climbing

- The algorithms are still greedy.
  - It chooses only move that improves the solution.

- Sometimes it is important to move to an inferior neighbor in order to escape a local optimum.



Simulated Annealing

- Simulated annealing is a local search approach based on the analogy of the Metropolis algorithm.

- The motivation is adopted from physics.
  - Annealing is subjecting (glass or metal) to a process of heating and slow cooling in order to toughen and reduce brittleness.

Simulated Annealing

- The idea:
  1. Pick initial state s
  2. Randomly pick t in neighbors.
  3. If $f(t)$ is better move to t.
  4. Otherwise move to t with a small probability

- How to choose the small probability?

Simulated Annealing

- According to the laws of thermodynamics.
  - The probability of an increase in energy of magnitude ΔE at temperature T is equal to $e^{-\frac{\Delta E}{kT}}$.
  - Where k is the Boltzmann constant.

- So, we could define the probability as
  - $e^{\frac{f(s)-f(t)}{kT}}$
  - In practice the Boltzmann constant can be safely removed.

Simulated Annealing

- The temperature is a parameter that cools (anneals) over time.
  - High temperature: almost always accept any t.
  - Low temperature: first-choice hill climbing
- If the energy difference $f(s) - f(t)$ is large, the probability is small.

Simulated Annealing

Procedure Simulated Annealing
Input: State space min. problem, initial temperature T
Output: State with low evaluation

$t \leftarrow 0$	;; Iteration counter
$u \leftarrow s$	;; Start search from initial state
while $(T > \epsilon)$	;; T not too close to 0
$Succ(u) \leftarrow Expand(u)$	;; Generate successors
$v \leftarrow Select(Succ(u))$	;; Choose (random) successor
if $(f(v) < f(u))$ $u \leftarrow v$	;; Evaluation improved, select v
else	;; Evaluation worse
$r \leftarrow Select(0,1)$	;; Choose random probability
if $\left(r < e^{\frac{f(u)-f(v)}{T}}\right)$	;; Check Boltzmann condition
$v \leftarrow u$	;; Continue search at v
$t \leftarrow t + 1$	;; Evaluation improved, select v
$T \leftarrow Cooling(T, t)$	;; Decrease T according to iteration count
return u	;; Output solution

Simulated Annealing

- The cooling scheme is important.
- Neighborhood design is the real ingenuity, not the decision to use simulated annealing.
- Not much to say theoretically.
- With infinitely slow cooling rate, finds global optimum with probability 1.
  - Increase the time complexity.
- Proposed by Metropolis in 1953 based on the analogy that alloys manage to find a near global minimum energy state, when annealed slowly.
- Try hill-climbing with random restarts first!