Artificial Intelligence

Local and Randomized/Stochastic Search

Lecture 6

CS 444 – Spring 2019

Dr. Kevin Molloy
Department of Computer Science
James Madison University

JAMES MADISON UNIVERSITY

Outline for Today

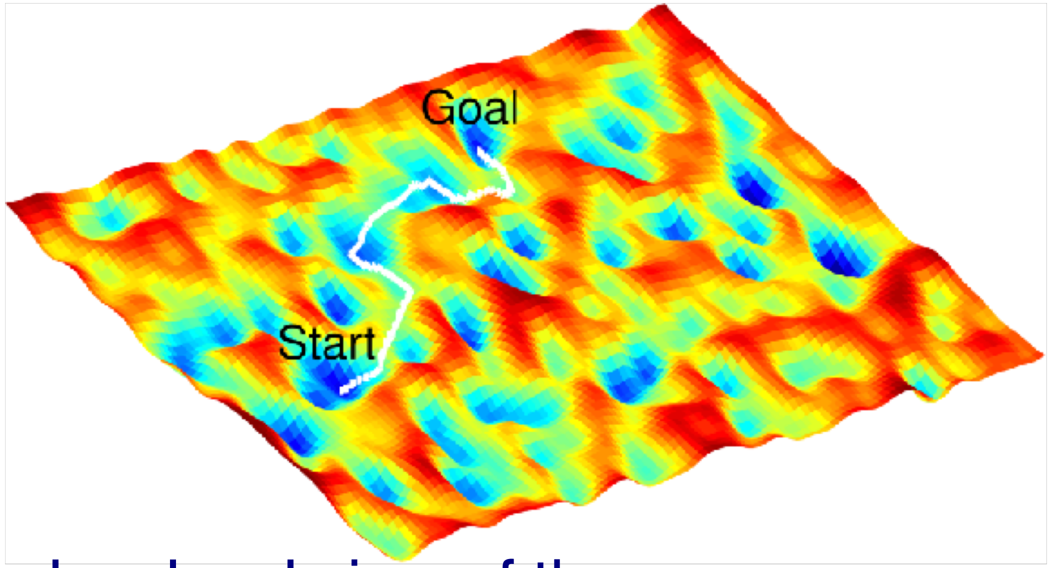
- Search in Unobservable or Large Environments
- Hill Climbing
  - Discrete Spaces
  - Continuous spaces
  - Premature convergence

Randomization in Local Search

- Random-restart/multi-start
- Iterated Local Search
- Memory-based search/optimization
  - Tabu search
  - Tree-guided search
  - Evolutionary Algorithms (EA)
  - Genetic Algorithms (GAs)

Summary of Uninformed Search Algorithms

- Graph search algorithms conduct systematic search
- Assume state space is finite and can fit in memory (not always the case)
- Environment may not even be observable
- No model of the environment available



- Local Search: how to find solutions quickly with only a local view of the space
- Randomized Search: Address premature convergence of local search
- Fundamental to local search: **iterative improvement mechanism**

Iterative Improvement Mechanism in Local Search

In many optimization problems, path is irrelevant; the goal state itself is the solution. Examples?

Traveling salesman (TSP), n-queens, circuit layout (VLSI), factory floor design, protein structure prediction.

Then state space = set of "complete" configurations

Find the **optimal** configuration (explicit constraints or objective/fitness function)

Iterative improvement: keep a single "current" state, try to improve it

that is, no memory of what has been found so far

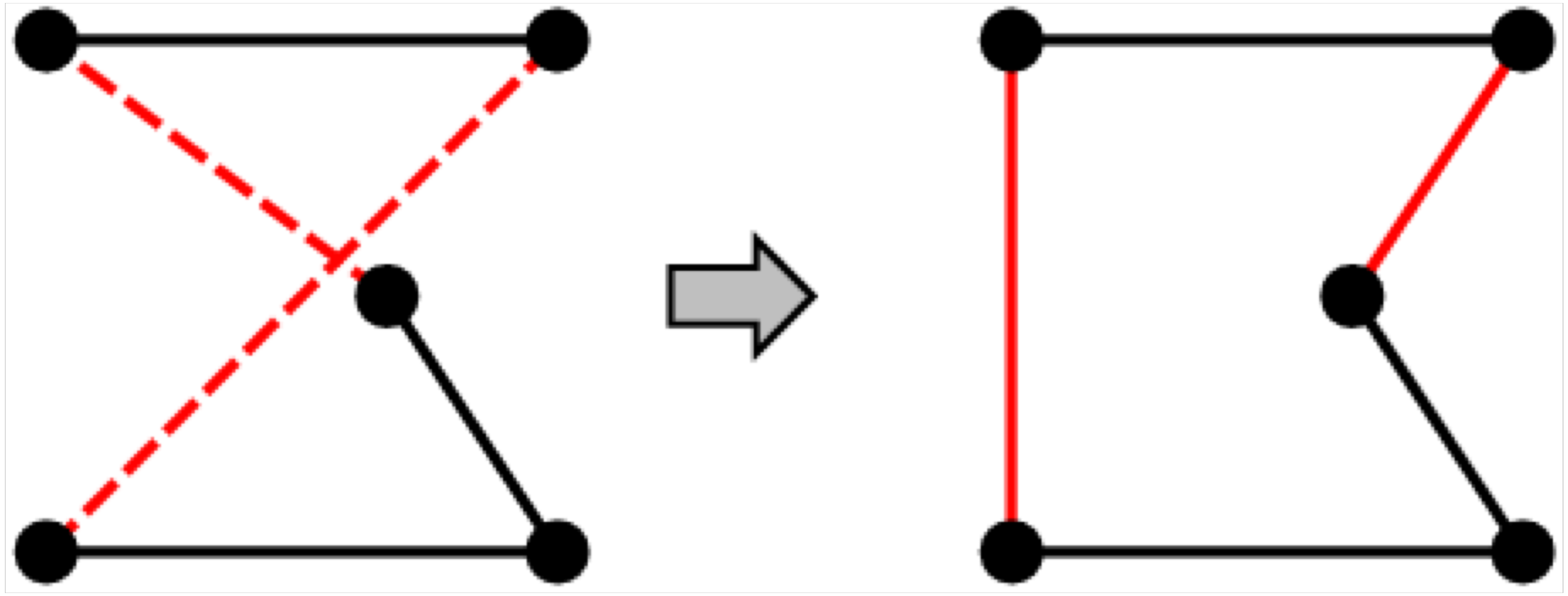
hence (memory-less) local search

Iterative refers to iterating between states

Improvement refers to later states improving some objective/goal function or satisfying more of the specified constraints over earlier states

Example: Traveling Salesman Problem (TSP)

Start with any complete tour, perform pairwise exchanges

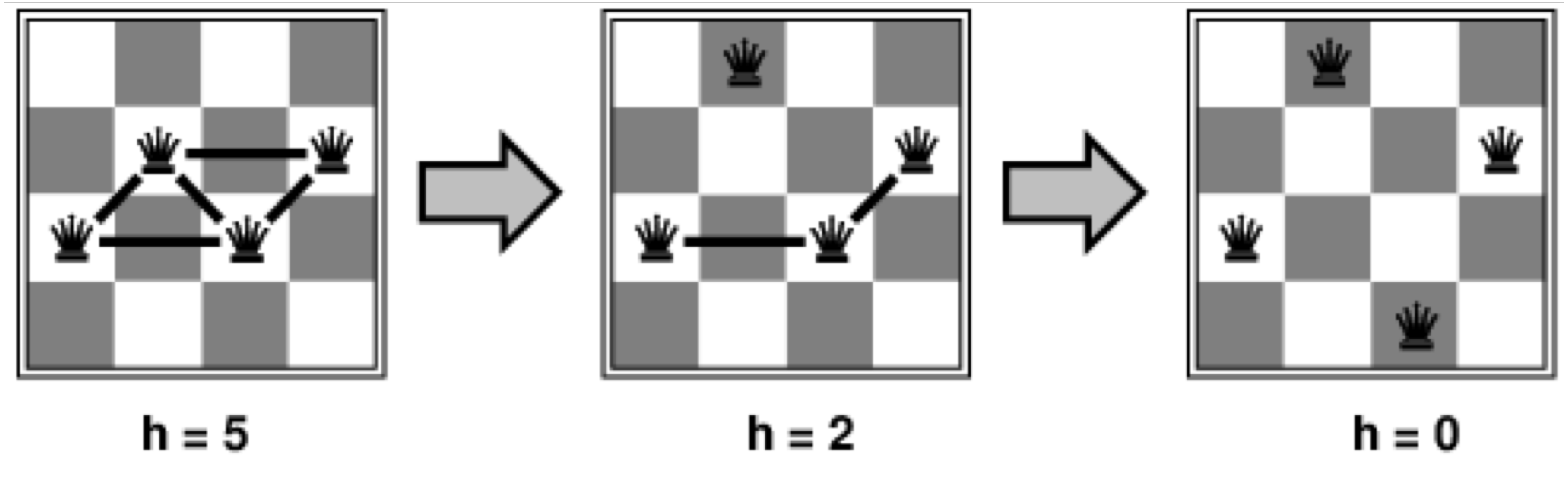


Variants of this approach get within **1%** of the optimal solution very quickly (even with thousands of cities)

Example *n*-queens

Put *n* queens on an n x n board with no two queens on the same row, column, or diagonal.

Move a queen to reduce number of conflicts.



Local search techniques can solve this problem almost instantaneously for very large n (n = 1 million) (recall an 8x8 board has 8^8 states (≈ 17 million states).

(Simple) Hill Climbing

"Like climbing Everest in thick fog with amnesia".

```
function Hill-Climbing(problem) returns a state (local optimum)
    inputs: problem, a problem
    local variables:  current (a node)
                      neighbor (a node)
    current ← MAKE-NODE(INITIAL-STATE [problem])
    loop do
        neighbor ←  a successor of current
        If Value[neighbor] is not better than Value[current]
                then return State ← [current]
        current ← neighbor
    end
```

Hill Climbing for Discrete State Spaces

How is the neighbor of a current state generated? **Varies with approach…**

If state space is discrete and neighbor list is finite, all neighbors of a current state can be considered:

- **Steepest hill climbing**: compare best neighbor to current
- **First-choice hill climbers** use the first choice that is improves on current

What if neighbors cannot be enumerated? What if state space is continuous?

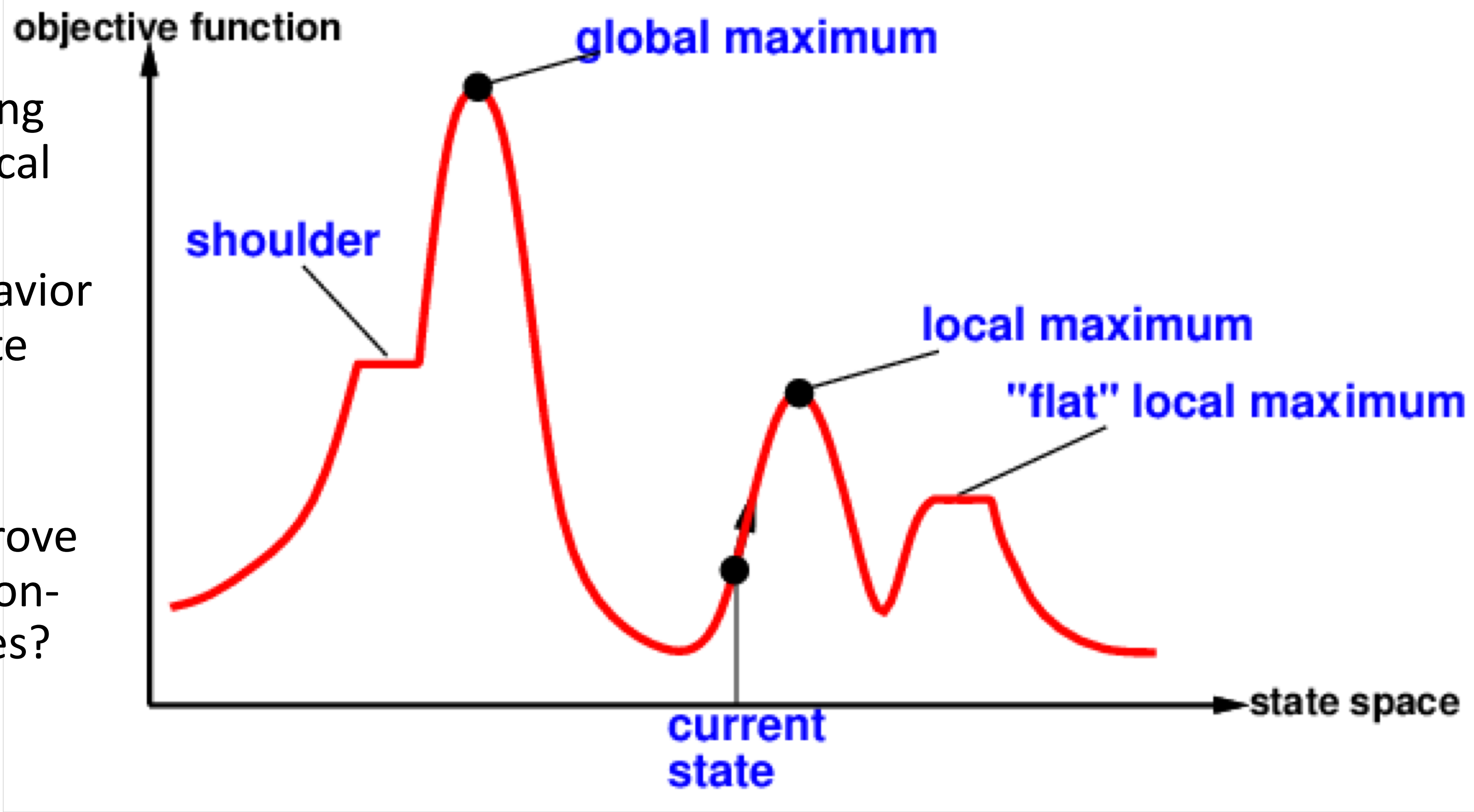
- **Stochastic hill climbing**: generate neighbor at random (continuous spaces, perform a small perturbation to generate neighbor)
- **Gradient-based variants**: for continuous state spaces
  - (Conjugate) Gradient Descent/Ascent
  - Other numerical optimization algorithms (beyond scope of CS 444)

Hill Climbing and Premature Convergence

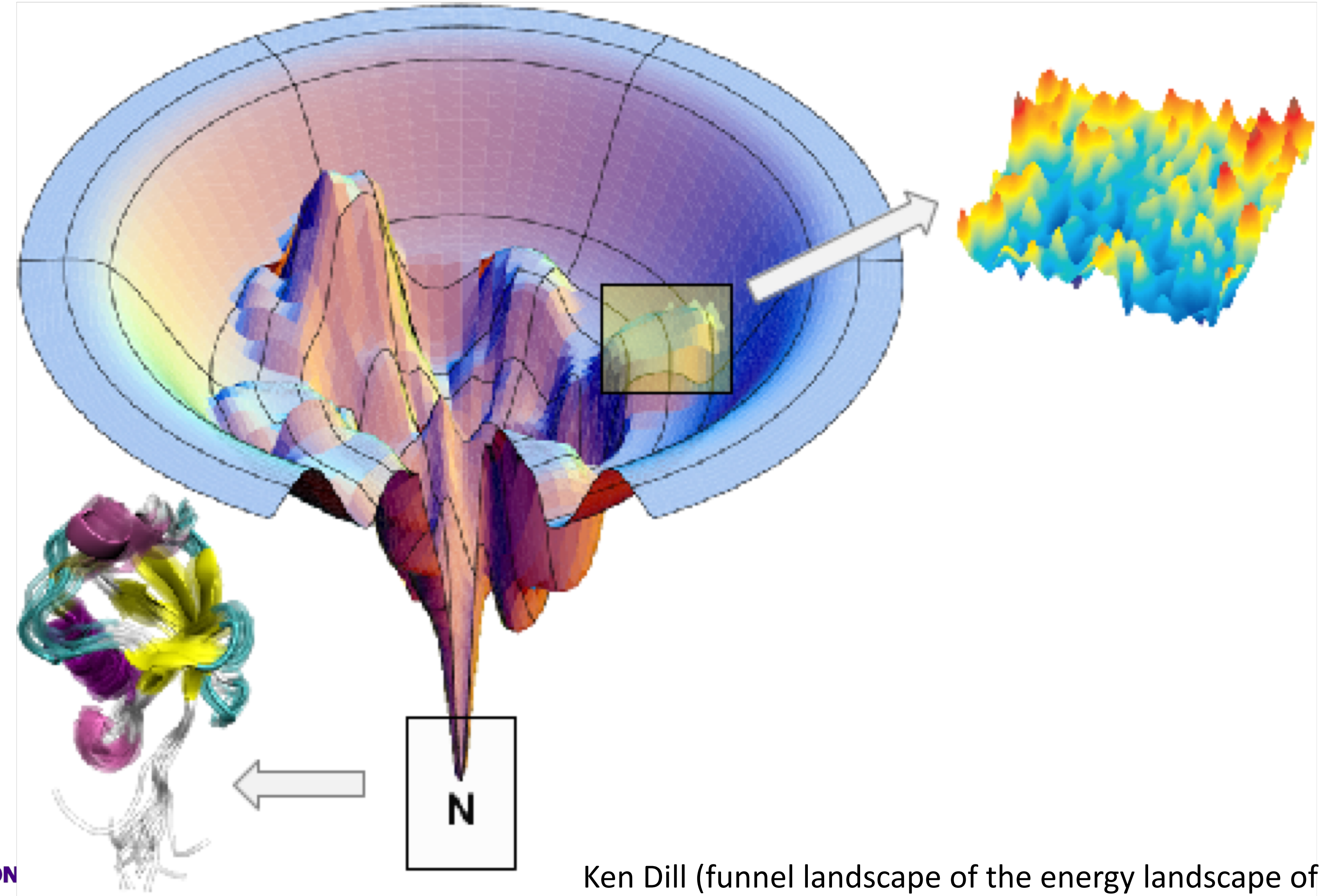
Why is simple hill climbing an its variants realizations of local search?

Useful to consider state space landscape

- Simple hill climbing converges to a local optimum
- When is this behavior sufficient to locate the goal = global optimum?
- How can we improve its behavior on non-convex landscapes?



Challenging Nonconvex Fitness Landscapes



Ken Dill (funnel landscape of the energy landscape of a protein)

JAMES MADISON UNIVERSITY

Three General Mechanisms to Avoid Premature Convergence

Randomization:

- Random/multi restart allows embarrassing parallelization
- Iterated Local Search (ILS)

Memory-less randomized/stochastic search optimization:

Monte Carlo search

Simulated Annealing Monte Carlo

Memory-based randomized search:

- Memory via search structure
  - List: tabu search
  - Tree-/graph based search
- Memory via population
  - Evolutionary search strategies
  - Evolutionary Algorithms (Eas)
  - Genetic Algorithms (GA)

Random-restart Hill Climbers

Idea: Launch multiple hill climbers from different initial states/configurations.

Bonus: Amenable to embarrassing parallelization.

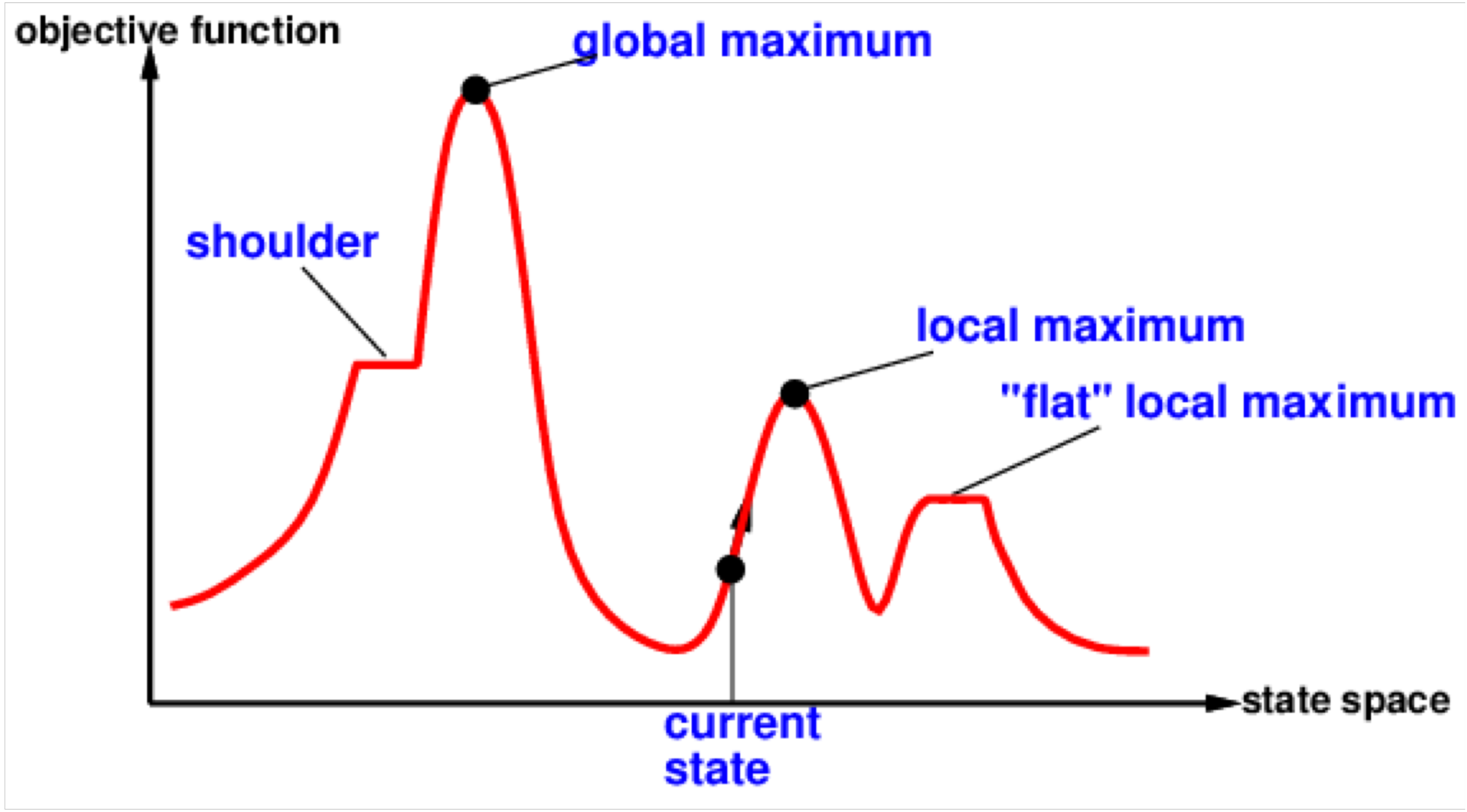
Take-away: It is often better to spend CPU time exploring the space, then carefully optimizing from an initial condition.

Why?

Repeated restarts give a global view of the state space (instead of just the local one provided by each climber).

Drawback? The hill climbers do not talk to one another.

Escaping a local maximum/minimum?



How to escape from a local minimum?

Make a random move – this is what we call **Iterated Local Search (ILS)**

Iterated Local Search

Start at a given initial state

Until some budget is exhausted or other termination criterion is reached.

Iterate between two types of moves:

- **Local improvement** Go from current state to a neighboring local optimum
- **Local randomization** Modify some variable of a local optimum to get a worse, adjacent state (not necessarily a neighbor

ILS is also known as Basin Hopping (BH)

How to design effective local randomization strategies?

- Domain-specific
- Introduce enough change but not too much change

Olson, Hashmi, Molloy, Shehu. Basin Hopping as a General and Versatile Optimization Framework for the Characterization of Biological Macromolecules. Advances in Artificial Intelligence Journal, 2012 (special issue on AI Applications in Biomedicine).

Monto Carlo (MC) Search

While hill climbing is **monotonic** (strictly improvements), MC allows hopping to a worse neighbor. Temperature controls how often.

function MC(problem, T) **returns** a state (local optimum)
  Inputs: problem, a problem
    T, temperature
  Local variables: current (a node)
    next (a node)
  current $\leftarrow$ MAKE-NODE(INITIAL-STATE [problem])
  for t $\leftarrow$ 1 to ∞ **do**
    if $T = 0$ **then return** *current*
    next $\leftarrow$ RANDOM-SUCCESSOR(current)
    $\Delta E \leftarrow$ VALUE[next] – VALUE[current]
    if $\Delta E > 0$ *current* $\leftarrow$ next
    else current $\leftarrow$ next with probability $e^{\Delta E / T}$
  end

Simulated Annealing Monte Carlo (SA-MC)

Idea: escape local maxima/minima allowing some bad moves, but, **gradually decrease their size and frequency**

```
function SA(problem, T) returns a state (local optimum)
    Inputs:    problem, a problem
               schedule, a mapping from time to "temperature"
    Local variables: current (a node)
                     next (a node)
                     T, a "temperature" controlling prob of bad
                        move
    current ← MAKE-NODE(INITIAL-STATE [problem])
    for t ← 1 to ∞ do
        T ← schedule[t]
        if T = 0 then return current
        next ← RANDOM-SUCCESSOR(current)
        ΔE ← VALUE[next] – VALUE[current]
        if ΔE > 0 current ← next
        else current ← next with probability e^(ΔE/T)
    end
```

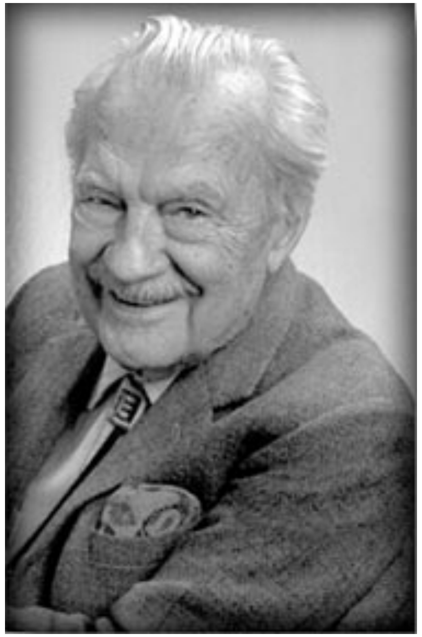
Concepts of Exploiting Temperature

How should a temperature schedule be constructed?

- Fixed, proportional cooling schedule
- Dynamic, adaptive (popular in tempering, chemistry, material science, robotics)

Other way to use temperature:

- Diversify restarts (each can use a different temperature schedule)
- Threads exchange states (known as replica exchange, very popular in physical and chemistry)



At fixed "temperature" T, the state occupation probability reaches the Boltzman distribution (https://en.wikipedia.org/wiki/Boltzmann_distribution).

$$p(x) = \alpha e^{\frac{E(x)}{kT}}$$

Sometimes this is called Metropolis Monte Carlo (MC), devised by Nicholas Metropolis (and some other Manhattan scientists) in 1953 that allow them to model the "states" of systems using computers.

Combination Strategies

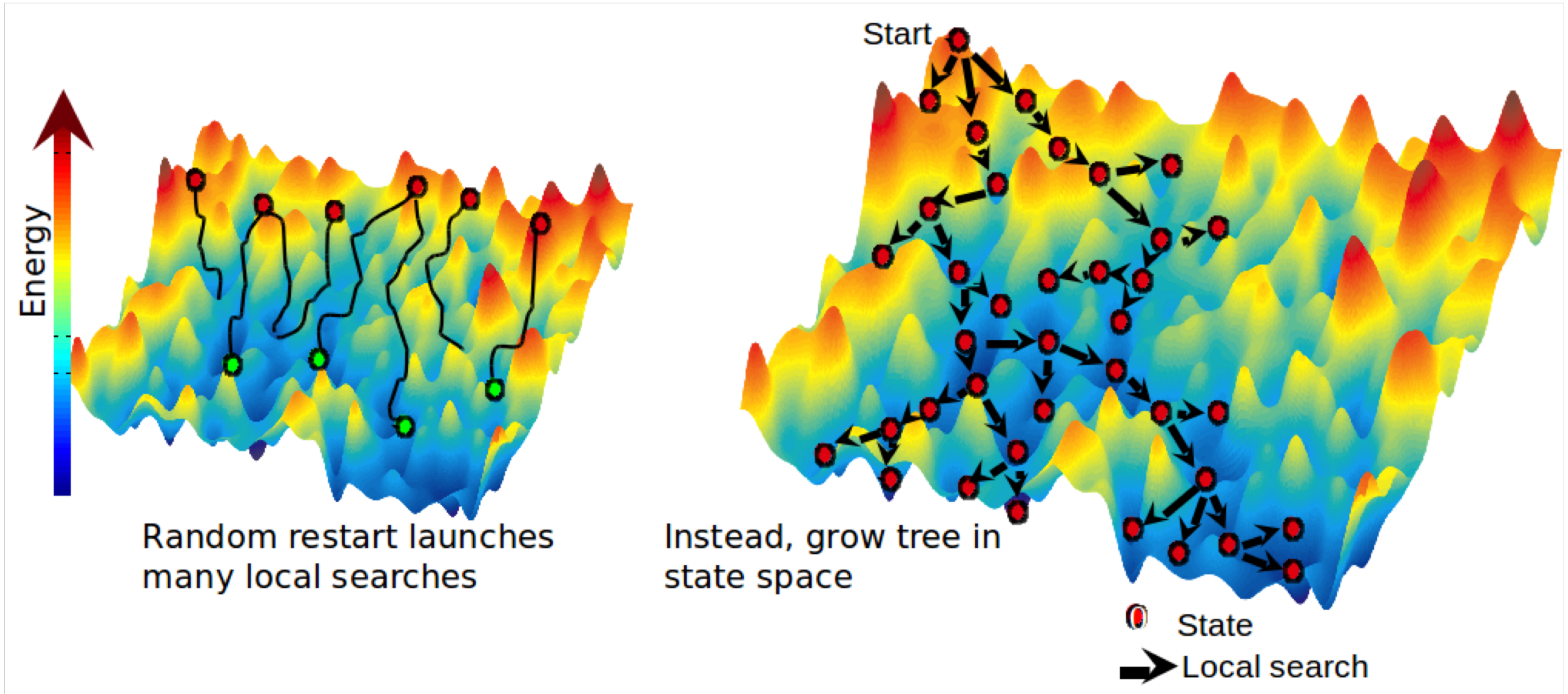
ILS + MC ← Monte Carlo with minimization

Very popular in biomolecular structure/energy optimization

Characterizing Energy Landscapes of Peptides using a Combination of Stochastic Algorithms. Didier Devaurs, Kevin Molloy, Marc Vaisset, Amarda Shehu, Thierry Siméon, and Juan Cortés. *IEEE Transactions in NanoBioScience, 2015*.

Idea: Keep states generated so far in a tree or graph. Centralizes redundant local searches.

Integrate local searches in a global search structure.



Probabilistic Search and Energy Guidance for Biased Decoy Sampling in Ab-initio Protein Structure Prediction. Molloy et al. IEEE Trans in Computational Biology and Bioinformatics, 2013.

Tabu Search

Idea: Avoid generating same state

Tabu: list of states generated so far

Tabu list may also include set of moves that yield redundant states

Tabu considered an **evolutionary search strategy**

More general concept: **hall of fame**

Local Beam Search

Idea: Don't keep just a single state, keep *k* states.

Not the same a *k* searches in parallel!

Search that finds good states recruits other searches to join them

Generate *k* starting states at random.

REPEAT

For each state *k* generate a successor state

Pick the best *k* states from the set of 2*k states (the originals and the "offspring"*

Issues/Problems?

Quite often, all k states end up on some local "hill"

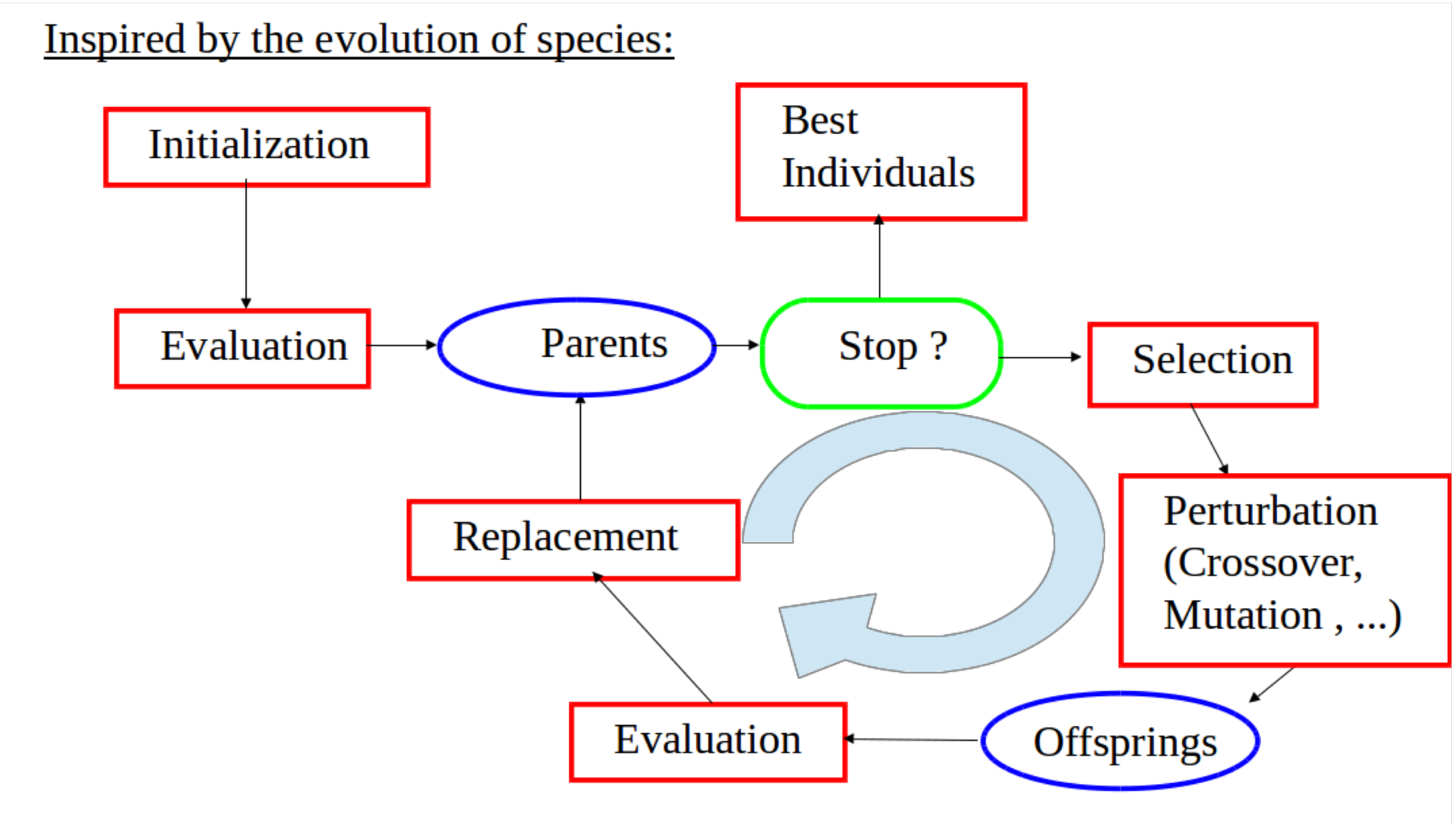
Solution: choose k successors randomly (biased towards "good" states". This is call Monte Carlo sampling (robotics/computer vision use this in particle filters).

JMU JAMES MADISON UNIVERSITY

Memory-based Search via Population: Evolutionary Computation

This is a subfield of AI and is very popular

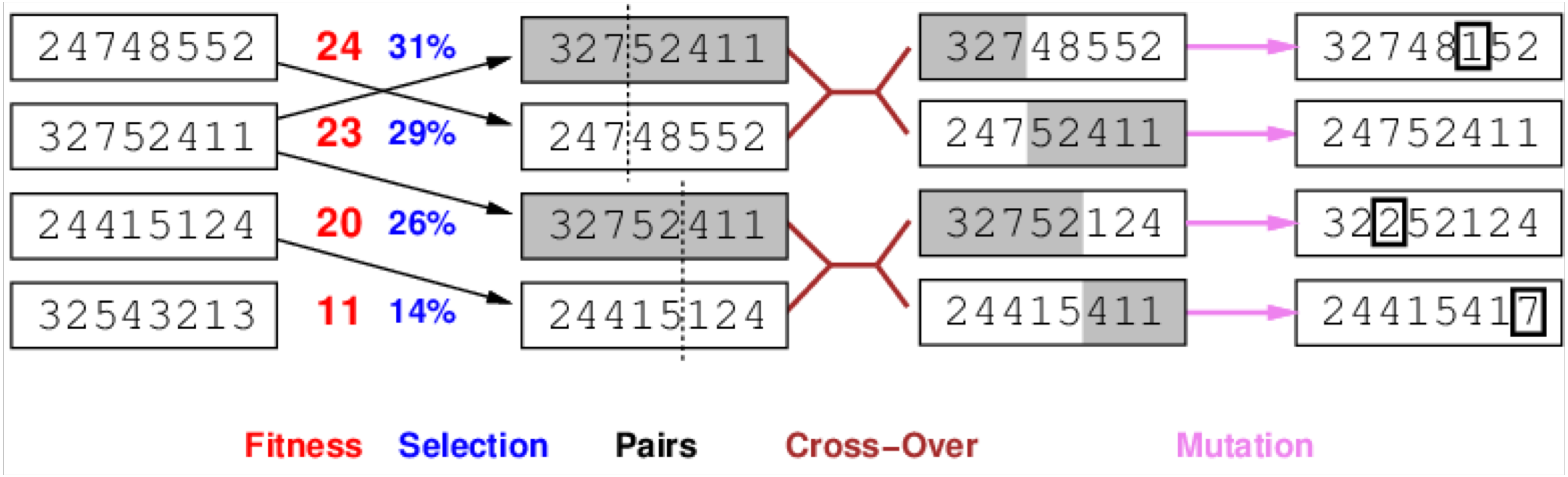
Idea: Mimic natural selection to arrive at solutions that have a beter chance of including the global optimum than local search. Many strategies exist.



Genetic Algorithms (GAs)

Similar to stochastic beam search + generate successors from pairs of states.

<u>States representing the 8 queens problem</u>



Typically new states are added to old states and the selection process continues (only the strong survive).

EA Summary

EAs currently some of the most powerful (randomize) solves for the toughest academic and industrial optimization problems.

Some EAs methods look similar to what we have already discussed. This is true:

Example: ILS is just 1+1 EA

Awareness of developments in different communities inspires new strategies or combination of strategies for more powerful randomized search algorithms.

Summary

Local search involves moving from "state" to "state" through a successor function and, in general, in a memory-less way.

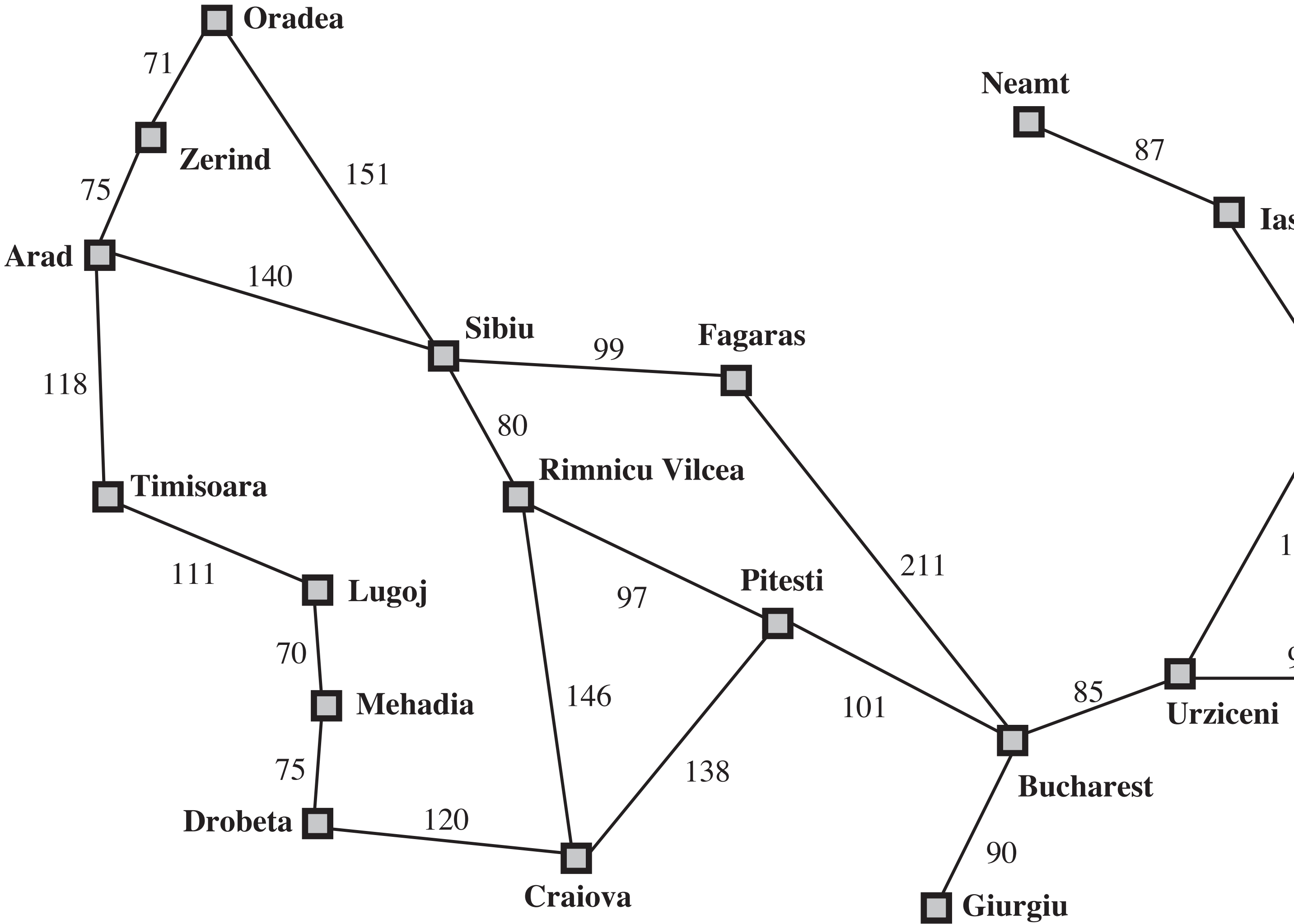
We discussed:

- Simple Hill Climbers (strictly better moves)
- Hill climbers that escape local minimum/maximums (iterated local search/random restarts)
- Incorporate "temperature" to allow some bad moves to escape local minimum (potentially with a "cooling" schedule")
- Local beam search (close to an EA)
- Evolutionary algorithms for successor functions and selection

Problem A* Search

Map out the tree that A* would use utilizing the straight line distance heuristic for a trip from Sibiu to Bucharest.

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



Problem 4.1

Give the name of the algorithm that results from each of the following special cases: local beam search with $k = 1$