Lecture 4: Local and Randomized/Stochastic Search

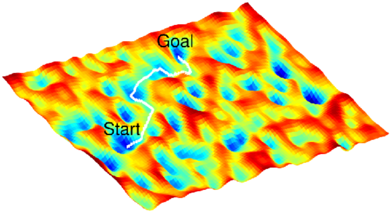
CS 580 (001) - Spring 2018

Amarda Shehu

Department of Computer Science
George Mason University, Fairfax, VA, USA

February 14, 2018

- Graph search algorithms conduct systematic search
- Assume state space is finite and can fit in memory
- State space can be large, not even finite
- 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

In many **optimization** problems, **path** is irrelevant;
the goal state itself is the solution

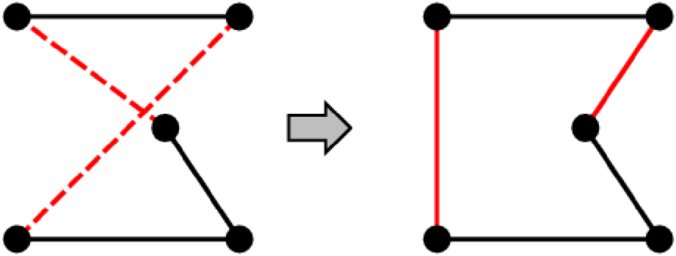
Then state space = set of "complete" configurations;
find **optimal** configuration (explicit constraints or objective/fitness function)

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
improvement may not be immediate (more on this later)

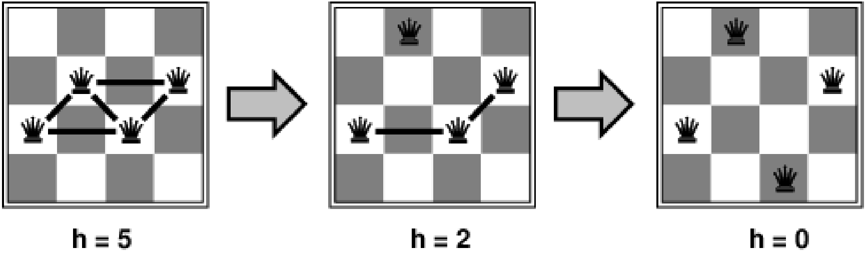
Start with any complete tour, perform pairwise exchanges



Variants of this approach get within 1% of optimal very quickly with thousands of cities

Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal

Move a queen to reduce number of conflicts



Almost always solves n-queens problems almost instantaneously for very large n, e.g., $n = 1 million$

"Like climbing Everest in thick fog with amnesia"
"Like hopping kangaroos"

function HILL-CLIMBING(*problem*) **returns** a state that is a 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

(Simple) Hill Climbing for Discrete State Spaces

How is the neighbor of a current state generated?

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

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

- **Stochastic hill climbing:** select neighbor at random

- **Gradient-based variants:** for continuous state spaces
  - (Conjugate) Gradient Descent/Ascent
  - Other numerical optimization algorithms (taught in Numerical Methods courses)

Suppose we want to site three airports in Romania:
– 6-D state space defined by (x_1, y_2), (x_2, y_2), (x_3, y_3)
– objective function $f(x_1, y_2, x_2, y_2, x_3, y_3) =$
sum of squared distances from each city to nearest airport

Gradient-based methods (referred to as potential field methods in robotics) compute

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3}\right)$$

to increase/reduce f, e.g., by $\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x})$
Sometimes can solve for $\nabla f(\mathbf{x}) = 0$ exactly (e.g., with one city).
Steepest descent, gradient descent
Conjugate gradient descent methods, like Newton–Raphson (1664, 1690) iterate
$\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x}) \nabla f(\mathbf{x})$
to solve $\nabla f(\mathbf{x}) = 0$, where $\mathbf{H}_{ij} = \partial^2 f / \partial x_i \partial x_j$

What if cannot analytically calculate the derivatives? empirical gradient considers $\pm\delta$ change in each coordinate

Suppose we want to site three airports in Romania:
– 6-D state space defined by (x_1, y_2), (x_2, y_2), (x_3, y_3)
– objective function $f(x_1, y_2, x_2, y_2, x_3, y_3) =$
sum of squared distances from each city to nearest airport

Gradient-based methods (referred to as potential field methods in robotics) compute

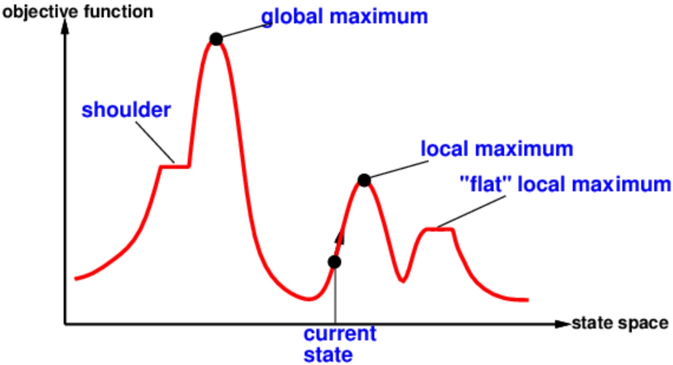
$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3}\right)$$

to increase/reduce f, e.g., by $\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x})$
Sometimes can solve for $\nabla f(\mathbf{x}) = 0$ exactly (e.g., with one city).
Steepest descent, gradient descent
Conjugate gradient descent methods, like Newton–Raphson (1664, 1690) iterate
$\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x}) \nabla f(\mathbf{x})$
to solve $\nabla f(\mathbf{x}) = 0$, where $\mathbf{H}_{ij} = \partial^2 f / \partial x_i \partial x_j$

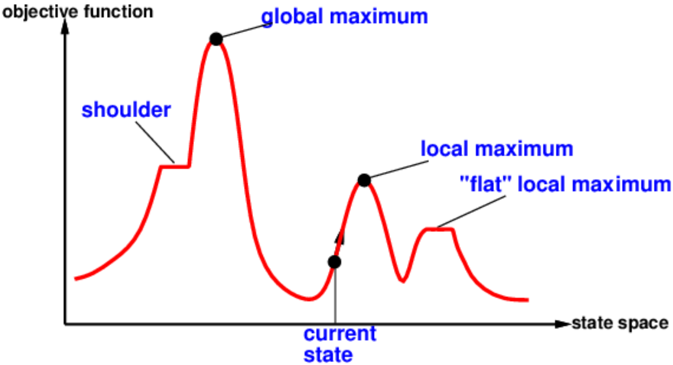
What if cannot analytically calculate the derivatives? empirical gradient considers $\pm\delta$ change in each coordinate

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

Why is simple hill climbing and its variants realizations of local search?
Useful to consider state space landscape

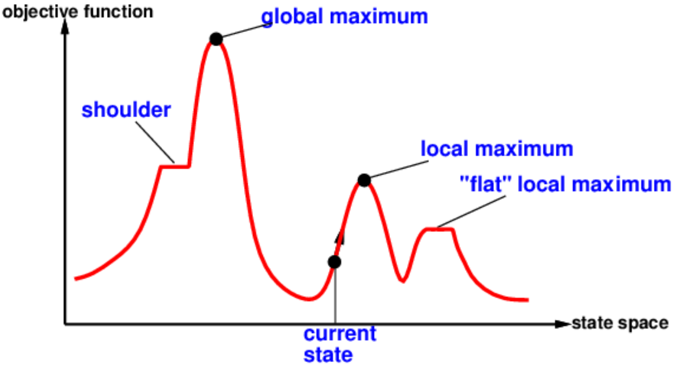


Why is simple hill climbing and its variants realizations of local search?
Useful to consider state space landscape



- simple hill climbing converges to a local optimum ☹

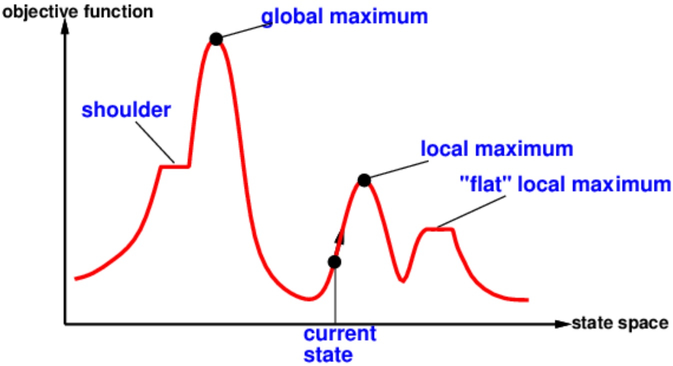
Why is simple hill climbing and its variants realizations of local search?
Useful to consider state space landscape



- simple hill climbing converges to a local optimum ☹
- when is this behavior sufficient to locate the goal=global optimum?

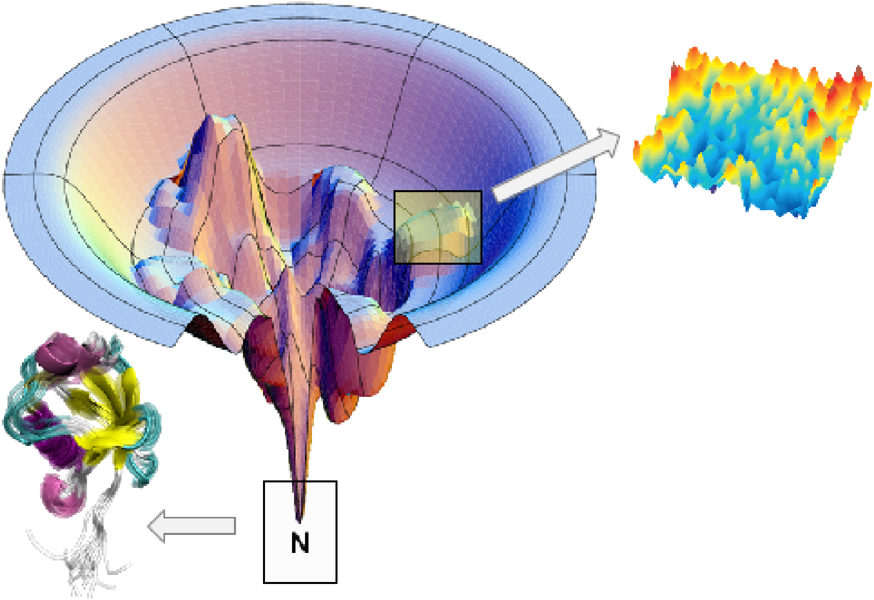
(Simple) Hill Climbing and Premature Convergence

Why is simple hill climbing and 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?

Premature Convergence in Nonconvex (Fitness) Landscapes



Three General Mechanisms to Avoid **Premature Convergence**

- Randomization:
  - Random/multi restarts allows embarrasing parallelization
  - Iterated Local Search (ILS)
- Memory-less randomized/stochastic search/optimization:
  - Monte Carlo
  - 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 (GAs),
      Genetic Programming Algorithms (GPs)

Three General Mechanisms to Avoid Premature Convergence

- Randomization:
  - Random/multi restarts allows embarrasing parallelization
  - Iterated Local Search (ILS)
- Memory-less randomized/stochastic search/optimization:
  - Monte Carlo
  - 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 (GAs),
      Genetic Programming Algorithms (GPs)

Categorizations can be subjective, with no clear borders

Random-restart Hill Climbing

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Random-restart Hill Climbing

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Random-restart Hill Climbing

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Also known as shotgun hill climbing (or shooting kangaroos)

Random-restart Hill Climbing

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Also known as shotgun hill climbing (or shooting kangaroos)

Surprisingly effective on many difficult optimization problems.

Random-restart Hill Climbing

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Also known as shotgun hill climbing (or shooting kangaroos)

Surprisingly effective on many difficult optimization problems.

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

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Also known as shotgun hill climbing (or shooting kangaroos)

Surprisingly effective on many difficult optimization problems.

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

Why?

Random-restart Hill Climbing

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Also known as shotgun hill climbing (or shooting kangaroos)

Surprisingly effective on many difficult optimization problems.

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

Why? Restarts give global view of state space

Random-restart Hill Climbing

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Also known as shotgun hill climbing (or shooting kangaroos)

Surprisingly effective on many difficult optimization problems.

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

Why? Restarts give global view of state space

Drawback?

Random-restart Hill Climbing

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Also known as shotgun hill climbing (or shooting kangaroos)

Surprisingly effective on many difficult optimization problems.

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

Why? Restarts give global view of state space

Drawback? Memory-less,

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Also known as shotgun hill climbing (or shooting kangaroos)

Surprisingly effective on many difficult optimization problems.

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

Why? Restarts give global view of state space

Drawback? Memory-less, The **hill climber threads** do not talk to one another.

Random-restart Hill Climbing

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Also known as shotgun hill climbing (or shooting kangaroos)

Surprisingly effective on many difficult optimization problems.

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

Why? Restarts give global view of state space

Drawback? Memory-less, The **hill climber threads** do not talk to one another. More on this later.

Random-restart Hill Climbing

A meta-algorithm that can encapsulate any local search algorithm, not just hill climbing

Launch multiple hill climbers from different initial states/configurations

Amenable to embarrasing parallelization

Also known as shotgun hill climbing (or shooting kangaroos)

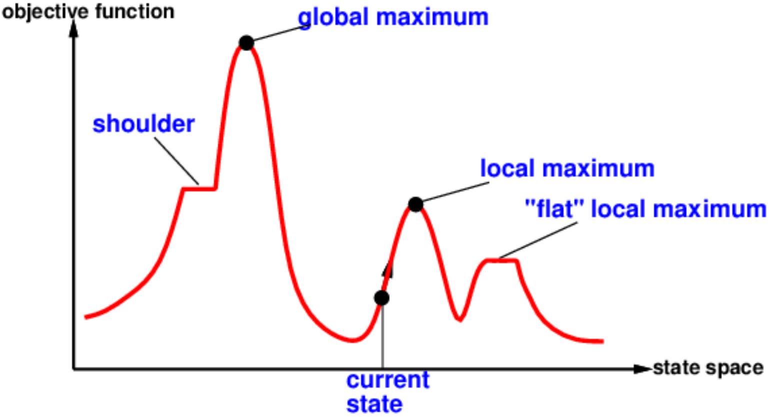
Surprisingly effective on many difficult optimization problems.

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

Why? Restarts give global view of state space

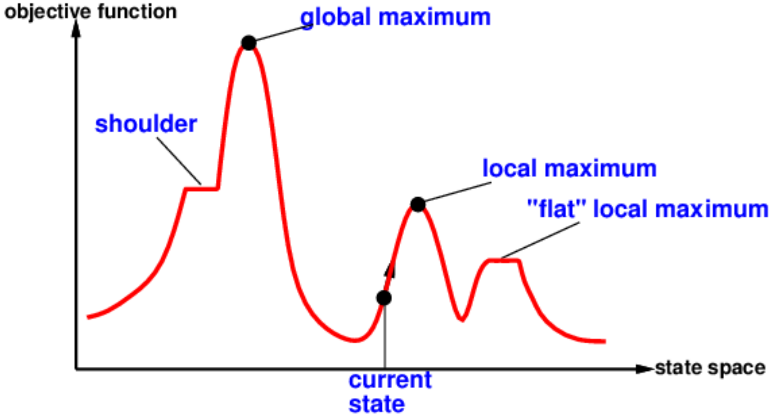
Drawback? Memory-less, The **hill climber threads** do not talk to one another. More on this later.

Kicking Hill Climbing out of Local Optimum



How to escape from a local optimum?

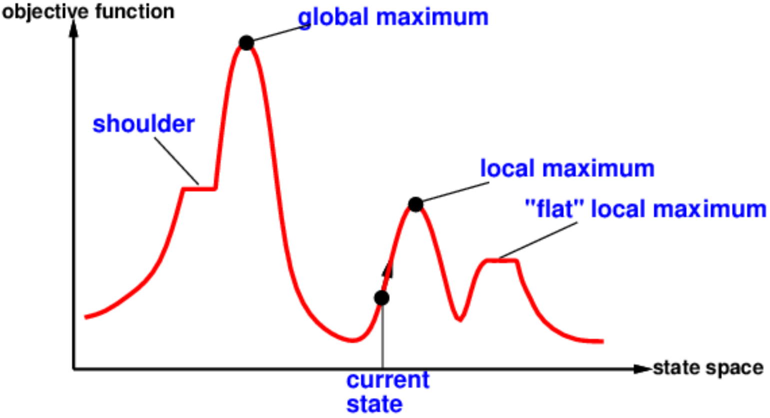
Kicking Hill Climbing out of Local Optimum



How to escape from a local optimum?

Kick the kangaroo out - make a random move 😉

Kicking Hill Climbing out of Local Optimum

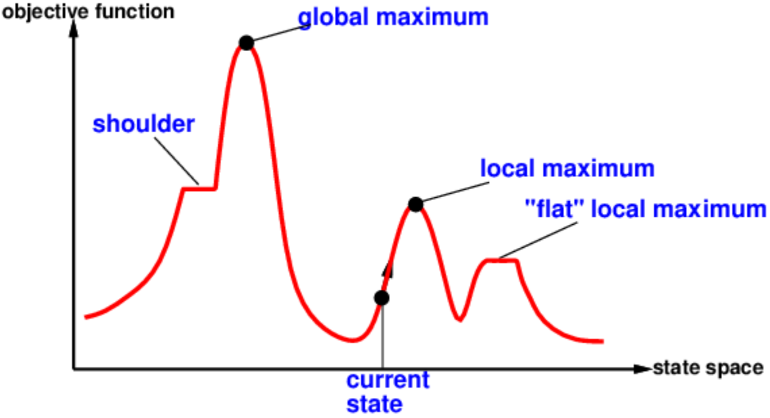


How to escape from a local optimum?

Kick the kangaroo out - make a random move 😆

Iterated Local Search

Kicking Hill Climbing out of Local Optimum



How to escape from a local optimum?

Kick the kangaroo out - make a random move 😉

Iterated Local Search

Iterated Local Search

Start at given initial state

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

Start at given initial state

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

Iterate between two types of moves:

Iterated Local Search

Start at given initial state

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

Iterate between two types of moves:

local improvement

Iterated Local Search

Start at given initial state

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

Iterate between two types of moves:

- **local improvement**
- **local randomization**/perturbation/variation

Iterated Local Search

Start at given initial state

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

Iterate between two types of moves:

- **local improvement**
- **local randomization**/perturbation/variation

Local improvement: as before, go from current state to a neighboring local optimum

Iterated Local Search

Start at given initial state

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

Iterate between two types of moves:

local improvement

local randomization/perturbation/variation

Local improvement: as before, go from current state to a neighboring local optimum

Local randomization: modify some variable of local optimum to get a worse, adjacent state (not necessarily neighbor)

Iterated Local Search

Start at given initial state

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

Iterate between two types of moves:

- **local improvement**
- **local randomization**/perturbation/variation

Local improvement: as before, go from current state to a neighboring local optimum

Local randomization: modify some variable of local optimum to get a worse, adjacent state (not necessarily neighbor)

ILS also known as Basin Hopping (BH)

How to design effective local randomization strategies?

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

Examples from Research Literature

Gross, Jamali, Locatelli, Schoen. Solving the problem of packing equal and unequal circles in a circular container. J Glob Optim 47:63-81, 2010.

Olson, Hashmi, Molloy, Shehu. Basin Hopping as a General and Versatile Optimization Framework for the Characterization of Biological Macromolecules. Advances in Artificial Intelligence J 2012, 674832 (special issue on Artificial Intelligence Applications in Biomedicine).

Can encapsulate ILS within random restart

ILS also known as Basin Hopping (BH)

How to design effective local randomization strategies?

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

Examples from Research Literature

Gross, Jamali, Locatelli, Schoen. Solving the problem of packing equal and unequal circles in a circular container. J Glob Optim 47:63-81, 2010.

Olson, Hashmi, Molloy, Shehu. Basin Hopping as a General and Versatile Optimization Framework for the Characterization of Biological Macromolecules. Advances in Artificial Intelligence J 2012, 674832 (special issue on Artificial Intelligence Applications in Biomedicine).

Can encapsulate ILS within random restart

Can be seen as a variant of hill climbing
While hill climbing is monotonic (strict on improvement), MC allows hopping to a worse neighbor
Temperature parameter controlls how often

function MC(*problem*, *T*) **returns** a solution state
    inputs: *problem*, a problem
            T, a "temperature" controlling prob. of downward steps
    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 $\leftarrow$ a randomly selected successor of *current*
        $\Delta E \leftarrow$ VALUE[*next*] – VALUE[*current*]
        if $\Delta E > 0$ **then** *current* $\leftarrow$ *next*
        else *current* $\leftarrow$ *next* only with probability $e^{\Delta E/T}$

Simulated Annealing Monte Carlo (SA-MC)

Idea: escape local maxima by allowing some "bad" moves **but gradually decrease their size and frequency**

function SA(*problem*, *schedule*) **returns** a solution state
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 downward steps

current ← MAKE-NODE(INITIAL-STATE[*problem*])
for $t \leftarrow 1$ **to** ∞ **do**
$T \leftarrow schedule[t]$
if $T = 0$ **then return** *current*
next ← a randomly selected successor of *current*
$\Delta E \leftarrow$ VALUE[*next*] – VALUE[*current*]
if $\Delta E > 0$ **then** *current* ← *next*
else *current* ← *next* only with probability $e^{\Delta E/T}$

SA-MC Behavior

At fixed "temperature" T, state occupation probability reaches Boltzman distribution

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

T decreased slowly enough $\Longrightarrow$ always reach best state x^* because $e^{\frac{E(x^*)}{kT}} / e^{\frac{E(x)}{kT}} = e^{\frac{E(x^*)-E(x)}{kT}} \gg 1$ for small T

Is this necessarily an interesting guarantee??

Devised by Metropolis et al., 1953, for physical process modelling

Sometimes referred to as **Metropolis** Monte Carlo (MMC)

Widely used in VLSI layout, airline scheduling, computational biology, chemistry, physics to find lowest-energy states of a complex system composed of many modules that constrain motions/placements of one another

Exploiting Concept of Temperature

How should next temperature be picked?

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

Other ways to use temperature

To diversify restart threads
Different MCs, each at their own temperature
Trivial way threads can exchange information:
- exchange current states every so often
- known as parallel tempering or replica exchange (popular in physics and chemistry)

Combination of Strategies

- ILS+MC $\rightarrow$ Monte Carlo with minimization

  very popular in biomolecular structure/energy optimization

- SA-MC + random restart

- Many **enhancement** strategies proposed to broaden the view of the state space afforded by local search

  Example: Local Beam Search

Local Beam Search

Idea: keep k states instead of 1; choose top k of all their successors

Not the same as k searches run in parallel!
Searches that find good states recruit other searches to join them

Problem: quite often, all k states end up on same local hill

Idea: choose k successors randomly, biased towards good ones

Observe the close analogy to natural selection!

Combination of Strategies

- ILS+MC $\rightarrow$ Monte Carlo with minimization

  very popular in biomolecular structure/energy optimization

- SA-MC + random restart

- Many **enhancement** strategies proposed to broaden the view of the state space afforded by local search

  Example: Local Beam Search
  Nomenclature: domain-specific
  In computational chemistry/physics: enhancement strategies
  In evolutionary computation: hybridization mechanisms
  In AI: local + global search

- Where is the global view?
  - a data structure that records visited states (robotics)
  - a more general concept: population (evolutionary computation)

Tabu Search

Idea: Avoid generating same state

Tabu: list of states generated so far

A generated state compared to tabu list for redundancy

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

Tabu considered an evolutionary search strategy

More general concept: **hall of fame** (next in evolutionary computation)

Tabu Search

Idea: Avoid generating same state

Tabu: list of states generated so far

A generated state compared to tabu list for redundancy

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

Tabu considered an evolutionary search strategy

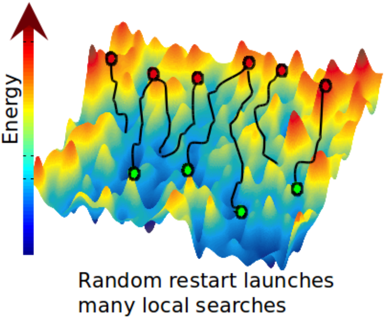
More general concept: **hall of fame** (next in evolutionary computation)

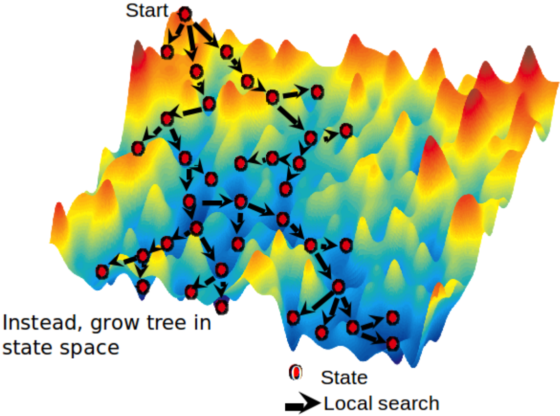
Tree-guided Search

Keep states generated so far in a tree or graph

Centralizes redundant local searches

Integrate local searches in a global search structure





Tree-guided Search

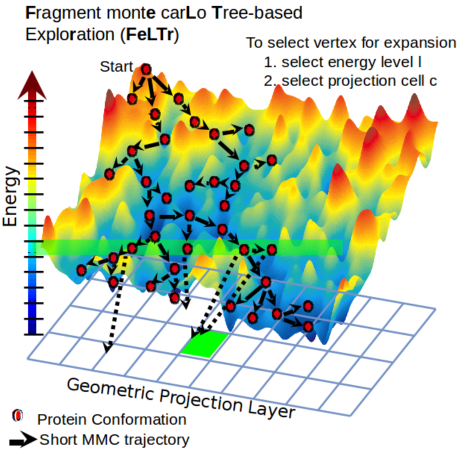
Keep states generated so far in a tree

Generate a new state in a two-punch, select-and-expand mechanism:

- Selection mechanism: Query tree to give you a (parent) state
  - **Probability distribution function** can be used to select parents (guide tree)
- Expand (with local search) from that state to get a candidate child state
- If candidate state not already similar to something in tree, add it as child, with corresponding edge
  - **Discretization/projection layers** to organize states so as to quickly tell whether a state is really new

Popular in robotics and computational biology:

RRT (robot motion planning)
EST (robot motion planning)
FeLTr – Olson, Shehu. IJRR 2010, SPRINT – Molloy, Shehu. BMC Struct Biol 2013 (for protein structure prediction and motion computation)



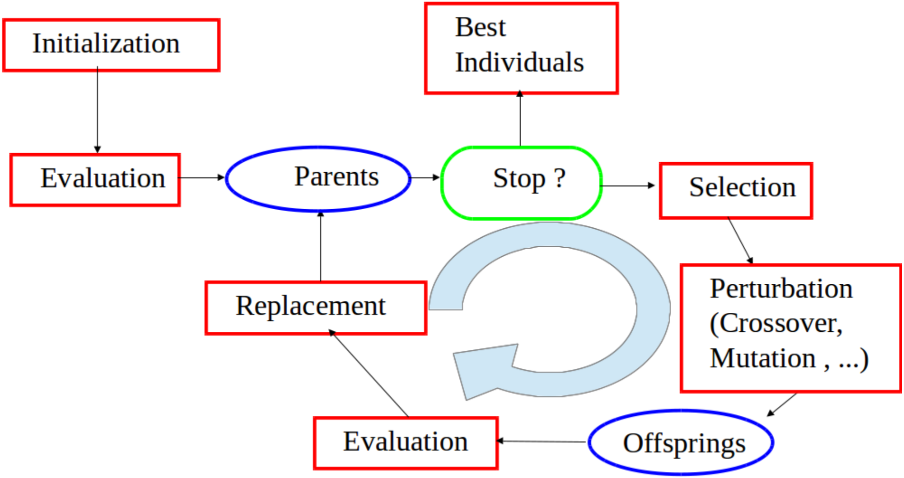
Subfield of AI

Idea: mimick natural selection to arrive at solutions that have a better chance of including the global optimum than local search

Many evolutionary search (ES) strategies exist
can learn about them in Nature-inspired Computing course (K. A. De Jong)

We will summarize main idea behind evolutionary algorithms (EAs) and provide specific realizations such as GAs and GPs

Inspired by the evolution of species:



Initalization Mechanism Define an initial population of states

Population **evolves over generations**

At each generation:
selection mechanism selects parents that will give offspring
Variation operator applied to one or two parents yield offspring
Replacement mechanism selects new population from only the offspring, or offspring and parent combined

Variation/perturbation operators:
- mutation changes one parent
- cross-over combines two parents to yield one or two offspring

Very rich framework:
e.g., if offspring subjected to local improvement, memetic EA
only value maintained but offspring not replaced – Baldwinian EA
offspring replaced with improved version – Lamarckian EA

Different decisions in each of the components yield different, complex behavior
Question: How is ILS/BH an EA?

Initalization Mechanism Define an initial population of states

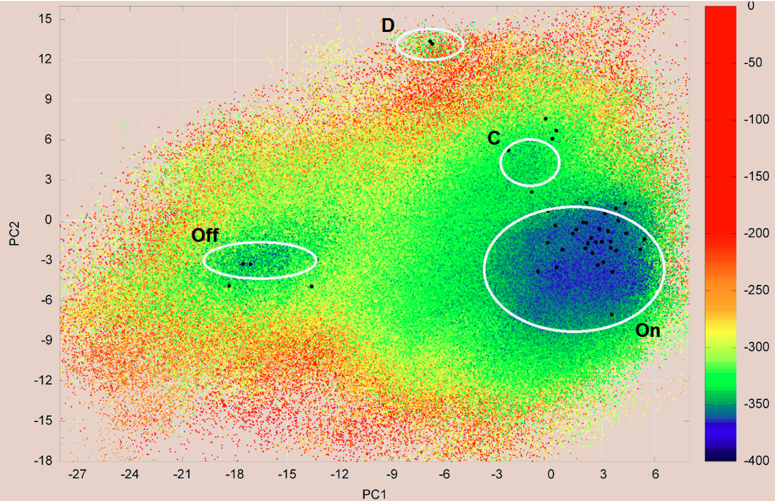
Population **evolves over generations**

At each generation:
selection mechanism selects parents that will give offspring
Variation operator applied to one or two parents yield offspring
Replacement mechanism selects new population from only the offspring, or offspring and parent combined

Variation/perturbation operators:
- mutation changes one parent
- cross-over combines two parents to yield one or two offspring

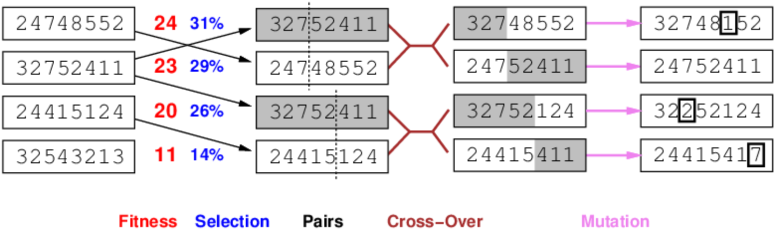
Very rich framework:
e.g., if offspring subjected to local improvement, memetic EA
only value maintained but offspring not replaced – Baldwinian EA
offspring replaced with improved version – Lamarckian EA

Different decisions in each of the components yield different, complex behavior
Question: How is ILS/BH an EA?



Mapping state space of H-Ras, a flexible, oncogenic protein.
Sapin, De Jong, Shehu IEEE BIBM 2015.

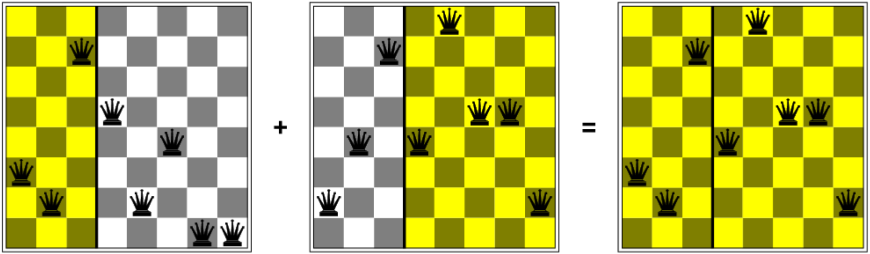
= stochastic local beam search + generate successors from **pairs** of states
an EA with crossover



From GAs to GPs

GAs require states encoded as strings (GPs use programs)

Crossover helps **iff substrings are meaningful components**



GAs $\neq$ evolution: e.g., real genes encode replication machinery!

GPs designed to evolve programs

Attributed to Koza, 1992

Main change from a GA: states not binary or real-valued, but complex tree-based structure representations of programs

Adapted by Kamath, De Jong, and Shehu to evolve features capturing complex signals of function in DNA sequences (IEEE TCBB 2013, PLoS One 2014)

Summary

EAs currently some of the most powerful (randomized) solvers for the toughest academic and industrial optimization problems

Tree-guided search popular in robotics can be encapsulated in EA template

Literature on randomized search/optimization algorithms is very rich

Developments from different sub-communities within AI and different communities outside of computer science

Often, similar ideas reproduced but differently termed
Example: basin hopping is just ILS

Example: ILS is just 1+1 EA
Example: Metropolis Monte Carlo (MMC) with minimization is just ILS
Example: MMC with minimization is just 1+1 memetic/hybrid EA

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