Computational Biomolecular Design

Algorithms for Protein Design

Presented by

Dr. Fayyaz-ul-Amir Afsar Minhas

http://faculty.pieas.edu.pk/fayyaz

Department of Computer & Information Sciences
Pakistan Institute of Engineering & Applied Sciences
PO Nilore, Islamabad 45650
Pakistan

From Previous Lecture

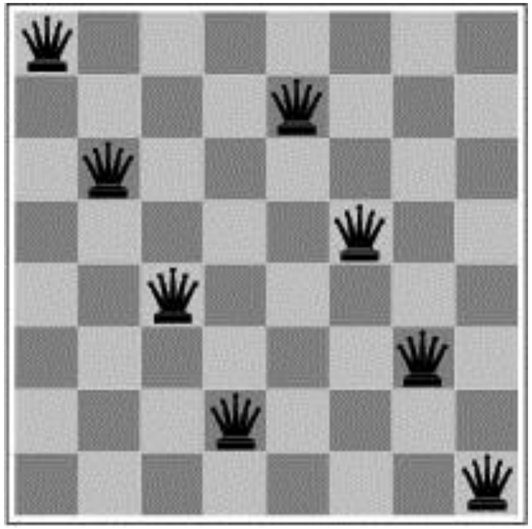
- Let S (x_1, x_2, …, x_N) represent a protein of length N
  - x_i represents the ith residue in the protein
    - Its type
    - Its torsion angles
    - Its rotamer conformation

- Its energy values is given E (X) = E (S (X))

- There are two problems now
  - Protein Re-design
    - Given a protein, optimize its torsion angles (no change in type of amino acid)
  - Protein Design
    - Given a backbone, find the type of the amino acid and its optimal configuration

Local search and optimization

- Previously: systematic exploration of search space.
  - Path to goal is solution to problem
- YET, for some problems path is irrelevant.
  - E.g 8-queens

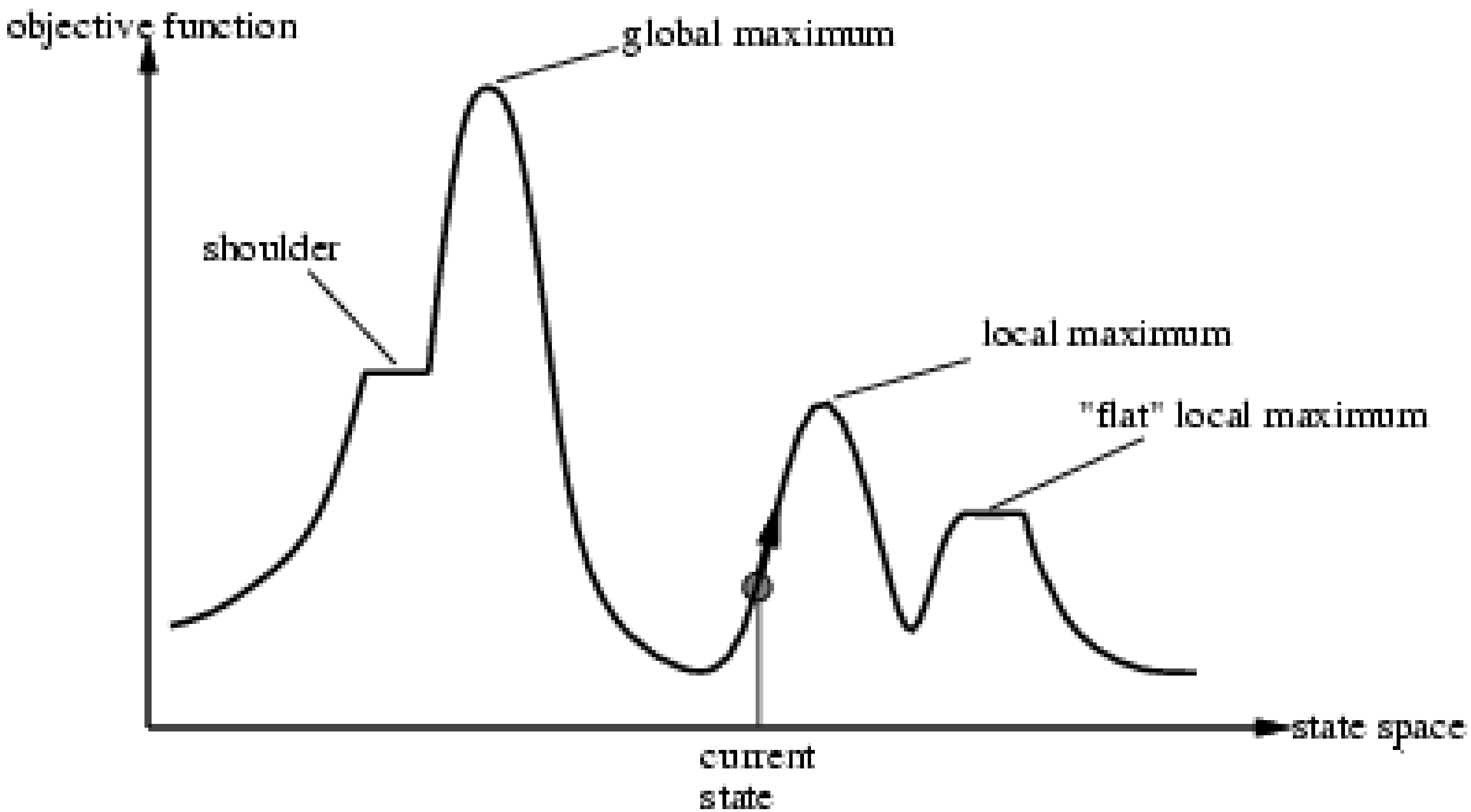


- Different algorithms can be used
  - Local search

Local search and optimization

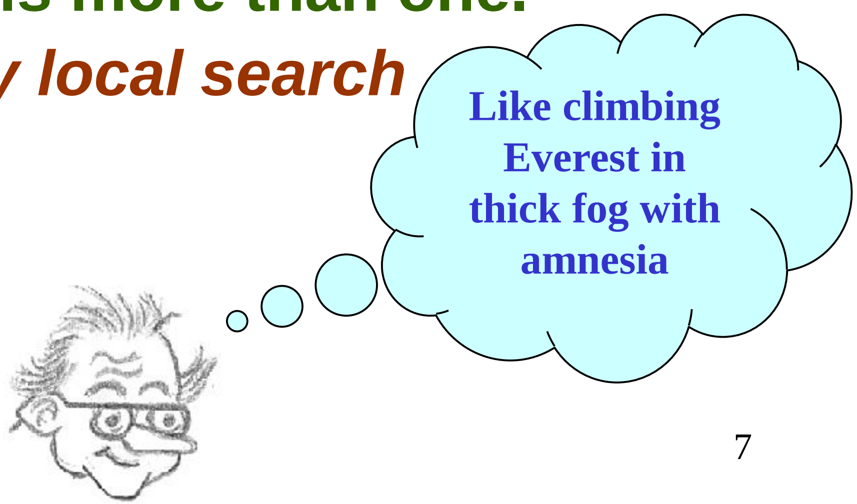
- **Local search = use single current state and move to neighboring states.**
- **Advantages:**
  - **Use very little memory**
  - **Find often reasonable solutions in large or infinite state spaces.**
- **Are also useful for pure optimization problems.**
  - **Find best state according to some *objective function*.**
  - **e.g. survival of the fittest as a metaphor for optimization.**

Local search and optimization



Hill-climbing search

- Hill-Climbing Search is a loop that continuously moves in the direction of increasing value
  - It terminates when a peak is reached.
- Hill climbing does not look ahead of the immediate neighbors of the current state.
- Hill-climbing chooses randomly among the set of best successors, if there is more than one.
- Hill-climbing a.k.a. *greedy local search*



Hill-climbing search

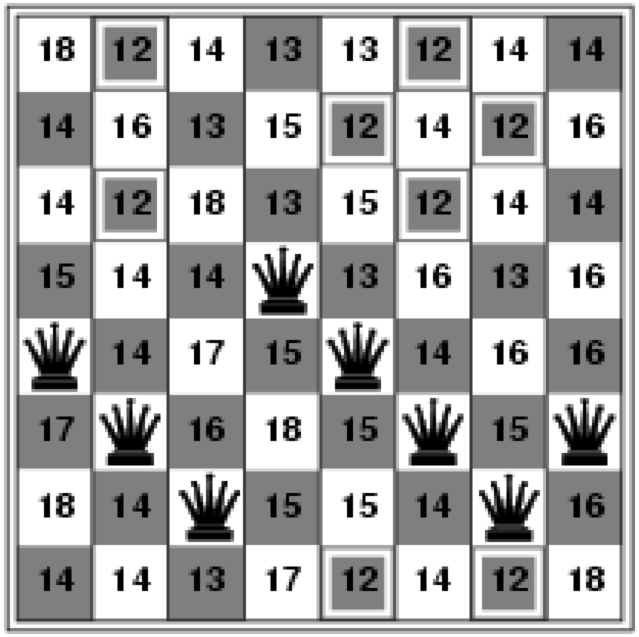
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
      neighbor ← a highest-valued successor of current
      if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
      current ← neighbor
```

Hill-climbing example

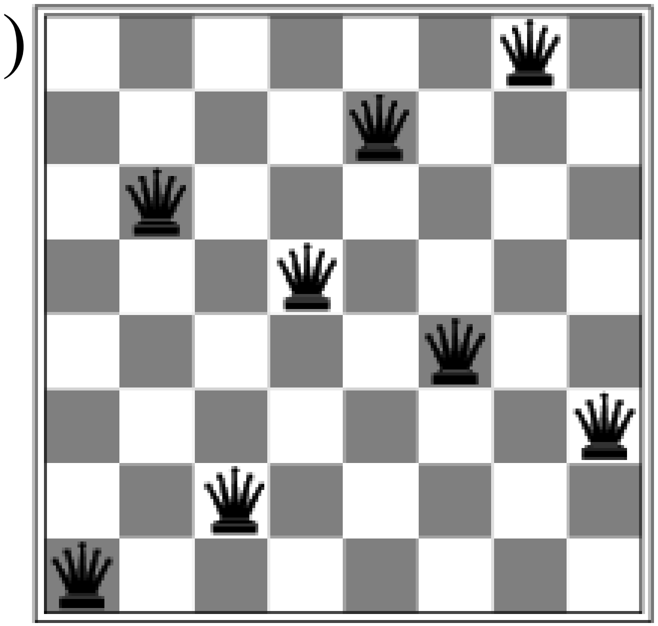
- 8-queens problem (complete-state formulation).
- Successor function: move a single queen to another square in the same column.
- Heuristic function *h(n)*: the number of pairs of queens that are attacking each other (directly or indirectly).

Hill-climbing example

a)



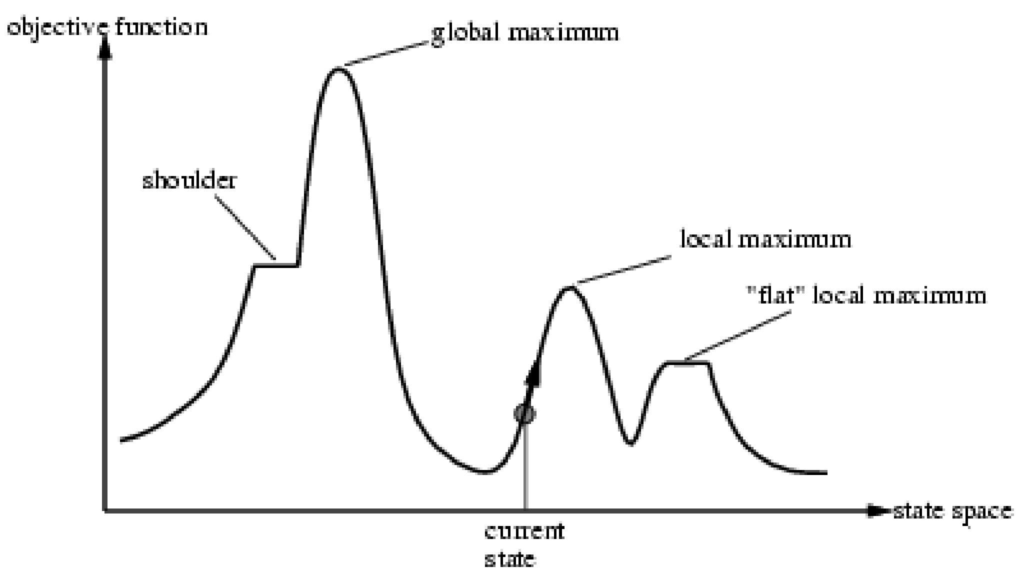
b)

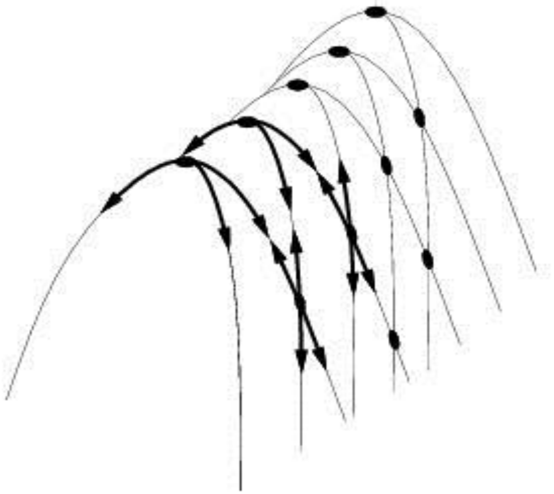


a) shows a state of h=17 and the h-value for each possible successor.

b) A local minimum in the 8-queens state space (h=1).

Drawbacks





- **Ridge = sequence of local maxima difficult for greedy algorithms to navigate**
- **Plateaux = an area of the state space where the evaluation function is flat.**
- **Gets stuck 86% of the time.**

Hill-climbing variations

- Stochastic hill-climbing
  - Random selection among the uphill moves.
  - The selection probability can vary with the steepness of the uphill move.
- First-choice hill-climbing
  - Stochastic hill climbing by generating successors randomly until a better one is found.
- Random-restart hill-climbing
  - Tries to avoid getting stuck in local maxima.

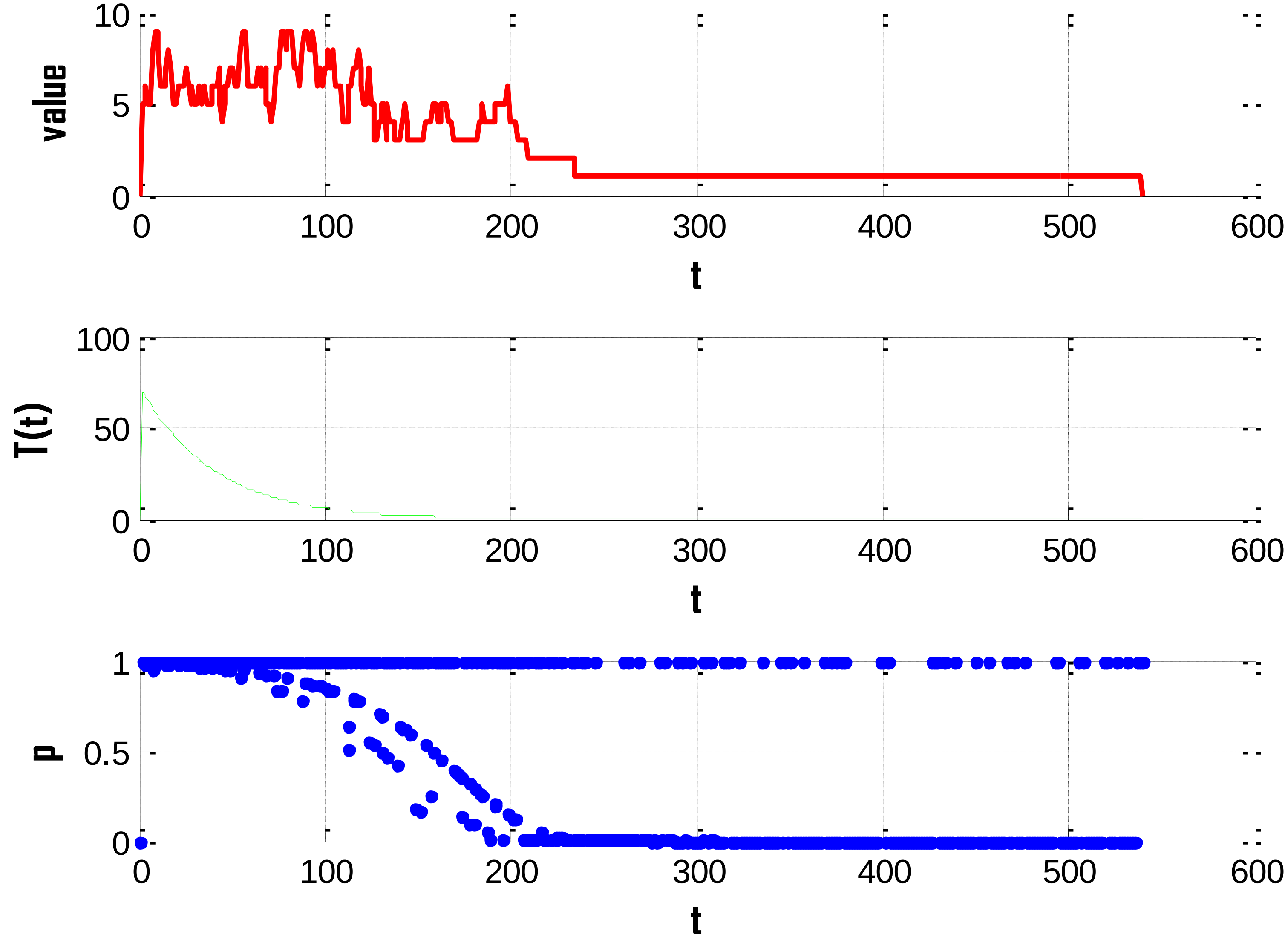
Simulated annealing

- Escape local maxima by allowing *“bad”* moves.
  - Idea: but gradually decrease their size and frequency.
- Origin: Metallurgical Annealing
- Bouncing ball analogy
  - Shaking hard (= high temperature).
  - Shaking less (= lower the temperature).
- If T decreases slowly enough, best state is reached.
- Applied for VLSI layout, airline scheduling, etc.

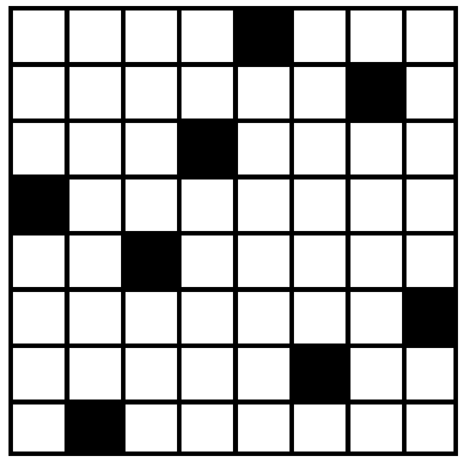
Simulated annealing

```
function SIMULATED-ANNEALING(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 ← 1 to ∞ do
        T ← schedule[t]
        if T = 0 then return current
        next ← a randomly selected successor of current
        ΔE ← VALUE[next] – VALUE[current]
        if ΔE > 0 then current ← next
        else current ← next only with probability e^(ΔE/T)
```

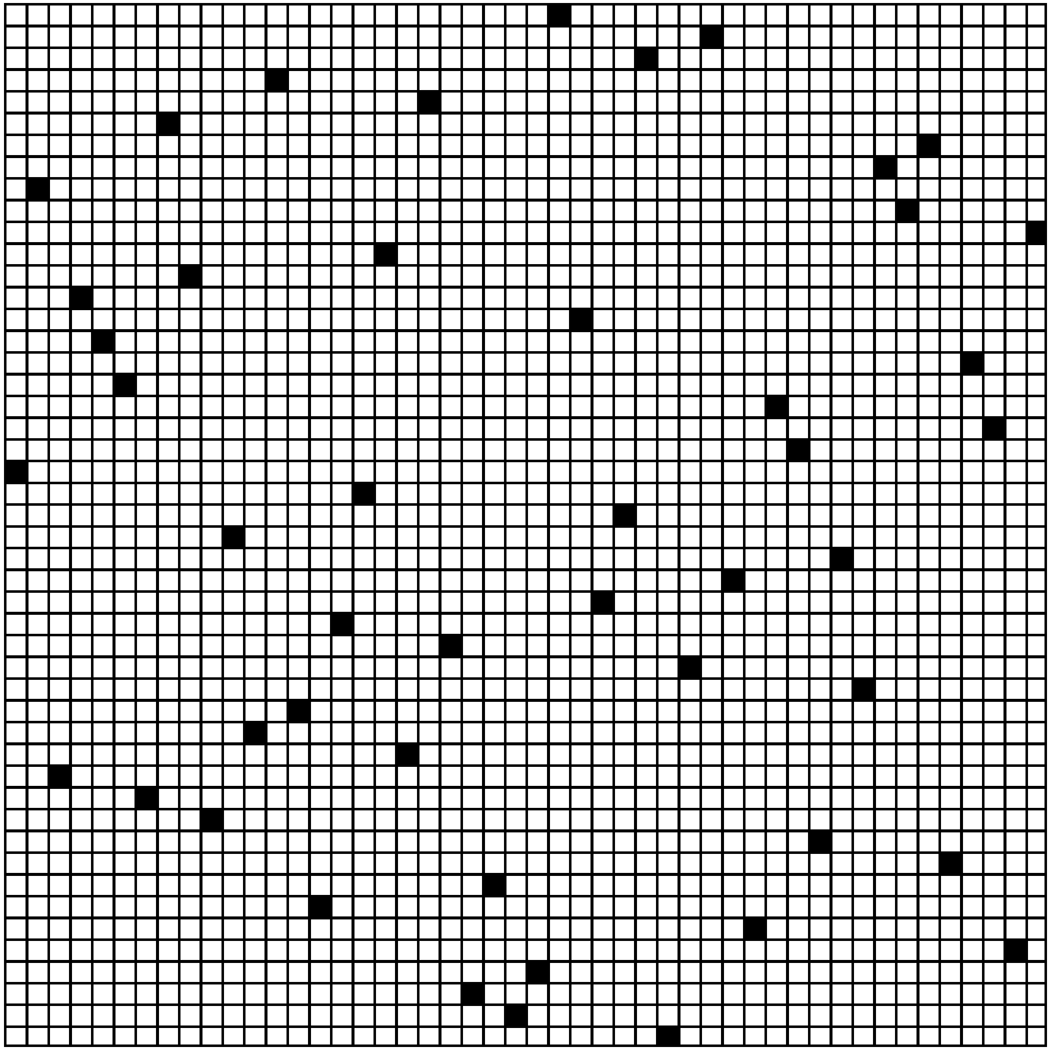
Solution of 8 queens Puzzle using Simulated Annealing



Solution of the 8 Queens Puzzle using SA



Solution to 48 Queens Problem using Simulated Annealing



Local beam search

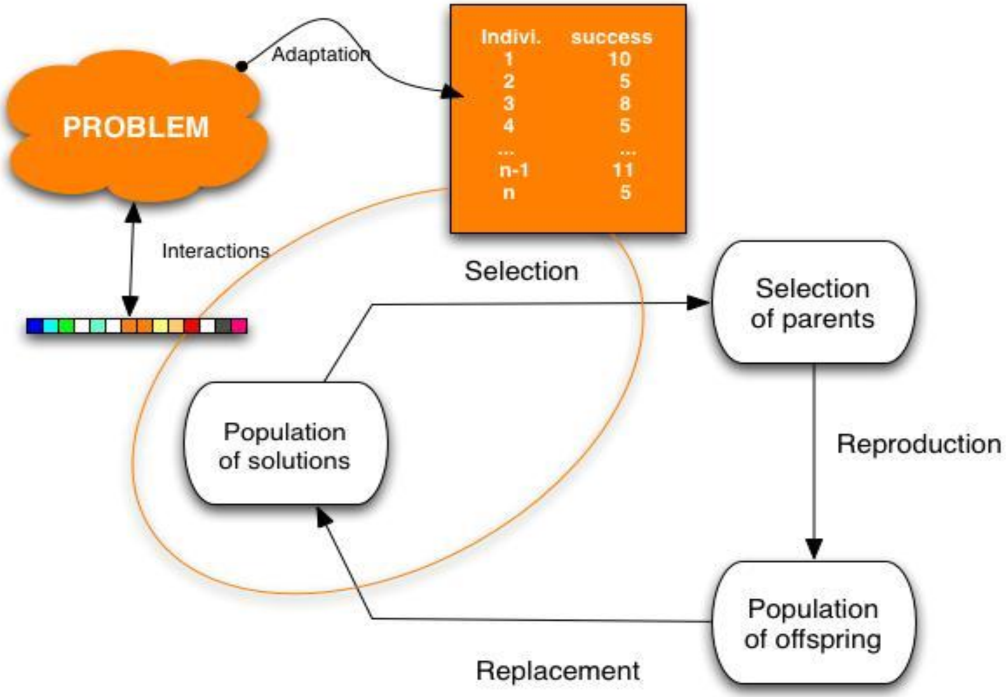
- **Keep track of *k* states instead of one**
  - **Initially: *k* random states**
  - **Next: determine all successors of *k* states**
  - **If any of successors is goal → finished**
  - **Else select *k* best from successors and repeat.**
- **Major difference with random-restart search**
  - **Information is shared among *k* search threads.**
- **Can suffer from lack of diversity.**
  - **Stochastic variant: choose k successors at proportionally to state success.**

Local Search in Continuous Spaces

- Use methods such as Gauss Newton Method which are based on gradient calculation
- Linear Programming

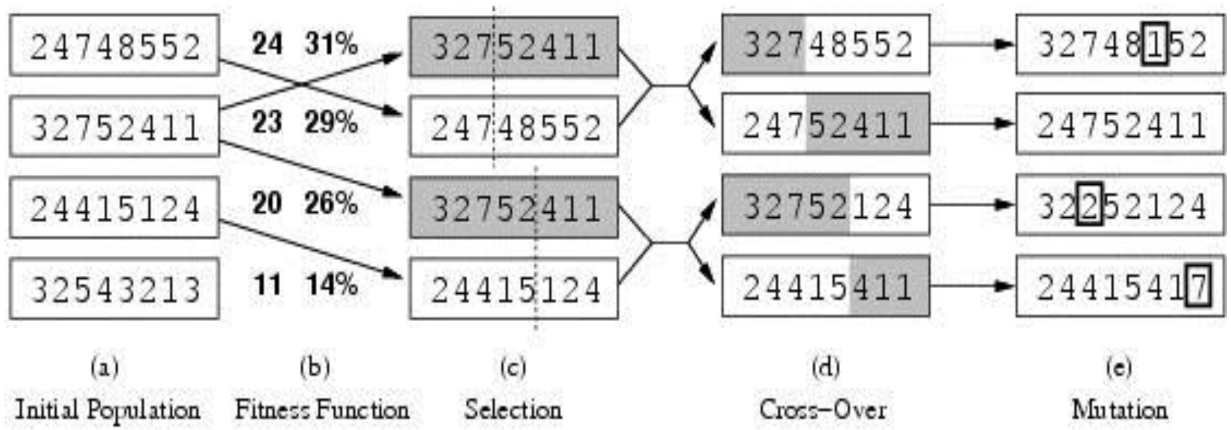
Genetic algorithms

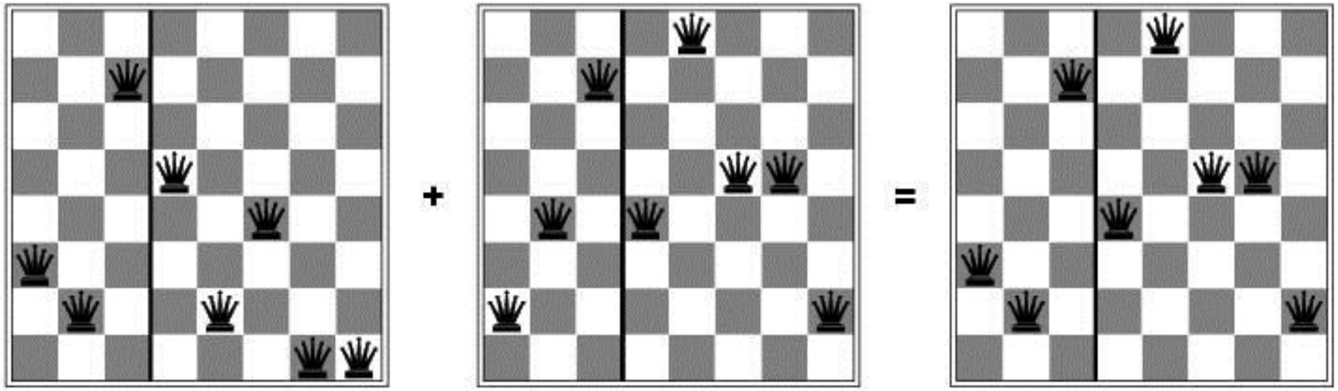
- **Variant of local beam search with *genetic recombination.***



Genetic algorithms

- Variant of local beam search with *genetic recombination.*





Genetic algorithm

function **GENETIC_ALGORITHM(** ***population,*** **FITNESS-FN)** return **an individual**

input: ***population*****, a set of individuals**

FITNESS-FN, a function which determines the quality of the individual

repeat

new_population ← **empty set**

loop for **i** from **1** to **SIZE(*population*)** do

x ← **RANDOM_SELECTION(*population*, FITNESS_FN)**
y ← **RANDOM_SELECTION(*population*, FITNESS_FN)**

child ← **REPRODUCE(*x,y*)**

if **(small random probability)** then ***child*** ← **MUTATE(*child*)**

add *child* to *new_population*

population ← ***new_population***

until **some individual is fit enough or enough time has elapsed**

return **the best individual**

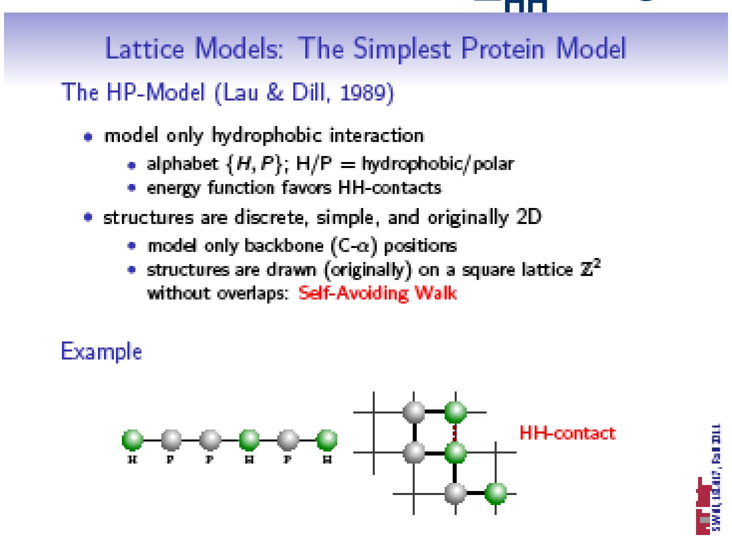
Assignment

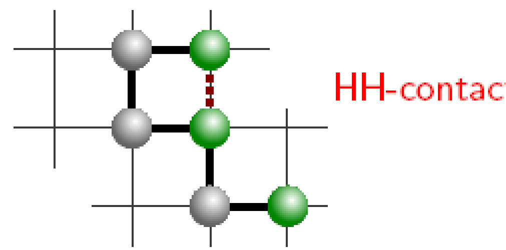
- Consider a lattice model of a protein with only two types of amino acids (H or P)
  - Hydrophobic
  - Polar
- The Energy Function is:

$$H = \sum_{i<j} E_{p_i p_j} \delta(r_i - r_j)$$

  - $\delta(r_i - r_j) = 1$ if monomers I and j are adjacent non-bonded nearest neighbors and 0 otherwise

- **Part-A: Energy Calculations**
  - **Given a configuration of a given sequence on an infinite lattice grid, calculate its energy**
    - **No Over-laps : Self Avoiding Walk**
    - **Energy Values**
      - **$E_{HH} = -3$**





- **Part-A (Re-Design)**
  - **Find the optimal “structure” of the sequence**
    - *H-H-P-P-H-H-P-H-H-P-P-H-H-P-H-H-P-P-H-H*

Up Next

- Simplification of the task
  - Lattice proteins
- 8-Queens using Monte-Carlo and Simulated Annealing (assignment)
- Energy Calculations on the computer for a given protein structure using pyRosetta
  - Exercise-1: Launch PyMOL and link to PyRosetta, make changes
  - Follow the manual
- Conceptutally follow Baker's talk
- Map of Pakistan?
- Energy optimization
- Sequence based search

- Get the lowest energy state of unbound proteins
- Predict ddG(Tanja Kortemme)

- Modulating Calmodulin Binding Specificity through Computational Protein Design

- Designing Dengue NS5 Binding

- Kortemme, Tanja, David E. Kim, and David Baker. “Computational Alanine Scanning of Protein-Protein Interfaces.” *Science Signaling* 2004, no. 219 (February 10, 2004): pl2. doi:10.1126/stke.2192004pl2.
- Kortemme, Tanja, Lukasz A. Joachimiak, Alex N. Bullock, Aaron D. Schuler, Barry L. Stoddard, and David Baker. “Computational Redesign of Protein-Protein Interaction Specificity.” *Nature Structural & Molecular Biology* 11, no. 4 (April 2004): 371–79. doi:10.1038/nsmb749.
- Kortemme, Tanja, and David Baker. “Computational Design of Protein–protein Interactions.” *Current Opinion in Chemical Biology* 8, no. 1 (February 2004): 91–97. doi:10.1016/j.cbpa.2003.12.008.

- The law of heredity is that all undesirable traits come from the other parent.