Meta-Heuristics

1

1

Meta-Heuristic

- A meta-heuristic designates a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality.

2

2

Why Heuristic Algorithms?

- It is hard to collect exact data in real-world problems.
- It impose fewer mathematical requirements
- It is easy to program a heuristic algorithm.
- It can be adopted to solve various problems.
- They can run in parallel. The quality of solutions may be improved time by time.
- Acceptable solution can be obtained in reasonable time.
- Many real-world problems are classified as NP-hard problems.
- “No free lunch Theorem ”

Adnan Shahzada

3

3

No Free Lunch Theorem

- If any algorithm **A** outperforms another algorithm **B** in the search for an extreme of a cost function, then algorithm **B** will outperform **A** over other cost functions.

4

4

Nature-Inspired Heuristic Algorithms

- Genetic Algorithm (evolution)
- Neural Network (artificial neural network)
- Simulated Annealing (metal annealing)
- Tabu Search (animal’s brain)
- Ant Colony Optimization
- Bee Colony Optimization

5

5

Local Search Algorithms (Local Search is a Meta- Heuristic)

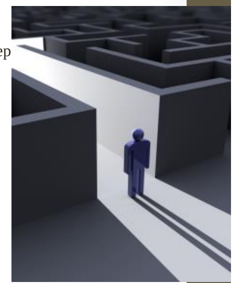
- Hill Climbing,
- Simulated Annealing,
- Tabu Search

Adnan Shahzada

6

6

Local Search Algorithms



- The search algorithms we have seen so far keep track of the current state, the "fringe" of the search space, and the *path* to the final state.
- In some problems, one doesn't care about a solution path but only the orientation of the final goal state
  - Example: 8-queen problem
- Local search algorithms operate on a *single* state – current state – and move to one of its neighboring states
  - Solution path needs not be maintained
  - Hence, the search is "local"

7

7

Local Search Algorithms

Example:

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

Initial state … Improve it … using local transformations (perturbations)

8

8

Local Search Algorithms

Basic idea: Local search algorithms operate on a *single* state – current state – and move to one of its neighboring states.

The principle: keep a single "current" state, try to improve it

Therefore: Solution path needs not be maintained.
Hence, the search is "local".

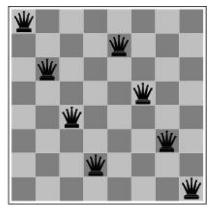
- Two advantages
  - Use little memory.
  - More applicable in searching large/infinite search space. They find reasonable solutions in this case.

9

9

Local Search Algorithms for optimization Problems

- Local search algorithms are very useful for optimization problems
- systematic search doesn't work
- however, can start with a suboptimal solution and improve it
- Goal: find a state such that the objective function is optimized



Minimize the number of attacks

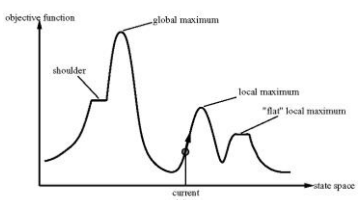
Adnan Shahzada

10

10

Local Search: State Space

A *state space landscape* is a graph of states associated with their costs

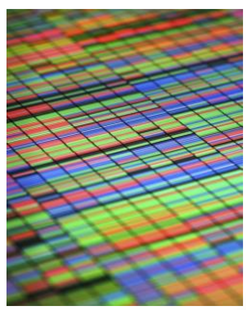


11

11

Local Search Algorithms

- Hill Climbing,
- Simulated Annealing,
- Tabu Search



12

12

Hill climbing

LOCAL SEARCH

13

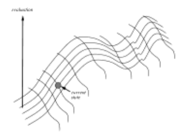
13

Hill Climbing

- **"Like climbing Everest in thick fog with amnesia"**

- Hill climbing search algorithm (also known as greedy local search) uses a loop that continually moves in the direction of increasing values (that is uphill).

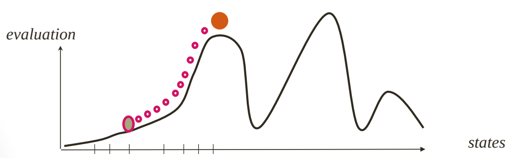
- It terminates when it reaches a peak where no neighbor has a higher value.



14

14

Hill Climbing



15

15

Hill Climbing

Steepest ascent version

function HILL-CLIMBING(*problem*) **returns** a solution state
  inputs: *problem*, a problem
  static: *current*, a node
    next, a node

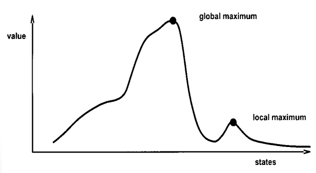
  current ← MAKE-NODE(INITIAL-STATE[*problem*])
  loop do
    next ← a highest-valued successor of *current*
    if VALUE[*next*] ≤ VALUE[*current*] **then return** *current*
    current ← *next*
  end

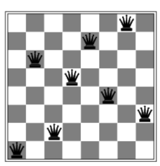
16

16

Hill Climbing Drawbacks

- Local maxima/minima : local search can get stuck on a local maximum/minimum and not find the optimal solution





Local minimum

- Cure
  + Random restart
  + Good for Only few local maxima

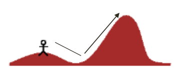
17

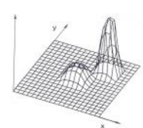
17

Issues

The Goal is to find GLOBAL optimum.

1. How to avoid LOCAL optima?
2. When to stop?
3. Climb downhill? When?





18

18

How to Avoid LOCAL Optima?

1. Local optima are points in the search space where the algorithm gets stuck because there are no neighboring states with better objective values. To avoid local optima, several strategies can be employed:
- **Random Restarts**: If the algorithm gets stuck, restart the search from a random state. Multiple restarts increase the chances of finding the global optimum.
- **Simulated Annealing**: Introduce a probability of accepting worse solutions initially, gradually decreasing over time. This allows the algorithm to escape local optima.
- **Tabu Search**: Maintain a list of previously visited states (tabu list) and prohibit the algorithm from revisiting them, thus preventing cycling back to local optima.
- **Stochastic Hill Climbing**: Introduce randomness in the selection of the next move. Instead of always choosing the best move, sometimes choose a random neighbor.

19

19

When to Stop?

Determining the stopping criteria is essential to ensure the algorithm terminates appropriately. Common stopping conditions include:
- **Fixed Number of Iterations**: Stop after a predetermined number of iterations or steps, regardless of whether the optimal solution has been found.
- **Convergence**: Stop if the algorithm has made little or no improvement over a set number of iterations. For example, if the objective value hasn't improved for the last n iterations.
- **Threshold Value**: Stop when the objective value reaches a predefined threshold, indicating a sufficiently good solution.
- **Time Limit**: Stop after a certain amount of time has elapsed, useful for applications where time is a constraint.

20

20

Climb Downhill? When?

In traditional hill climbing, climbing downhill is not part of the strategy, as the algorithm only considers moves that improve the objective value. However, there are situations where climbing downhill can be beneficial:
- **Escape from Local Optima**: Temporarily accept worse solutions (climb downhill) to escape local optima, a concept used in simulated annealing.
- **Plateaus**: When the algorithm encounters a plateau (a flat region with no better neighboring states), slight downhill moves might help find a path to better regions.
- **Noise**: In noisy environments, occasional downhill moves might help in smoothing out the noise and finding a better overall solution.

Summary:
- **Avoiding Local Optima**: Utilize strategies like random restarts, simulated annealing, tabu search, or stochastic hill climbing.
- **Stopping Criteria**: Use fixed iterations, convergence, threshold value, or time limits.
- **Climbing Downhill**: Employ when trying to escape local optima, dealing with plateaus, or in noisy environments, often in the context of techniques like simulated annealing.

21

21

SIMULATING ANEALING

- LOCAL SEARCH

22

22

Simulated Annealing

- Key Idea: escape local maxima by allowing some "bad" moves but gradually decrease their frequency
- Take some uphill steps to escape the local minimum
- Instead of picking the best move, it picks a random move
- If the move improves the situation, it is executed. Otherwise, move with some probability less than 1.
- Physical analogy with the annealing process:
  - Allowing liquid to gradually cool until it freezes
- The heuristic value is the energy, E
- Temperature parameter, T, controls speed of convergence

23

23

Simulated Annealing

- Basic inspiration: What is annealing?
- In metallurgy, annealing is the physical process used to temper or harden metals or glass by heating them to a high temperature and then gradually cooling them, thus allowing the material to coalesce into a low energy crystalline state.
  - Heating then slowly cooling a substance to obtain a strong crystalline structure.
- Key idea: Simulated Annealing combines Hill Climbing with a random walk in some way that yields both efficiency and completeness.
- Used to solve VLSI layout problems in the early 1980

24

24

Simulated Annealing

```
Start with the system in a known configuration, at known energy E
T = temperature = hot; frozen = false;
while (! frozen) {
        repeat {
                Perturb system slightly (e.g., move a particle)
                Compute .E , change in energy due to perturbation
                if (ΔE < 0)
                        then accept this perturbation, this is the new system config
                else accept maybe, with probability = exp(-ΔE/KT)
        } until (the system is in thermal equilibrium at this T)
        If (ΔE still decreasing over the last few temperatures)
        then T = 0.9 T // cool the temperature; do more perturbations
        else frozen = true
}
return (final configuration as low-energy solution)
```

25

25

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^(Δ E/T)
```

26

26

Simulated Annealing

- Temperature T
  - Used to determine the probability
  - High T : large changes
  - Low T : small changes
- Cooling Schedule
  - Determines rate at which the temperature T is lowered
  - Lowers T slowly enough, the algorithm will find a global optimum
- In the beginning, aggressive for searching alternatives, become conservative when time goes by

27

27

Tabu Search

- The basic concept of Tabu Search as described by Glover (1986)
- The overall approach is to avoid cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the solution space previously visited (hence "tabu").

28

28

Tabu Search Algorithm (simplified)

- 1. Start with an initial feasible solution
- 2. Initialize Tabu list
- 3. Generate a ***subset*** of neighborhood and find the best solution from the generated ones
- 4. If move is not in **tabu list** then accept it as the current best solution and also include it in the tabu list.
- 5. Repeat from 3 until **terminating condition**

29

29

Local Beam Search

- Unlike Hill Climbing, Local Beam Search keeps track of k states rather than just one.
- It starts with k randomly generated states.
- At each step, all the successors of all the states are generated.
- If any one is a goal, the algorithm halts, otherwise it selects the k best successors from the complete list and repeats.

30

30

Local Beam Search

- Idea: keep k states instead of just 1
  - Begins with k randomly generated states
  - At each step all the successors of all k states are generated.
  - If one is a goal, we stop, otherwise select k best successors from complete list and repeat

31

31

Population based Heuristics

32

32

Population Based Heuristics

- **Common characteristics: At every iteration search process considers a set (a population) of solutions instead of a single one**
- The performance of the algorithms depends on the way the solution population is manipulated
- Take inspiration from natural phenomena
- Two main approaches:
- Evolutionary Computation (Genetic Algorithms)
- Ant Colony Optimization

33

33

Ant Colony Optimization

- Ant Colony Optimization is based on Swarm Intelligence
- **Swarm intelligence (SI)** describes the collective behavior of decentralized and self organized systems, natural or artificial.
- ACO is proposed by Dorigo in 1992 as a PhD thesis

34

34

Ant Colony Optimization

- Fairly simple units generate complicated global behaviour.
- An ant colony expresses a complex collective behavior providing **intelligent solutions to problems** such as:
  - carrying large items
  - **finding the shortest routes from the nest to a food source**, prioritizing food sources based on their distance and ease of access.
- *"If we knew how an ant colony works, we might understand more about how all such systems work, from brains to ecosystems."*
  (Gordon, 1999)

35

35

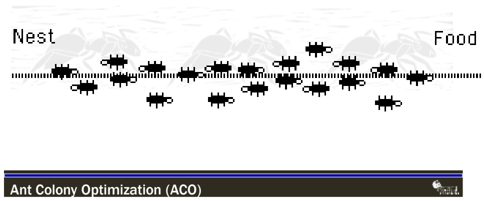
ACO

- Ants need to find the food and the shortest path to it
- They need to communicate this information to other ants
- Ants are able to let on the ground a chemical substance (pheromone) to mark trails.
- Ants are able to smell the pheromone
- Ants are both able to find new food sources and to go back to "known" sources to continue to bring back the food
- And if we have a simple model we can prove that ants will converge to the shortest path.

36

36

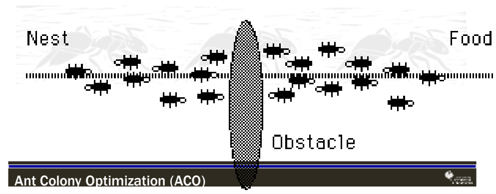
Naturally Observed Ant Behavior



All is well in the world of the ant.

37

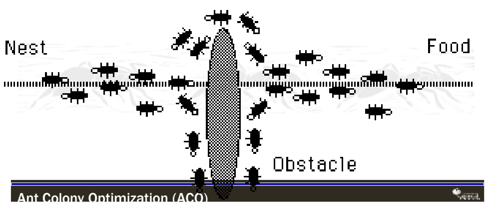
Naturally Observed Ant Behavior



Oh no! An obstacle has blocked our path!

38

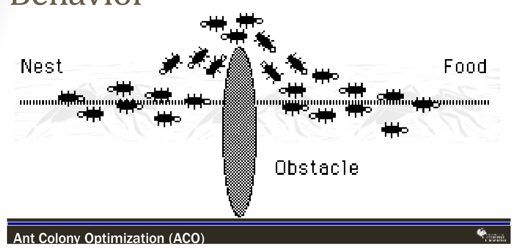
Naturally Observed Ant Behavior



Where do we go? Everybody, flip a coin.

39

Naturally Observed Ant Behavior



Shorter path reinforced.

40

Ant Colony Optimization

- ACO algorithms are multi-agent systems that exploit artificial **stigmergy** for the solution of optimization problems.
- **Stigmergy** is a mechanism of indirect coordination between agents or actions. The principle is that the trace left in the environment by an action stimulates the performance of a next action, by the same or a different agent.
- Artificial ants live in a discrete world. They construct solutions making stochastic transition from state to state.
- They deposit artificial pheromone to modify some aspects of their environment (search space). Pheromone is used to dynamically store past history of the colony.
- Artificial Ants are sometime "augmented" with extra capabilities like local optimization or backtracking

Adnan Shahzada

41

41

The Algorithm overview

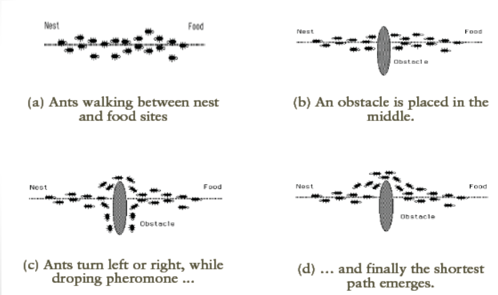
- Algorithm is well described on different sources. In short, the goal is to simulate the behavior of real ants
- Each ant will build a solution. For the TSP, where it has to travel through the set of cities. Once the solution is completed, the ant will "leave" pheromone on the tracks to strengthen the amount of pheromone on that track.
- Each ant will choose his path (i.e. next city) by a biased choice (path with more pheromone are more likely to be chosen)
- And at each iteration the pheromone is evaporating a little

Adnan Shahzada

42

42

Shortest Path Discovery

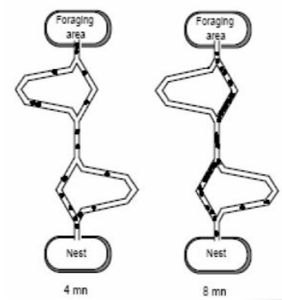


43

43

Shortest Path Discovery

- Ants get to find the shortest path after few minutes ...



44

44

Ant Colony Optimization

- Each artificial ant is a probabilistic mechanism that constructs a solution to the problem, using
- Artificial pheromone deposition
- Heuristic informatior pheromone trails, already visited cities memory ...

Algorithm 1 Ant colony optimization metaheuristic

```
Set parameters, initialize pheromone trails
while termination conditions not met do
    ConstructAntSolutions
    ApplyLocalSearch        {optional}
    UpdatePheromones
end while
```

45

45

Bee Colony Optimization

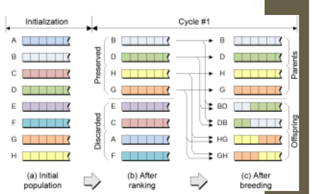
- New as first real publication was in 2005
- similar to ant colonies but some differences
- like one bee for each food source.
- Each employee bee has in memory one food source
- Communication is done by a dance

46

46

Genetic Algorithms

- They are based on the Darwin theory of evolution
- Individuals that better fit with the environment have more chance to survive
- Auto-organization as in the biological systems
- Evolution as natural selection mechanism
- Populations of individuals move from one generation to the next.



47

47

Genetic Algorithms

- Individual reproduction capabilities are "proportional" to their ability to fit with the environment
- Reproduction allows the best individual to generate children similar to them
- Generation after generation the population always fit better with the environment
- The environment is the objective function (**fitness) to optimize, and the individuals are a population of solutions.**

48

48

Genetic Algorithm

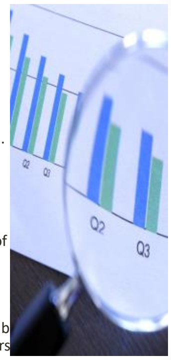
Introduced by Holland (1960) at UNI Michigan

It is a parallel search in the solution space,
where the search is driven by past experiences

- **Components:**
  - Individual is described by his chromosome.
  - Chromosome is defined by a set (sequence) of genes.
  - Population is a set of individuals.
  - Generations are defined by a sequence of different populations
  - Individuals are evaluated using a fitness function (to be optimized) that is their adaptation to the environment

49

Summary



- Local search methods keep small number of nodes in memory.
- They are suitable for problems where the solution is the goal state itself and not the path.
- *Hill climbing, simulated annealing and local beam search are examples of local search algorithms.
- Stochastic algorithms represent another class of methods for informed search.
- Genetic algorithms are a kind of stochastic hill-climbing search in which a large population of states is maintained. New states are generated b mutation and by crossover which combines pairs of states from the population.

50

50

Assignment !!

- Write a comparative analysis between the Genetic Algorithms and other meta-heuristic based search methods.

Adnan Shahzada

51

51