CS 235: Artificial Intelligence

Local Search Algorithm Part-I

Dr. Moumita Roy
CSE Dept., IIITG

Local Search and Optimization

- Previous searches: keep paths in memory, and remember alternatives. Solution is a path to a goal.
- Path may be irrelevant, if only the final configuration is needed (eg. 8-queens)

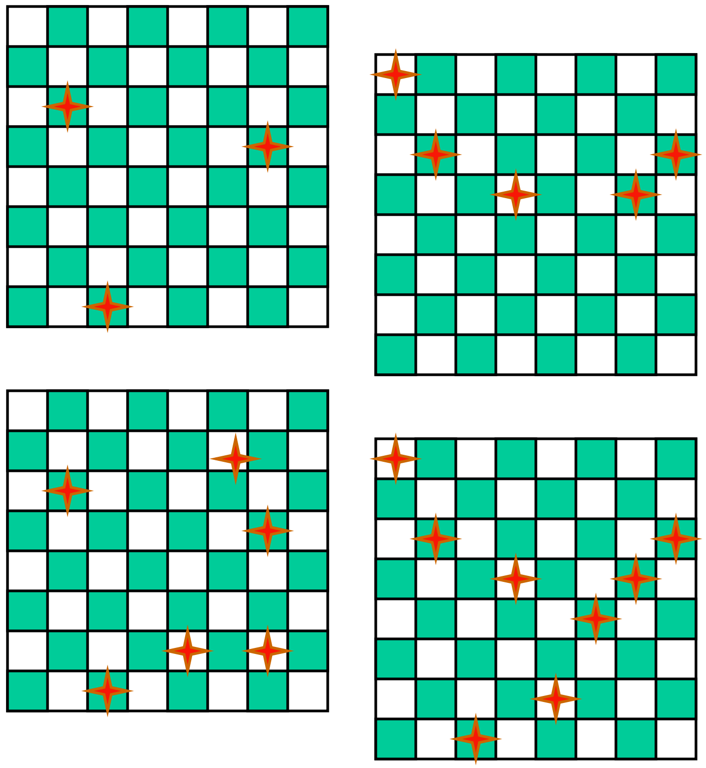
Local Search

- Use a single current state and move only to neighbors.
- Use little space
- Can find reasonable solutions in large or infinite (continuous) state spaces for which the other strategies are not suitable

Optimization

- Local search is often suitable for optimization problems.
- Search for best state by optimizing an objective function F(S).
- F(S) where often S is a vector of continuous or discrete values
- Begin with a complete configuration of a state (a possible solution)
- Local search typically use complete-state formulation (not incremental formulation where actions augment the state description)
- Move from the current state to a successor state
- Low memory requirements, because the search tree or graph is not maintained in memory (paths are not saved)

8-queens problem (incremental formulation)



- **States:** all arrangements of 0, 1, 2, ..., 8 queens on the board
- **Initial state:** 0 queens on the board
- **Successor function:** each of the successors is obtained by adding one queen in an empty square
- **Arc cost:** irrelevant
- **Goal test:** 8 queens are on the board, with no queens attacking each other

→ ~ 64x63x...x57 ~ 3×10^{14} states

Observation:

1. Final configuration matters, not the order in which they have added.
2. Path to goal is irrelevant.

Complete-state formulation for 8-queen problem

- 8 queens: find an arrangement of 8 queens on a chess board such that no two queens are attacking each other
- **State:** some arrangement of the 8-queens on board, one per column (typically represented by vector)
- **Initial state:** Start with some arrangement of the queens, one per column

  $S^0=[1\ 4\ 6\ 7\ 1\ 2\ 5\ 7]$
- ***Goal State/test***: an arrangement of 8 queens on a chess board such that no two queens are attacking each other

  (mostly define in terms of the objective function for the problem)
- **Successor function:** Successor of a state is obtained by moving a single queen to another square in the same column (8*7=56 successors)
- **Objective function(F)**: number of pairs attacking each other

Travelling Salesman Problem (TSP)

- Traveling salesman problem: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city

- Start with some ordering of the cities (5 cities A, B, C, D,E)

- State representation – order of the cities visited
  (for example S^0=[A C B E D A])

- Successor state: a change to the current ordering

- F: length of the route

Comparison to earlier search framework

- start state is typically not a complete configuration.

  All states are typically a complete configuration (a possible solution)
- Binary goal test

  No binary goal test, unless one can define one in terms of the objective function for the problem (e.g., no attacking pairs in 8-queens)
- h: estimate of the distance to the nearest goal

  objective function: preference/quality measure – how good is this state?
- saving paths

  start with a complete configuration and make modifications to improve it

Hillclimbing (Greedy Local Search)

- Generate nearby successor states to the current state
- Pick the best and replace the current state with that one.
- Loop

Algorithm:

```
Step 1: Start with initial state s
Step 2: Pick a successor state t of s with largest F(…) value
Step 3: If F(t)<F(s) then stop, return s
Step 4: else s=t Goto Step 2.
```

Visualization of state space for local search technique

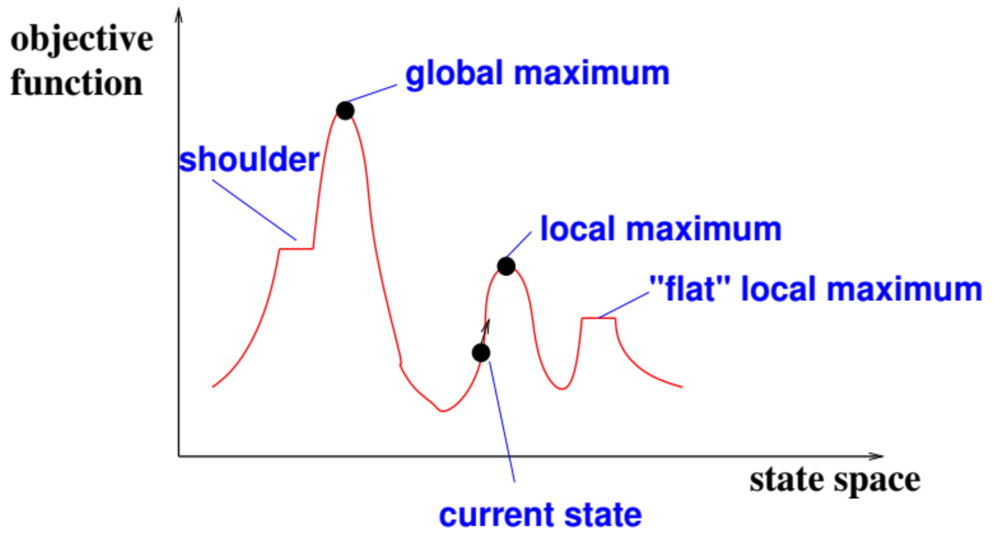
- States are laid out in a landscape
- Height corresponds to the objective function value
- Move around the landscape to find the highest (or lowest) peak
- Only keep track of the current states and immediate neighbors

Hill-climbing search problems

(this slide assumes maximization rather than minimization)

- *Local maximum*: a peak that is lower than the highest peak, so a suboptimal solution is returned (greedy in nature, easily stuck in local maximum)

- *Plateau*: flat area in state space. It may be flat local maximum (no uphill possible) or sometime progress is possible.



Random restart hill-climbing (note: there are many variants of hill climbing)

- Start different hill-climbing searches from random starting states stopping when a goal is found or K number of times
- Save the best result from any search so far
- Finding an optimal solution becomes the question of sufficient number of restarts
- Surprisingly effective, if there aren't too many local maxima

Variants of hill climbing algorithm

- **First-choice hill climbing**: Generate a successor at a time; if better move or else generate next. The good strategy when state has many successors.

- **Steepest hill climbing:** Choose best successor if better move next. It may easily stuck into local maximum.

- **Stochastic hill climbing:**
  - randomly select among better successors (uphill moves)
  - Probability of selection can vary with steepness of uphill move
  - $\Delta E = F(t) - F(s)$, t->next state, s->current state

  No downhill movement is allowed.

Simulated Annealing

- Annealing: harden metals and glass by heating them to a high temperature and then gradually cooling them
- In this case, the idea of annealing process is used.
- At the start, make lots of moves (downhill moves also) and then gradually slow down

Simulated annealing

- If the move improves the situation, it is always accepted.
- Otherwise, the algorithm accepts the move with some probability.
- Probability of a move decreases exponentially with the amount ΔE
- A second parameter *T (temperature)* is also used to determine the probability: The probability also decreases as the T goes down.
- High *T* allows more worse moves, *T* close to zero results in few or no bad moves
- *Schedule* input determines the value of *T* as a function of the completed cycles

Simulated annealing

```
function Simulated-Annealing(start,schedule)
    current ← start
    for $t$ ← 1 to ∞ do
        $T$ ← schedule[$t$]

        if $T$=0 then return current

        next ← a randomly selected successor of current
        $\Delta E$ ← F[next] – F[current]

        if  $\Delta E > 0$ then current ← next
        else current  ← next only with probability $e^{\Delta E/T}$
```

Intuitions

- Hill-climbing is incomplete
- Pure random walk, keeping track of the best state found so far, is complete but very inefficient
- Combine the ideas: add some randomness to hill-climbing to allow the possibility of escape from a local optimum
- the algorithm wanders around during the early parts of the search, hopefully toward a good general region of the state space
- Toward the end, the algorithm does a more focused search, making few bad moves

Theoretical Completeness

- There is a proof that if the schedule lowers T slowly enough, simulated annealing will find a global optimum with probability approaching 1
- In practice, that may be way too many iterations
- In practice, though, SA can be effective at finding good solutions

Local Beam Search

- Keep track of k states rather than just one, as in hill climbing
- K is the beam size.

Local Beam Search

- Begins with k randomly generated states
- At each step, all successors of all k states are generated
- If any one is a goal, algorithm halts
- Otherwise, selects best k successors from the complete list, and repeats

Local Beam Search

- Successors can become concentrated in a small part of state space if less diversity in k successors (like hill climbing with high cost)
- Stochastic beam search: choose k successors at random with a probability based on their goodness
- Like natural selection: successors (offspring) of a state (organism) populate the next generation according to its value (fitness)

Complete state-formulation (Clustering)

Formulate the following problem for intelligent agent:

Divide the N students in k groups based on their marks in AI such as diversity in each group is minimized.

For demonstration, you may use:

N=5, k=2, set of marks={50, 20,10, 5, 15, 45} (out of 60)

- State representation:
- Initial state:
- Goal state:
- Successor function:
- Objective function:

Complete state-formulation (Version 1) (Clustering)

Set of students/marks $A=\{a_1, a_2, a_3,\ldots\ldots,a_N\}= =\{50, 20,10, 5, 15, 45\}$

- State representation:

  $S=[C_1\ C_2\ C_3\ \ldots\ldots\ C_N]$

  if a_j belongs to ith group, then $C_j=i$, where $i\in\{1, 2, 3,\ldots.,k\}$

- Initial state: (example for demonstration)

  $S^0=[1\ 2\ 2\ 2\ 2\ 1]$ → better representation is needed

  $F(S^0)=(50-45)^2 + [(20-10)^2+(20-5)^2+(20-15)^2+(10-5)^2+(10-15)^2+(5-15)^2]$

- Goal state/test:

  No binary goal state

  report the state when we have the minimum value of objective function

- Successor function: Successor is generated by changing the group of one student at a time
- Objective function: summation of diversity in each cluster

Complete state-formulation (Version 2) (Clustering)

Set of students/marks $A=\{a_1, a_2, a_3,\ldots\ldots,a_N\}= =\{50, 20,10, 5, 15, 45\}$

- State representation:

  $S=[C_1\ C_2\ C_3\ \ldots\ldots\ C_k]$

  if C_j is representative of the jth group, then $C_j \in \{a_1, a_2, a_3,\ldots,a_N\}$

- Initial state: (example for demonstration)

  $S^0=[50\ 20]$ (assigned in a group with nearest representative; then same as [1 2 2 2 2 1])

  $F(S^0)=[(50-50)^2+(50-45)^2]+[(20-20)^2+(20-10)^2+(20-5)^2+(20-15)^2]$

$F(S^0)=(50-45)^2 + [(20-10)^2+(20-5)^2+(20-15)^2+(10-5)^2+(10-15)^2+(5-15)^2]$

- Goal state/test:

  No binary goal state

  report the state when we have the minimum value of objective function

- Successor function: Successor is generated by changing a group representative at a time
- Objective function: summation of diversity in each cluster

Observation

- We need some strategy to generate successors

- State space may be continuous or discrete

Genetic Algorithms

- Variant of stochastic beam search
- Combine two parent states to generate successors

Natural Evolution Process

- GA mimics the natural evolution process.
- **Evolution:** Evolution is the process by which modern organisms have descended from ancient ones
- **Microevolution** is evolution within a single population; (a population is a group of organisms that share the same gene pool).
- Components:

➢ Heredity

Information needs to be passed on from one generation to the next

➢ Genetic Variation

There has to be differences in the characteristics of individuals in order for change to occur (mutation)

➢ Differential Reproduction

Some individuals need to (get to) reproduce more than others thereby increasing the frequency of their genes in the next generation; often the term fitness is used to describe the relative ability of individuals to pass on their genes

Start with MAXONE problem

- Formulate the problem for Intelligent agent:

  Task is to maximize the number of 1's in a string of L binary digits.

- Is it a very simple one? We know the answer beforehand.

- However, we can think of it as maximizing the number of correct answers, each encoded by 1, to L yes/no difficult questions.

Complete state-formulation (MAXONE)

- State representation:
  $S=[C_1\ C_2\ C_3\ \ldots\ldots\ C_L]$
  $C_j\ = 0$ or 1

- Initial state: (example for demonstration; L=6)
  $S^0=[1\ 0\ 1\ 0\ 1\ 0]$ / Encoding
  $F(S^0)=3$

- Goal state/test:
  report the state when we have the maximum value of objective function

- Successor function: Successor is generated by changing one element at a time

- Objective function: number of 1's in a string (Fitness values)

Basic steps of GA

Step 1: **Encoding** of the problem (state representation)

Step 2: Random generation of Initial population (set of initial states)

Step 3: Compute fitness of each individual of the current population (objective function of each state/a possible solution)

Step 4: Select the pairs of parents from the current population based on fitness value **(Selection)**

Step 5: Generate the offspring (successor) using **Crossover and Mutation**

Step 6: Repeat step 3 to 6 until satisfying solution is reached or convergence is reached

Encoding/Selection/Crossover/ Mutation

Solve the MAXONE problem using GA

Encoding

- State/Solution is represented by Chromosome
- Each element of state is represented by gene (here each 0/1)

0	0	1	0	1	1	1	0	0	1

- **Genes** are joined into a string to form Chromosome (solution)
- Representation should easy to apply mutation and crossover

➢ **Binary representation**: same as above (binary string)

➢ **Real valued representation**: For problems where we want to define the genes using continuous rather than discrete variables, the real valued representation is the most natural.

0.5	0.2	0.6	0.8	0.7	0.4	0.3	0.2	0.1	0.9

➢ **Integer Representation**: For discrete valued genes, we cannot always limit the solution space to binary 'yes' or 'no'. For example, if we want to encode the clustering with 4 groups({1,2,3,4}). In such cases, integer representation is desirable.

1	2	3	4	3	2	4	1	2	1

➢ **Character Representation:** also possible

➢ **Permutation Representation:** In many problems, the solution is represented by an order of elements. In such cases permutation representation is the most suited. A classic example of this representation is the travelling salesman problem .

MAXONE Problem using GA

Step 1: **Encoding** of the problem (state representation)

Here, we use binary representation

Step 2: Random generation of Initial population (set of initial states)

Step 3: Compute fitness of each individual of the current population (objective function of each state/a possible solution)

Step 4: Select the pairs of parents from the current population based on fitness value (Selection)

Random initialization (Population size: 4/L=6)

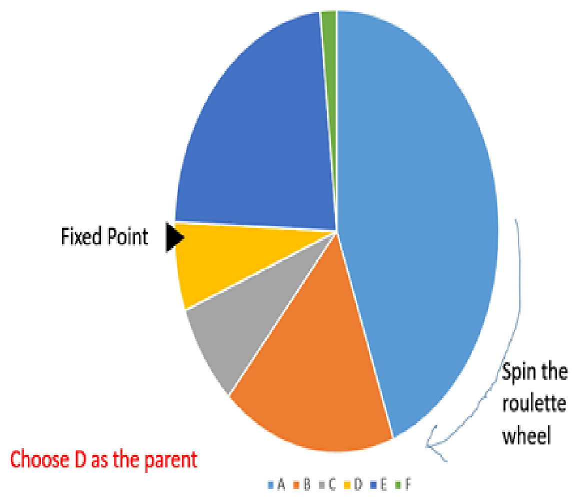
Initial Population	Fitness value
(Chromosome 1) [0 1 0 1 1 1]	4
(Chromosome 2) [1 1 0 1 1 1]	5
(Chromosome 3) [1 0 1 0 1 0]	3
(Chromosome 4) [1 0 0 0 0 1]	2

Selection Operator (survival of the fittest)

- In this case, every individual can become a parent with a probability which is proportional to its fitness.

➢ **Roulette Wheel Selection:**

1. Consider a circular wheel.
2. The wheel is divided into n pies, where n is the number of individuals in the population.
3. Each individual gets a portion of the circle which is proportional to its fitness value.
4. A fixed point is chosen on the wheel circumference as shown and the wheel is rotated.
5. The region of the wheel which comes in front of the fixed point is chosen as the parent.
6. For the second parent, the same process is repeated.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Problem: If initial population contains one or two very fit but not best individual and the rest of the members are not so good, then these fit individuals dominate others; prevent to explore the population. (premature/fast convergence)

Selection Operator

- Rank selection:

1. every individual in the population is ranked according to their fitness.
2. The selection of the parents depends on the rank of each individual and not the fitness.
3. The higher ranked individuals are preferred more than the lower ranked ones.

Initial Population	Fitness value	Portion in wheel	Rank	Portion in wheel
A	40	80%	4	40%
B	5	10%	3	30%
C	3	6%	2	20%
D	2	4%	1	10%
Fitness of the population	50		10	

Problem: Slower convergence/ Can't differ best chromosome from others

Selection Operator

- **Tournament selection:**

1. In k-way tournament selection, select k individuals from the population at random and select the best out of these to become a parent.
2. The same process is repeated to select the next parent.

- **Random selection:** We randomly select parents from the existing population. There is no selection pressure towards fitter individuals and therefore this strategy is usually avoided.

- **Elitism:**

1. Copy the best chromosome (solution) to the new population
2. Intuition: creating a new population (set of successors) using crossover and mutation; may lost the best one.
3. Retain some number of best individual at each generation.

MAXONE Problem using GA

Step 1: **Encoding** of the problem (state representation)

Here, we use binary representation

Step 2: Random generation of Initial population (set of initial states)

Step 3: Compute fitness of each individual of the current population (objective function of each state/a possible solution)

Step 4: Select the pairs of parents from the current population based on fitness value (Selection)

Random initialization (Population size: 4/L=6)

Initial Population	Fitness value
(Chromosome 1) [0 1 0 1 1 1]	4
(Chromosome 2) [1 1 0 1 1 1]	5
(Chromosome 3) [1 0 1 0 1 0]	3
(Chromosome 4) [1 0 0 0 0 1]	2

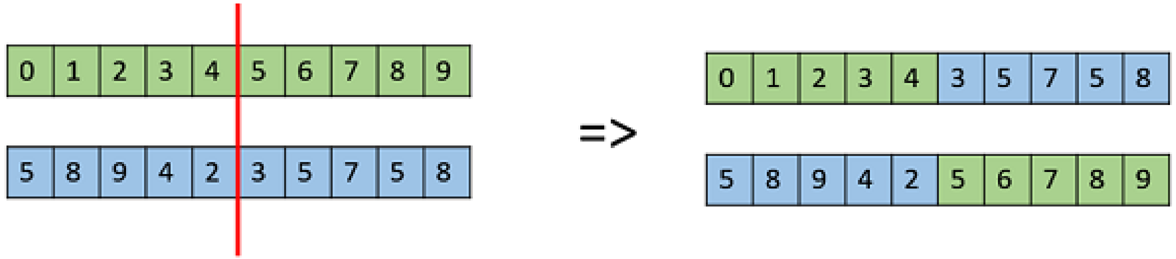
Select Chromosome 1 as parent 1 and Chromosome 3 as parent 2

Step 5: Generate the offspring (successor) using **Crossover and Mutation**

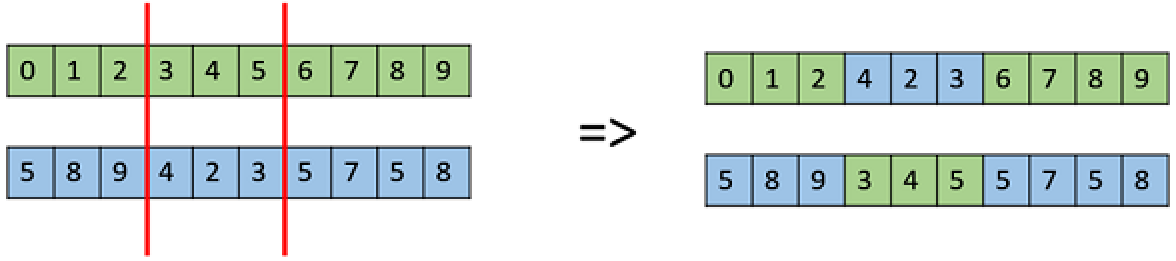
Generation of successors (crossover operators)

- We need to generate the same number of successor as the population size.
- It means we need to repeat the step 4 (selection) until the desired number of successors are generated.

➢ **Single-point crossover**: In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.

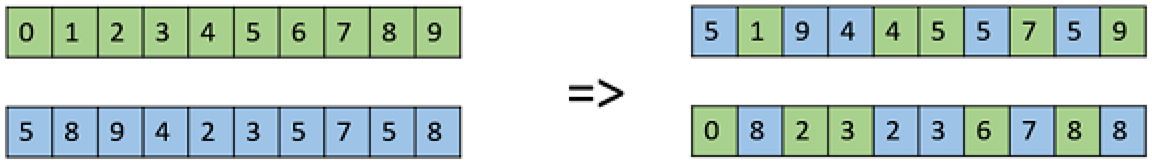


➢ **Multi Point Crossover:** Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.



Generation of successors (crossover operators)

- Uniform Crossover
- In a uniform crossover, we don't divide the chromosome into segments, rather we treat each gene separately.
- In this scheme, at each bit position (gene) of the parent, we toss a coin to determine whether there will be swap of the bits or not.
- Each gene is chosen from either parent with equal probability

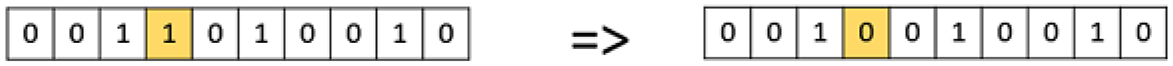


- Crossover is performed on two parents to form two new offspring.
- The GA has a crossover probability (in the range of [0, 1]) that determines if crossover will happen. A randomly generated floating-point value is compared to the crossover probability, and if it is less than the probability, crossover is performed; otherwise, the offspring are identical to the parents.
- If crossover is to occur, one or more crossover points are generated, which determines the position in the chromosomes where parents exchange genes. In this GA, once two parents are selected, two crossover points are randomly generated.

Mutation Operators

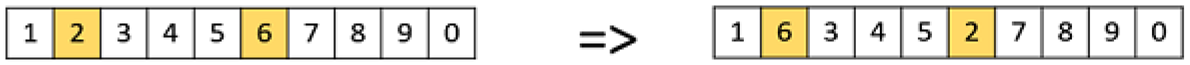
- Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next.

- **Bit Flip Mutation**: select one or more random bits and flip them. This is used for binary encoded GAs.



- **Random Resetting**: the extension of the bit flip for the integer representation. In this, a random value from the set of permissible values is assigned to a randomly chosen gene.

- **Swap Mutation**: select two positions on the chromosome at random, and interchange the values.



Mutation Probability

- After the offspring are generated from the selection and crossover, the offspring chromosomes may be mutated.
- Like crossover, there is a mutation probability (in the range of [0, 1]).
- If a randomly selected floating-point value is less than the mutation probability, mutation is performed on the offspring; otherwise, no mutation occurs.

Parameters in GA

- Population size: too few, search space not explored; too large slow down GA
- Crossover Probability/Crossover Points
- Mutation Probability

MAXONE Problem using GA

Step 1: **Encoding** of the problem (state representation)

Here, we use binary representation

Step 2: Random generation of Initial population (set of initial states)

Step 3: Compute fitness of each individual of the current population (objective function of each state/a possible solution)

Step 4: Select the pairs of parents from the current population based on fitness value (Selection)

Random initialization (Population size: 4/L=6)

Initial Population	Fitness value
(Chromosome 1) [0 1 0 1 1 1]	4
(Chromosome 2) [1 1 0 1 1 1]	5
(Chromosome 3) [1 0 1 0 1 0]	3
(Chromosome 4) [1 0 0 0 0 1]	2

Select Chromosome 1 as parent 1 and Chromosome 3 as parent 2

Step 5: Generate the offspring (successor) using **Crossover and Mutation**

MAXONE Problem using GA

Step 5: Generate the offspring (successor) using Crossover and Mutation

Crossover:

Selected Individuals	Successors (Crossover)
(Chromosome 1) [0 1 0 1 1 1]	[0 1 0 0 1 0]
(Chromosome 3) [1 0 1 0 1 0]	[1 0 1 1 1 1]

By this way, generate the individuals in the next generation (single-point crossover/crossover probability):

Current Population	Next Population (after crossover)
(Chromosome 1) [0 1 0 1 1 1]	[0 1 0 0 1 0]
(Chromosome 2) [1 1 0 1 1 1]	[1 0 1 1 1 1]
(Chromosome 3) [1 0 1 0 1 0]	[1 1 0 0 1 0]
(Chromosome 4) [1 0 0 0 0 1]	[1 0 1 1 1 1] (Repetition)

MAXONE Problem using GA

Step 5: Generate the offspring (successor) using Crossover and Mutation

Mutation (mutation probability)

Current Population	Next Population (after crossover)	Next Population (after crossover and mutation)
[0 1 0 1 1 1]	[0 1 0 0 1 0]	[0 1 0 0 1 0]
[1 1 0 1 1 1]	[1 0 1 1 1 1]	[1 0 0 1 1 1]
[1 0 1 0 1 0]	[1 1 0 0 1 0]	[1 1 0 0 0 0]
[1 0 0 0 0 1]	[1 0 1 1 1 1]	[1 0 1 0 1 1]

Step 6: Continue till convergence (number of generations, no change in the fitness between populations, reach the goal state (if knows))

Note: The next states are generated without storing the all the successors of k current states

Complete state-formulation (Clustering)

Formulate the following problem for intelligent agent:

Divide the N students in k groups based on their marks in AI such as diversity in each group is minimized.

For demonstration, you may use:

N=5, k=2, set of marks={50, 20,10, 5, 15, 45} (out of 60)

Complete state-formulation (Version 2) (Clustering)

Set of students/marks $A=\{a_1, a_2, a_3, \ldots\ldots, a_N\}= =\{50, 20, 10, 5, 15, 45\}$

- State representation:
  $S=[C_1\ C_2\ C_3\ \ldots\ldots\ C_k]$
  if C_j is representative of the jth group, then $C_j \in \{a_1, a_2, a_3, \ldots, a_N\}$

- Initial state: (example for demonstration)
  $S^0=[50\ 20]$ (assigned in a group with nearest representative; then same as [1 2 2 2 2 1])

  $F(S^0)=[(50-50)^2+(50-45)^2]+[(20-20)^2+(20-10)^2+(20-5)^2+(20-15)^2]$

- Goal state/test:
  No binary goal state
  report the state when we have the minimum value of objective function
- Successor function: Successor is generated by changing a group representative at a time
- Objective function: summation of diversity in each cluster

Clustering using GA

- Encoding: Binary/Integer/Real-valued/Character/Permutation

- Parameters: Population size
  crossover probability
  crossover point (randomly)
  mutation probability
  number of generations

- Strategy for selection: Roulette wheel selection/rank selection/tournament selection/Elitism

- Type of crossover: Single-point/multi-point

- Type of mutation: Bit flip/Random setting/Swap mutation

Clustering using GA

- discuss using discrete state space
- GA is initially designed using binary representation. Here, the integer values may be converted to binary string for encoding.

Current Population	Selection/crossover (3rd and 4th chromosomes)
[50 20]	
[10 5]	
[50 5]	[50 45]
[20 45]	[20 5]

Observation

- Discuss using discrete state space; but it can be generalized to continuous state space.
- In case of clustering problem, we may use real-valued representation also and continuous state space design.
- Here, the representation of the groups (state element) may not be from

  $A=\{a_1, a_2, a_3, \ldots\ldots, a_N\}= =\{50, 20,10, 5, 15, 45\}$

- We are searching for a better representative
- Offspring generation (crossover) policy is different.

Real-valued GA (Crossover)

- Ch1=[X_m Y_m] and Ch2 =[X_d Y_d]
- we first randomly select one of the parameters to be the point of crossover
- Suppose, *X* is the crossover point for a particular *Ch 1 and Ch2*
- Then we introduce a random value between 0 and 1 represented by *β* and the *X*-values in the offspring are

$$X_{new1} = (1 - \beta)X_m + \beta X_d$$
$$X_{new2} = (1 - \beta)X_d + \beta X_m$$

- The remaining parameter (*y* in this case) is inherited directly from each parent, so the completed offspring are

  offspring1 = [X_{new1} Y_m]
  offspring2 = [X_{new2} Y_d]
- Apply for clustering, Ch1= [50 20] and Ch2=[20 45]

Real-valued GA (Crossover)

- Ch1=[X_m Y_m] =[50 5] and Ch2 =[X_d Y_d]=[20 45]
- $X_{new1} = (1 - \beta)X_m + \beta X_d$ =(1-0.2)*50+0.2*20
  $X_{new2} = (1 - \beta)X_d + \beta X_m$ =(1-0.2)*20+0.2*50
- offspring1 = [Xnew1 Ym]
  offspring2 = [Xnew2 Yd]

Current Population	Selection/crossover (3rd and 4th chromosomes)
[50 20]	
[10 5]	
[50 5]	[44 5]
[20 45]	[26 45]

Particle Swarm optimisation

- Inspired from the nature social behavior and dynamic movements with communications of insects, birds and fish