Metaheuristics Unit-8.

Introduction:

Heuristic methods are commonly used to search for a good feasible solution. A heuristic method is a procedure that is likely to discover a very good feasible solution, but not necessarily optimal solution, for the specific problem being considered. No assurance can be given about the quality of the solution obtained, but a well-designed heuristic method usually can provide a solution that is atleast nearly optimal.

A metaheuristic is a general solution method that provides both a general structure & strategy guidelines for developing a specific heuristic method to fit a particular kind of problem.

Three most commonly used metaheuristics are
(i) Tabu Search
(ii) Simulated annealing and
(iii) Genetic algorithms.

Nature of Metaheuristics

Metaheuristics is a general kind of solution method that orchestrates the interaction between local improvement procedures & higher level of strategies to create a process that is capable of escaping from local optima & performing a robust search of the feasible region.
A key feature of the metaheuristic is its

ability to escape from a local optimum.

* After reaching (or nearly reaching) a local optimum, different metaheuristics execute this escape in different ways.
* However, a common characteristic is that the trial solution that immediately follow a local optimum are allowed to be inferior to this local optimum
* The advantage of a well-designed metaheuristic is that it tends to move relatively quickly toward very good solutions, so it provides a very efficient way of dealing with large & complicated problems.

* The disadvantage is that there is no guarantee that the best solution found will be optimum solution or even a near optimal solution.

Simulated Annealing.

Simulated annealing focuses mainly on searching for the tallest hill. Since tallest hill can be anywhere in the feasible region, the emphasis is on taking steps in random directions. Along the way, we reject some, but not all steps that would go downward rather than upward.

Since most of the accepted steps are going upward, the search will gradually gravitate toward those part of feasible region containing the tallest hills. Therefore the search process gradually increases the emphasis on climbing upward by rejecting an increasing proportion of steps that go downward. Simulated annealing moves from current solution to an immediate neighbor in the local neighborhood of the solution.

How is an immediate neighbor is selected?

Let Z_c = objective fnc value for the current trial sol^n

Z_n = " " " " " " candidate to be the next trial solution.

T = a parameter that measures the tendency to accept the current candidate to be the next trial solution if this candidate is not an improvement on the current trial solution.

Move Selection rule

* Among all the immediate neighbors of the current trial solution, select one randomly to become the current candidate to be the next trial solution.

* Assuming the objective is maximization of the objective function, accept or reject this candidate to be the next trial solution as per the following rule.

1. If $Z_n >= Z_c$ always accept this candidate
2. If $Z_n <= Z_c$ accept the candidate with the following probability:

$$\text{Prob}\{\text{acceptance}\} = e^{x}, \text{ where } X = \frac{(Z_n < Z_c)}{T}$$

* If the candidate is rejected, repeat the process with a new randomly selected immediate neighbor of the current trial solution.

* If no immediate neighbor remains, terminate the algorithm.

* T = a parameter that measures the tendency to accept the current candidate to be the next trial solution if this candidate is not an improvement on the current trial solution.

Outline of the Basic Simulated Annealing Algorithm.

- * Initialization ~~Start Stea~~ Start with a feasible initial trial solution.
- * Iteration Use the move selection rule to select the new trial solution. If none of the immediate neighbors of the current trial solution are accepted, the algorithm is terminated.
- *. Check the current temperature schedule when the desired number of iterations have been performed at the current value of T, decrease T to the next value in the temperature schedule & resume performing iterations at this next value.
- * Stopping Rule: when the desired number of iterations have been performed at the smallest value of T in the temperature schedule, stop. Algorithm is also stopped when none of the immediate neighbors of the current trial solution are accepted.
- * Accept the best trial solⁿ found at any iteration (including for larger values of T) as the final solⁿ

The traveling Sales Man problem (Routing problem)

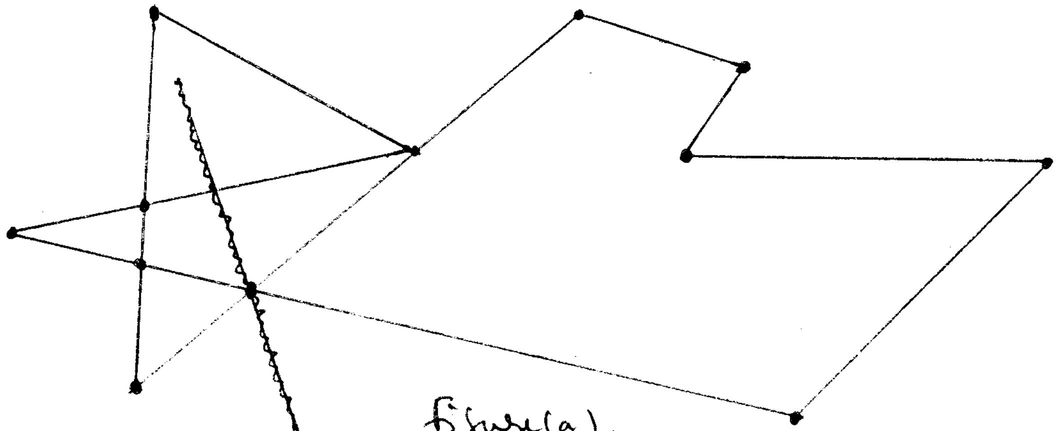
Suppose a salesman has to visit many cities, needs to start from a particular city, visit each city once, & then return to his starting point. the objective is to select the sequence in which the cities are visite in such a way that his total travelling time is minimized Starting from a given city the salesman will have a total of $(n-1)!$ different sequences

Applications of travelling salesman problem

i) Postal deliveries
ii) Cable connections
iii) Inspection
iv) School bus routes.

Example of travelling salesman problem (simulated annealing).

Simulated annealing quickly gives a reasonabl answer close enough to the true minimum path for practical purposes. It starts with the cities connected in a random order, & then considers making random changes in that order. If changing the order of cities leads to a shorter path, we accept that chang If it results a longer path, we give a certain probability of accepting the modification less likely. the larger & proposed increase in path length. we then gradually reduce this probability over time, in order to rule out shorter path & path length increases. there by converging toward a path length close to the absolute minimum.



Figure(a)

The modified routing after applying simulated annealing concept is shown in figure (b).

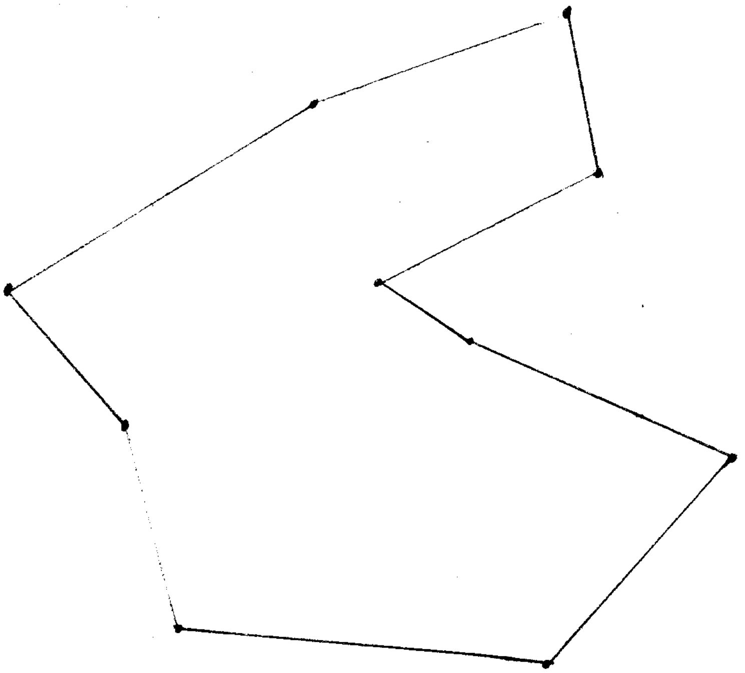


fig (b).

Conclusion: we can notice that, the route represented in fig (b) is optimum (shorter) compared to route shown in fig (a).

Steps in obtaining optimization:

* Initial trial solution: we may enter any feasible solution (sequence of cities on the tour), perhaps by randomly generating the sequence. It might be helpful to enter a good feasible solution as initial trial solⁿ.

* Neighborhood structure: An immediate neighbor of the current trial solution is one that is reached by making a sub-tour reversed.

* We must, however, rule out the sub-tour reversal that simply reverses the direction of tour provided by the current trial solution.
* Random selection of immediate neighbor: Selecting the sub-tour to be reversed requires selecting the slot in the current sequence of cities where the sub-tour currently begins & then the slot where the subtour currently ends.
* The ending slot must be somewhere after the beginning slot, excluding the last slot.
* We can use random numbers to give equal probabilities to selecting any of the eligible beginning slots & then any of the eligible ending slots.
* If this selection turns out to be infeasible, then the process is repeated until a feasible selection is made.

Tabu search.

Tabu search begins by using a local search procedure as a local improvement procedure in the usual sense to find the local optimum.

The strategy in tabu search is that it continues the search by allowing non-improving moves to the best solutions in the neighborhood of the local optimum solution.

Outline of a Basic Tabu Search Algolithm.

* Initialization
* Start with a feasible initial trial solution.

Iteration

* Use an appropriati local search procedure to define the feasible moves into the local neighborhood of the current solution.
* Eliminate from consideration any move on the current tabu list unless the move would result in a better solution than the best trial solution found so far.
* Determine which of the remaining moves provides the best solution.
* Adopt this solution as the next trial solution, regardless of whether it is better or worse than the current trial solution.
* Update the tabu list to forbid cycling back to what had been the current trial solution.
* If the tabu list already had been full, delete the oldest member of the tabu list to provide more flexibility for future moves.

Stopping rule.

* Use any stopping criteria, such as fixed number of iterations, a fixed amount of CPU time, a fixed number of consecutive iterations without an improvement in the best objective fnc value.
* Also stop at any iterations when there are no

feasible moves in the local neighborhood of the current trial solution.

* Accept the best trial sol^n found at any iteration as the final solution.

These steps only provide a generic structure & strategy guidelines for developing a specific heuristic method to fit a specific situation.

Steps in obtaining optimization

* local search algorithm.

At each iteration, choose the best immediate neighbor of the current trial sol^n that is not ruled out by the tabu status.

* Neighborhood structure.

An immediate neighbor of the current trial solution is one that is reached by making a sub tour reversal. Such a reversal requires adding two links & deleting two other links from the current trial sol^n.

* Form of tabu moves.

List the links such that a particular sub tour reversal would be tabu if both links to be deleted in this reversal are on the list.

: * Addition of tabu move.

At each iteration, after choosing the two links to be added to the current sol^n. also add these two links to the tabu list.

Maximum size of the tabu list.

Four (two from each of the two most recent iterations whenever, a pair of links is added to a full list, delete the two links that already have be on the list the longest.

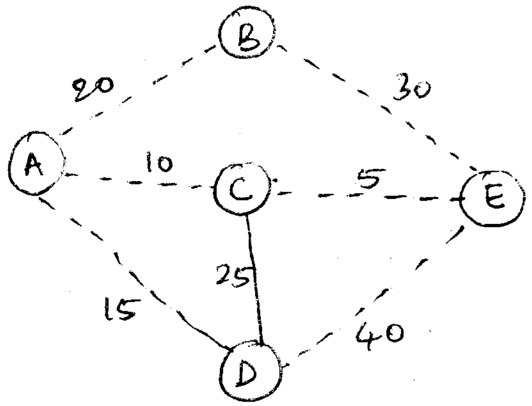
* Stopping rule

Stop if after three consecutive iterations there is no improvement in the best objective fnc value. Also stop at any iteration where the current solution has no immediate feasible neighbor.

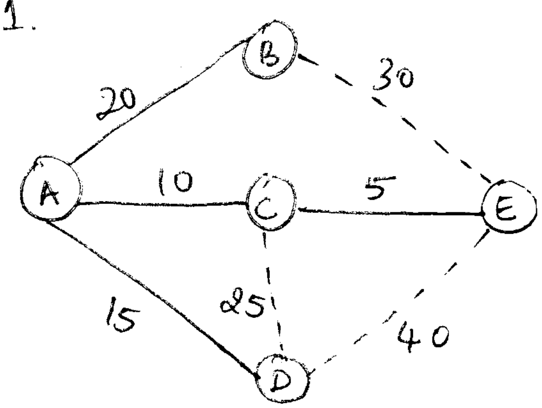
Eg A minimum spanning tree problem with constraints. Use Tabu search algorithm to find the optimal solution of the following illustration.

Constraint 1: link AD can be included only if link DE also is included.

Constraint 2: At most one of the three links AD, CD & AB can be included. Charge a penalty of 100 if constraint 1 is violated. Charge a penalty of 100 if two of the three links specified in constraint 2 are included. Increase this penalty to 200 if all three of the links are included.



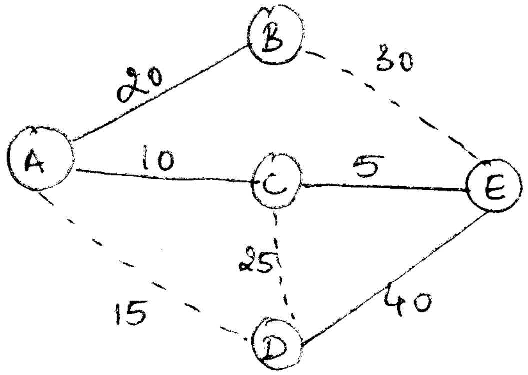
Sol^n Iteration- 1.



cost = 20+10+5+15+100+100 (200 is a penalty as per the given constraints)

= 250.

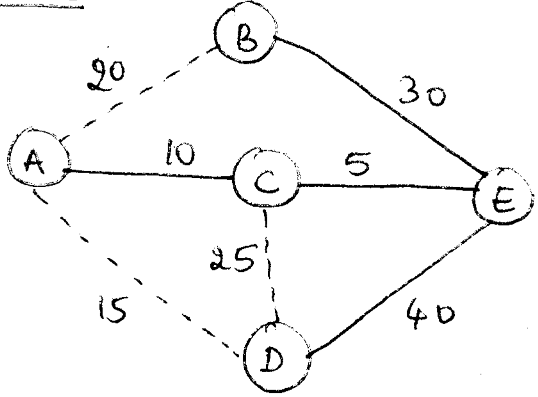
Iteration - 2



Cost = 20+10+5+40

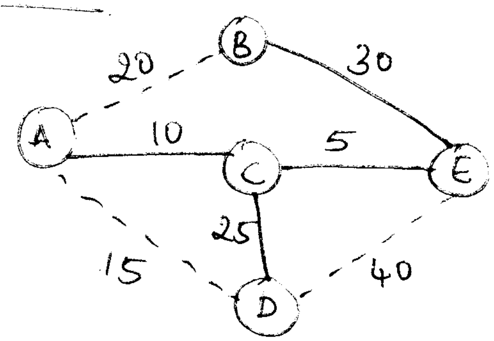
= 75

Iteration-3



cost in Rs = 30 + 10 + 5 + 40
= 85

Iteration-4



cost in Rs. = 25 + 10 + 5 + 30
= 70

The optimal Solution is Rs. 70/- hence iteration-4 has optimal path.

Genetic Algorithms.

Genetic algorithms are completely based on natural phenomenon. The analogy is the biologica theory of evolution formulated by Charles Darwin (in mid 19th century).

Outline of a basic genetic algorithm.

Initialization: Start with initial population of feasib trial solns, perhaps by generating them randoml Evaluate the fitness - the objective func value - for each member of this current generation.

Iteration

- * Use a random process that is biased towards more fit members of the current population to select some of its members to become parents.
- * pair up the parents randomly & then have each pai of parents give birth to two children - new feasible solutions - whose features (genes) are a random mixture of the features of the parents.
- * what is the random mixture of features and/or any mutations result in an infeasible solution?
- * These cases are miscarriages - so the process of attempting to give birth is repeated until a child is born that corresponds to a feasible soln.
- * Retain the children & enough of the best member of the current population to form the new populati of the same size of the next iteration.
- * we discard the other members of the populatio

* Evaluate the fitness for each new member (the children) in the new ~~pt~~ populations.

Stopping rule

* Use some stopping rule, such as a fixed number of iterations, a fixed amount of time, or a fixed number of consecutive iterations without any improvement in the best trial solⁿ found so far.
* Use the best trial solⁿ found on any iteration as the final solⁿ.

Example of travelling salesman problem.

Many practical applications can be modeled as a travelling salesmen problem 'or as its' variants. The constraints in travelling salesmen problem are.

* He should come back to the home city after visiting only once all the cities/places assigned.
* The salesmen can only be in one city at a time.

Steps in obtaining optimization.

* we represent the solution by just the sequence in which cities are represented. However, in most of the application of GA, typically the members of population are coded so that it is easier to generate children, create mutation etc.

- * First task is then to generate population for the initial generation.
- * Starting with home base city (1), random numbers are used to Select the next city from amongst those that have a link to the city 1.
- * Same process is repeated to select the subseque cities that would be visited in this tour (member).
- * we stop if all the cities are visited & we are back to the home base city.
- * Or we reach a dead end (because there is no link from the current city to any of the remaining city's that are still not in the tour) In this case, we start the process all over again.

Limitations of Standard Genetic Algorithm.

The following problems have to be addressed to use a standard genetic algorithm.

- * A binary representation for tours is found such that it can be easily translated into chromosome
- * An approprate fitness fun is to be designed, taking the constraints into consideration.

Genetic algorithms can generate some chromosomes that do not represent valid sol^n due to random initialization step of the genetic algorithms.

→ Two tours including the same places in the same order but with different starting points/directions are represented by diff^n matrices & hence by different chromosome.

A proper fitness fnc is obtained using penalty fnc method to enforce the constraints.

But, poor results may be obtained by ordinary genetic operators.