Artificial Intelligence for Robotics

Unit 1: Introduction to AI Techniques

Dr. Padmakar J. Pawar

Professor and Head

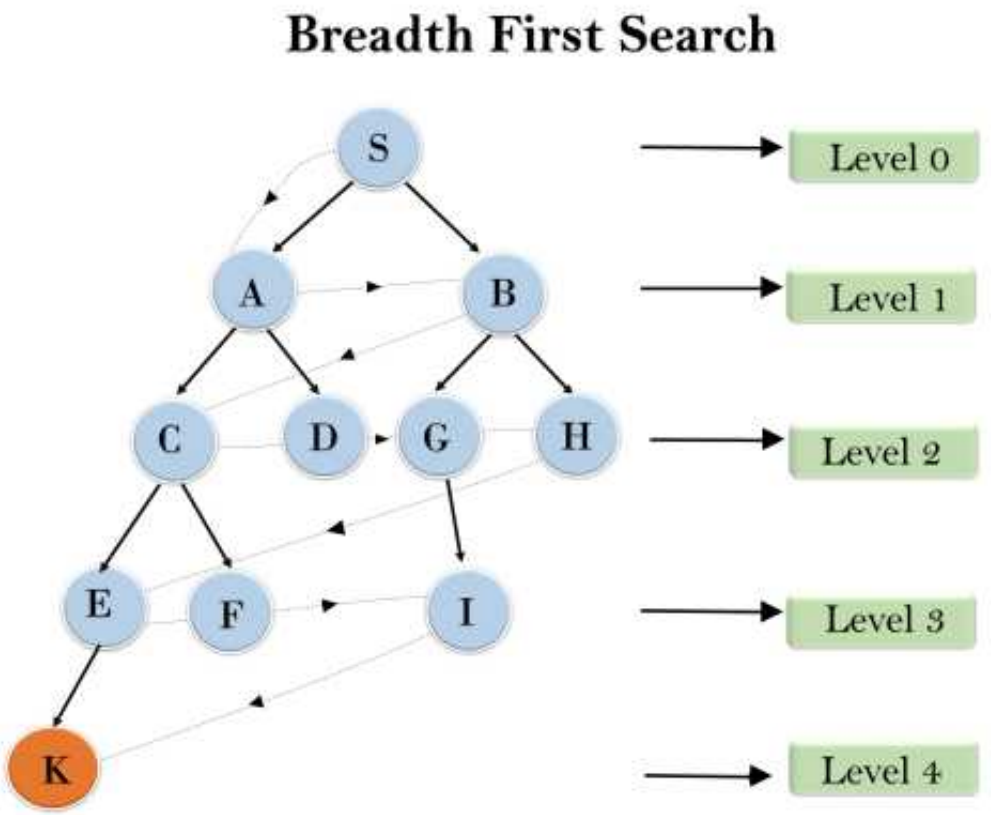
Department of Robotics and Automation

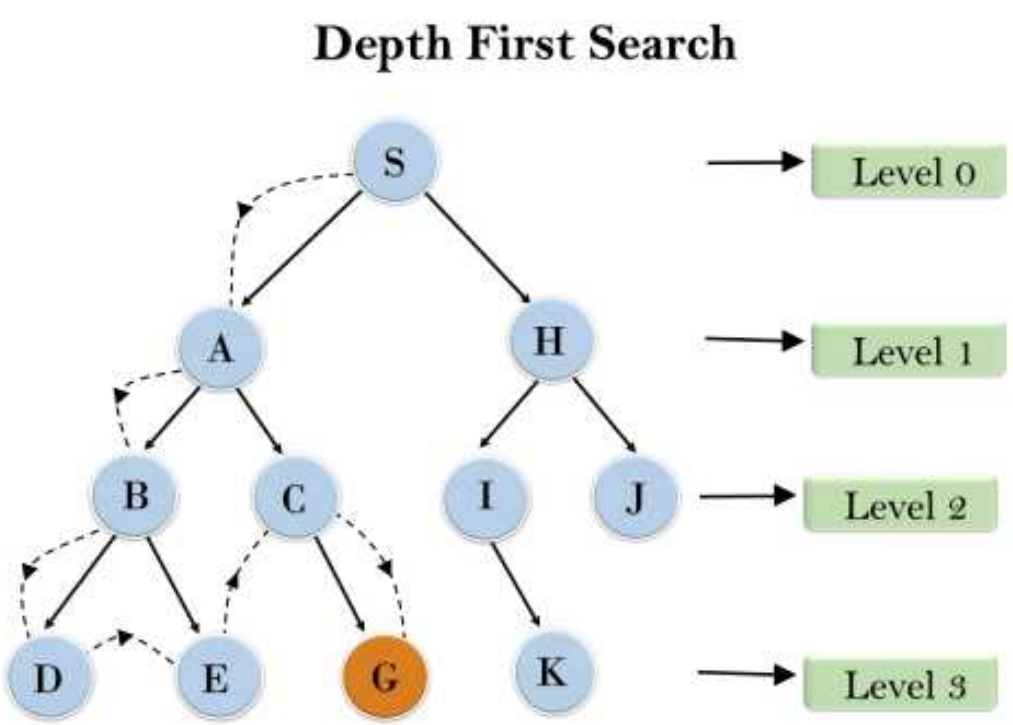
K. K. Wagh Institute of Engineering Education and Research, Nasik

Outline:

1. Search algorithms
2. Heuristics and Metaheuristics
3. Handling uncertainties : Fuzzy logic
4. Probabilistic methods for uncertain reasoning
5. Learning methods: Statistical
6. Learning methods : Soft computing - Neural networks

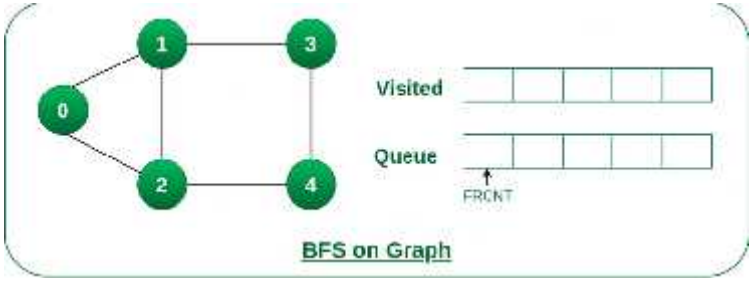
Uninformed search algorithms:

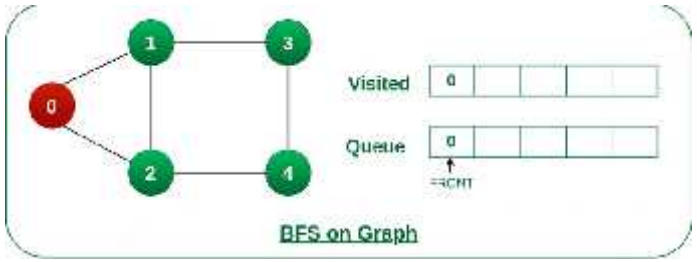


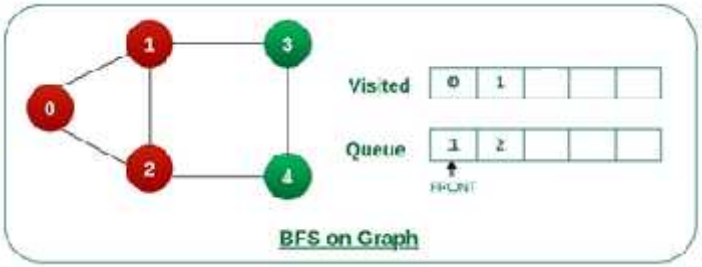


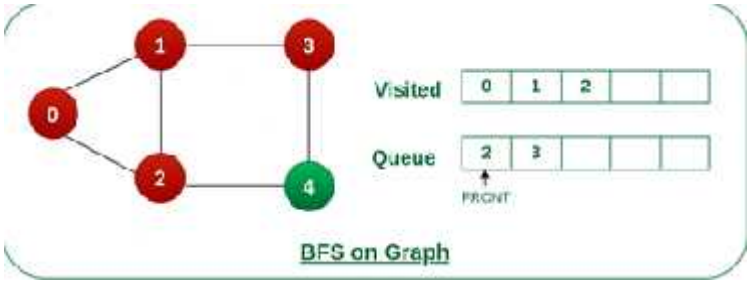
Uninformed search algorithms:

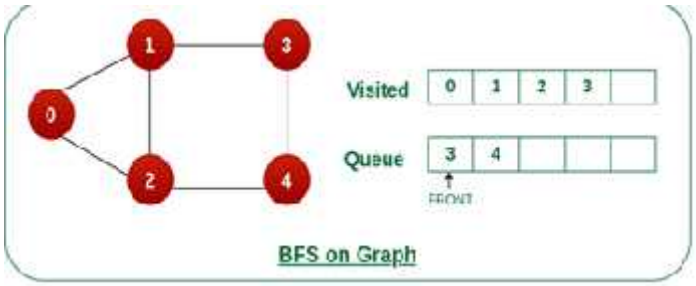
- **Systematic exploration –** uninformed search algorithms explore the search space systematically, either by expanding all children of a node (e.g. BFS) or by exploring as deep as possible in a single path before backtracking (e.g. DFS).
- **No heuristics –** uninformed search algorithms do not use additional information, such as heuristics or cost estimates, to guide the search process.
- **Blind search –** uninformed search algorithms do not consider the cost of reaching the goal or the likelihood of finding a solution, leading to a blind search process.
- **Simple to implement –** uninformed search algorithms are often simple to implement and understand, making them a good starting point for more complex algorithms.
- **Inefficient in complex problems –** uninformed search algorithms can be inefficient in complex problems with large search spaces, leading to an exponential increase in the number of states explored.
- Example: Breadth first search (BFS), Depth search first (DFS), branch and bound etc.

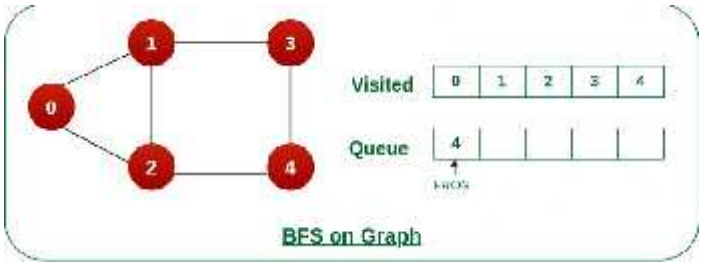

0
1
2
3
4
Visited
Queue
FRONT
BFS on Graph

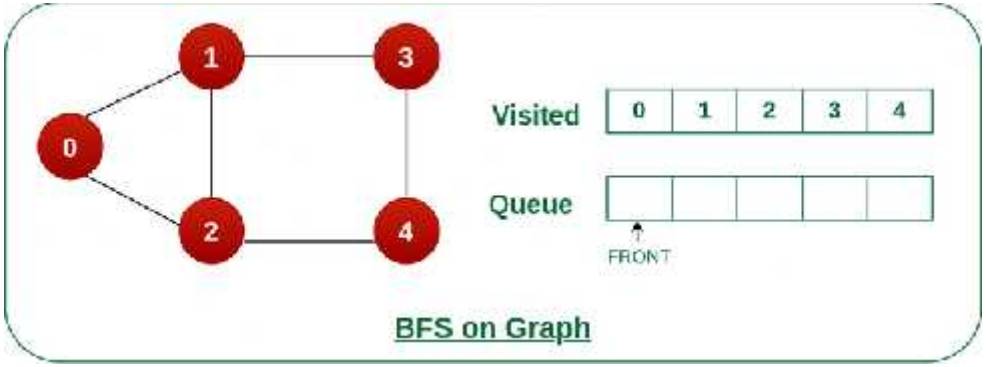

0
1
2
3
4
Visited
0
Queue
0
FRONT
BFS on Graph

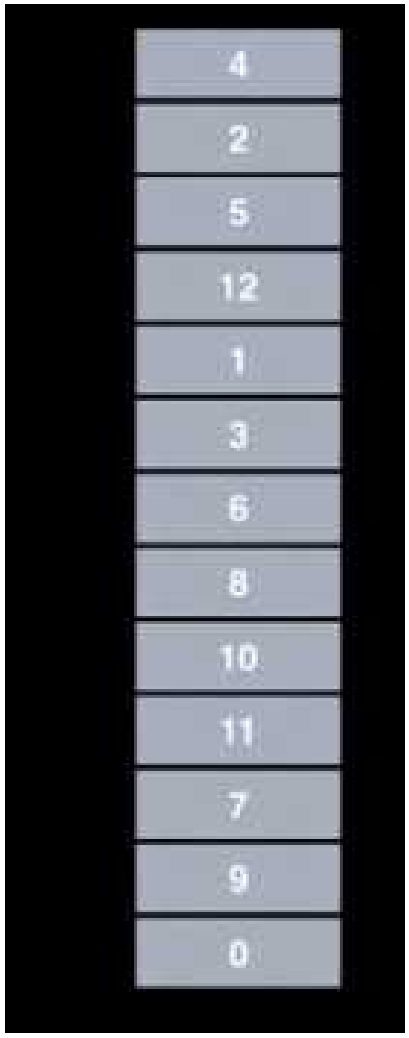

0
1
2
3
4
Visited
0 1
Queue
1 2
FRONT
BFS on Graph

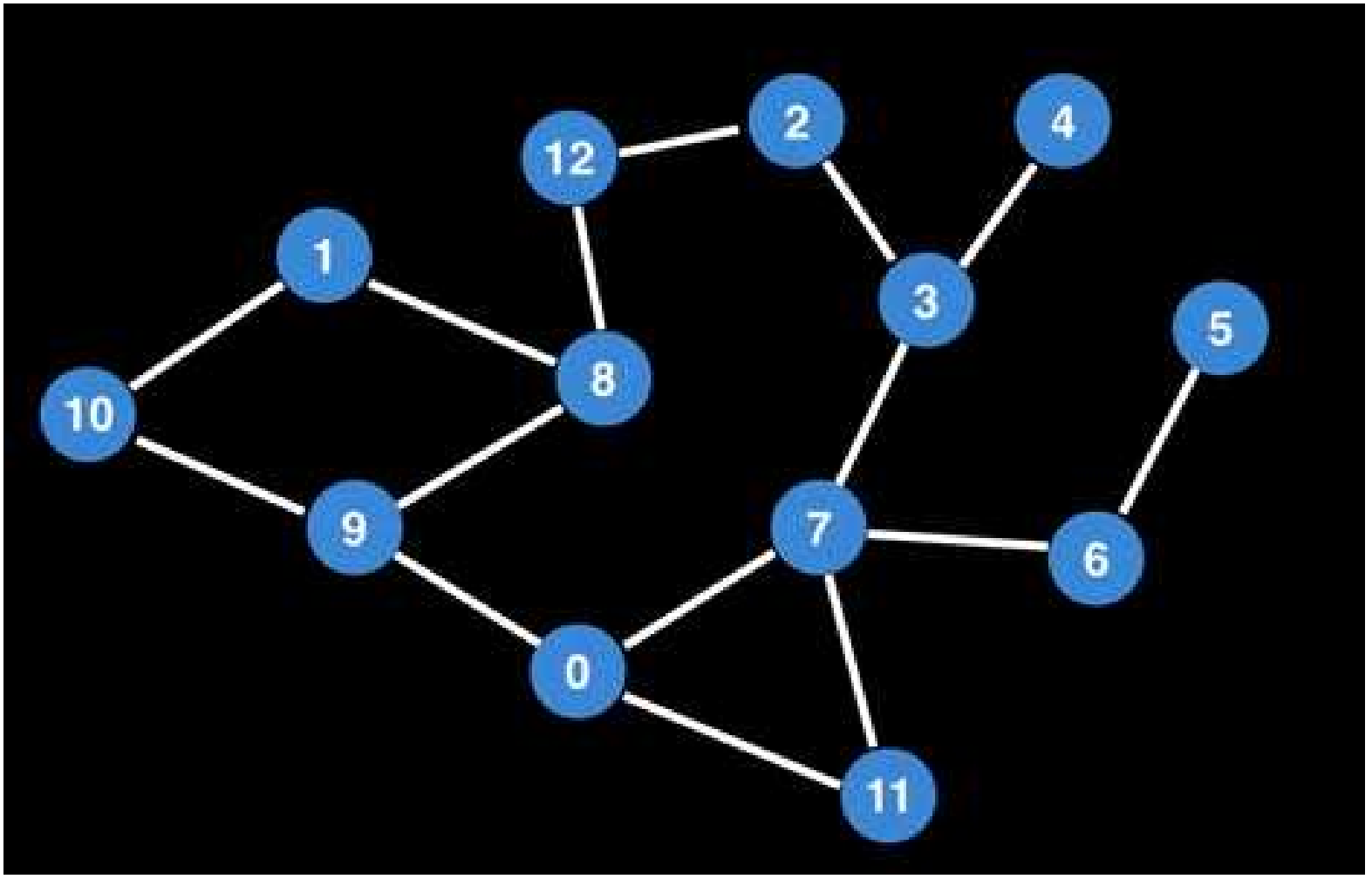

0
1
2
3
4
Visited
0 1 2
Queue
2 3
FRONT
BFS on Graph

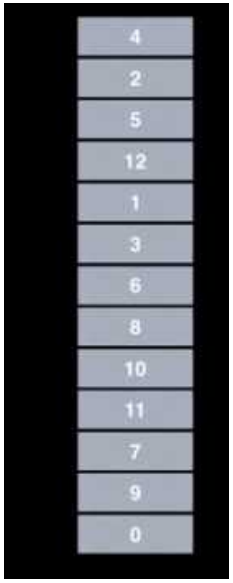

0
1
2
3
4
Visited
0 1 2 3
Queue
3 4
FRONT
BFS on Graph

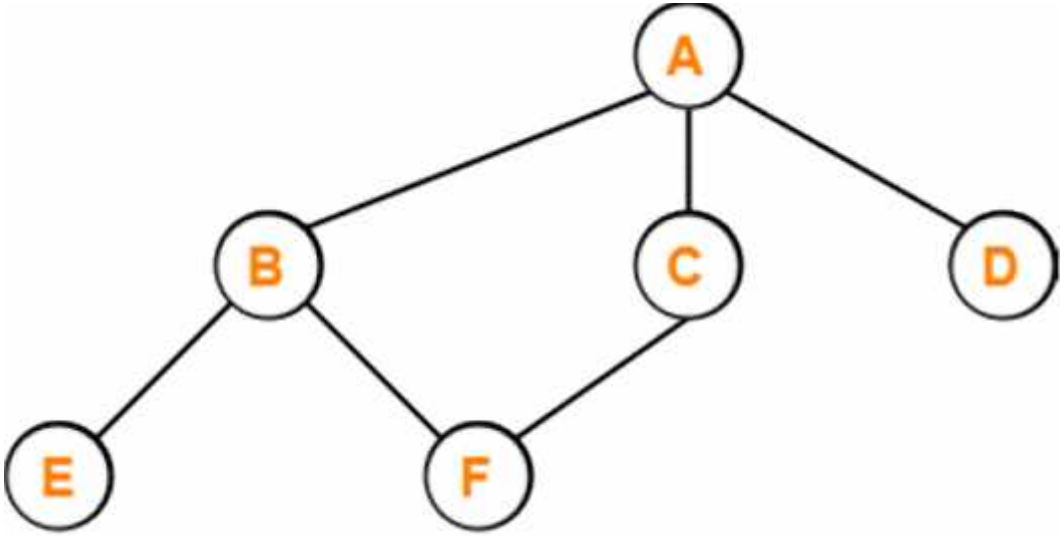

0
1
2
3
4
Visited
0 1 2 3 4
Queue
4
FRONT
BFS on Graph


0
1
2
3
4
Visited
0 1 2 3 4
Queue
FRONT
BFS on Graph


4
2
5
12
1
3
6
8
10
11
7
9
0


12
2
4
1
3
5
8
10
9
7
6
0
11

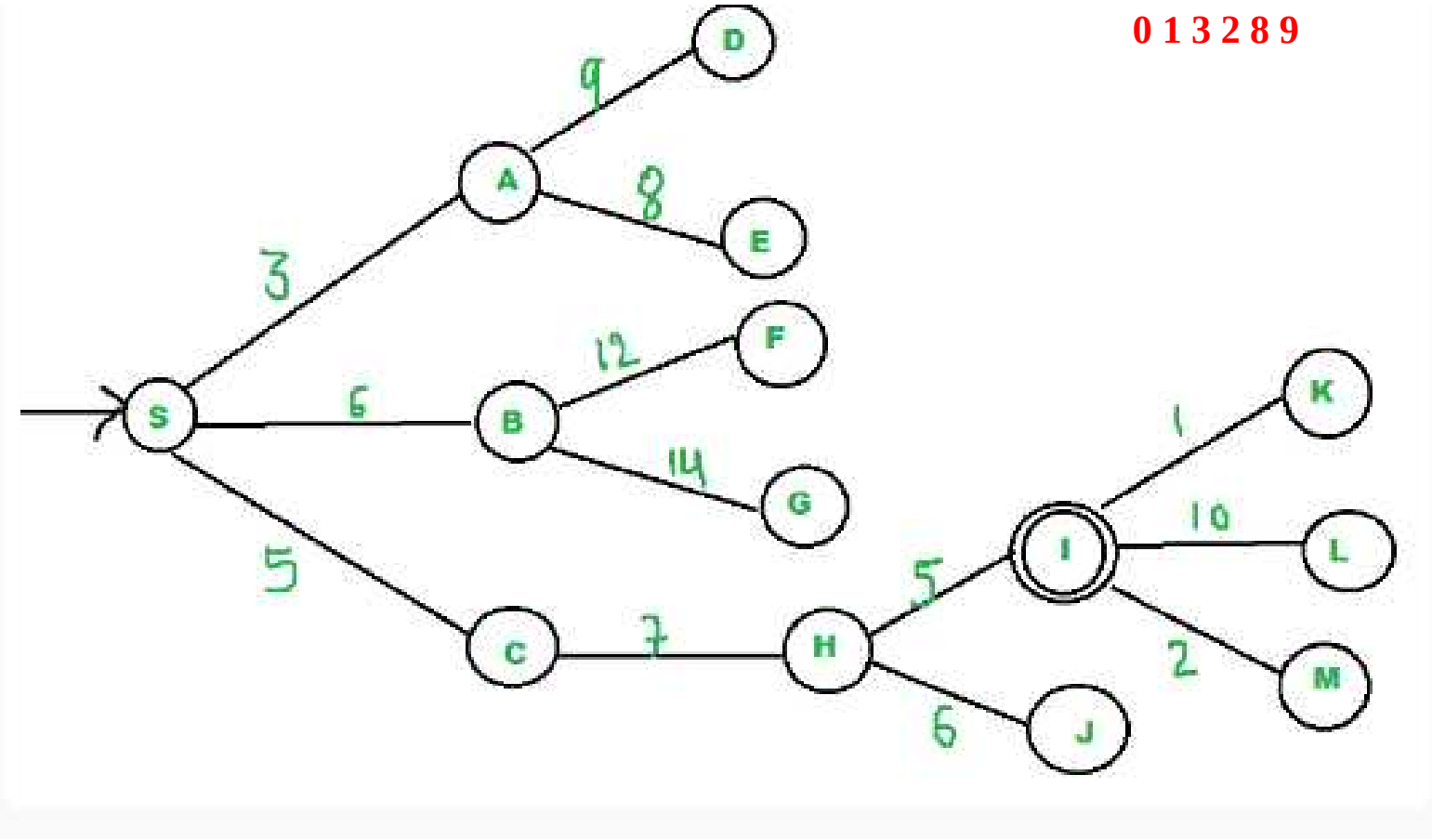

4
2
5
12
1
3
6
8
10
11
7
9
0


A
B
C
D
E
F

Informed search algorithms:

- **Use of Heuristics –** informed search algorithms use heuristics, or additional information, to guide the search process and prioritize which nodes to expand.
- **More efficient –** informed search algorithms are designed to be more efficient than uninformed search algorithms, such as breadth-first search or depth-first search, by avoiding the exploration of unlikely paths and focusing on more promising ones.
- **Goal-directed –** informed search algorithms are goal-directed, meaning that they are designed to find a solution to a specific problem.
- **Cost-based –** informed search algorithms often use cost-based estimates to evaluate nodes, such as the estimated cost to reach the goal or the cost of a particular path.
- **Prioritization –** informed search algorithms prioritize which nodes to expand based on the additional information available, often leading to more efficient problem-solving.
- **Optimality –** informed search algorithms may guarantee an optimal solution if the heuristics used are admissible (never overestimating the actual cost) and consistent (the estimated cost is a lower bound on the actual cost).
- Example: A*, Hill climbing, Best first search

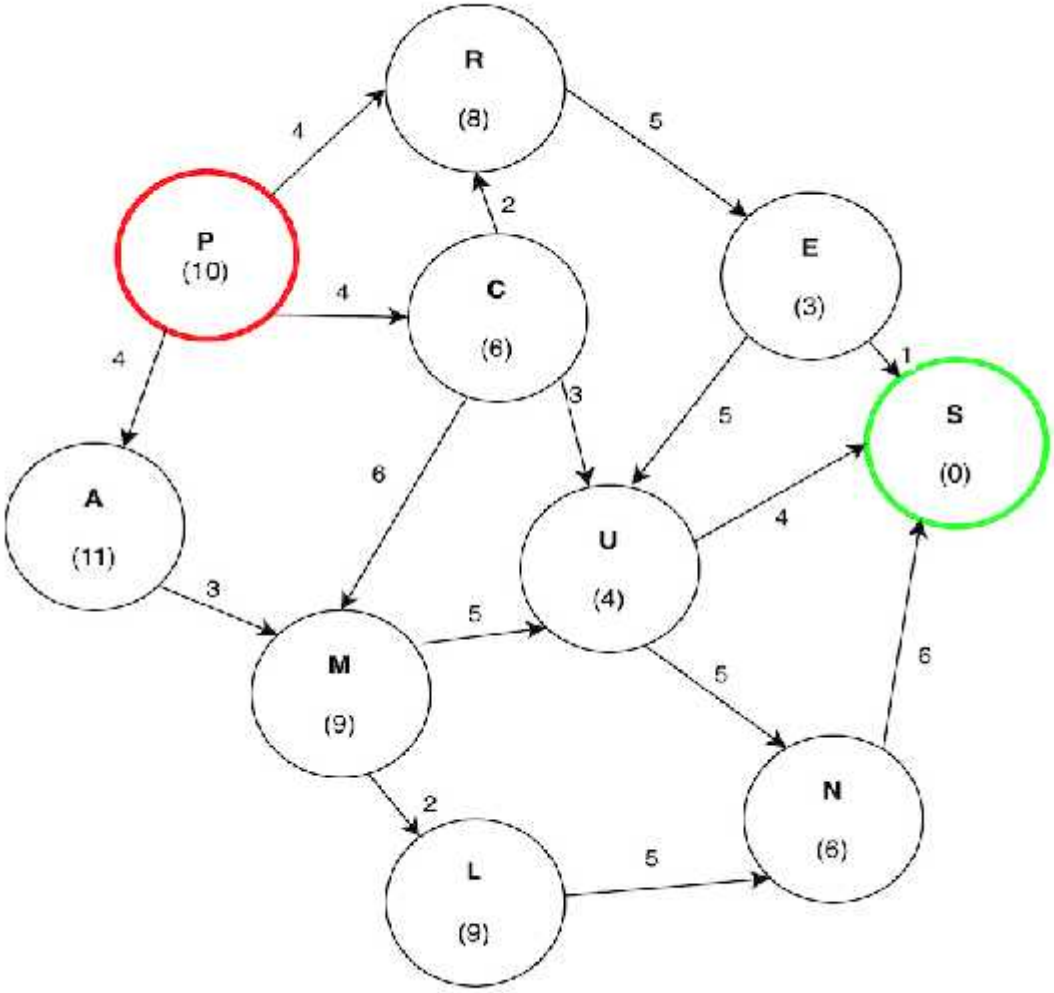
Best First Search (Greedy algorithm)

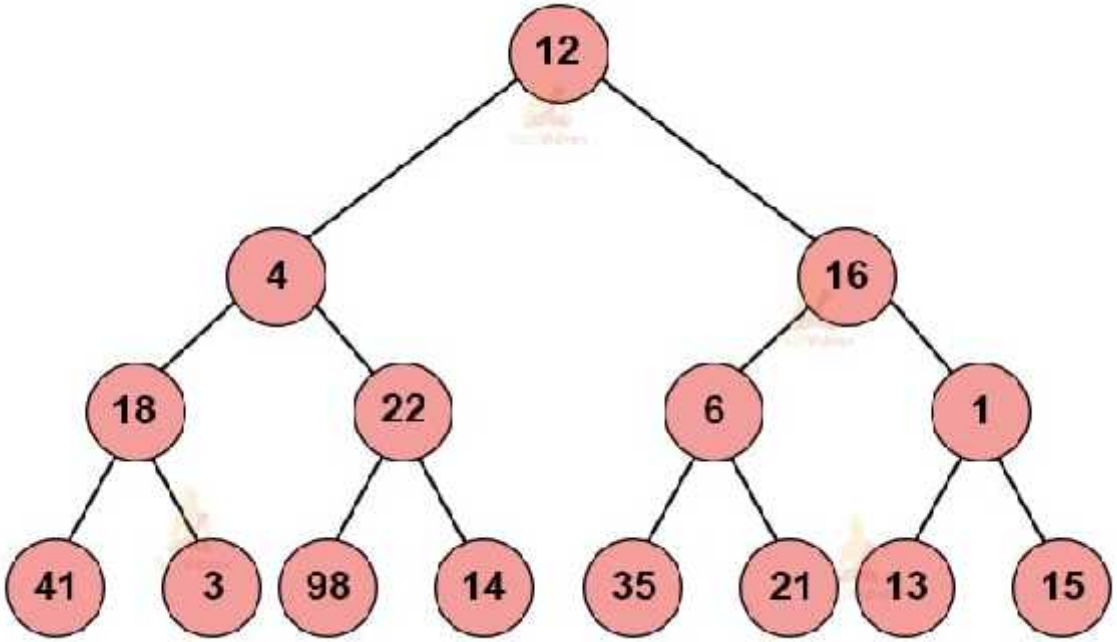


Applications:

- ***Path finding-***
  Navigation system in robotics
- ***Machine learning***
  To find most promising path
- ***Optimization:***
  To determine best state of process

Greedy Search Algorithm




12
4
16
18
22
6
1
41
3
98
14
35
21
13
15

Ant colony optimization

Presented by

Dr. Padmakar J. Pawar

Professor and Head
Department of Robotics and Automation
K. K. Wagh Institute of Engineering Education and Research,
Nasik, Maharashtra.

Ant Colony Optimization (ACO) Algorithm

- Dorigo (1992) proposed the idea of the ant colony algorithm to mimic intelligent behaviour of ants.
- In the natural world, ants wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but instead to follow the trail, returning and reinforcing it if they eventually find food Over time.
- However, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones.

Ant Colony Optimization (ACO) Algorithm

- Pheromone evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained. The influence of pheromone evaporation in real ant systems is unclear, but it is very important in artificial systems.

- The overall result is that when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads to many ants following a single path.

Steps of ACO Algorithm

Step 1: Determine the algorithm specific parameters such as Number of ants (Population size), initial pheromone level, control parameters α and β respectively for pheromone level and desirability, Evaporation rate ° , Constant Q

Step 2: Determine the probability with which the ant will select particular path

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in \text{allowed}_x} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})}$$

Step 3: Update the pheromone level

When all the ants have completed a solution, the trails are updated by $\tau_{xy} \leftarrow (1-\rho)\tau_{xy} + \sum_{k} \Delta\tau_{xy}^{k}$

where τ_{xy} is the amount of pheromone deposited for a state transition xy, ρ is the *pheromone evaporation coefficient* and $\Delta\tau_{xy}^{k}$ is the amount of pheromone deposited by kth ant, typically given for a TSP problem (with moves corresponding to arcs of the graph) by

$$\Delta\tau_{xy}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ uses curve } xy \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

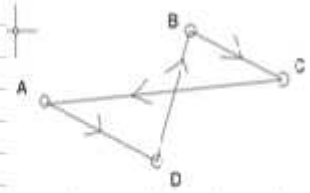
where L_k is the cost of the kth ant's tour (typically length) and Q is a constant.

Applications of ACO Algorithm

- **Scheduling problem**
- **Vehicle routing problem**
- **Assignment problem**
- **Set problems**
- **Design of CMOS based sense amplification circuit**
- **Antennas optimization and synthesis**
- **Image processing**
- **Other applications:** Data mining, classification, Electricity Network design, distribution systems etc.

	A	B	C	D
A	0	14.93	22.92	11.63
B	14.93	0	9.63	10.36
C	22.92	9.63	0	14
D	11.63	10.36	14	0

54.54



τ_new by Ant 1

	A	B	C	D	
A	0.5	0.5	0.5	0.5199	A-D-C-B-A
B	0.5199	0.5	0.5	0.5	
C	0.5	0.5199	0.5	0.5	
D	0.5	0.5	0.5199	0.5	

Route by Ant 1: A-D-C-B-A
Total distance travelec by ant 1 = 11.63+14+9.63+14.33 = 50.13
Pheromone deposite by An: 1 (Dt) = 1/50.19 = 0.0199
Updated Pheromone level: (1-r)*t+Dt = (1-0.5)*1+0.0192 = 0.5199

Route by Ant 2: A-C-E-C-A
Total distance travelec by ant 2 = 22.92+9.63+10.36+11.63 = 54.54
Pheromone deposite by An: 2 (Dt) = 1/54.54 = 0.0183

η

	A	B	C	D
A	0.000	0.067	0.044	0.086
B	0.067	0.000	0.104	0.097
C	0.044	0.104	0.000	0.071
D	0.086	0.097	0.071	0.000

τ

	A	B	C	D
A	1	1	1	1
B	1	1	1	1
C	1	1	1	1
D	1	1	1	1

τ_new by Ant 2

	A	B	C	D	
A	0.5	0.5	0.5183	0.5199	A-C-B-D-A
B	0.5199	0.5	0.5	0.5183	
C	0.5	0.5382	0.5	0.5	
D	0.5183	0.5	0.5199	0.5	

Route by Ant 3: A-D-E-C-A
Total distance travelec by ant 3 = 11.63+10.36+9.63+22.93 = 54.54
Pheromone deposite by An: 3 (Dt) = 1/54.54 = 0.0183

Probability from Ant 1 from Location A

	A	B	C	D
A	0.000	0.067	0.044	0.086
B	0.000	0.000	0.104	0.097
C	0.000	0.104	0.000	0.071
D	0.000	0.097	0.071	0.000

B	C	D	Total	Rand()
0.004	0.002	0.007	0.014	0.48
0.325	0.138	0.536		
0.325	0.463	1.000		

τ_new by Ant 3

	A	B	C	D	
A	0.5	0.5	0.5	0.5382	A-D-E-C-A
B	0.5199	0.5	0.5183	0.5	
C	0.5183	0.5382	0.5	0.5	
D	0.5	0.5183	0.5199	0.5	

Probability from Ant 1 from Location D

	A	B	C	D
A	0.000	0.067	0.044	0.000
B	0.000	0.000	0.104	0.000
C	0.000	0.104	0.000	0.000
D	0.000	0.097	0.071	0.000

B	C	Total	Rand()
0.009	0.005	0.014	0.74
0.547	0.354		
0.547	1.000		


THANK YOU !

Real Coded Genetic Algorithm

Problem statement

To make a complete hole, tools with different sizes may be needed. This is specially a must when the diameter of the hole to be made is large. In this case, the hole is initially made using the small-sized tools and then it is enlarged to the size of interest using the large-sized tools.The selection of the set of tools and their sequence can directly affect the machining time and cost. It is common in practice that several holes need a particular tool and a hole may need different tools. The time needed to move from a hole to one another is called as airtime .To minimize tool airtime, it may be initially thought that a hole should be completed through different tools before movement into another hole. However, this may result in excessive tool switches and thus increments in tool switch time. On the other hand, one may decide to process all the holes which need the tool currently in use. Although, this decision will decrease the tool switch time, it can result in a huge increment in tool airtime

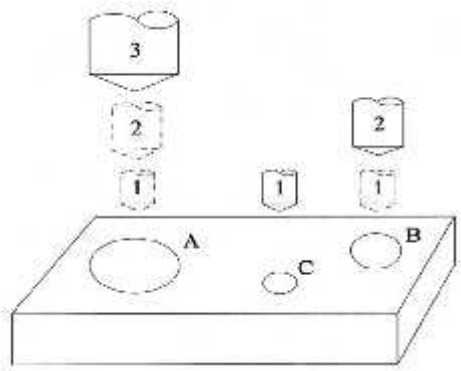


Figure 1. A schematic representation of alternative sets of tools for hole making

For each hole in Fig. 1 the largest tool, shown by solid lines, has to be used to drill the hole to its final size. Some pilot or intermediate tools, shown by dashed lines, may also be used. For instance, for hole A, there could be four different sets of tools; {1,2,3}, {2,3}, {1,3}, and {3}. The selection of tool set for each hole directly affects the required number of tools switches, and tool travel distance. The problem is now to select a set of operations along with the optimum sequence of those operations in such a way that the total processing cost is minimized.

The cost components considered in this paper include:

a) Tool travel cost: This is the cost of moving the tool from its previous location to the current drilling position. Tool travel cost is proportional to the distance required for the spindle to move between two consecutive drilling locations (Kolahan et al, 2000).

b) Tool switch cost: This cost occurs whenever a different tool is used for the next operation. If for any operation tool type is not available on the spindle, then the required tool must be loaded on the spindle prior to performing operation. This causes a longer tool switch time and hence a higher tool switch cost (Kolahan et al, 2000).

Problem formulation

The objective of interest in this paper is to minimize the summation of tool airtime and tool switching time and thereby to minimize the production cost. The following mathematical model (Kolahan et al, 1996) is used in this work.

$$Min\,(Z) = Min\,(y) \sum_{\substack{i=1\,j=1 \\ i \neq j}}^{k} \left[a\,p_{ij} + b\,q_{ij}\right] + penalty\,value \times No.\,of\,constraints\,voilations$$

The following notation is used in the proposed mathematical model.

i = tool type index, *i= 1,……,*

j = hole index, *j=1,……,*

k = number of possible operations in sequence

a = cost per unit non productive travelling distance in Rs/mm.

b = cost per unit tool switch time in Rs/min.

p_{ij} = non productive travelling distance between current hole and previous hole in mm.

q_{ij} = tool switch time between current tool and tool required by previous hole in minutes.

y = summation of travelling and switching cost in Rs.
Z= total cost.

Application example

This example presents the optimization of hole making of drilling operation where precedence of sequence is required. The proposed algorithm was coded in C-programme to determine the optimum sequence of operations for the part shown in Fig.3 which requires drilling operation. All types of moulds, moulded parts and plastic moulded goods are the applications of this industrial problems.

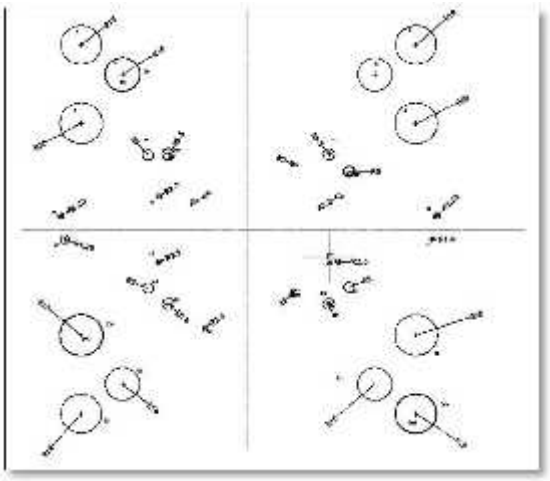


Figure 3 Top view of example part

The selection of tool set for each hole directly affects the optimum required number of tools switches, and tool travel distance. The problem is now to select a set of operations along with the optimum sequence those operations in such a way that the total processing cost is minimized. The process parameters are: *a=0.04 Rs/min* and *b=50 Rs/min.* The tool switch times are considered to be in the range of 0.2 to 0.5 minutes depending on the operator skills.The computation is carried out for 10 different starting sequences. To investigate the effect of genetic algorithm on search performance, the search was repeated for optimum result.

The steps of optimization using RCGA algorithm are discussed below.

4.1 Parameter setting

The following parameters as shown in Table 1 are set to give better result for optimization of hole-making problem to run RCGA.

Table 1. Parameters for Genetic Algorithm.

S.N	Parameters	Value
1	Population Size	10
2	Crossover Fraction	0.8
3	Mutation Fraction	0.2

4.2 Initialize the population

In the present work population size of 10 is considered. This example presents the optimization of hole making of drilling operation where precedence of sequence is required drilling operation must be followed by reaming if this is not happened then constraint violates.

The objective function value for the initial population is calculated by using formula,

Cost (Z) = Cost(y) + Penalty value × Number of constraint violation.

The penalty value is selected such that any solution violating the constraint (i.e. if precedence of operation is not maintained) should never appear in optimum sequence. For this particular problem penalty value is calculated 750. Following table shows objective function value and number of constraints violation for initial 10 sequences.

Table2. Initial population

S. N	Initial 10 random sequences	Z (Rs.)	No. of constraint violations
1	2r 5d 22d 3d 11d 30d 7d 23d 13d 19d 8r 27d 24d 4d 10d 1d 32d 28d 20d 2d 17d 12d 29d 27r 6d 16d 8d 25d 9d 32r 11r 15d 21d 14d 18d 26d 23r 31d	2453.5	2

2	23r 17d 22d 2r 14d 30d 26d 13d 6d 27r 8r 21d 10d 31d 18d 28d 15d 5d 12d 9d 19d 16d 24d 25d 29d 8d 11d 20d 23d 4d 3d 32d 7d 2d 11r 1d 27d 32r	3963.0	4
3	11r 18d 32r 4d 23d 1d 5d 8d 31d 19d 27d 7d 2d 20d 24d 13d 17d 28d 15d 26d 12d 29d 6d 32d 16d 10d 25d 11d 9d 23r 21d 14d 3d 22d 8r 2r 30d 27r	2515.7	2
4	27r 17d 29d 8d 2r 15d 22d 9d 30d 14d 32r 13d 20d 3d 7d 11r 28d 1d 26d 12d 18d 6d 21d 23r 25d 8r 31d 10d 16d 23d 27d 24d 5d 32d 4d 19d 2d 11d	4653.9	5
5	5d 28d 12d 23d 2d 24d 10d 7d 16d 32d 6d 11d 1d 22d 4d 21d 30d 25d 14d 31d 8d 27d 18d 13d 3d 19d 2r 29d 32r 26d 17d 8r 15d 11r 27r 20d 23r 9d	923.78	0
6	24d 30d 8r 9d 10d 19d 27d 20d 2d 21d 22d 15d 4d 32d 29d 16d 27r 18d 12d 17d 11d 7d 26d 31d 3d 32r 25d 14d 13d 11r 23d 28d 5d 1d 6d 8d 2r 23r	1710.9	1
7	2d 6d 13d 26d 17d 14d 4d 5d 7d 11r 1d 18d 29d 19d 25d 15d 31d 22d 23r 9d 2r 20d 8d 16d 28d 32d 10d 30d 12d 3d 27d 21d 24d 8r 11d 23d 32r 27r	2414.1	2
8	23r 32r 30d 25d 24d 22d 2d 10d 20d 3d 31d 9d 23d 27r 21d 12d 13d 32d 7d 17d 1d 11d 8d 4d 2r 28d 16d 15d 27d 14d 6d 19d 5d 8r 18d 29d 26d 11r	3142.8	3
9	19d 14d 28d 6d 4d 17d 8r 5d 32r 12d 29d 16d 23r 26d 9d 1d 20d 7d 15d 3d 10d 8d 18d 32d 24d 21d 11r 23d 25d 22d 30d 31d 2r 27r 13d 11d 27d 2d	5474.84	6
10	2d 18d 23d 15d 25d 21d 11d 22d 8d 3d 16d 27r 26d 24d 32d 5d 31d 12d 30d 7d 10d 1d 28d 8r 20d 14d 29d 4d 13d 2r 6d 27d 17d 9d 23r 19d 32r 11r	1705.09	1

4.3 Reproduction

In RCGA, the reproduction of the solution is based on the shared fitness value.

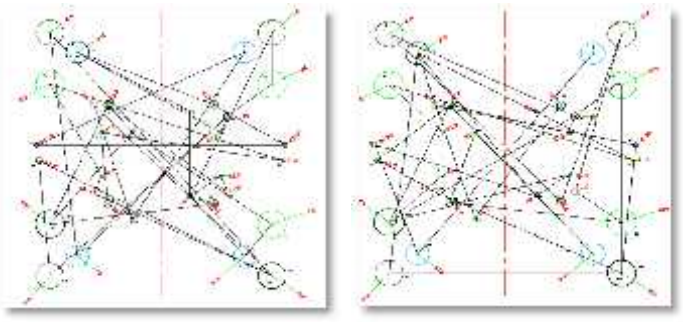
4.4 Updating the solutions

The new solution (offspring) is obtained using crossover and mutation. In the present work, single point crossover is used along with mutation in order to generate new population.
Steps 2 and 3 are repeated till the optimum sequence obtained which has minimum cost (Z). The following section provides the existing method used in actual industrial practice in hole making operations without any optimum sequence for case study.
The results of optimization using the proposed approach is compared with that obtained with current industrial practice and is presented is Table 3.

Table 2 Result of optimization

Method	**Sequence**	**Z (Rs.)**
Existing practice	5d 28d 12d 23d 2d 24d 10d 7d 16d 32d 6d 11d 1d 22d 4d 21d 30d 25d 14d 31d 8d 27d 18d 13d 3d 19d 2r 29d 32r 26d 17d 8r 15d 11r 27r 20d 23r 9d	923.78
RCGA	2d 10d 16d 12d 5d 24d 7d 32d 1d 28d 11d 6d 4d 23d 22d 27d 14d 30d 31d 8d 25d 13d 19d 3d 18d 26d 32r 17d 29d 11r 15d 2r 9d 23r 20d 21d 8r 27r	820

Fig. 4 (a) shows optimal sequence for making the holes in hole using RCGA whereas Fig. 4 (b) shows the existing sequence of machining holes.



(a) (b)

Fig. 4 Sequence of machining holes using RCGA and existing practice.

Simulated Annealing Approach for Robot Inverse Kinematic Solutions

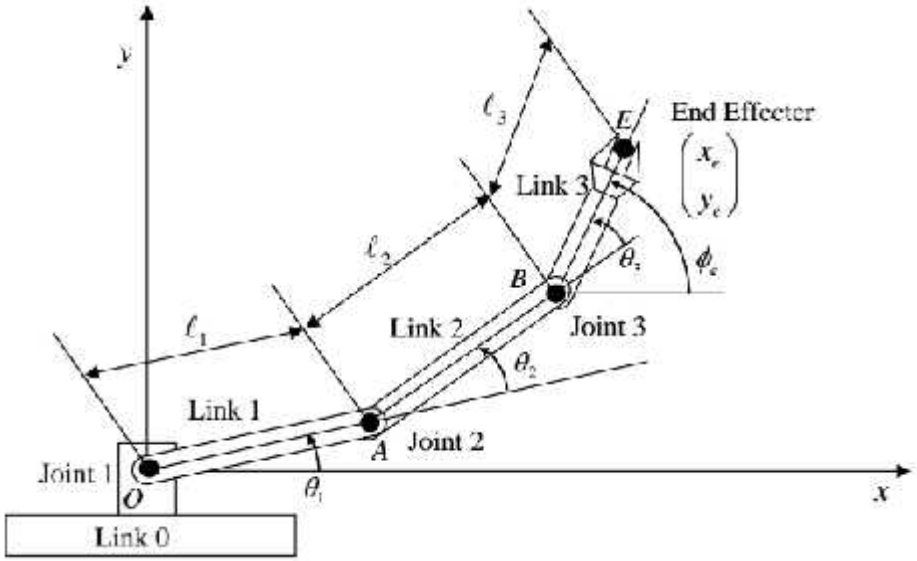
Dr. P. J. Pawar

K. K.Wagh Institute of Engineering Education and Research, Nashik, Maharashtra

Inverse kinematics

Forward kinematics: Forward kinematics refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters.

Inverse kinematics is the mathematical process of calculating the variable joint parameters needed to place the end effector

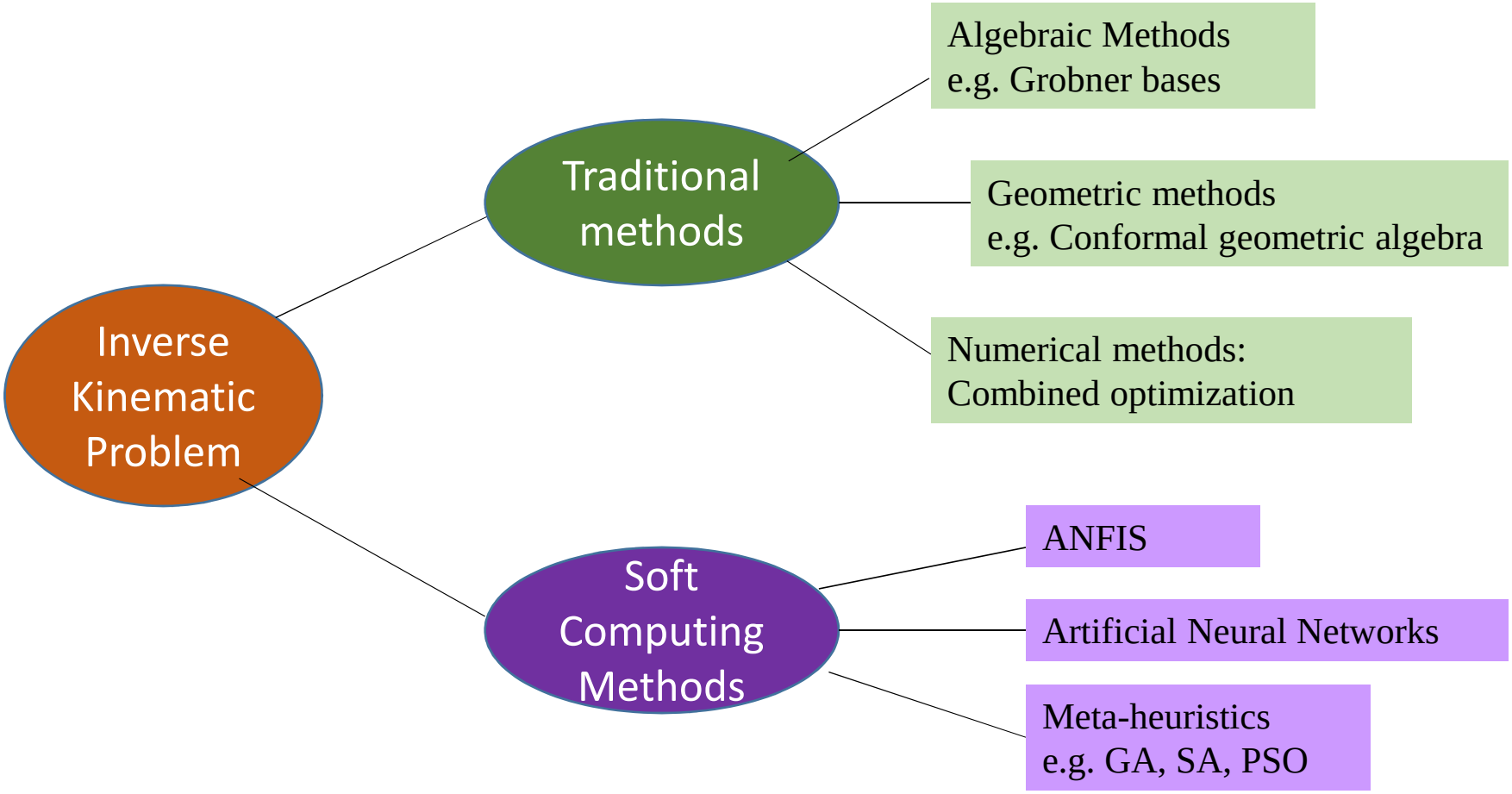


Need of soft computing to solve inverse kinematics problem

- Solving inverse kinematics is one of the most challenging problems in industrial robots.

- The complexity of inverse kinematics problem is decided by geometry of the robot arm and its nonlinear equations which describe mapping between joint and Cartesian space.

- In addition to the complexity of the inverse kinematics problem, solution must be very accurate to allow the robot to perform the task successfully



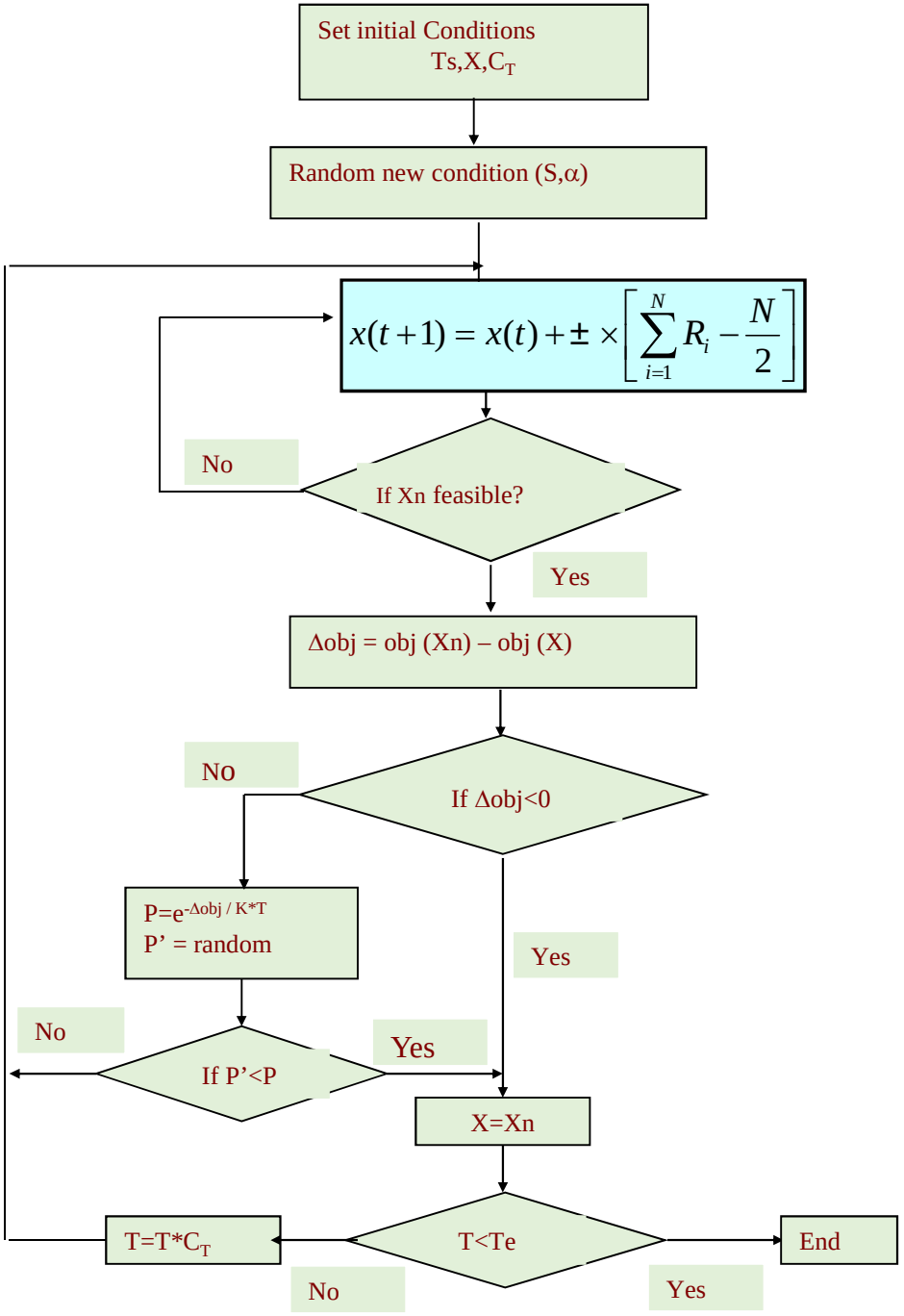
For complex robot structure involving large number of DOF as well as for long paths, the traditional methods are usually:

- Inadequate
- Very complex
- Very slow

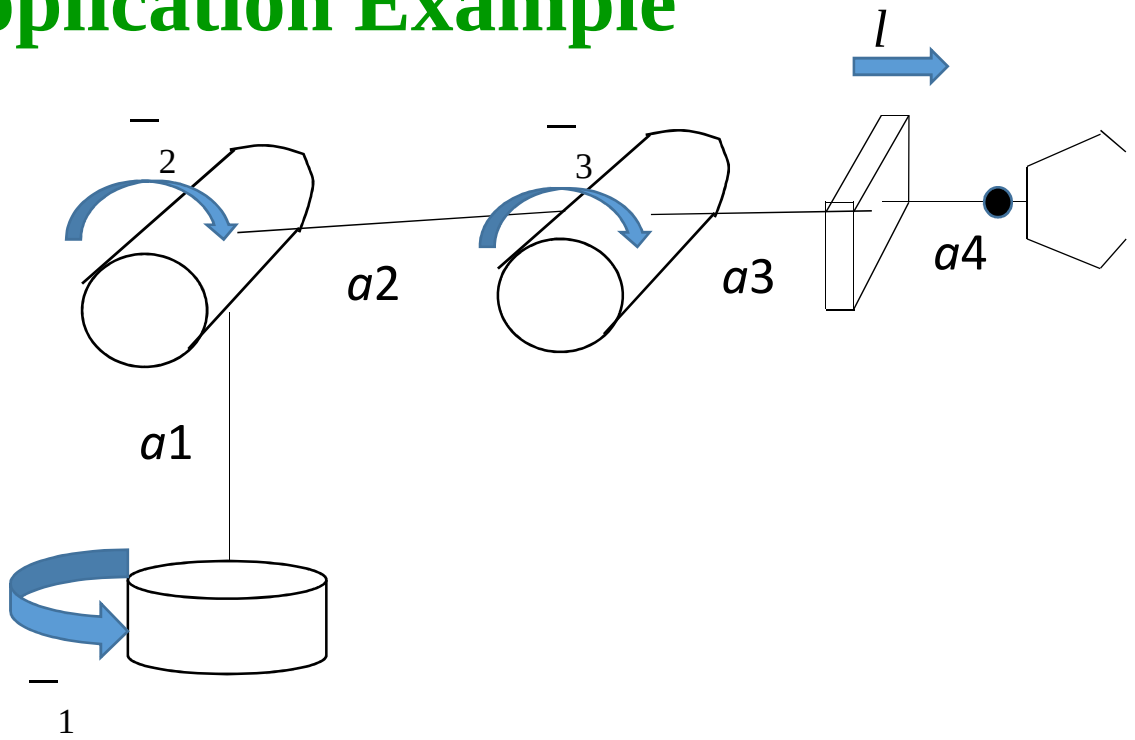
Simulated Annealing (SA)

- The simulated annealing algorithm developed by Kirkpatrick *et al.* (1983) resembles the cooling process of molten metals through annealing.
- The cooling phenomenon is simulated by controlling a temperature like parameter introduced with the concept of Boltzman probability distribution.
- If ‘i’ is the current configuration with cost C(i) then probability of accepting ‘j’ as next configuration with is:

$$\Pr\{new = j \mid current = i\} = \begin{cases} 1 & if \quad \Delta C \le 0 \\ e^{-\Delta C/T} & otherwise \end{cases}$$


Set initial Conditions
Ts,X,C_T
Random new condition (S,α)
$x(t+1) = x(t) + \pm \times \left[\sum_{i=1}^{N} R_i - \frac{N}{2} \right]$
No
If Xn feasible?
Yes
Δobj = obj (Xn) – obj (X)
No
If Δobj<0
$P=e^{-\Delta obj / K*T}$
P' = random
Yes
No
If P'<P
Yes
X=Xn
T=T*C_T
T<Te
End
No
Yes

Application Example



$a_1 = 70$
$a_2 = 100$
$a_3 = 50$
$a_4 = 40$

Joint No.	θ	α	d	r
1	θ_1	90	a_1	0
2	θ_2	0	0	a_2
3	θ_3+90	90	0	0
4	90	0	a_3+a_4+l	0

D-H Matrix

Cosθ	-sinθ.cosα	sinθ.sinα	r.cosθ
Sinθ	cosθ.cosα	-cosθ.sinα	r.sinθ
0	sinα	cosα	d
0	0	0	1

Transformation Matrix

Optimization Model

Min. $\sqrt{(x_c-250)^2 + (y_c - 150)^2+(z_c-200)^2}$

Variable and Variable bounds

$-90 \leq \bar{}_1 \leq 90$

$-90 \leq \bar{}_2 \leq 90$

$-90 \leq \bar{}_3 \leq 90$

$0 \leq l \leq 200$

Algorithm specific parameters

For SA:

- Initial temperature: 500°C,
- cooling rate: 10%,
- Termination temperature: 0.01°C.

Initial Solutions

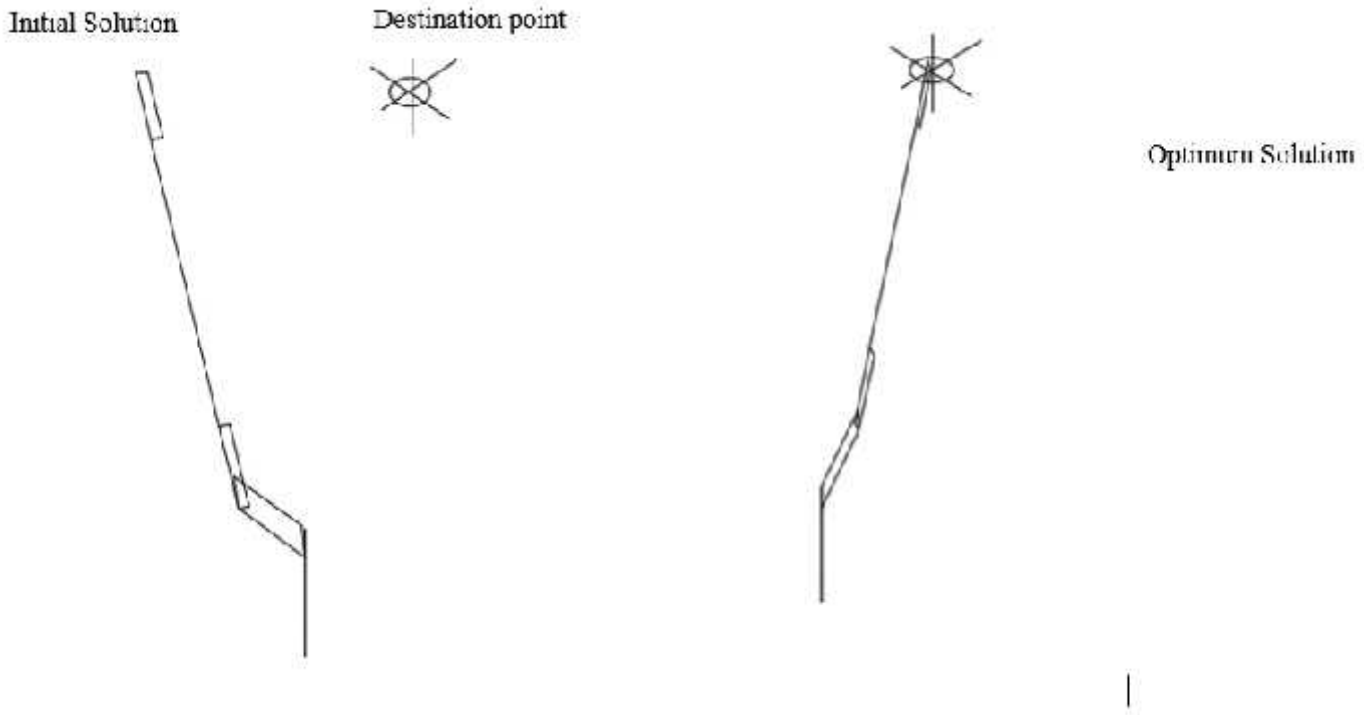
θ_1 (°)	θ_2 (°)	θ_3 (°)	l (mm)	e (mm)	End effector position (x, y, z)
73	-15	71	172	206.13	(71.36, 232.69, 261.15)

Updating global optimum solution using SA

$\bar{\theta}_1$ ($\hat{\imath}$)	$\bar{\theta}_2$ ($\hat{\imath}$)	$\bar{\theta}_3$ ($\hat{\imath}$)	l (mm)	e (mm)	End effector position (x, y, z)
62.06	-3.85	74.21	163.80	192.89	(86.99, 163.77, 302.19)

Results of optimization

Algorithm	$\bar{1}$ ($\hat{1}$)	$\bar{2}$ ($\hat{1}$)	$\bar{3}$ ($\hat{1}$)	l (mm)	e (mm)	End effector position (x, y, z)
SA	31.00	-5.88	42.09	148.00	0.19	(250.09, 150.08, 200.14)




Thank
You!