**Session 4: Local Search and Optimization Strategy**

1. **OBJECTIVES**

* To be able to understand hill-climbing local search and beam search strategies;
* To be able to implement simpler variants of hill-climbing and genetic algorithms in Prolog and in Python.

1. **Demonstration of Useful Resources**
2. **Developing a multi-modular system for Hill-climbing Local Search**

We take the 8-queens problem to demonstrate the working of the Hill-climbing search strategy. A state is represented as an eight-digit positive integer (with 1, 2, 3, …, 8 only). We generate all 56 successors of a current state, and choose the one that appears best as per a heuristic function. The process is repeated until a state with a specified value is found. Here is the possible outcome of a typical implementation of the algorithm.

For the initial state 23456578, with threshold value 27, after 3 iterations a solution was found in the following form:

Iteration max: 20

Iteration max: 24

Iteration max: 25

Found! Id:45 s [7,3,4,6,1,5,2,8] Value:27

And the states were as follows:

state(1, c, [7, 3, 4, 6, 1, 5, 7, 8], 25).

state(2, s, [1, 3, 4, 6, 1, 5, 7, 8], 23).

state(3, s, [2, 3, 4, 6, 1, 5, 7, 8], 24).

…

state(44, s, [7, 3, 4, 6, 1, 5, 1, 8], 25).

state(45, s, [7, 3, 4, 6, 1, 5, 2, 8], 27).

state(46, s, [7, 3, 4, 6, 1, 5, 3, 8], 24).

…

state(55, s, [7, 3, 4, 6, 1, 5, 7, 5], 25).

state(56, s, [7, 3, 4, 6, 1, 5, 7, 6], 24).

state(57, s, [7, 3, 4, 6, 1, 5, 7, 7], 23).

The system gets stuck up frequently at local maxima if the threshold value is set at 28. To avoid the local maxima we consider the following three variants of the algorithm:

1. **Random restart hill climbing:** If stuck up at a local maximum, then begin with a new randomly **generated** state.
2. **Stochastic Hill-climbing:** Choose one at random from among the uphill moves.
3. **Simulated annealing:** Choose one at random from among the successors. Allow an uphill successor directly, but sometimes allow a downhill one, with a given probability. [Temperature change may be taken from 25.0 to 0.0 with an interval of 0.1, and the formula for probability as eΔE/T, where ΔE means change in energy (downhill value) and T means temperature.]
4. **Developing a multi-modular system for a typical genetic algorithm**

We take a few states of the 8-queens problem as the initial population, set a threshold value for formation of parent generation, and also set a target value of the fitness function. Crossover in parent population is allowed, and sometimes mutation in some new individual is also allowed.

Sample initial population may look as follows:

intl\_sts(12345678).

intl\_sts(87654321).

intl\_sts(18273645).

intl\_sts(45362718).

intl\_sts(15263748).

intl\_sts(84736251).

intl\_sts(13572468).

intl\_sts(24681357).

Formation of new population and evaluation of the individuals are carried out until an individual with the target fitness is found.

1. **Lab Exercise**
2. Explore thoroughly the supplementary material provided for this session.
3. Run and analyze the codes demonstrated in this session.
4. With the help of the supplementary materials and demonstrated codes implement the variants of hill-climbing and genetic algorithms discussed above in Prolog and Python.