Assignment 2: Game Playing, Tabu Search and Simulated Annealing
(Due Date : June 20,2020- midnight)

This assignment can be approached in groups. Each group consists of three students.

Question 1:

We want to minimize the Easom function of two variables:

$$\min_{x} f(x) = -\cos x_1 \cos x_2 \exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right), \quad x \in [-100, 100]^2$$

The Easom function is plotted in Figure 2 for x∈ [−10,10]. The global minimum value is −1 at $x = (\pi, \pi)^T$. This problem is hard since it has wide search space and the function rapidly decays to values very close to zero, and the function has numerous local minima with function value close to zero. This function is similar to a needle-in-a-hay function. The global optimum is restricted in a very small region.

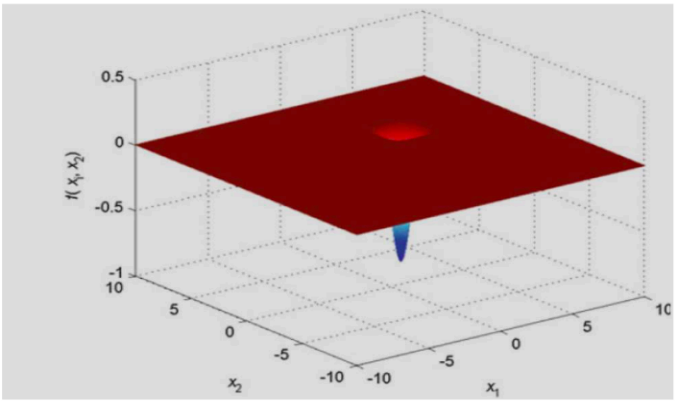


Figure 1: Easom function for x∈ [−10,10]

a. Find a proper problem formulation, neighborhood function and cost function for this problem to apply SA algorithm to solve it.
b. Report your observations on SA performance for solving this problem by:
    i. Selecting 10 different initial points randomly in [-100,100]
    ii. Selecting 10 different initial temperatures in a reasonable range
    iii. Selecting 10 different annealing schedules
c. What was the best solution you could find? What was the setting of SA to achieve that solution? Why was that setting performing better than other settings?

** You may use SA solver from MATLAB Global Optimization toolbox.

Question 2:
The game of Conga was developed by Matin Franke, and published by "Das Spiel Hamburg" in 1998. It is a two-player game played on a square 4X4 board, as shown in Figure1.

Player 1

(1,4)	(2,4)	(3,4)	(4,4)
(1,3)	(2,3)	(3,3)	(4,3)
(1,2)	(2,2)	(3,2)	(4,2)
(1,1)	(2,1)	(3,1)	(4,1)

Player 2

Figure 1: A Conga board and two players.

Initially, Player 1 has ten black stones in (1,4) and Player 2 has ten white stones in (4,1). The players alternate turns. On each turn, a player chooses a square with some of his stones in it, and picks a direction to move them, either horizontally, vertically or diagonally. The move is done by removing the stones from the square and placing one stone in the following square, two in the next one, and the others into the last one. The stones can only be moved in consecutive squares that are not occupied by the opponent; if a direction has less than three squares not occupied by the opponent in a row, then all remaining stones are placed in the last empty square. If a square has no neighbouring squares that are not occupied by the opponent, then the stones in that square cannot be moved.

You can move stones from a square to a (series of) neighbouring square(s), provided the squares you are moving to are not occupied by the opponent. It makes no difference if the squares are free or occupied by yourself. You do not need any of the squares you are moving to be free; they could all already be occupied by your other stones. The only distinction is between squares that are occupied by the opponent (which block you) and squares that are not occupied by the opponent (which you can move to). For example, let's say you have a number of stones in square (1,4) and you want to move them right.

There can be four possible cases:
1) The opponent occupies square (2,4). You cannot move right. That is the example in

Figure 4.
2) The opponent occupies square (3,4), but square (2,4) is either free or occupied by yourself. You move all the stones from (1,4) to (2,4).
3) The opponent occupies square (4,4), but squares (2,4) and (3,4) are either free or occupied by yourself. You move one stone from (1,4) to (2,4), and all other stones in (3,4). That is the example in Figure 3.
4) The opponent doesn't occupy any squares in that row, and all squares are either free or occupied by yourself. You move one stone from (1,4) to (2,4), two stones to (3,4), and all other stones in (4,4). That is the example in Figure 2.
The goal of the game is to block the opponent, so that he has no legal moves. In other words, all of the opponents' stones must be trapped in squares that are all surrounded by the player's stones.

Requirements
For this question, you must design and implement a computer program to play the Conga game. The agents implemented should use the Minimax search algorithm and Alpha-Beta pruning, as well as some evaluation function to limit the search. They should also have a reasonable response time. Since it is unlikely that you will discover an optimal evaluation function on the first try, you will have to consider several evaluation functions and present them in your report.
If you wish, you may implement improvements of the basic Minimax/Apha-Beta agent, such as an iterative deepening Minimax agent or the Null-Window Alpha-Beta pruning, for bonus points.
Your implementation should output information about the search, including the depth and the number of nodes explored.
In order to write the report, you will also need to implement a Random Agent, or an agent that always plays a random legal move.

Notes on the game:
1. While researching the game of Conga, you may realize that there is a second victory condition. A player can win by making a line four squares long containing the same number of stones. This victory condition has been removed from the game for this project, in order to simplify the evaluation function.
2. A consequence of this simplification is that the game is now much harder to win. In fact, two players of equal strength can play for hours without being able to end the game. However, it is still possible for a player to defeat a player of lesser skill, or for an agent to defeat another agent with a shallower search and a less accurate evaluation function. And it is definitely possible for a rational agent to defeat the Random Agent, in as little as 30 moves.
3. Given the requirement of making your agent play against the Random Agent, you may come up with the idea of somehow optimizing their agent to play better against that opponent specifically, by exploiting its random, non-optimal play. That is not acceptable for this project. The goal of the project is to design an agent that can play rationally against any opponent, not one that is optimized to play only against the Random Agent.

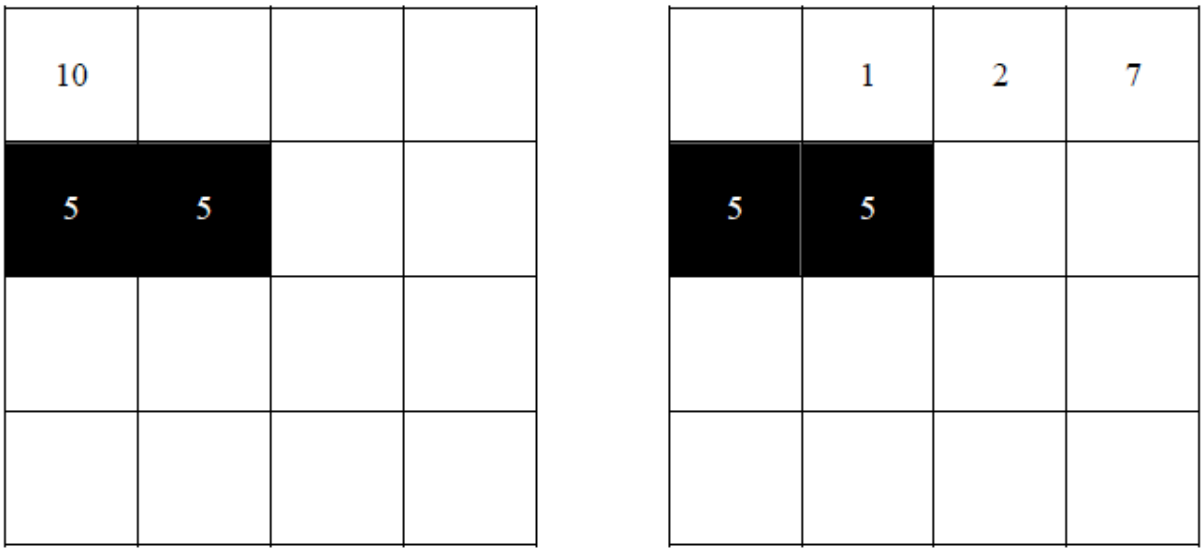


Figure 2: From the setup of the left board, white has only one legal move. The result of that move is shown in the right board.

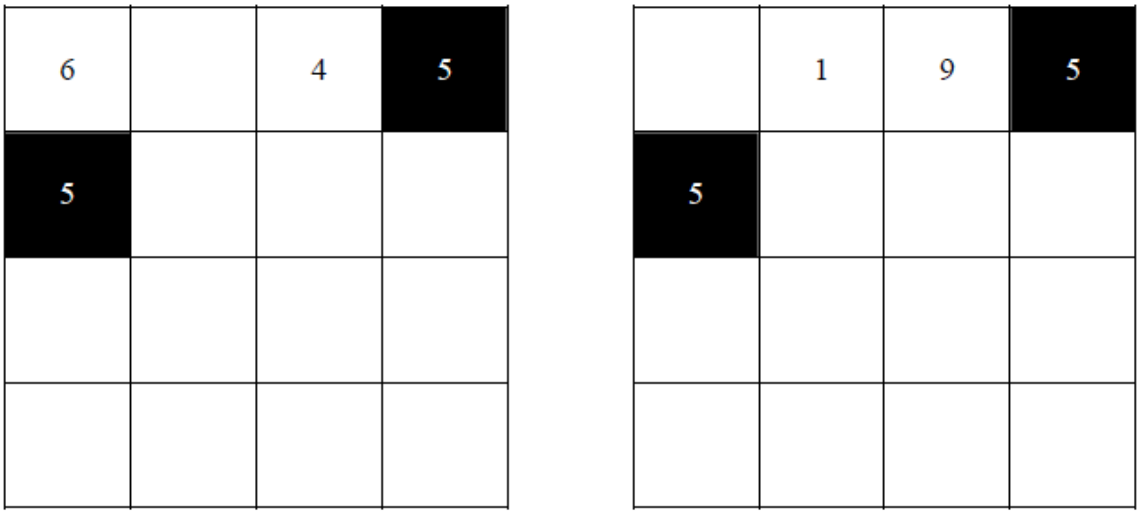


Figure 3: From the setup of the left board, white has six legal moves, and decides to move (1,4) to the right. The result of that move is shown in the right board.

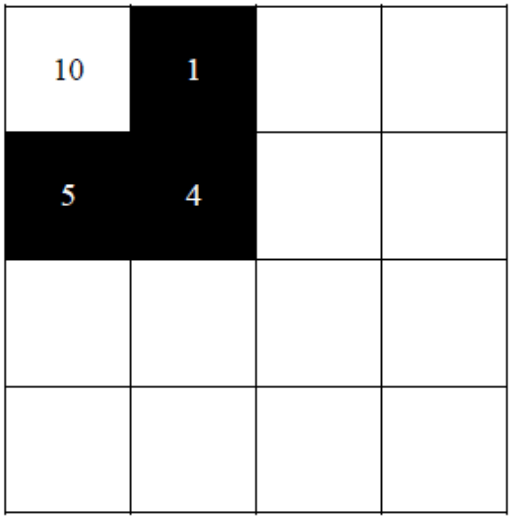


Figure 4: In the setup of this board, white has no legal moves. Black has won.

Question 3:

This is one of the QAP (quadratic assignment problem) test problems of Nugent et al. 20 departments are to be placed in 20 locations with five in each row (see below). The objective is to minimize costs between the placed departments. The cost is (flow * rectilinear distance), where both flow and distance are symmetric between any given pair of departments. Below are the flow and distance matrices, respectively. The optimal solution is 1285 (or 2570 if you double the flows).

1.1.1 Layout of department locations

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

1. Code a simple TS to solve the problem. To do this you need to encode the problem as a permutation, define a neighborhood and a move operator, set a tabu list size and select a stopping criterion. Use only a recency based tabu list and no aspiration criteria at this point.
2. Run your TS.
3. Perform the following changes on your TS code (one by one) and compare the results.
   - Change the initial starting point (initial solution) 10 times
   - Change the tabu list size – smaller and larger than your original choice
   - Change the tabu list size to a dynamic one – an easy way to do this is to choose a range and generate a random uniform integer between this range every so often

(i.e., only change the tabu list size infrequently)
- Add one or more aspiration criteria such as best solution so far, or best solution in the neighborhood, or in a number of iterations
- Use less than the whole neighborhood to select the next solution
- Add a frequency based tabu list and/or aspiration criteria (designed to encourage the search to diversify)

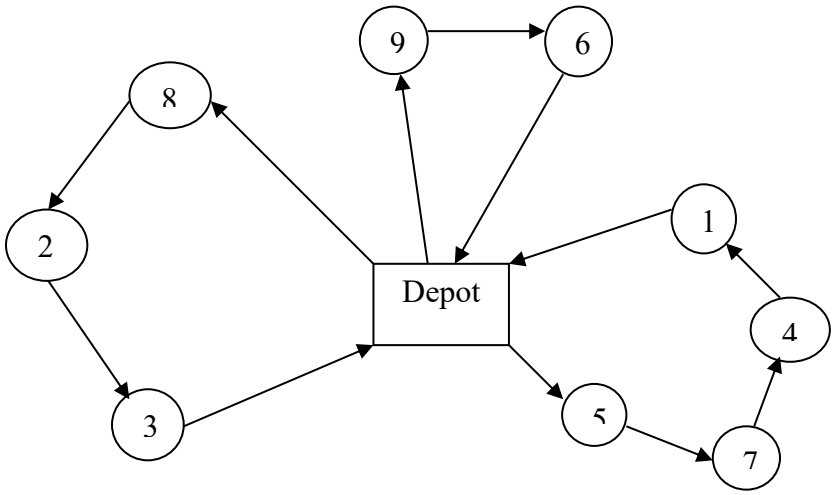
4. Requirements and deliverables (as above)

Question 4:

Simulated Annealing can be used to solve The Vehicle Routing Problem (VPR) defined by having *m* vehicles at a depot that need to service customers in ***c*** cities. The travel distances between every two cities are defined by a matrix ***D*** with element d_{ij} denoting distance between cities *i* and *j*. The travel distances from the depot to each of the cities are given by a vector ***Depot***. Each customer *j* has a service time s_j The VPR consists of determining the routes to be taken by a set of ***m*** vehicles satisfying the following conditions:

- Starting and ending at the depot,
- Having minimum cost (travel+service),
- Each customer is visited exactly once by exactly one vehicle.

The figure below illustrate possible routes for a 9 customers and 3 vehicles problem



a) Assuming that the number of required routes is fixed and that it's equal to the number of vehicles. Simulated Annealing could be applied in order to solve this problem, it's required to:
   i. Find a suitable solution representation ,
   ii. Define a suitable neighborhood operator,
   iii. Define the objective function used to calculate the cost of a solution .

iv. Show how your solution performs on instance A-n39-k6.vrp which can be found at NEO site: http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/ and compare your best solution to the reported optimal solution.

b) How would the problem formulation change if we add the constraint that the total duration of any route shouldn't exceed a preset bound T?

Load Adjustment:

If you are attempting this assignment as a group, then you are required to solve all questions. If you are doing it solo, then you can ignore either question 1 (re continuous SA), or question 4 (re combinatorial SA).

Deliverables:

As to the deliverables: please ignore what is in the assignment, given the circumstances.

The requirements are, applicable to all questions, unless otherwise specific requirements are stated in the question.:

You are expected to turn in a short paper (the shorter the better) which should include the following:
1) A short description of your program.
2) A description of your implementation of the search agent with memory and time complexity analysis.

3) Sample output of your program, to demonstrate its functioning.

You must also upload the code you implemented to the assignment dropbox.

Your grade in this question will be function of the quality and completeness of the report you hand in.

Marking Scheme:

- **The four questions are equally weighted**
- **Solution description (40%): strategy (20%), algorithm/pseudo code (15%),**
- **Code and output sample 30%**
- **Memory and time complexity analysis 20%**
- **Working code (demo) 15%**