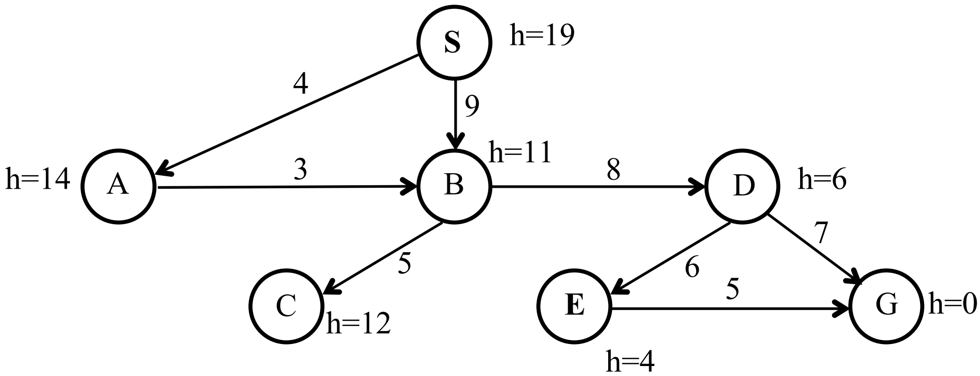
Homework 1: Search

Due date: Sep/30

1) (30 pts) Consider the state space given in the graph below. Step costs are shown near to each arc. Heuristic values *h* are shown near each node. The reachable successors of each node are indicated by the arrows connecting the nodes. Successors should be returned in a sequence from left to right, i.e., select nodes in alphabetical order, in this case.



Execute tree search using the following strategies: Breadth-First search, Depth-First search, Greedy Best-First search and A* search.

For each search strategy:

a. List the order in which nodes will be are expanded (in other words, to expand a node means that its successors are generated), ending with the goal node that is found.

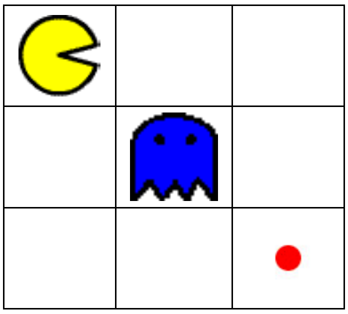
b. List the path from start to goal.

c. Calculate the cost of the path found to reach the goal.

Finally, is the heuristic admissible? Defend.

Tip: You will need to compare the heuristic value and the cost of reaching the goal for each node. To be admissible, the heuristic estimate for any given node should not be higher than the lowest possible cost to reach the goal from that node.

2) (30 pts) In this Pacman-like game, Pacman is playing against a goblin that is trying to block its path. You will use the minimax algorithm to help Pacman. Each player can move to an unoccupied cell in one of the four cardinal directions (N, S, E, W). The players take turns making one move at a time, but they cannot stay in the same cell, they must move to a different cell. Pacman wins the game by eating the dot, which gives him a score of 1. The goblin cannot score points by eating Pacman, it can only block Pacman's moves. The game begins in the configuration shown in the figure below. The first to move is Pacman.



a. Draw a game tree limited to two moves for each player. Tip: Consider only the legal moves, i.e. players cannot move outside the 3x3 grid.

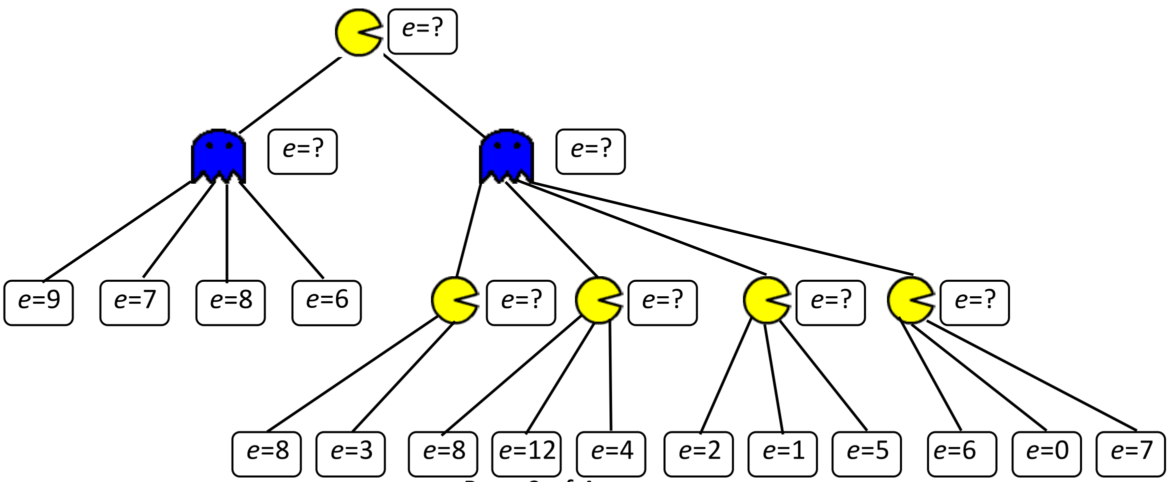
b. What is the value of the game considering the game tree with two moves you drew in (a)? Tip: Your evaluation function must use Pacman's score, i.e., number of dots he has eaten. Defend.

c. What would be the value of the game if you were to expand the game tree to up to ten moves for each player, instead of just two? Tip: Consider if Pacman is able to eat the dot in moves. Defend.

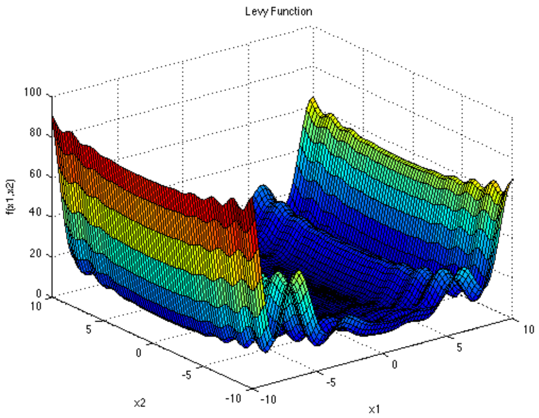
   Tip: As in the standard Pacman, the game ends when there are no more dots, and the ghost can pass through dots. Only (a) requires you to draw the game tree.

Consider now a second game on a more complicated grid. An unknown evaluation function *e* is used to draw a (partial) game tree as shown below.

d. Using the minimax algorithm, calculate the values of all internal nodes for each player. Tip: Fill in the missing values in the boxes.
e. Using alpha-beta pruning, what are the nodes that need not be evaluated, assuming a standard left-to-right traversal of the search tree. Tip: You must provide a step-by-step description of running alpha-beta pruning on the game tree, showing values of alpha and beta at each step, and explaining why some nodes can be pruned.

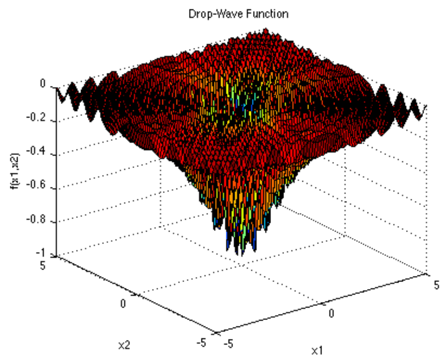


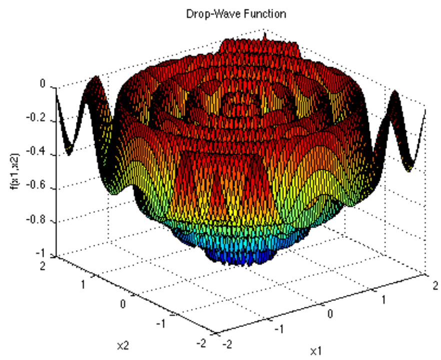
3) (40 pts) In this coding exercise you will compare and contrast two local search methods for function optimization: simulated annealing and genetic algorithms. The functions to be minimized are the 1D Levy function and the Drop-Wave function, see matlab code provided. They are illustrated below.



$$f(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2\left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_d - 1)^2\left[1 + \sin^2(2\pi w_d)\right], \text{ where}$$

$$w_i = 1 + \frac{x_i - 1}{4}, \text{ for all } i = 1, \ldots, d$$





$$f(\mathbf{x}) = -\frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{0.5(x_1^2 + x_2^2) + 2}$$

Use the algorithm descriptions found in the textbook and in the MATLAB documentation. Experiment with each algorithm to find algorithm parameters that work well. Spend some time to make each algorithm work as well as you can on this particular function. The bounds for the search should be $x_i \in [-10, 10]$ for the Levy function and $x_i \in [-5, 5]$ for the Drop-Wave function, where *i* is dimension. Starting location should be random.

Start with the Levy function using 1 dimension. To see what each algorithm is doing, plot the function being optimized using the command FPLOT (or for two dimensions PLOTOBJECTIVE). On the same graph, plot marks that show the progression of values of (x1, x2) that are sampled by the search algorithm at fixed intervals, say, beginning, middle and end of search procedure. The first points sampled will appear high in the graph and the last points will appear near the bottom of the graph. This will show you the progression of the search algorithm by showing where the focus of the search shifts with experience. For the genetic algorithm, for example, you should see more and more points clustering around the lowest trough as the marks are plotted lower on the graph.

Once you are satisfied with the algorithms' parameter values, run each algorithm repeatedly, for 30 times. During each step of the search, keep track of the best fitness found so far.

a. Plot the best value so far, averaged over all repetitions, versus the number of values evaluated to show how quickly, on average, each algorithm finds better values. Make a plot of these averaged curves from the two algorithms. Tip: Your plot should show the best average value at each iteration (search step) of the algorithm.

b. Include at least one graph for each algorithm showing the points evaluated superimposed on the 1D Levy function. Tip: Your plot should show the progression of the search, from the beginning to middle, and the end of the search.

c. Describe your results comparing the performance of the two algorithms and explain why your results are the way they are. Add any additional graphs that demonstrate your findings.

Tip: You can use matlab's standard implementation of both algorithms, functions SIMULANNEALBND() and GA(). You experiment with the algorithm's parameters (e.g. temperature schedule and population size) to improve the average performance.

What to hand in

Upload to myCourses' dropbox two separate files: your write-up and responses in PDF, and your Matlab scripts (zipped if more than one file).
You can scan your handwritten response (or take a photo with your cellphone) instead of typing in the computer. Include your name as part of the filename.