National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 4: Local Search

Issued: Week 5 Discussion in: Week 6

Important Instructions:

- **Assignment 4** consists of **Question 3** from this tutorial.
- Your solution(s) must be TYPE-WRITTEN, though diagrams may be hand-drawn.
- You are to submit your solution(s) during your **Tutorial Session in Week 6**.

Note: you may discuss the assignment question(s) with your classmates, but you must work out and write up your solution individually. Solutions that are plagiarised will be heavily penaltised.

---

1. Suppose that you would like to move into a new apartment, and you have n items, $a_1, ..., a_n$, each with size $s(a_i) > 0$. There are m boxes, $b_1, ..., b_m$, which you could use to help you transport your items, each with capacity $c(b_i) > 0$. Assume that $\sum_{i=1}^{m} c(b_i) > \sum_{j=1}^{n} s(a_j)$.

   Your goal is to pack all of your items into as few boxes as possible. You are allowed to put as many items as you want into a box, as long as the sum of their sizes does not exceed the capacity of the box. Formulate this as a local search problem.

2. The travelling salesman problem (TSP)[3] is a well-known problem in computer science that asks the following question:

   "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once, and returns to the origin city?"

   It can be solved with the minimum spanning tree (MST) heuristic, which estimates the cost of completing a tour through all the cities, given that a partial tour has already been constructed. The MST cost of a set of cities is the sum of the edge weights (i.e., distance between two cities) of any minimum spanning tree that connects all the cities.

[3]See TSP (`https://en.wikipedia.org/wiki/Travellingsalesmanproblem`) for more information.

(a) Show how this heuristic can be derived from a relaxed version of the TSP.

(b) Determine whether this heuristic is an admissible heuristic.

(c) Suggest a hill-climbing algorithm to solve TSP.

3. Suppose you are given a 3×3 board with 8 tiles, where each tile has a distinct number between 1 to 8, and one empty space, as shown in Figure 1. Your goal is to reach the goal state, shown in Figure 2.

2	1	3
8	6	4
7	5	

Figure 1: An initial state of the puzzle.

1	2	3
8		4
7	6	5

Figure 2: The goal state of the puzzle.

There are specific rules to solve this problem:

- An action comprises of moving the empty space to a position with a numbered tile. With such a move, the numbered tile in question will now take the place of the empty space's previous position.
- The empty space can only move in four directions, i.e., up, down, right, or left. It may not move diagonally.
- The empty space can only be moved by one position at a time.

You are provided with a cost function associated with every state, given by

$$f(\text{state}) = \text{number of mismatched tiles compared to the goal state}$$

You will move if and only if the cost of the next state is less than or equal to the cost of the current state. If there are different possible actions, you will always choose the action that leads to the state with the lowest cost.

(a) Given the following initial state in Figure 3, list the sequence of actions taken by the hill-climbing (steepest descent) algorithm to achieve the goal state given in Figure 2. Further, specify if the trace terminated at a local or global minima.

2	3	
1	8	4
7	6	5

Figure 3: The initial state of the puzzle for Question 3(a).

(b) Given the initial state in Figure 4, list the sequence of actions taken by the hill-climbing (steepest descent) algorithm to achieve the goal state given in Figure 2. Further, specify if the trace terminated a valid solution (i.e., goal state). Further, specify if the trace terminated a valid solution (i.e., goal state). Further, consider if a local or global minima was achieved.

2	3	
1	7	4
8	6	5

Figure 4: The initial state of the puzzle for Question 3(b).

4. Let G be the simple graph shown below. Our goal is to find a colouring of the set of vertices using only the colours **RED**, **YELLOW**, and **BLUE**, so that no two adjacent vertices are assigned the same colour.

   For the sake of simplifying your solutions, you may model the problem with the set of variables $c_A, c_B, ..., c_H$, so for example, $c_B =$ **RED** would denote that the color assigned to vertex B is red.

   (Note: one of the edges shown in Figure 5 is coloured green merely to highlight that the overlapping edges are distinct from each other; there are no additional constraints for that edge.)

   (a) Give an example of a solution state for the graph G if it exists.

   (b) You are provided with a cost function associated with every state, given by

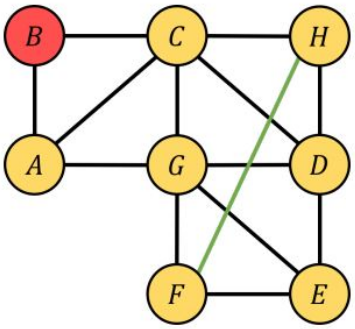

Figure 5: Graph G, shown here for Question 4.

$$f(\text{state}) = \text{number of pairs of adjacent vertices with the same colour}$$

Each step consists of changing the colour of a single vertex. You will take a step if and only if the cost of the next state is less than or equal to the cost of the current state. If there are different possible actions, you will always choose the action that leads to the state with the lowest cost. (If there are ties, you may decide any of the tied actions to take.)

By using the hill-climbing (steepest descent) algorithm, give a sequence of steps that would help you arrive at your aforementioned solution state. Alternatively, provide an explanation if you believe that graph G does not have a solution state, or that there is no sequence of steps that leads to your solution state.