CS3243 Introduction to Artificial Intelligence

AY2021/2022 Semester 1

Tutorial 3: Local Search

Important Instructions:

- **Assignment 3** consists of **Question 4** and **Question 5** from this tutorial.
- Your solutions for the above questions may be handwritten or typewritten, but handwritten solutions must be legible for marks to be awarded. If you are submitting handwritten solutions, please append the question paper in front of your solutions.
- You are to submit your solutions on LumiNUS by **Week 6 Monday (13 September), 2359 hours**, under **Tutorial Submission** → **Tutorial 3**.
- Questions marked with an asterisk (*) are harder and unlikely to appear in tests/exams.

Note: you may discuss the content of the questions with your classmates, but you must work out and write up your solution individually - solutions that are plagiarised will be heavily penalised.

Tutorial Questions

(1) Suppose that you would like to move into a new apartment, and you have n items $\{a_1, \ldots, a_n\}$, each with size $s(a_i) > 0$. There are m boxes $\{b_1, \ldots, b_m\}$ 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.

Solution: Note that there isn't a unique solution to this question. There exist multiple ways to formulate the given problem as a local search problem for different neighbour selection strategies and valuation functions. Here, we will discuss one possible solution, whose cost function is suggested by Hyde et al[1].

Before discussing the basic idea, let us define some notations:

- There are n items $\{a_1, \ldots, a_n\}$, each with size $s(a_i)$.
- There are m boxes $\{b_1, \ldots, b_m\}$, each with capacity $c(b_i)$.
- $x_{i,j}$: represents the i^{th} item in the j^{th} box, where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. If the i^{th} item is packed in the j^{th} box, then $x_{i,j} = 1$. Otherwise, $x_{i,j} = 0$.

[1] Matthew Hyde, Gabriela Ochoa, T Curtois, and JA Vazquez-Rodrguez. A hyflex module for the one dimensional bin-packing problem. School of Computer Science, University of Nottingham, *Tech. Rep*, 2009.

The cost function given by Hyde et al. (which you may treat as a black box) is given as:

$$\text{val(state)} = 1 - \left[\frac{\sum_{j=1}^{m} \left(\frac{\sum_{i=1}^{n} (x_{i,j} \times s(a_i))}{c(b_j)} \right)^2}{m} \right]$$

Basically, the function puts a premium on boxes that are filled completely, or nearly so. It returns a value between zero and one, where lower is better, and a set of completely full boxes would return a value of zero.

Assume that all boxes are sorted in non-increasing order of their capacities. Without loss of generality, we will assume the order of boxes to be $b_1, b_2, \ldots, b_m$. The basic idea of the problem formulation is given below:

Initial state:

- Generate a random sequence of items.
- Then, we pack each item into boxes using the *First Fit* heuristic[2].
- Evaluate the value of the initial state using the above equation.

Finding the next state:

- We choose a box randomly from the boxes used in the *current_state*. Let the chosen box be b_{empty}.
- next_state $\leftarrow$ remove the items from b_{empty}, and repack them into boxes used in *current_state* using the *First Fit* heuristic.
- Evaluate the value of *next_state* using Hyde et al.'s equation.

Stopping criteria:

- If val(next_state) $\leq$ val(current_state), then set *next_state* as *current_state*, and repeat the process.
- If val(next_state) $>$ val(current_state), then terminate the process, and return *current_state* as the solution.

* (2) The travelling salesman problem (TSP)[3] is a well-known problem in computer science which 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.

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

Solution: The TSP problem is to find a minimal length path through the cities, which forms a closed loop. MST is a relaxed version of the TSP problem because it only seeks for a minimal length graph that need not be a closed loop – it can be any fully-connected graph.

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

[2] The heuristic is as follows: for each item in the sequence, we attempt to place the item in the first box that can accommodate the item. If no box is found, we will open a new box and put the item into the new box.

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

Solution: As a heuristic, MST is admissible because it is always shorter than or equal to the length of a closed loop through the cities.

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

Solution: Here is a possible hill-climbing algorithm to solve TSP:

- Connect all the cities into an arbitrary path.
- Pick two points along the path at random.
- Split the path at those points, producing three partitions.
- Try all six possible ways to connect the three partitions.
- Keep the best way, and connect the path accordingly.
- Iterate the steps above until no improvement is observed for k (an arbitrary number of) iterations.

(3) Consider the state space given below. Our starting state is the position denoted as S, and our goal is to find the state which maximizes our value (objective function). For each the following search algorithms, explain whether the algorithm terminates, and whether the algorithm is able to find the global maximum.

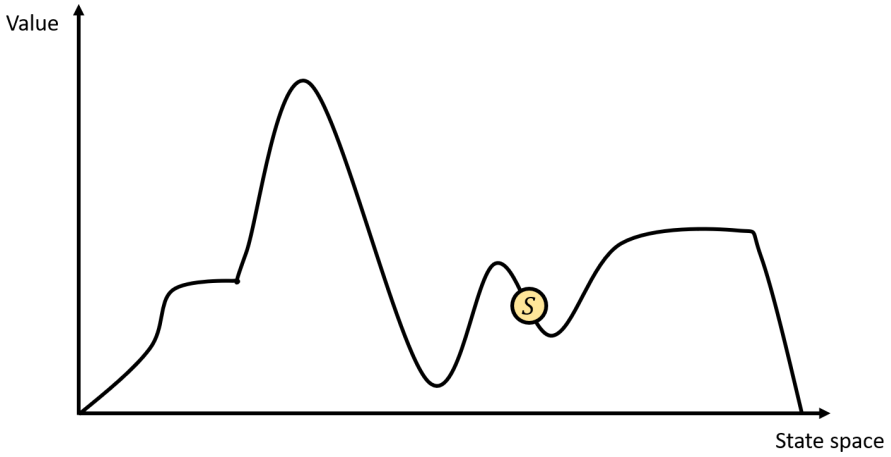

FIGURE 3.1. State space for question 3.

(a) Hill-climbing.

Solution: Hill-climbing search will terminate at the local maximum to the left of S.

(b) Simulated annealing.

Solution: Using simulated annealing, our algorithm *may* be able to terminate at the global maximum. However, there is also a chance that the algorithm may end up in a local maxima instead.

Assignment Questions

(4) 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 3.2. Your goal is to reach the goal state, shown in Figure 3.3.

2	1	3
8	6	4
7	5	

Figure 3.2. An initial state of the puzzle.

1	2	3
8		4
7	6	5

Figure 3.3. 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.
- The empty space can only move in four directions, i.e. up, down, right, or left, and it cannot 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 start state, show each step required to achieve the goal state given in Figure 3.3.

2	3	
1	8	4
7	6	5

Figure 3.4. Initial state of the puzzle for question 3(a).

Solution: Cost of the current state is 4, as 4 tiles are misplaced with respect to the goal state. Since the empty space is at the top right corner, only two moves are possible: left & down.

- With left move: $f(\text{next-state}) = 3$.
- With down move: $f(\text{next-state}) = 5$.

Since we have to choose a move which will give rise to a state whose cost is less than or equal to the current state, we will choose **left**, and obtain the following state:

Here, three moves are possible: left, right, & down.

- With left move: $f(\text{next-state}) = 2$.

2		3
1	8	4
7	6	5

- With right move: $f(\text{next-state}) = 4$.
- With down move: $f(\text{next-state}) = 3$.

Hence, we will choose **left** again.

	2	3
1	8	4
7	6	5

Now, two moves are possible: right & down.

- With right move: $f(\text{next-state}) = 3$.
- With down move: $f(\text{next-state}) = 1$.

Therefore, we will choose **down**.

1	2	3
	8	4
7	6	5

Here, three moves are possible: up, down, & right.

- With right move: $f(\text{next-state}) = 0$.
- With up move: $f(\text{next-state}) = 2$.
- With down move: $f(\text{next-state}) = 2$.

Hence, we will choose **right**, and reach the goal state.

(b) Given the following start state, show each step required to achieve the goal state given in Figure 3.3.

2	3	
1	7	4
8	6	5

FIGURE 3.5. Initial state of the puzzle for question 3(b).

Solution: Cost of the current state is 5, as 5 tiles are misplaced with respect to the goal state. Since the empty space is at the top right corner, only two moves are possible: left & down.

- With left move: $f(\text{next-state}) = 4$.
- With down move: $f(\text{next-state}) = 6$.

Since we have to choose a move which will give rise to a state whose cost is less than or equal to the current state, we will choose **left**, and obtain the following state:

Here, three moves are possible: left, right, & down.

- With left move: $f(\text{next-state}) = 3$.
- With right move: $f(\text{next-state}) = 5$.

2		3
1	7	4
8	6	5

- With down move: $f(\text{next-state}) = 4$.

Hence, we will choose **left** again.

	2	3
1	7	4
8	6	5

Now, two moves are possible: right & down.

- With right move: $f((next - state)) = 4$.
- With down move: $f(\text{next-state}) = 2$.

Therefore, we will choose **down**.

1	2	3
	7	4
8	6	5

Here, three moves are possible: up, down, & right.

- With right move: $f(\text{next-state}) = 2$.
- With up move: $f(\text{next-state}) = 3$.
- With down move: $f(\text{next-state}) = 1$.

Hence, we will choose **down** again.

1	2	3
8	7	4
	6	5

Now, two moves are possible: right & up.

- With right move: $f(\text{next-state}) = 2$.
- With up move: $f(\text{next-state}) = 2$.

For both moves, the cost of the next state increases from 1 to 2, and thus we are unable to choose any of the moves. We are therefore stuck in a local minima, and cannot reach the goal state.

(5) Let G be the simple graph shown below. Our goal is to find a coloring 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 $c_A, c_B, \ldots, c_H$, 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 3.6 is coloured green merely to highlight that the overlapping edges are distinct from each other; there are no additional constraints for that edge.)

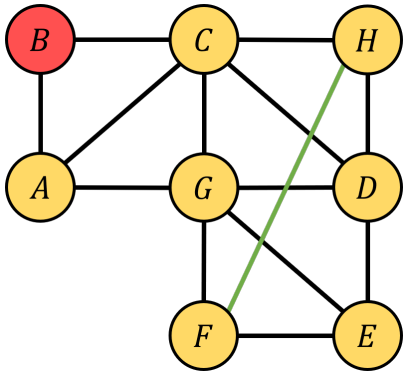

FIGURE 3.6. Graph G, shown here for question 5.

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

Solution:

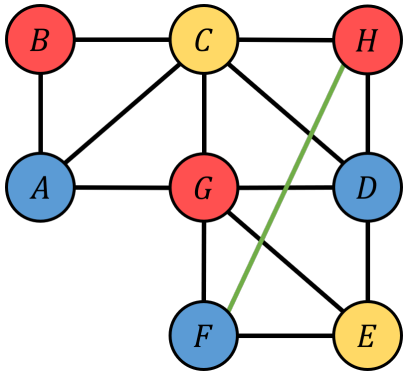

FIGURE 3.7. A possible solution state.

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

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

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.)

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.

Solution: