National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 4: Local Search

SOLUTIONS

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 the following.

$$\sum_{i=1}^{m} c(b_i) > \sum_{j=1}^{n} s(a_j)$$

Your goal is to pack all 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 notation:

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

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

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

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 $b_1, ..., b_m$. The basic idea of the problem formulation is thus as follows.

Initial state

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

[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, Technical Report, 2009.
[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.

Finding the next state

- Choose a box randomly from the boxes used in the *current_state*.
  Let the chosen box be b_{empty}.
- *next_state* ← remove the items from b_{empty} and repack them into boxes used in current state using the ***First Fit heuristic***.
- Evaluate the value of the next state using the equation defined by Hyde et al.

Stopping criteria

- If *value*(*next_state*) ≤ *value*(*current_state*), then set *next_state* as *current_state*, and repeat the process.
- Else, if *value*(*next_state*) > *value*(*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 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.

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

   Solution: The TSP requires us 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 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.

   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: The following 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] Refer to the following link for more on information on the TSP. https://en.wikipedia.org/wiki/Travellingsalesmanproblem

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: An goal state of the puzzle.

There are specific rules to solve this problem:

- An action comprises moving the empty space to an adjacent 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(*state*) = 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).

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(*next_ state*) = 4.
- With down move: f(*next_state*) = 6.

Since we must choose the move that gives rise to a state whose cost is less than or equal to the current state, we will choose **left**, and obtain the following state.

2		3
1	7	4
8	6	5

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

- With left move: f(*next_ state*) = 2.
- With right move: f(*next_ state*) = 4.
- With down move: f(*next_ state*) = 3.

Hence, we will choose **left** again. We thus obtain the following state.

	2	3
1	8	4
7	6	5

Now, two moves are possible: right, & down.

- With right move: f(*next_ state*) = 3.
- With down move: f(*next_ state*) = 1.

Therefore, we will choose **down**. We thus obtain the following state.

1	2	3
	8	4
7	6	5

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

- With right move: f(*next_ state*) = 0.
- With up move: f(*next_ state*) = 2.
- With down move: f(*next_ state*) = 2.

Hence, we will choose **right** and reach the goal state. This is a **global minima**.

(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, 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).

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(*next_ state*) = 4.
- With down move: f(*next_state*) = 6.

Since we must choose the move that gives rise to a state whose cost is less than or equal to the current state, we will choose **left**, and obtain the following state.

2		3
1	7	4
8	6	5

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

- With left move: f(*next_ state*) = 3.
- With right move: f(*next_ state*) = 5.
- With down move: f(*next_ state*) = 4.

Hence, we will choose **left** again. We thus obtain the following state.

	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(*next_ state*) = 2.

Therefore, we will choose **down**. We thus obtain the following state.

1	2	3
	7	4
8	6	5

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

- With right move: f(*next_ state*) = 2.
- With up move: f(*next_ state*) = 3.
- With down move: f(*next_ state*) = 1.

Hence, we will choose **down** again. We thus obtain the following state.

1	2	3
8	7	4
	6	5

Now, two moves are possible: right, & up.

- With right move: f(*next_ state*) = 2.
- With up move: f(*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 minima that does not correspond to a goal state. However, we notice that this is a **global minima** since the given puzzle is not solvable!

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 colour assigned to vertex B is red.

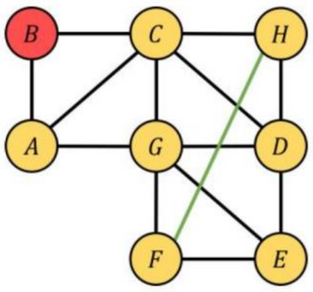
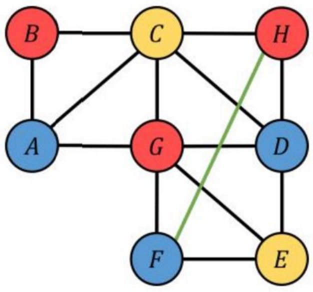


Figure 5: Graph G, shown here for Question 4.

(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.

Solution: A possible solution is as follows.



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

f(*state*) = 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.

Solution: A possible solution is as follows.