AI Assignment-3 Report

Team -14
Tabish Khalid Halim ,
200020049
Anand Hegde ,
200020007

Department of Computer Science, IIT Dharwad

January 10, 2022

Contents

1 Problem Statement **1**

2 Approach **1**

3 Pseudo Code **2**

4 Heuristic function **3**

5 Beam Search Analysis **4**

6 Tabu Search Analysis **4**

7 Analysis & Observations **5**

8 Results **6**

9 References **7**

1 Problem Statement

We are given the question to find the satisfiability of a Uniform 4-SAT {which is a family of SAT problems distribution obtained from generating 4-CNF formulae randomly.}
So , if all the clauses are true , then we can prove the satisfability of a formula .

2 Approach

The basic approach we used for the A.I assignment is to use the concept of Exploration and apply VND {Variable Neighborhood Descent}, Beam Search and Tabu Search.

State Space

A state space is the set of all configurations that a given problem and its environment could achieve .

We have been given with the fact that there are n bits of strings of 0s and 1s for an n variable state space.
Thus , the total number of state spaces in the state space set is 2^n .

Start & Goal node

The Start node is a randomly generated n bit string which will be a part of the state space that will be given in 'input.txt' file.
The Goal Node is a part of the state space that represents the final/goal state to be reached through our algorithm .
One such example is the following :
For $n = 5$ i.e. 5 variables,
Start Node :
10001
Goal Node :
10101

3 Pseudo Code

The main pseudo code used in our assignment is as follows :

move_gen function

```
def move_gen(state,bits_toggled):
    neighbors = [ ]
    comb = iter.combinations(0 to total no. of states)),bits_toggled)
    for i in comb:
      new= state
      for j in i:
        j= int(j)
        new= new[:j]+str((int(new[j])+1)%2)+new[j+1:]
              neighbors.append(new)
    return neighbors
```

goal_state function

```
def goal_state(formula,inpu):
    global no_clauses
    if(no_of_clauses(formula,inpu)==no_clauses):
       return 1
     return 0
```

4 Heuristic function

We are considering the no. of satisfied clauses as the heuristic for our program. If the heuristic value is n , i.e. no. of clauses, then that node is our goal state.

Pseudo Code:

```
def evaluate_clause(clause,inpu):
    for i in clause:
      if(i>0):
            if(inpu[i-1]=='1'):
                return 1
       else:
            if(inpu[i-1]=='1'):
                return 1
     return 0

def no_of_clauses(all_clauses,inpu):
    no = 0
    for i in clause:
      if(evaluate_clause(i,inpu)):
            no += 1
     return no
```

5 Beam Search Analysis

The time taken to reach the goal state increases exponentially as the input size increases.
Also , as the width of the beam search is increased , we see that the computation time of the beam search to reach the goal state also increases.

6 Tabu Search Analysis

7 Analysis & Observations

Based on observation , we have plotted the following graphs :

From the above graphs , we have analysed that the best first search algorithm always finds the best and optimal solution but at the cost of time consumed is high as BFS has a time complexity of $O(b^d)$.

8 Results

The following conclusions have been made after evaluation of our program :

States Explored : The Hill-Climbing algorithm in general will have lesser no. of states explored than that of BFS . In the worst scenario, the best first search algorithm will explore all the states but Hill-Climbing algorithm will explore uptill $O(bd)$,
where b = the maximum beam width
d = the depth of the search graph from the start state.

Time Complexity : The time complexity for the Best First Search Algorithm is $O(b^d)$ and for Hill-Climbing algorithm it is $O(bd)$.

Optimal Solution : For reaching to the optimal solution , the Best First Search Algorithm is the better algorithm as it explores all the nodes/states but in the case of Hill-Climbing algorithm it doesn't explore all the states and it may get stuck in the local extremum while finding the goal state .

Completeness : The Best First Search Algorithm is Complete as it explores all the states sooner or later. But the Hill Climbing Algorithm is not complete. As it almost always gets stuck in local minima if we do not come up with a really good heuristic.

Conclusion

The Best First Search Algorithm is the more optimal choice as it helps to solve the given problem efficiently .

Although it takes more time than the Hill-Climbing algorithm, it ensures that the best solution has been found to reach the goal state by exploring all the nodes/states in $O(b^d)$, whereas in Hill-Climbing algorithm not all the nodes/states are visited as sometimes the algorithm gets stuck at a local extremum while finding the optimal path to the goal state.

9 References

- http://geeksforgeeks.com
- https://wikipedia.org
- https://stackoverflow.com