Artificial Intelligence Lab
Assignment 3

Group Number: 27

Pavan Kumar V Patil 200030041

Karthik J Ponarkar 200010022

January 2022

Contents

1 Brief description about the domain

1.1 State space

- According to the given question, each state consists of variable number of neighbours which depends upon the number of bits we want to toggle at once.
- Mathematically, the number of neighbours for the current state is equal to the number of ways of selecting *choice* number of variables from the total number of variables.
- State space S is a set which consists of all possible configuration of toggling of the variables which is stored in a object constructed according to the **class** *state*.

1.2 Start node and goal node

- The input SAT formula is fetched from the file named "input.txt".
- The start state is assumed to be the case where all the boolean variables are assigned **True**
- Start state is the initial state where, all algorithms are applied. Goal state is the final state which should be obtained using these algorithms from the start state.
- Goal state is the only state where all the clauses result in **True** output.
- We can also infer that goal state has the highest heuristic value which is mathematically equal the number of the clauses.
- Goal state conditions are used in the goal test function which checks whether the current is a final solution state or not.

2 Pseudocode

2.1 MoveGen for Variable Neigbourhood descent

```
function moveGen(node, choice)
    neighbours = [ ]
    selectedIndices = combination(choice)
    for eachCombintion in selectedIndices do
        child = toggle(node, eachCombination)
        calculateValueForClauses(child)
        child.hValue = heuristic()
        neighbours.append(child)
    end for
    return neighbours
end function
```

2.2 MoveGen for Beam Search and Tabu Search

```
function moveGen(node, choice = 1)
    global closeList
    neighbours = [ ]
    selectedIndices = combination(choice)
    for eachCombintion in selectedIndices do
        child = toggle(node, eachCombination)
        calculateValueForClauses(child)
        child.hValue = heuristic()
        if child ∉ closeList then
            neighbours.append(child)
        end if
    end for
    return neighbours
end function
```

2.3 Goal Test

```
function goalTest(currentState)
    global numberOfClauses
    if currentState.hValue == numberOfClauses then
        return true
    end if
    return false
end function
```

Note:

- **combination**() is a function which returns all the possible number of selection which depends upon parameter choice. Choice represents the number of bits to be toggled at once.
- **toggle**() is a function which alters some of the bits and returns the new altered object

3 Heuristic function considered

- This function assigns a value of +1 to the clauses of the current state which are satisfied i.e, the output is **true** and assigns a value of 0 to the clause of the current state which are not satisfied i.e, the output is **false**
- The sum of assigned values to all the clauses is the heuristic value returned by this function for a state
- Intuitively this function basically returns the number of clauses satisfied for a given state
- Higher the heuristic value returned by this function for a state, more is priority to choose that node during a search algorithm because exploring that node will have more probability of getting a solution for the given SAT
- Using this heuristic function leads to a maximization problem.

4 Beam Search Analysis

- Beam search is a heuristic search algorithm that explores the search graph by expanding the most promising node in a limited set. Beam search is an optimization of Best first search algorithm that reduces its space complexity.
- In beam search, only a predetermined number (beam width) of best heuristic value nodes are kept as candidates. It is thus a greedy search algorithm.
- If beam width b is very large the algorithm basically boils down to a best first search algorithm, which has a worst case of super polynomial space complexity.
- As b increases space complexity also increase due to exploration of more nodes.
- Also, as b increases the probability of getting the best node also increases because the more nodes we explore, more is probability of finding the goal state.

Beam Search for N = 26 & K = 50	
Beam Width	Total Number of States Explored
2	10
3	17
4	22
5	19

5 Tabu Search Analysis

- Tabu search is a meta-heuristic local search method used for mathematical optimization. Local search methods have the tendency to be stuck in local optimum regions. One of such algorithms is hill climbing.
- This algorithm enhances the performance of these techniques by prohibiting already visited best solutions for some tenure. This enables us to overcome local optima and reach our final goal state of global optima.
- We denote the Tabu tenure of a Tabu search by t. As the value of t increases we can explore more number of states which are not visited and can overcome the local optima with more probability.
- It is sane to assume that the value of t should not be greater than the number of variables n.
- We can intuitively say that the optimum value of t is proportional to the the maximum height of the local optima. Let us say that this optimum value is t_{opt}.
- Henceforth, we can conclude that the range for possible values of t is (t_{opt}, n)

Tabu Search for N = 26 & K = 50	
Tenure	Total Number of States Explored
2	9
3	6
4	5
5	19

6 Comparison between VND, Beam search and Tabu Search

Comparison for N = 26 & K = 50	
	Total Number of States Explored
Beam search(B = 2)	10
Tabu search(T = 2)	9
VND	5

7 Result (Program output)

For N = 26 and K = 50

```
=>> 3-CNF SAT is satisfied by Variable Neighbourhood Descent search
=>> Total Number of explored states is: 5
=>> The boolean values of the goal state for which the given SAT problem is satisfied is:
{'a': 'False', 'b': 'False', 'c': 'True', 'd': 'False', 'e': 'False', 'f': 'True', 'g': 'False', 'h': 'False', 'i':
 'False', 'j': 'False', 'k': 'False', 'l': 'False', 'm': 'False', 'n': 'False', 'o': 'False', 'p': 'False', 'q': 'F
alse', 'r': 'False', 's': 'False', 't': 'True', 'u': 'False', 'v': 'True', 'w': 'True', 'x': 'False', 'y': 'False',
 'z': 'False'}
```

```
Enter the Beam Width Value: 2

=>> 3-CNF SAT is satisfied by Beam search
=>> Total Number of explored states is: 10
{'a': 'False', 'b': 'False', 'c': 'False', 'd': 'False', 'e': 'False', 'f': 'False', 'g': 'False', 'h': 'False', 'i
': 'False', 'j': 'True', 'k': 'False', 'l': 'False', 'm': 'False', 'n': 'False', 'o': 'True', 'p': 'True', 'q': 'Tr
ue', 'r': 'False', 's': 'False', 't': 'False', 'u': 'False', 'v': 'False', 'w': 'True', 'x': 'False', 'y': 'False',
 'z': 'False'}
```

```
Enter the Tabu Tenure Value: 2

=>> 3-CNF SAT is satisfied by Tabu search
=>> Total Number of explored states is: 9
{'a': 'False', 'b': 'False', 'c': 'False', 'd': 'False', 'e': 'False', 'f': 'False', 'g': 'False', 'h': 'False', 'i
': 'False', 'j': 'False', 'k': 'False', 'l': 'False', 'm': 'False', 'n': 'False', 'o': 'False', 'p': 'True', 'q': '
True', 'r': 'False', 's': 'False', 't': 'True', 'u': 'True', 'v': 'False', 'w': 'True', 'x': 'True', 'y': 'True', '
z': 'True'}
```