CS312:Artificial Intelligence Laboratory

Lab 3 Report Group 24

Utkarsh Prakash - 180030042 Manjeet Kapil - 180010021

1. Introduction

In this assignment we try to solve the Uniform Random-4-SAT problem using Beam Search, Variable Neighbourhood Descent and Tabu Search. Uniform Random-4-SAT is a family of SAT problems distributions obtained by randomly generating 3-CNF formulae.

2. State Space

Uniform Random-4-SAT is a family of SAT problems distributions obtained by randomly generating 3-CNF formulae in the following way: For an instance with n=4 variables and k=5 clauses, each of the k clauses is constructed from 3 literals which are randomly drawn from the 2n possible literals (the n variables and their negations) such that each possible literal is selected with the same probability of 1/2n. Clauses should not contain multiple copies of the same literal and should not be a tautology. Each choice of n and k thus induces a distribution of Random-4-SAT instances. Uniform Random-4-SAT is the union of these distributions over all n and k. We are going to use n=4 and k=5. The different possible states can be obtained by changing the values of different bits corresponding to the different variables. Hence, each state in the state space can be represented as an array of 4 binary numbers (i.e. 0 and 1). For example, [0, 0, 0, 0] represents that the value of the variables a, b, c and d is False.

3. Start State and Goal State

Start State: Randomly generated array of 4 binary numbers (i.e. 0 and 1). For example, [0, 0, 0, 0] represents that the value of the variables a, b, c and d is False.

Goal State: An array of 4 binary numbers (i.e. 0 and 1) such that all the clauses are satisfied when the values of the variables are plugged in.

For example,

Start State: `[0, 0, 0, 0]`

Clauses:

`~a + d + ~b`

`c + a + ~d`

`~a + c + b`

`a + ~d + ~b`

`d + b + a`

Goal State: `[1, 0, 1, 1]`

NOTE: '+' represents the logical OR operation.

Here as we can see all clauses have been satisfied and the heuristic for goal node is 5. So it is one of the Goal State.

4. Pseudo Code

MoveGen(state)

This function accepts a state and returns all the states that can be reached from the given state in one step i.e. by moving a block.

function MoveGen(state, num_bits):

```
1. neighbors = []
2. for each possible neighbor obtained by changing
   num_bits in state:
3.      neighbor.e = evaluate_node(neighbor,clauses)
4.      neighbors.append(neighbor)
5. return neighbors
```

`evaluate_node()` is a function that calculates the number of clauses satisfied.

GoalTest(state, target_state)

This function checks whether the state is the target(goal) state or not. If it is a target state it returns true otherwise false.

function GoalTest(state, total_numberof_clauses):

```
1. if evaluate_node(state) == total_numberof_clauses
   then
2.      return true
3. return false
```

5. Heuristic Function

The heuristic function or evaluate node function finds the number of clauses satisfied by the given values of variables. For example, if the node is [0, 0, 0, 0] and the clauses are:

```
~a + d + ~b
c + a + ~d
~a + c + b
a + ~d + ~b
d + b + a
```

The number of clauses satisfied for the above value of node is 4.

6. Beam Search Analysis

Clauses :

```
~a + c + ~d
~d + ~c + ~a
~a + ~b + d
~d + ~b + a
~b + ~c + a
```

Beam Length	Goal state	Number of states explored
2	[1, 0, 1, 0]	2
3	[0, 0, 1, 1]	3
4	[0, 0, 1, 1]	20
5	[0, 1, 0, 0]	5

There is no pattern between the number of states explored and the beam length. We can see that for beam length 5 the number of states explored is less than for beam length 4. This can be attributed to the fact that as we increase the beam length more states are considered and hence, chances of finding the solution early increases. However, if the solution is not found early then the number of states explored increases considerably (as in case of beam

length 4) as we expand all the neighbours for all the best neighbours found in the previous iteration. Hence, beam length of 2 is optimal for the above set of clauses.

7. Tabu Search Analysis

Clauses:

```
~a + ~b + ~c

~d + ~b + c

~a + ~b + ~d

b + c + d

~a + b + c
```

Tabu tenure	Goal State	States explored
2	[0, 1, 0, 0]	5
3	[0, 1, 0, 0]	5
4	[0, 1, 0, 0]	4
5	[0, 1, 0, 0]	4

8. Comparison of VND, Beam Search and Tabu Search

We compare the output of VND, Beam Search and Tabu Search by considering different test cases. For Tabu Search we use Tabu tenure = 2 and Beam Search we use beam length = 2.

Clauses:

```
b + ~d + a

~c + d + a

~a + c + ~d

b + ~a + d

c + d + a
```

Beam width : 2

Tabu tenure : 2

Algorithms	Goal State	States Explored
Beam Search	[1, 0, 1, 1]	10
Tabu Search	[1, 1, 0, 0]	11
VND	[1, 1, 0, 0]	8

Clauses:

~c + d + a

~a + ~b + d

b + c + a

~b + d + a

~a + b + c

Beam width : 2

Tabu tenure : 2

Algorithms	Goal State	States Explored
Beam Search	[1, 0, 1, 1]	2
Tabu Search	[1, 0, 1, 0]	11
VND	[1, 0, 1, 0]	8