Report - Assignment 3

Bhavesh Borse(200010005), Eshita Pagare(200010016)

January 10, 2022

1 Introduction

Uniform Random-4-SAT is a family of SAT problems distributions obtained by randomly generating 4-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 4 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 are not accepted for the construction of the problem instance if they contain multiple copies of the same literal or if they are tautological (i.e., they contain a variable and its negation as a literal). 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.

2 Description

2.1 State Space

Every state is defined using class having literal values. Each variable can hold only two values 0 or 1. In case of Tabu Search there are two lists, one for storing state and other for Tabu Tenures.
For Beam search and Variable Neighbourhood Descent,

$$state = [a, b, c, d]$$

For Tabu Search,

$$state = ([a, b, c, d], [x1, x2, x3, x4])$$

2.2 Start Node and Goal Node

For Beam search and Variable Neighbourhood Descent start state is,

$$state = [0, 0, 0, 0]$$

For Tabu Search start state is,

$$state = ([0, 0, 0, 0],\ [0, 0, 0, 0])$$

Goal Node is a list containing boolean values of literals A, B, C, D which give 1 when put in 4-SAT CNF expression.

2.3 MoveGen() and GoalTest() Pseudo codes

```
MoveGen(state)
    intermediateStates ← ()
        #initializing next states to empty states
    for neighbor n of state in order(HeuristicValue) do
    bit_value =∼ (bit_value)
    neighbour ← new.node()
```

```
GoalTest(state)
    if [a, b, c, d] satisfies CNF then
        return True
    return False
        #state is not goal
```

3 Heuristic Function

Following Heuristic Function returns an integer value which is number of clauses satisfied in the formula using given state.

```
Heuristic_Val(expression)
    count ← 0
    for clause in expression do
        if result == 1 then          # clause satisfied
          count ← count + 1
        else continue
    return count
```

4 Beam Search and analysis for different beam lengths

In the following table, no of states explored are compared for all algorithms. The beam length is varied from 1 to 3 for different initial states.

Clauses Initial State	Beam Width	States Explored
[['A', 'C', 'D', 'b'], ['A', 'C', 'b', 'd'], ['A', 'b', 'c', 'd'], ['B', 'D', 'a', 'c'], ['B', 'a', 'c', 'd']] {'a': 1, 'b': 0, 'c': 0, 'd': 0, 'A': 0, 'B': 1, 'C': 1, 'D': 1}	1	5
	2	5
	3	5
[['A', 'B', 'D', 'c'], ['A', 'b', 'c', 'd'], ['B', 'C', 'a', 'd'], ['B', 'a', 'c', 'd'], ['a', 'b', 'c', 'd']] {'a': 0, 'b': 0, 'c': 1, 'd': 1, 'A': 1, 'B': 1, 'C': 0, 'D': 0}	1	1
	2	1
	3	1
[['A', 'C', 'D', 'b'], ['A', 'D', 'b', 'c'], ['B', 'C', 'D', 'a'], ['B', 'D', 'a', 'c'], ['B', 'a', 'c', 'd']] {'a': 0, 'b': 0, 'c': 0, 'd': 0, 'A': 1, 'B': 1, 'C': 1, 'D': 1}	1	1
	2	1
	3	1

5 Tabu search for different values of Tabu tenure

In the following table, no of states explored are compared for all algorithms. The Tabu tenure is varied from 1 to 3 for different initial states.

Clauses Initial State	Tabu Tenure	States Explored
[['A', 'B', 'C', 'D'], ['A', 'B', 'C', 'd'], ['A', 'C', 'D', 'b'], ['B', 'C', 'D', 'a'], ['C', 'a', 'b', 'd']] {'a': 0, 'b': 0, 'c': 0, 'd': 0, 'A': 1, 'B': 1, 'C': 1, 'D': 1}	1	1
	2	1
	3	1
[['A', 'C', 'D', 'b'], ['A', 'D', 'b', 'c'], ['A', 'b', 'c', 'd'], ['B', 'C', 'a', 'd'], ['a', 'b', 'c', 'd']] {'a': 0, 'b': 0, 'c': 0, 'd': 0, 'A': 1, 'B': 1, 'C': 1, 'D': 1}	1	5
	2	5
	3	5
[['A', 'C', 'b', 'd'], ['A', 'D', 'b', 'c'], ['B', 'D', 'a', 'c'], ['D', 'a', 'b', 'c'], ['a', 'b', 'c', 'd']] {'a': 0, 'b': 0, 'c': 0, 'd': 0, 'A': 1, 'B': 1, 'C': 1, 'D': 1}	1	5
	2	5
	3	5

6 Comparison of Variable neighborhood descent, Beam Search, Tabu Search: Nodes explored by each.

The comparison between states explored of all three search algorithms is tabulated below:

Clauses Initial State	Algorithms	States Explored
[['A', 'C', 'b', 'd'], ['B', 'C', 'D', 'a'], ['B', 'C', 'a', 'd'], ['B', 'D', 'a', 'c'], ['a', 'b', 'c', 'd']] {'a': 1, 'b': 1, 'c': 0, 'd': 1, 'A': 0, 'B': 0, 'C': 1, 'D': 0}	Beam	1
	Tabu	1
	VND	1
[['A', 'B', 'C', 'D'], ['A', 'B', 'D', 'c'], ['B', 'C', 'a', 'd'], ['B', 'D', 'a', 'c'], ['a', 'b', 'c', 'd']] {'a': 0, 'b': 0, 'c': 0, 'd': 0, 'A': 1, 'B': 1, 'C': 1, 'D': 1}	Beam	5
	Tabu	5
	VND	5
[['A', 'C', 'b', 'd'], ['B', 'C', 'a', 'd'], ['B', 'D', 'a', 'c'], ['B', 'a', 'c', 'd'], ['C', 'a', 'b', 'd']] {'a': 1, 'b': 0, 'c': 0, 'd': 1, 'A': 0, 'B': 1, 'C': 1, 'D': 0}	Beam	1
	Tabu	1
	VND	1