AI Assignment-3 Report

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

Department of Computer Science, IIT Dharwad

January 11, 2022

Contents

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. Goal Node satisfies all the clauses in the given formula.
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
```

movegen_tabu function

```
def movegen_tabu(state):
    global tenure, formu
    formula = formu
    best_state = ""
    best_heu = -1
    index=-1
    for i in range(len(state)):
        if(tenure[i] == 0):
            if(state[i] == '1'):
                new_state = state[:i] + "0" + state[i+1:]
            else:
                new_state = state[:i] + "1" + state[i+1:]
            h = no_of_clauses(formula,new_state)
            if(h>best_heu):
                index = i
                best_heu = h
                best_state = copy.deepcopy(new_state)
    if(best_state != ""):
        return [best_state,best_heu,index]
    else:
        return 0
```

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.

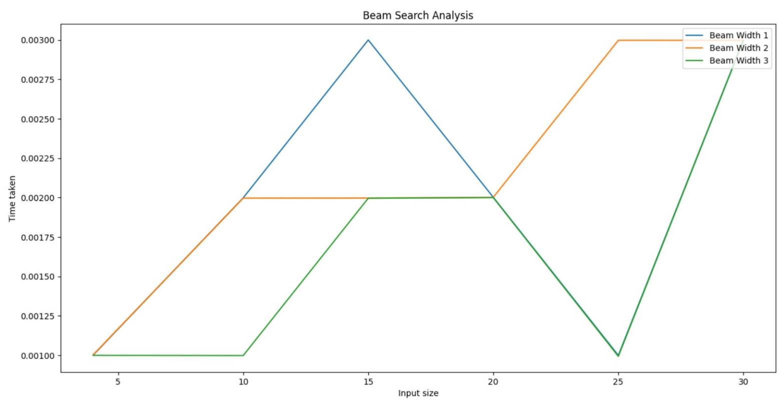
Pseudo Code:

```
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 as the input size increases. As the width of the beam search is increased, the probability of getting the best node also increases because the more the nodes get explored, more is the chance of finding the goal state.
Also , as the width of the beam search is increased , we see that the computation time and space complexity of the beam search to reach the goal state also increases.

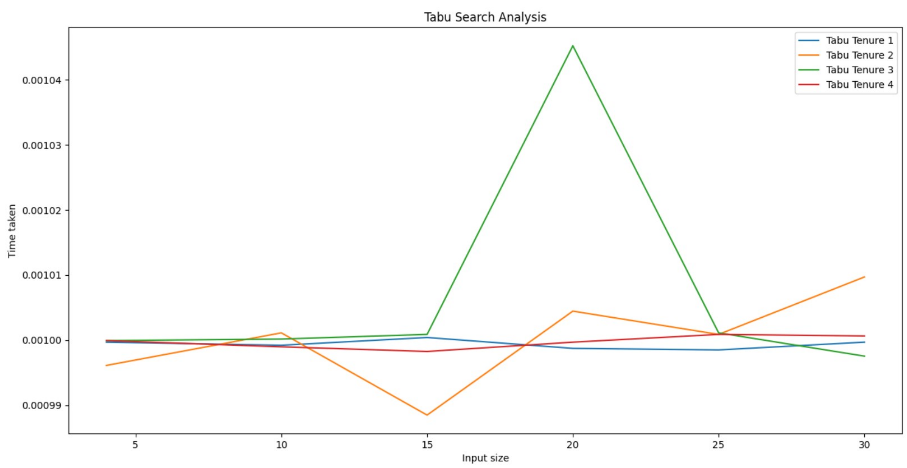


6 Tabu Search Analysis

We have compiled the following results for Tabu Search with different values of Tabu Tenure .

Tabu Tenure	No. of Variables	No. of Clause	Goal	States explored
1	4	5	Reached	3
2	4	5	Reached	6
3	4	5	Reached	6
4	4	5	Reached	6
1	10	50	Not Reached	8
2	10	50	Not Reached	18
3	10	50	Not Reached	22
4	10	50	Not Reached	63
1	20	75	Not Reached	13
2	20	75	Not Reached	24
3	20	75	Reached	34
4	20	75	Reached	57

Here is a graph for different Tabu Tenures with number of clauses = 10.



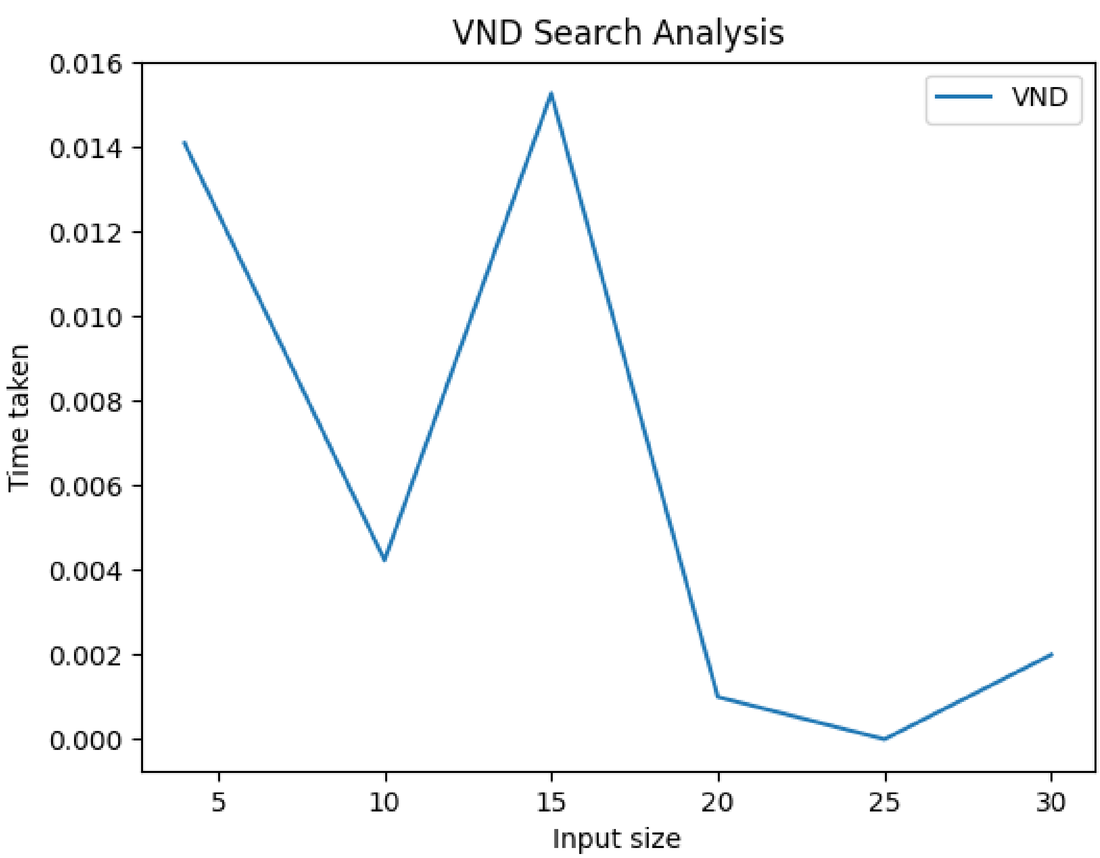
7 Analysis & Observations

Based on observation , we have compiled the following data pertaining to VND, Beam Search and Tabu Search :

No. of variables	Clause	VND	Beam Width 1	Beam Width 2	Beam Width 3	Beam Width 4	Tabu Tenure 1	Tabu Tenure 2	Tabu Tenure 3	Tabu Tenure 4
4	5	2	4	7	11	16	3	6	6	6
6	10	2	4	3	4	5	5	5	5	5
8	25	4	8	15	25	38	5	5	5	5
10	50	17	21	25	29	34	8	18	22	63
15	150	26	31	42	58	79	23	29	80	102
20	75	15	22	35	54	79	13	24	34	57
25	100	20	25	34	47	64	8	11	15	31

In the above table , the green cells represents success of finding the goal state and the red cells represents the failure of reaching to the goal state.

Here is the graphical analysis for VND.



8 Results

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

VND {Variable Neighborhood Descent} : If there exist a solution to reach the goal state then this algorithm is considerably fast but , if there does not exist a solution then, it will be very slow.

Beam Search Algorithm : As the beam width of Beam Search Algorithm increases , the time taken also increases but the possibility to reach the goal state will also increase.

Tabu Search Algorithm : If the tabu tenure is very low , then the algorithm will roam around the local optima but if the tabu tenure very high then the algorithm will explore very few nodes but for a well chosen tenure , we will reach the goal state in a very short amount of time.
We can conclude that the optimal value of the Tabu Tenure should be around : No. of literals/5 { if $n \geq 10$, otherwise 2}

9 Conclusion

All the above described algorithms help to overcome the problem of escaping local minima and they work reasonably well when the hyper-parameters like Tabu Tenure and Beam Width are chosen wisely.

10 References

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