AI - LAB
Assignment-3
Report

Arvind Kumar M - 200020008

Tarun Saini - 200010051

January 11, 2022

Contents

1 Introduction

In this task,we are finding satisfiability of uniform random 4 - SAT
Uniform Random-4-SAT is a family of SAT problems distributions obtained by randomly generating 4-CNF formulae.
So a formula is satisfiable if all the clauses are true.

We are finding that using Variable neighbourhood descent, Beam Search and Tabu Search.

2 State space

In the problem for n variables state space is a n bit string of '0's and '1's.
So the total possible state spaces are 2^n.

We are generating 4-SAT using random choice and storing it in a file.

3 Start state and Goal state

Start state is a randomly generated n bit string representing a
solution candidate of the problem

An example of Start State: is as below:
start = "01001" for 5 variables

Goal state is n bit string for which all clauses gives true.

Correspondingly, the Goal State: is as below:
goal = "11101" for 5 variables
(heuristic(goal) is **k**)

4 Pseudo code for Movegen()

```
def movegen(state, toggle):
    list = [ ]
    for i combinations of (n,toggle):
        if(state[i]=='1'):
            newstate[i] = '0'
        else:
            newstate[i] = '1'
        if (newstate not in CLOSED):
            list.append(newstate)
    return list
```

5 Pseudo Code for Goaltest()

```
def goaltest(current state):
    if(heuristic(state) == k):
            return True
    return False
```

6 Heuristic Function

We are taking number of satisfied clauses as the heuristic.

If the heuristic is k(number of clauses) then it is a goal state.

def heuristic1(current state):

h_val = 0

h = numpy.zeros(n)

for each element in current state:

if(elt == '1' and clause has elt and h[index] == 0):

h_val += 1

h[index] = 1

elif(elt == '0' and clause has ! elt and h[index] == 0):

h_val += 1

h[index] = 1

return **h_val**

7 Variable Neighborhood Descent

In this, we make the neighborhood denser by increasing the number of bits to be toggled if it get stuck in a local maxima.

Pseudo code

def vdn (state, toggle)

while(toggle $\leq n$) :

$if(goaltest == True)$:
return

$movegen(toggle)$

$if(heuristic < previous)$:

$toggle+ = 1$

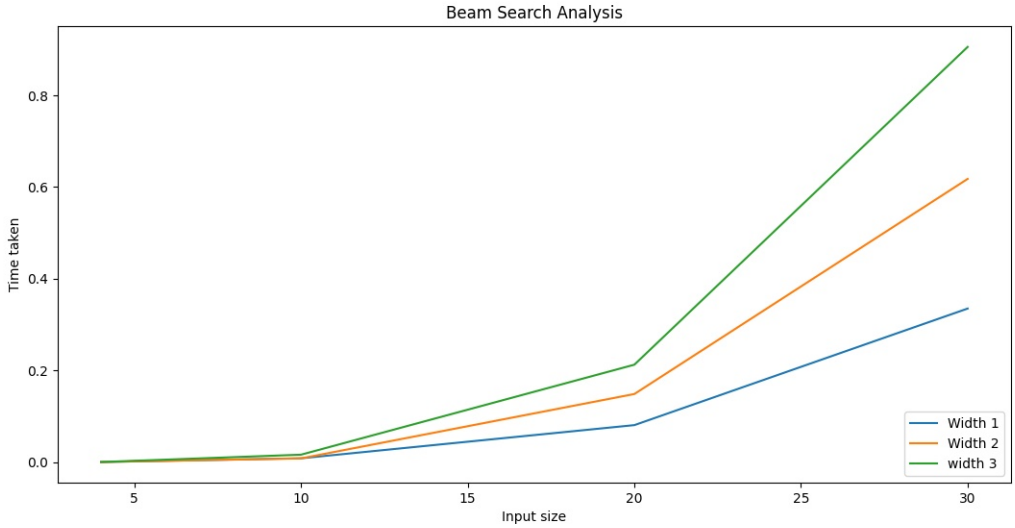
$return \quad state$

8 Beam Search Analysis

Beam search is taking high time for higher width and exploring more states compared to smaller width.

Beam width	No of vars	No of clause	Goal	States explored
1	4	5	yes	1
2	4	5	yes	1
3	4	5	yes	1
1	10	50	no	2
2	10	50	no	4
3	10	50	no	6
1	20	100	no	4
2	20	100	no	8
3	20	100	no	12

The above table shows that beam search is not optimal.



The above graph shows time taken vs beam search

9 Tabu search Analysis

Tabu search explores less state as we increase the Tabu tenure.

But optimality is not guaranteed.

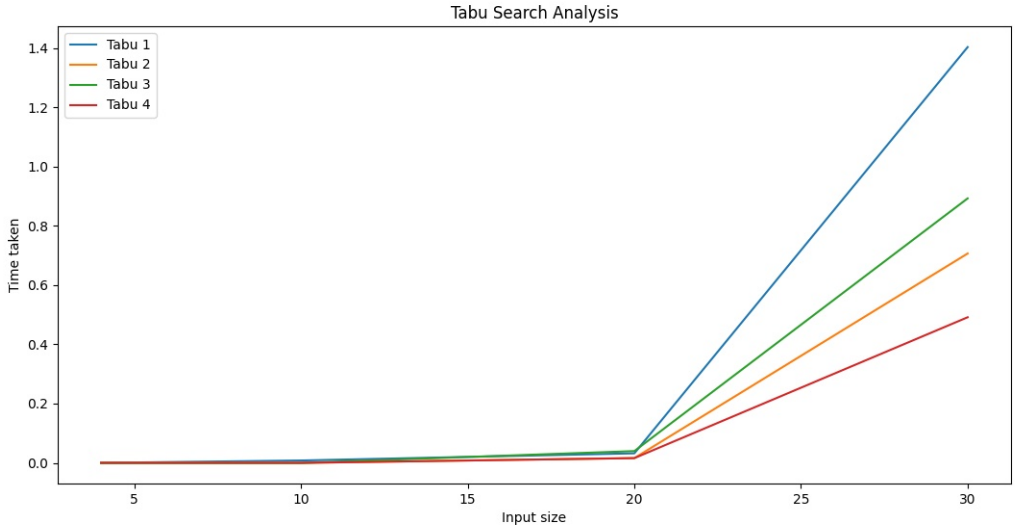
Tabu tenure 1 is reaching goal state many times but with more time.

On the other hand larger Tabu tenure runs in less time but reaching local maxima many times.

Tabu Tenure	No of vars	No of clause	Goal	States explored
1	4	5	yes	4
2	4	5	yes	2
3	4	5	yes	2
4	4	5	yes	2
1	10	50	yes	8
2	10	50	yes	7
3	10	50	yes	3
4	10	50	no	5
1	20	100	yes	33
2	20	100	no	10
3	20	100	yes	15
4	20	100	no	12

Below graph shows time vs input size analysis for Tabu search.

Time decreases as the tenure increases.



10 Optimal values

Beam Search - beam width

- Beam width of 2 and 3 are exploring more states. Hence there is more probability of reaching the goal.
- But it is also not reaching goal state sometimes.
- Beam width 3 is taking long time for larger inputs.
- So beam width **2** is optimal

Tabu Search - tabu tenure

- Tabu tenure of 1 is exploring more states comparitively and taking more time but more probability of reaching goal.
- Tabu tenure of 2 and 3 are exploring less states and sometimes end up in local maxima.
- So Tabu tenure **1** and **2** are optimal.

11 Comparison

VND vs Beam vs Tabu
Comparison w.r.t. number of states explored.

No of vars	Clause	VDN	Beam 1	Beam 2	Beam 3	Tabu 1	Tabu 2	Tabu 3	Tabu 4
4	5	1	1	1	1	2	1	1	1
6	10	2	2	2	3	7	4	3	3
8	25	3	2	3	4	8	3	5	4
10	50	5	2	6	9	8	7	3	5
15	75	8	4	8	12	25	10	21	33
20	100	9	2	4	6	33	10	15	12
25	150	13	5	10	15	76	46	14	22

Green ———Goal state
Red———Local maxima

The above table shows that Variable neigbourhood descent and Tabu search are more optimal than beam search.