AI Lab 3

Heuristic Search Algorithms

Group 11
180010014 | Amogh Dasture
180010024 | Paritosh Gavali

Domain Description

Uniform Random-4-SAT is a family of SAT problems distributions obtained by randomly generating 3-CNF formulae. 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.

The **state space** is the all possible permutations of truth values of all literals. **Start node** is selected by assigning random truth values to all the literals. **Goal node** is the one where the given formula of clauses are satisfied.

```
# for generating a start node
def generateRandomState(n):
    state=[]
    for i in range(n):
        state.append(random.randint(0,1))
    return state
```

Move Generation

For a given state, we can have the next generation based on the number of literals whose values we toggle. For both Beam and Tabu search we find the neighbours by toggling a single literal at a time. For Variable neighbourhood, we keep on increasing the value ‘k’ if we are stuck in a local maximum, where ‘k’ is the number of literals toggled to generate the new neighbours (going towards denser neighbourhood).

Goal Test : The goal is found if all the clauses are satisfied by the current state.

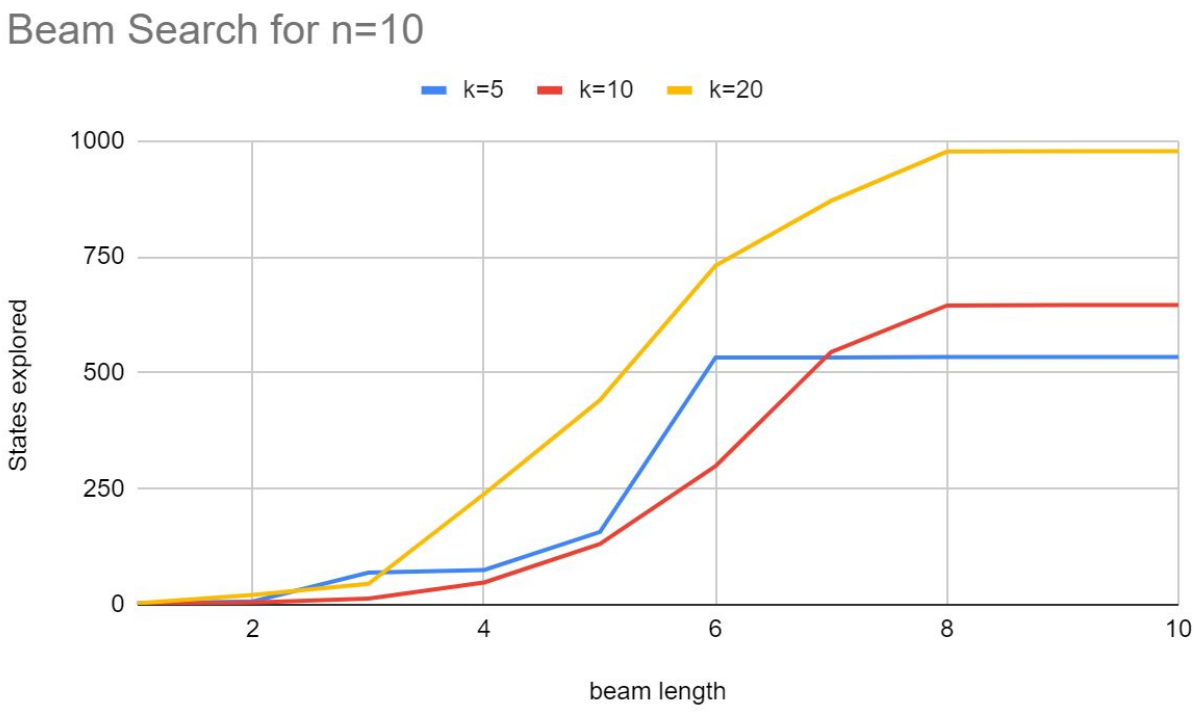
Heuristic Function

Our heuristic function is very simple and effective. We consider the total number of clauses satisfied as the heuristic value of the current state.

```
def heuristic(formula,state):
    val=0
    for clause in formula:
      Check if the state is satisfies the clause
      if yes then increment the values of val
    return val
```

Beam Search analysis for different beam lengths

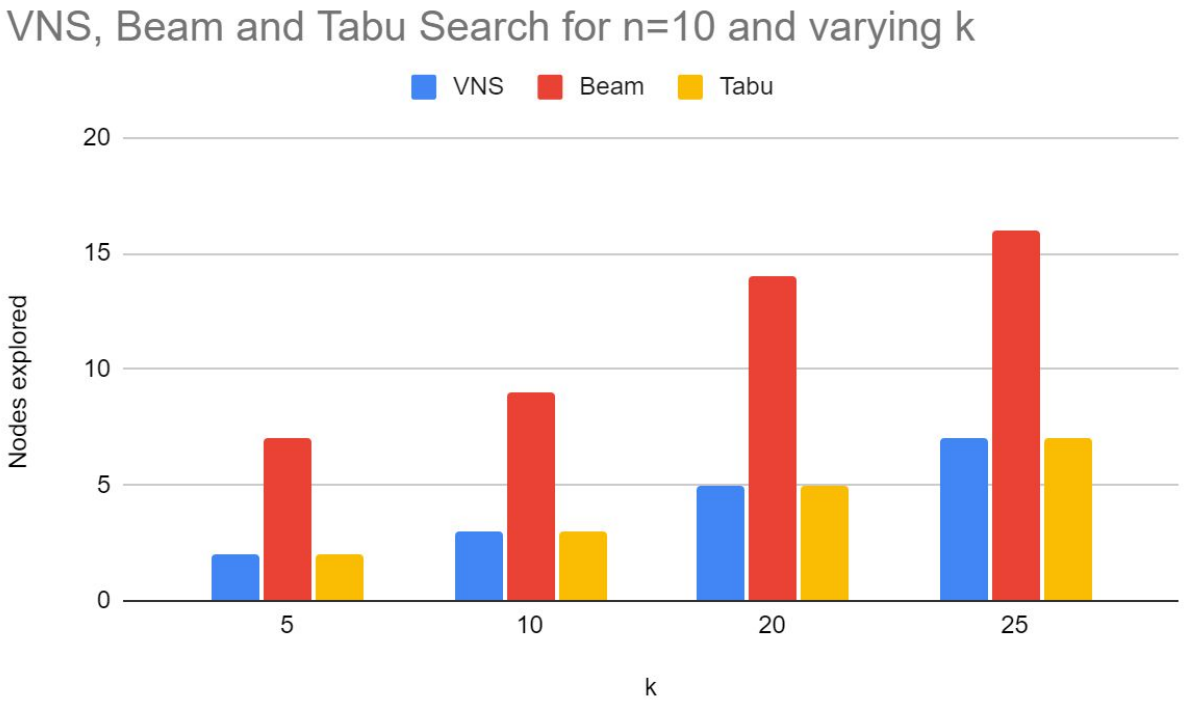
We ran 3 tests for different values of k and we found that as beam length increases the number of states explored also increases. As beam length increases we move from exploitative to explorative search, thus the number of states explored also increases.



Tabu Search for different values of tabu tenure

We found that varying tabu tenure doesn't vary the number of states explored much. This is due to not having many local minima. There are very few cases where the number of states explored have changed (<10% of the test we ran).

Comparison of all three algorithms



As we can infer from the data that VNS and Tabu explore less number of states as compared to Beam search. As k increases for the same value of n the number of states explored by each algorithm (as the solution space reduces as the number of clauses increases).