AI Lab 3 Report

Om Patil (200010036)
Hrishikesh Pable (200010037)

January 2022

Contents

1 Implementation Overview

1.1 State Space

Each state in the state space is a different truth assignment of the n input variables. In our implementation this is stored as an array of Boolean where the n^{th} element corresponds to the n^{th} variable in the list of variables.

1.2 Start State

Start State is the state from which we begin our search for the goal. In our implementation, the start state is taken as the state where all the variables are assigned the Boolean value False.

1.3 Goal State

The goal state is a unknown state for which all the given clauses are satisfied. The Hill Climb variations try to reach the Goal State starting from the Start State with the help of the heuristic.

1.4 GoalTest()

The GoalTest() function tries to check whether the current state (i.e. the current truth assignment of the variables) satisfies the entire boolean expression or not. If yes, then it declares that the goal state is found. If no, then the search is continued.

```
function GoalTest($State, Clauses$)
    $satisfied \leftarrow True$
    for Clause in Clauses do
        if not $isSatisfied(Clause, State)$ then
            $satisfied \leftarrow False$
        end if
    end for
    return $satisfied$
end function
```

1.5 MoveGen

This function returns the neighbours of any state passed to it as an argument. Since each of the 3 search methodologies has a different way of finding/handling the neighbours of the current state, the $MoveGen()$ function is unique to each them, corresponding $MoveGen()$ functions have been explained in the Exploration Methodologies section.

2 Heuristic

The heuristic used in our implementation is the number of clauses satisfied.

```
function HEURISTIC($State, Clauses$)
    $heuristic \leftarrow 0$
    for Clause in Clauses do
        if $isSatisfied(Clause, State)$ then
            $heuristic++$
        end if
    end for
    return $heuristic$
end function
```

3 Exploration Methods

3.1 Variable Neighbourhood Descent (VND)

This method is a variation of Hill Climbing, in which, if we are stuck in a state, due to all neighbours being explored (i.e a dead-end), we increase the density of the $MoveGen()$ Function, thereby allowing further exploration towards the goal state. In our case, the density of the $MoveGen()$ function being increased results in increasing the number of bits being toggled. (Since more combinations can be achieved by toggling more number of bits)

Following is the pseudocode for $MoveGen()$ Function of the Variable Neighbourhood Descent:

```
function $MoveGen(State, Density)$
    $newStates \leftarrow []$
    for $Density$ number of $indexs$ in $State$ do
        $newState \leftarrow State$
        $ToggleBits(newState, indexs)$
        $newStates.append(newState)$
    end for
    return $newStates$
end function
```

3.2 Beam Search

In Beam Search a beam of a given number (beam width) of states is maintained. After exploring all the states in the beam, the top (according to heuristic) beam width number of newly found states form the new beam which is further explored in a similar manner. The pseudocode of the corresponding $MoveGen()$ function is given below.

```
function $MoveGen(State)$
```

```
    $newStates \leftarrow []$
    for $index$ in $State$ do
        $newState \leftarrow State$
        $ToggleBit(newState, index)$
        $newStates.append(newState)$
    end for
    return $newStates$
end function
```

3.3 Tabu Search

In this method, the explored list is not maintained. This can result in oscillation around the peak/valley, when we reach a local extremum while exploration. So, to avoid that, a Tabu Tenure is set whenever a bit is toggled to generate the neighbors of a state. While the Tabu Tenure for a bit is non zero, we cannot toggle that bit again to generate neighbours. This restricts us from generating the already explored state in the next few moves. The pre-existing Tabu tenures are decremented by 1 at each move, which allow the bit to be re-toggled post a fixed number of generations. $MoveGen()$ for Tabu Tenure is as follows:

```
function $MoveGen(State)$
    $newStates \leftarrow []$
    for $index$ in $State$ do
        if not $State[index].tenure$ then
            $newState \leftarrow State$
            $ToggleBit(newState, index)$
            $SetTenure(newState, index)$
            $newStates.append(newState)$
        end if
    end for
    return $newStates$
end function
```

4 Analysis and Comparisons

4.1 Effect of Beam Width

For a input of n=4 and k=5 a beam width of 2 is seen to be ideal. This is illustrated by the below examples.

4.1.1 Example 1

$(x1 + x3 + x0)$
$(x0 + \neg x2 + x3)$
$(x2 + x3 + \neg x0)$

$(\neg x0 + \neg x3 + \neg x2)$
$(x0 + \neg x3 + \neg x2)$

Beam Width	Result	Solution	Explored States
1	No goal state found!	NA	NA
2	Goal state found!	[False, True, False, False]	4
3	Goal state found!	[False, True, False, False]	4
4	Goal state found!	[False, True, False, False]	4

4.1.2 Example 2

$(x1 + x3 + \neg x2)$
$(x2 + \neg x0 + x3)$
$(x1 + x0 + x2)$
$(\neg x3 + \neg x2 + \neg x0)$
$(x3 + x0 + x2)$

Beam Width	Result	Solution	Explored States
1	No goal state found!	NA	NA
2	Goal state found!	[True, False, False, False]	2
3	Goal state found!	[True, False, False, False]	2
4	Goal state found!	[True, False, False, False]	2

4.2 Effect of Tabu Tenure

The selection of an optimal Tenure is essential for finding the solution. A smaller Tenure would lead us to loop back to the same state multiple times i.e. it would get stuck in an infinite loop. A larger tenure would make it difficult to find the Goal State (Because, larger the tenure, more neighbouring states will be locked for the present state. This would make it highly difficult for us to find the goal state. (Because there are more chances of us passing very nearby the goal state, but locking it due to higher Tenure).

For the case of n=4 k=5 it seems a reasonable value for Tabu Tenure is 3. This can be observed in Example 2.

4.2.1 Example 1

$(\neg x0 + \neg x2 + x3)$
$(\neg x2 + x5 + \neg x0)$
$(\neg x1 + x4 + \neg x6)$
$(\neg x5 + \neg x4 + \neg x3)$
$(x6 + \neg x0 + x4)$
$(x3 + \neg x6 + x0)$
$(\neg x5 + \neg x4 + \neg x6)$
$(x2 + x6 + x1)$

$(x2 + \neg x3 + \neg x4)$
$(x4 + \neg x2 + \neg x5)$
$(x2 + x4 + \neg x1)$
$(x6 + \neg x4 + \neg x3)$
$(x4 + \neg x5 + x2)$
$(\neg x5 + \neg x6 + \neg x0)$

Tabu Tenure	Result	Solution	Explored States
1	No goal state found!	NA	NA
2	No goal state found!	NA	NA
3	Goal state found!	[False, True, True, False, False, False, False]	5
4	Goal state found!	[False, True, True, True, False, False, False]	6
5	No goal state found!	NA	NA
6	No goal state found!	NA	NA

Table 1: Comparison of Tabu Tenures for n=7, k=14

4.2.2 Example 2

$(x2 + x3 + \neg x0)$
$(\neg x2 + x3 + x0)$
$(x3 + \neg x0 + \neg x1)$
$(\neg x1 + \neg x0 + x2)$
$(x0 + x3 + x2)$

Tabu Tenure	Result	Solution	Explored States
1	No goal state found!	NA	NA
2	No goal state found!	NA	NA
3	Goal state found!	[True, True, True, True]	5

Table 2: Comparison of Tabu Tenures for n=4, k=5

4.3 VND vs. Beam Search vs. Tabu Search

As per our observation the 3 methods are ordered as follows with respect to time complexity and completeness:

VNF >Tabu >Beam

This can be observed in the below examples as well.

4.3.1 Example 1

$(x3 + x2 + \neg x1)$
$(x1 + x3 + \neg x0)$

$(x2 + x3 + \neg x0)$
$(\neg x2 + x0 + x1)$
$(x1 + x2 + x3)$

Method	Result	Solution	Explored States
Beam ($\beta = 2$)	Goal state found!	[True, True, True, False]	4
VNF	Goal state found!	[False, False, False, True]	2
Tabu (Tenure=2)	Goal state found!	[True, True, True, False]	4

4.3.2 Example 2

$(\neg x0 + \neg x2 + x3)$
$(\neg x2 + x5 + \neg x0)$
$(\neg x1 + x4 + \neg x6)$
$(\neg x5 + \neg x4 + \neg x3)$
$(x6 + \neg x0 + x4)$
$(x3 + \neg x6 + x0)$
$(\neg x5 + \neg x4 + \neg x6)$
$(x2 + x6 + x1)$
$(x2 + \neg x3 + \neg x4)$
$(x4 + \neg x2 + \neg x5)$
$(x2 + x4 + \neg x1)$
$(x6 + \neg x4 + \neg x3)$
$(x4 + \neg x5 + x2)$
$(\neg x5 + \neg x6 + \neg x0)$

Method	Result	Solution	Explored States
Beam ($\beta = 6$)	Goal state found!	[False, True, True, False, False, False, False]	6
VNF	Goal state found!	[False, False, True, False, False, False, False]	2
Tabu (Tenure=3)	Goal state found!	[False, True, True, False, False, False, False]	4