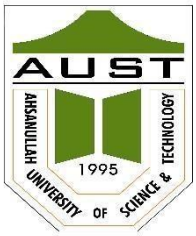
AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

DHAKA-1208, BANGLADESH.



Department of Computer Science and Engineering

Spring 2019

Program: Bachelor of Science in Computer Science and Engineering

Course No: CSE 4108

Course Title: Artificial Intelligence Lab

Term Project: 02

Date of Submission: 30/07/2019

Submitted to

Dr. S.M. Abdullah Al-Mamun

Professor, Department of CSE, AUST.

Md. Siam Ansary

Adjunct Faculty, Department of CSE, AUST.

Submitted By

Name : Robiul Hasan Nowshad

Student ID : 16.01.04.061

Lab Group : B1

Question: Implementation Stochastic Hill-climbing algorithm in Prolog or Python.

H**ill climbing:** Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbour has a higher value.

Algorithm for Hill Climbing:

Step 1: Evaluate the initial state, if it is goal state then return success and Stop.

Step 2: Loop Until a solution is found or there is no new operator left to apply.

Step 3: Select and apply an operator to the current state.

Step 4: Check new state:

If it is goal state, then return success and quit.

Else if it is better than the current state then assign new state as a current state.

Else if not better than the current state, then return to step2.

Step 5: Exit.

Types of Hill Climbing Algorithm:

- Simple hill Climbing
- Steepest-Ascent hill-climbing
- Stochastic hill Climbing

Here, we implemented the Stochastic hill Climbing Algorithm

Stochastic hill climbing does not examine for all its neighbour before moving. Rather, this search algorithm selects one neighbour node at random and decides whether to choose it as a current state or examine another state.

Solution:

List_app.pl

```
:-module(list_apps,
[soe/2, nthel/3, rplc_nthel/4, del_el/3, del_1st_n_el/3,
del_last_n_el/3, write_list/1]).


% A procedure to find the sum of the elements
soe([],0):-!.
soe([H|T],N):- soe(T,N1), N is N1+H.

% A procedure to find the nth element
nthel(1,[H|_],H):-!.
nthel(N,[_|T],El):- N1 is N-1, nthel(N1,T,El).

% A procedure to replace the nth element
rplc_nthel(1,X,[_|T],[X|T]):-!.
rplc_nthel(N,X,[H|T],L1):- N1 is N-1, rplc_nthel(N1,X,T,L2), L1=[H|L2].

% Procedures to delete list elements
del_el(X,[X|T],T):-!.
del_el(X,[H|T],L1):-del_el(X,T,L2),L1=[H|L2].

del_1st_n_el(1,[_|T],T):-!.
del_1st_n_el(N,[_|T],L):-N1 is N-1,del_1st_n_el(N1,T,L).

del_last_n_el(N,L1,L2):- reverse(L1,Lx),del_1st_n_el(N,Lx,Ly), reverse(Ly,L2).

% Procedure to display elements of a list
write_list([]):-!.
write_list([H|T]):- write(H), write_list(T).
```

eval_state.pl

```
:-module(eval_state, [do_eval/2, eval/2,
getdigits/9  /*, hl/2, di_up/2, di_dn/2,incr_hval/0,
chk_incr/3, do_incr/2, chkup_incr/4, doup_incr/3,
chkdn_incr/4, dodn_incr/3*/  ]).
:-use_module(list_apps).
:-dynamic(hval/1).


/* Evaluates a 8-queens' state, S given as an 8-digit number  */
do_eval(S,V):-getdigits(S,D1,D2,D3,D4,D5,D6,D7,D8),
L=[D1,D2,D3,D4,D5,D6,D7,D8],eval(L,V).

eval(L,V):- assert(hval(0)),
hl(1,L), di_up(1,L),di_dn(1,L),    hval(V1),V is 28-V1,
retract(hval(_)).


getdigits(S,D1,D2,D3,D4,D5,D6,D7,D8):-
    D1 is S div   10000000,
    R1 is S mod   10000000,
    D2 is R1 div  1000000,
    R2 is R1 mod  1000000,
    D3 is R2 div  100000,
    R3 is R2 mod  100000,
    D4 is R3 div  10000,
```

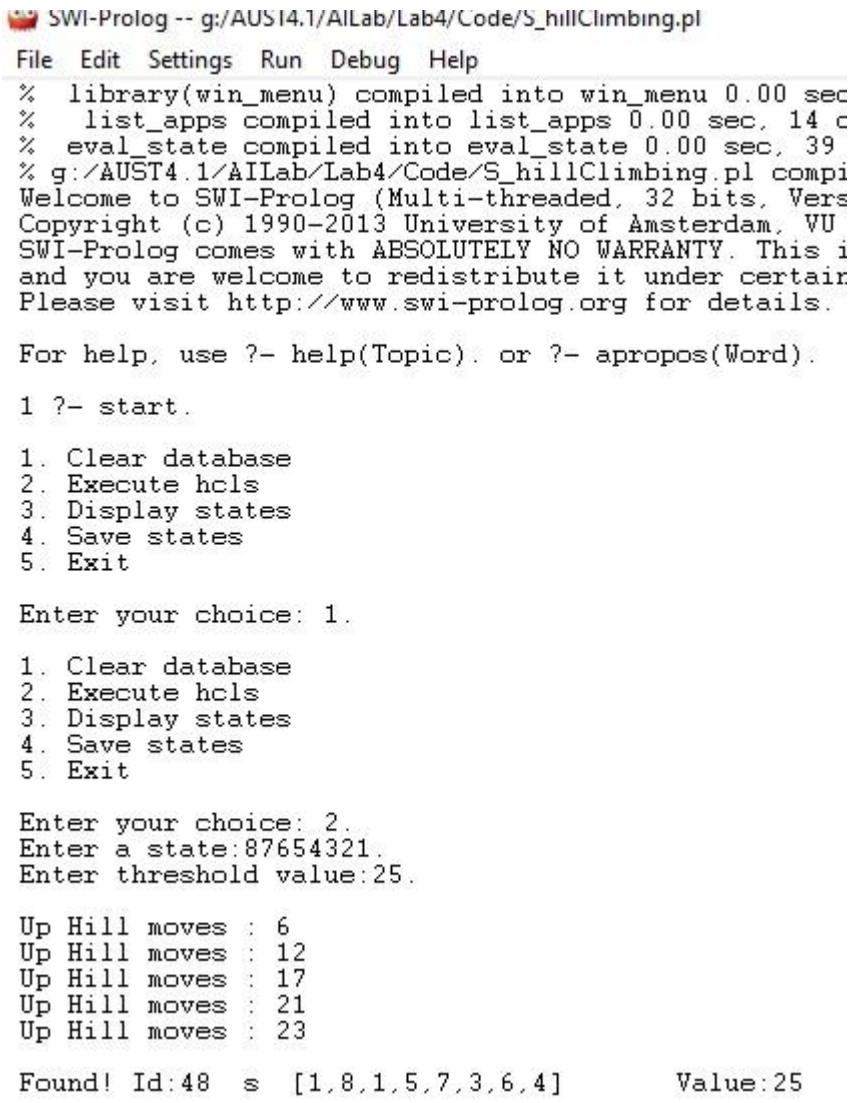
```
    R4 is R3 mod  10000,
    D5 is R4 div  1000,
    R5 is R4 mod  1000,
    D6 is R5 div  100,
    R6 is R5 mod  100,
    D7 is R6 div  10,
    D8 is R6 mod  10.

hl(8,_):-!.
hl(I,L):- nthel(I,L,X), chk_incr(I,L,X), I1 is I+1, hl(I1,L).

chk_incr(8,_,_):-!.
chk_incr(I,L,X):- I1 is I+1, nthel(I1,L,Y), do_incr(X,Y),
    chk_incr(I1,L,X).

do_incr(X,Y):- X=Y, incr_hval.
do_incr(_,_).

incr_hval:-hval(V), V1 is V+1, retract(hval(_)), assert(hval(V1)).


di_up(8,_):-!.
di_up(I,L):- nthel(I,L,X), chkup_incr(I,L,X,0), I1 is I+1, di_up(I1,L).

chkup_incr(8,_,_,_):-!.
chkup_incr(I,L,X,K):- I1 is I+1, nthel(I1,L,Y), K1 is K+1,
    doup_incr(X,Y,K1), chkup_incr(I1,L,X,K1).

doup_incr(X,Y,K1):- X1 is X+K1, Y=X1, incr_hval.
doup_incr(_,_,_).


di_dn(8,_):-!.
di_dn(I,L):- nthel(I,L,X), chkdn_incr(I,L,X,0), I1 is I+1, di_dn(I1,L).

chkdn_incr(8,_,_,_):-!.
chkdn_incr(I,L,X,K):- I1 is I+1, nthel(I1,L,Y), K1 is K+1, dodn_incr(X,Y,K1),
    chkdn_incr(I1,L,X,K1).

dodn_incr(X,Y,K1):- X1 is X-K1, Y=X1, incr_hval.
dodn_incr(_,_,_).
```

Stochastic_hill_climbing:

```
:-use_module(eval_state).
:-use_module(list_apps).
:-dynamic(state/4). /* id,type,state,h_value*/
:-dynamic(uphill_mv/3). /* id,state,h_value(store up hillmoves)*/
:-dynamic(up_id/1).
:-dynamic(id/1).
:-dynamic(max_val/1).
:-dynamic(threshold/1).
:-dynamic(restrt_cntr/1).
:-dynamic(list_st/1).

/* Organizing a Menu */
start:- repeat,
    write('\n1. Clear database'),
    write('\n2. Execute hcls'),
```

```
    write('\n3. Display states'),
    write('\n4. Save states'),
    write('\n5. Exit'),
    write('\n\nEnter your choice: '),
    read(N), N >0, N < 6,
    do(N), N=5,!.

do(1):- retractall(state(_,_,_,_)),retractall(id(_)), retractall(max_val(_)),
    retractall(threshold(_)), retractall(restrt_cntr(_)).
do(2):- go_hcs.
do(3):- listing(state).
do(4):- write('Enter a new file name:'), read(Flnm),
    tell(Flnm),listing(state),told.
do(5):- abort.

/* Beginning of search */
go_hcs:- write('Enter a state:'), read(S),
     write('Enter threshold value:'), read(V),
     assert(threshold(V)),assert(restrt_cntr(0)),
     getdigits(S,D1,D2,D3,D4,D5,D6,D7,D8),
     L=[D1,D2,D3,D4,D5,D6,D7,D8],
     gnrt_sucsr(L).

/* Generating the successors of a 8-queens' state given as a list */
gnrt_sucsr(L):- assert(id(1)), assert(state(1,'c',L,50)),
    assert(up_id(1)),incr_id, mk_new(1,L), retract(id(_)), evaluate.

incr_id:-id(V), V1 is V+1, retract(id(_)), assert(id(V1)).
incr_up_id:-up_id(V), V1 is V+1, retract(up_id(_)), assert(up_id(V1)).

mk_new(9,_):-!.
mk_new(N,L):- nthel(N,L,X), del_el(X,[1,2,3,4,5,6,7,8],L1),
    cng_mk(N,L,L1), N1 is N+1, mk_new(N1,L).

cng_mk(_,_,[]):-!.
cng_mk(N,L,L1):- L1=[H|T], rplc_nthel(N,H,L,L2),id(Id),
    assert(state(Id,'s',L2,50)), incr_id, cng_mk(N,L,T).

/* Evaluating the states */
evaluate:- eval_all, checkall.

eval_all:- state(I,T,L,_), eval(L,V),retract(state(I,_,_,_)),
    assert(state(I,T,L,V)), fail.
eval_all:-!.

/* Determining and displaying the best state */
checkall:- state(_,'c',_,V1), threshold(V2), V1 >= V2, I is 1, dsply(I),!.
checkall:- best(I1,V1), threshold(V2), V1 >= V2, I is I1, dsply(I),!.
checkall:- state(_,'c',_,V1), best(I,V2) ,V2>V1,state(I,_,L,_),retractall(state(_,_
    ,_,_)),
    write_list(['\nUp Hill moves : ',V2]),
    retractall(uphill_mv(_,_,_)),up_id(Z),assert(uphill_mv(Z,L,V2)),incr_up_id,
    gnrt_sucsr(L),!.
checkall:- rndm_upHill,!.

best(I,Max):- state(_,'s',_,Val), assert(max_val(Val)),
    updt_max, max_val(Max), state(I,_,_,Max), retract(max_val(_)),!.

updt_max:- state(_,_,_,V2),  max_val(V1), V2>V1,
    retract(max_val(_)), assert(max_val(V2)), fail.
updt_max:-!.

dsply(I):-state(I,T,L,V),
    write_list(['\n\nFound! Id:',I,'  ',T,'  ', L,'  ','Value:',V,'\n']),!.

rndm_upHill:-up_id(V), X is random(V)+1,uphill_mv(X,L,_),
    write('\nSelecting up hill moves.'),gnrt_sucsr(L).
```

Output:


SWI-Prolog -- g:/AUST4.1/AILab/Lab4/Code/S_hillClimbing.pl
File Edit Settings Run Debug Help
% library(win_menu) compiled into win_menu 0.00 sec
% list_apps compiled into list_apps 0.00 sec, 14 c
% eval_state compiled into eval_state 0.00 sec, 39
% g:/AUST4.1/AILab/Lab4/Code/S_hillClimbing.pl compi
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Vers
Copyright (c) 1990-2013 University of Amsterdam, VU
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This i
and you are welcome to redistribute it under certain
Please visit http://www.swi-prolog.org for details.
For help, use ?- help(Topic). or ?- apropos(Word).
1 ?- start.
1. Clear database
2. Execute hcls
3. Display states
4. Save states
5. Exit
Enter your choice: 1.
1. Clear database
2. Execute hcls
3. Display states
4. Save states
5. Exit
Enter your choice: 2.
Enter a state:87654321.
Enter threshold value:25.
Up Hill moves : 6
Up Hill moves : 12
Up Hill moves : 17
Up Hill moves : 21
Up Hill moves : 23
Found! Id:48 s [1,8,1,5,7,3,6,4] Value:25

```
SWI-Prolog -- g:/AUST4.1/AILab/Lab4/Code/S_hillClimbing.pl
File  Edit  Settings  Run  Debug  Help

1. Clear database
2. Execute hcls
3. Display states
4. Save states
5. Exit

Enter your choice: 3.
:- dynamic state/4.

state(1, c, [1, 8, 1, 5, 7, 3, 2, 4], 23).
state(2, s, [2, 8, 1, 5, 7, 3, 2, 4], 22).
state(3, s, [3, 8, 1, 5, 7, 3, 2, 4], 21).
state(4, s, [4, 8, 1, 5, 7, 3, 2, 4], 23).
state(5, s, [5, 8, 1, 5, 7, 3, 2, 4], 23).
state(6, s, [6, 8, 1, 5, 7, 3, 2, 4], 24).
state(7, s, [7, 8, 1, 5, 7, 3, 2, 4], 22).
state(8, s, [8, 8, 1, 5, 7, 3, 2, 4], 20).
state(9, s, [1, 1, 1, 5, 7, 3, 2, 4], 21).
state(10, s, [1, 2, 1, 5, 7, 3, 2, 4], 20).
state(11, s, [1, 3, 1, 5, 7, 3, 2, 4], 21).
state(12, s, [1, 4, 1, 5, 7, 3, 2, 4], 21).
state(13, s, [1, 5, 1, 5, 7, 3, 2, 4], 22).
state(14, s, [1, 6, 1, 5, 7, 3, 2, 4], 23).
state(15, s, [1, 7, 1, 5, 7, 3, 2, 4], 19).
state(16, s, [1, 8, 2, 5, 7, 3, 2, 4], 23).
state(17, s, [1, 8, 3, 5, 7, 3, 2, 4], 22).
state(18, s, [1, 8, 4, 5, 7, 3, 2, 4], 22).
state(19, s, [1, 8, 5, 5, 7, 3, 2, 4], 22).
state(20, s, [1, 8, 6, 5, 7, 3, 2, 4], 21).
state(21, s, [1, 8, 7, 5, 7, 3, 2, 4], 22).
state(22, s, [1, 8, 8, 5, 7, 3, 2, 4], 23).
state(23, s, [1, 8, 1, 1, 7, 3, 2, 4], 22).
state(24, s, [1, 8, 1, 2, 7, 3, 2, 4], 23).
state(25, s, [1, 8, 1, 3, 7, 3, 2, 4], 24).
state(26, s, [1, 8, 1, 4, 7, 3, 2, 4], 23).
state(27, s, [1, 8, 1, 6, 7, 3, 2, 4], 23).
state(28, s, [1, 8, 1, 7, 7, 3, 2, 4], 24).
state(29, s, [1, 8, 1, 8, 7, 3, 2, 4], 22).
state(30, s, [1, 8, 1, 5, 1, 3, 2, 4], 21).
state(31, s, [1, 8, 1, 5, 2, 3, 2, 4], 22).
state(32, s, [1, 8, 1, 5, 3, 3, 2, 4], 22).
state(33, s, [1, 8, 1, 5, 4, 3, 2, 4], 20).
state(34, s, [1, 8, 1, 5, 5, 3, 2, 4], 21).
state(35, s, [1, 8, 1, 5, 6, 3, 2, 4], 23).
...
state(53, s, [1, 8, 1, 5, 7, 3, 2, 3], 22).
state(54, s, [1, 8, 1, 5, 7, 3, 2, 5], 22).
state(55, s, [1, 8, 1, 5, 7, 3, 2, 6], 23).
state(56, s, [1, 8, 1, 5, 7, 3, 2, 7], 23).
state(57, s, [1, 8, 1, 5, 7, 3, 2, 8], 22).


1. Clear database
2. Execute hcls
3. Display states
4. Save states
5. Exit

Enter your choice: 4.
Enter a new file name:'DB.txt'.

1. Clear database
2. Execute hcls
3. Display states
4. Save states
5. Exit

Enter your choice: 5.
% Execution Aborted
2 ?-
```