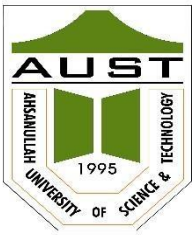
AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

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

Assignment No: 05

Date of Submission: 22/07/2019

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 the variants of hill-climbing and genetic algorithms discussed above in Prolog and Python.

1) **Random restart hill climbing:** If stuck up at a local maximum, then begin with a new randomly generated 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(_,_,_).
```

Hlclmb_random:

```
 :-use_module(eval_state).
 :-use_module(list_apps).
 :-dynamic(state/4). /* id,type,state,h_value*/
 :-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)),
    incr_id, mk_new(1,L), retract(id(_)), evaluate.

incr_id:-id(V), V1 is V+1, retract(id(_)), assert(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(['\nIteration max: ',V2]),
    gnrt_sucsr(L),!.
checkall:- rndm_restrt,!.

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_restrt:- retractall(state(_,_,_,_)), restrt_cntr(V), V<5,
    write('\n\nStuckup! Restarting.\n\n'), restart,!.
rndm_restrt:-write('\n\nStuckup once again! Ending.\n\n').

restart:- incr_r_c, restrt_cntr(V), write_list(['\n\nRestart index: ',V]),
    get_rndm_st(L),gnrt_sucsr(L).

incr_r_c:- restrt_cntr(V), V1 is V+1, retract(restrt_cntr(_)),
```

```
84.     assert(restrt_cntr(V1)).
85.
86. get_rndm_st(L):- assert(list_st([])), lp8(8), list_st(L), retract(list_st(_)).
87.
88. lp8(0):-!.
89. lp8(N):- N1 is N-1, X is random(8)+1, list_st(L1),append(L1,[X],L2),
90.     retract(list_st(L1)),assert(list_st(L2)),lp8(N1).
```

Output:

```
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:12345678.
Enter threshold value:27.

Iteration max: 6
Iteration max: 12
Iteration max: 16
Iteration max: 20
Iteration max: 22
Iteration max: 24
Iteration max: 26

Found! Id:5   s   [5,1,1,6,8,3,7,4]        Value:27

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

Enter your choice:
```

2) Genetic solution for 8-queen problem:

```
3)  :-use_module(eval_state).
4)  :-use_module(list_apps).
5)  :-dynamic(state/4). /* id,type,state,h_value*/
6)  :-dynamic(id/1).
7)  :-dynamic(max_val/1).
8)  :-dynamic(threshold/1).
9)  :-dynamic(restrt_cntr/1).
10) :-dynamic(list_st/1).
11)
12) /* Organizing a Menu */
13) start:- repeat,
14)     write('\n1. Clear database'),
15)     write('\n2. Execute hcls'),
16)     write('\n3. Display states'),
17)     write('\n4. Save states'),
18)     write('\n5. Exit'),
19)     write('\n\nEnter your choice: '),
20)     read(N), N >0, N < 6,
21)     do(N), N=5,!.
22)
23) do(1):- retractall(state(_,_,_,_)),retractall(id(_)), retractall(max_val(_)),
24)     retractall(threshold(_)), retractall(restrt_cntr(_)).
25) do(2):- go_hcs.
26) 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).

/* Performing Crossover */

go_cross(X,Y,CP):- state(X,'p',L1,_), state(Y,'p',L2,_),CP1 is 8-CP,
    del_1st_n_el(L1,CP,L12),del_last_n_el(L1,CP1,L11),
    del_1st_n_el(L2,CP,L22),del_last_n_el(L2,CP1,L21),
    append(L11,L22,LO1),append(L21,L12,LO2), count_sts(_,N),
    N1 is N+1, N2 is N+2,
    assert(state(N1,'o',LO1,50)), assert(state(N2,'o',LO2,50)).

/* Performing Mutation*/

do_mutn:- count_sts('o',N), N1 is random(N)+1,
    assert(id1(0)),get_offspr(N1,I,T,L,V), retract(id1(_)),
    N2 is random(8)+1, N3 is random(8)+1, rplc_nthel(N2,N3,L,L1),
    retract(state(I,T,L,V)), assert(state(I,T,L1,50)).

get_offspr(N1,I,'o',L,V):- state(I,'o',L,V),incr_id1, id1(N), N1=N,!.


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

incr_id:-id(V), V1 is V+1, retract(id(_)), assert(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,_),I1 is I+1,go_cross(I
,I1,L),do_mutn,
    retractall(state(_,_,_,_)),write_list(['\nIteration max: ',V2]),
    gnrt_sucsr(L),!.
checkall:- rndm_restrt,!.

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_restrt:- retractall(state(_,_,_,_)), restrt_cntr(V), V<5,
    write('\n\nStuckup! Restarting.\n\n'), restart,!.
rndm_restrt:-write('\n\nStuckup once again! Ending.\n\n').
      rndm_restrt:-best(I,VI),I1 is I+1,go_cross(I,I1,CP),!.

      restart:- incr_r_c, restrt_cntr(V), write_list(['\n\nRestart index: ',V]),
          get_rndm_st(L),gnrt_sucsr(L).

      incr_r_c:- restrt_cntr(V), V1 is V+1, retract(restrt_cntr(_)),
          assert(restrt_cntr(V1)).

      get_rndm_st(L):- assert(list_st([])), lp8(8), list_st(L), retract(list_st(_)
  ).

      lp8(0):-!.
      lp8(N):- N1 is N-1, X is random(8)+1, list_st(L1),append(L1,[X],L2),
          retract(list_st(L1)),assert(list_st(L2)),lp8(N1).
```