

Session 2: Elements of Informed Search

I. OBJECTIVES

- To be able to implement simple heuristic functions in Prolog and in Python;
- To be able to use heuristic problems for simple search problems.

II. DEMONSTRATION OF USEFUL RESOURCES

Heuristic functions for informed search

A. Heuristic functions for general graph search problems

- A common practice for general graph search problems is to take 'straight line distance' between two nodes, computed somehow, as a heuristic function value.
- We consider this type of heuristics as 'given' for solving problems and will involve in upcoming sessions.

B. Heuristic functions for other types of problems

i) Consider the following instance of the 8-puzzle problem.

Goal state:

1	2	3
8		4
7	6	5

Current state:

8	1	2
3	6	4
	7	5

Prolog representation of the states may have the following form

gtp(1,1,1). gtp(2,1,2). gtp(3,1,3). gtp(4,2,3). gtp(5,3,3). gtp(6,3,2). gtp(7,3,1). gtp(8,2,1). gblnk(2,2).
tp(1,1,2). tp(2,1,3). tp(3,2,1). tp(4,2,3). tp(5,3,3). tp(6,2,2). tp(7,3,2). tp(8,1,1). blnk(3,1).

- We can think of a **heuristic function (h_1)** that determines the number of mismatching tiles.

Possible Prolog code may have the following form:

```
go:- calCH(1,0,H), write('Heuristics: '),write(H).
calCH(9,X,X):-!.    calCH(T,X,Y):- check(T,V), X1 is X+V, T1 is T+1, calCH(T1,X1,Y).
check(T,V):-tp(T,A,B), gtp(T,C,D), A=C, B=D, V is 0,!.    check(_):-!.
```

Possible Python representation and procedure may have the following form:

```
gtp=[(1,1,1), (2,1,2), (3,1,3), (4,2,3), (5,3,3), (6,3,2), (7,3,1), (8,2,1)]
gblnk = (2,2)
tp=[(1,1,2), (2,1,3), (3,2,1), (4,2,3), (5,3,3), (6,2,2), (7,3,2), (8,1,1)]
blnk = (3,1)
```

Procedure to find the number of mismatches

i,h=0,0

while(i<=7):

if ((gtp[i][1] != tp[i][1])|(gtp[i][2] != tp[i][2])):

h=h+1

i=i+1

print('Heuristics 1: ',h)

1	2	3
8		4
7	6	5

8	1	2
3	6	4
	7	5

- We can think of another **heuristic function (h_2)** where Manhattan distances of the tiles are calculated.

Possible Prolog code may have the following form:

```
go:- calcH(1,[],L), sumList(L,V),write('Heuristics: '),write(V).
calcH(9,X,X):-!.    calcH(T,X,Y):- dist(T,D), append(X,[D],X1), T1 is T+1, calcH(T1,X1,Y).
dist(T,V):-tp(T,A,B), gtp(T,C,D), V is abs(A-C) + abs(B-D).
sumList([],0):-!.    sumList(L,V):-L=[H|T], sumList(T,V1), V is V1+H.
```

- ii) Consider the following instance of 8-queens problem and a **heuristic function (h_3)** that returns the number of attacking pairs of queens.

8							Q	
7				Q				
6	Q							
5			Q					
4					Q			
3						Q		
2								
1		Q						Q
	1	2	3	4	5	6	7	8

State I

- Complete-state formulation of problem; I: 61574381, 1st queen is at the 6th row, 2nd queen at the 1st row,
- Any placement of queens can be taken as an initial state, but no fixed goal state.
- h will mean number of pairs of queens that are in attacking position (face to face); $h(I) = 5$; We try to minimize h; Global minimum = 0;

$h(I) = \text{face to face in the row} + \text{face to face diagonally up} + \text{face to face diagonally down} = 1+1+3 = 5$.

How to compute this function using Python?