

Introduction to Artificial Intelligence

Lab - blind search *

This lab will help you familiarise with two blind search strategies, namely breadth-first and depth-first, by implementing them in Prolog extending and/or modifying the Prolog program in file `coreSearch.pl`:¹

```
search(Paths,X):-
    choose([Node|Path],Paths,_),
    goal(Node),
    reverse([Node|Path],X).
```

```
search(Paths,Path):-
    choose(P,Paths,RestofPaths),
    findall([S|P],S expands P,Exps),
    combine(Exps,RestofPaths, NewPaths),
    search(NewPaths,Path).
```

```
NewState expands [State|_]:-
    arc(State,NewState).
```

where `arc` and `goal` define any given search problem (e.g. see files `testExample1.pl` and `testExample2.pl` for examples).

1. Extend `coreSearch.pl` to implement the breadth-first search strategy.
2. Extend `coreSearch.pl` to implement the depth-first search strategy.
3. Test both implementations with the search problem in `testExample1.pl` and compare the results.
4. Is the given implementation for depth-first search efficient? Could a more efficient implementation of depth-first search be obtained, by exploiting Prolog built-in back-tracking mechanism? If so, provide a new program.

*Thanks to Keith Clark

¹This program corresponds to the basic search algorithm on slide 11 on Search.

5. Test all implementations with the simple search space in file `testExample2.pl` (this search space contains a cycle) and compare/evaluate the results.
6. Modify the program in file `coreSearch.pl` to perform breadth-first and depth-first in the presence of loops (namely incorporating loop-checking).
7. Test the revised program with the simple search space in file `testExample2.pl` and compare/evaluate the results.

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs