

Object-Oriented Programming – 2006/07

Homework 4

Using an object-oriented language of your choice¹, develop a *framework* for solving *state space searching* problems.

1. Define entities for representing *state-space* problems. Implement the *water jugs* and the *missionaries and cannibals* problems.
2. Define entities for representing *search methods*. Implement the *depth-first* and *breadth-first* search methods.
3. Taking into account the chosen programming language and what you learned while developing the state-space searching framework, answer the following questions:
 - a) Describe the advantages of defining and implementing state-spaces and search methods as orthogonal concepts.
 - b) The goal of a *framework* is to help solving a whole class of problems, not a single, specific problem. Describe how to use a *framework* to solve a new state-space problem or to add a new search method.
 - c) One of the most commonly used *frameworks* when developing graphical user interfaces is the *model-view-controller (MVC)* framework. Find out how you may use this framework from within your favorite programming language and operating system.

You may present and discuss your answers during classes or during your teacher's office hours.

Notes, resources, and bibliography:

The depth-first and breadth-first search methods may be implemented using, respectively, a *stack* and a *queue*. In order to prevent cycles when searching, you may use a *list* to store nodes already visited.

Add a maximum depth parameter to all search methods. This parameter should be used to limit the maximum depth to be explored when expanding the state-space graph.

An example of a full implementation of this *framework* is available with the Logtalk object-oriented programming language (<http://logtalk.org/>).

¹ The use of the Java programming language is recommended, as this language is used in several other disciplines. In addition, is advisable to chose a second programming language allowing a broad overview of object-oriented programming concepts.

State-space search is a classical problem, often described in the Artificial Intelligence bibliography. Useful information on this problem is also easily found on the web. E.g.:

http://en.wikipedia.org/wiki/State_space

http://en.wikipedia.org/wiki/State_space_search

http://en.wikipedia.org/wiki/Depth-first_search

http://en.wikipedia.org/wiki/Breadth-first_search

http://en.wikipedia.org/wiki/Missionaries_and_Cannibals_Problem

http://en.wikipedia.org/wiki/Fox%2C_goose_and_bag_of_beans_puzzle

<http://www.mathsisfun.com/puzzles/measuring-puzzles-index.html>

Complementary work suggestions:

Add support for heuristic state spaces and heuristic search methods. For example, you may try to implement the *eight puzzle* problem and the *hill-climbing* search method.

Define a *performance* interface specifying a set of methods for evaluating the performance of search methods when solving a state-space searching problem. Similar functionality can be found on the Logtalk *searching* example.

Implement a graphical interface that allows a user to select a state space problem and a search method. The interface should also allow the user to specify the maximum depth to be explored when expanding the state-space graph.