Foundations of Artificial Intelligence

M. Helmert, T. Keller
S. Eriksson
Spring Term 2020

University of Basel
Computer Science

Exercise Sheet 6

Due: April 8, 2020

Important: For submission, consult the rules at the end of the exercise. Non-adherence to the rules will lead to your submission not being corrected.

Exercise 6.1 (1.5+1.5 marks)

Show with a counterexample that the following two statements are not correct:

(a) A* without reopening using an admissible but not consistent heuristic is optimal.

(b) A* without reopening using a consitent but inadmissible heuristic is optimal.

Exercise 6.2 (1.5+1.5 marks)

Formalize the following problems as a combinatiorial optimization problem (COP). Is the COP a pure search problem, a pure optimization problem, or a combined search and optimization problem?

(a) Given an undirected graph $G = \langle V, E \rangle$ with $V = \{v_1, \ldots, v_{|V|}\}$, the GraphColoring problem aims to find the minimum number of colors needed to color each vertex $v \in V$ such that no two verticies v_i and v_j with $\{v_i, v_j\} \in E$ have the same color.

Hint: You can use natural numbers to denote different colors.

(b) In the *traveling salesperson problem* (TSP), a salesperson is trying to find the shortest roundtrip around n cities. More formally, we are given a set of locations L and a total and symmetric cost function $c : L \times L \to \mathbb{N}_0$, and try to find a an order of all locations that incurs the lowest cost when visited in that order.

Exercise 6.3 (1+3+0.5+1.5 marks)

The task in this exercise is to write a software program. We expect you to implement your code on your own, without using existing code (such as examples you find online). If you encounter technical problems or have difficulties understanding the task, please let us know.
Download the file `hill-climbing.tar.gz` from the website of the course. It contains an incomplete implementation of hill climbing search for the 8 queens problem that was presented in the lecture.

(a) Implement the heuristic for the 8 queens problem that is presented on Slide 22 of Chapter 20 (print version), where the heuristic value is equal to the number of pairs of queens threatening each other. To do so, implement the function `public int h(Configuration _conf)` in the file `EightQueensProblem.java`.

(b) Implement hill climbing in the function `protected SearchResult search()` in the file `HillClimbing.java`. Since our heuristic is such that smaller values are better, we are considering a minimization variant here, so adapt the function presented on Slide 20 of Chapter 20 (print version) accordingly. Break ties among neighbors with minimal heuristic value uniformly at random. Note that `protected SearchResult search()` returns a `SearchResult` object, which contains information if hill climbing found a solution and on the number of steps.

(c) Test your implementation by verifying the statements on Slide 23 of Chapter 20 (print version), which state that hill climbing with a random initialization finds a solution in around 14% of the cases using around 4 steps on average. You can compile and run your code with `javac HillClimbing.java` followed by the command `java HillClimbing 8queens`. Report the percentage of successful runs and the average number of steps.

(d) Copy your hill climbing implementation into a new file `HillClimbingWithStagnation.java`. Adapt the implementation such that steps without improvement (stagnation) are allowed as described on Slide 8 of Chapter 21 (print version). Verify that approximately 94% of the runs with a bound of 100 steps yield a solution, and that if a solution is found, it took 21 steps on average. What is the percentage of successful runs and the average number of steps in case of success for your solution?

Submission rules:

- Create a single PDF file (ending .pdf) for all non-programming exercises. If you want to submit handwritten parts, include their scans in the single PDF. Put the names of all group members on top of the first page. Use page numbers or put your names on each page. Make sure your PDF prints on A4 (fits the page size).
- For programming exercises, create only those Java textfiles (ending .java) required by the exercise. Put your names in a comment on top of each file. Make sure your code compiles and test it!
- For the submission, you can either upload the single PDF or prepare a ZIP file (ending .zip, .tar.gz or .tgz; not .rar or anything else) containing the single PDF and the Java textfile(s) and nothing else. Please do not use subdirectories in the ZIP.
- Only upload one submission per group. Do not upload several versions, i.e., if you need to resubmit, use the same file name again so that the previous submission is overwritten.