Exploring the Unknown: An Agent in Graphland

Imagine you made it your personal goal to visit all pubs of the city you just moved to and know nothing about. Since you are ambitious, you want to do it in one night, but as you walk into the first pub next door you notice that you do not have a map of the city. Luckily, on one wall you find a list of some of the closest pubs with information how far away they are and how to get there. As you are assured, such a list is available in every pub of the city, so you decide to conduct your quest by using only the information on these lists. Of course, you want to minimize the total distance you have to walk during your endeavor, but, after all, how much could a map help if you have these lists? Probably not too much, right?

This is one of the main open questions in online graph exploration, a research area that examines algorithms for exploring all nodes of an unknown edge-weighted input graph where upon arriving at a new node, the algorithm obtains knowledge of the costs (= weights) of the incident edges. How good such an online algorithm is, is measured by comparing the accumulated costs of the exploration to the costs an optimal algorithm that knows the input graph in advance would incur. The ratio between the costs of the online and the optimal algorithm for the worst possible input graph is called competitive ratio. Now, we can rephrase the question above as follows: What is the competitive ratio of the best online algorithm?

One of your tasks is to study the case where the number of different edge weights a graph may have is restricted by a (small) constant. What is the exact competitive ratio if only 2 weights are allowed? What about 3? What can we infer for graphs with more distinct edge weights? Another possible task is to improve the known lower bound of 2.5 for the general case by computational means. How can we construct graphs (or more precisely graph classes) on which all algorithms have a competitive ratio of more than 2.5?

Requirements: An affinity to (graph) theory will be very helpful as the ideas involved are of a somewhat elaborate theoretical nature. But don’t worry, you will meet on a weekly basis with your advisor to discuss progress and open questions. Additionally, some programming experience will come in handy as you probably will have to write code in this thesis.

Interested? Just drop me an email and we fix a time for a small chat.

Contacts

• Sebastian Brandt: brandts@tik.ee.ethz.ch, ETZ G 61.4