Lesson: Approximate Solutions

In this lesson, we discuss how to consider a tradeoff between speed and correctness in the case of approximate algorithms. Such tradeoff need to be considered whenever exact solutions are not feasible.

Video - Approximate solutions

0:00 / 0:00 1.50x

Start of transcript. Skip to the end.

So, here we are again, nice to see you've made it to week 5 ! Today we'll talk about

comparing implementations.

When facing practical problems like the Traveling Salesman Problem, the TSP, we'd obviously

prefer the solution that's both fast and correct.

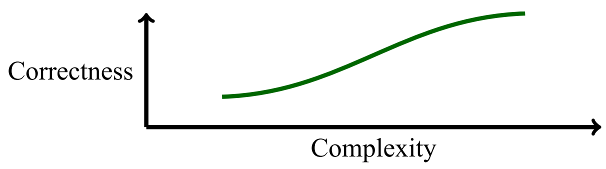
In the case of NP-complete problems, which is the case for the TSP, we know

Compare different approximate solutions to a problem in terms of accuracy and performance

So, here we are again, nice to see you've made it to week 5! Today we'll talk about comparing implementations. When facing practical problems like the Traveling Salesman Problem, the TSP, we'd obviously prefer the solution that's both fast and correct.

Pareto frontier and Pareto optimal

In the case of NP-complete problems, which is the case for the TSP, we know that this just isn't possible. We have to sacrifice speed, correctness, or a bit of both. This kind of trade-off is often referred to as a Pareto frontier, which we can define in the following way: what's the fastest algorithm we can find to solve a problem with a given amount of correctness?



We call Pareto optimal a point that is lying on a Pareto frontier, which means it corresponds to an optimal trade-off between speed, and correctness. A Pareto optimal is typically an optimal that depends on the relative importance of two or more parameters, such as the size of the problem we are dealing with.

Measuring speed and correctness

Let's define some notions that are related to measuring speed and correctness. We can measure how fast an algorithm is using complexity or measures of time of execution. But the correctness of an algorithm is tricky to define, which makes it difficult to measure.

For the TSP, we know that we want to find a shortest path, and so correctness could simply be measured as a deviation from the length of the shortest path.

But in practice, it isn't always so simple. Finding the actual shortest path can be impossible if there are more than a few dozen vertices, as we saw that the complexity of an exhaustive search scales exponentially. This means that correctness of an algorithm approaching a solution for the TSP is very hard to estimate.

Comparing algorithms

What we know is that the more complexity we have to deal with, the better the solution should be. In the worst case, we can

What we know is that the more complexity we have to deal with, the better the solution should be. In the worst case, we can always rely on a less complex solution that has already been proposed.

Let's take the TSP again as an example. There are many intermediate steps between the greedy algorithm, which corresponds to always targetting the next closest city, and the brute force one, which exhaustively examines all possible paths to visit every city. Finding an intermediate between these two extremes should only be considered if they actually provide improvements in complexity.

To determine whether a proposed solution is efficient, you should always compare it with a simple approach, both in terms of correctness and complexity. If your solution needs to be 10 times more complex to provide a 1 % improvement in terms of correctness, maybe it's not worth the effort, except if this 1 % improvement makes a significant difference compared with other solutions.

That's it for today, thanks for your attention. This is my last video ! I really enjoyed teaching for you guys. I will leave you in the hands of Patrick and Vincent for Week 6 on combinatorial game theory. Bye bye !