CS 6840 Algorithmic Game Theory January 22, 2020

Lecture 1: Introduction

Instructor: Eva Tardos *Scribe: Eva Tardos*

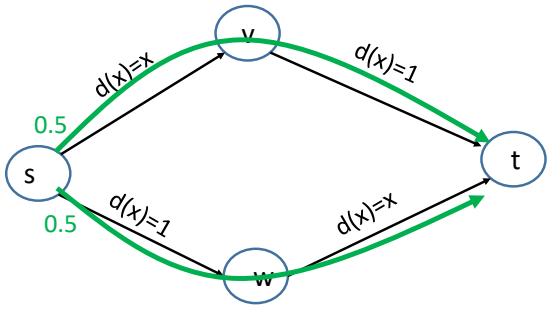
Algorithmic Game Theory combines algorithms and thinking about outcomes when participants act in their own self interest. With large online systems connecting large number of users this combination of algorithms having to deal with participants of varying interests has many important applications, which include applications to

- traffic routing (both cars on highways as well as packets)
- matching, such as matching cars to requests in Lyft and Uber
- Fair sharing of bandwidth or other resources between processes
- more generally will try to also pay attention to fairness issues in the outcomes of selfish interactions

A first example

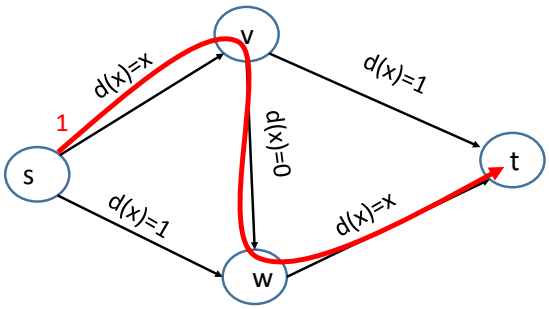
The first part of the course will focus on understanding quality of solutions achieved by selfish users. A simple and intuitive example of this phenomenon is the Braess Paradox. Consider the following network on 4 nodes

Each edge has a congestion sensitive delay, with x amount of flow is on the edge, it takes $d(x)$ time to traverse the edge. Now assume that there is 1 unit of traffic that needs to get from s to t (we mean something line 1 thousand cars, so imagine 1 unit here is big and flow can be divided fractionally). The solution shown on the figure in green is sending $1/2$ the flow along the s, v, t path and the other half along the s, w, t path. This way the traffic amount is $1/2$ on all edges, and the total time from s to t is 1.5, with $1/2$ on the edge with $d(x) = x$ and 1 on the edge with $d(x) = 1$. This solution is actually optimal, that is getting the flow from s to t as fast as possible, but we won't prove this now.



We will say that this solution is at *equilibrium* as now piece of the flow can be better off by switching path: moving to the other path only makes the delay worse.

Instead consider adding an edge (v, w) to the network with $d(x) = 0$, i.e., instantaneous transportation. Note that the green flow is no longer at equilibrium, as the path s, v, w, t now takes only 1 unit of time. Assume all the flow switches to this path, we get the solution shown below with delay 2, as x is now 1 in both congestible edges.

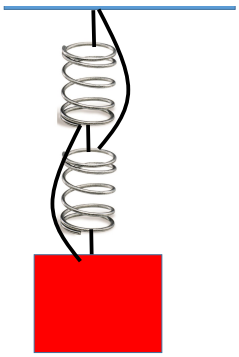


Note however that this is again at equilibrium, if any part of the flow switches to its old path, that doesn't help make the this flow get to its destination faster! In fact, it is not hard to see that this is the only flow at equilibrium, sending any flow on either of the old paths, makes the s, v, w, t path better. Of course the old solution with delay 1.5 is still possible in this network, just not an equilibrium. In this example, the solution deteriorates by a factor of $2/1.5 = 4/3$ due to the selfish path choice of the packets or cars.

It was well know to economists for a very long time that equilibrium may not be optimal, and there are many studies aiming to understand what selfish situations lead to equilibria that are also optimal solutions. The question of quantifying the loss of quality due to selfish behaviour is something computer scientist tend to ask. When arranging a better solution would be complicated or costly, it makes sense to want to know how much one can gain by organizing the solution better. In the next few lectures we will show that this is the worst case. So while a deterioration of a factor of $4/3$ is not ideal, it doesn't get worse by thinking about really large networks.

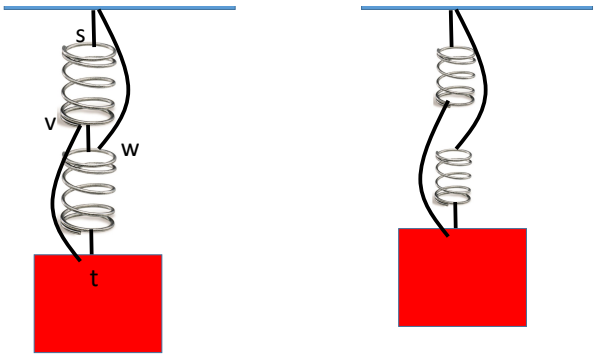
Braess' paradox with springs

As a detour consider the following physics question. The figure illustrates two springs connected by strings (in black). The red square represents a heavy weight that is hanging from the ceiling (in blue). As you know from physics the springs are stretched out due to the weight they have to support. Now suppose we cut the central small strong connecting the two springs.



The physics question is this: what will happen to the weight. The paradoxical fact is that cutting this

string will make the weight rise!! See below



To understand why the weight rises: a physical explanation is that the first version the weight was stretching both strings with its full force, while without the central string, the two springs each only carry 1/2 of the weight each. And springs stretch proportional to the weight they carry, so with less weight both springs will be shorter.

Nice to connect this physical example to the Braess paradox: notice the connection noted on the left of the second figure. This is the same 4 node graph. The strings have fixed length, while the length of the spring is proportional to the weight it carries. Just like the delay in the Braess paradox that is proportional to traffic. So traffic in the Braess paradox corresponds to weight, and delay corresponds to length in the spring example. But the point is, both are the same mathematical model.

Do players find equilibrium?

Another important question we will ask: what happens if users want to find a good solution. Turns out that the concept of players in games finding an equilibrium is rather problematic. Maybe the two most important issues are

- Finding equilibrium in many games is computationally hard
- the players need a lot of information to think about this, e.g., at least they need to know who all is there participating

The first item are more recent computer science results, but the second item is a widely held historic criticism of the equilibrium concept. We will consider these issues, but also take a different direction.

A natural way to try to find a good solution, at least if one can play the game repeatedly is by using a learning algorithm. The topic of learning in games dates back to the work of Julia Robinson from 1952, who showed that in certain simple games such learning finds an equilibrium. Unfortunately, this turned out a rather rare case, which is not surprising in light of the complexity result. We will consider what happens in a game when all the players learn, even if they do not find the equilibrium.

Prerequisites and course expectations

The prerequisite for undergraduates of this course is CS 4820: Algorithms. Concretely we will assume

- basic algorithms on graphs, such as flows on graphs as used by the Braess paradox
- NP-completeness, in arguing that finding equilibrium may be hard
- ability to prove things, as this is a theory course, the main topics will be proving things about selfish systems
- using probability and expectations. The Braess paradox is a deterministic system but many systems we encounter have a lot of randomness, which we will want to be able to work with