Online Algorithms for Configuration Games

Consider a typical situation occurring on daily basis in a datacenter. Such a place contains some number of tightly connected physical machines, each having a fixed capacity that allows it to run a specific number of virtual machines (VMs). The VMs are computational nodes that need to communicate with each other. While the communication between two VMs running on a single physical machine is basically free, if they are running on different physical machines, such communication incurs a certain cost corresponding to the incurred network load and latency.

A classic optimization objective would be to compute a placement of the VMs (a static configuration) so that the capacities of physical machines are not exceeded, and the total incurred communication cost is minimized. Computing a good fixed configuration is computationally hard already when the communication patterns are known up-front.

In reality, however, the communication requests are generated on the fly by computations running at particular VMs, and hence they are rarely known in advance. In effect, any static placement of VMs can be quickly rendered non-optimal. Modern virtualization solutions provide a way to deal with this issue on the technical level: we may migrate a given active VM to a new physical machine. Such action incurs a certain cost associated with suspending, migrating over a network, and resuming. On the algorithmic level, however, this poses a major paradigm shift: the communication requests appear *in an online manner*, one by one, and in response, a partitioning algorithm may modify the placement of VMs (change from one configuration to another). These algorithmic decisions have to be made solely on the basis of past communication requests and without knowledge about the future ones.

A dynamic partitioning problem described above is an example of an *online optimization problem*, where the goal is to maintain a *configuration* that allows to cheaply serve a sequence of unpredictable requests. To reduce the service cost, it is possible to change the configuration to another one, potentially better suited to the requests, but such choices have to be made without the knowledge of future requests. We call any problem that fits this description *configuration game*. Configuration games occur as crucial building blocks in many applications ranging from telecommunication and computer networks, through the organization of data centers, logistics and planning, to methods for efficient data structures.

Moving from a particular graph partitioning problem to the class of configuration games allow us to abstract away from the particular flavor of a given problem, and build methods and solutions that are applicable for a wide range of configuration games. That said, the generic solutions have to be fine-tuned later to the geometry of the configuration space and types of allowed requests. The goal of this project is to *construct and rigorously analyze algorithms for configuration games under uncertainty*.