Abstract

Scheduling and *routing* are classical fields of theoretical computer science which in the last decades gained a lot of flourish attention as tools for improving the performance of large-scale computer systems. The aim of both scheduling and routing is to ensure an efficient use of such a computer system. The objective of scheduling is to determine an *optimum assignment* of *jobs* to (one or more) *processors* with respect to some performance measure, whereas the goal of routing is to efficiently ship (splittable or unsplittable) *packets* through a common processor network.

The theory of scheduling and routing is characterized by a virtually unlimited number of problem types. In this thesis, we analyze three of them. Since they are independent, we extensively study them in three self-contained chapters.

- **Scheduling Identical Malleable Jobs:** We consider the problem of finding a *non-preemptive* schedule for *independent malleable identical jobs* on *identical processors* with *minimum total completion time*, the so-called *makespan*.
- **Flow Scheduling:** We consider *synchronous distributed processor networks*. We assume that the load on the processors consists of independent load units, called *tokens*. In order to balance the network, it is necessary to migrate parts of the processors' loads during runtime. We migrate the load according to a given *balancing flow* with the goal to use the minimum number of rounds to reach the balanced state.
- **Selfish Routing in Non-Cooperative Networks:** We consider a routing game, widely known as the *KP-model*. In this model, non-cooperative *users* wish to route their unsplittable *traffics* through a very simple network of *parallel links*. Each user employs a mixed strategy, trying to minimize its *expected latency*. A stable state in which no user has an incentive to unilaterally change its strategy is called a *Nash equilibrium*. We analyze how to compute a Nash equilibrium, and the influence of the selfish behavior of the users to the global system.