Mean-Payoff Optimization in Continuous-Time Markov Chains with Parametric Alarms

Ľuboš Korenčiak
Masaryk University, Brno, Czech Republic
korenciak@fi.muni.cz

Mean-payoff is widely accepted as an appropriate concept for measuring long-run average performance of systems with rewards or costs. In this presentation, we study the problem of synthesizing parameters for (possibly non-exponentially distributed) events in a given stochastic system to achieve an \( \varepsilon \)-optimal mean-payoff. One simple example of such events are timeouts widely used, e.g., to prevent deadlocks or to ensure some sort of progress in distributed systems. In practice, timeout durations are usually determined in an ad-hoc manner, requiring a considerable amount of expertise and experimental effort. This naturally raises the question of automating this design step, i.e., is there an algorithm synthesizing optimal timeouts?

In this presentation, we use continuous-time Markov chains with alarms (ACTMCs) as the underlying stochastic model. Intuitively, ACTMCs extend continuous-time Markov chains by generally distributed alarm events, where at most one alarm is active during a system execution and non-alarm events can disable the alarm. In parametric ACTMCs, every alarm distribution depends on one single parameter ranging over a given interval of eligible values. For example, a timeout is a Dirac-distributed alarm event where the parameter specifies its duration. A function assigning to every alarm a parameter value within the allowed interval yields a (non-parametric) ACTMC. Thus, the semantics of a parametric ACTMC is a family of ACTMCs induced by some parameter function. We aim towards an algorithm that synthesizes a parameter function for an arbitrarily small \( \varepsilon > 0 \) achieving \( \varepsilon \)-optimal mean-payoff.

Motivating example. To get some intuition about the described task, consider a dynamic power management of a disk drive inspired by [3]. The behavior of the disk drive can be described as follows (see Figure 1): At every moment, the drive is either active or asleep, and it maintains a queue of incoming I/O operations of capacity \( N \). The events of arriving and completing an I/O operation have exponential distributions with rates 1.39 and 12.5, respectively. When the queue is full, all newly arriving I/O operations are rejected. The I/O operations are performed only in the active mode. When the drive is active and the queue becomes empty, an internal clock is set to \( d_s \). If then no further I/O request is received within the next \( d_s \) time units, the sleep event changes the mode to

When the drive is asleep and some I/O operation arrives, the internal clock is set to $d_w$ and after $d_w$ time the wakeup event changes the mode to active. We annotate costs in terms of energy per time unit or instantaneous energy costs for events. The power consumption is 4 and 2 per time unit in the states active and asleep, respectively. Moving from asleep to active requires 4 units of energy. Rejecting a newly arrived I/O request when the queue is full is undesirable, penalized by costs of 6. All other transitions incur with cost 1. Obviously, the designer of the disk drive controller has some freedom in choosing the delays $d_s$ and $d_w$, i.e., they are free parameters of Dirac distribution. However, $d_w$ cannot be lower than the minimal time required to wake up the drive, which is constrained by the physical properties of the hardware used in the drive. Further, there is also a natural upper bound on $d_s$ and $d_w$ given by the capacity of the internal clock. Observe that if $d_s$ is too small, then many costly transitions from asleep to active are performed; and if $d_s$ is too large, a lot of time is wasted in the more power consuming active state. Similarly, if $d_w$ is too small, a switch to the active mode is likely to be invoked with a few I/O operations in the queue, and more energy could have been saved by waiting somewhat longer; and if $d_w$ is too large, the risk of rejecting newly arriving I/O operations increases. Now we may ask the following instance of an optimal parameter synthesis problem we deal with in this presentation:

What values should a designer assign to the delays $d_s$ and $d_w$ such that the long-run average power consumption is minimized?

**Contribution.** The main result is a symbolic algorithm for $\varepsilon$-optimal parameter synthesis that is generic in the sense that it is applicable to all systems where the optimized alarm events satisfy four abstractly formulated criteria. We show that these criteria are fulfilled, e.g., for timeout events modeled by Dirac distributions, uniformly distributed alarms (used in, e.g., in variants of the CSMA/CD protocol [1]), and Weibull distributions (used to model hardware failures [2]). For a given $\varepsilon > 0$, our algorithm first computes a sufficiently small discretization step such that an $\varepsilon$-optimal parameter function exists even when its range is restricted to the discretized parameter values. Since the discretization step is typically very small, an explicit construction of all discretized parameter values
and their effects is computationally infeasible. Instead, our algorithm employs a symbolic variant of the standard policy iteration technique for optimizing the mean-payoff in semi-Markov decision processes. It starts with some parameter function which is gradually improved until a fixed point is reached. In each improvement step, our algorithm computes a small candidate subset of the discretized parameter values such that a possible improvement is realizable by one of these candidate values. This is achieved by designing a suitable ranking function for each of the optimized events, such that an optimal parameter value is the minimal value of the ranking function in the interval of eligible parameter values. Then, the algorithm approximates the roots of the symbolic derivative of the ranking function, and constructs the candidate subset by collecting all discretized parameter values close to the approximated roots. This leads to a drastic efficiency improvement, which makes the resulting algorithm applicable to problems of realistic size.

References