Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. Susanne Albers
Sonja Lauer, Tim Nonner

Freiburg, April 28, 2008
Summer Term 08

Energy Efficient Algorithms
Assignment 1

Hand in on Monday, May 19, 2008, during the lecture.

Exercise 1: Two-State Power-Down. Consider the two-state power-down problem with the low-power state *sleep* and the high-power state *active*. The transition from the active state to the sleep state and back to the active state consumes D energy units. Unlike in the lecture, we now assume that the sleep state accounts for ϵ energy units per time unit ($0 < \epsilon < 1$). During the active state the energy consumption is still one energy unit per time unit.

a) What is the optimal strategy for the offline scenario of this problem?

b) On the lines of algorithm *PD–2* given in the lecture formulate a corresponding online algorithm A for this problem. Show that A is $(2 - \epsilon)$-competitive.

Exercise 2: Lower Bound for Two-State Power-Down. Show that for the power-down problem given in Eexercise 1 no deterministic online algorithm can achieve a competitive ratio smaller than $(2 - \epsilon)$.

Exercise 3: Randomized Power-Down. Consider the class of algorithms $RAND$ that is defined as follows. Each algorithm $A \in RAND$ is defined by a parameter α and a probability p ($0 \leq p \leq 1$, $0 \leq \alpha \leq 1$). With probability p, A powers down after $\alpha \cdot D$ time units in each idle period. Otherwise, A powers down after D time units in each idle period. Hence, all power down times of A depend on a single initial random choice.

a) Find the algorithm having the best competitive ratio in $RAND$ with $\alpha = 1/2$.

b) Use your favorite programming language to check numerically whether there is a better choice for α by plotting the graph of the best competitive ratio with respect to α for each $0 \leq \alpha \leq 1$.

Exercise 4: Minimax-Principle. Use the Minimax-Principle to prove a lower bound greater than 1.5 on the competitive ratio of the algorithms in $RAND$ with $\alpha = 1/2$. Note that using the Minimax-Principle, it is actually possible to show that the algorithm found in Exercise 3 is optimal.