

## Energy Efficient Algorithms Assignment 4

Hand in on Monday, June 30, 2008, during the lecture.

**Exercise 1: Combined scheduling and power-down strategies.** Consider a processor with two states that consumes one energy unit per time unit in active mode and no energy in sleep mode. Furthermore, the release times and deadlines of the unit-sized jobs are integers. Show that if the cost  $D$  for powering down is at most 1, then for any set of jobs, the problem of finding a schedule that minimizes the total energy consumption is equivalent to the problem of finding a schedule that minimizes the number of idle periods.

**Exercise 2: Earliest deadline first.** For a processor with two states as in Exercise 1 suppose you are given a *profile* that predefines at which times the processor powers down and up. A given set of jobs with varying processing times must now be scheduled complying this profile. (Note that here the energy consumption is fixed due to the given profile.) Thereby, it is allowed to interrupt the processing of a job and to resume it later.

Show that if there exists a feasible schedule for the given profile and the set of jobs, then the strategy *Earliest Deadline First (EDF)* that, at each point in time, processes the job with the earliest deadline will find one.

**Exercise 3: Combined scheduling and power-down with constant length job intervals.** In many practical situations, the time interval during which a job needs to be processed is quite small. In this case, we can assume that there is a constant  $c > 0$  such that  $d_i - r_i \leq c$  for each job  $i$ . Give an algorithm for this restricted problem whose running time is at most  $O(n^2)$ . Observe that  $O(n)$  is possible.

**Exercise 4: NP-hardness of combined scheduling and power-down for non-unit job sizes.** Consider the combined scheduling and power-down problem for arbitrary job-sizes. Specifically, each job  $i$  has a size  $p_i$ , and we need  $p_i$  time units to process  $i$ . Prove that it is quite unlikely to find a polynomial time algorithm for this extension by proving its NP-hardness.

*Hint:* Reduce PARTITION to this problem. PARTITION is the following problem: Given a set of weights  $w_1, w_2, \dots, w_n$  with  $n$  even, decide if we find two equal-sized subsets  $W_1, W_2$  of these weights, i.e.,  $|W_1| = |W_2| = n/2$ , with  $\sum_{w \in W_1} w = \sum_{w \in W_2} w$ .