Assignment 3 for Approximation Algorithms and Hardness of Approximation

Discussion: Thursday, 22 May 2014, 14 pm

Assignment 1 (Tightness of List Scheduling on \( P|\text{prec},p_j = 1|C_{\text{max}} \))

Assume we are given jobs \( J = \{1, \ldots, n\} \) to be scheduled on \( m \) parallel identical machines, together with a precedence relation \( \prec \subseteq J \times J \) such that each job \( j \) cannot be scheduled before all jobs \( k < j \) are completed. In the lecture, it was proven that List Scheduling gives a 2-approximation for minimizing the makespan. Show that this approximation guarantee is (almost) tight even for unit processing times, i.e., give instances for which \( p_i = 1 \) for all \( i \in J \) and List Scheduling achieves an approximation factor of exactly \( 2 - \frac{1}{m} \).

Extra: How could one improve the analysis of the lecture to a \( (2 - \frac{1}{m}) \)-approximation guarantee?

Assignment 2 (FPTAS for \( Pm||C_{\text{max}} \))

Assume we have a fixed constant number of identical machines \( m \). Give an FPTAS for minimum makespan scheduling, i.e., show how to compute a \( (1 + \varepsilon) \)-approximation to the optimal schedule in time polynomial in \( n \) and \( \frac{1}{\varepsilon} \).

Extra: Solve the scheduling problem optimally in \( (\omega(\frac{\varepsilon}{n}) \cdot n)^O(\text{dynamic programming approach}) \cdot |J|O(\text{rounding time}) \cdot \text{pseudopolynomial time}) \cdot (\omega(\text{dynamic programming}) \cdot |J|O(\text{rounding time})) \).

Hint: Solve the scheduling problem optimally in \( (\omega(\frac{\varepsilon}{n}) \cdot n)^O(\text{dynamic programming approach}) \cdot \text{pseudopolynomial time}) \cdot \text{pseudopolynomial time}) \).

Assignment 3 (2-approximation for \( Q||C_{\text{max}} \))

Assume that instead of all \( m \) machines being identical, each machine has an associated speed \( s_i \). If job \( j \) is processed on machine \( i \), it takes a time of \( \frac{p_j}{s_i} \) until completion. For a schedule \( S \), let \( S_i \) be the set of jobs scheduled on machine \( i \), then machine \( i \)'s load is \( \ell_i := \frac{\sum_{j \in S_i} p_j}{s_i} \). We aim to find a schedule minimizing the makespan \( \max_{i=1,\ldots,m} \ell_i \). The following steps will yield a 2-approximation for this problem:

1. Assume we have a polytime procedure that, given a guess \( T \) on the optimum, outputs either \( \text{OPT} > T \) or outputs a schedule \( S \) with makespan at most \( 2T \). Show how to compute a 2-approximation in polynomial time.

2. Show that the following algorithm provides such a procedure: Let the machines be ordered by decreasing speed, i.e., \( s_1 \ge s_2 \ge \cdots \ge s_m \). For each \( 1 \le i \le m \), let \( J_i := \{ j \in J \mid \frac{p_j}{s_i} \le T \} \). Consider the machines \( i = m, m-1, \ldots, 1 \) from slowest to fastest and schedule jobs \( j \in J_i \) (in arbitrary order) on machine \( i \) until either no unscheduled jobs are left in \( J_i \) or the last scheduled job leads to \( \ell_i \ge T \). After this is completed, return this schedule if all jobs are scheduled or say that \( \text{OPT} > T \) if there are unscheduled jobs remaining.

Assignment 4 (Solving \( P|r_j,p_j = 1|C_{\text{max}} \) optimally)

Assume that jobs have release dates, but unit processing times \( p_j = 1 \). Show how to compute a minimum-makespan schedule in polynomial time.