18.415/6.854 Advanced Algorithms	December 5, 2001
Problem Set 6	
Lecturer: Michel X. Goemans	*Due: December 12, 2001*

1. The *betweenness problem* is defined as follows: We are given n and a set T of m triples of the elements of $\{1, \ldots, n\}$. We say that an ordering π of $\{1, \ldots, n\}$ satisfies a triple (i, j, k), if j is between i and k in π. (For example, the ordering $(5, 3, 1, 2, 4)$ satisfies the triples $(5, 1, 2)$ and $(1, 3, 5)$, but not $(3, 2, 1)$). The question is to find an ordering of $\{1, \ldots, n\}$ that satisfies the maximum number of triples in T.

   This problem is known to be NP-hard, even if we restrict to instances for which an ordering that satisfies all the triples exist.

   (a) Use randomization to find a simple $\frac{1}{3}$-approximation algorithm for this problem. Prove the correctness of your algorithm.

   (b) Use the method of conditional expectations to derandomize your algorithm.

   (c) Assume there is an ordering that satisfies all the triples in T. Prove that there are vectors $v_1, \ldots, v_n \in \mathbb{R}^n$ such that

   $$\begin{aligned} ||v_i - v_j|| &\geq 1 \qquad \text{for all } i, j, \\ (v_i - v_j)(v_k - v_j) &\leq 0 \qquad \text{for all } (i, j, k) \in T \end{aligned} \tag{1}$$

   Show how we can find such $v_1, \ldots, v_n$ using semidefinite programming.

   (d) Give an example where the program (1) is satisfiable, but there is no ordering that satisfies all the triples in T.

   (e) Assume that $v_1, \ldots, v_n \in \mathbb{R}^n$ is a solution of the program (1). Choose r uniformly at random from $\{p \in \mathbb{R}^n : ||p|| = 1\}$, and consider the ordering obtained by sorting the elements of $\{1, \ldots, n\}$ with respect to their $r^T v_i$ value. Show that in expectation this ordering satisfies at least half the constraints in T.

2. Consider the following scheduling problem. We are given n jobs that are all available at time 0 and that can be processed on any of m machines. Each job has a processing time p_j which represents the amount of time a machine (any one of them) needs to process it (without interruption). A machine can only process one job at a time. This scheduling problem is to assign each job to a machine and schedule the jobs so as to minimize $\sum_j p_j C_j$ where C_j represents the time at which the processing of job j completes. (For example, if we have 5 jobs of unit processing time and 3 machines, there are many ways of obtaining an objective function value of $1+1+1+2+2=7$.)

   (a) Show that the problem is equivalent to minimizing $\sum_{i=1}^{n} M_i^2$ where M_i is the total amount of processing time assigned to machine i.

(b) Let $L = \frac{1}{m} \sum_j p_j$ be the average load of any machine. Show that any optimum solution for $\sum_{i=1}^n M_i^2$ will be such that each machine i either satisfy $M_i \leq 2L$ or processes a single job j with $p_j > 2L$.

(c) Assume that $p_j \geq \alpha L$ for some constant $\alpha > 0$ for every job j, and assume that all p_j's can only take k different values, where k is a fixed constant. Design a polynomial-time algorithm for this case.

(d) Assume that $p_j \geq \alpha L$ for some constant $\alpha > 0$ for every job j. Design a polynomial-time approximation scheme for this case.

(e) Not part of the problem set and not graded, just for "fun". Can you also get a polynomial-time approximation scheme without the $p_j \geq \alpha L$ assumption?