Theory of Computation

Homework 5

Due: 2015/1/06

Problem 1 Suppose that there are n jobs to be assigned to m machines. Let t_i be the running time for job $i \in \{1 \ldots n\}$, $A[i] = j$ mean that job i is assigned to machine $j \in \{1 \ldots m\}$, and $T[j] = \sum_{A[i]=j} t_i$ be the total running time for machine j. The makespan of A is the maximum time that any machine is busy, given by

$$\text{makespan}(A) = \max_j T[j].$$

The LoadBalance problem is to compute the minimal makespan of A. Note that LoadBalance problem is NP-hard. Consider the following algorithm for LoadBalance:

for $i \leftarrow 1$ to m **do**
    $T[i] \leftarrow 0$;
end for
for $i \leftarrow 1$ to n **do**
    $\min \leftarrow 1$;
    for $j \leftarrow 2$ to m **do**
        if $T[j] < T[\min]$ **then**
            $\min \leftarrow j$;
        end if
    end for
    $A[i] \leftarrow \min$;
    $T[\min] \leftarrow T[\min] + t_i$;
end for
return A;

Show that this algorithm for LoadBalance is a $\frac{1}{2}$-approximation algorithm, meaning that it returns a solution that is at most $\dfrac{1}{1-\frac{1}{2}} = 2$ times the optimum.

Proof: Let OPT be the optimal makespan. Note that $OPT \geq \max_i t_i$ and $OPT \geq \dfrac{1}{m}\sum_{i=1}^{n} t_i$. Suppose that machine i^* has the largest total running time, and let j^* be the last job assigned to machine i^*. Since $T[i^*] - t_{j^*} \leq T[i]$ for all $i \in \{1, 2, \ldots, m\}$, $T[i^*] - t_{j^*}$ is less than or equal to the average running time over all machines. Thus,

$$T[i^*] - t_{j^*} \leq \frac{1}{m}\sum_{i=1}^{m} T[i] = \frac{1}{m}\sum_{i=1}^{n} t_i \leq OPT. \tag{1}$$

We conclude that $T[i^*] \leq 2 \times OPT$. ■

Problem 2 Define $\mathbf{IP}^*$ as $\mathbf{IP}$ except that the prover now runs in deterministic polynomial space instead of exponential time. Show that $\mathbf{IP}^* \subseteq \mathbf{PSPACE}$. (You cannot use the known fact $\mathbf{IP} = \mathbf{PSPACE}$.)

Proof: Let $L \in \mathbf{IP}^*$, (P, V) be an interactive proof system, V be a probabilistic polynomial-time verifier, P be a polynomial-space prover, c and k be some positive integers, n be the length of the input, $m_i \in \{0,1\}^*$ be ACCEPT/REJECT or the message sent in round i, and $r \in \{0,1\}^{n^k}$ be the random bit string used by V in each round (for brevity, we had assumed r is of the same length in each round). Assume P and V interact for at most n^c rounds, and V accepts or rejects the input before or at round n^c. Construct deterministic TM M to simulate (P, V) as follows. Assume without loss of generality that V sends the first message. In the algorithm, t is the total number of choices for the random bits generated by V up to round i, and a is the number of choices for which V accepts up to round i. On any input x, M computes a and t recursively as follows by calling $\Gamma(x, 1)$:

Algorithm $\Gamma(x, i, m_i, \ldots, m_{i-1})$

$(a, t) = (0, 0)$;
if $i = n^c$ **then**
  for all $r \in \{0,1\}^{n^k}$ **do**
    if $V(x, i, m_1, m_2, \ldots, m_{i-1}, r) = \text{ACCEPT}$ **then**
      $a = a + 1$;
    end if
  end for
  return $(a, 2^{n^k})$;
else
  for all $r \in \{0,1\}^{n^k}$ **do**
    $m_i = V(x, i, m_1, \ldots, m_{i-1}, r)$;
    if $m_i = \text{ACCEPT}$ **then**
      $(a, t) = (a + 1, t + 1)$;
    else if $m_i = \text{REJECT}$ **then**
      $(a, t) = (a, t + 1)$;
    else
      $m_{i+1} = P(x, i + 1, m_1, \ldots, m_i)$;
      $(a, t) = (a, t) + \Gamma(x, i + 2, m_1, \ldots, m_{i+1})$;
    end if
  end for
  return (a, t);
end if

Let $s = \frac{a}{t}$. If $s \geq 2/3$, then M accepts x; otherwise, M rejects x. This algorithm performs in polynomial space. So M decides L in polynomial space. ■