HMMT Scoring Algorithm Technical Specification

February 19, 2012

1 Overview

We start by choosing prior probability distributions on the competitors' scores and on problem difficulty. These are chosen to match some qualitative goals:

- Every competitor's score should be nonnegative.
- Competitors' scores should be able to be unbounded, though should not be infinite in the case where they solve every problem.
- Problem values should range between 2 and 10 to make the emphasis on hard problems similar to previous years.
- The distributions are smooth.

To this end, we choose the following prior probability distributions.

- For a competitor c, their score α_c is a priori distributed as $P(\alpha_c) = e^{-\alpha_c}$ over the interval $[0, \infty)$.
- For a problem p, its difficulty β_p is a priori distribued as $P(\beta_p) = e^{\frac{8}{(\beta_p-2)(10-\beta_p)}}$ over the interval $(2, 10)$.

Finally, we need a way to estimate the likelihood that a person with a given score solves a problem with a given difficulty. For this, we choose the probability function $P(\alpha_c, \beta_p) = \frac{e^{-\beta_p/\alpha_c}}{1+e^{-\beta_p/\alpha_c}}$. We further make the assumption for the purposes of this algorithm that all these events are independent. This allows us to compute the probability of a set of results given a set of scores and difficulties. We perform a Bayesian update on our priors, then use the most likely set of scores and difficulties as the result of the contest.

2 Implementation

After computing the Bayesian update, we see that we are trying to maximize the function

$$F(\vec{\alpha}, \vec{\beta}) = \prod_{\text{problems } p} e^{-\frac{8}{(10-\beta_p)(\beta_p-2)}} \prod_{\text{competitors } c} e^{-\alpha_c} \prod_{c \text{ solved } p} e^{-\beta_p/\alpha_c} \prod_{c \text{ took } p} \frac{1}{1+e^{-\beta_p/\alpha_c}}.$$

Maximizing this function is equivalent to maximizing its natural logarithm

$$\log F(\vec{\alpha}, \vec{\beta}) = \sum_{\text{problems } p} -\frac{8}{(10-\beta_p)(\beta_p-2)} + \sum_{\text{competitors } c} -\alpha_c + \sum_{c \text{ solved } p} -\beta_p/\alpha_c + \sum_{c \text{ took } p} \log \frac{1}{1+e^{-\beta_p/\alpha_c}}.$$

Notice that this function is concave, so we have a unique maximum where all the partial derivatives are 0. For any given competitor c, taking the partial derivative with respect to α_c gives us the equation

$$\alpha_c^2 + \sum_{c \text{ took } p} \beta_p \frac{e^{-\beta_p/\alpha_c}}{1+e^{-\beta_p/\alpha_c}} - \sum_{c \text{ solved } p} \beta_p = 0 \tag{1}$$

and for any given problem p, taking the partial derivative with respect to β_p gives us the equation

$$\frac{1}{(\beta_p-2)^2} - \frac{1}{(10-\beta_p)^2} + \sum_{c \text{ took } p} \frac{1}{\alpha_c} \cdot \frac{e^{-\beta_p/\alpha_c}}{1+e^{-\beta_p/\alpha_c}} - \sum_{c \text{ solved } p} \frac{1}{\alpha_c} = 0. \tag{2}$$

Notice that the equations defined by (1) only depend on the α corresponding to the particular competitor, and the equations defined by (2) only depend on the β corresponding to the particular problem. Additionally, all of these equations are monotonic in the variable that we differentiated with respect to, so we can solve them with a binary search. This means that if we have a vector of αs, we can solve for the βs, and if we have a vector of βs, we can solve for the αs. If we let $\vec{\alpha}_0$ be an arbitrary point inside the search space, then let $\vec{\beta}_n$ be the solution to the equations (2) given $\vec{\alpha}_n$ and $\vec{\alpha}_{n+1}$ be the solution to the equations (1) given $\vec{\beta}_n$. As these equations are the partial derivatives of a concave function, the sequence of pairs $(\vec{\alpha}_n, \vec{\beta}_n)$ converges to the pair $(\vec{\alpha}, \vec{\beta})$ which satisfies both (1) and (2), and thus to the maximizing set of parameters. This gives us an iterative procedure which we run until we have the desired approximation to the maximizing set of parameters.