# Randomized algorithms asking few input bits

Mátyás Susits

## Excerpt

Many Boolean functions, functions from $\{-1,1\}^n$ to $\{-1,1\}$, can be evaluated with only partial knowledge of the input. A simple example is the majority function on $n$ bits. Where if we know that more than $n/2$ bits with value 1, then the function does not depend on bits not yet examined. An algorithm $A$ for a Boolean function $f$ is an algorithm which queries the bits one by one and makes decisions of which bit to ask next based on previously queried bits. In the case of majority function the chosen algorithm makes no difference in *cost*, the expected number of bits asked by the algorithm, however this is not always the case. For many functions, like the depth $n$ three way majority, careful construction of an algorithm has tremendous effect on the cost.

Obviously we cannot make algorithms with arbitrarily low cost for a given boolean function $f$. Thus the definition of the cost of a function arises naturally: the infimum of the cost of $A$ over all algorithms $A$ for $f$. If we have a lower bound on the cost of $f$ and are able to construct an algorithm with the same cost, then we got an optimal algorithm in terms of cost. The thesis shows different lower bounds using different properties, like variance and influence and discusses nontrivial functions where for some of them these bounds are tight, and for some of them it is not know whether they are tight or not.

Constructing an algorithm with minimal cost is no easy task as there is no uniformly usable construction process discovered yet. In this thesis we examine an algorithm arising from identifying a group of games called random turn win-or-lose selection games with Boolean functions, using the optimal strategy for winning the game as an algorithm for evaluating the corresponding Boolean function. This algorithm if translated back to Boolean functions, tries to minimize part of the law of total variance, conditioned on individual bits. We will show functions like the AND-OR tree where this algorithm is indeed optimal in cost and discuss a Boolean function, the recursive tree way majority, where this method does not give such results.