

Are Search and Decision Problems Computationally Equivalent?

Richard Karp
Eli Upfal
Avi Wigderson

Abstract

From this point of view of sequential polynomial time computation, the answer to the question in the title is “yes”. The process of *self-reducibility* is a linear time Turing (oracle) reduction from a given combinatorial search problem to an appropriately defined decision problem.

However, from the point of view of fast parallel computation, the answer is not so clear. Many of the sequential algorithms that were “marked off” as being “inherently sequential” embed within them the self-reducibility process. Can this inherently sequential process be parallelized?

To study this problem, we define an abstract setting (namely that of an independence system) in which *one, universal* search problem captures all combinatorial search problems. We consider several natural decision and function oracles to which this search problem may be reduced.

On the positive side, we give efficient probabilistic parallel reductions to these oracles. These reductions constitute a scheme for parallelizing search problems in case the oracles for these problems are themselves efficiently computable in parallel. We give examples of problems that did not yield to parallelism before, but can be parallelized using this scheme.

On the negative side, we prove lower bounds on any *deterministic* parallel reductions to the same oracles. If p processors are used, the sequential (linear) running time cannot be enhanced by more than a factor of $O(\log p)$ and hence for any polynomial number of processors the problem remains inherently sequential. This proves that randomization can be exponentially more powerful than determinism in our model and suggests that $NC \neq Random\ NC$.

Finally, we state some intriguing conjectures and suggest new directions of research in complexity theory that arise from this work.

In this paper we compare the power of the two most commonly used concurrent-write models of parallel computation, the COMMON PRAM and the PRIORITY PRAM. These models differ in the way they resolve write conflicts. If several processors want to write into the same-shared memory cell at the same time, in the COMMON model they have to write the same value. In the PRIORITY model, they may attempt to write different values; the processor with smallest index succeeds.

We consider PRAM's with n processors, each having arbitrary computational power. We provide the first separation results between these two models in two extreme cases: when the size m of the shared memory is small ($m \leq n^c$, $\forall \epsilon < 1$), and when it is infinite.

In the case of small memory, the PRIORITY model can be faster than the COMMON model by a factor of $\Theta(\log n)$, and this lower bound holds even if the COMMON model is probabilistic. In the case of infinite memory, the gap between the models can be a factor of $\Omega(\log \log \log n)$.

We develop new proof techniques to obtain these results. The technique used for the second lower bound is strong enough to establish the first tight time bounds for the PRIORITY model, which is the strongest parallel computation model. We show that finding the maximum of n numbers requires $\Theta(\log \log n)$ steps, generalizing a result of Valiant for parallel computation trees.