A Geometric Parallel Search Problem Related To Group Testing

Zoltán Füredi [1]
Mathematical Institute of the Hungarian Academy of Sciences,
P. O. B. 127, Budapest 1364 Hungary
and
Department of Mathematics
University of Illinois
Urbana, IL 61801, USA

Nathan Linial [2]
Hebrew University of Jerusalem

Abstract. Given n real number whose sum is zero, find one of the numbers that is non-negative. In the model under consideration, an algorithm is allowed to compute p linear forms in each time step until it knows an answer. We prove that exactly $\lceil \log n / \log(p+1) \rceil$ time steps are required. Some connections with parallel group-testing problems are polnted out.

1 Search By Boolean Functions

Karp, Upfal and Wigderson [KUW] studied the following fundamental problem on parallel search: Every element of $N = \{1, \ldots, n\}$ is classified as either *good* or *bad* and algorithms are considered which search and find a good element. The algorithm can make a *query* which is a subset Q of N. The answer to the query is Yes or No indicating whether or not Q contains good elements. If p queries may be posed at each step, then the time complexity is $\lceil \log n / \log(p+1) \rceil$ by [KUW]. (The sequential version, $p = 1$, is trivial.) The subject was further developed by Impagliazzo and Tardos [IT] who considered more general answers, monotone boolean functions, for the queries.

2 Search By Linear Forms

What happens, asked Motwani, Naor and Naor [MNN], if a query is answered in a more informative way? The answer for a query is the number of good elements in Q, i.e., a *rank oracle*. What is the time complexity, $C(g, n, p)$, of the parallel search problem with this kind of answers? The sequential version of the problem is not difficult, ([W] and [MNN]): $C(g, n, 1) = \lceil \log_2(n + 1 - g) \rceil$. We return to the parallel version in Section 4.

---

[1] This research was done while the author visited the Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. Research supported in part by the Hungarian National Science Foundation under grant No. 1812
[2] This research was while the author visited the IBM Almaden Research Center, Stanford University

Let x be the characteristic vector of the set of good elements (i.e., $x \in R^n$ and x_i is 0 or 1 according to the ith element being bad or good.) A query $Q \subseteq N$ is replied by $\sum_{i \in Q} x_i$. In other words, we receive an inner product $\langle a, x \rangle$. In the geometric version the restriction that a and x must be 0,1 vectors is removed.

For integers $n \geq p \geq 1$ define $L(n,p)$ to be the least number of rounds taken by any algorithm to solve the following problem: the input consists of a real n-vector $x = (x_1, \ldots, x_n)$ with $\sum x_i, = 0$. We look for an algorithm which finds an index i such that $x_i \geq 0$. In each time step the algorithm presenting p vectors $a^1, \ldots, a^p$ in R^n and receiving their inner products with x, $\langle x, a^1 \rangle, \ldots, \langle x, a^p \rangle$.

The obvious algorithm which worked for the combinatorial problem applies here as well: Partition the set of coordinates into $p + 1$ roughly equal parts (all part sizes are within one from each other) and let a^i be the characteristic vector of the ith part. Note that the sum on the $(p+1)$-st part can be deduced and that at least one of the $p + 1$ partial sums is nonnegative. The algorithm then proceeds with the corresponding subvector of x.

Theorem. *For any* $n \geq p \geq 1$, $\quad L(n,p) = \lceil \frac{\log n}{\log(p+1)} \rceil$.

3 Proof Of The Lower Bound For The Time Complexity

For every algorithm, after k rounds of queries there is an affine subspace $X^{(k)}$ such that the information gathered so far by the algorithm can be summarized by saying that x belongs to $X^{(k)}$. Let $X^{(0)}$ be the set of those points in R^n whose coordinate sum is zero. Define N_i to be those points in $X^{(0)}$ whose ith coordinate is negative. If the algorithm stops after s steps by singling out the coordinate i, then $x_i \geq 0$ is equivalent to $N_i \cap X^{(s)} = \emptyset$. In the beginning, not only $N_i \cap X^{(0)} \neq \emptyset$ for all i, but the intersection of any $n-1$ of them is nonempty in $X^{(0)}$. The Lemma below implies that the intersection of any $\lfloor (n-1)/(p+1)^k \rfloor$ of the sets N_i is non-empty in $X^{(k)}$. Hence the algorithm cannot stop until $(n-1)/(p+1)^s < 1$ implying $L(n,p) = s \geq \lceil \log n / \log(p+1) \rceil$. A family of sets is *d-wise intersecting* if every d sets in the family share a point, $(d \geq 1)$.

Lemma. *Let $\{N_i\}$ be a d-wise intersecting, finite family of convex sets in the Euclidean space X, and let T be an affine transformation from X to R^p. Then for some point y in R^p, the family $\{N_i \cap T^{-1}(y)\}$ is $\lfloor d/(p+1) \rfloor$-wise intersecting.*

To apply the Lemma to the kth round of the algorithm define the affine transformation $T(x) := (\langle a_1, x \rangle, \ldots, \langle a_t, x \rangle)$ for the query $a^1, \ldots, a^p, T: X^{(k-1)} \to R^p$. Then the values of the answer are given by the coordinates of y supplied by the Lemma, $\langle a_j, x \rangle = y_j$. Finally, $X^{(k)} := T^{-1}(y) \cap X^{(k-1)}$.

Proof: For each index set I of size $\lfloor d/(p+1) \rfloor$, let N_I be the intersection of those N_i for which $i \in I$, and let $Y_I = T(N_I)$. From the hypotheses, the sets $\{N_I\}$ and hence $\{Y_I\}$ is a $(p+1)$-wise intersecting family of convex sets. By Helly's theorem, it follows that there is a point y in all of the Y_I. Then this point y satisfies the conclusion. ∎

4 On The Parallel Combinatorial Search Problem

If $g > n-p$, then clearly $C(g,n,p) = 1$. So unlike in the previous section, there is a genuine dependency on the total sum, g. In the case $g = 1$, $C(1,n,p) = (1+o(1))\log_2 n/p$ and so this is an instance where optimal speed-up is attained. In the same way we have $C(g,n,p) \leq 1 + \max\{C(g,\lceil n/2^p \rceil,p), C(g/2,\lceil n/2 \rceil,p)\}$, which implies by induction:

$$C(g,n,p) \leq \frac{\log_2 n}{p} + (1 - \frac{1}{p})\log_2 g.$$

The following upper bound follows using the Gilbert-Varshamov bound concerning linear error-correcting codes [BM, p. 84] for $p >> \log n$.

$$C(g,n,p) = O\left(\frac{g\log_2 n}{p}\right).$$

If $p = \Omega(n/\log_2 n)$, then it is possible in a single parallel step not only to find a good element, but in fact, to identify completely all the good ones.

Conjecture. *For every p and a large enough n there are values for g such that*

$$C(g,n,p) = \Omega\left(\frac{\log n}{\log(p+1)}\right).$$

The problems considered above belong to the field of group testing, see [HS]. Also see the monograph [AW], its 6th chapter is "Weighting problems and geometric problems".

Acknowledgements

The second author thanks Rajeeb Motwani, Moni Naor and Seffi Naor for conversations which helped in formulating the problem. We are also indebted to an unknown referee for simplifying the original proof.

References

[AW] R. Ahlswede and I. Wegener, *Search Problems*, Wiley, New York (1987).

[BM] I.F. Blake and R.C. Mullin, *An Introduction to Algebraic and Combinatorial Coding Theory*, Academic Press, New York (1976).

[HS] F.K. Hwang and V.T. Sós, *Nonadaptive hypergeometric group testing*, Studia Sci. Math. Hungar. **22** (1987), 257–263.

[IT] R. Impagliazzo and G. Tardos, *Decision versus search problem in super-polynomial time*, 30th FOCS, 222–227.

[KUW] R. Karp, E. Upfal and A. Wigderson, *The complexity of parallel research*, Jour. Comp. Syst. Sci. **36** (1988), 225–253.

[MNN] R. Motwani, M. Naor and J. Naor, *The probabilistic method yields determinstic parallel algorithm*, 30th FOCS; also private communications with the authors, 8–13.

[W] A. Wigderson, *Private communications.*