The hardness of speeding-up Knapsack

Sandeep Sen *
Department of Computer Science and Engineering
Indian Institute of Technology,
New Delhi 110016, India.

Abstract

We show that it is not possible to speed-up the Knapsack problem efficiently in the parallel algebraic decision tree model. More specifically, we prove that any parallel algorithm in the fixed degree algebraic decision tree model that solves the decision version of the Knapsack problem requires $\Omega(\sqrt{n})$ rounds even by using $2^{\sqrt{n}}$ processors. We extend the result to the PRAM model without bit-operations. These results are consistent with Mulmuley's [6] recent result on the separation of the strongly-polynomial class and the corresponding NC class in the arithmetic PRAM model.

Keywords lower-bounds, parallel algorithms, algebraic decision tree

1 Introduction

The primary objective of designing parallel algorithms is to obtain faster algorithms. Nonetheless, the pursuit of higher speed has to be weighted against the concerns of efficiency, namely, if we are getting our money's (processor's) worth. It has been an open theoretical problem whether all the problems in the class P can be made to run in polylogarithmic running time using a polynomial number of processors, often referred to as the $NC = P$ problem. From a practical viewpoint, linear speed-up is more desirable.

Recently Mulmuley [6] showed that not all problems that have strongly polynomial-time algorithm can attain polylogarithmic running time using a polynomial number of processors - in fact he established the following stronger result.

Theorem 1.1 (Mulmuley[6]) *The max-flow problem for graphs with k nodes cannot be solved in the PRAM model without bit operations in $\sqrt{k}/b$ time using $2^{\sqrt{k}/b}$ parallel processors for some sufficiently large $b > 0$ even when the capacities often edges are bounded by $O(k^2)$ bit integers.*

This may be viewed as a fairly strong evidence that P is not contained in NC. The proof of this result is quite non-trivial and uses some tools from algebraic geometry. In this note, we present a different proof of a slightly weaker result, namely that the Knapsack problem is difficult to parallelize in the algebraic model. However, the simplicity of our proof is notable - it exploits

*Part of the work was done when the author was visiting BRICS, University of Aarhus, Denmark in summer of 1998.

a previous general result of the author [8]. There have been related results (for example see Grigoriev [3]) for more restricted parallel models. Interestingly, in the algebraic model, Meyer [4] had described a non-uniform polynomial-time algorithm for the knapsack problem.

2 A general lower bound

The model of computation is the parallel analogue of the *algebraic decision tree model* (PAD Tree). At each node of this tree, each of the p processors compares the value of a fixed degree polynomial with 0. Accordingly each processor gets a a *sign* $\in \{0, +, -\}$ depending on the result of the comparison being $\{=, >, <\}$ respectively. Subsequently the algorithm branches according to the *sign vector*, that is by considering the signs of all the processors. The algorithm terminates when we reach a leaf node containing the final answer. If the polynomials are restricted to be of the form $x_i - x_j \leq 0$, then it is the Parallel Comparison Tree (PCT) model. While there is no cost for branching (that includes processor allocation and read-write conflicts), it does not have the full arithmetic instruction set of the PRAM model. So, strictly speaking, it is incomparable with the PRAM model.

For completeness, we rederive some of the results from Sen [8]. The *arity* of a tree is the maximum number of children at any node.

Fact 2.1 *In a parallel comparison (PAD) tree of l leaves and maximum arity a, the average path-length is at least $\Omega(\log l/\log a)$.*

Given this fact (credited to Shannon), one needs a reasonably tight upper-bound on the arity of the parallel decision tree and a lower bound on the number of leaves to establish a lower-bound of any PAD algorithm. The number of leaves is related to the number of connected components in the solution space in R^n where n is the dimension of the solution space (which is often the input size). The arity of this tree is the number of distinct outcomes of computations performed by $p > 1$ processors. For sorting, this tree has $n!$ leaves.

If the PAD algorithm uses p processors then the signs of p polynomials can be computed simultaneously. Each test yields a sign and we branch according to the *sign-vector* resulting from all the tests. We shall use the following result on the number of connected components induced by m polynomial inequalities, due to Pollack and Roy [7], who had extended a result of Warren [9] to bound the number of such sign-vectors.

Lemma 2.2 *The number of connected components of all nonempty realizations of sign conditions of m polynomials in d variables, each of degree at most b is bounded by $((O(bm/d))^d$.*

This gives us a bound on the arity of the PAD tree model as well as the number of connected components associated with a leaf node at depth h. The number of polynomials defining the space in a leaf-node at depth h is hp and hence the number of connected components associated with such a node is $((O(bhp/d))^d$. In our context, the number of processors and (hence the polynomial signs computed at each stage) is bounded by kn and d is the dimension of the solution space which is approximately the size of the input. This gives us the following theorem.

Theorem 2.3 *Let $W \subset R^n$ be a set that has $|W|$ connected components. Then any PAD tree algorithm that decides membership in W using kn ($k \geq 1$) processors has time complexity $\Omega(\log |W|/n \log k)$.*

Proof: If h is the length of the longest path in the tree then from Lemma 2.2

$$(ekn/n)^{hn} \cdot (ehkn/n)^n \geq |W|$$

where e is a constant that subsumes the degree of the polynomials. The first expression on the left hand side represents the maximum number of leaves and the second expression is the maximum number of connected components associated with a leaf at depth h. By rearranging terms we obtain

$$h \cdot \log(ek) + \log(ehk) \geq \log|W|/n.$$

Using $h\log(ek) > \log(ehk)$ for $h > 1$,

$$2h\log(ek) \geq \log|W|/n,$$

from which we arrive at the required result. □

The *algebraic computation model* is more powerful than the decision tree model as it also allows arithmetic operations. This is as powerful as the RAM model without bit-operations and indirect addressing. We can view this as a tree where some of the nodes involve arithmetic operations and the others are branching nodes. Ideally, we would like to extend our previous results to the algebraic model. The main difficulty arises from rapid growth in the degree of the polynomials involved. For example, after t rounds (depth t in the computation tree), the degree could be as high as 2^t by repeated squaring. Ben-Or tackled this problem using auxiliary variables, that is by trading degree with dimension of the underlying space. This does not work in the parallel model because a large number of variables can be introduced in every round (equal to the number of processors). Consequently, we obtain a weaker result for the parallel algebraic computation tree by setting the degree of the polynomials to 2^t at depth t and rederiving the bounds.

Theorem 2.4 *Let $W \subset R^n$ be a set that has $|W|$ connected components. Then any parallel algebraic computation tree algorithm for deciding membership in W using kn ($\log k = o(\log|W|)$) processors has time complexity $\Omega(\sqrt{\log|W|/n + \log^2 k} - \log k)$.*

Proof: Using the same notations as in the proof of Theorem 2.3 we obtain the following

$$\prod_{t=0}^{h} (2^t k)^n \cdot \sum_{j=0}^{h} (2^t k)^n \geq |W| \tag{1}$$

The first term on the L.H.S. represents the product of out-degrees for different levels up to h and the second term is the number of components associated with polynomials along a fixed path of the tree. This implies

$$\begin{aligned} k^{hn+1}[2^{h(h+1)n/2}][2^{hn+1}/2^n - 1] &\geq |W| \\ \Rightarrow k^{hn+1}[2^{(h+1)(h+2)n/2}] &\geq |W| \\ \Rightarrow (hn+1)\log k + h^2 n &\geq \log|W| \end{aligned} \tag{2}$$

using $h^2 \geq \frac{(h+1)(h+2)}{2}$. Solving the degree two equation (in h) and using $\log k \ll \log|W|$ yields the required bound. □

We note here that Grigoriev [3]) proved an an $\Omega(\sqrt{\log|W|/n})$ lower-bound for recognizing a semi-algebraic set W on a more restricted parallel model. His model has bounded in-degree that forms the crux of the lower-bound and is similar to the EREW model. Consequently, the exact nature of processor-time trade-off is not explicit in the bounds. On the other hand, for problems like convex-hulls, a matching upper-bound is shown to exist (Sen[8]) in our model that is similar to the CRCW model.

3 Application to the Knapsack problem

Our first lower bound proof is for the decision version of the Knapsack problem that has been defined in Meyer [5] as follows:

> Given an integer $s \geq 1$, and a vector $x \in R^n$, does there exist an integer vector $a \in \{0, 1 \dots s\}^n$ such that $x \cdot a = s$?

From Meyer [5] and Dobkin and Lipton [2], the solution space of the usual Knapsack problem ($s = 1$) has $2^{\Omega(n^2)}$ components. Using $|W| = 2^{\Omega(n^2)}$ and $k = 2^{\sqrt{n}}/n$ in Theorem 2.3, we obtain the following corollary.

Corollary 3.1 *The Knapsack problem in n variables requires $\Omega(\sqrt{n})$ rounds in the parallel algebraic decision tree using $2^{O(\sqrt{n})}$ processors.*

We can extend the result to the parallel computation tree (or equivalently PRAM model without bit-operations) by working with equation 2 in the proof of Theorem 2.4. Taking logarithm on both sides, we obtain

$$(hn + 1) \log k + (h + 1)(h + 2) \cdot n/2 \geq cn^2$$

for some positive constant c. Dividing both sides by n, it is clear that h is $\Omega(\sqrt{n})$ for $k < 2^{\sqrt{n}}$.

Corollary 3.2 *The Knapsack problem in n variables requires $\Omega(\sqrt{n})$ rounds in the arithmetic PRAM (without bit-operations) using $2^{O(\sqrt{n})}$ processors.*

References

[1] M. Ben-Or. Lower bounds for algebraic computation trees. *Proc. of the Fifteenth STOC*, 80–86, 1983.

[2] D. Dobkin and R.J. Lipton. A lower bound of $\frac{1}{2}n^2$ on Linear Search Programmes for the Knapsack problem. *SIAM J. on Computing*, 16 : 413–417, 1978.

[3] D. Grigoriev. Nearly sharp complexity bounds for multiprocessor algebraic computations. *Journal of Complexity*, 13 : 50–64 , 1997

[4] F. Meyer Auf Der Heide. A polynomial time linear search algorithm for the n-dimensional knapsack problem. *Journal of the ACM*, 31(3) : 668–676, 1984.

[5] F. Meyer Auf Der Heide. Lower bounds for solving linear Diophantine equations on Random Access Machines. *Journal of the ACM*, 32 : 929– 937, 1985.

[6] K. Mulmuley. Lower bounds in a parallel model without bit-operations. *to appear in SIAM Journal on Computing*, 1998.

[7] M.F. Roy and R. Pollack. On the number of cells defined by a set of polynomials. *Competes Rendus*, 316:573–577, 1992.

[8] S. Sen. Lower bounds for parallel algebraic decision trees, parallel complexity of convex hulls and related problems *Theoretical Computer Science*, 188 : 59 – 78 , 1997.

[9] H.E. Warren. Lower bounds for approximation of non-linear manifolds. *Tran. of Amer. Math. Soc.*, 133:167 – 178, 1968.