STOC(Milwaukee 1981),123-127.

On the Parallel Computation for the Knapsack Problem*

Andrew Chi-Chih Yao
Computer Science Department
Stanford University
Stanford, California 94305

Abstract. We are interested in the complexity of solving the knapsack problem with n input real numbers on a parallel computer with real arithmetic and branching operations. A processor-time tradeoff constraint is derived; in particular, it is shown that an exponential number of processors have to be used if the problem is to be solved in time $t \leq \sqrt{n}/2$.

Keywords: Branching operations, complexity, knapsack, parallel computation.

1. Introduction.

Given n real numbers $x_1, x_2, \ldots, x_n$, the *knapsack problem* is to determine if there exists a subset $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} x_i = 1$. We are interested in the complexity of solving the knapsack problem on a parallel computer with real arithmetic and branching operations (but without ceiling and floor functions). A constraint on the time-processor tradefoff will be derived. In particular, it implies that $p \geq \frac{1}{4} 2^{\sqrt{n}/2}$ processors have to be used if the problem is to be solved in time $t \leq \sqrt{n}/2 - 1$. This seems to be a rare case, where a natural problem is shown to be not solvable in simultaneous $O((\log n)^k)$ time and $O(n^k)$ processors for any k, without being shown to require exponential time for the sequential computation ($p = 1$).

* This research was supported in part by National Science Foundation under grant MCS-77-05313-A01.

In the literature, the sequential complexity of the knapsack problem has been considered in a somewhat different model (Dobkin and Lipton [3], Steele and Yao [8]). In Section 4, an extension of the time-processor tradeoff constraint to a parallel version of that model will be considered. (We recommend the readers to [3] [8] for background reading, where many concepts used in this paper were originaly introduced.)

Hardware size and parallel time required for computations in a variety of models have received much attention recently (see, e.g., Borodin [2], Dymond and Cook [4], Hong Jai-Wei [5]). Our model differs from them in that the inputs x_i are treated as real numbers and not as finite bit strings. Thus, our conclusions do not translate into results in those models, in which the individual bits in the inputs can be manipulated. However, our model is quite proper for the knapsack problem, and other problems such as network flows, finding shortest paths, etc., as many algorithms for these problems treat x_i just as real numbers (see e.g. [1] [6]).

2. The Parallel Arithmetic Model.

Let $W \subseteq R^n$ be any set. The *membership problem* for W is the following: Given $\vec{x} = (x_1, x_2, \ldots, x_n) \in R^n$, determine if $\vec{x} \in W$. As in [3], we regard the n-input knapsack as a membership problem with $W = W^{(n)}$, where $W^{(n)}$ is defined to be

$$W^{(n)} = \left\{ \vec{x} \;\middle|\; \prod_{S \subseteq \{1,2,\ldots,n\}} \left(\sum_{i \in S} x_i - 1 \right) \neq 0 \right\}. \quad (1)$$

We will be interested in solving the knapsack problem on a parallel computer with p processors. Each processor can perform an arithmetic operation and test for the sign (less than, equal to, or greater than 0) of

the resulting value. The selection of the next (parallel) step may depend on the results of all the previous tests (thus, a priori, there are 3^p possible branchings at each step). Formally, a *parallel program with p processors* T is a multi-way tree, with each internal node v containing a set of p rational functions

$$\mathcal{F}_v = \{f_{v1}(\vec{x}), f_{v2}(\vec{x}), \ldots, f_{vp}(\vec{x})\},$$

and with each leaf ℓ containing ϵ_ℓ, a "yes" or "no" answer. Let

$$\mathcal{G}_v = \{1, x_1, x_2, \ldots, x_n\} \bigcup \left(\bigcup_{v' < v} \mathcal{F}_{v'} \right),$$

where $v' < v$ denotes that v' are ancestors of v. Each $f_{vi}(\vec{x})$ is either of the form $g(\vec{x}) \circ h(\vec{x})$ or of the form $c \circ h(\vec{x})$, where $\circ \in \{+, -, *, /\}$, $g, h \in \mathcal{G}_v$, and c is any constant. Each branch leaving v is labeled by a distinct string $\tilde{b} = b_1\, b_2 \ldots b_p \in \{1, 0, -1\}^p$. (Hence there are at most 3^p branchings at any node v; there may be fewer.) Given an input $\vec{x} \in R^n$, the program traverses a path $P(\vec{x})$ in the tree T from the root down. At each internal node v, the comparisons $\{f_{vi}(\vec{x})\colon 0\}$ are made and the branch labeled $\tilde{b}$ (where $b_i = 1, 0, -1$ according to whether $f_{vi}(\vec{x})\colon 0$ is $<, =, >$) chosen. When a leaf ℓ is reached, the answer ϵ_ℓ is returned. We say "$\vec{x}$ passes through v" if v is a node on the path $P(\vec{x})$. We require that, if an input $\vec{x}$ passes through a node v and f_{vi} is g/h or c/h, then the value $h(\vec{x}) \neq 0$.

We say that a parallel program T solves the membership problem for W, if the answer returned is correct for every input $\vec{x} \in R^n$. Let $Cost(\vec{x}, T)$ denote the number of internal nodes that $\vec{x}$ passes through. The *running time* t of T is given by the maximum of $Cost(\vec{x}, T)$ for any $\vec{x}$.

For any open set W, let $\#W$ denote the number of connected components W has. It was proved in Dobkin and Lipton [3] that, for the knapsack problem,

$$\#W^{(n)} \geq 2^{n^2/2}. \tag{2}$$

This last inequality was used in [3] [8] to derive lower bounds on the knapsack complexity, and will again be crucial in this paper.

We will prove the following results:

Theorem 1. *Let $W \subseteq R^n$ be an open set, and T be a parallel program with p processors and running time t that solves the membership problem for W. Then*

$$p \geq (\#W)^{1/((t+1)n)} 2^{-(t+3)}.$$

Theorem 2. *Let T be a parallel program with p processors and running time t that solves the n-input knapsack problem. Then*

$$p \geq 2^{n/(2(t+1))-(t+3)}.$$

Corollary. *If a parallel program solves the knapsack problem in time $t \leq \frac{1}{2}\sqrt{n} - 1$, then the number of processors p is at least $2^{\frac{1}{2}\sqrt{n}-2}$.*

Theorem 2 and its corollary follows from Theorem 1 and inequality (2) easily. Thus, we need only to prove Theorem 1.

3. Proof of Theorem 1.

We need some preliminary definitions. For any polynomial $q(x_1, x_2, \ldots, x_n)$, let $S_q = \{\vec{x} \mid q(\vec{x}) \neq 0\}$. For any integer $m, n > 0$, let

$$\beta(m, n) = \max\{\#S_q \mid q \text{ is a polynomial in } n \text{ variables and of degree at most } m\}.$$

It follows from a fairly deep result of J. Milnor [7] that

$$\beta(m, n) \leq (m+2)(m+1)^{n-1}. \tag{3}$$

For more discussions of inequality (3), see reference [8, Section 3].

We now begin the proof. Without loss of generality, we assume that no branching in T is redundant, i.e., there is at least one input $\vec{x}$ taking each branching. This implies that each leaf can be reached by some input $\vec{x}$. The running time t is now the same as the height of the tree T. The *depth* of a node v, denoted by $depth(v)$, is its distance from the root; the root has depth 0.

We can also assume that no test function $f_{vi}(\vec{x})$ is identically 0, because we can replace such a function by any non-zero constant function (say 1) and then relabel the branchings.

For each node v, let V_v be the set of inputs $\vec{x}$ that pass through v.

Lemma 1. *Let v be an internal node of depth j, and let $f \in \mathcal{F}_v$. Then there exist polynomials $a(\vec{x})$, $b(\vec{x})$ each of degree at most 2^{j+1} such that, for all $\vec{x} \in V_v$, we have $d(\vec{x}) \neq 0$ and $f(\vec{x}) = a(\vec{x})/d(\vec{x})$.*

Proof. We prove by induction on j. The lemma is true for $j = 0$ by inspection. Suppose the lemma is

true for all $j < \ell$ ($\ell > 0$), we will prove it for $j = \ell$. There are two possibiities:

Case 1. $f(\vec{x}) = g(\vec{x}) \circ h(\vec{x})$, where

$$g(\vec{x}) \in \{1, x_1, \ldots, x_n\} \bigcup \mathcal{F}_{v'}$$

and

$$h(\vec{x}) \in \{1, x_1, \ldots, x_n\} \bigcup \mathcal{F}_{v''}$$

for some $v' < v$, $v'' < v$.

By the induction hypothesis, we can write $g(\vec{x}) = g_1(\vec{x})/g_2(\vec{x})$ for $\vec{x} \in V_{v'}$ and $h(\vec{x}) = h_1(\vec{x})/h_2(\vec{x})$ for $\vec{x} \in V_{v''}$, such that $deg(g_i) \leq 1^{1+depth(v')} \leq 2^j$ and $deg(h_i) \leq 2^{1+depth(v'')} \leq 2^j$ for $i = 1, 2$, and such that $g_2(\vec{x}) \not\equiv 0$ for $\vec{x} \in V_{v'}$ and $h_2(\vec{x}) \not\equiv 0$ for $\vec{x} \in V_{v''}$. In particular, this implies $g_2(\vec{x}) \not\equiv 0$, $h_2(\vec{x}) \not\equiv 0$ for all $\vec{x} \in V_v$.

If $\circ = \pm$, then $f(\vec{x}) = a(\vec{x})/d(\vec{x})$ for $\vec{x} \in V_v$, where

$$a(\vec{x}) = g_1(\vec{x})h_2(\vec{x}) \pm g_2(\vec{x})h_1(\vec{x}),$$

$$d(\vec{x}) = g_2(\vec{x})h_2(\vec{x}).$$

Clearly, $deg(a(\vec{x})), deg(d(\vec{x})) \leq 2^{j+1}$ and $d(\vec{x}) \not\equiv 0$ for all $\vec{x} \in V_v$.

If $\circ = *$, then we can take $a(\vec{x}) = g_1(\vec{x})h_1(\vec{x})$ and $d(\vec{x}) = g_2(\vec{x})h_2(\vec{x})$. Then $f(\vec{x}) = a(\vec{x})/d(\vec{x})$ for $\vec{x} \in V_v$, and all conditions are satisfied.

If $\circ = /$, then by assumption $h(\vec{x}) \not\equiv 0$ for $\vec{x} \in V_{v''}$, and hence $h_1(\vec{x}) \not\equiv 0$ for $\vec{x} \in V_v$. One can then easily verify that $a(\vec{x}) = g_1(\vec{x})h_2(\vec{x})$, $d(\vec{x}) = g_2(\vec{x})h_1(\vec{x})$ satisfy the conditions.

Case 2. $f(\vec{x}) = c \circ h(\vec{x})$ where c is a constant and $h(\vec{x}) \in \{1, x_1, \ldots, x_n\} \bigcup \mathcal{F}_{v'}$ for some $v' < v$.

A similar (and simpler) argument as given above takes care of this case. We omit the details.

This completes the inductive proof. ∎

Let

$$\Delta = \{\tilde{b} \mid \tilde{b} = b_1\, b_2 \ldots b_p \text{ with } b_i \in \{1, -1\} \text{ for all } i\}.$$

Lemma 2. *Let v be any internal node, and B_v be the number of branches from v that have labels $\tilde{b} \in \Delta$. Then $B_v \leq \beta(2^{j+2}p, n)$ where $j = depth(v)$.*

Proof. Let $\mathcal{F}_v = \{f_1, f_2, \ldots, f_p\}$. By Lemma 1, we can choose polynomials a_i, d_i for $1 \leq i \leq p$ such that $deg(a_i) \leq 2^{j+1}$, $deg(d_i) \leq 2^{j+1}$ and $f(\vec{x}) = a_i(\vec{x})/d_i(\vec{x})$, $d_i(\vec{x}) \not\equiv 0$ for all $\vec{x} \in V_v$.

For each $\tilde{b} \in \Delta$, let $X_{\tilde{b}} = \{\vec{x} \mid d_i(\vec{x}) \not\equiv 0, b_i f_i(\vec{x}) < 0 \text{ for all } i\}$. Then B_v is no greater than the number of $\tilde{b} \in \Delta$ with $X_{\tilde{b}} \not\equiv \emptyset$. Now, we can write $X_{\tilde{b}} = \{\vec{x} \mid b_i a_i(\vec{x})d_i(\vec{x}) < 0 \text{ for all } i\}$. Note that $X_{\tilde{b}}$ is an open set. Consider the polynomial $q(\vec{x}) = \prod_{1 \leq i \leq p}(a_i(\vec{x})d_i(\vec{x}))$. Then $S_q = \bigcup_{\tilde{b} \in \Delta} X_{\tilde{b}}$. Since all $X_{\tilde{b}}$ are disjoint open sets, we have $B_v \leq$ (the number of $\tilde{b} \in \Delta$ with $X_{\tilde{b}} \not\equiv \emptyset$) $\leq \#S_q \leq \beta(p2^{j+2}, n)$. ∎

Let us write $W = \bigcup_{1 \leq i \leq \#W} W_i$, the disjoint union of nonempty open sets. Let $\mathcal{L}$ be the set of leaves of T with a "yes" answer and which can be reached from the root using only "inequality" branches (i.e., branches labeled by $\tilde{b} \in \Delta$). Clearly, $\bigcup_{\ell \in \mathcal{L}} V_\ell \subseteq W$.

Lemma 3. *The set $\bigcup_{\ell \in \mathcal{L}} V_\ell$ intersects all the open sets W_i.*

Proof. The set $W - \bigcup_{\ell \in \mathcal{L}} V_\ell$ is contained in the union of $V_{\ell'}$ where ℓ' are leaves that satisfy at least one equality constraint along the path from the root. Such $V_{\ell'}$ have measure 0 and so does $W - \bigcup_{\ell \in \mathcal{L}} V_\ell$. As each W_i is of non-zero measure, the lemma follows. ∎

Lemma 4. *For each $\ell \in \mathcal{L}$, V_ℓ intersects at most*

$$\beta(p2^{t+2}, n)$$

of the W_i.

Proof. Let $root = v_0, v_1, v_2, \ldots, v_s = \ell$ be the sequence of nodes from the root to ℓ ($s \leq t$). Consider $\mathcal{F}_{v_j} = \{f_{j1}, f_{j2}, \ldots, f_{jp}\}$. By Lemma 1, we can write $f_{ji}(\vec{x}) = a_{ji}(\vec{x})/d_{ji}(\vec{x})$ with $d_{ji}(\vec{x}) \not\equiv 0$ for all $\vec{x} \in V_{v_j}$, and $deg(a_{ji}) \leq 2^{j+1}$, $deg(d_{ji}(\vec{x})) \leq 2^{j+1}$.

Let $\tilde{b}^{(j)} = b_1^{(j)}\, b_2^{(j)} \ldots b_p^{(j)}$ be the label on the branch from v_j to v_{j+1}. Then

$$V_\ell = \{\vec{x} \mid b_i^{(j)} a_{ji}(\vec{x})d_{ji}(\vec{x}) < 0 \text{ for } 0 \leq j < s, 1 \leq i \leq p\}.$$

Clearly the open set V_ℓ is the union of some components of the set S_q, where $q(\vec{x}) = \prod_{i,j}(a_{ji}(\vec{x})d_{ji}(\vec{x}))$. Thus, $\#V_\ell \leq \beta(m, n)$, where

$$m = \sum_{0 \leq j < s} (p2^{j+2}) = p(2^{s+2} - 4) < p2^{t+2}.$$

Since $V_\ell \subseteq W$, each component of V_ℓ is completely contained in some component of W. Thus V_ℓ can intersect at most $\#V_\ell \leq \beta(m,n)$ components of W. The lemma then follows from the monotonicity of $\beta(m,n)$ in m. ∎

We now complete the proof of Theorem 1. By Lemmas 3 and 4, we have

$$|\mathcal{L}|\beta(p2^{t+2},n) \geq \#W.$$

But from Lemma 2, $|\mathcal{L}| \leq (\beta(p2^{t+2},n))^t$. Hence,

$$(\beta(p2^{t+2},n))^{t+1} \geq \#W.$$

Applying (3), we obtain

$$(p2^{t+2}+2)^n \geq (\#W)^{1/(t+1)}.$$

The theorem follows. ∎

4. Parallel Algebraic Decision Trees.

Dobkin and Lipton [3] considered the decision-tree complexity of the knapsack problem for sequential computation, and showed that $\Omega(n^2)$ tests are needed if only "linear" tests $\sum_i \lambda_i x_i$: c are employed. Steele and Yao [8] extended this $\Omega(n^2)$ lower bound to *d-th order algebraic decision trees* for any fixed d, where tests $f(x_1,x_2,\ldots,x_n)$: 0 are allowed with f being any polynomial of degree $\leq d$.

A natural extension is to consider *parallel d-th order algebraic trees with p processors*, which are multiway trees with each internal node storing p d-th (or less) degree polynomial tests $f_1(\vec{x})$: 0, $f_2(\vec{x})$: 0, $\ldots$, $f_p(\vec{x})$: 0, and with its branches corresponding to the different combinations of outcomes for the tests. The *running time* t is again the maximum number of internal nodes any input $\vec{x}$ can encounter. This model differs from the "arithmetic" model considered earlier, in that the latter model counts the arithmetic cost in computing the test functions and can build up high order test functions. However, using the same basic approach (and technically simpler), one can prove the following results.

Theorem 3. *Let $W \subseteq R^n$ be an open set, and T a parallel d-th order algebraic tree that solves the membership problem for W. Then the number of processors p and the running time t must satisfy*

$$p \geq \frac{1}{2(d+2)}(\#W)^{1/(n(1+t))}.$$

Corollary. *For the knapsack problem, $p = \Omega(\frac{1}{d}\lambda^{n/t})$ where $\lambda = \sqrt[4]{2}$.*

Corollary. *In the knapsack problem, one cannot solve the problem in time $o\left(\frac{n}{\log n}\right)$ and with a polynomial number of processors, for fixed d.*

Proof. We will only sketch the proof. We can assume that all leaves of T can be reached and that, in every test $f(\vec{x})$: 0, f is not the identically 0 polynomial.

Consider the set $\mathcal{L}$ of leaves ℓ with "yes" answer and with no test with equality results along the path from the root to ℓ. Consider the part of the tree T that consists of the paths (and the nodes on them) from the root to leaves in $\mathcal{L}$. Noting that at most $\beta(pd,n)$ branchings can occur at each node, we have

$$|\mathcal{L}| \leq (\beta(pd,n))^t. \tag{4}$$

Let V_ℓ be the set of inputs $\vec{x}$ leading to leaf ℓ. Then each V_ℓ for $\ell \in \mathcal{L}$ can be shown to intersect at most $\beta(ptd,n)$ of the components of W. This leads to

$$\#W \leq |\mathcal{L}| \cdot \beta(ptd,n). \tag{5}$$

Using (2), (4), we obtain from (5)

$$\begin{aligned}\#W &\leq (pd+2)^{nt}(ptd+2)^n \\ &\leq (p(d+2))^{n(1+t)}t^n.\end{aligned}$$

The theorem follows by noting $t^{-(1/(1+t))} \geq 1/2$ for $t \geq 1$.

The corollaries follow from the theorem and inequality (2). ∎

5. Remarks.

In this paper we have shown that, in some real-arithmetic models, the knapsack problem is hard to solve fast in parallel. However, this does not imply the same behavior in the bit-oriented models [2] [4] [5], which are commonly employed in complexity theory. The following example illustrates the point:

Given $x_1,x_2,\ldots,x_n$, determine if $\sum_{1\leq i\leq n} x_i = j$ for some integer $1 \leq j < 2^{n^2}$. As $W = \{\vec{x} \mid \prod_{1\leq j<2^{n^2}}(\sum_{1\leq i\leq n} x_i - j) \neq 0\}$ satisfies $\#W = 2^{n^2}$, we obtain the same type of time-processor tradeoff constraint as in the knapsack problem. However, circuits of small depth and size can obviously be built for this problem with finite-precision inputs x_i. It is interesting

to note that this problem becomes trivial even in real arithmetic models, if we add the floor function $\lfloor y \rfloor$ to the models. (Thus, Theorems 1 and 3 cannot be true if the floor function is allowed.)

The above example also gives in our model a problem whose sequenial complexity is $O(n^2)$, but a speedup to time $\leq \sqrt{n}$ would call for an exponential number of procesors. In a way this can be considered as an extension of the problem of locating an item in an ordered table to m items. With p parallel probes at a step, it takes $\approx (\log m)/(\log p)$ steps. Set $m = 2^n$ and try to reduce the number of steps from n to $\sqrt{n}$, we will find that we need $p \approx 2^{\sqrt{n}}$ processors.

To conclude this paper, we remark that a major open problem on this subject is to determine the sequential complexity of the knapsack problem in real arithmetic models. Even a determination of the complexity with only linear tests would be of great interest.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Massachusetts, 1974.

[2] A. Borodin, "On Relating Time and Space to Size and Depth," *SIAM J. on Computing* 6 (1977), 733–744.

[3] D. Dobkin and R. J. Lipton, "A Lower Bound of $\frac{1}{2}n^2$ on Linear Search Programs for the Knapsack Problem," *J. Comput. System Sci.* 16 (1978), 413–417.

[4] P. W. Dymond and S. A. Cook, "Hardware Complexity and Parallel Computation," *Proc. 21-st IEEE Annual Symp. on Foundations of Computer Science*, Syracuse, New York, Oct. 1980, 360–372.

[5] Hong Jai-Wei, "On Similarity and Duality of Computation," *Proc. 21-st IEEE Annual Symp. on Foundations of Computer Science*, Syracuse, New York, Oct. 1980, 348–359.

[6] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[7] J. Milnor, "On the Betti Numbers of Real Varieties," *Amer. Math. Soc.* 15 (1964), 275–280.

[8] J. M. Steele and A. C. Yao, "Lower Bounds for Algebraic Decision Trees," Stanford Computer Science Department Report STAN-CS-80-810, July 1980, submitted to the *Journal of Algorithms*.