THE COMPLEXITY OF PARALLEL COMPUTATION ON MATROIDS

Richard M. Karp†

Computer Science Division
University of California at Berkeley

Eli Upfal‡

IBM Research Laboratory.
San Jose

Avi Wigderson††

IBM Research Laboratory
San Jose

ABSTRACT

In [KUW1] we have proposed the setting of independence systems to study the relation between the computational complexity of search and decision problems. The universal problem that captures this relation, which we termed the $S-search$ problem, is: "Given an oracle for the input system, find a maximal independent subset in it".

Many interesting and important search problems can be described by a special class of independence systems, called *matroids*. This paper is devoted to the complexity of the S-search problem for matroids.

Our main result is a lower bound on any *probabilistic* algorithm for the S-search problem that acquires information about the input system by interrogating an independence oracle. We prove that the expected time of any such probabilistic algorithm that uses a $sub-exponential$ number of processors is $\Omega(n^{\frac{1}{3}-\epsilon})$. This is one of the first nontrivial, super-logarithmic lower bounds on a randomized parallel computation. It implies that in our model of computation Random-NC is strictly contained in P. Another consequence of the lower bound is that the $O(\sqrt{n})$ time probabilistic upper bound for arbitrary independence systems, presented in [KUW1], is close to optimal and cannot be significantly improved, even for matroids.

However, this $O(\sqrt{n})$ upper bound can be improved in a different sense for matroids - it can be made deterministic, still with polynomially many processors.

Finally, we show that the lower bound can be beaten for the special case of graphic matroids. Here, the S-search problem is simply to find a spanning forest of a graph, when the algorithm cannot see the graph, but can only ask whether subsets of edges are forests or not. We give an $O(\log n)$ time deterministic parallel algorithm that uses $n^{O(\log n)}$ processors.

From the upper bounds on parallel time above we deduce similar bounds (up to a polylog factor) on the sequential space required by a deterministic Turing machine with an independence oracle to solve the S-search problem.

† Research supported by NSF Grant #DCR-8411954.

‡ Part of this work was done while the author was visiting Stanford University supported by a Weizmann Post-Doctoral fellowship, and by DARPA Grant N00039-83-C-1036.

†† Current address: Mathematical Research Institute, 1000 Centennial dr. Berkeley, CA 94720.

0272-5428/85/0000/0541$01.00 © 1985 IEEE

1. INTRODUCTION

1.1. PRELIMINARIES AND PREVIOUS RESULTS

In [KUW1] we began an investigation of parallel algorithms within the context of independent systems. This abstract setting enables us to unify a number of constructions that had previously been investigated separately, in connection with parallel algorithms for problems such as computing the rank of a matrix [BGH], finding a maximal independent set of vertices in a graph [KW, Lu], and finding a maximum matching in a graph [KUW2]. The setting of independence systems also provides an abstraction of the parallel self-reducibility process, in which one solves a search problem in parallel by making calls on an oracle for an associated decision problem.

An *independence system* S is defined by a pair (E,I). E is a finite set of n elements, called the *ground set*. I is a family of subsets of E, called the *independent sets* of S. The independent sets are closed under containment, so if $A \subset B \subset E$ and $B \in I$, then also $A \in I$. The sets in $2^E - I$ are called *dependent*. A set $B \subset E$ is a *maximal independent* set of S if B is independent, but every superset of B is dependent. A system is *uniform* if all its maximal independent sets have the same cardinality.

Many combinatorial search problems are naturally captured by the following, canonical search problem, whose complexity is the main issue of both [KUW1] and this paper.

S-search: Given an independence system $S=(E,I)$, find a maximal independent set of S.

In studying the complexity of this problem, we shall assume that the input is given by an oracle to one of the following two set functions on S.

Independence: $IND_S: 2^E \rightarrow \{TRUE, FALSE\}$ is defined by $IND_S(A)=TRUE$ iff $A \in I$. (IND_S simply decides whether the set A is independent in S or not).

Rank: $RANK_S: 2^E \rightarrow \{0,1,\ldots,n\}$ is defined by $RANK_S(A)=MAX\{|B|: B \subseteq A,\ B \in I\}$.
(Rank computes the *size* of the largest independent subset of A. Note that $IND_S(A)=TRUE$ iff $RANK_S(A)=|A|$)

The time complexity T of S-search will be measured in terms of n, the size of the ground set E, and p, the parallelism allowed in the algorithm. By parallelism p we mean that p queries to the input oracle are allowed per step (a precise definition of the model is give later). We shall study both probabilistic and deterministic algorithms. For probabilistic algorithms $T^{ind}_{prob}(n,p)$ will denote the expected time to solve S-search on a worst case input, given by an independence oracle. $T^{rank}_{prob}(n,p)$ is similarly defined. For deterministic algorithms we use $T^{ind}_{det}(n,p)$ and $T^{rank}_{det}(n,p)$ for the worst case complexity.

Observe that for sequential computation ($p=1$) the S-search problem is trivial. The self-reducibility (greedy) algorithm gives $T^{ind}_{det}(n,1) = T^{rank}_{det}(n,1)=n$. What speed-up is possible with parallelism p?

To put our new results in perspective, we summarize the results of [KUW1]. There we gave probabilistic upper bounds and deterministic lower bounds on the complexity of S-search for arbitrary independence systems.

Probabilistic Upper Bounds:

(U1) $T^{ind}_{prob}(n,n) = O(\sqrt{n})$

(U2) For a uniform system
$T^{rank}_{prob}(n,n) = O((\log n)^2)$

Deterministic Lower Bounds:

(L1) $T^{ind}_{det}(n,p) = \Omega(\frac{n}{\log p})$

(L2) $T^{rank}_{det}(n,p) = \Omega(\frac{n}{\log np})$

(A harder problem than S-search, namely to find a lexicographically first maximal independent set, was shown by Martel [Ma] to require time $\Omega(\frac{n}{\log p})$ by any probabilistic algorithm when the input is given by an independence oracle).

Three main questions remained open:
1. What is the inherent limitation of the independence oracle? Can (U1) be improved, or is there a matching probabilistic lower bound?
2. Is there a lower bound that separates between the power of the independence and the rank oracles?
3. Are there interesting classes of independence systems for which the deterministic lower bounds do not hold?

Matroids, the most well-studied subclass of independence systems, provides us with interesting answers to these questions.

1.2. MATROIDS - NEW RESULTS

A *matroid* is an independence system $M=(E,I)$ in which the following condition is satisfied:
Borrowing Axiom: if $A\in I$, $B\in I$ *and* $|A|=|B|+1$, *then there exists* $e\in A$ *such that* $B\cup\{e\}\in I$.

Some very important combinatorial problems have a matroid structure [La]. For example:

Graphic Matroids: E is the set of edges in a graph G, and a subset $A\subset E$ is independent iff it is a forest in G.

Linear Matroids: E is a set of vectors in a vector space over some field, and a subset $A\subset E$ is independent iff the vectors in A are linearly independent.

Matching Matroids: E is a set of vertices of a graph G, and a subset $A\subset E$ is independent iff some matching in G covers all vertices in A.

A maximal independent set in a matroid is called a *base*. A minimal dependent set is called a *circuit*. The S-search problem for matroids is to find a base of the input matroid.

The borrowing axiom which matroids satisfy adds a lot of structure. For example, a matroid is always a uniform system, i.e., all its bases have the same size. A striking example of the algorithmic usefulness of the additional structure is the following easy result.

THEOREM 1: *For matroids,*
$$T_{\det}^{rank}(n,n)=O(1).$$

So the lower bound (L2) does not hold for matroids, and even the probabilistic upper bound (U2) can be improved. In fact, theorem 1 says that for matroids the rank oracle is too strong to be interesting.

One might expect a similar improvement for computation with an independence oracle. However, we can prove

THEOREM 2: *For matroids,*
$$T_{prob}^{ind}(n,p)=\Omega((\frac{n}{\log np})^{\frac{1}{3}}).$$

This is the main result of our paper. It says that even if p is exponential in n, say 2^{n^ϵ}, any probabilistic algorithm will require expected time $\Omega(n^{\frac{1}{3}-\epsilon})$ to solve the S-search problem for some matroid. Theorem 2 presents one of the very few non-trivial, super-logarithmic, probabilistic lower bounds for parallel computation known today. In particular, it shows that polylogarithmic time is not achievable with polynomially many processors and randomness, and so the theorem supplies a natural model of computation with an oracle, in which a problem that is solvable in P is not solvable in Random-NC.

The lower bound implies that the probabilistic upper bound (U1) cannot be significantly improved, even for matroids. It seems at this point that matroids are not easier than arbitrary independence systems, if the

input is an independence oracle. However, the upper bound (U1) can be improved in a different sense - it can be made deterministic!

THEOREM 3: *For matroids,*
$T^{ind}_{det}(n,n)=O(\sqrt{n})$.

Theorem 3 shows that that the deterministic lower bound (L1) does not hold for matroids. The deterministic upper bound of theorem 3 almost matches the probabilistic lower bound of theorem 2. Are there interesting matroids for which the lower bound of theorem 2 can be beaten? The answer is yes, graphic matroids.

THEOREM 4: *For graphic matroids* $T^{ind}_{det}(n,n^{O(\log n)})=O(\log n)$

For graphic matroids, the S-search problem is to find a spanning forest in a graph. There are simple NC algorithms for this problem. To motivate the difficulties in proving theorem 4 note that the only way the algorithm can learn about the input graph is by independence queries, which tell it if subsets of edges are forests or not. It is easy to see that there is no way to determine the vertices of the graph in this way, and hence impossible to carry out a "transitive closure" type operation, which all known fast parallel algorithms use. Still, the fact that there exists an underlying graph describing the matroid is a source of useful properties. In proving the theorem, we first devise a randomized algorithm for the problem with the same complexity. Then, generalizing techniques of [ΛCGS] and [Lu], we use the idea of "k-wise independent random variables" to eliminate randomness from our algorithm.

The relationship between parallel time and sequential space is well established (e.g Goldschlager [Go], Borodin [Bo]). In [KUW1] we extended one direction of this simulation to machines with oracles (in which the oracle tape is a write-only tape), and thus obtained lower bounds on the sequential space required to solve S-search, from our deterministic lower bounds on parallel time. In this paper we prove that the other direction of this simulation is also valid for machines with oracles, and can thus obtain upper bounds on sequential space from our deterministic upper bounds on parallel time, theorems 3 and 4. These analogous theorems are

THEOREM 3': *There is a Turing machine that given an independence oracle for a matroid, finds a base of this matroid using only* $O(\sqrt{n}\log n)$ *space.*

THEOREM 4': *There is a Turing machine that given an independence oracle for a graphic matroid, finds a base of this matroid in space* $O((\log n)^3)$

2. A PROBABILISTIC LOWER BOUND FOR GENERAL MATROIDS

Our model of computation is a parallel decision tree with an input oracle. A *probabilistic parallel decision tree of parallelism p and oracle f* is a tree with three types of nodes:

1. Randomization nodes. Every such internal node has some number b of branches, each is taken with probability $\frac{1}{b}$.
2. Oracle query nodes. Every such internal node is labeled with p subsets $B_1,\ldots,B_p$ of E. The branches from the node are labeled with all possible oracle answers $f(B_1),\ldots,f(B_p)$.
3. Leaves. Every leaf is labeled with one subset of E.

Assume without loss of generality that all matroids on n elements have the same ground set E. A tree H, that uses the independence oracle IND, is said to solve the problem of constructing a base for every matroid on n elements if for any such input matroid M and for any root-leaf path the tree may take on this input, if the leaf is labeled with the set B, then $|B|$ is equal to the rank of M, and the edge leading to the leaf is labeled with the oracle answer $IND(B) = TRUE$.

For a given tree H, let $c(H,M)$, the cost of M, be the expected (with respect to the randomization nodes) number of oracle query nodes (steps) on a root-leaf path that M may take. Let $c(H,n)$ be the maximum value of $c(H,M)$ over all input matroids M with n elements. Finally, the expected time to construct a base for a matroid of size n with the independence oracle and parallelism p, denoted by $T^{ind}_{prob}(n,p)$, is the minimum of $c(H,n)$ over all trees H of parallelism p that solve the problem.

THEOREM 2: $T^{ind}_{prob}(n,p) = \Omega[(\frac{n}{\log np})^{\frac{1}{3}}]$.

PROOF: Although the result can be proved directly by combinatorial analysis, we find it more transparent to present the proof through the following reduction, (which in our case simply converts a combinatorial proof to a probabilistic one).

Proposition 2.1: Yao [Ya]. *Let T_1 be the expected running time for a given probabilistic algorithm solving problem P, maximized over all possible inputs. Let T_2 be the average running time for a given input distribution, minimized over all possible deterministic algorithms to solve P. Then $T_1 \geq T_2$.*

A deterministic algorithm is modeled in our formalization by a *deterministic parallel decision tree* which is a special case of a probabilistic one, in which there are no randomization nodes. The deterministic computation time $T^{ind}_{det}(n,p)$ is similarly defined. The average running time for a given input distribution D is $AT^{ind}_{det}(n,p,D) = \min_H \mathrm{E}[c(H,n)]$, where the average is taken over the distribution D. To prove the lower bound we have to present an input distribution D such that

$$AT^{ind}_{det}(n,p,D) = \Omega[(\frac{n}{\log np})^{\frac{1}{3}}].$$

For our proof we consider only partition matroids of the following form: The n ground elements are partitioned into $2t$ equal size sets, $A_1,\ldots,A_{2t}$, $|A_i| = \frac{n}{2t}$. A set B is independent iff $|B \cap A_i| \leq i\frac{n}{4t^2}$, for all $i=1,\ldots,2t$. The input probability space D consists of all such partition matroids that are obtained from all possible partitions of n labeled elements into $2t$ sets of equal size. All the matroids in this probability space have equal probability.

A computation path in a decision tree H defines a sequence of events $O_1,O_2,\cdots$ where O_i is the event defined by the oracle answers at the first $i-1$ steps of the algorithm. Without loss of generality, we can assume that the algorithm is given more information through its execution. Namely, that at the end of the j^{th} step, the content of the set A_j is revealed to the algorithm. It is straightforward to extend the tree H to a decision tree H' that models this computation. A computation path in H' defines an additional sequence of events $Q_1,Q_2,\ldots$, where Q_i is the event defined by the assignments of elements to the sets $A_1,\ldots,A_{i-1}$. The probability of an input instance at the start of step i of the computation is its conditional probability (in D) given the event $O_i \cap Q_i$.

We say that an oracle query $IND(B)$ in the j^{th} step is *local* if

$IND(B \cap (\bigcup_{k\geq j} A_k) = FALSE$ and B has a circuit in A_j,

or

$IND(B \cap (\bigcup_{k\geq j} A_k) = TRUE$ and for every $i>j$,

$|B \cap (\bigcup_{k\geq i} A_k)| \leq (j+\frac{1}{4})(2t-i+1)\frac{n}{4t^2}$.

Intuitively, as long as the algorithm executes only local queries, it can only learn about the content of one set A_j at a time.

Claim 2.1: *For any $i \leq t$, if all the queries in the first $i-1$ steps were local, then with probability $1-4pte^{-\frac{n}{1200t^3}}$ all the queries in the i^{th} step are local.*

Proof: Our probabilistic calculations are based on the following version of the Chernoff bound for the approximation of the tail of the Binomial distribution [AV, Ch]:

For every $n > 0$, $0 < p < 1$, and $0 < \beta < 1$,

$$\sum_{k=0}^{\lfloor(1-\beta)np\rfloor} \binom{n}{k} p^k (1-p)^{n-k} \leq Exp(-\frac{\beta^2 np}{2})$$

and

$$\sum_{k=\lceil(1+\beta)np\rceil}^{n} \binom{n}{k} p^k (1-p)^{n-k} \leq Exp(-\frac{\beta^2 np}{3}).$$

Let E denote the event "*at least one of the queries in the i^{th} step was not local*". Then

$$Prob[E \mid O_i \cap Q_i] = \frac{Prob[E \cap O_i \mid Q_i]}{Prob[O_i \mid Q_i]} \leq$$

$$\frac{Prob[E \mid Q_i]}{Prob[O_i \mid Q_i]}.$$

We first compute the probability of the event O_i, for $i>1$, $(O_1 = D)$. Let $IND(B \cap (\bigcup_{k \geq j} A_k))$ be an oracle query in the j^{th} step, $j<i$. If $IND(B \cap (\bigcap_{k \geq j} A_k)) = FALSE$ then since the query is local, B has a circuit in A_j, thus this oracle answer does not give any information about the distribution of the elements in $B \cap (\bigcup_{k \geq i} A_k)$. If $IND(B \cap (\bigcup_{k \geq j} A_k))$ was true then it gives the information that $B \cap (\bigcup_{k \geq i} A_k)$ is an independent set in the input matroid. Since the query was local we also know that

$$|B \cap (\bigcup_{k \geq i} A_k)| \leq (i-1+\frac{1}{4})(2t-i+1)\frac{n}{4t^2}.$$

$Prob[O_i \mid Q_i]$ is lower bounded by the probability that $p(i-1)$ given sets of size $(i-\frac{3}{4})(2t-i+1)\frac{n}{4t^2}$ are all independent in the input matroid restricted to $\bigcup_{k \geq i} A_i$. If a set B is not independent it must have an intersection of size at least

$$i\frac{n}{4t^2} > \left\lfloor (i-\frac{3}{4})(2t-i+1)\frac{n}{4t^2}\frac{1}{(2t-i+1)} \right\rfloor (1+\frac{1}{2i})$$

with one of the sets A_k. For given sets B and A_k, the distribution of the size of the intersection $B \cap A_k$ is stochastically bounded by the Binomial distribution with the parameters

$$\left\lfloor (i-\frac{3}{4})(2t-i+1)\frac{n}{4t^2} \ , \ \frac{1}{(2t-i+1)} \right\rfloor.$$

Using the Chernoff bound [Ch], we compute

$$Prob[|B \cap A_k| \geq i\frac{n}{4t^2}] \leq$$

$$EXP(-\frac{(i-\frac{3}{4})}{3}(2t-i+1)\frac{n}{4t^2}\frac{1}{(2t-i+1)}\frac{1}{4i^2})$$

$$\leq e^{-\frac{n}{96t^3}}.$$

Since there are no more than $p(i-1) \leq pt$ possible B's, and no more than $2t$ A_k's we get

$$Prob[O_i \mid Q_i] \geq 1 - 2pt^2 e^{-\frac{n}{96t^3}}.$$

Let $IND(C_1),\ldots,IND(C_p)$ be the p oracle queries at the i^{th} step, We distinguish between two cases:

If $|C_l \cap (\bigcup_{k \geq i} A_k)| \geq i(2t-i+1)\frac{n}{4t^2}(1+\frac{1}{8i})$ then by the Chernoff bound with probability $1 - e^{-\frac{1}{1200}\frac{n}{4t^3}}$ C_l has a circuit in A_l.

Else, (if C_l is smaller) then, again by the Chernoff bound, with probability $1 - 2te^{-\frac{1}{1200}\frac{n}{4t^3}}$, $|C_l \cap A_k| \leq (i+\frac{1}{4})\frac{n}{4t^2}$ for any $k>i$. Thus, with that probability, either $C_l \cap (\bigcup_{k \geq i} A_k)$ is not independent and has a circuit in A_i, or $C_l \cap (\bigcup_{k \geq i} A_k)$ is independent and for every $g>t$

$$|C_l \cap (\bigcup_{k \geq g} A_k)| \leq (i+\frac{1}{4})(2t-i+1)\frac{n}{4t^2}.$$

In both cases $IND(C_l)$ is a *local* query, and we can conclude that

$$Prob[E \mid Q_i \cap O_i] \leq \frac{Prob[E \mid Q_l]}{Prob[O_i \mid Q_l]} \leq$$

$$\frac{2pte^{-\frac{n}{1200t^3}}}{1-2pt^2e^{-\frac{n}{96t^3}}} \leq 4pte^{-\frac{n}{1200t^3}}.$$

The result of Claim 2.1 implies:

Claim 2.2: *With probability greater than* $1-4pt^2e^{-\frac{n}{1200t^3}}$ *all the queries in the first* t *steps are local.*

If the algorithm terminates in step j then this step includes a query $IND(B)$ where $IND(B) = TRUE$ and $|B|$ equal the rank of the matroid. In any of the first t steps such a query is not *local*, since if B is a base of the matroid then

$$|B \cap (\bigcup_{k \geq j+1} A_k)| \geq \sum_{k=j+1}^{2t} k\frac{n}{4t^2} >$$

$$(j+\frac{1}{4})(2t-j+1)\frac{n}{4t^2}.$$

The probability of executing a query that is not *local* in the first t steps of the algorithm is not greater than $4pt^2e^{-\frac{n}{1200t^3}}$. Therefore, with probability $\geq 1-4pt^2e^{-\frac{n}{1200t^3}}$ the number of parallel steps is at least t, and hence $AT^{ind}_{det}(n,p,D) \geq t(1-4pt^2e^{-\frac{n}{1200t^3}})$. Choosing $t = (\frac{n}{2400 \log np})^{\frac{1}{3}}$ we obtain

$$T^{ind}_{prob}(n,p) \geq AT^{ind}_{det}(n,p,D) = \Omega[(\frac{n}{\log np})^{\frac{1}{3}}].$$

Remarks:

1. We prove a somewhat stronger result than the statement of Theorem 2.1 since the lower bound remains valid even if we restrict the input to the set of partition matroids which is a special case of linear matroids.

2. The rank of the matroids used in the proof is linear in their size, therefore we also proved that $\Omega[(\frac{n}{\log pn})^{\frac{1}{3}}]$ is a lower bound for the worst-case expected time to construct a base of a rank n matroid using the independence oracle and parallelism p.

3. UPPER BOUNDS

3.1. PRELIMINARIES

Our algorithms will use two basic operations on matroids, restriction and contraction [We].

Let $M = (E,I)$ be a matroid, and let $E' \subseteq E$.

The matroid M *restricted* to E', $M' = M|E' = (E',I')$ is defined by $I' = \{B \subseteq E' : B \in I\}$.

Let A be an independent set in M. The matroid M *contracted* on the set $E-A$, $\tilde{M} = M.(E-A) = (E-A,\tilde{I})$ is defined by $\tilde{I} = \{B \subseteq E-A : B \cup A \in I\}$.

Both operations will be used to repeatedly decrease the size of the problem. The following properties about the smaller system are used in the proofs [We]:

Fact 1: *If* M *is a matroid so are* M' *and* $\tilde{M}$.

Fact 2: *If* $B \subseteq E'$ *then* $IND_{M'}(B) = IND_M(B)$. *If* $B \subseteq E-A$ *then* $IND_{\tilde{M}}(B) = IND_M(A \cup B)$.

Fact 3: *If* $\tilde{U}$ *is a base of* $\tilde{M}$ *then* $A \cup \tilde{U}$ *is a base of* M.

We say that a restricted matroid M' is *full* if $rank(M') = rank(M)$.

Fact 4: *If* M' *is full then every base of* M' *is a base of* M.

Fact 5: *Let* Γ *be a family of circuits in* M *and let* $R \subseteq E$ *be a set s.t.*

1. $R \subseteq \bigcup_{C \in \Gamma} C$ *(every member of* R *appears in at least one circuit).*

2. $|R \cap C| \leq 1$ *for every* $C \in \Gamma$.

Then $M' = M|(E-R)$ is full.

3.2. GENERAL MATROIDS

THEOREM 3: $T^{ind}_{det}(n,n) = O(\sqrt{n})$.

PROOF: Let $M = (E,I)$ denote the input matroid, $E = n$. We show that in $O(1)$ steps and using n processors we can reduce the problem of constructing a base in M to that of constructing a base in a submatroid $M'=(E',I')$ where $E' \leq E(1 - \frac{1}{\sqrt{n}})$. Let the ground set E be ordered, i.e. $E = \{e_1, \ldots, e_n\}$, define $A[j,k] = \{e_i : j \leq i \leq k\}$ and let $r = \lfloor \sqrt{n} \rfloor$.

For all $j=0, \ldots, r$ test $IND(A[jr,(j+1)r-1])$.

If there exist j' such that $IND(A[j'r,(j'+1)r-1] = TRUE$ then it is enough to find a base, for the contracted matroid $M' = M.(E - A[j'r,(j'+1)r-1])$.

Else for every $j=1, \ldots, r$ let k_j be the smallest index such that $k_j > jr$, $IND(A[jr,k_j-1] = TRUE$ and $IND(A[jr,k_j] = FALSE$. Let $R = \{e_{k_j} : 0 \leq j < r\}$; then the elements of R lie on a disjoint circuits, thus the restricted matroid $M' = M|(E-R)$ is full.

In both cases the size of M', $|E'| \leq |E|(1-\frac{1}{\sqrt{n}})$.

3.3. GRAPHIC MATROIDS

A matroid $M = (E,I)$ is *graphic* if there exists a graph $G = (V,E)$ such that for all $A \subseteq E$, $A \in I$ iff A is a forest in G. Any such graph G is said to *represent* the matroid M. For the following proof we need the notion of the girth. The *girth* of a graph G, $g(G)$, is the size of its smallest cycle. The *girth* of matroid M, $g(M)$ is the size of its smallest circuit. If G represents M than $g(G) = g(M)$.

THEOREM 4: *For graphic matroids* $T^{ind}_{det}(n, n^{O(\log n)}) = O(\log n)$.

PROOF: The algorithm works in iterations, each iteration reduces the rank of the matroid that is considered. This will be done in two steps. First a restricted full matroid of high girth is obtained, then a large independent set is found in the restricted matroid. The input to the next iteration is the restricted matroid contracted on the independent set, thus a matroid with smaller rank. The algorithm for increasing the girth works for any matroid, not necessarily graphics. The idea is simply to fix an ordering on the elements and then to remove in parallel the last element in every small cycle.

Procedure Increase_girth(M,k).
input: $M = (E,I)$, $0 \leq k < n$, $E = \{e_1, \ldots, e_n\}$.
$R \leftarrow \{e_i : exists A_i \subseteq \{e_1, \ldots, e_{i-1}\},$
$|A_i| < k$ s.t. $IND(A_i) = TRUE$ and
$IND(A_i \cup e_i) = FALSE\}$;
$E' \leftarrow E - R$;
$M' \leftarrow M|E'$;
output: M'.

Claim 3.1: *1. M' is full.*
2. $g(M') > k$.
3. The procedure takes $O(1)$ time and uses n^{k+2} processors.

Having a matroid $M = (E,I)$, $|E| = n$, with girth $g(M) > 6\log n$, we show how to construct in constant time an independent set $A \subseteq E$ with $|A| \geq \alpha n$ for a fixed $\alpha > 0$. We first describe a probabilistic algorithm that uses $O(n)$ processors, then we shaw how to eliminate the randomness using $n^{O(\log n)}$ processors.

Claim 3.2: *Let $A \subseteq E$, $|A| = \frac{n}{2}$ be a random subset chosen uniformly among all subsets of size $\frac{n}{2}$, then*

$$Prob[IND(A) = TRUE] \geq 1 - \frac{2}{n}.$$

Proof: Let $G = (V,E)$ be a graph that represents M. Note that as $|E| = n$,

$|V| \leq 2n$. Also, the girth of G satisfies $g(G) = g(M) > 6\log n$. If $g(G) > 2r$, there can be only one path of length r between any two vertices in G, therefore the total number of paths of length r in G is bounded by $\left\lfloor \binom{|V|}{2} \right\rfloor \leq 2n^2$. If A is dependent it contains a cycle in G. As all cycles in G are longer than $6\log n$, A certainly contains a path of length $3\log n$. Thus

$$Prob[IND(A) = TRUE] \geq 1 - 2n^2 2^{-3\log n} \geq 1 - \frac{2}{n}.$$

Claim 3.2 clearly implies a probabilistic algorithm. It also shows that the problem of finding a large independent set can be formulated in terms of constructing a large set that does not contain any member of a small predefined family of small forbidden subsets. The next section shows how to solve this problem deterministically.

3.3.1. EXPLICIT CONSTRUCTION OF PACKINGS

Let U be a universe of n elements. A (b,k)-family is a family of at most b subsets of U, each containing at least k elements.

A (b,k)-family W is (s,t)-*universal* if for any (s,t)-family F, there exists a set from W that contains no set from F.

Note that in the context of the previous discussion, F is the family of forbidden subsets (paths). The family F itself is unknown except for its parameters $s = 2n^2$, and $t = 3\log n$. A (b,k)-family W that is $(2n^2, 3\log n)$ -universal, with $k = \Omega(n)$ will give us a way of finding an independent set by simply testing in parallel all members of W for independence. Hence we need an explicit construction of W in which b is as small as possible (it corresponds to the number of processors required for the execution).

We first reduce the problem of finding a universal families to that of finding packings. A (b,k)-family W is a (t,λ)-packing if every t-subset of U is a subset of at most λ members of W.

Claim 3.3: *If a (b,k)-family W is a (t,λ)-packing, and $b > s\lambda$, then W is (s,t)-universal.*

To construct such a packing, assume $n = 2^m$ and consider the elements of U as the elements of the finite field $GF(2^m)$. Let $P: GF(2^m) \rightarrow \{0,1\}$ be any fixed predicate which partitions the field into two equal halves. Let $f \in GF(2^m)[x]$ be a polynomial over the field. We interpret the n-bit vector $v(f) = (P(f(0)), \ldots, P(f(2^m - 1)))$ as a characteristic vector of a subset $V(f)$ of U. Consider the family of subsets

$$V_t = \{V(f) : f \in GF(2^m)[x],\ deg(f) < t\}.$$

Claim 3.4: *Every t-subset of U is a subset of exactly $(\frac{n}{2})^t$ sets in V_t.*

Proof: V_t has n^t members. By Lagrange's interpolation theorem, we know that for any t elements in the field $i_1, \ldots, i_t$, $\langle f(i_1), \ldots, f(i_t)\rangle$ takes all possible n^t values as we vary over all polynomials f of degree $< t$. Hence every t-bit vector occurs $(\frac{n}{2})^t$ times in $\langle P(f(i_1)), \ldots, P(f(i_t))\rangle$ as we vary over all polynomials. This is true in particular for the all 1's bit vector, which corresponds to the number of occurrences of the subset $\{i_1, \ldots, i_t\}$ in subsets of V_t.

Set $B_t = \{V \in V_t : |V| \geq \frac{n}{3}\}$.

Claim 3.5: *B_t is a $(\frac{n^t}{4}, \frac{n}{3})$-family which is $(2^{t-3}, t)$-universal.*

Proof: The fact that B_t has at least $\frac{n^t}{4}$ members follows from Markov's inequality. By construction, as it is a subfamily of V_t, it is a $(t, (\frac{n}{2})^t)$-packing. Since $b = \frac{n^t}{4} > 2^{t-3}(\frac{n}{2})^t = s\lambda$, by Claim 4.3, B_t is $(2^{t-3}, t)$-universal.

Returning now to the problem of finding a

large independent set in a graphic matroid with high girth, we note that for $t = 3\log n$, $2^{t-3} > 2n^2$, so by Claim 4.5 at least one of the $O(n^t)$ sets in B_t is independent.

REFERENCES:

[ACGS] W. Alexi, B. Chor, O. Goldreich and C.P. Schnorr. RSA/Rabin bits are $\frac{1}{2} + \frac{1}{poly(\log N)}$ secure. *Proc. 25th FOCS*, 1985. pp 449-457.

[AV] D. Angluin and L.G. Valiant. Fast probabilistic algorithm for Hamiltonian circuits and matchings. *J. Comp. Syst. Sci.* 12, 6 (1979) pp 155-193.

[Bo] A. Borodin. On relating time and space to size and depth. *SIAM J. Comp.*, 6 (1977) pp 733-744.

[BGH] A. Borodin, J. von zur Gathen and J. Hopcroft. Fast parallel matrix and GCD computations. *Proc. 23rd FOCS*, 1982. pp 65-71.

[Ch] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. of Mathematical Statistics* 23 (1952) pp 493-509.

[Go] L.M. Goldschlager. A unified approach to models of synchronous parallel machines. *Proc. 10th STOC 1978. pp 89-94.*

[HK] D. Hausmann and B. Korte. On algorithmic versus axiomatic definitions of matroids. *Mathematical Programming Study*, 14 (1981) pp. 98-111.

[KW] R.M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. *Proc. 16th STOC*, 1984. pp 266-272.

[KUW1] R.M. Karp, E. Upfal and A. Wigderson. Are search and decision problems computationally equivalent?. *Proc. 17th STOC*, 1985.

[KUW2] R.M. Karp, E. Upfal and A. Wigderson. Constructing a perfect matching is in Random NC. *Proc. 17th STOC*, 1985.

[La] E.L. Lawler. *Combinatorial Optimization, Networks and Matroids.* Holt, Rienhart and Winston. 1976.

[Lu] M. Luby. A simple parallel algorithm for the maximum independent set problem. *Proc. 17th STOC 1985.*

[Ma] C. Martel. Lower bounds on parallel algorithms for finding the first maximal independent set. Submitted 1985.

[We] D.J.A. Welsh. *Matroid Theory.* Academic Press, 1976.

[Ya] A.C. Yao. A probabilistic computation: towards a unified measure of complexity. *Proc 18th FOCS*, 1977. pp 222-227.