Efficient Sequential and Parallel Algorithms for Maximal Bipartite Sets

David Pearson*
Vijay V. Vazirani**

TR 91-1224
August 1991

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Supported by a General Electric Foundation Graduate Fellowship
**Supported in part by NSF Grant DCR 85-52938 and PYI matching funds from AT&T Bell Laboratories and Sun Microsystems Inc. at Cornell University.

Efficient Sequential and Parallel Algorithms for Maximal Bipartite Sets

David Pearson*
Computer Science Department
Cornell University
Ithaca, NY 14853

Vijay V. Vazirani†
Department of Computer Science and Engineering
Indian Institute of Technology
New Delhi – 110016, India

Abstract

A maximal bipartite set (MBS) in an undirected graph $G = (V, E)$ is a maximal collection of vertices $B \subseteq V$ whose induced subgraph is bipartite. In this paper we present efficient sequential (linear time) and parallel (NC) algorithms for constructing an MBS.

1 Introduction

In the last few years several efficient parallel algorithms have appeared for the maximal independent set (MIS) problem [GS, KW, Luby]. This may be thought of as the problem of finding a maximal 1-colorable set in a given graph. In this paper we consider the natural extension to finding a maximal 2-colorable set.

*Supported by a General Electric Foundation Graduate Fellowship

†Supported in part by NSF Grant DCR 85-52938 and PYI matching funds from AT&T Bell Laboratories and Sun Microsystems Inc. at Cornell University.

A maximal bipartite set (MBS) in an undirected graph $G = (V, E)$ is a maximal collection of vertices B such that the subgraph induced on B is bipartite. We give a linear-time sequential algorithm for constructing an MBS; this algorithm greedily adds vertices as long as they do not introduce an odd length cycle. There appears to be no way to parallelize this algorithm, however. Our main result is a parallel algorithm for finding an MBS, using an entirely different approach.

It is natural to use a parallel algorithm for MIS as a subroutine, and start as follows: Let I_1 be an MIS in $G = (V, E)$ and I_2 be an MIS in the graph induced on $V - I_1$. The graph induced on $I_1 \cup I_2$ is bipartite; however, it may not be maximal, since a recoloring of $I_1 \cup I_2$ may allow us to include additional vertices. This is where the difficulty of our problem lies.

The extension to maximal k-colorable subgraphs, $k \geq 3$, is NP-hard since the problem of deciding whether a graph is k-colorable is NP-complete for $k \geq 3$ [GJ].

2 Sequential MBS algorithms

A straightforward algorithm for finding an MBS is to greedily add vertices as long as the induced subgraph on the chosen vertices remains bipartite. Since testing for bipartiteness can be done in $O(m)$ time, this is an $O(mn)$ algorithm. Notice that this algorithm will find the lexicographically first MBS if the vertices are tested in their lexical order. In this section we give an $O(m)$ algorithm for finding an arbitrary MBS, and an $O(m\alpha(m, n))$ algorithm for finding the lexicographically first MBS.

2.1 Finding an arbitrary MBS

Our algorithm is a modification of breadth-first search (BFS). The usual BFS, started at vertex v, assigns levels to all vertices—if the shortest path from v to u has length l, then $level(u) = l$. All the edges of G are between adjacent levels or within a single level. Clearly, G has an odd-length cycle (and hence is not bipartite) iff two vertices at the same level are connected by an edge.

Our modified BFS discards vertices during the search to remove odd cycles. After constructing the ith level, the algorithm searches from all the vertices at level i to find the set S of newly reachable vertices. It then finds an MIS I in the subgraph induced on S. The vertices in $S - I$ are discarded, and the vertices in I are assigned level $i + 1$.

Let B be the final set of undiscarded vertices. We have essentially conducted a BFS on the graph induced on B. Since no two vertices of B having the same level are connected by an edge, the induced graph is bipartite. Moreover, including any vertex from $V - B$ will violate this condition. Hence B is an MBS. The running time of the algorithm is clearly $O(m)$, where $m = |E|$.

2.2 Finding the lexicographically first MBS

We will implement the test for bipartiteness efficiently using the union-find data structure in the $O(mn)$ greedy algorithm given above, and thereby obtain an $O(m\alpha(m,n))$ algorithm for finding the lex-first MBS.

Let B be the set of vertices picked at an arbitrary point during the running of the greedy algorithm. In general, the graph induced on B will consist of several connected components, each one bipartite. We will maintain each side of each component as a separate set, using the union-find data structure. We determine whether the next vertex, v, can be added to B as follows:

1. Examine all edges (u, v) incident on v.
2. If two such edges are incident on the opposite sides of the same connected component, omit v.
3. Otherwise, union all the sets at which these edges are incident. Also union the other sides of these components and add v to this second set.

Clearly B will be an MBS, and the algorithm will require $O(m)$ find operations and $O(n)$ set-union operations. So its total running time is $O(m\alpha(m,n))$, where $\alpha(m,n)$ is the functional inverse of Ackerman's function, the smallest integer i such that $A(i, \lfloor m/n \rfloor) > \log n$) [Tar].

3 A parallel MBS algorithm

This algorithm proceeds in stages, maintaining a bipartite set which is extended incrementally until it cannot be extended any further. Let $G = (V, E)$

be the original graph, and let $B \subseteq V$ be the current bipartite set. Initially, let B be the null set.

For each bipartite set B, we will define an incompatibility relation R_B on the vertices of $V - B$. Two vertices v_1 and v_2 are *incompatible* if there is an odd-length path from v_1 to v_2 whose intermediate vertices, if any, come from B. (The length of a path is the number of edges it traverses.) Any two vertices which share an edge in the original graph will always be incompatible, so R_B may be considered as an extension of the original edge set. Intuitively, two vertices are compatible if they can both be added to B and given the same color.

At each iteration, perform the following steps:

1. Set $U = V - B$.

2. Compute the incompatibility relation R_B on vertices in U.

3. Let I be an MIS of the graph with vertices U and edges R_B. If $I = \phi$ then stop and output B.

4. Set $B = B \cup I$.

Note that the incompatibility relation computed in step 2 may be reflexive, i.e. a vertex v may be incompatible with itself. If this happens, $\{v\}$ is a dependent set. We assume the MIS procedure will never include such a vertex in any independent set. This is why the MIS I may be empty, even if $B \neq V$. Alternatively, it is simple to remove any such vertices with self-loops before invoking the MIS subroutine.

3.1 Correctness and running time

Lemma 1 *At each iteration, the set B is a bipartite set.*

Proof. By induction. At the start of the algorithm, $B = \phi$, which is trivially bipartite. At each iteration, assume that the original B is bipartite. We wish to establish that $B \cup I$ is also bipartite. Consider any cycle in $B \cup I$. If the cycle contains no vertices from I, it lies entirely within B, and since B is bipartite the cycle must have even length. Otherwise we may write the cycle as $i_1, \overline{b_1}, i_2, \overline{b_2}, \ldots, i_n, \overline{b_n}, i_1$ where $i_j \in I$ and $\overline{b_j}$ is a sequence of elements of B. Since all the elements of I are compatible, the path length from i_j to i_{j+1} (or i_n to i_1) must be even, so the entire cycle must also have even length. Since every cycle has even length, $B \cup I$ is a bipartite set. □

Lemma 2 *If at some iteration $I = \phi$, then B is a maximal bipartite set.*

Proof. Suppose $I = \phi$, but B is not maximal. Then there is a bipartite set B', such that $B \subset B'$. Choose $v \in B' - B$. There can be no odd-length path from v to itself using intermediate vertices in B, since any such path would be an odd-length cycle in B'. So v is compatible with itself, and the set $\{v\}$ is therefore an independent set on (U, R_B). This contradicts the fact that the maximal independent set I was null. □

Step 3 of the algorithm can be performed using Luby's MIS algorithm in time $O(\log^2 n)$ on an EREW PRAM using $O(n^4)$ processors deterministically, or $O(n^2)$ processors using randomization [Luby]. Step 2, forming the incompatibility relation, can also be implemented in $O(\log^2 n)$ time on n^3 processors using matrix multiplication and matrix powers. We will establish in lemma 3 that the number of iterations is $O(\log n)$, making the total running time of the MBS algorithm $O(\log^3 n)$. This establishes that the MBS problem is in NC^3 (see [Cook] for a discussion of the class NC).

Lemma 3 *The number of iterations required by the MBS algorithm is $O(\log n)$.*

Proof. To see why $O(\log n)$ iterations are sufficient, it is convenient to start from the end and work backward to the beginning. Let k be the total number of iterations performed, and let I_j and B_j be the sets I and B computed in stage $j \geq 1$.

Suppose a new vertex v is introduced at step $j > 2$. Since I_{j-1} was a maximal independent set, we know that v must have been incompatible with some vertex $u \in I_{j-1}$, relative to B_{j-1}. By the same argument, v must have been incompatible with some vertex $t \in I_{j-2}$, and u must have been incompatible with some vertex $s \in I_{j-2}$. Leaving aside the question of whether these vertices are all distinct, we can apply the same argument to s and t and construct the binary tree in Figure 1.

We must now establish that all the vertices in this tree are distinct. Clearly the vertices on different levels are distinct, since they were added

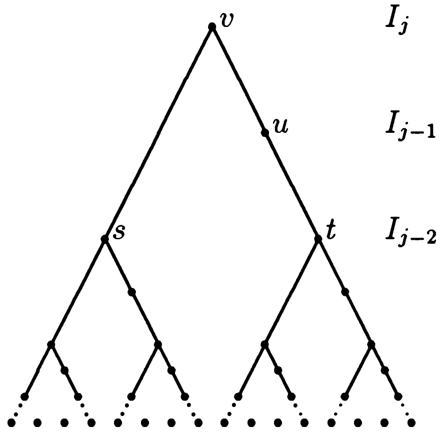

Figure 1

to B at different iterations. But what about vertices on the same level? Suppose that two of these are actually the same vertex. Let a be such a vertex, occurring at the highest level containing any shared vertices. Let $b \in I_j$ be the lowest common ancestor of the two tree positions where a occurs, and let c and d be the vertices two levels below b, as in Figure 2. Then we know

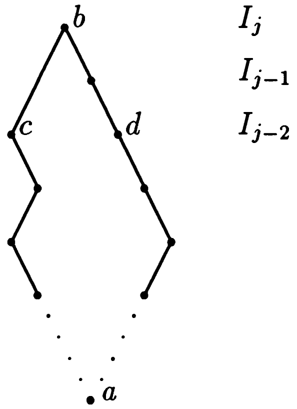

Figure 2

that there is a path from c to d in B_{j-2}, through the shared vertex (in fact, it may be that $c = d = a$, which we will consider to be a zero-length path). Since c and d both belong to I_{j-2}, we know that there is no odd-length path between them in B_{j-2}, so the path must have even length. But this would give us an odd-length cycle in B_j, by combining the three odd-length paths connecting c to d through b, then returning by the even-length path from d to c through a. This contradicts the fact that B_j is bipartite, so there can be no such shared vertex a.

Since all the vertices in the tree in Figure 1 are distinct, this tree contains at least $\Omega(2^{k/2})$ vertices, and so the total height k can only be $O(\log n)$ where n is the number of vertices in the original graph. $\square$

The following theorem summarizes the results above.

Theorem 1 *There is a parallel algorithm that solves the MBS problem in* $O(\log^3 n)$ *time using* $O(n^4)$ *processors.*

3.2 A lower bound

We have established that the MBS algorithm will never take more than $O(\log n)$ phases. Now we consider the question of whether it will ever actually require that many. We can show that the bound is tight, in the following sense.

Theorem 2 *An adversary that may choose the initial graph and the MIS obtained at each phase can force the MBS algorithm to use* $\Omega(\log n)$ *phases.*

Proof. By construction. We will define a class of graphs G_k as follows. Each G_k will be a forest of k trees, and will contain $2^k - 1$ vertices. G_1 is just a single vertex. G_{k+1} consists of two copies of G_k with one new root vertex, and edges from the new vertex to all the root vertices of one copy of G_k. Figure 3 shows G_5.

G_k contains no cycles, so the entire graph is clearly bipartite. Note also that every non-leaf vertex has an edge to some vertex at *each* lower level.

To force the MBS algorithm to use k phases, we will give it the graph G_k. At each iteration i our adversary MIS routine will return the vertices at level i. At this point, the bipartite set (B in the algorithm) will consist of levels 1 through $i-1$, and the remaining vertices (U) will contain levels i through k. Because every connected component of B has an edge to at most one vertex outside B, the only incompatible vertices in U will be those that share an

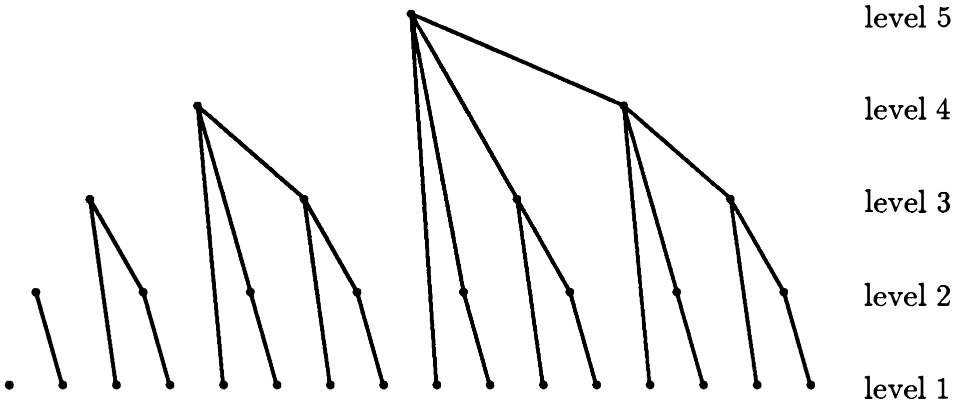

Figure 3

edge—the incompatibility relation will add no new edges. The vertices at level i are therefore an independent set, and since every vertex at a higher level has a neighbor at level i, this set will also be maximal.

Let $n = 2^k - 1$, the number of vertices in G_k. Then the number of phases required, k, is $\Omega(\log n)$. $\square$

To establish this result we treated the MIS subroutine as a black box, that might make the worst possible choice of independent set each time it is called. We would like something stronger—a lower bound on the total running time, given a *specific* MIS algorithm. To obtain such a bound, one would need a precise analysis of the independent sets actually constructed by the various MIS algorithms (or for randomized algorithms, of the distribution of the sets). We would also need a lower bound for the time required by the MIS algorithms. These are still open problems.

3.3 Finding the lexicographically first MBS

We will show that the problem of finding the lex-first MBS is P-complete for NC, thereby giving evidence that it is not parallelizable.

Cook, in [Cook] proved that finding the lexicographically first MIS is P-complete (actually his proof was for the equivalent problem, maximal clique). We can reduce lex-first MIS to lex-first MBS as follows: Suppose we are asked to find an MIS in a graph $G = (V, E)$ with the vertices ordered $\langle v_1, v_2, ..., v_n\rangle$. Form the graph $G' = (V', E')$ by adding one new vertex b with the ordering $\langle b, v_1, v_2, ..., v_n\rangle$ and adding an edge (b, v_i) between b and every vertex $v_i \in V$. The lex-first MBS of G' must contain b and an MIS of G. Since we maintain the original ordering among all the vertices of V, the MIS we find will be the lex-first MIS of V. This reduction can clearly be done in NC^1, so lex-first MBS is also P-complete.

4 Maximal bipartite subgraphs

The MBS problem was, in effect, to find a maximal set of vertices whose induced subgraph is bipartite. In this section, we consider the problem of finding a maximal set of *edges* so that the graph induced on them is bipartite. The resulting graph is called a maximal bipartite subgraph (MBG). Most of the techniques discussed above can be used for the MBG problem.

In the sequential case, both of the MBS algorithms can be easily adapted to find an MBG. To find an arbitrary MBG, we find a depth-first search tree for G, then add all back-edges that span an odd number of levels in the tree. To find the lex-first MBG, use the same data structure as for lex-first MBS, but iterate over the edges instead of the vertices; an edge is omitted iff it runs between two vertices in the same side of a component.

In the parallel case, to obtain an arbitrary MBG, we give a simple reduction of MBG to MBS. Notice that the parallel MBS algorithm need not be started with $B = \phi$. The initial B may be any bipartite set, in which case the MBS algorithm will compute a maximal bipartite extension of B. Given a graph G, we can compute an MBG in NC as follows:

1. Form a new graph G' in which every edge (u, v) of G has been replaced by the three edges, (u, s_{uv}), (s_{uv}, s_{vu}), and (s_{vu}, v), where s_{uv} and s_{vu} are new vertices.

2. Find an MBS in G' which includes all the original vertices of G.

3. Form the MBG by including an edge (u, v) whenever both the added vertices s_{uv} and s_{vu} are in the MBS.

The correctness follows directly from the observation that $v_1, v_2 ... v_k, v_1$ is an even cycle in G iff $v_1, s_{v_1 v_2}, s_{v_2 v_1}, v_2 ... v_k, s_{v_k v_1}, s_{v_1 v_k}, v_1$ is an even cycle in G'.

For the lex-first MBG problem, the parallel complexity is difficult to determine. As with lex-first MBS, the problem seems to require inherently sequential decisions, and a fast parallel algorithm seems unlikely. The main difficulty in carrying out a P-completeness proof is simulating fan-out. A similar difficulty arises in carrying out a P-completeness proof for lex-first maximal matching; however, this problem has been shown to be CC-complete [MS] (a weaker evidence of inherent sequentiality). It would be interesting to show a similar result for the lex-first MBG problem.

5 Summary

Given a graph $G = (V, E)$, let $n = |V|$ and $m = |E|$. The following table summarizes our results for the problems discussed above.

	Maximal Bipartite Set	Maximal Bipartite Subgraph
Sequential arbitrary	$O(m)$	$O(m)$
Sequential lex-first	$O(m\alpha(m,n))$	$O(m\alpha(m,n))$
Parallel arbitrary	NC^3	NC^3
Parallel lex-first	P-complete	OPEN

6 Acknowledgements

We wish to thank Steve Mitchell, Samir Khuller, and Alessandro Panconesi for valuable discussions about this problem.

References

[ABI] N. Alon, L. Babai and A. Itai, A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem, *J. Algorithms* **7** (1986), 567-583.

[Cook] S. A. COOK, Taxonomy of Problems with Fast Parallel Algorithms, *Information and Control* **64** (1985), 2-22.

[GJ] M. R. GAREY and D. S. JOHNSON, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman, New York (1979).

[GS] M. GOLDBERG and T. SPENCER, A New Parallel Algorithm for the Maximal Independent Set Problem, *SIAM J. Computing* **18** (1989), 419-427.

[Har] F. HARARY, "Graph Theory," Addison-Wesley, Reading, MA, 1969.

[KW] R. M. KARP and A. WIGDERSON, A Fast Parallel Algorithm for the Maximal Independent Set Problem, *in* "Proc. 16th Annual ACM Symposium on Theory of Computing," pp. 266-272, 1984.

[Luby] M. LUBY, A Simple Parallel Algorithm for the Maximal Independent Set Problem, *SIAM J. Computing* **15** (1986), 1036-1053.

[MS] E. W. MAYR and A. SUBRAMANIAN, The Complexity of Circuit Value and Network Stability, *in* "Proc. 4th Structure in Complexity Theory," 114-123, 1989.

[Tar] R. E. TARJAN, Efficiency of a good but not linear set union algorithm, *J. Assoc. Computing Mach.* **22** (1975), 215-225.