A New Parallel Algorithm for Planarity Testing

David A. Bader* Sukanya Sreshta

Electrical and Computer Engineering Department
University of New Mexico, Albuquerque, NM 87131
{dbader, sukanya}@ece.unm.edu

October 1, 2003

Abstract

This paper presents a new parallel algorithm for planarity testing based upon the work of Klein and Reif [14]. Our new approach gives correct answers on instances that provoke false positive and false negative results using Klein and Reif's algorithm. The new algorithm has the same complexity bounds as Klein and Reif's algorithm and runs in $O\left(\frac{n \log^2 n}{p}\right)$ time for any number $p \leq n$ processors of a Concurrent Read Exclusive Write (CREW) Parallel RAM (PRAM). Implementations of the major steps of this parallel algorithm exist for symmetric multiprocessors and exhibit speedup when compared to the best sequential approach. Thus, this new parallel algorithm for planarity testing lends itself to a high-performance shared-memory implementation.

1 Introduction

Informally, a graph is planar if it can be embedded onto the plane without edge crossings. The planarity problem consists of detecting if the given graph is planar and if so of constructing a planar embedding of the given graph. Planar graphs have a wide range of applications such as in VLSI layout and are of great theoretic interest as many NP-Hard problems such as clique, feedback arc set, bipartite subgraph, and max cut, can be solved in polynomial time in the case of planar graphs [9].

A *path addition approach*, developed by Hopcroft and Tarjan [10] was the first linear time sequential planarity algorithm. A *vertex addition approach*, developed by Lempel, Even and Cederbaum [15] was made to run in linear time by using the PQ-Trees data structure of Booth and Lueker [5] and the *st-numbering* algorithm of Even

*This work was supported in part by NSF Grants CAREER ACI-00-93039, ITR ACI-00-81404, DEB-99-10123, ITR EIA-01-21377, Biocomplexity DEB-01-20709, and ITR EF/BIO 03-31654.

and Tarjan [8]. JáJá and Simon [11] first showed that testing planarity is in NC, the class of problems with parallel algorithms that run in polylog time with a polynomial number of processors. Using [5, 15], Klein and Reif [14] presented a parallel planarity algorithm, based on Klein's master's thesis [12], that runs in $O\left(\log^2 n\right)$ time with a linear number of processors on the CREW PRAM. Ramachandran and Reif [18] presented a logarithmic time parallel planarity algorithm for the CRCW PRAM based on open ear decomposition. Caceres *et al.* [6] presented a coarse-grained parallel (CGM) planarity algorithm based on Klein and Reif's PRAM algorithm.

This paper shows cases for which the planarity algorithm of Klein and Reif, and hence, also that of Caceres *et al.*, gives both false positive and negative results. We begin in section 2 with a brief overview of the Klein and Reif approach. In section 3 we illustrate instances that yield false positive and false negative results and the corresponding flaws in their algorithm. Our new parallel algorithm that works correctly on all inputs and runs in the same bounds as Klein and Reif's algorithm is described in section 4. Finally, we conclude in section 5 with future work.

2 Overview of Klein and Reif's Algorithm

Klein and Reif [13, 14], inspired by the sequential work of Lempel, Even and Cederbaum [15], described a parallel algorithm for planarity testing and finding an embedding of a planar graph. Their parallel algorithm uses a divide-and-conquer strategy, computing embeddings of subgraphs and combining them to form embeddings of larger subgraphs.

Their approach parallelizes the PQ-Tree data structure of Booth and Lueker [5] used to represent sets of graph embeddings. Three operations on PQ-Trees are defined, namely *multiple-disjoint-reduction*, *intersection* and *join*, and linear-processor parallel algorithms are given for these operations. The *multiple-disjoint-reduction* operation extends the *reduce* operation of Booth and Lueker to reduce the given PQ-Tree T simultaneously with respect to multiple disjoint sets. The *intersection* operation is used to reduce the given PQ-Tree T simultaneously with respect to multiple sets that are not necessarily disjoint. The *join* operation is a generalization of the tree-splicing operation of Booth and Lueker. The *multiple-disjoint-reduction* operation is used in the *segregation* of subsets in the PQ-Tree. The *join* operation is split into two procedures, one for computing the *provisional join* and the other for *verifying* the join. The *provisional join* is the correct join provided the correct join is not the empty PQ-Tree T_{null}. The *verification step* verifies the correctness of the *provisional join*, that is, determines if the correct join is T_{null}, using the *intersection* operation of PQ-Trees. The *provisional join* is used to proceed quickly to the next stage of the planarity algorithm and the *verification* step verifies all joins simultaneously once the final stage is complete.

A brief step-by-step outline of Klein and Reif's planarity algorithm is given below.

1. A graph is planar iff all its *biconnected components* are planar. Hence, compute the biconnected components of the given graph G and check each component for

planarity.

2. Compute the *st-numbering* of each biconnected component. Renumber all vertices according to the *st-numbering* with $s = 1$ and $t = n$.

3. Let $G^{(0)} = G$. A vertex $v \neq s, t$ in $G^{(i)}$ is *joinable* if v is adjacent to some vertex $u \neq s, t$ in $G^{(i)}$. A phase consists of a sequence of four stages namely *s-main*, *s-cleanup*, *t-main* and *t-cleanup* to reduce the number of *joinable* vertices by at least a factor of two. For a detailed description of these stages, please refer Pg. 237 of [14]. Hence, $O(\log n)$ phases are needed by the planarity algorithm, and let $G^{(i)}$ be the graph computed in phase i by contracting the joinable vertices in $G^{(i-1)}$. Each vertex v in G is initially represented by a PQ-Tree $T(v)$ (depicted in Fig. 1) whose root is a Q-node with two P-node children, v_{in} and v_{out} whose children are the incoming and the outgoing edges of V respectively in accordance with the *st-numbering*.

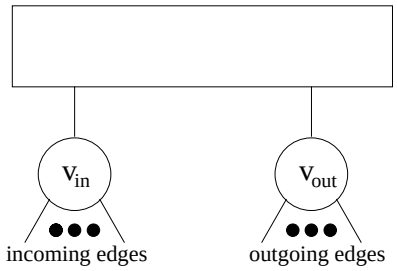

Figure 1: The PQ-Tree $T(v)$ representing vertex v in $G^{(0)}$

At every stage, the PQ-Trees of selected joinable vertices are combined using the *join* operation (described in Pg. 222 of [14]) to obtain the PQ-Trees of the resultant vertices. If a null tree T_{null} arises as a result of the *join* operation for some vertex v, then the graph G is *not planar*. On the other hand, if the contraction process continues until there are no joinable vertices remaining in $G^{(\mathcal{F})}$, the algorithm concludes that the given graph *is planar* using the result (from [16]), that a graph is planar iff its triconnected components are planar. The rationale for this step given by Klein and Reif is that s and t form a separation pair and the remaining vertices in $G^{(\mathcal{F})}$ are triconnected components which are planar (since their PQ-Trees are not T_{null}). For more details refer Pg. 239 of [14]. Finally, they sketch a method for obtaining a planar embedding if the planarity-testing algorithm has successfully terminated.

3 Instances for which Klein and Reif's Planarity Algorithm Fails

Klein and Reif's planarity algorithm has several flaws. We begin with example instances that provoke false positive and false negative results and describe the corresponding errors in their algorithm.

3.1 False Positive Example

To illustrate a case that yields a false positive, we consider a graph G_1 (Fig. 2) homeomorphic to the Kuratowski graph K_5.

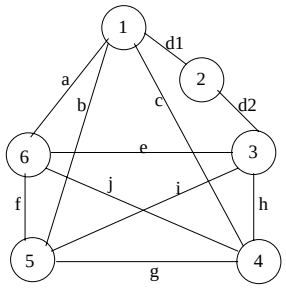

Figure 2: Graph G_1 homeomorphic to K_5

G_1 with $|V| = 6$ and $|E| = 11$ satisfies the Euler's formula, $|E| \leq 3|V| - 6$, and is itself a biconnected component, so it is further investigated by Klein and Reif's algorithm. The initial numbering of the vertices of the graph G_1 satisfies the constraints of *st-numbering*. The initial PQ-Trees of the vertices in G_1 are shown in Fig. 3. Note that in this paper, the label in the root Q-node of each PQ-Tree is the vertex/component it represents.

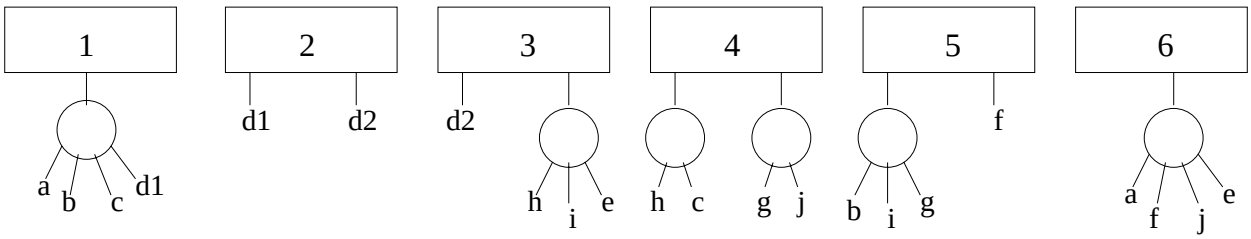

Figure 3: Initial PQ-Trees representing the vertices in G_1

The spanning trees (Fig. 4) constructed in the s and t phases of the algorithm are just a simple chain of vertices from $s = 1, \ldots, 6 = t$. Hence all our *join* operations are simple pairwise joins of two PQ-Trees T_0 and T_1 resulting in T_0'. To ensure clarity, we adopt similar notations from [14] and in the following description E_1 denotes the set of edges common to both components being joined (and which will be abstracted from

the merged component by the *join* operation) while D_1 denotes the set of remaining edges in T_1 (which are not in E_1).

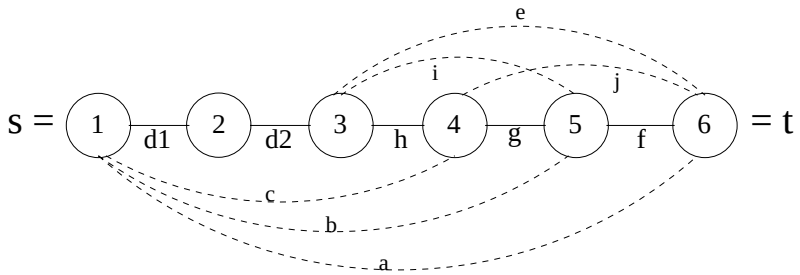

Figure 4: Spanning Tree constructed in the s and t phases

The first *s-main* stage consists of joining the PQ-Trees representing vertices 2 and 3 to form that of $2'$ and vertices 4 and 5 to form that of $4'$. The join of vertices 4 and 5 to form that of $4'$ is illustrated in detail. In this case $E_1 = \{g\}$ and $D_1 = \{b, i, f\}$. Following the *join* operation procedure (Pg. 222 of [14]), T_0 representing PQ-Tree of vertex 4 is segregated with respect to E_1 (Since $|E_1| = 1$, the initial PQ-Tree representing vertex 4 is already segregated with respect to E_1) and T_1 representing PQ-Tree of vertex 5 is segregated with respect to E_1 and D_1 (Fig. 5). Note that the labels under the arrows indicate the template from Booth and Lueker [5] that is applied to the PQ-Tree.

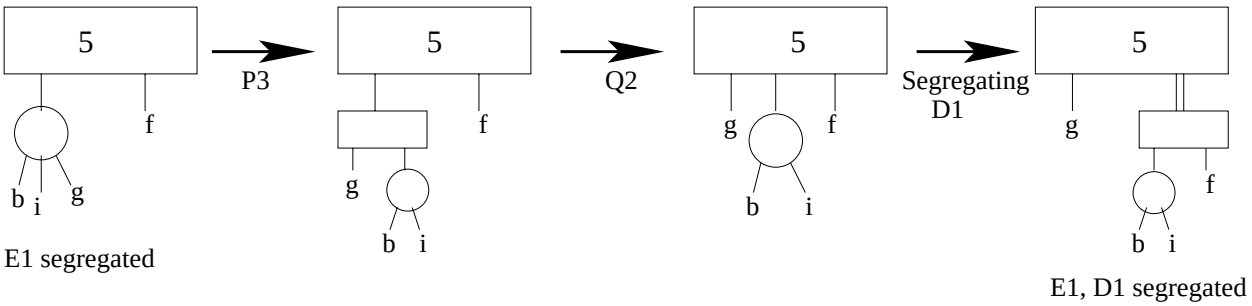

Figure 5: PQ-Tree representing vertex 5 segregated with respect to E_1 and D_1

The resulting trees representing components $2'$ and $4'$ at the end of *s-main* are shown in Fig. 6.

There are no odd leaves and hence the *s-cleanup* stage is empty. In the first *t-main* stage, the PQ-Trees representing components $4'$ and $2'$ are joined to form that of $4''$. In this case T_0 represents $4'$, T_1 represents $2'$, $E_1 = \{h, i\}$ and $D_1 = \{d_1, e\}$. T_0 is segregated with respect to E_1 (Fig. 7) and T_1 is segregated with respect to E_1 and D_1 (Fig. 8). The resultant PQ-Tree $4''$ at the end of *t-main* stage is shown in Fig. 9.

At the end of the *t-main* stage, there are no other joinable vertices and hence the algorithm concludes that G_1 (a graph homeomorphic to K_5) is planar.

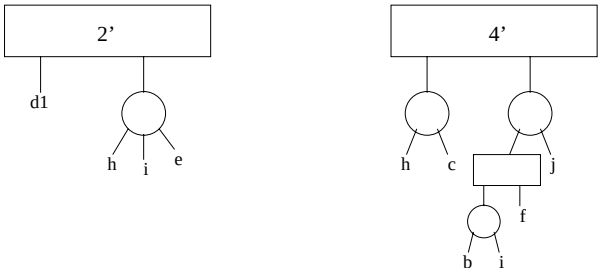

Figure 6: PQ-Trees representing components $2'$ and $4'$ at the end of *s-main* stage

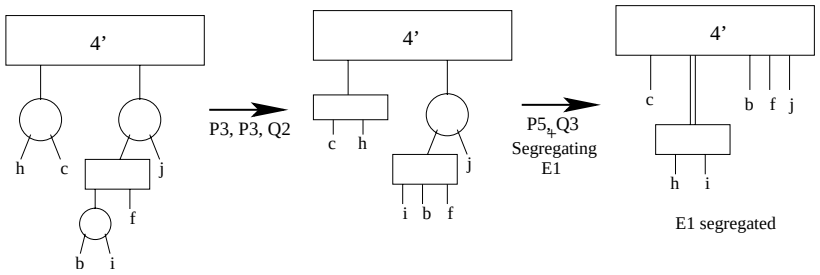

Figure 7: PQ-Tree representing component $4'$ segregated with respect to E_1

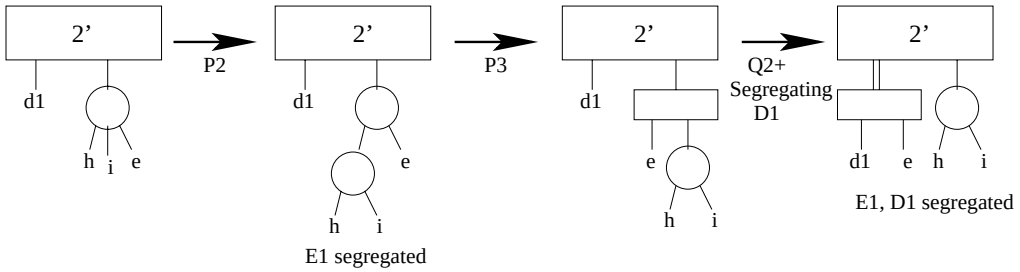

Figure 8: PQ-Tree representing component $2'$ segregated with respect to E_1 and D_1

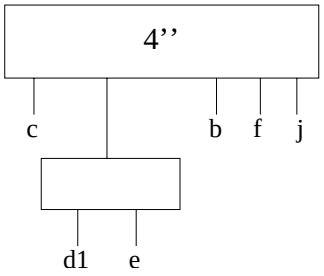

Figure 9: PQ-Tree representing component $4''$ at the end of the *t-main* stage

3.2 False Negative Example

To illustrate an instance that yields a false negative result, we consider a planar sub-graph G_2 (Fig. 10).

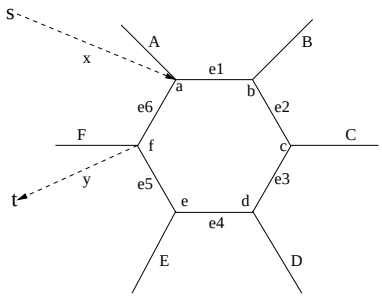

Figure 10: A Planar sub-graph G_2. Note that $\{s, a, b, c, d, e, f, t\}$ are vertices and all other labels are edges

This subgraph can be easily extended to a planar graph but is omitted here for clarity. The *st-numbering* is such that $a < b < c < d < e < f$. Let us assume that the edges A, B, C, D, E, F are outgoing edges from the vertices a, b, c, d, e, f, respectively. Further, for the sake of generality, let us assume that there is a path from s to a represented by the incoming edge x of a and there is a path from f to t represented by the outgoing edge y of f. The initial PQ-Trees of these six vertices in G_2 are shown in Fig. 11.

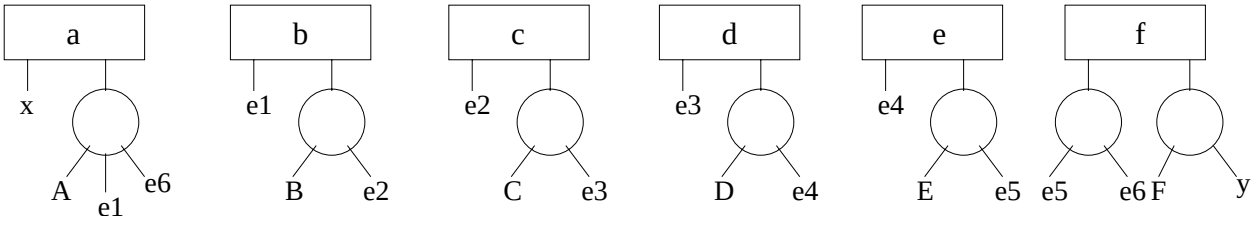

Figure 11: Initial PQ-Trees representing the vertices in G_2

The spanning trees constructed in the s and t phases of the algorithm are just a simple chain of vertices from $s, \ldots, a, b, c, d, e, f, \ldots, t$. Hence, similar to the false positive case, all our *join* operations are simple pairwise joins of two PQ-Trees T_0 and T_1 resulting in T_0'. The first *s-main* stage consists of joining the PQ-Trees representing vertices a and b to form that of a', vertices c and d to form that of c' and vertices e and f to form that of e'. The resulting trees representing components a', c' and e' at the end of *s-main* are shown in Fig. 12.

There are no odd leaves, and hence the *s-cleanup* stage is empty. In the first *t-main* stage, the PQ-Trees representing components e' and c' are joined to form that of e''. Fig. 13 shows the resulting PQ-Tree representing component e''.

Since, there are no odd leaves, the *t-cleanup* stage is empty. In the second *s-main* stage, the PQ-Trees representing components a' and e'' are joined to form that of

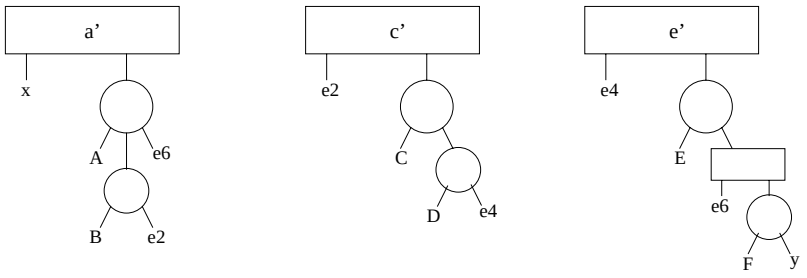

Figure 12: PQ-Trees representing components a', c', and e', at the end of *s-main* stage

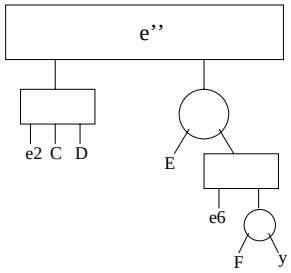

Figure 13: PQ-Tree representing component e'' at the end of *t-main* stage

a''. The PQ-Tree T_0 representing component a' can be segregated with respect to $E_1 = \{e_2, e_6\}$. However, PQ-Tree T_1 representing component e'' cannot be segregated with respect to both E_1 and $D_1 = \{C, D, E, F, y\}$. Therefore, step $J4$ of the join operation fails and the resulting tree is T_{null}. Hence, the algorithm may conclude that any graph having the planar subgraph G_2 is non-planar.

3.3 Errors in Klein and Reif's Algorithm

The false positive and false negative cases illustrate the following errors in Klein and Reif's algorithm.

1. The ending condition of Klein and Reif (quoted from Pg. 239 of [14]): "Using the fact ([16]) that a graph G is planar iff its triconnected components are, it follows that $G^{(0)}$ is planar, for s and t form a separation pair whose blocks are $H(v_1), \ldots, H(v_k)$" is incorrect for the following reasons:

   a) s and t need not necessarily form a separation pair. In the case of G_1 the vertices $1 = s$ and 3 form the separation pair. In other cases, like K_5 the vertices s and t do not form a separation pair and the entire graph is one triconnected component.

   b) Irrespective of s and t forming a separation pair, the edges in the graph that are adjacent to s and t (other than the edge (s, t)) must be embedded into the components at $G^{(\mathcal{F})}$ to verify that these edges do not violate planarity conditions. It is interesting to note that no such check is performed in Klein and Reif's algorithm.

c) The paper [16] cited does not contain the stated condition directly. Further, the result has no bearing on the algorithm at its ending stage since the embedded components adjacent to s and/or t need not be triconnected. As an example, consider a graph (refer Fig. 14) with a number of parallel chains between s and t. At the final stage, s and t would form a separation pair but the resulting components of $G^{(\mathcal{F})}$ would not even be biconnected, let alone triconnected.

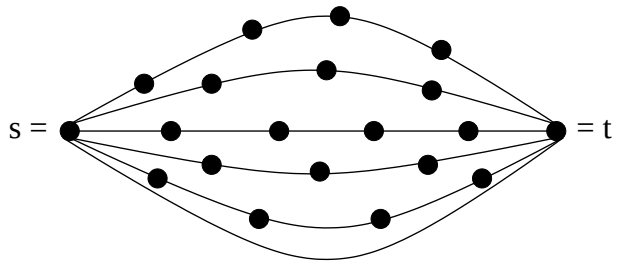

Figure 14: A graph with a number of parallel chains between s and t

2. PQ-Trees represent permitted permutations of a set (and hence linear orders) naturally. The join operation must handle cycles since in the parallel algorithm planar components may be rotated to join with other components. PQ-Trees can represent cycles of the linear orders. Klein and Reif realized that operations on PQ-Trees do not account for the representation of cycles of the linear orders. Their conclusion that Lemmas 2.2 and 2.3 (Pg. 197 of [14]) can be used in conjunction to reduce a PQ-Tree used to represent a set of cycles is not correct since the Lemmas 2.2 and 2.3 do not clearly state how the *reduce* operation on PQ-Tree can be modified to handle cycles. Hence, operations on the PQ-Trees are not generalized to handle cycles of the linear orders.

3. Klein and Reif define a *ROTATE(A,B,C)* operation (Lemma 2.2, Pg. 197 of [14]). However, they do not explicitly state the situations in which this operation is used in the planarity algorithm. Their statement (quoted from Pg. 198 of [14]): "Sometimes (not always), we can apply Lemma 2.2 to modify T so that all the A_i's are contained in a set A that is contiguous in T" implies that their algorithm does not always work.

4. If the join operation deals with cycles, operations like *multiple-disjoint-reduction* and *intersection* used in the *join* operation must also deal with cycles. It is interesting to note that these operations deal with only linear orders in the paper.

4 An Improved Parallel Algorithm for Planarity Testing

Our planarity algorithm is based on the work of Klein and Reif. We provide corrections to the errors listed in the previous section which results in a correct parallel algorithm for planarity testing. Our algorithm differs from Klein and Reif in the following ways:

1. We only conclude that a biconnected component is planar after verifying that all the edges in the biconnected component can be merged into a single PQ-Tree that represents the entire biconnected component. This ensures that the edges adjacent to s and t do not violate planarity conditions. This is achieved by verifying that it is possible to segregate the edges in $G^{(\mathcal{F})}$ adjacent to either s or t. If both these segregations fail, then the resultant graph is non-planar.
2. The operations such as *multiple-disjoint-reduction* are modified to handle cycles. This is achieved easily by trying first to reduce the PQ-Tree T_i with the set E_i, and if this fails, reducing instead with set D_i. This follows from the property of cyclic orders in that a PQ-Tree reduced with one set D_i immediately implies that it is reduced with respect to both the sets E_i and D_i. A reduction fails only if both these reductions (with respect to E_i and D_i) fail.
3. Cyclic orders (and not the frontiers as stated in the paper) are examined to ensure that the reverse condition mandated by the join operation (achieved by steps $J2$ and $J3$ in Pg. 222 of [14]) is satisfied.

To summarize, our algorithm differs from Klein and Reif's by changing the ending condition of the algorithm and modifying the join operation (looking at cyclic orders rather than frontiers of PQ-Tree to ensure the reverse condition) and the *multiple-disjoint-reduction* operation to handle cycles (to reduce with both sets E_i and D_i if the reduction fails for set E_i). These modifications enable our approach to give correct answers on instances that provoke false positive and false negative solutions using Klein and Reif's algorithm. It is trivial to note that these improvements do not increase the time/work complexity of the algorithm. Hence our improved planarity algorithm also has the same complexities as that of the Klein and Reif's algorithm. Therefore our improved algorithm runs in $\mathrm{O}\left(\frac{n \log^2 n}{p}\right)$ time for any number $p \leq n$ processors of a CREW PRAM.

The coarse-grained parallel model (CGM) [7] analysis of our new parallel planarity testing algorithm is similar to that of the analysis by Caceres *et al.* ([6]) and run in $\mathrm{O}\left(\log^2 p\right)$ communication rounds and linear sequential work per round, assuming that the local memory per processor, $\frac{n}{p}$, is larger than p^ϵ for some fixed $\epsilon > 0$. Further, the CGM result implies a Bulk-Synchronous Parallel (BSP) [19] algorithm with $\mathrm{O}\left(\log^2 p\right)$ supersteps, $\mathrm{O}\left(g\frac{n}{p}\log^2 p\right)$ communication, and $\mathrm{O}\left(\frac{n}{p}\log^2 p\right)$ local computation.

We have efficient shared memory implementations for most of the major steps involved in the algorithm such as finding the biconnected components of a graph, spanning tree, and determining the *st-numbering* (e.g., see [4, 3, 2]). In addition

several automatic graph layout packages contain implementations of PQ-trees, such as AGD [1] that interfaces with LEDA [17]. As such, our new planarity algorithm should be suitable for implementation on symmetric multiprocessors.

5 Conclusions and Future Work

This paper presents a new parallel algorithm for planarity testing based upon the work of Klein and Reif. The new algorithm has the same complexity bounds as Klein and Reif's algorithm and runs in $O\left(\frac{n \log^2 n}{p}\right)$ time for any number $p \leq n$ processors of a CREW PRAM. Further, the algorithm is also suitable for the more realistic CGM and BSP models that adhere to the assumptions of commercially-available multiprocessors. Our new approach gives correct answers on instances that provoke false positive and false negative solutions using Klein and Reif's algorithm and lends itself to an efficient implementation on symmetric multiprocessors.

References

[1] D. Alberts, C. Gutwenger, P. Mutzel, and S. Näher. AGD–Library: A library of algorithms for graph drawing. In G.F. Italiano and S. Orlando, editors, *Proc. 1st Workshop on Algorithm Engineering (WAE'97)*, pages 112–123, Venice, Italy, September 1997.

[2] D. A. Bader and G. Cong. Fast shared-memory algorithms for computing the minimum spanning forest of sparse graphs. Technical Report 03-002, Electrical and Computer Engineering Department, University of New Mexico, July 2003.

[3] D. A. Bader and G. Cong. A new, fast parallel spanning tree algorithm for symmetric multiprocessors (SMPs). Technical Report 03-001, Electrical and Computer Engineering Department, University of New Mexico, March 2003.

[4] D.A. Bader, A.K. Illendula, B. M.E. Moret, and N. Weisse-Bernstein. Using PRAM algorithms on a uniform-memory-access shared-memory architecture. In G.S. Brodal, D. Frigioni, and A. Marchetti-Spaccamela, editors, *Proc. 5th Int'l Workshop on Algorithm Engineering (WAE 2001)*, volume 2141 of *Lecture Notes in Computer Science*, pages 129–144, Århus, Denmark, 2001. Springer-Verlag.

[5] K.S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Computer and System Sciences*, 13(3):335–379, 1976.

[6] E. Caceres, A. Chan, F. Dehne, and S. W. Song. Coarse grained parallel graph planarity testing. In *Proc. Int'l Conf, on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, June 2000.

[7] F. Dehne (Ed.). Course grained parallel algorithms. *Algorithmica*, 24(3–4):173–426, 1999.

[8] S. Even and R. E. Tarjan. Computing an st-numbering. *Theoretical Computer Science*, 2:339–344, 1976.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[10] J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[11] J. JáJá and J. Simon. Parallel algorithms in graph theory: Planarity testing. *SIAM J. Computing*, 11(2):314–328, 1982.

[12] P.N. Klein. An efficient parallel algorithm for planarity. Master's thesis, MIT, June 1986.

[13] P.N. Klein and J.H. Reif. An efficient parallel algorithm for planarity. In *Proc. 27th Ann. IEEE Symp. on Foundations of Computer Science*, pages 465–477, Toronto, Canada, October 1986.

[14] P.N. Klein and J.H. Reif. An efficient parallel algorithm for planarity. *J. Computer and System Sciences*, 37(2):190–246, 1988.

[15] A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In P. Rosenstiel, editor, *Proc. Int'l Symp. Theory of Graphs*, pages 215–232, New York, 1967. Gordon and Breach.

[16] S. MacLane. A combinatorial condition for planar graphs. *Fundamenta Mathematicae*, 28:22–32, 1937.

[17] K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[18] V. Ramachandran and J. Reif. Planarity testing in parallel. *J. Computer and System Sciences*, 49(3):517–561, 1994.

[19] L. G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, 1990.