Chapter 1
On Certificates and Lookahead in Dynamic Graph Problems

Sanjeev Khanna* Rajeev Motwani† Randall H. Wilson‡

Abstract

Recent work in dynamic graph algorithms has led to efficient algorithms for dynamic *undirected* graph problems such as connectivity. However, no efficient deterministic algorithms are known for the dynamic versions of fundamental *directed* graph problems like strong connectivity and transitive closure, as well as some undirected graph problems such as maximum matchings and cuts. We provide some explanation for this lack of success by presenting quadratic lower bounds on the *certificate complexity* of the seemingly difficult problems, in contrast to the known *linear* certificate complexity for the problems which have efficient dynamic algorithms. A direct outcome of our lower bounds is the demonstration that a generic technique for designing efficient dynamic graph algorithms, viz., sparsification, will not apply to the difficult problems. More generally, it is our belief that the boundary between tractable and intractable dynamic graph problems can be demarcated in terms of certificate complexity.

In many applications of dynamic (di)graph problems, a certain form of *lookahead* is available. Specifically, we consider the problems of assembly planning in robotics and the maintenance of relations in databases. These give rise to dynamic strong connectivity and dynamic transitive closure problems, respectively. We explain why it is reasonable, and indeed natural and desirable, to assume that lookahead is available in these two applications. Exploiting lookahead to circumvent their inherent complexity, we obtain efficient fully-dynamic algorithms for strong connectivity and transitive closure.

*Department of Computer Science, Stanford University, Stanford, CA 94305. Supported by a Schlumberger Foundation Fellowship, an OTL grant, and NSF Grant CCR-9357849.

†Department of Computer Science, Stanford University, Stanford, CA 94305. Supported by Alfred P. Sloan Research Fellowship, an IBM Faculty Development Award, an OTL grant, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

‡Intelligent Systems and Robotics Center, Sandia National Laboratories, Albuquerque, NM, 87185. Supported by DOE contract DE-AC04-94AL85000. This work was performed while the author was at the Department of Computer Science, Stanford University.

1 Introduction.

A dynamic graph problem is that of maintaining some property of a graph as it undergoes changes such as insertion and deletion of edges. An "efficient" algorithm for such problems will be able to incrementally maintain the graph property at a cost (per update operation) that is significantly smaller than the cost of computing the graph property from scratch.

While much of the prior research in the design of dynamic graph algorithms involved problem-specific approaches [11, 12, 13, 22, 27], recently Eppstein, Galil, Italiano, and Nissenzweig [9] presented a general paradigm called *sparsification* that is useful in both designing new dynamic graph algorithms, as well as speeding up the existing ones. The sparsification technique is based on the notion of a *sparse strong certificate* which, informally speaking, is a sparse graph whose structure is representative of the input graph with respect to the property of interest. Eppstein *et al* established the existence of sparse strong certificates for a variety of *undirected* graph properties[3] such as edge and vertex connectivity, bipartiteness, and minimum spanning tree, thereby obtaining efficient dynamic algorithms; recently, Henzinger and King [16] improved some of these results and obtained randomized fully-dynamic algorithms requiring only *poly-logarithmic* time per operation.

In practice, two particularly important dynamic graph problems are *strong connectivity* (in industrial robotics applications [28]) and *transitive closure* (in database applications [31]). Despite intensive efforts, no efficient deterministic dynamic graph algorithms have been devised for these two problems, as well as a host of others. Recently, Henzinger and King [17] made some progress towards designing randomized dynamic algorithm for a related problem, namely, the directed reachability query problem. Their algorithm answers reachability queries in directed graphs in query time $O(n/\log n)$ and an amortized update time $O(nm^{0.58})$; n and m denote the number of vertices of edges respectively. It appears that known techniques are of little

[3]As we explain in §2, the word *property* is used in a broad sense to include non-boolean functions on graphs.

use for designing deterministic dynamic algorithms for non-trivial *directed* graph problems. We attempt to explain this lack of success by initiating a study of the *strong certificate complexity* of graph properties. For a variety of graph properties, we establish the existence of n-vertex graphs for which every strong certificate must have $\Omega(n^2)$ edges, thereby ruling out the application of sparsification to design efficient deterministic dynamic algorithms for these problems. We also show that similar bounds hold even when a more relaxed notion of certificates is used. Our results demonstrate a qualitative distinction between the certificate complexity of fully-dynamic graph problems with known efficient deterministic algorithms and the (primarily digraph) problems for which no efficient deterministic algorithms are known. This provides a partial explanation for the lack of progress on some dynamic graph problems.

Fortunately, many real systems have some form of *lookahead* available, i.e., the dynamic algorithm is provided with information about future updates. For instance, a frequently arising problem in industrial robotics, namely *assembly planning*, requires an efficient dynamic algorithm for strong connectivity in presence of naturally available lookahead [18, 28, 29, 30]. Also, the problem of dynamic transitive closure arises in numerous database applications (see Yannakakis [31]). This is equivalent to maintaining the transitive closure of a relation undergoing frequent updates, and it is usually desirable to update the transitive closure only after accumulating batches of updates, leading to the availability of lookahead. While the issue of lookahead is receiving increasing attention in *online* algorithms [3, 20, 21], there has not been much work along these lines for dynamic graph algorithms. Edelsbrunner and Overmars [7] studied efficient batch processing of updates for searching problems which have a nice decomposition property, namely, given answers to a query q over sets A and B, one can efficiently compute an answer to q over the set $A \cup B$. Schaeffer and Varman [23] studied parallel batch update of minimum spanning trees, while Eppstein [8] considered the offline version of the same problem. Dai, Imai, Iwano, and Katoh [6] describe an approach for constructing semi-online algorithms, i.e., algorithms based on partial knowledge of future deletes, using partially dynamic algorithms, and apply their techniques to *undirected* graph connectivity problem.

We propose the study of lookahead as a means to circumvent the inherent complexity of certain dynamic graph problems. We present a lookahead-based variant of sparsification and use it to obtain efficient fully-dynamic algorithms for the digraph properties that motivated our investigation, viz., strong connectivity and transitive closure. Our algorithms require only a *weak form of lookahead* where it suffices to know only the vertices involved rather than the exact sequence of edge operations.

In §2, we briefly review sparsification in a slightly more general framework than that of Eppstein *et al* [9]. §3 develops the concepts of *strong certificate complexity* and *witness complexity*. We present a theorem relating these two measures and hence obtain a general technique for bounding the certificate complexity. We present quadratic lower bounds on this complexity for a variety of problems. We also show that these bounds essentially continue to hold under a relaxed notion of certificates. We conclude this section by pointing out some intriguing trends that emerge from our results, and pose a conjecture. In §4, we present a framework for incorporating lookahead into dynamic graph algorithms via a variant of sparsification. §5 is devoted to the presentation of two algorithms for strong connectivity using our new framework. Finally, in §6, we extend our results to the transitive closure problem with lookahead and provide a brief discussion of the relevance of this problem in database applications such as maintenance of materialized views.

2 Preliminaries.

We view a graph G as a set of edges whose cardinality is denoted $|G|$, enabling us to employ set operators to graphs. The underlying vertex set will be a fixed set of n labeled vertices, or will be clear from the context.

The sparsification technique was originally presented in a framework for boolean graph properties, along with applications to non-boolean functions such as minimum spanning tree [9]. We review the sparsification technique using a more general formalism directly applicable to non-boolean graph properties.

Let $\mathcal{P}$ denote an arbitrary function, also referred to as a *graph property*, which maps graphs to *non-empty sets of objects*. For example, the minimum spanning forest function will presumably map a given weighted undirected graph to several possible minimum weight spanning forests. In case of boolean properties, $\mathcal{P}$ may map graphs to symbols TRUE and FALSE. The sparsification technique is based on the notion of certificates.

DEFINITION 2.1. (STRONG CERTIFICATE) *For any graph property $\mathcal{P}$, a* strong $\mathcal{P}$-certificate *of a graph G is a graph G' on the same vertex set as G such that for any graph H, $\mathcal{P}(G' \cup H) \subseteq \mathcal{P}(G \cup H)$.*

DEFINITION 2.2. (SPARSE STRONG CERTIFICATE) *A property $\mathcal{P}$ has* sparse strong certificates *if for every graph G, there exists a strong $\mathcal{P}$-certificate for G with $O(n)$ edges.*

We claim that the next two propositions due to Eppstein *et al* [9] hold under the extended definition too.

PROPOSITION 2.1. (TRANSITIVITY) *If G' is a strong $\mathcal{P}$-certificate for G, and G'' is a strong $\mathcal{P}$-certificate for G', then G'' is a strong $\mathcal{P}$-certificate for G.*

PROPOSITION 2.2. (MONOTONICITY) *If G' is a strong $\mathcal{P}$-certificate for G, and H' is a strong $\mathcal{P}$-certificate for H, then $G' \cup H'$ is a strong $\mathcal{P}$-certificate for $G \cup H$.*

Now suppose $\mathcal{P}$ is a property with sparse strong certificates. Propositions 2.1 and 2.2 together imply that a recursive divide-and-conquer approach can be used for computing the sparse strong certificates. The basic idea in the sparsification technique is to maintain a complete binary tree with $\lceil |G|/n \rceil$ leaves such that each leaf represents n edges of the graph. Each internal node maintains a sparse strong certificate of the graph formed by the union of its two children. Using the transitivity and the monotonicity properties, it follows that the root contains a sparse strong certificate for the whole graph. As the edges are inserted and deleted from the graph, the certificates along the appropriate leaf-root paths are updated. Since each certificate is sparse, an update on the whole graph now reduces to $O(\log(|G|/n))$ updates on sparse graphs. In essence, this technique has cost which is the same as that incurred while processing updates on a sparse graph.

3 Lower Bounds on Strong Certificate Size.

The sparsification technique allows us to devise efficient dynamic algorithms for a host of graph properties including connectivity and minimum spanning trees. However, sparsification has its limitations: many dynamic graph properties such as strong connectivity, maximum matchings, transitive closure and graph search trees, do not yield to sparsification. In this section we explain this phenomenon by establishing the lack of sparse strong certificates for these properties; in particular, we establish that they have instances requiring quadratic-sized certificates, even for a very *relaxed notion of certificates.* We begin with a few definitions.

DEFINITION 3.1. (PROPERTY VALUES) *A graph property $\mathcal{P}$ is said to be* mono-valued *if for any graph G, $\mathcal{P}(G)$ is a singleton; otherwise, it is said to be a* multi-valued property.

For example, all boolean graph properties are mono-valued. However, the class of mono-valued graph properties is a strictly larger class as it also includes properties such as transitive closure.

DEFINITION 3.2. (CRITICAL GRAPH) *A graph G is called* $\mathcal{P}$-critical *if for every proper subgraph G' of G, we have $\mathcal{P}(G') \neq \mathcal{P}(G)$.*

DEFINITION 3.3. (WITNESS COMPLEXITY) *Let $\mathcal{H}$ denote the family of all n-vertex $\mathcal{P}$-critical graphs for a mono-valued property $\mathcal{P}$. Then the* witness complexity *$g(n)$ of $\mathcal{P}$ is defined as $\max_{G \in \mathcal{H}} |G|$.*

DEFINITION 3.4. (DISTINGUISHING GRAPH) *For two graphs G_1 and G_2, a graph H is called a $\mathcal{P}$-*distinguishing graph *for G_1 and G_2 if $\mathcal{P}(G_1 \cup H) \neq \mathcal{P}(G_2 \cup H)$.*

Let $\mathcal{C}(G)$ denote the set of graphs G' which are strong $\mathcal{P}$-certificates for G. A graph $G' \in \mathcal{C}(G)$ if and only if there does not exist a $\mathcal{P}$-distinguishing graph for G and G'.

DEFINITION 3.5. (STRONG CERT. COMPLEXITY) *Let $\mathcal{G}$ denote the family of all n-vertex graphs. Then the* strong certificate complexity *$f(n)$ of a graph property $\mathcal{P}$ is defined as*

$$\max_{G \in \mathcal{G}} \min_{G' \in \mathcal{C}(G)} |G'|.$$

3.1 Mono-Valued Properties and Witness Complexity.

We show that the witness complexity of a mono-valued property $\mathcal{P}$ can be used to derive lower bounds on its strong certificate complexity.

THEOREM 3.1. *Let $\mathcal{P}$ be a mono-valued graph property with witness complexity $\Omega(g(n))$. Then the strong certificate complexity of $\mathcal{P}$ is $\Omega(g(n)/\log n)$.*

Proof. Let G be a largest $\mathcal{P}$-critical graph and let $\mathcal{F}$ denote the family of all labeled subgraphs of G. We claim that for any distinct $G_1, G_2 \in \mathcal{F}$, $\mathcal{C}(G_1) \cap \mathcal{C}(G_2) = \emptyset$. Suppose not; let $G' \in \mathcal{C}(G_1) \cap \mathcal{C}(G_2)$. Assume without loss of generality that $|G_1| \geq |G_2|$, and let $H = G \setminus G_1$. Clearly, $G_1 \cup H = G$. On the other hand, by assumption, G_1 must contain at least one edge not in G_2. Therefore, $(G_2 \cup H) \subset G$ and, since G is $\mathcal{P}$-critical, we conclude that $\mathcal{P}(G_1 \cup H) \neq \mathcal{P}(G_2 \cup H)$, thereby contradicting the existence of such a G'. Thus, if $f(n)$ denotes the strong certificate complexity of $\mathcal{P}$, we must have

$$\sum_{i=0}^{f(n)} \binom{n(n-1)}{i} \geq |\mathcal{F}|.$$

Using Stirling's approximation and the fact that $|\mathcal{F}| = 2^{\Omega(g(n))}$, we obtain the desired result. □

The following corollary lists some immediate applications of Theorem 3.1.

COROLLARY 3.1. *The following (di)graph properties have*

$\Omega(n^2/\log n)$ strong certificate complexity: (1) transitive closure; (2) diameter; and, (3) minimum cut value.

Proof. Each of these three properties is mono-valued and we obtain the results by applying Theorem 3.1 to the following critical graphs: (1) a directed bipartite graph with all edges directed from one bipartition to the other; (2) the complete bipartite graph $K_{n,n}$ (diameter two); and, (3) the complete graph K_n. □

We now indicate how all three of the above lower bounds can be improved to $\Omega(n^2)$. For the transitive closure property, the lower bound of $\Omega(n^2)$ follows immediately from the proof of an identical lower bound for strong connectivity as established in the §3.2.

For the diameter two property, we consider the graph G obtained by deleting a perfect matching M from $K_{n,n}$. We claim that given any graph G' such that $G \not\subseteq G'$, there exists a distinguishing graph H for G and G'. If $G' \subset G$, then M is the distinguishing graph. Otherwise, there exists an edge $e = (x, y) \in G' \setminus G$. If $e \in M$ then $K_{n,n} \setminus \{e\}$ is the distinguishing graph; else let x' and y' respectively denote the mates of the vertices x and y in the perfect matching. Observe that x and y could not have been matched together in M. If we define the graph H to be all the edges in $K_{n,n}$ except (x, x') and (y, y'), then clearly, $G' \cup H$ has diameter two. On the other hand, in the graph $G \cup H$, the pairs $\{x, x'\}$ and $\{y, y'\}$ are still at distance three. Hence any strong certificate for G must be in fact, identical to G. This establishes the lower bound of $\Omega(n^2)$.

Finally, for the property of the minimum cut value, we consider the graph $G = K_n \setminus \{e\}$ for some edge e in the graph. If G' is a strong certificate for G, then G' must have at least $n(n-2)+1$ edges; otherwise the minimum cut value of G' is strictly less than that of G (i.e. G and G' are distinguished by the empty graph).

3.2 Multi-Valued Properties and Properties with Sparse Witnesses.

Theorem 3.1 has its limitations in yielding good lower bounds. Several graph properties have linear witness complexity and yet no sparse strong certificates. The properties of strong connectivity and perfect matching are two such examples. In this section, we study several such properties and establish an $\Omega(n^2)$ lower bound on the strong certificate complexity of these properties.

3.2.1 Strong Connectivity.

We show that the strong certificate complexity of strong connectivity is $\Omega(n^2)$. We begin with a crucial lemma.

LEMMA 3.1. *Let G_1, G_2 be two labeled directed graphs on the same set of vertices, say V, and let H_1 and H_2 respectively denote their transitive closure graphs. Then if $H_1 \not\subseteq H_2$, there exists a graph H such that $G_1 \cup H$ is strongly connected and $G_2 \cup H$ is not.*

Proof. Consider an edge $e = (x, y)$ such that $e \in H_1$ and $e \notin H_2$. Let H be the set of edges (y, z) and (z, x) where $z \in V \setminus \{x, y\}$. Clearly, $H \cup \{e\}$, and hence $H \cup H_1$, is strongly connected. On the other hand, the edges in H cannot contribute to a path from x to y and hence there is no such path in $H \cup H_2$. The lemma now follows by observing that for any two graphs G and H, we have $\mathrm{TC}(G \cup H) = \mathrm{TC}(\mathrm{TC}(G) \cup H)$ where TC(.) denotes the transitive closure of a given graph. □

Consider now the directed bipartite graph $G = (X \cup Y, E)$ where $|X| = |Y| = n$ and E contains an edge (x, y) for all $x \in X$ and $y \in Y$. It is easily seen that $\mathrm{TC}(G) = G$. We claim that for any graph H such that $G \not\subseteq H$, $\mathrm{TC}(G) \neq \mathrm{TC}(H)$. To see this, consider such a graph H and an edge $(x, y) \in G \setminus H$. Suppose there is a path from x to y in H. Such a path in H must contain one of the following: an edge between two vertices in X, an edge between two vertices in Y, or an edge from Y to X. But none of these edges are present in $G = \mathrm{TC}(G)$. Therefore, $\mathrm{TC}(G) \neq \mathrm{TC}(H)$. On the other hand, if there is no path from x to y in H, then $\mathrm{TC}(G) \neq \mathrm{TC}(H)$. Thus, by Lemma 3.1, there exists a graph which distinguishes G and H for the property of strong connectivity. We conclude that any certificate of the graph G must contain every edge in the graph G. This establishes the following theorem.

THEOREM 3.2. *The strong certificate complexity of strong connectivity is $\Omega(n^2)$.*

3.2.2 Maximum Matching.

We now turn to the problem of maintaining a maximum matching in an undirected graph. Consider the bipartite graph $G = (X \cup Y, E)$ where $|X| = 2n$, $Y = Y_1 \cup Y_2$, $|Y_1| = |Y_2| = n$, and $E = \{(x, y) | x \in X,\ y \in Y_1\}$. Observe that it is useless for a certificate G' of G to have any edge $e \notin E$ because such an edge does not belong to any maximum matching of G. Thus $G' \subseteq G$. We claim that G' must have at least $n^2 + 1$ edges of G. Suppose not; then, there exists a vertex $y \in Y_1$ such that y is connected to no more than $k \leq n$ vertices in X in the graph G'. Let X' denote the set of neighbors of y. Consider the graph $H = (X \cup Y, E_H)$, where E_H contains all edges (x, y) such that $x \in X'$ and $y \in Y_2$. Clearly, the graph $G \cup H$ has a maximum matching of size $n + k$ while no matching of $G' \cup H$ can have size more than $n + k - 1$. Thus we obtain the following theorem.

THEOREM 3.3. *The strong certificate complexity maximum matching is $\Omega(n^2)$.*

This theorem can be easily extended to the boolean property of having a perfect matching in the graph. The basic idea is to consider again a specific bipartite graph G and show that if the certificate contains an edge not in H, then using the special structure of G, we can always create a distinguishing subgraph for this property. The quadratic lower bound for the strong certificate complexity of matching was independently pointed out by Eppstein *et al* [10].

3.2.3 Breadth-first Search.

Next, we consider the strong certificate complexity of maintaining a breadth-first search tree in a graph, from a specific vertex, say v_1. Consider first the directed case; let G be a graph on the vertices $v_1, v_2, \ldots, v_n$ where $n = 3k+1$ for some integer k, whose edges consist of the following four sets:

- the set E_1 containing all edges from v_1 to v_i, for $2 \le i \le k+1$;
- the set E_2 containing all edges from v_{i+1} to v_{k+i+1}, for $1 \le i \le k$;
- the set E_3 containing all edges from v_{k+i+1} to v_{2k+i+1}, for $1 \le i \le k$;
- the set E_4 containing all edges from v_i to v_j, for $2k+2 \le i < j \le 3k+1$.

Any breadth-first search tree from the vertex v_1 in G will contain precisely the edges in E_1, E_2, and E_3, with the vertices in $V_1 = \{v_2, v_3, \ldots, v_{k+1}\}$ at the first level of the tree, the vertices in $V_2 = \{v_{k+2}, v_{k+3}, \ldots, v_{2k+1}\}$ at the second level of the tree, and the remaining vertices $V_3 = \{v_{2k+2}, v_{2k+3}, \ldots, v_n\}$ at the third level of the tree.

Consider any graph G' that is a strong certificate of G for this property. We claim that G' must include all edges in E_4. To see this, suppose that G' does not contain some edge $e = (v_i, v_j) \in E_4$. Then consider the graph $G' + e'$, where $e' = (v_1, v_i)$. The distance from vertex v_1 to v_j is three in the graph $G' + e'$, since G' does not have any edges from vertices in $\{v_1\} \cup V_1$ to v_j. On the other hand, the distance from v_1 to v_j is exactly two in the graph $G + e'$. Clearly then, the edge e' is a distinguishing graph for G and G'. It follows that any strong certificate must have all the edges in E_4, and therefore $|G'| \ge |E_4| = \Omega(n^2)$.

It is easy to verify that the above argument also applies to undirected graphs: simply consider the underlying undirected edge for each edge used in the above construction. We obtain the following theorem.

THEOREM 3.4. *The strong certificate complexity of the breadth-first search tree property from a given vertex in a directed or undirected graph is $\Omega(n^2)$.*

3.2.4 Lexicographic-first Depth-first Search.

Finally, we consider the property of maintaining a lexicographic-first depth-first search forest of a graph. Let $\mathcal{D}(G)$ denote the lexicographic-first depth-first search forest of a graph G.

Consider the directed graph G on $n = 2k$ vertices partitioned into the vertex sets $X = \{v_1, v_3, \ldots, v_{2k-1}\}$ and $Y = \{v_2, v_4, \ldots, v_{2k}\}$. The edge set of G comprises of all edges (v_i, v_j) such that either both i and j are odd or both are even. Thus there are no edges between the vertices in X and Y.

Let G' be a strong certificate for G for this property and let E' be the set of edges (v_j, v_i) where $1 < i < j$, and both i and j are odd. Suppose one of the edges $e = (v_j, v_i) \in E'$ is missing in G'. Then we claim that the graph H containing the edges (v_{i-2}, v_2) and (v_{j-1}, v_j) is a distinguishing graph for G and G'. To see this, observe that the edge e belongs to $\mathcal{D}(G+H)$. But, on the other hand, $e \notin G'$ and therefore $e \notin \mathcal{D}(G'+H)$. Thus, a strong certificate for G must contain all edges in E'. Since $|E'| = \Omega(n^2)$, any strong certificate for G must contain $\Omega(n^2)$ edges.

As before, one can easily verify that the above argument also applies to the underlying undirected graphs. We obtain the following theorem.

THEOREM 3.5. *The strong certificate complexity of a lexicographic-first depth-first search forest property in a directed or undirected graph is $\Omega(n^2)$.*

3.3 An Extended Notion of Certificates.

So far we assumed that a certificate has the same vertex set as its underlying graph. However, conceivably, by using a larger vertex set, one may reduce the overall certificate complexity, measured now in terms of both the number of vertices and edges in the certificate. We refer to this new notion of certificates as *extended strong certificates* and their complexity as *extended strong certificate complexity.* Surprisingly, the quadratic complexity results of §3.1 and §3.2 generalize to this extended notion of certificates.

THEOREM 3.6. *Let $\mathcal{P}$ be a mono-valued graph property with witness complexity $\Omega(g(n))$. Then the extended strong certificate complexity of $\mathcal{P}$ is $\Omega(g(n)/\log n)$.*

Proof. Let $\mathcal{E}_n(.)$ denote a mapping from the set of n-vertex graphs to the set of n^*-vertex graphs such that $\mathcal{E}_n(G)$ is an extended strong certificate for a n-vertex graph G. As in Theorem 3.1, we consider a largest $\mathcal{P}$-critical graph G on n vertices and define $\mathcal{F}$ as the family of all its labeled subgraphs. We claim that for any distinct $G_1, G_2 \in \mathcal{F}$, $\mathcal{E}_n(G_1) \cap \mathcal{E}_n(G_2) = \emptyset$. To see this, assume $|G_1| \ge |G_2|$, and define $H = G \setminus G_1$. By definition, $\mathcal{P}(\mathcal{E}_n(G_1) \cup \mathcal{E}_n(H)) = \mathcal{P}(G_1 \cup H) = P(G)$. On the other hand, $\mathcal{P}(\mathcal{E}_n(G_2) \cup \mathcal{E}_n(H)) = \mathcal{P}(G_2 \cup H) = \mathcal{P}(G') \ne \mathcal{P}(G)$ since $G' \subset G$. Therefore, $\mathcal{E}_n(G_1)$ and $\mathcal{E}_n(G_2)$ must be distinct. Thus, if $f(n)$ denotes the extended strong certificate complexity of $\mathcal{P}$, we must have

$$\sum_{i=0}^{f(n)} \binom{f(n)(f(n)-1)}{i} \geq |\mathcal{F}|.$$

Using Stirling's approximation and the fact that $|\mathcal{F}| = 2^{\Omega(g(n))}$, we obtain the desired result. □

THEOREM 3.7. *Suppose that for a graph property $\mathcal{P}$ there exists an n-vertex graph G with m edges such that $\mathcal{C}(G) = G$. Then the extended strong certificate complexity of $\mathcal{P}$ is $\Omega(m/\log n)$.*

Proof. If $\mathcal{C}(G) = G$, we show that for any two distinct subgraphs of G, say G_1 and G_2, there exists a graph H such that $\mathcal{P}(G_1 \cup H) \neq \mathcal{P}(G_2 \cup H)$. To see this, assume $|G_1| \geq |G_2|$ and define $H_1 = G \setminus G_1$. Clearly, $G_1 \cup H_1 = G$ and $G_2 \cup H_1 = G' \subset G$. But $G' \notin \mathcal{C}(G)$ and, therefore, there exists a graph H_2 such that $\mathcal{P}(G \cup H_2) \neq \mathcal{P}(G' \cup H_2)$. Now the argument used in Theorem 3.6 implies that $\mathcal{E}_n(G_1) \cap \mathcal{E}_n(G_2) = \emptyset$. Since there are 2^m possible subgraphs of G, the result follows from the counting argument used earlier. □

COROLLARY 3.2. *The extended strong certificate complexity is $\Omega(n^2/\log n)$ for each of the following (di)graph properties: (1) transitive closure; (2) diameter; (3) minimum cut; (4) strong connectivity; (5) maximum matching; (6) breadth-first search tree; and, (7) lexicographic-first depth-first search.*

The lower bound for the first three properties are an immediate corollary of Theorem 3.6 while those for the last four can be obtained from Theorem 3.7.

3.4 Canonical Hard Instances and A Conjecture.

We conclude our discussion of certificate complexity by commenting on some interesting trends that emerge from our work.

It is rather surprising that the "hard" instances for most of the properties were simply bipartite graphs. A natural question is: do bipartite graphs provide canonical hard instances for strong certificate complexity? The following theorem asserts that this is almost the case.

THEOREM 3.8. *Let $f(n)$ denote the strong certificate complexity of a property $\mathcal{P}$, and $h(n)$ denote the strong certificate complexity of $\mathcal{P}$ when the input is restricted to bipartite graphs. Then, $h(n) = \Omega(f(n)/\log n)$.*

Proof. It is well-known that every graph contains a bipartite subgraph with at least half the edges (a greedy bipartition of a graph creates yields a cut containing at least half the edges, and this gives the desired bipartite subgraph). Thus, given a graph G, its edges can be partitioned into $k = O(\log n)$ graphs $G_1, G_2, \ldots, G_k$ such that each G_i is bipartite. Let C_i denote an optimum strong certificate for G_i. By Lemmas 2.1 and 2.2, $C = \bigcup_{i=1}^{k} C_i$ is a strong certificate for G. Clearly, the size of C is no more than $O(h(n) \log n)$. □

A rather intriguing trend which seems to emerge is that a somewhat restricted class of graph properties appears to have either linear or quadratic strong certificate complexity, but not in between. We define a *non-degenerate* graph property as a graph property $\mathcal{P}$ such that there exist two graphs $G_1, G_2 \in \mathcal{P}$ such that G_1 has $O(n)$ edges while G_2 has $\Omega(n^2)$ edges. (An example of a degenerate property would be a threshold function on the number of edges with a super-linear threshold.)

CONJECTURE 3.1. *Let $\mathcal{P}$ be a non-degenerate, monotone graph property. Then the strong certificate complexity of $\mathcal{P}$ is either $O(n)$ or $\Omega(n^2)$.*

In light of our results and the above conjecture, a review of known efficient dynamic graph algorithms suggests that a non-degenerate, monotone graph problem has an efficient dynamic algorithm only if it has sub-quadratic certificate complexity.

4 Sparsification with Lookahead.

The strong certificate complexity results of the previous sections provide some explanation for the lack of efficient dynamic graph algorithms for a variety of problems. However, these problems routinely arise in many practical applications and there is a need for efficient algorithms. Fortunately, at least some of these applications have the crucial feature of a *lookahead* in terms of some knowledge about the updates to be performed in the future. This motivates us to study the complexity of maintaining some of these seemingly intractable properties under lookahead. While doing so, we assume only a weak form of lookahead: we know only the set of vertices involved in the updates to be performed in the immediate future. In other words, we are aware of the current hot spots of activity in the underlying graph.

Our results are based on a new notion of sparsification, namely, *lookahead-based sparsification*. The sparsification technique as presented by Eppstein *et al* [9] is essentially an edge-sparsification technique in which the sparse certificates encapsulate the essential structure of the original (dense) graph into a sparse set of edges. The lookahead-based sparsification, on the other hand, is a vertex-sparsification technique where we use the knowledge of the future to construct a representative graph H on a smaller set of vertices such that processing the edge operation sequence on the original graph is "equivalent" to performing the sequence on this smaller (possibly dense) graph H.

We develop the basic technique in the next two sections through applications to strong connectivity and transitive closure.

5 Assembly Sequencing and Lookahead in Strong Connectivity.

Assembly sequencing is a subproblem of assembly planning. Given a set of parts and their final positions in a product, assembly sequencing attempts to find a sequence of object motions that will build the assembly from the parts without inducing part collisions. Assuming that the parts are rigid and each operation mates two rigid subassemblies to produce a larger one, an assembly sequence can be found by disassembling the product and then reversing the operations and their order. The initial assembly is divided into two subassemblies, each of which is disassembled recursively to find the disassembly sequence.

For our purposes, the assembly sequencing problem can be abstracted as follows: given a graph G and a sequence σ of t edge insertions and deletions, determine for each $i \in [1 \ldots t]$ whether the graph obtained after the ith edge operation is strongly connected. The order of t could lie anywhere between n^2 and n^5. Let m denote an upper bound on the maximum number of edges the graph may have at any time. In practice, the graphs arising in this application have $\Omega(n^2)$ edges. Thus the straightforward algorithm for verifying the strong connectivity from scratch after each edge operation leads to an $\Omega(tn^2)$ solution to this problem, which could be as large as n^7 and hence impractical even for the most trivial applications.

We can improve the running time by using a lookahead of k into the sequence [4] of edge operations. The idea is to decompose the processing into t/k phases with each phase involving precisely k edge operations. Let G_p denote the graph at the beginning of phase p and, for $1 \leq i \leq k$, let $G_{p,i}$ denote the graph obtained after the ith edge operation within the phase p. The total number of vertices involved in the k edge insertion and deletions within a phase cannot exceed $2k$; call these vertices the *active* vertices during the phase. We will need the following notation:

- A denotes the set of active vertices;
- E^A denotes the set of all possible directed edges between the active vertices;
- E^A_p denotes the set of the edges actually present between the active vertices in the graph G_p;
- $E^A_{p,i}$ denotes the set E^A_p after the ith edge operation in the phase p;
- G^-_p denotes the graph $G_p \setminus E^A$ and let H_p denote the induced subgraph on the set of active vertices in the transitive closure of the graph G^-_p (i.e., in the graph H_p, there is an edge from u to v if and only if G_p has a path from u to v in which u and v are the only active vertices);
- $H_{p,i}$ is the graph on the active vertices with precisely the edges in H_p and $E^A_{p,i}$;
- finally, G^+_p is the graph $G_p \cup E^A$.

LEMMA 5.1. *If $G_{p,i}$ is strongly connected, then the graph G^+_p is also strongly connected.*

Proof. Simply observe that $G_{p,i} \subseteq G^+_p$. □

LEMMA 5.2. *For any two active vertices, say u and v, there exists a directed path from u to v in $G_{p,i}$ if and only if there exists a directed path from u to v in $H_{p,i}$.*

Proof. Since $H_{p,i}$ is a subgraph of the transitive closure graph of $G_{p,i}$, a directed path from u to v in $H_{p,i}$ always implies a directed path from u to v in $G_{p,i}$. To see the other direction, consider a path from an active vertex u to an active vertex v in $G_{p,i}$. Such a path consists of alternating segments of active and non-active vertices. However, by our construction of the graph $H_{p,i}$, corresponding to each such path Γ, there exists a path Γ' in $H_{p,i}$ such that Γ' is simply Γ with all segments of non-active vertices removed. Thus there exists a directed path from u to v in $G_{p,i}$ if and only if there exists a directed path from u to v in $H_{p,i}$. □

THEOREM 5.1. *The graph $G_{p,i}$ is strongly connected iff both G^+_p and $H_{p,i}$ are strongly connected.*

Proof. If $G_{p,i}$ is strongly connected then the graph G^+_p must also be strongly connected by Lemma 5.1. By Lemma 5.2, the strong connectivity of $G_{p,i}$ implies that $H_{p,i}$ is strongly connected. For the converse, observe that the strong connectivity of $H_{p,i}$ implies that $\mathrm{TC}(G_{p,i})$ includes all edges in the set E^A where $\mathrm{TC}(.)$ denote the transitive closure of a given graph. Therefore,

$$\mathrm{TC}(G_{p,i}) \supseteq \mathrm{TC}(G^-_p \cup E^A) = \mathrm{TC}(G^+_p).$$

Thus $G_{p,i}$ must be strongly connected. □

In what follows, $\mathrm{T}(n, m)$ denotes the time complexity of computing transitive closure on a n-vertex graph with m edges, $\mathrm{T}(n)$ denotes the time complexity of computing transitive closure over n-vertex graphs with no

[4] Actually, the problem involves an implicitly-defined offline sequence but the huge length of the sequence makes it desirable to generate it in smaller blocks of length k so as to reduce the *space* (and time) requirement.

constraint on the number of edges, and $\mathrm{M}(n)$ denote the time complexity of boolean matrix multiplication.

The running time is determined by how efficiently strong connectivity of the graphs G_p^+ and $H_{p,i}$ can be verified. The strong connectivity of the graph G_p^+ can be verified at the beginning of the phase in $O(m+n)$ time by simply collapsing all active vertices into a single vertex and then running the standard strong connectivity algorithm. Then the graph H_p can be constructed in $O(\mathrm{T}(n,m))$ time. Finally, the strong connectivity of each $H_{p,i}$ can be verified in $O(k^2)$ time. Hence the running time for each phase is given by $O(m + \mathrm{T}(n,m) + k^3)$ and the total running time is given by $O(t(k^2 + \mathrm{T}(n,m)/k))$. This expression is minimized when a lookahead of size $\Theta(\mathrm{T}(n,m)^{1/3})$ is used. The running time for the whole sequence becomes $O(t\mathrm{T}(n,m)^{2/3})$ which is an improvement over the naive approach for any $m = \Omega(\mathrm{T}(n,m)^{2/3})$.

Observe that if the graph contains $o(\mathrm{T}(n,m)^{2/3})$ edges, our algorithm is slower than the naive approach. This can be handled easily by adding a simple feature to the lookahead algorithm. At the beginning of each phase, the algorithm computes the number of edges in the current graph. If the graph contains $\Omega(\mathrm{T}(n,m)^{2/3})$ edges, it uses this algorithm, else it simply uses the naive approach of running the strong connectivity algorithm for each operation in the phase.

THEOREM 5.2. *There is an algorithm for fully-dynamic maintenance of the strong connectivity of a graph at an amortized cost of $O(\min\{m, k^2 + \mathrm{T}(n,m)/k\})$ per operation using a lookahead of size $k = O(\mathrm{T}(n,m)^{1/3})$.*

We now observe that the problem of determining strong connectivity of each $H_{p,i}$ within a phase p is identical to our original problem. This leads to a recursive approach with a significantly improved running time.

THEOREM 5.3. *There is an algorithm for fully-dynamic maintenance of the strong connectivity of a graph at an amortized cost of $O(\mathrm{T}(n)/n)$ per operation using a lookahead of $\Theta(n)$.*

Proof. This recursive algorithm and its analysis is as follows. From the preceding analysis, we know that using a lookahead of size k in each phase, we obtain the following recurrence:

$$\mathrm{R}(n,t) \leq \lceil t/k \rceil \times [c_1 m + c_2 \mathrm{T}(n) + \mathrm{R}(2k,k)]$$

where $\mathrm{R}(n,t)$ denotes the time complexity of the recursive algorithm on a n-vertex graph for a sequence of t operations, and c_1 and c_2 are suitably large constants.

We claim that $\mathrm{R}(n,t) \leq \lceil t/n \rceil \times \alpha\mathrm{T}(n)$ for some sufficiently large constant α. The proof is by a simple induction on n. Clearly, the base case with $n = 1$ is satisfied. To see the inductive step, observe that for $k = n/8$,

$$\begin{aligned} \mathrm{R}(n,t) &\leq \lceil 8t/n \rceil [c_1 m + c_2 \mathrm{T}(n) + \mathrm{R}(n/4, n/8)] \\ &\leq 8\lceil t/n \rceil [(c_1 + c_2)\mathrm{T}(n) + \alpha\mathrm{T}(n/4)] \\ &\leq \lceil t/n \rceil [8(c_1 + c_2)\mathrm{T}(n) + 8\alpha\mathrm{T}(n/4)] \\ &\leq \lceil t/n \rceil [8(c_1 + c_2) + \alpha/2]\mathrm{T}(n) \\ &\leq \lceil t/n \rceil \times \alpha\mathrm{T}(n), \end{aligned}$$

provided $\alpha \geq 16(c_1 + c_2)$. Note that we are using the fact that $\mathrm{T}(n/4) \leq \mathrm{T}(n)/16$, which follows from the observation that $\mathrm{T}(n) = \Omega(n^2)$. □

Transitive closure of a graph can be computed using boolean matrix multiplication [2]. Let M denote the complexity of a boolean matrix multiplication algorithm. Then for instance, if we use the best-known (but impractical) algorithm for matrix multiplication due to Coppersmith and Winograd [5] with $\mathrm{M}(n) = n^{2.376}$, then the time per operation of the recursive algorithm is $O(n^{1.376})$. On the other hand, using the more practical algorithm due to Strassen [26] with $\mathrm{M}(n) = n^{2.808}$, we achieve a bound of $O(n^{1.808})$ per operation. These should be contrasted with the naive algorithm requiring $\Omega(n^2)$ time per operation; the graphs arising in the assembly sequencing application have $\Omega(n^2)$ edges at most points in the update sequence.

6 Database Updates and Lookahead in Transitive Closure.

Consider the problem of maintaining the transitive closure of a directed graph undergoing edge updates. The primary motivation for this problem is the maintenance of the transitive closure of relations in databases [31]. In several applications, it is known that the updates will be restricted to a particular portion of the database that will be referred to as the *active set*. The knowledge of an active set provides us with lookahead that can be used to significantly speed up the transitive closure computations. Specifically, we show that it is possible to maintain transitive closure at an amortized cost of $O(k^3 + n^2k/\log^2 n + mn/k)$ per operation, where k is the size of the active set (see Theorem 6.1 and Corollary 6.1). In applications where the amount of lookahead available is unrestricted, such as in the off-line construction of a persistent data structure for transitive closure, we show that it can be maintained at an amortized cost of $O(n\sqrt{mn}/\log n)$ per operation using a lookahead of $\Theta(\sqrt{m/n}\log n)$ (see Corollaries 6.2 and 6.3).

Our techniques also yield a solution to the problem of efficiently updating materialized views in databases [1, 4, 14, 19, 24, 25]. In many query-intensive database applications, materialized views, derived from a base relation at a central site, are maintained at multiple local sites. Abstractly speaking, a materialized view at a local site may simply correspond to the reachability information for a subset of vertices of the original graph. Since the base relation may be very large, there are tremendous costs associated with globally updating the views after each update. To save on such costs, a commonly employed strategy is to trade off currency for efficiency, e.g., by batching the updates and updating the views only after periodic intervals. Another possibility is to always keep the local views current with respect to the updates that affect the vertices in the local view, and to perform global updates only periodically. Our results indicate how the latter may be efficiently implemented by performing only local transitive closure computations after each update (see Theorem 6.2).

We describe the extension of the techniques from §5 to all-pairs transitive closure computation; the same ideas also apply to single-source transitive closure. We follow the notation from §5 and outline the computation done in each phase.

At the beginning of a phase p, we first compute the transitive closure of G_p^- in $O(T(n,m))$ time. For each non-active vertex v, let $\Gamma_p^{N \leadsto A}(v)$ and $\Gamma_p^{N \leadsto N}(v)$ denote, respectively, the set of all active vertices and the set of all non-active vertices that it can reach in the graph G_p^-. Similarly, for each active vertex v, let $\Gamma_p^{A \leadsto N}(v)$ denote the set of all non-active vertices that it can reach in G_p^-.

At the ith step in this phase, we compute the transitive closure of $H_{p,i}$ in $O(T(k,k^2))$ time. For each active vertex v, let $\Gamma_{p,i}^{A \leadsto A}(v)$ denote the set of all active vertices it can reach in the graph $H_{p,i}$. Now, if $\Gamma_{p,i}(v)$ denotes the set of all vertices reachable from a vertex v in $G_{p,i}$, we may compute it as follows. We begin by computing for each active vertex v,

$$\Gamma_{p,i}(v) = \Gamma_{p,i}^{A \leadsto A}(v) \cup \{ \bigcup_{x \in \Gamma_{p,i}^{A \leadsto A}(v)} \Gamma_p^{A \leadsto N}(x) \}.$$

Next, for each non-active vertex v, we compute

$$\Gamma_{p,i}(v) = \Gamma_p^{N \leadsto N}(v) \cup \{ \bigcup_{x \in \Gamma_p^{N \leadsto A}(v)} \Gamma_{p,i}(x) \}.$$

The computation of $\Gamma_{p,i}(v)$ for active vertices can be easily performed in $O(k^2 n)$ time. Similarly, a straightforward implementation has $O(kn^2)$ running time for computing $\Gamma_{p,i}(v)$ for all non-active vertices. This may be slightly improved as follows. Consider a bipartite graph $H^* = (A, V, E^*)$ such that there is an edge (x,y) in E^*, $x \in A$ and $y \in V$, if and only if $y \in \Gamma_{p,i}(x)$. It can be shown that the computation of $\Gamma_{p,i}(v)$ for a non-active vertex v reduces to answering a reachability query in the graph H^*, i.e., given $X = \{x \in \Gamma_p^{N \leadsto A}(v)\}$, find the set of vertices adjacent to X in H^*. Using the approach of Hellerstein, Klein, and Wilber [15], we can construct a data structure to perform this computation in $O(n^2 k/\log^2 n)$ time for all the non-active vertices. Thus the total running time of each phase is given by $O(T(n,m) + kT(k,k^2) + n^2k^2/\log^2 n)$, which is minimized when the lookahead is $k = \sqrt{T(n,m)}\log n/n$. We obtain the following theorem.

THEOREM 6.1. *Given lookahead k, fully-dynamic maintenance of transitive closure can be performed at an amortized cost of $O(T(n,m)/k + T(k,k^2) + n^2k/\log^2 n)$ per update.*

COROLLARY 6.1. *Given lookahead k, fully-dynamic maintenance of transitive closure can be performed at an amortized cost of $O(k^3 + n^2k/\log^2 n + mn/k)$ per update.*

COROLLARY 6.2. *Fully-dynamic maintenance of transitive closure can be performed at an amortized cost of $O((n\sqrt{mn})/\log n)$ per update using lookahead $\Theta(\sqrt{m/n}\log n)$.*

It is worthwhile to contrast these bounds with the bound of $O(nm)$ time per operation achieved by the naive algorithm which recomputes transitive closure at each step.

In the preceding algorithm, we explicitly maintained the transitive closure of the resulting graph after each update. We now describe a variant in which the update time is significantly improved by maintaining only an implicit representation of the transitive closure. This implicit representation can support reachability queries but at a significantly increased cost. The basic idea is the following: after each update during a phase, we compute only the transitive closure of $H_{p,i}$; queries are processed by using this along with the transitive closure of G_p^-. We present only the resulting time bounds.

THEOREM 6.2. *Suppose the updates are restricted to a set of k active vertices. Then, there exists a fully-dynamic data structure for processing reachability queries at an amortized cost of $O(n^3/k + k^3)$ per update, with a query cost that is $O(k)$ for queries involving at least one active vertex, and $O(k^2)$ otherwise.*

The preceding bounds can all be improved using fast matrix multiplication. For example, Corollary 6.2 can be replaced by the following.

COROLLARY 6.3. *Using fast matrix multiplication, fully-dynamic maintenance of transitive closure can be performed at an amortized cost of $O(n^{2.188}/\log n)$ per update using lookahead $\Theta(n^{0.188}\log n)$.*

Acknowledgements

We are grateful to Jean-Claude Latombe for sharing his insights about assembly planning and his constant encouragement. We thank Sergey Brin, Surajit Chaudhry, Waqar Hasan, Jeff Ullman, and Jennifer Widom for useful discussions about transitive closure and databases. Many thanks also to Julien Basch, Chandra Chekuri and Delphine Khanna.

References

[1] R. Agrawal and H.V. Jagdish. Materialization and Incremental Update of Path Information. In *Proceedings of Fifth International IEEE Conference on Data Engineering* (1989), pp. 374–383.

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

[3] S. Albers. The influence of lookahead in competitive on-line algorithms. Technical Report MPI-I-92-143, Max Planck Institute, Germany, 1993.

[4] J.A. Blakaley, P-A. Larson, and F.W. Tompa. Efficiently Updating Materialized Views. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data* (1986), pp. 61–71.

[5] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:1–6 (1990).

[6] Y. Dai, H. Imai, K. Iwano, and N. Katoh. How to treat delete requests in semi-online problems. In *Proceedings of the 4th International Symposium on Algorithms and Computation* (1993), pp. 48-57.

[7] H. Edelsbrunner, and M. H. Overmars. Batched Dynamic Solutions to Decomposable Searching Problems. *Journal of Algorithms*, 6:515-542 (1985).

[8] D. Eppstein. Offline Algorithms for Dynamic Minimum Spanning Tree Problems. *Journal of Algorithms*, 17:237-250 (1994).

[9] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification – A technique for speeding up dynamic graph algorithms. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science* (1992), pp. 60–69.

[10] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification – A technique for speeding up dynamic graph algorithms. *Tech. Rep. RC 19272 (83907)*, IBM (1993).

[11] G.N. Frederickson. Data Structures for On-line Updating of Minimum Spanning Trees, with Applications. *SIAM Journal on Computing*, 14:781–798 (1985).

[12] G.N. Frederickson. Ambivalent Data Structures for Dynamic 2-edge Connectivity and k Smallest Spanning Trees. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science* (1991), pp. 632–641.

[13] Z. Galil and G.F. Italiano. Fully dynamic algorithms for edge-connectivity problems. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing* (1991), pp. 317-327.

[14] E. Hanson. A Performance Analysis of View Materialization Strategies. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data* (1987), pp. 440–453.

[15] L. Hellerstein, P. Klein, and R. Wilber. On the time-space complexity of reachability queries for preprocessed graphs. *Information Processing Letters*, 35:261–267 (1990).

[16] M. Rauch Henzinger and V. King. Randomized Dynamic Algorithms with Polylogarithmic Time per Operation. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing* (1995), pp. 519-527.

[17] M. Rauch Henzinger and V. King. Fully Dynamic Biconnectivity and Transitive Closure. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science* (1995), pp. 664-672.

[18] L.S. Homem de Mello and S. Lee (editors). *Computer-Aided Mechanical Assembly Planning.* Kluwer, 1991.

[19] H.V. Jagdish. A Compression Technique to Materialize Transitive Closure. *ACM Transactions on Database Systems*, 15:558–598 (1990).

[20] E. Koutsoupias and C.H. Papadimitriou. Beyond competitive analysis. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science* (1994), pp. 394–400.

[21] R. Motwani, V. Saraswat, and E. Torng. On-line Scheduling with Lookahead: Multipass Assembly Lines. Report CPS-94-41, Department of Computer Science, Michigan State University, 1994.

[22] M. Rauch. Fully dynamic biconnectivity in graphs. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science* (1992), pp. 50–59.

[23] A. Schaeffer and P. Varman. Parallel Batch Update of Minimum Spanning Trees. Technical Report COMP TR90-140, Rice University, 1990.

[24] A. Segev and J. Park. Maintaining Views in Distributed Databases. In *Proceedings of Fifth International IEEE Conference on Data Engineering* (1989), pp. 262–270.

[25] O. Shmueli and A. Itai. Maintenance of Views. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data* (1984), pp. 240–255.

[26] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 14:354–356 (1969).

[27] J. Westbrook and R.E. Tarjan. Maintaining bridge-connected and biconnected components on-line. *Algorithmica*, 7:433–464 (1992).

[28] R.H. Wilson. *On Geometric Assembly Planning.* PhD thesis, Stanford University, 1992. Stanford Technical Report STAN-CS-92-1416.

[29] R.H. Wilson and J-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence*, 71 (1994).

[30] J.D. Wolter. *On the Automatic Generation of Plans for Mechanical Assembly.* PhD thesis, University of Michigan, 1988.

[31] M. Yannakakis. Graph-theoretic Methods in Database Theory. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (1990), pp. 230-42.